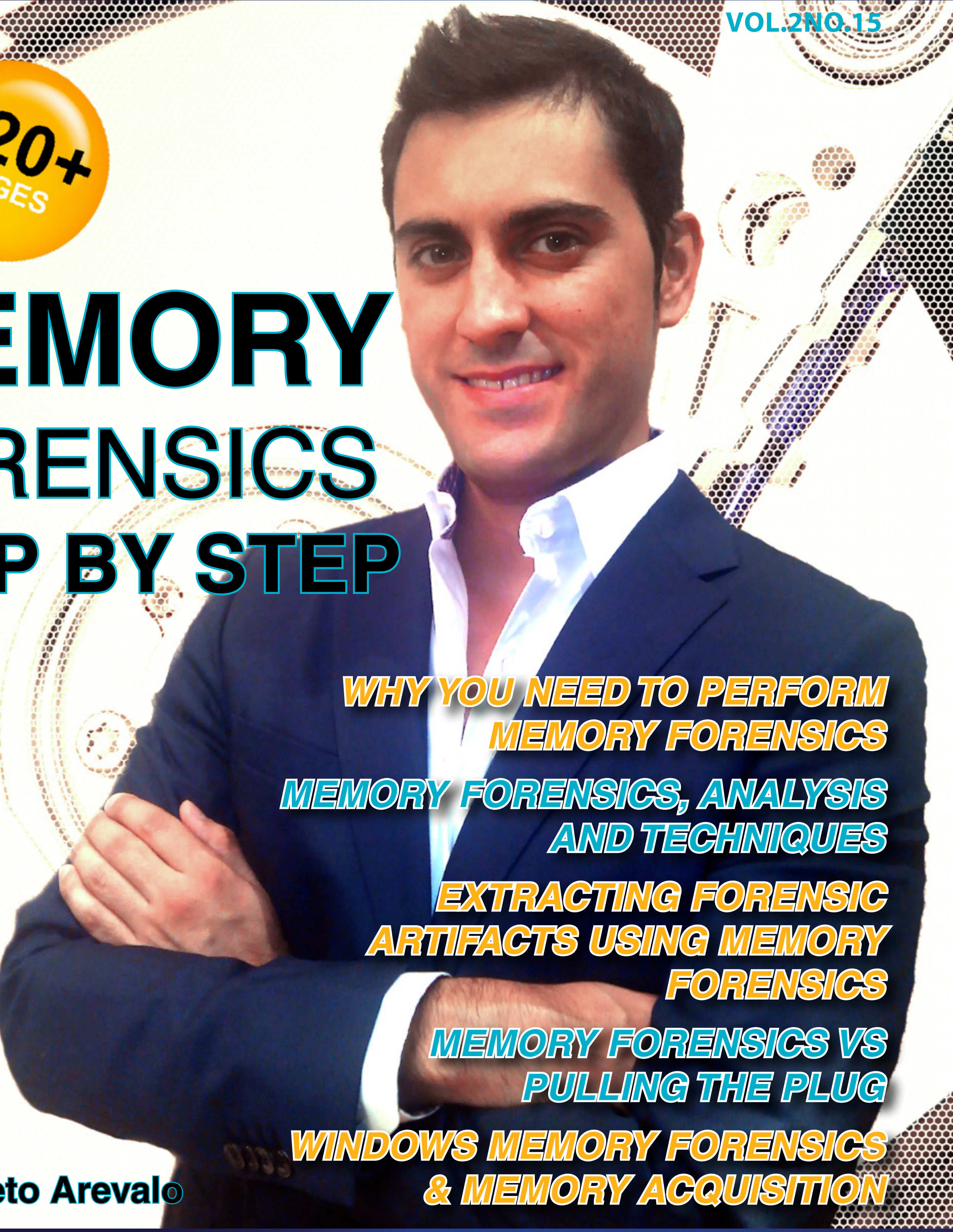
eForensics Magazine

Computer

VOL.2NO.15

120+ PAGES



MEMORY FORENSICS STEP BY STEP

WHY YOU NEED TO PERFORM MEMORY FORENSICS

MEMORY FORENSICS, ANALYSIS AND TECHNIQUES

EXTRACTING FORENSIC ARTIFACTS USING MEMORY FORENSICS

MEMORY FORENSICS VS PULLING THE PLUG

WINDOWS MEMORY FORENSICS & MEMORY ACQUISITION

Javier Nieto Arevalo

Issue 15/2013 (19) October ISSN 2300-6986

@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®

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

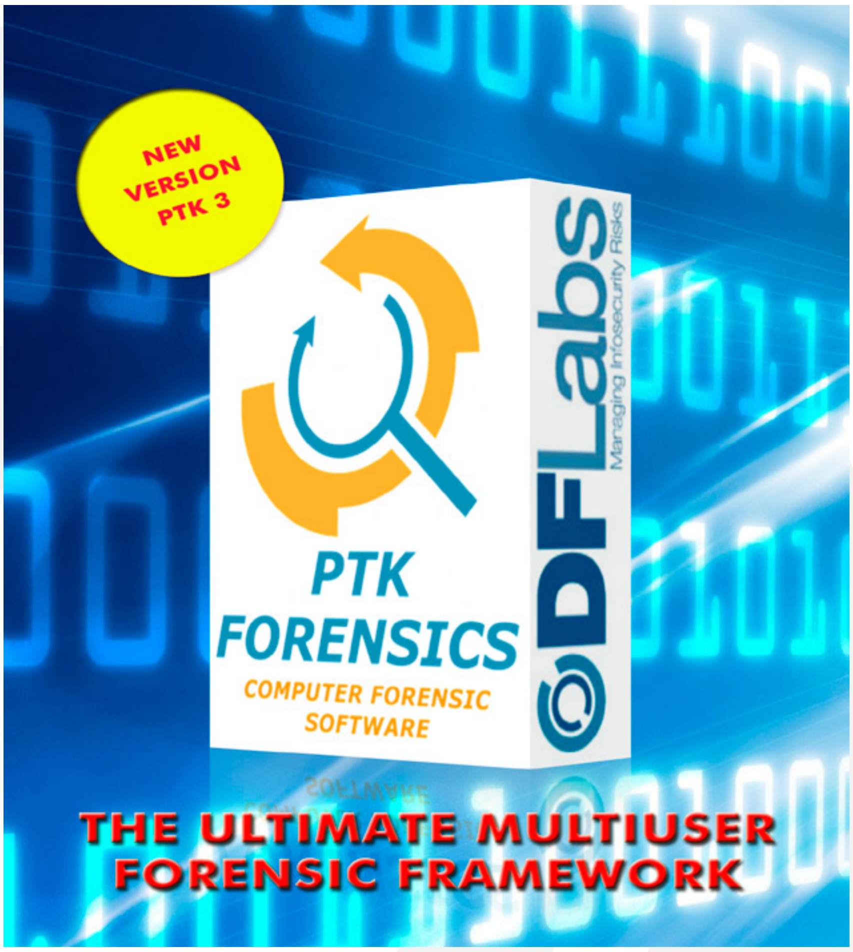
Timeline

Gallery

Keyword Search

Pre-Processing

Advanced Reporting System


NEW VERSION PTK 3
PTK FORENSICS
COMPUTER FORENSIC SOFTWARE
DFLabs
Managing Infosecurity Risks
THE ULTIMATE MULTIUSER FORENSIC FRAMEWORK

SPECIAL PROMO 15% OFF
single user perpetual license

http://www.ptkforensic.com

promo code **E-FORNCS13**

www.dflabs.com - email: info@dflabs.com phone: +1 202 4702688 DFLabs

eForensics Magazine TEAM

Editors:
Dominika Policht
dominika.policht@software.com.pl
Joanna Kretowicz
jaonna.kretowicz@eforensicsmag.com

Betatesters/Proofreaders:
Kishore P.V, Salvatore Fiorillo, Leighton Johnson, Sundar N, Nicolas Villatte, Olivier Caleff, Andrew J Levandoski, Richard Leitz, Jan-Tilo Kirchhoff, Johan Scholtz

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

Dear Readers!

What you have in front of you is a brand new edition of „Memory Forensics". Due to the fact that our last edition covering an issue of Memory Forensics appeared to be a successful one, we have decided to write about it once more – different points of view, different experts and different problems this time. We believe that reading this will be both – interesting and beneficial for all of you. This time we are giving you practical tips, interesting points of view and useful information. From a step by step guidance in Memory Forensics to Integrating Digital Forensics with Archival Science. All this to make you satisfied with what you see. Hopefully you will find this one special and broadening your digital horizons.

We would like to thank you for subscribing to our Magazine, we are doing our best to keep you pleased with our work. You are invited to visiting our website, commenting and sharing your opinion with us.

Only to remind you – you can follow us on Facebook, LinkedIn and Twitter (@eForensics_Mag). Join eForensics friends and fans – we would be more than happy to have you there!

Dominika Policht
and eForensics Team

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

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

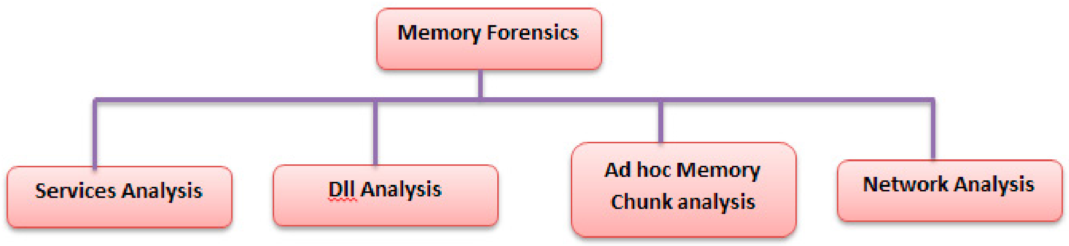


Figure 1. *Memory Forensic*

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.

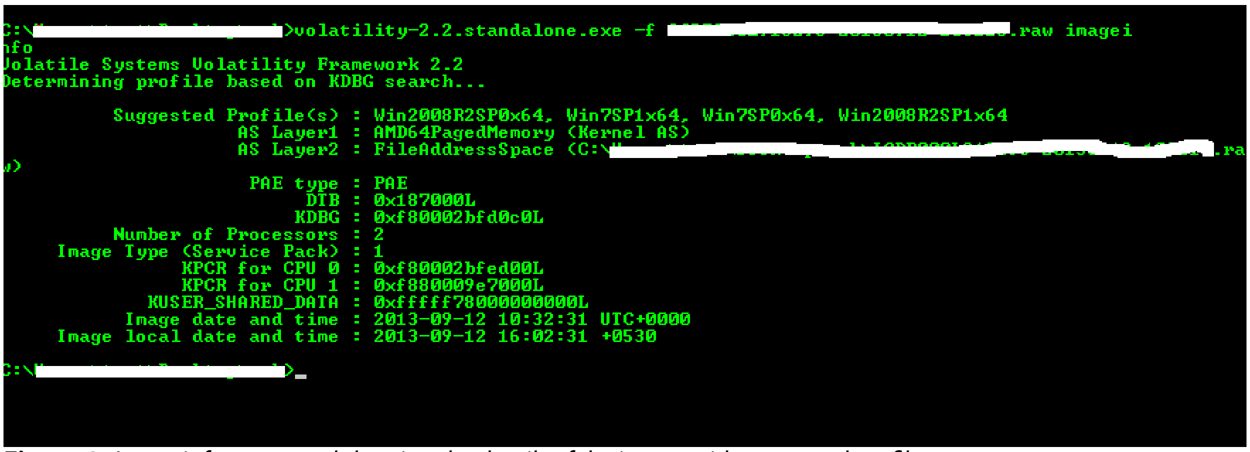
This runs via command line. To access it simply go to the folders location via command prompt and run the exe present. Once up and running the HELP command comes in handy by typing **–h** to know various kinds of options available with volatility.

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..

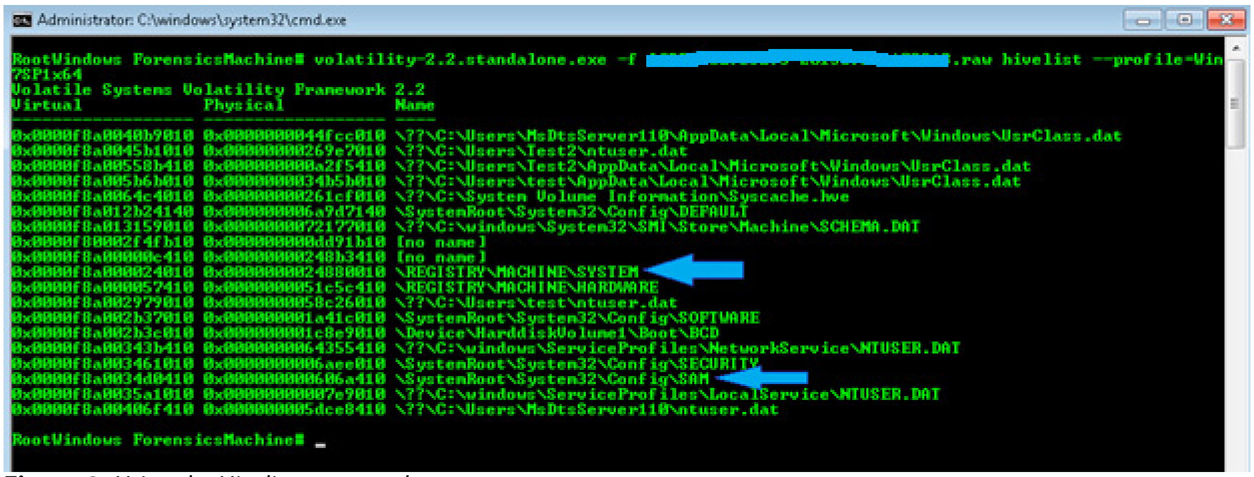
First of all we see on what is the information available with the image captured. For acquiring this information we use the `imageinfo` command.



```
C:\[redacted]>volatility-2.2.standalone.exe -f [redacted].raw imagei
nfo
Volatile Systems Volatility Framework 2.2
Determining profile based on KDBG search...

          Suggested Profile(s) : Win2008R2SP0x64, Win7SP1x64, Win7SP0x64, Win2008R2SP1x64
                     AS Layer1 : AMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\[redacted].ra
w)
                      PAE type : PAE
                           DTB : 0x187000L
                          KDBG : 0xf80002bfd0c0L
          Number of Processors : 2
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xf80002bfed00L
                KPCR for CPU 1 : 0xf880009e7000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2013-09-12 10:32:31 UTC+0000
     Image local date and time : 2013-09-12 16:02:31 +0530

C:\[redacted]>_
```

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.

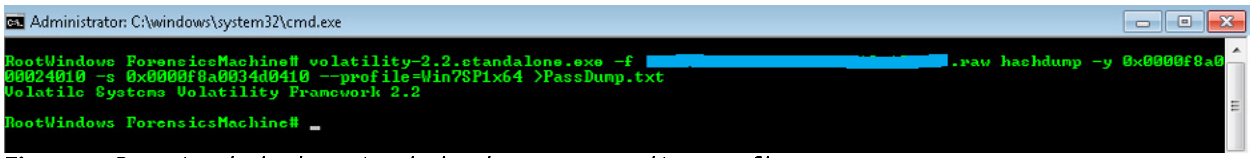


```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f [redacted].raw hivelist --profile=Win
7SP1x64
Volatile Systems Volatility Framework 2.2
Virtual            Physical           Name
------------------ ------------------ ----
0x0000f8a0040b7010 0x0000000044fcc010 \??\C:\Users\MsDtsServer110\AppData\Local\Microsoft\Windows\UsrClass.dat
0x0000f8a0045b1010 0x00000000269e7010 \??\C:\Users\Test2\ntuser.dat
0x0000f8a00558b410 0x000000000a2f5410 \??\C:\Users\Test2\AppData\Local\Microsoft\Windows\UsrClass.dat
0x0000f8a005b6b010 0x0000000034b5b010 \??\C:\Users\test\AppData\Local\Microsoft\Windows\UsrClass.dat
0x0000f8a0064c4010 0x00000000261cf010 \??\C:\System Volume Information\Syscache.hve
0x0000f8a012b24140 0x000000006a9d7140 \SystemRoot\System32\Config\DEFAULT
0x0000f8a013159010 0x0000000072177010 \??\C:\windows\System32\SMI\Store\Machine\SCHEMA.DAT
0x0000f80002f4fb10 0x00000000dd91b10 [no name]
0x0000f8a00000c410 0x00000000248b3410 [no name]
0x0000f8a000024010 0x0000000024880010 \REGISTRY\MACHINE\SYSTEM
0x0000f8a000057410 0x0000000051c5c410 \REGISTRY\MACHINE\HARDWARE
0x0000f8a002979010 0x0000000058c26010 \??\C:\Users\test\ntuser.dat
0x0000f8a002b37010 0x000000001a41c010 \SystemRoot\System32\Config\SOFTWARE
0x0000f8a002b3c010 0x000000001c8e9010 \Device\HarddiskVolume1\Boot\BCD
0x0000f8a00343b410 0x0000000064355410 \??\C:\windows\ServiceProfiles\NetworkService\NTUSER.DAT
0x0000f8a003461010 0x000000006aee010 \SystemRoot\System32\Config\SECURITY
0x0000f8a0034d0410 0x00000000606a410 \SystemRoot\System32\Config\SAM
0x0000f8a0035a1010 0x000000007e9010 \??\C:\windows\ServiceProfiles\LocalService\NTUSER.DAT
0x0000f8a00406f410 0x000000005dce8410 \??\C:\Users\MsDtsServer110\ntuser.dat

RootWindows ForensicsMachine# _
```

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.



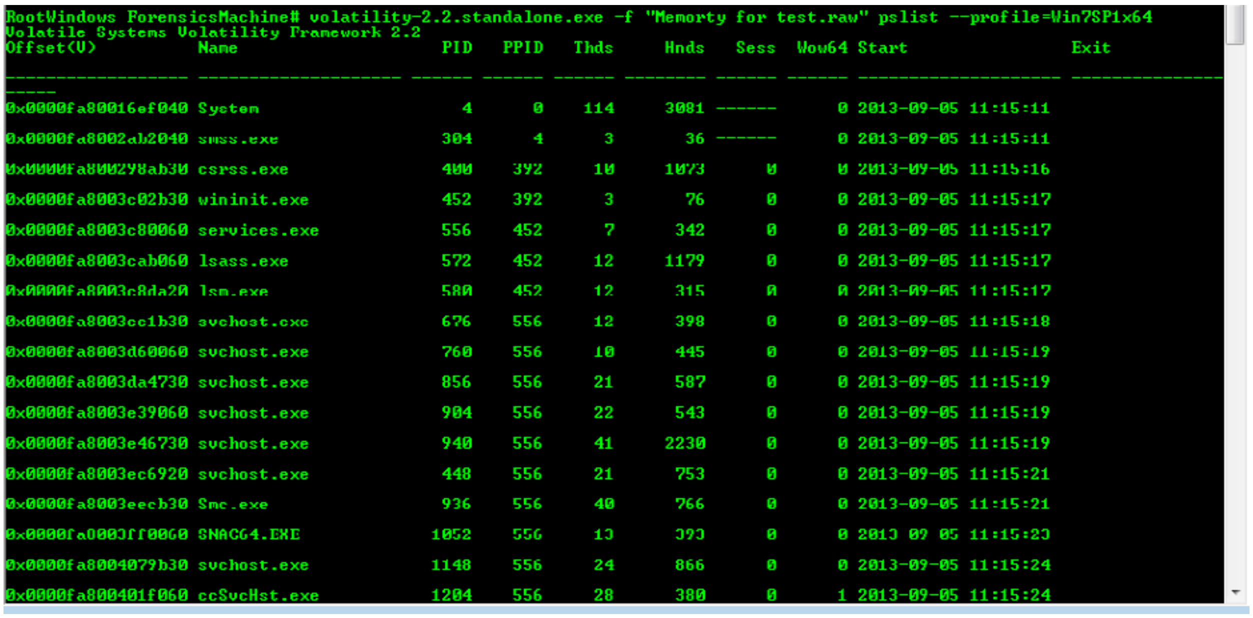
```
Administrator: C:\windows\system32\cmd.exe

RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f [redacted].raw hashdump -y 0x0000f8a0
00024010 -s 0x0000f8a0034d0410 --profile=Win7SP1x64 >PassDump.txt
Volatile Systems Volatility Framework 2.2

RootWindows ForensicsMachine# _
```

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.

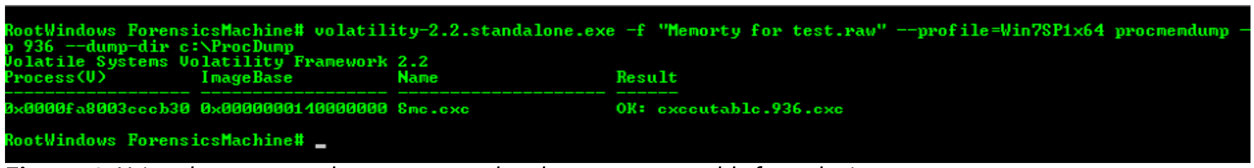


```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f "Memorty for test.raw" pslist --profile=Win7SP1x64
Volatile Systems Volatility Framework 2.2
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                Exit

------------------ -------------------- ------ ------ ------ -------- ------ ------ -------------------- ------------------
-----
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

0x0000fa800298ab30 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

0x0000fa0003ff0060 SNAC64.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
------------------ ------------------ -------------------- ------
0x0000fa8003cccb30 0x0000000140000000 Smc.exe              OK: executable.936.exe

RootWindows ForensicsMachine# _
```

Figure 6. *Using the procmemdump command to dump an executable from the image*

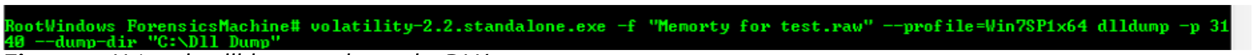
If it was a Windows machine Dlls will be the favorites. `DllList` is another command which can be used to see on how many Dll's were in the memory while the dump of the memory was taken.

```
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*

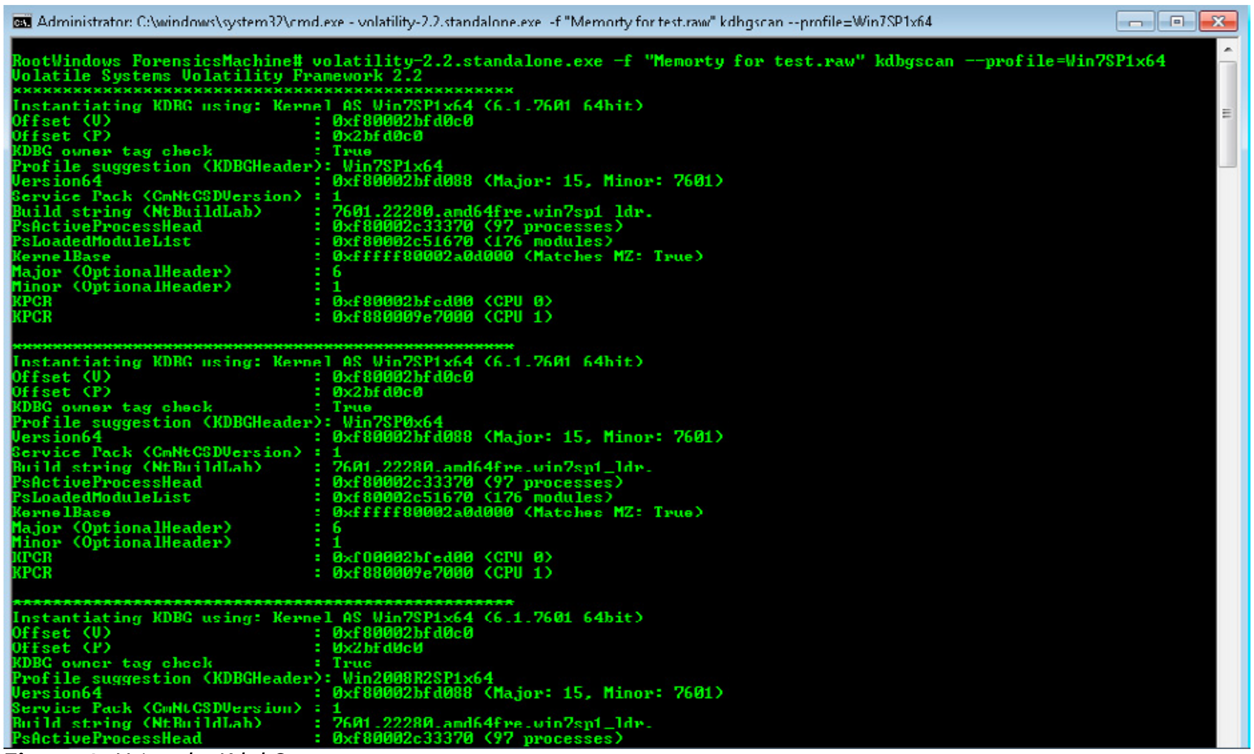
Apart from these let us have a look on lot of different options which can be used to scan the image to get an idea of the state of the system. Below we are going to describe some if the famous ones and by famous one, I mean that these are very often used as a part of your analysis. Though the list is 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.



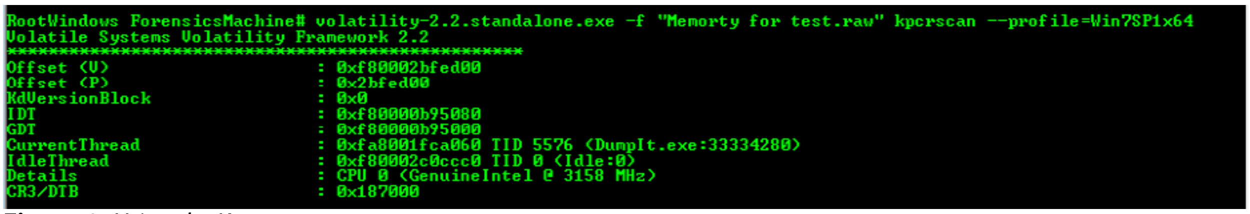
Administrator: C:\windows\system32\cmd.exe - volatility-2.2.standalone.exe -f "Memorty for test.raw" kdbgscan --profile=Win7SP1x64

```
RootWindows ForensicsMachine# volatility-2.2.standalone.exe -f "Memorty for test.raw" kdbgscan --profile=Win7SP1x64
Volatile Systems Volatility Framework 2.2
**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf80002bfd0c0
Offset (P)                    : 0x2bfd0c0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win7SP1x64
Version64                     : 0xf80002bfd088 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.22280.amd64fre.win7sp1 ldr.
PsActiveProcessHead           : 0xf80002c33370 (97 processes)
PsLoadedModuleList            : 0xf80002c51670 (176 modules)
KernelBase                    : 0xfffff80002a0d000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xf80002bfcd00 (CPU 0)
KPCR                          : 0xf880009e7000 (CPU 1)

**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf80002bfd0c0
Offset (P)                    : 0x2bfd0c0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win7SP0x64
Version64                     : 0xf80002bfd088 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.22280.amd64fre.win7sp1_ldr.
PsActiveProcessHead           : 0xf80002c33370 (97 processes)
PsLoadedModuleList            : 0xf80002c51670 (176 modules)
KernelBase                    : 0xfffff80002a0d000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xf80002bfcd00 (CPU 0)
KPCR                          : 0xf880009e7000 (CPU 1)

**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf80002bfd0c0
Offset (P)                    : 0x2bfd0c0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win2008R2SP1x64
Version64                     : 0xf80002bfd088 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.22280.amd64fre.win7sp1_ldr.
PsActiveProcessHead           : 0xf80002c33370 (97 processes)
```

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!

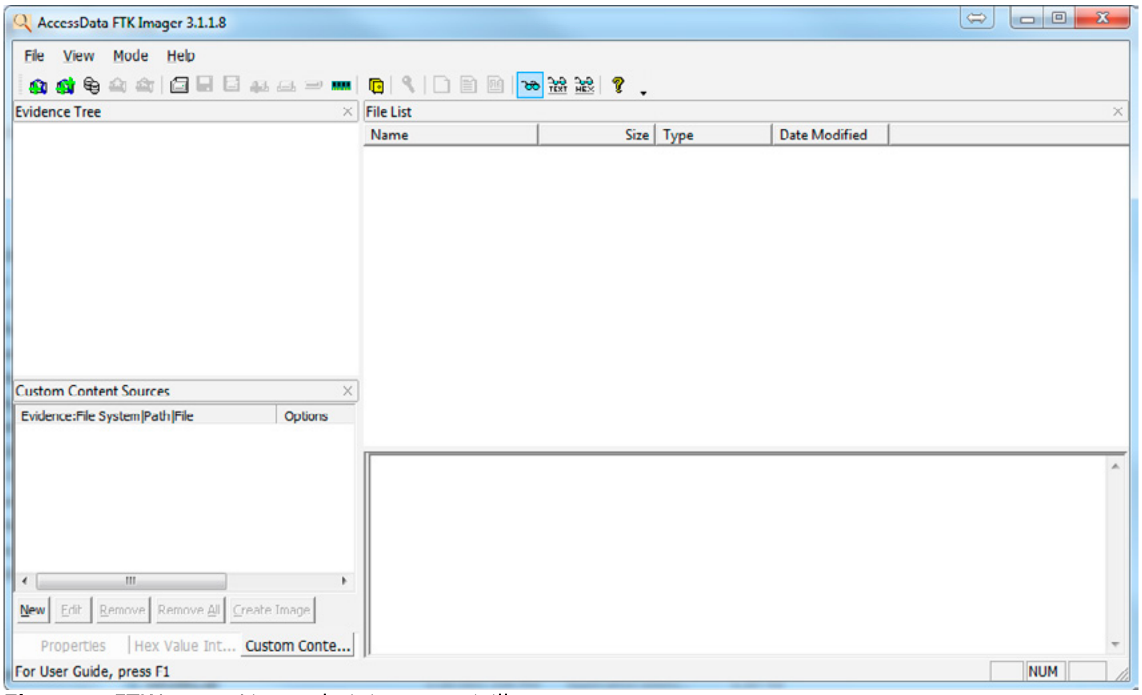
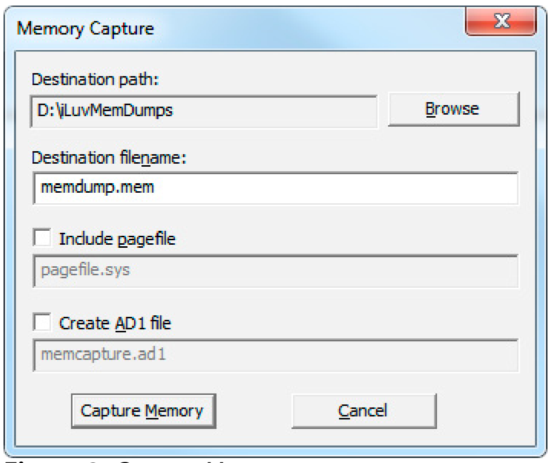


Figure 1. *FTK Imager Lite – administrator privillages*

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/*.


Memory Capture

Destination path: D:\iLuvMemDumps

Destination filename: memdump.mem

Include pagefile: pagefile.sys

Create AD1 file: memcapture.ad1

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
```

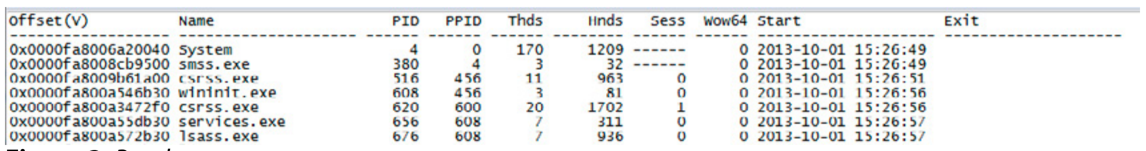
For a bird's-eye view of what Volatility can do, check out the cheat sheet here: *https://code.google.com/p/volatility/downloads/detail?name=CheatSheet_v2.3.pdf*.

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:

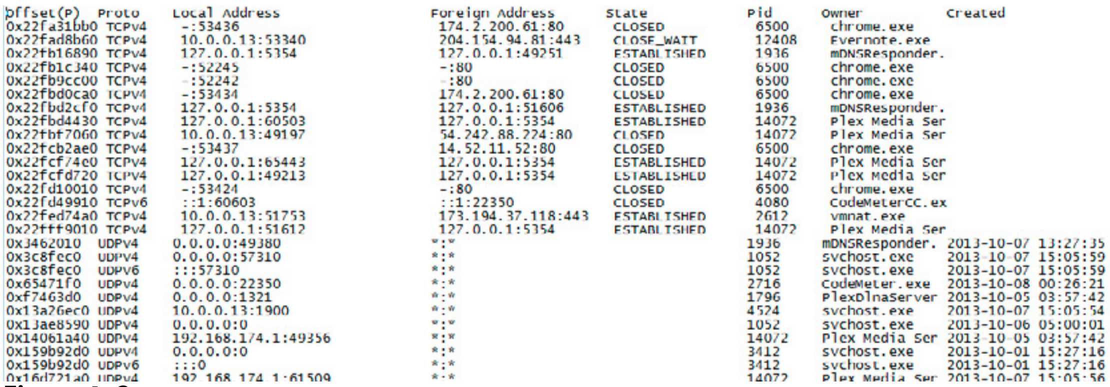
```
volatility.exe -f D:\iLuvMemDumps\memdump.mem --profile=Win7SP1x64 consoles > consoles.txt
```

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.
0x22fb1c340 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
0x22fcf74e0 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
0x3462010  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
0x13a26ec0 UDPv4    10.0.0.13:1900                 *:*                                 4524     svchost.exe    2013-10-07 15:05:54
0x13ae8590 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.

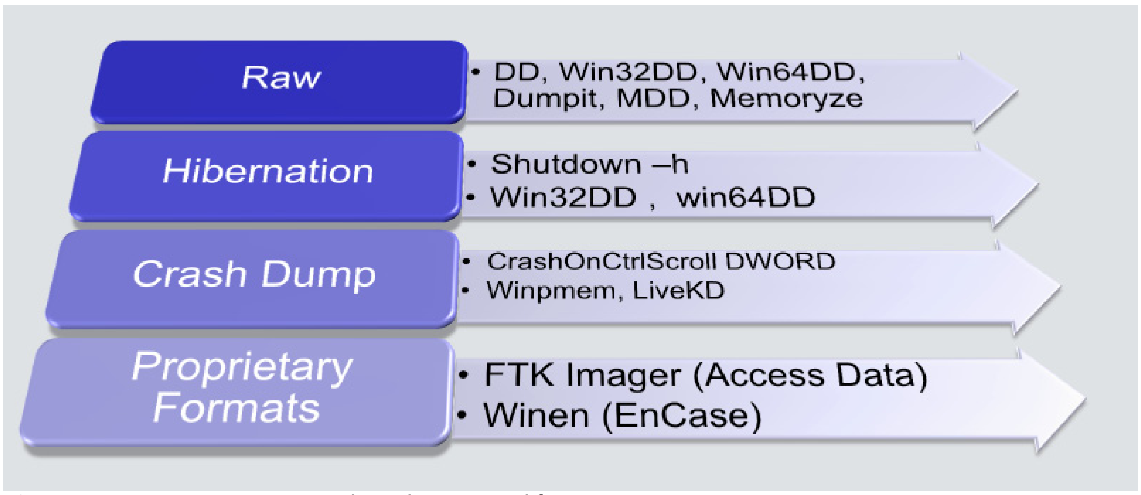


Figure 1. *Memory Dumping tools and associated formats*

CHALLENGES IN GETTING THE MEMORY DUMP

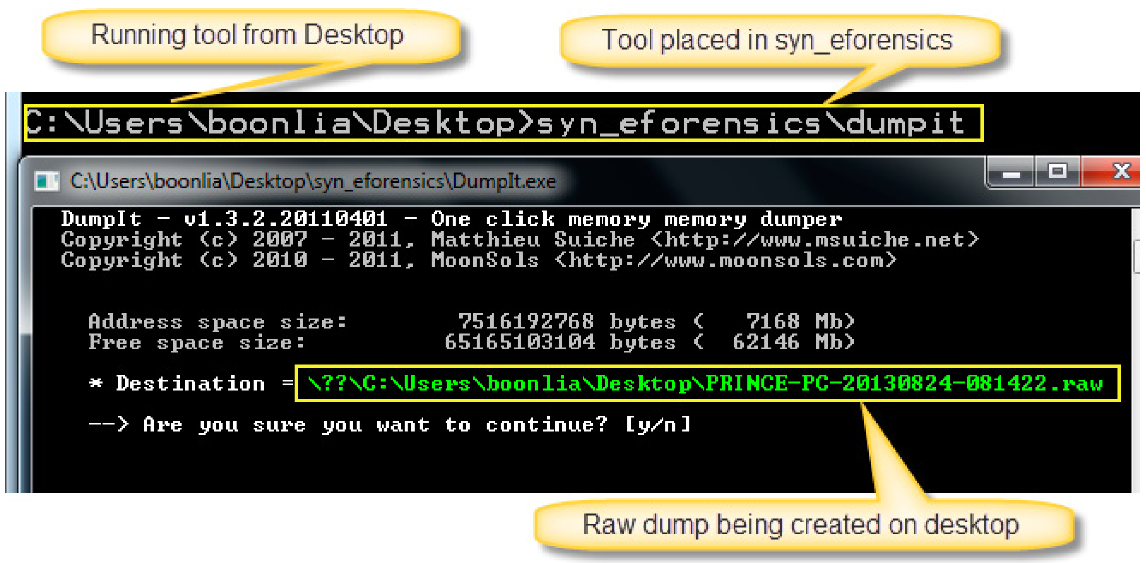
The memory itself is very dynamic and volatile. It keeps on changing continuously therefore even two memory dumps created at the same time cannot be identical. There is no way as of now to capture this dump without writing in the memory itself (Fortunately due to the way windows handle memory, the impact 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

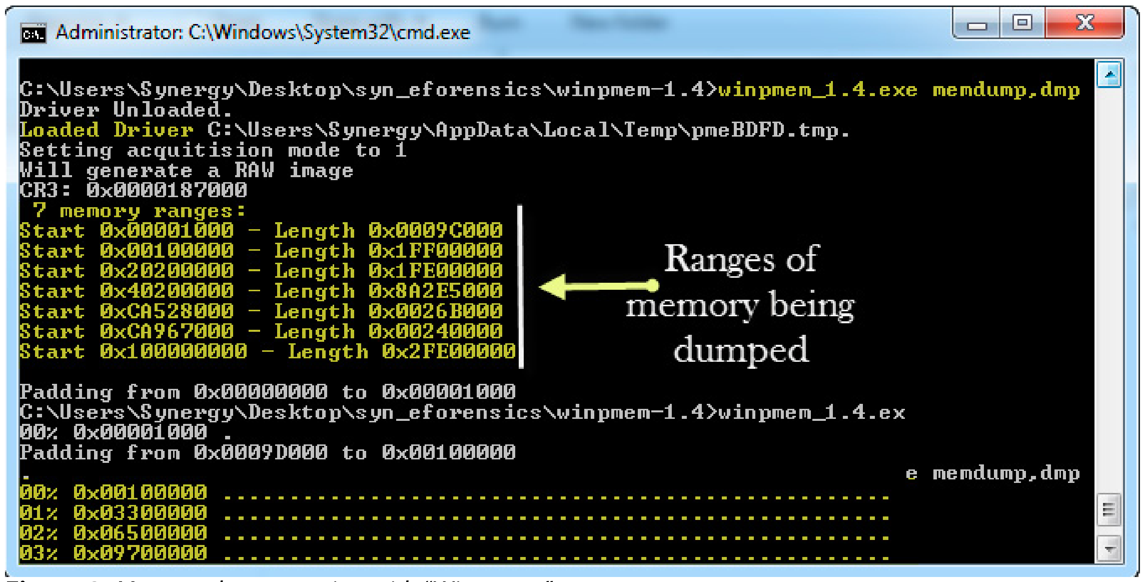
"Winpmem". Both leaves minimal footprint in the memory and are quiet efficient.

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.


Running tool from Desktop
Tool placed in syn_eforensics
C:\Users\boonlia\Desktop\syn_eforensics\DumpIt.exe
Raw dump being created on desktop

```
C:\Users\boonlia\Desktop>syn_eforensics\dumpit

  DumpIt - v1.3.2.20110401 - One click memory memory dumper
  Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
  Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>


    Address space size:         7516192768 bytes (   7168 Mb)
    Free space size:           65165103104 bytes (  62146 Mb)

    * Destination = \??\C:\Users\boonlia\Desktop\PRINCE-PC-20130824-081422.raw

    --> Are you sure you want to continue? [y/n]
```

Figure 2. *Memory dump creation with "Dumpit"*


Administrator: C:\Windows\System32\cmd.exe
Ranges of memory being dumped

```
C:\Users\Synergy\Desktop\syn_eforensics\winpmem-1.4>winpmem_1.4.exe memdump.dmp
Driver Unloaded.
Loaded Driver C:\Users\Synergy\AppData\Local\Temp\pmeBDFD.tmp.
Setting acquitision mode to 1
Will generate a RAW image
CR3: 0x0000187000
 7 memory ranges:
Start 0x00001000 - Length 0x0009C000
Start 0x00100000 - Length 0x1FF00000
Start 0x20200000 - Length 0x1FE00000
Start 0x40200000 - Length 0x8A2E5000
Start 0xCA528000 - Length 0x0026B000
Start 0xCA967000 - Length 0x00240000
Start 0x100000000 - Length 0x2FE00000

Padding from 0x00000000 to 0x00001000
C:\Users\Synergy\Desktop\syn_eforensics\winpmem-1.4>winpmem_1.4.ex
00% 0x00001000 .
Padding from 0x0009D000 to 0x00100000
.                                                                  e memdump.dmp
00% 0x00100000 ..................................................
01% 0x03300000 ..................................................
02% 0x06500000 ..................................................
03% 0x09700000 ..................................................
```

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.

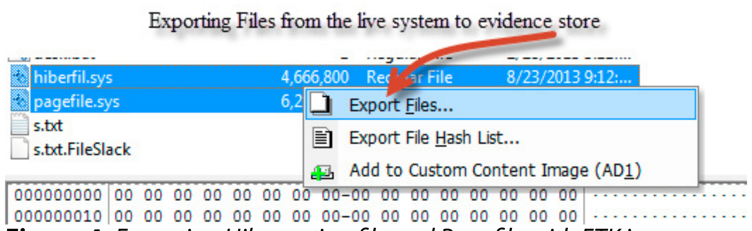


Figure 4. *Exporting Hibernation file and Pagefile with FTK imager*

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

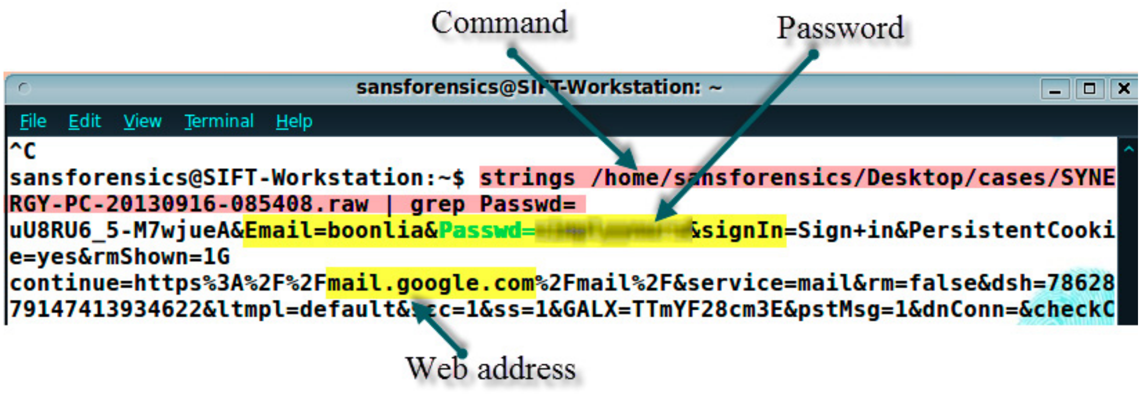
While there are several tools and frameworks available for systematic analysis of memory dump, raw analysis still holds good and can carve out information that may not be available with these tools.

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.


Command
Password
sansforensics@SIFT-Workstation: ~
File Edit View Terminal Help
Web address

```
^C
sansforensics@SIFT-Workstation:~$ strings /home/sansforensics/Desktop/cases/SYNE
RGY-PC-20130916-085408.raw | grep Passwd=
uU8RU6_5-M7wjueA&Email=boonlia&Passwd=[redacted]&signIn=Sign+in&PersistentCooki
e=yes&rmShown=1G
continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&service=mail&rm=false&dsh=78628
79147413934622&ltmpl=default&scc=1&ss=1&GALX=TTmYF28cm3E&pstMsg=1&dnConn=&checkC
```

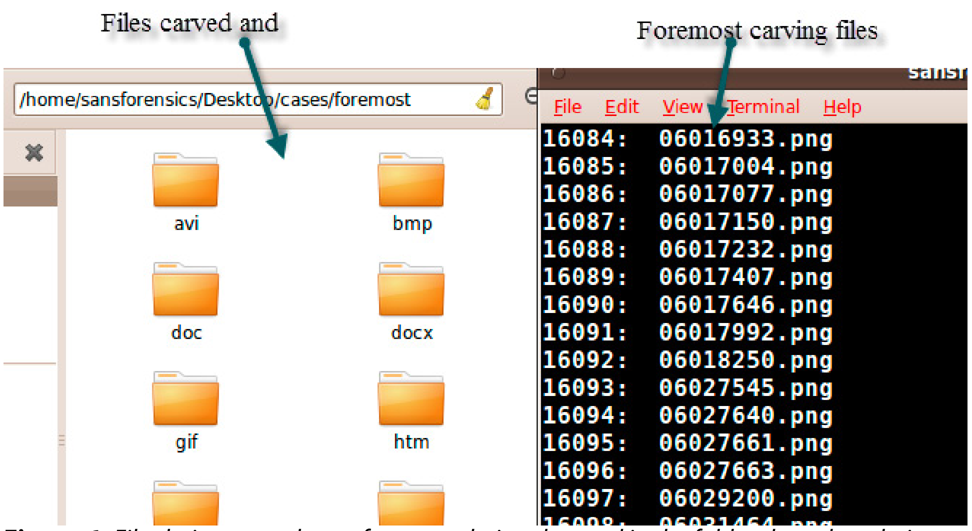
Figure 5. *Searching Gmail password through string searches*

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}`.


Files carved and
Foremost carving files
/home/sansforensics/Desktop/cases/foremost
avi
bmp
doc
docx
gif
htm
File Edit View Terminal Help

```
16084:   06016933.png
16085:   06017004.png
16086:   06017077.png
16087:   06017150.png
16088:   06017232.png
16089:   06017407.png
16090:   06017646.png
16091:   06017992.png
16092:   06018250.png
16093:   06027545.png
16094:   06027640.png
16095:   06027661.png
16096:   06027663.png
16097:   06029200.png
```

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

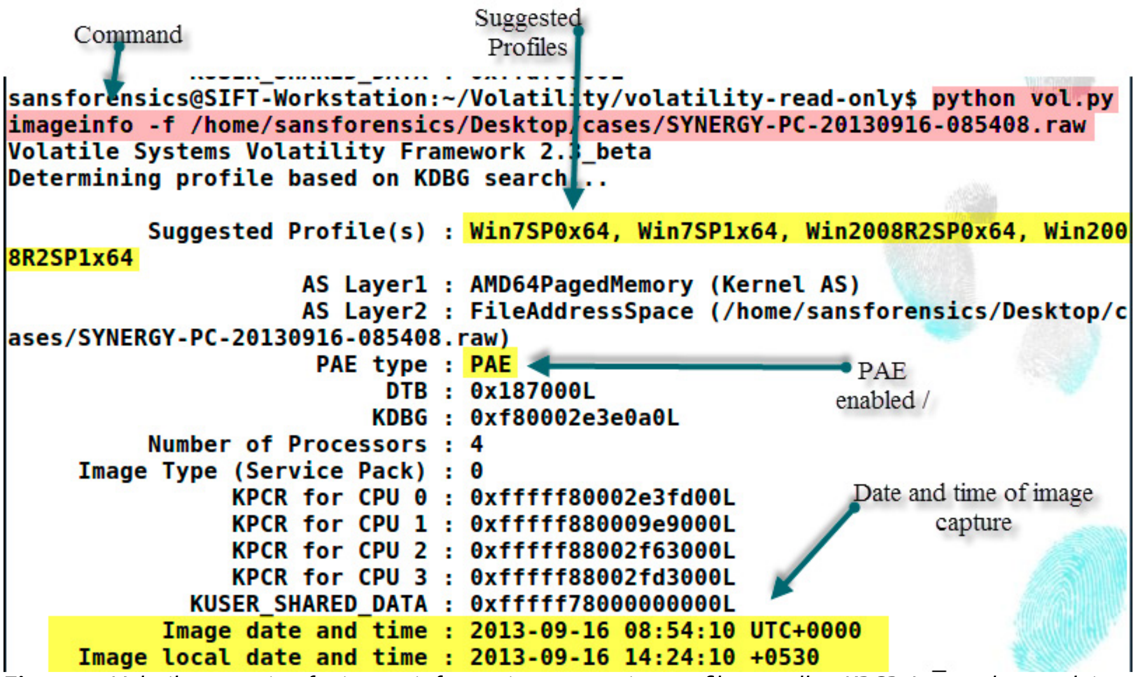
Written in Python, Volatility is most widely used framework for memory forensics. This open source framework is modular and flexible enough to incorporate your own code and plugins. Throughout this section I have used word "option" to indicate the "plugins" that are provided with volatility.

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.



```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py
imageinfo -f /home/sansforensics/Desktop/cases/SYNERGY-PC-20130916-085408.raw
Volatile Systems Volatility Framework 2.3_beta
Determining profile based on KDBG search...

          Suggested Profile(s) : Win7SP0x64, Win7SP1x64, Win2008R2SP0x64, Win200
8R2SP1x64
                     AS Layer1 : AMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/sansforensics/Desktop/c
ases/SYNERGY-PC-20130916-085408.raw)
                      PAE type : PAE
                           DTB : 0x187000L
                          KDBG : 0xf80002e3e0a0L
          Number of Processors : 4
     Image Type (Service Pack) : 0
                KPCR for CPU 0 : 0xfffff80002e3fd00L
                KPCR for CPU 1 : 0xfffff880009e9000L
                KPCR for CPU 2 : 0xfffff88002f63000L
                KPCR for CPU 3 : 0xfffff88002fd3000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2013-09-16 08:54:10 UTC+0000
     Image local date and time : 2013-09-16 14:24:10 +0530
```

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.

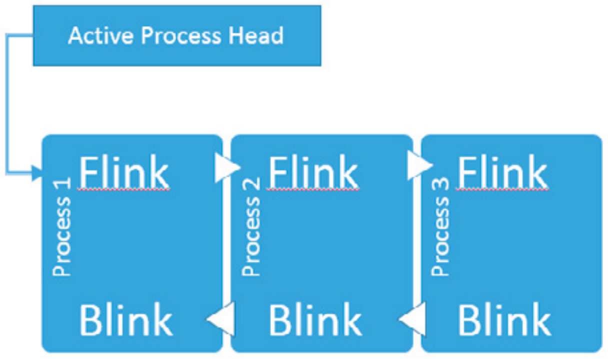


Figure 8. *Processes connected with Flink and Blink forming a linked chain of processes*

With Direct kernel Manipulation a process can be hidden by removing it from this chain as shown in “Figure 9”.

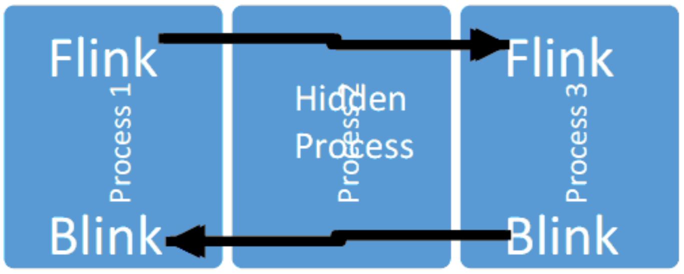


Figure 9. *Process being hidden by Direct Kernel object manipulation (DKOM) as process 2 removes the Flink and Blink structures*

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.

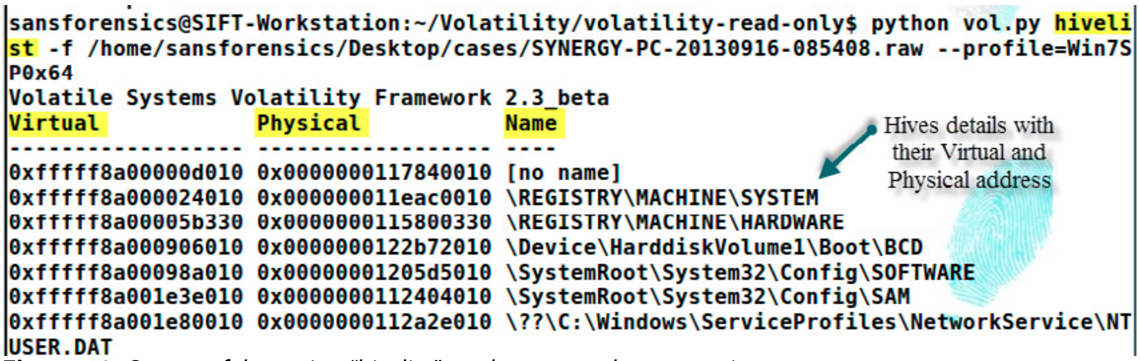
- *Dlllist*: lists the dll's along with their path for every process separately.
- *Vadinfo*: shows the information about the VAD tree structure
- *Handles*: Provides the information about open handles and their types (File, registry key, process, section, event etc.) along with the PID to which they 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
```

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

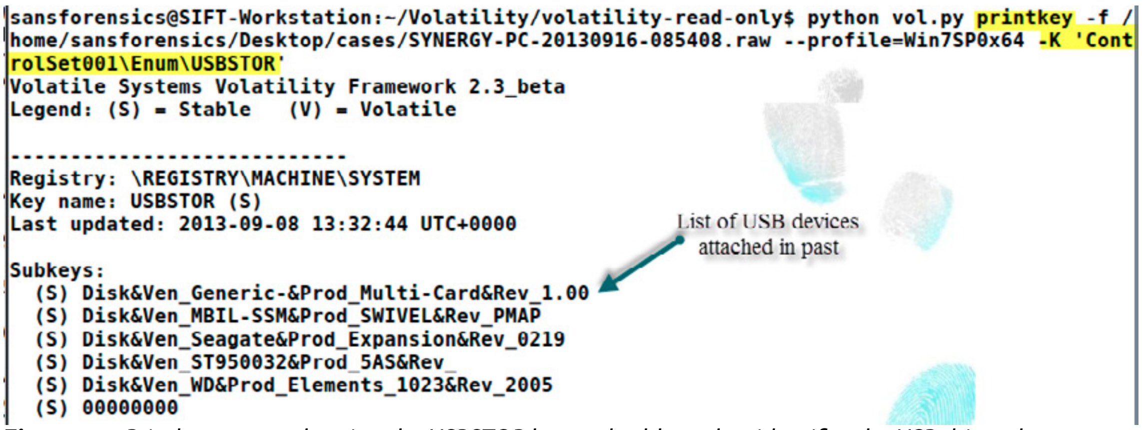
```
“python vol.py hivedump -f {memory dump file with the path} --profile={Profile} -o {Virtual address of the
   relevant hive (0xfffff8a000024010)} > {File name with path where we want to dump the output (system.
   txt)} ”
```

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 “print-key” 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.



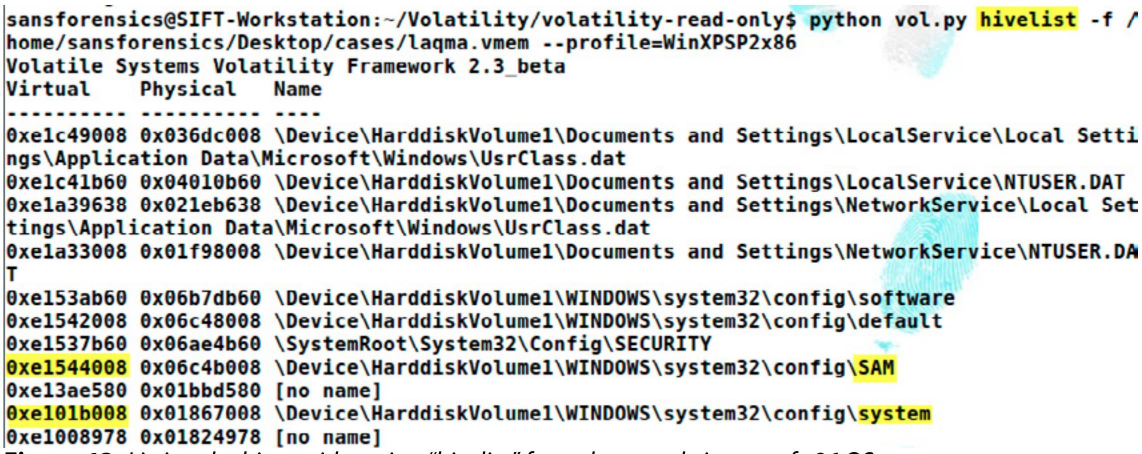
```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py printkey -f /
home/sansforensics/Desktop/cases/SYNERGY-PC-20130916-085408.raw --profile=Win7SP0x64 -K 'Cont
rolSet001\Enum\USBSTOR'
Volatile Systems Volatility Framework 2.3_beta
Legend: (S) = Stable   (V) = Volatile

----------------------------
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
```

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.

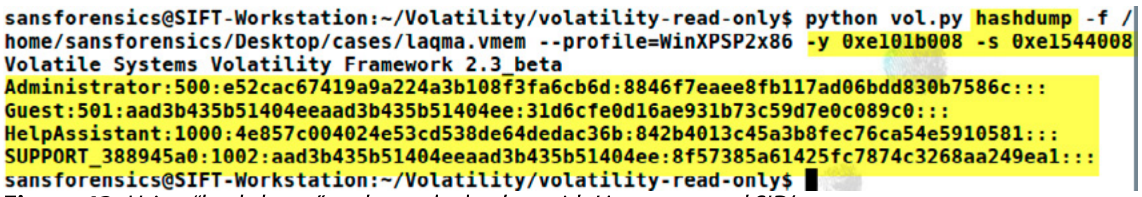
First we identify the location of System and SAM hives with the option "hivelist" as shown in "Figure 12"



```
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)

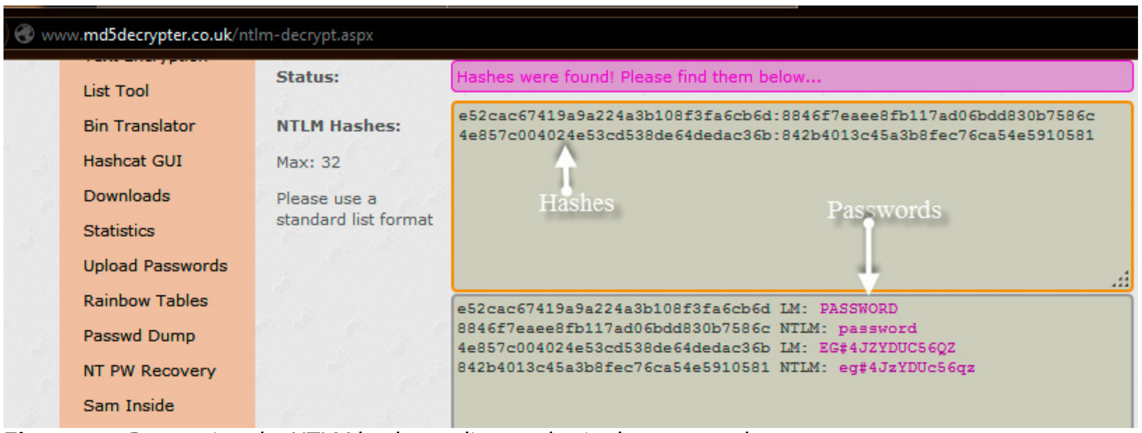


```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py hashdump -f /
home/sansforensics/Desktop/cases/laqma.vmem --profile=WinXPSP2x86 -y 0xe101b008 -s 0xe1544008
Volatile Systems Volatility Framework 2.3_beta
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
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).



Status: Hashes were found! Please find them below...

NTLM Hashes:

Max: 32

Please use a standard list format

```
e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c
4e857c004024e53cd538de64dedac36b:842b4013c45a3b8fec76ca54e5910581
```

```
e52cac67419a9a224a3b108f3fa6cb6d LM: PASSWORD
8846f7eaee8fb117ad06bdd830b7586c NTLM: password
4e857c004024e53cd538de64dedac36b LM: EG#4JZYDUC56QZ
842b4013c45a3b8fec76ca54e5910581 NTLM: eg#4JzYDUc56qz
```

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

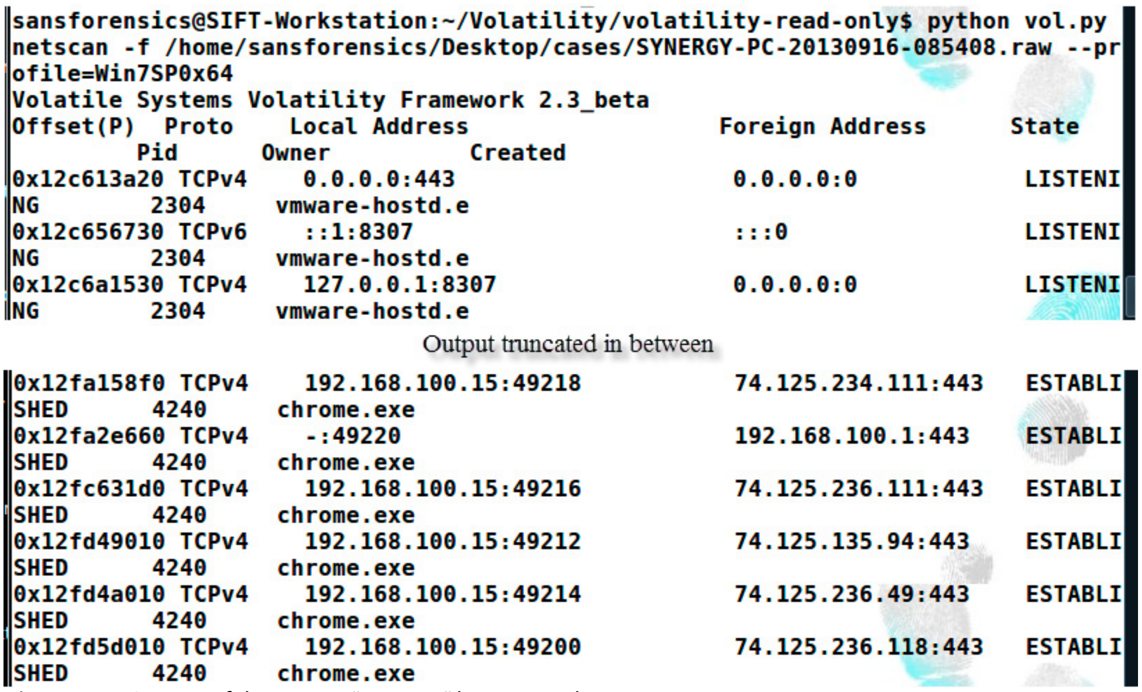
Memory forensics has a major role to play in malware analysis. Whatever activities are being carried out by malware passes through memory. As a strategy most malware tries to hide themselves from memory by adopting various methodologies, Attempts to connect to some remote location, Tries to modify the 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"



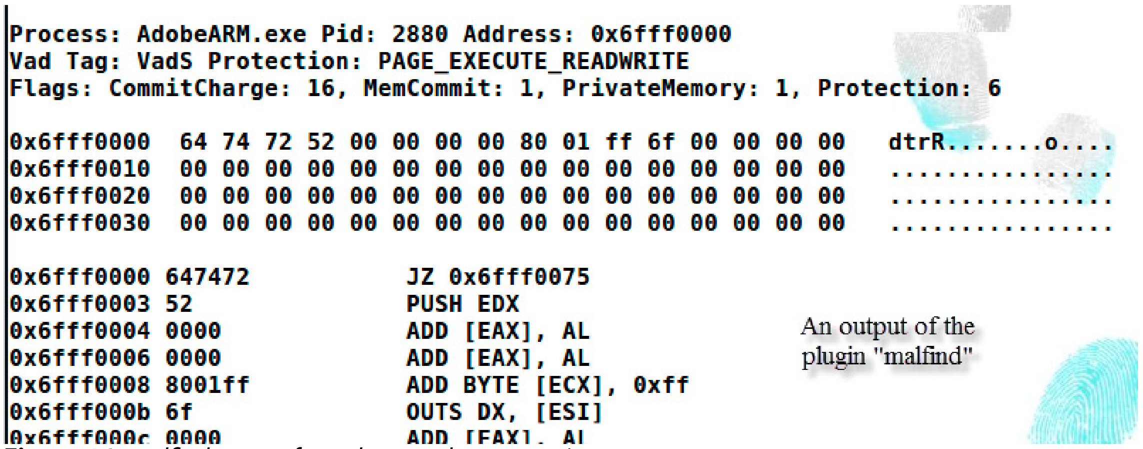
```
sansforensics@SIFT-Workstation:~/Volatility/volatility-read-only$ python vol.py
netscan -f /home/sansforensics/Desktop/cases/SYNERGY-PC-20130916-085408.raw --pr
ofile=Win7SP0x64
Volatile Systems Volatility Framework 2.3_beta
Offset(P)  Proto    Local Address                  Foreign Address      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.



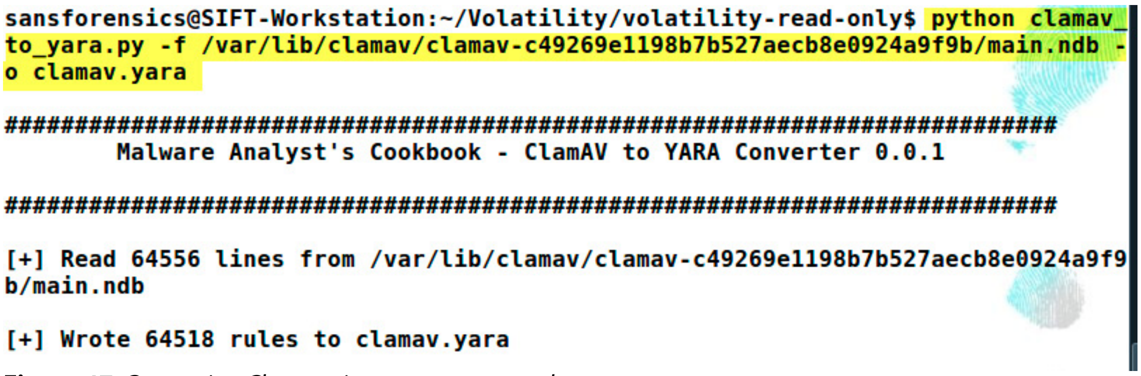
```
Process: AdobeARM.exe Pid: 2880 Address: 0x6fff0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 16, MemCommit: 1, PrivateMemory: 1, Protection: 6

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
```

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

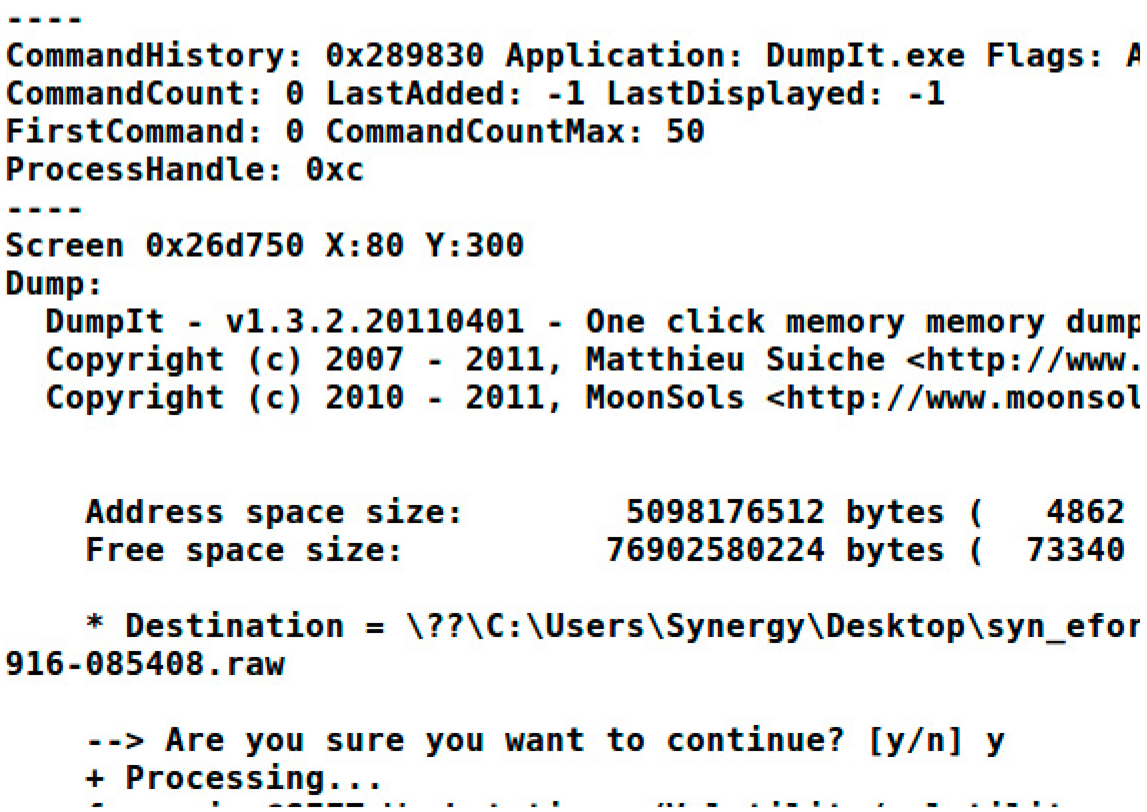
There are several other plugins that can be used in malware analysis. A brief description is being provided herewith.

- *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 andvalue. 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.

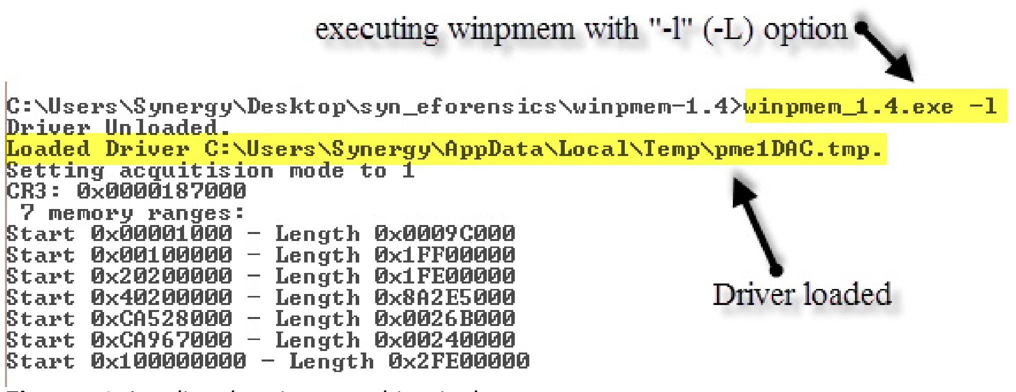
PORTING THE VOLATILITY TO THE LIVE MEMORY

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.



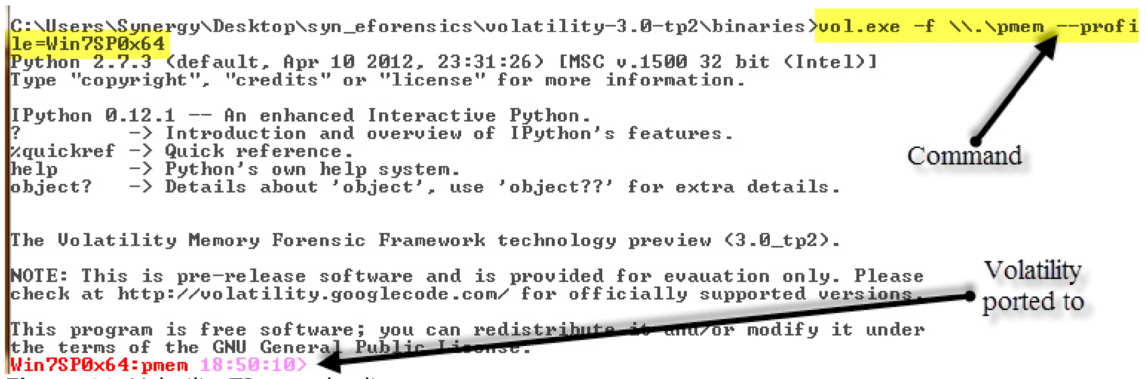
```
C:\Users\Synergy\Desktop\syn_eforensics\winpmem-1.4>winpmem_1.4.exe -l
Driver Unloaded.
Loaded Driver C:\Users\Synergy\AppData\Local\Temp\pme1DAC.tmp.
Setting acquitision mode to 1
CR3: 0x0000187000
 7 memory ranges:
Start 0x00001000 - Length 0x0009C000
Start 0x00100000 - Length 0x1FF00000
Start 0x20200000 - Length 0x1FE00000
Start 0x40200000 - Length 0x8A2E5000
Start 0xCA528000 - Length 0x0026B000
Start 0xCA967000 - Length 0x00240000
Start 0x100000000 - Length 0x2FE00000
```

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`

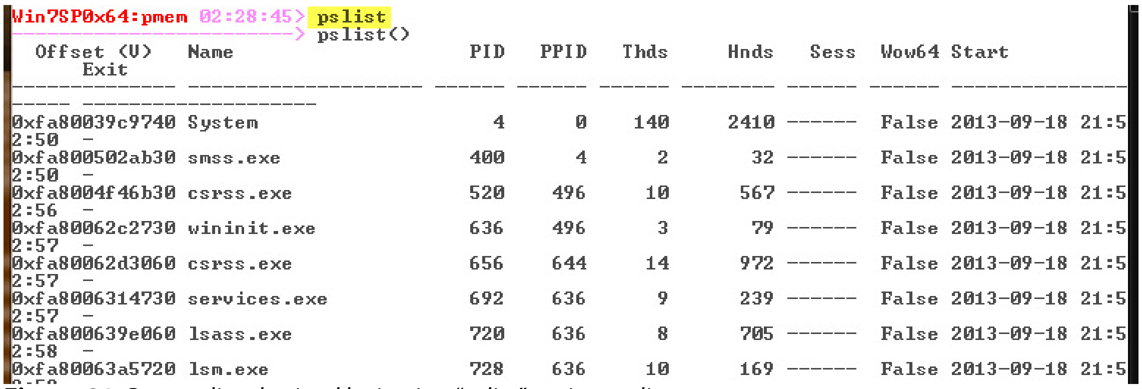
Porting the volatility tech preview (tp): Volatality tp can now be ported to the winpmem driver we loaded. This can be done by providing the psudo location as the input file. On windows system make sure that you use "vol.exe" in command line in place of "vol" as vol defaults to "Volume" that will show 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).



```
C:\Users\Synergy\Desktop\syn_eforensics\volatility-3.0-tp2\binaries>vol.exe -f \\.\pmem --profi
le=Win7SP0x64
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 0.12.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.


The Volatility Memory Forensic Framework technology preview (3.0_tp2).

NOTE: This is pre-release software and is provided for evauation only. Please
check at http://volatility.googlecode.com/ for officially supported versions.

This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License.
Win7SP0x64:pmem 18:50:10>
```

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.

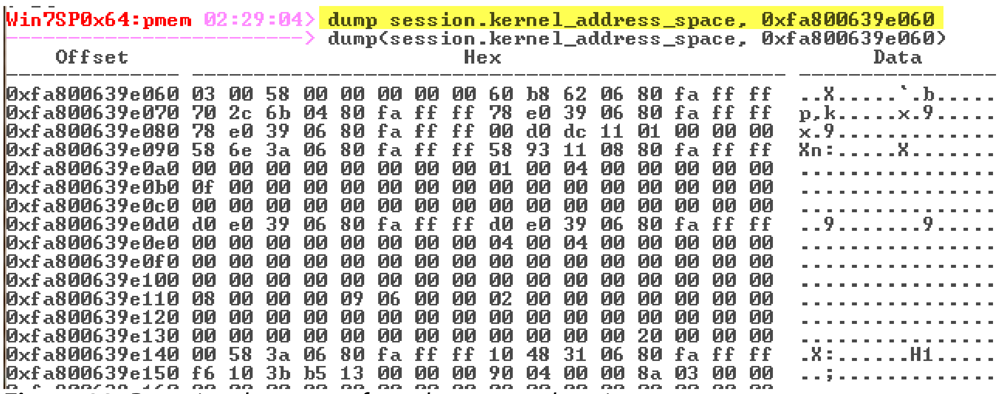
While one can run all the options available with volatility, this porting allows one to use other python command as well. As it's based on IPython (Interactive Python) one can get all the help required along with the options / commands available.

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

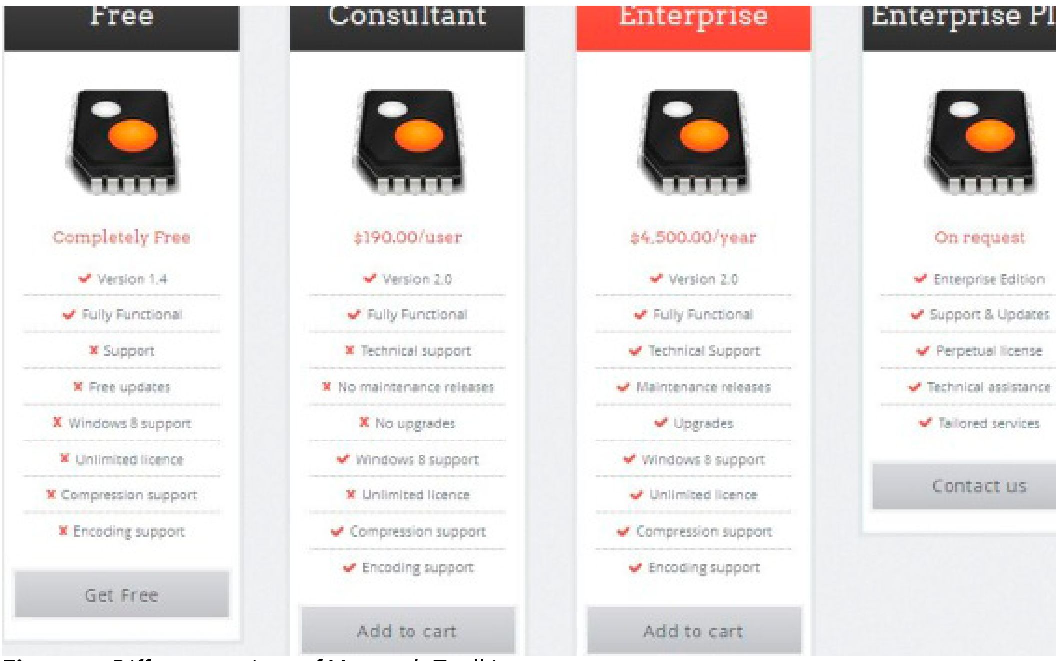
This tool can be downloaded here: *http://www.moonsols.com/#pricing.*

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:



Free	Consultant	Enterprise	Enterprise Pl
Completely Free	$190.00/user	$4,500.00/year	On request
✓ Version 1.4	✓ Version 2.0	✓ Version 2.0	✓ Enterprise Edition
✓ Fully Functional	✓ Fully Functional	✓ Fully Functional	✓ Support & Updates
✗ Support	✗ Technical support	✓ Technical Support	✓ Perpetual license
✗ Free updates	✗ No maintenance releases	✓ Maintenance releases	✓ Technical assistance
✗ Windows 8 support	✗ No upgrades	✓ Upgrades	✓ Tailored services
✗ Unlimited licence	✓ Windows 8 support	✓ Windows 8 support	Contact us
✗ Compression support	✗ Unlimited licence	✓ Unlimited licence	
✗ Encoding support	✓ Compression support	✓ Compression support	
Get Free	✓ Encoding support	✓ Encoding support	
	Add to cart	Add to cart	

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.

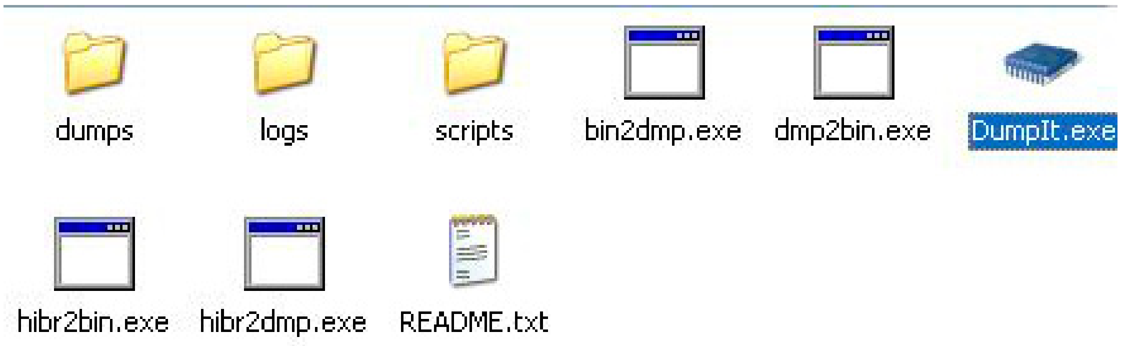
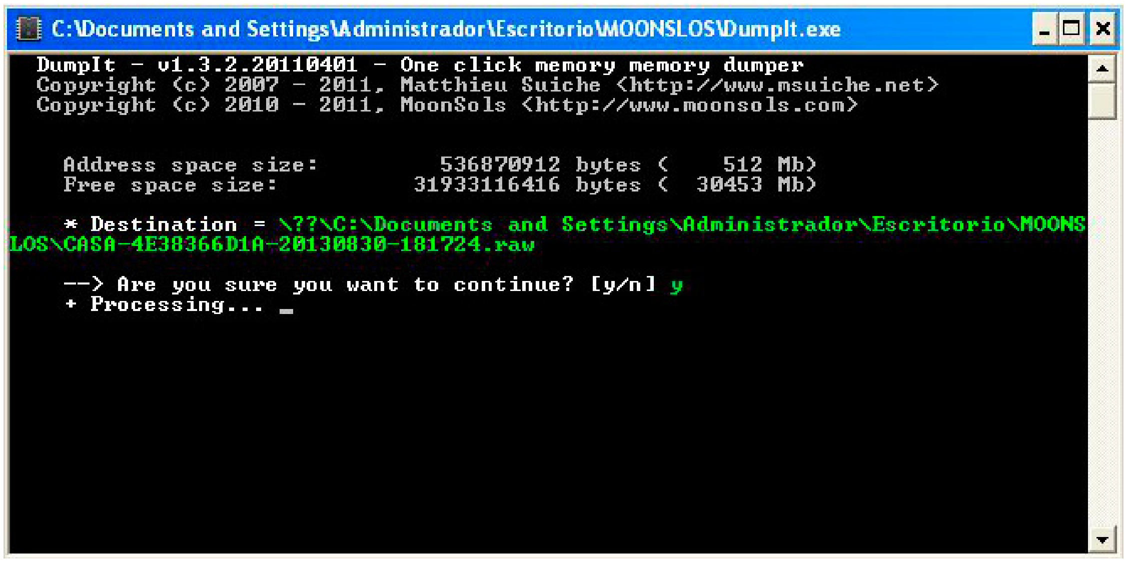


Figure 2. *Uncompressed files*

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.

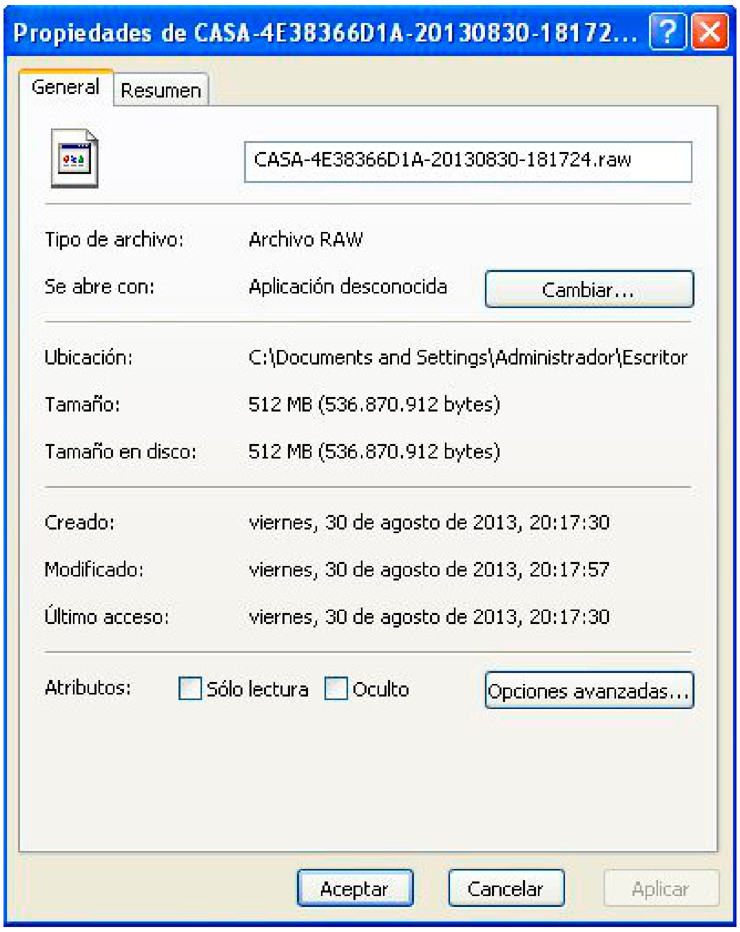
In this case, we agree with the destination folder.



```
C:\Documents and Settings\Administrador\Escritorio\MOONSLOS\DumpIt.exe
  DumpIt - v1.3.2.20110401 - One click memory memory dumper
  Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
  Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>


    Address space size:          536870912 bytes (    512 Mb)
    Free space size:           31933116416 bytes (  30453 Mb)

    * Destination = \??\C:\Documents and Settings\Administrador\Escritorio\MOONS
LOS\CASA-4E38366D1A-20130830-181724.raw

    --> Are you sure you want to continue? [y/n] y
    + Processing... _
```

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.


Propiedades de CASA-4E38366D1A-20130830-18172...

CASA-4E38366D1A-20130830-181724.raw

Tipo de archivo: Archivo RAW

Se abre con: Aplicación desconocida

Ubicación: C:\Documents and Settings\Administrador\Escritor

Tamaño: 512 MB (536.870.912 bytes)

Tamaño en disco: 512 MB (536.870.912 bytes)

Creado: viernes, 30 de agosto de 2013, 20:17:30

Modificado: viernes, 30 de agosto de 2013, 20:17:57

Último acceso: viernes, 30 de agosto de 2013, 20:17:30

Atributos: ☐ Sólo lectura ☐ Oculto

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…

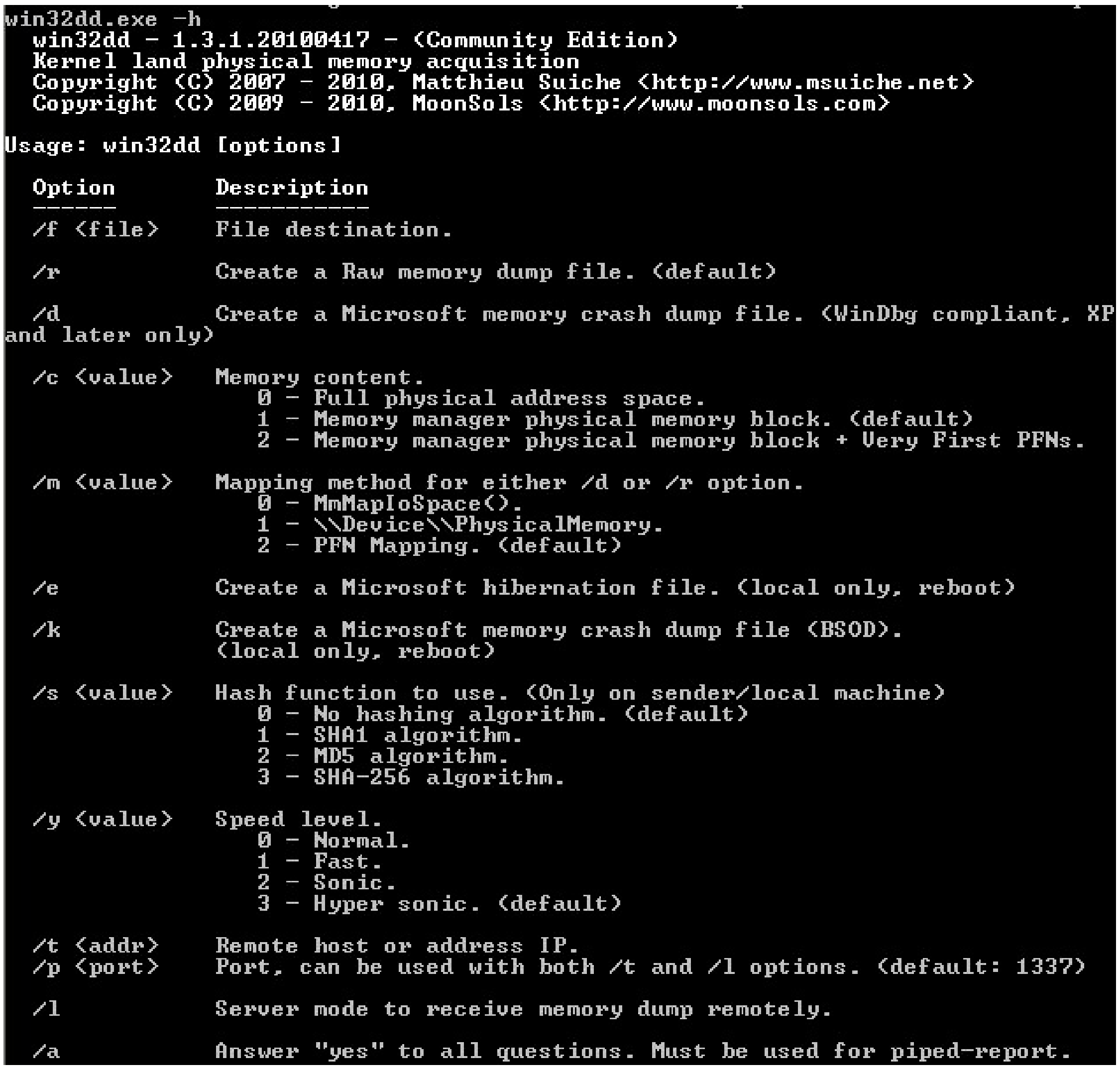
Features:

- 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
```



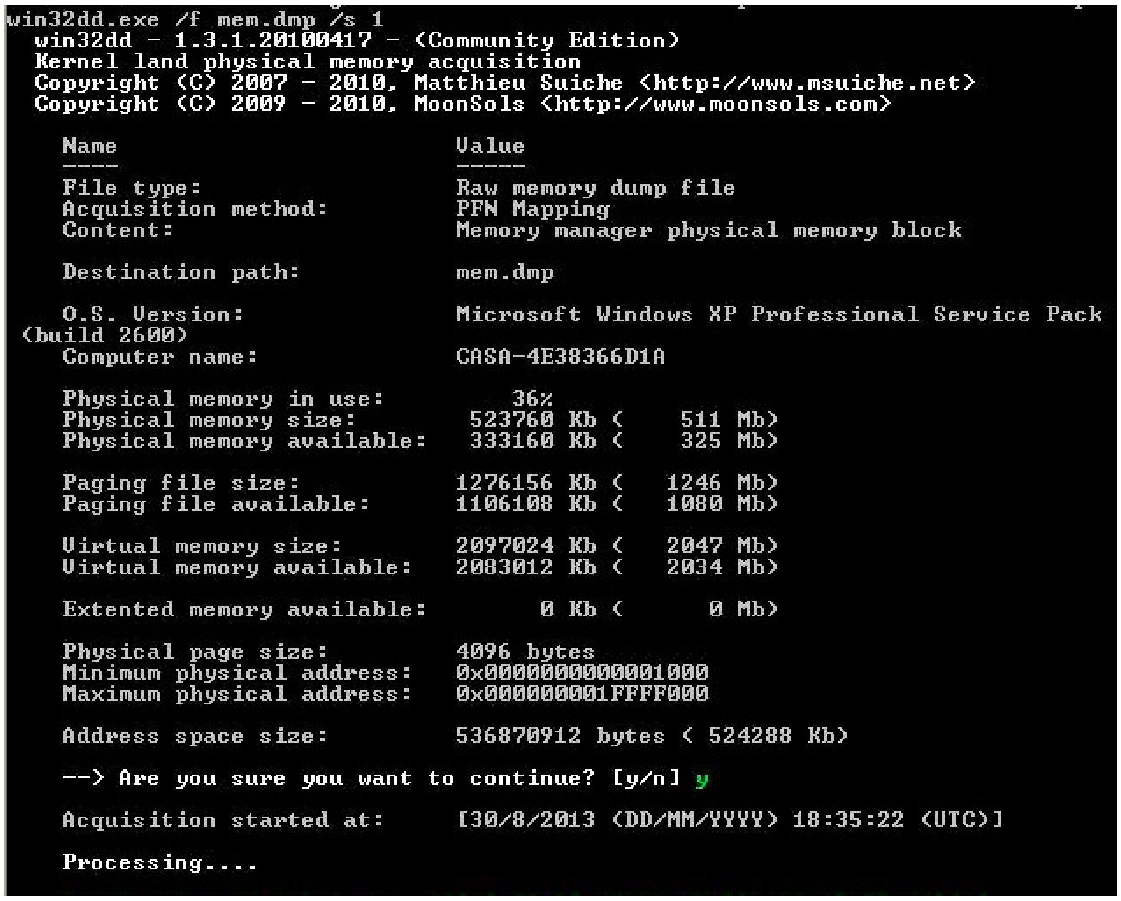
```
win32dd.exe -h
  win32dd - 1.3.1.20100417 - (Community Edition)
  Kernel land physical memory acquisition
  Copyright (C) 2007 - 2010, Matthieu Suiche <http://www.msuiche.net>
  Copyright (C) 2009 - 2010, MoonSols <http://www.moonsols.com>

Usage: win32dd [options]

  Option        Description
  ------        -----------
  /f <file>     File destination.

  /r            Create a Raw memory dump file. (default)

  /d            Create a Microsoft memory crash dump file. (WinDbg compliant, XP
and later only)

  /c <value>    Memory content.
                   0 - Full physical address space.
                   1 - Memory manager physical memory block. (default)
                   2 - Memory manager physical memory block + Very First PFNs.

  /m <value>    Mapping method for either /d or /r option.
                   0 - MmMapIoSpace().
                   1 - \\Device\\PhysicalMemory.
                   2 - PFN Mapping. (default)

  /e            Create a Microsoft hibernation file. (local only, reboot)

  /k            Create a Microsoft memory crash dump file (BSOD).
                (local only, reboot)

  /s <value>    Hash function to use. (Only on sender/local machine)
                   0 - No hashing algorithm. (default)
                   1 - SHA1 algorithm.
                   2 - MD5 algorithm.
                   3 - SHA-256 algorithm.

  /y <value>    Speed level.
                   0 - Normal.
                   1 - Fast.
                   2 - Sonic.
                   3 - Hyper sonic. (default)

  /t <addr>     Remote host or address IP.
  /p <port>     Port, can be used with both /t and /l options. (default: 1337)

  /l            Server mode to receive memory dump remotely.

  /a            Answer "yes" to all questions. Must be used for piped-report.
```

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
```

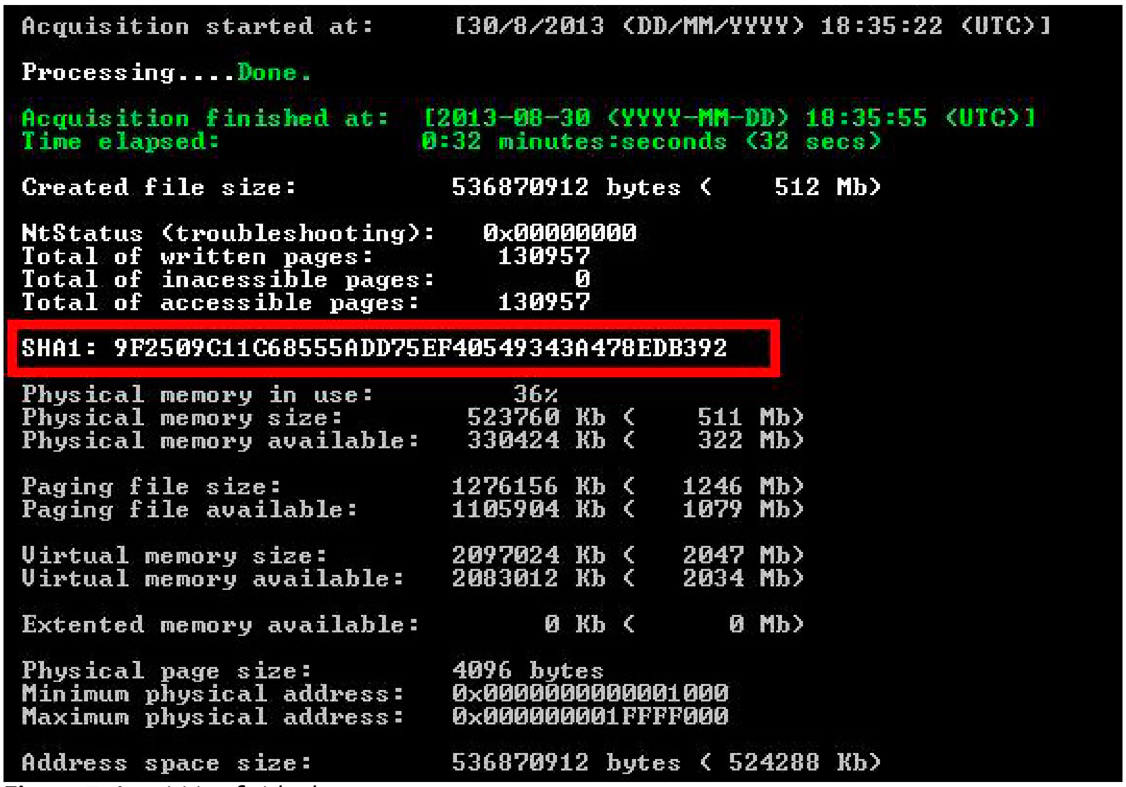
- `/f` File destination
- `/s <value>` Hash function to use. In this case we have selected 1: SHA1 algorithm.



```
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.

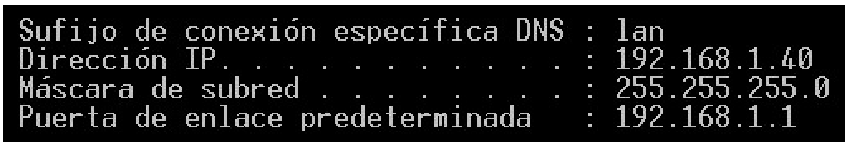
In Linux, just type this command to achieve this goal.

```
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.



```
Sufijo de conexión específica DNS : lan
Dirección IP. . . . . . . . . . . : 192.168.1.40
Máscara de subred . . . . . . . . : 255.255.255.0
Puerta de enlace predeterminada   : 192.168.1.1
```

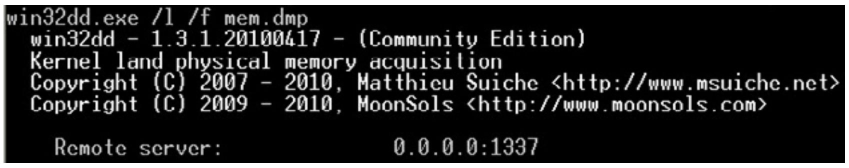
Figure 8. *Server IP address*

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
```

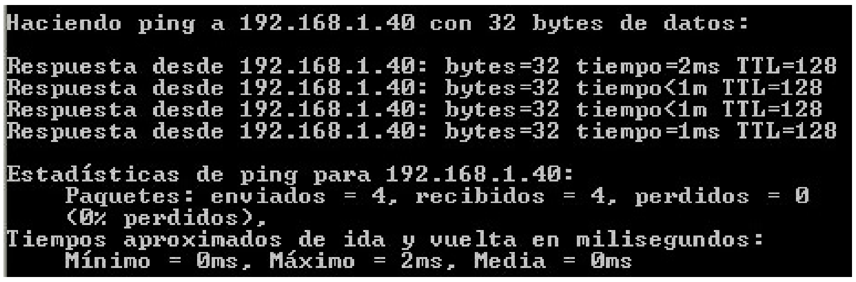


```
win32dd.exe /l /f mem.dmp
  win32dd - 1.3.1.20100417 - (Community Edition)
  Kernel land physical memory acquisition
  Copyright (C) 2007 - 2010, Matthieu Suiche <http://www.msuiche.net>
  Copyright (C) 2009 - 2010, MoonSols <http://www.moonsols.com>

    Remote server:              0.0.0.0:1337
```

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.



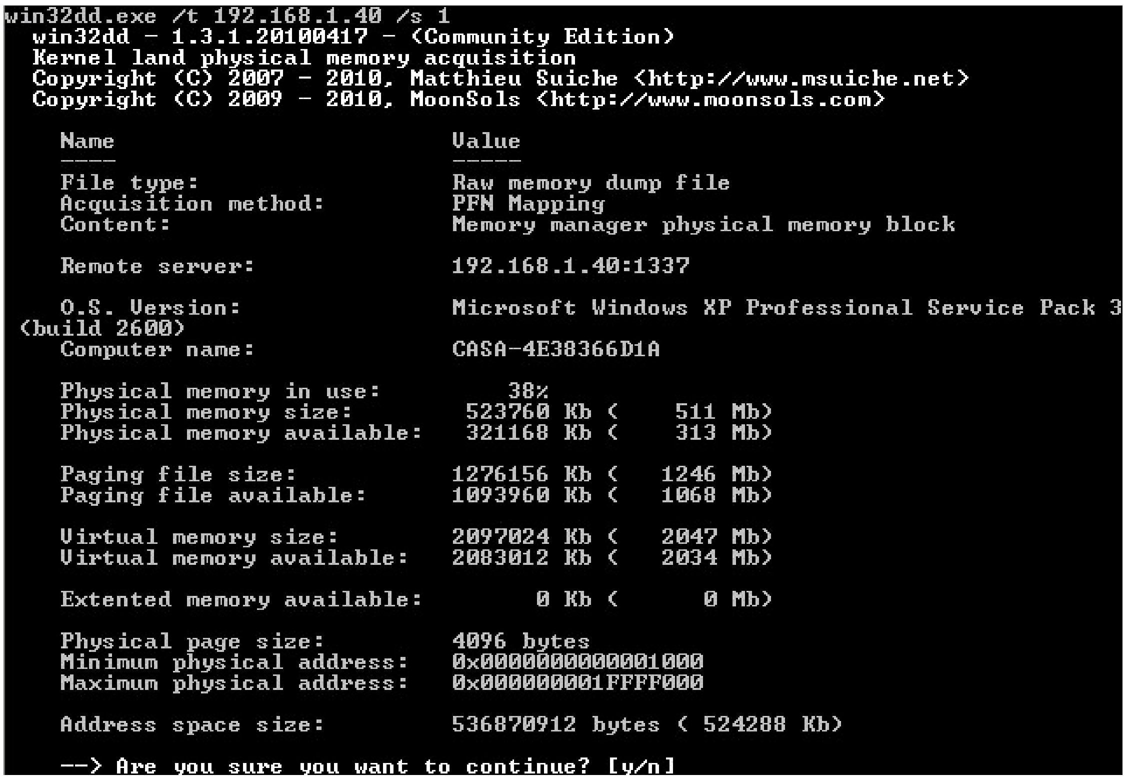
```
Haciendo ping a 192.168.1.40 con 32 bytes de datos:

Respuesta desde 192.168.1.40: bytes=32 tiempo=2ms TTL=128
Respuesta desde 192.168.1.40: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.1.40: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.1.40: bytes=32 tiempo=1ms TTL=128

Estadísticas de ping para 192.168.1.40:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 2ms, Media = 0ms
```

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
```

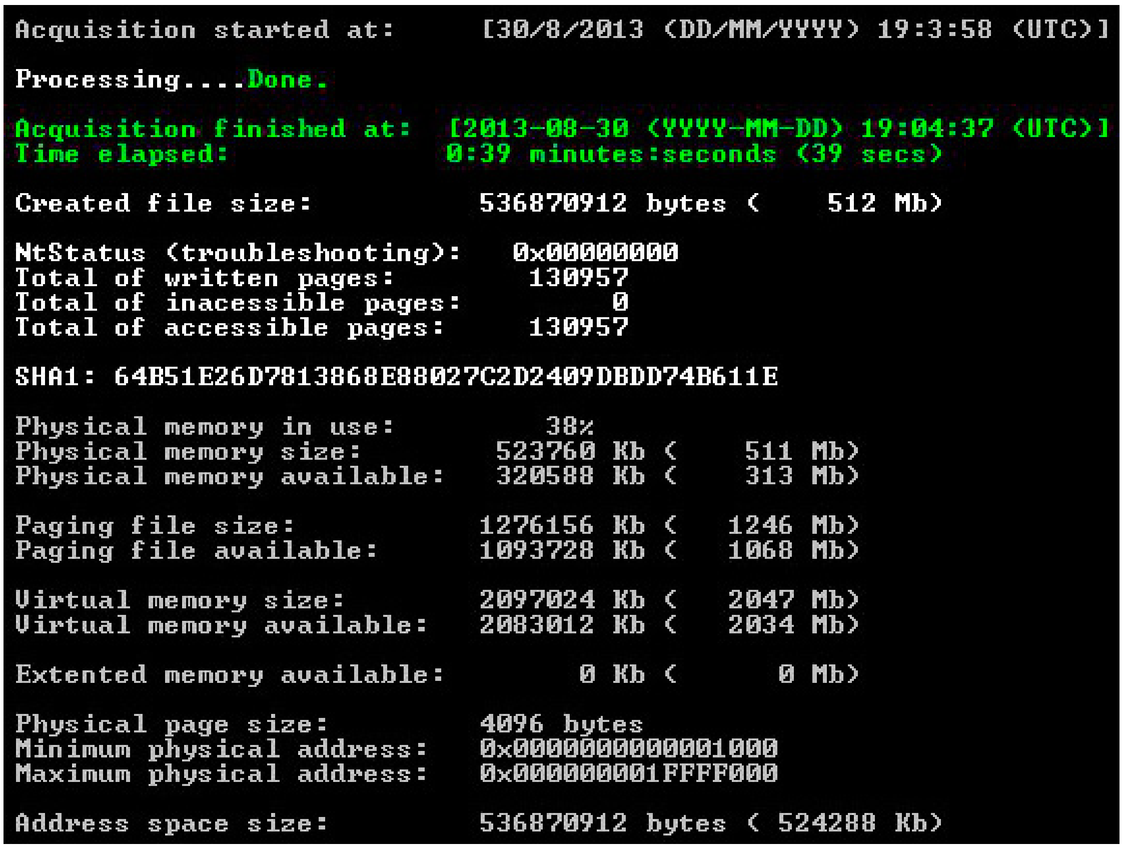
Noticed we have just selected the same options that in the example before.



```
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*

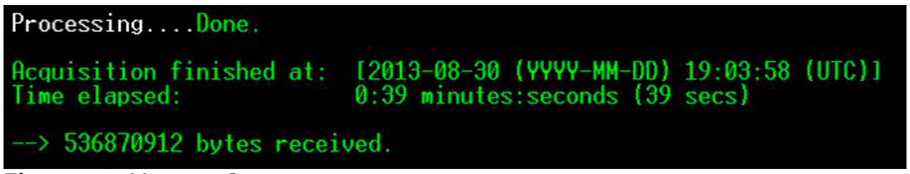
We wait just a few seconds and that it is all.



```
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.
```

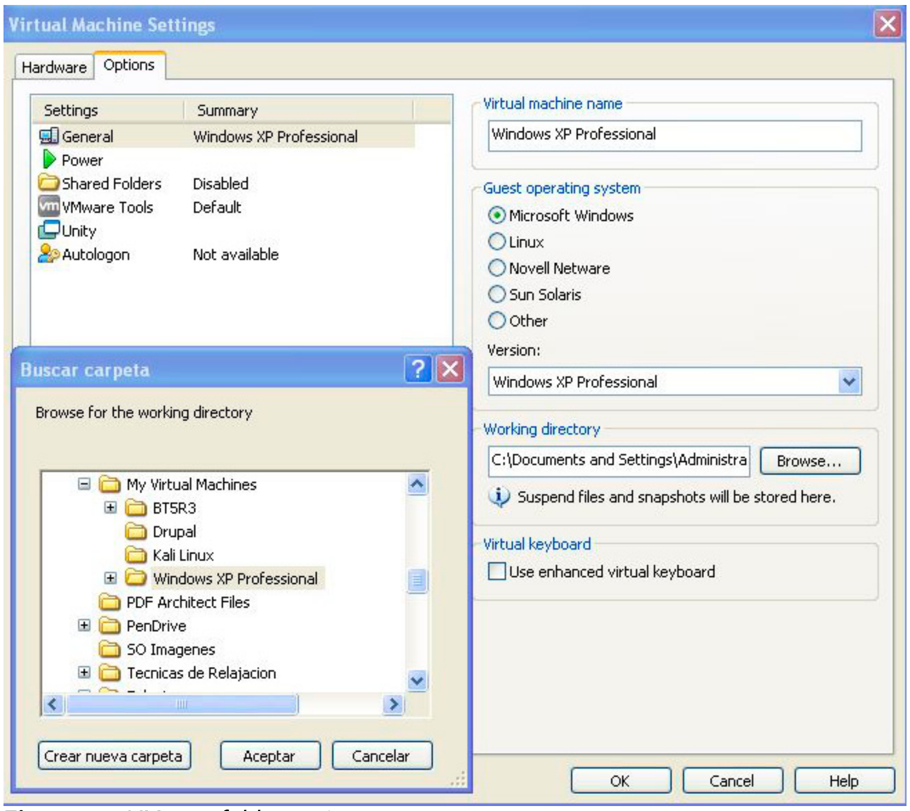
Figure 13. *Memory Gump*

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.



Virtual Machine Settings

Settings	Summary
General	Windows XP Professional
Power	
Shared Folders	Disabled
VMware Tools	Default
Unity	
Autologon	Not available

Virtual machine name: Windows XP Professional

Guest operating system: [✓ Microsoft Windows] Linux Novell Netware Sun Solaris Other

Version: Windows XP Professional

Working directory: C:\Documents and Settings\Administra

Suspend files and snapshots will be stored here.

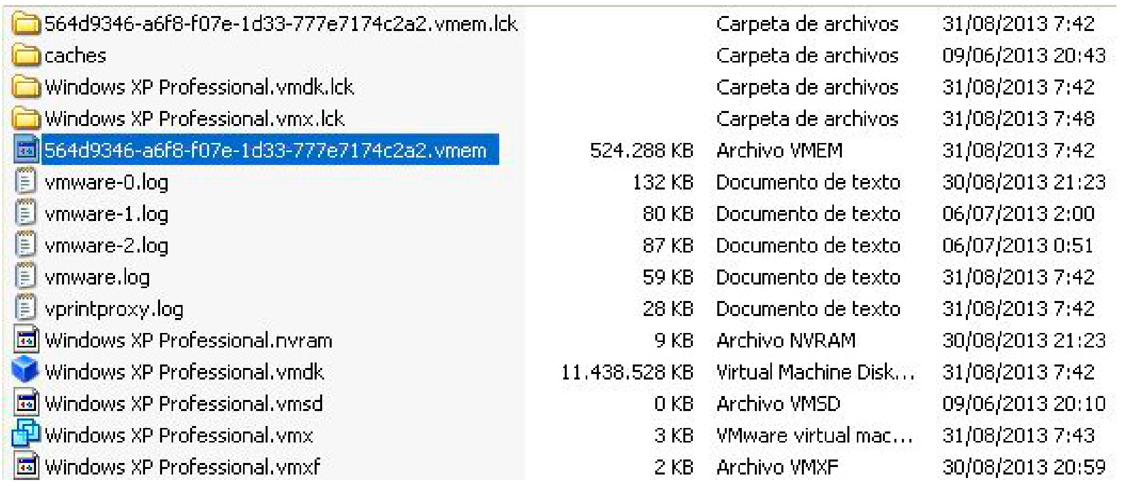
Virtual keyboard: Use enhanced virtual keyboard

Buscar carpeta

Browse for the working directory

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*

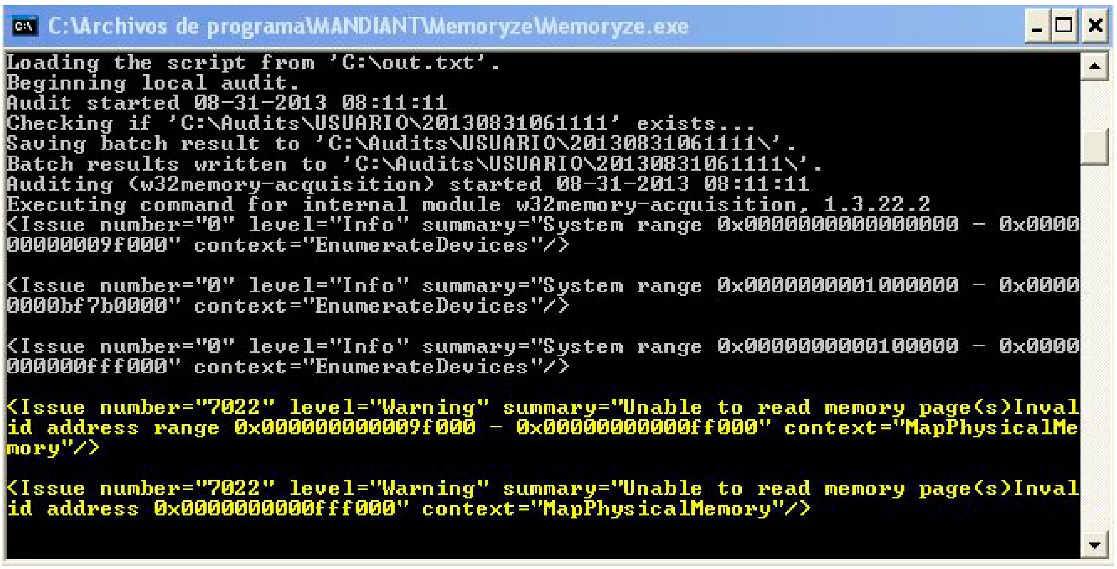
MEMORYZE

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.

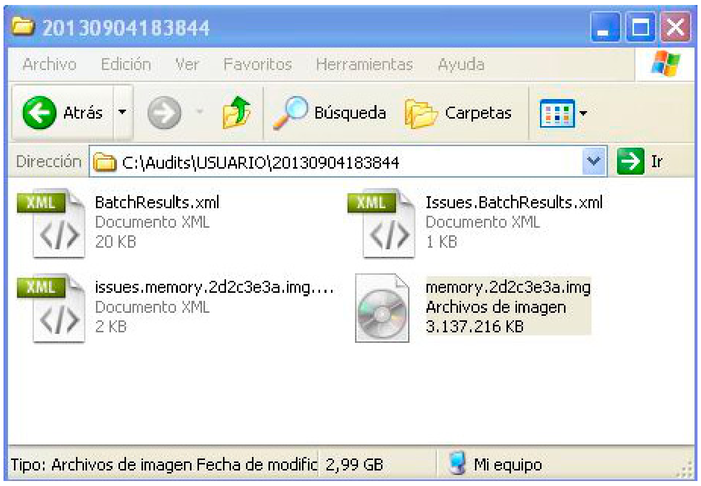
```
memoryDD.bat -output C:\
```



```
Loading the script from 'C:\out.txt'.
Beginning local audit.
Audit started 08-31-2013 08:11:11
Checking if 'C:\Audits\USUARIO\20130831061111' exists...
Saving batch result to 'C:\Audits\USUARIO\20130831061111\'.
Batch results written to 'C:\Audits\USUARIO\20130831061111\'.
Auditing (w32memory-acquisition) started 08-31-2013 08:11:11
Executing command for internal module w32memory-acquisition, 1.3.22.2
<Issue number="0" level="Info" summary="System range 0x0000000000000000 - 0x0000
00000009f000" context="EnumerateDevices"/>

<Issue number="0" level="Info" summary="System range 0x0000000001000000 - 0x0000
0000bf7b0000" context="EnumerateDevices"/>

<Issue number="0" level="Info" summary="System range 0x0000000000100000 - 0x0000
000000fff000" context="EnumerateDevices"/>

<Issue number="7022" level="Warning" summary="Unable to read memory page(s)Inval
id address range 0x000000000009f000 - 0x0000000000ff f000" context="MapPhysicalMe
mory"/>

<Issue number="7022" level="Warning" summary="Unable to read memory page(s)Inval
id address 0x0000000000fff000" context="MapPhysicalMemory"/>
```

Figure 16. *Memory dump process*

MemoryDD.bat has just created a directory called audits where memory image has been saved.


20130904183844

Archivo Edición Ver Favoritos Herramientas Ayuda

Atrás Búsqueda Carpetas

Dirección C:\Audits\USUARIO\20130904183844 Ir

BatchResults.xml
Documento XML
20 KB

Issues.BatchResults.xml
Documento XML
1 KB

issues.memory.2d2c3e3a.img....
Documento XML
2 KB

memory.2d2c3e3a.img
Archivos de imagen
3.137.216 KB

Tipo: Archivos de imagen Fecha de modific 2,99 GB Mi equipo

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.

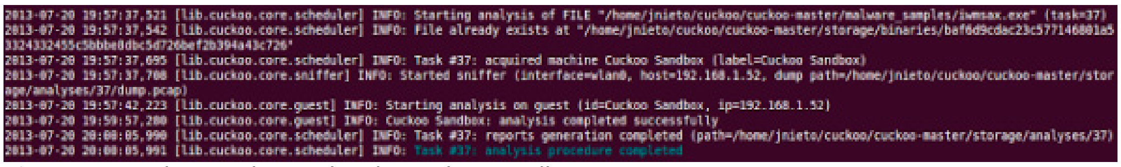
In this article, we are going to focus in how to create a memory dump with this tool. It's really easy to do. First of all, it's necessary to install Cuckoo on a Linux Machine. You can learn how to install Cuckoo in my blog site: *http://www.behindthefirewalls.com/2013/07/how-to-install-cuckoo-sandbox-on-ubuntu.html*.

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.

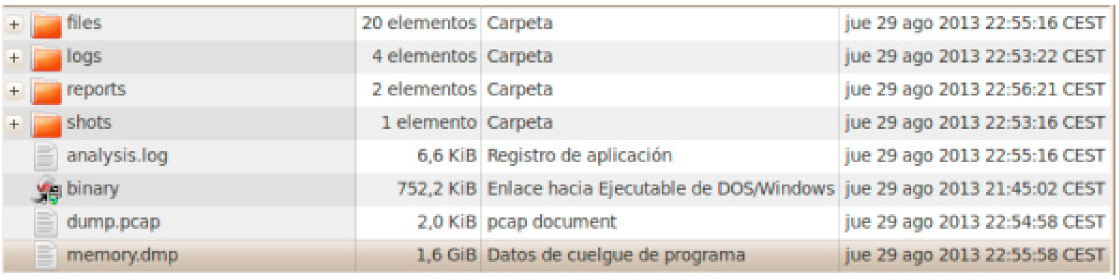
```
python submit.py cuckoo/cuckoo-master/malware_samples/iwmsax.exe
```



```
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
3324332455c5bbbe0dbc5d726bef2b394a43c726"
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 19:59:57,200 [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.

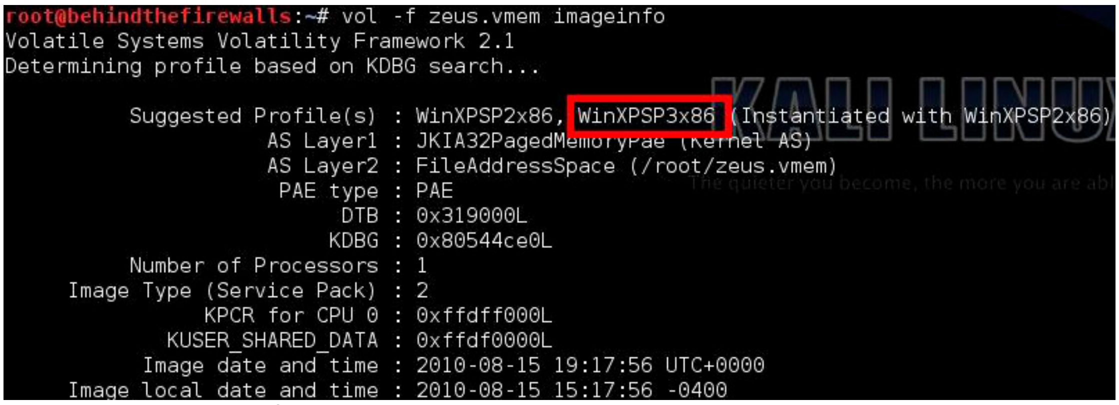
I think is a good option for beginners to start with this memory dump because there are a lot of examples in the Internet that could help you.

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*

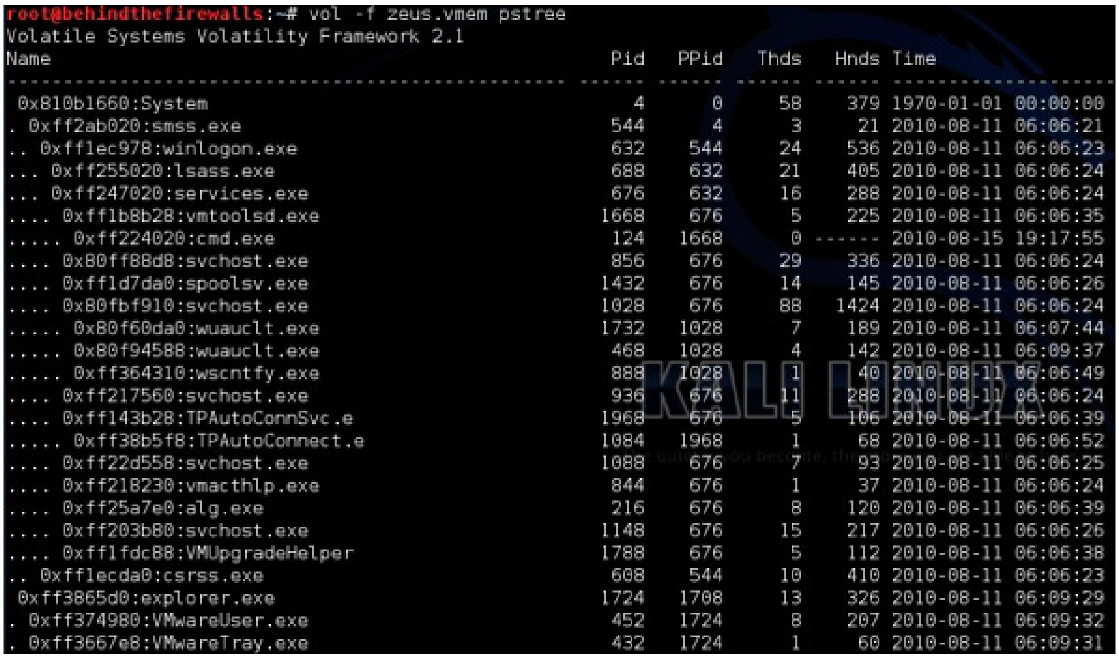
Now, we are going to export the profile "WinXPSP3x86" in order to don't specify this profile in the Volatility's commands when we are working with this tool. This will do our live easier.

```
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
```



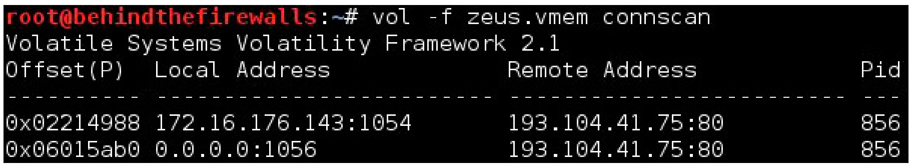
```
root@behindthefirewalls:~# vol -f zeus.vmem pstree
Volatile Systems Volatility Framework 2.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ -------------------
 0x810b1660:System                                      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.

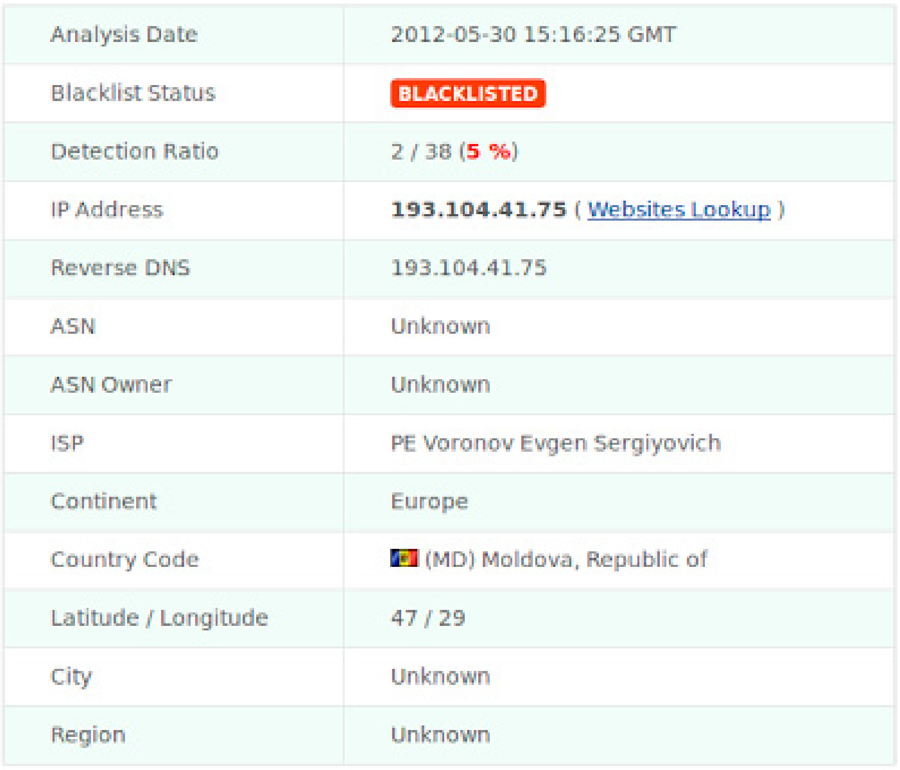
```
vol -f zeus.vmem connscan
```



```
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/*.



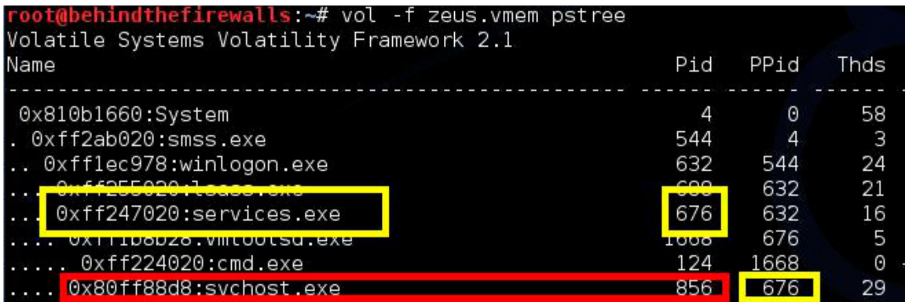
Analysis Date	2012-05-30 15:16:25 GMT
Blacklist Status	BLACKLISTED
Detection Ratio	2 / 38 (5 %)
IP Address	193.104.41.75 (Websites Lookup)
Reverse DNS	193.104.41.75
ASN	Unknown
ASN Owner	Unknown
ISP	PE Voronov Evgen Sergiyovich
Continent	Europe
Country Code	(MD) Moldova, Republic of
Latitude / Longitude	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
... 0xff255020:lsass.exe                              688    632     21
... 0xff247020:services.exe                           676    632     16
.... 0xff1b8b28:vmtoolsd.exe                         1668    676      5
..... 0xff224020:cmd.exe                              124   1668      0
.... 0x80ff88d8:svchost.exe                           856    676     29
```

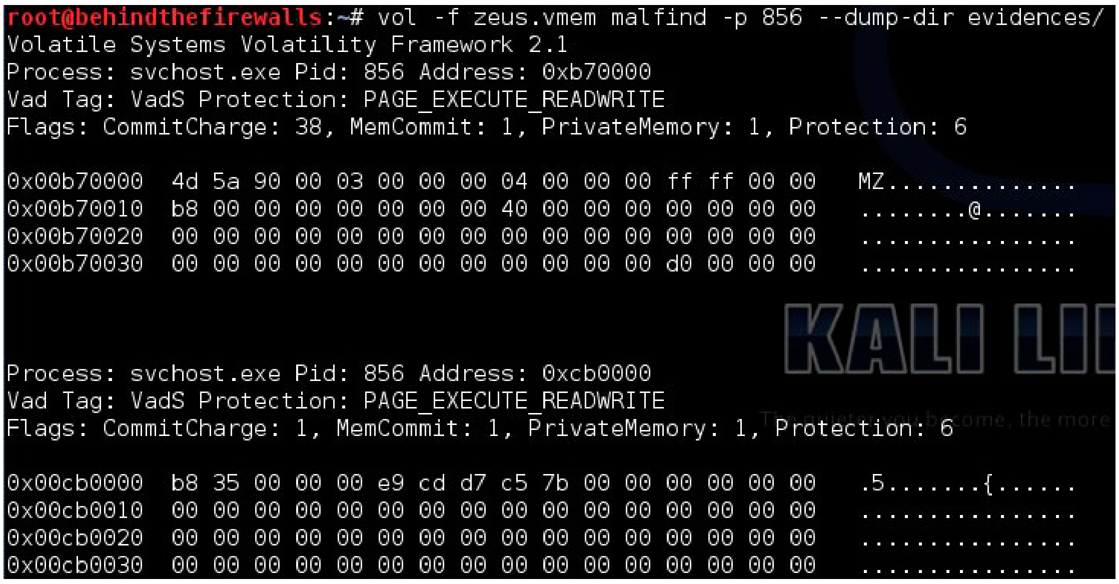
Figure 24. *Suspicious processes*

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   ................
```

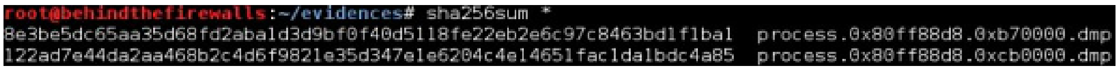
Figure 25. *Injected code*

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 *
```

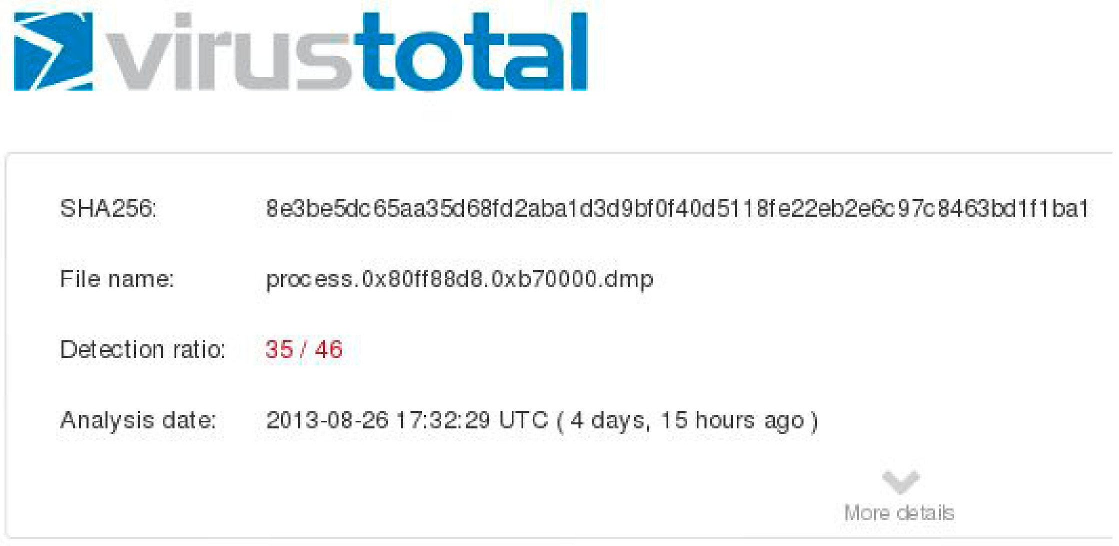


```
root@behindthefirewalls:~/evidences# sha256sum *
8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1  process.0x80ff88d8.0xb70000.dmp
122ad7e44da2aa468b2c4d6f9821e35d347e1e6204c4e14651fac1da1bdc4a85  process.0x80ff88d8.0xcb0000.dmp
```

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.



SHA256: 8e3be5dc65aa35d68fd2aba1d3d9bf0f40d5118fe22eb2e6c97c8463bd1f1ba1

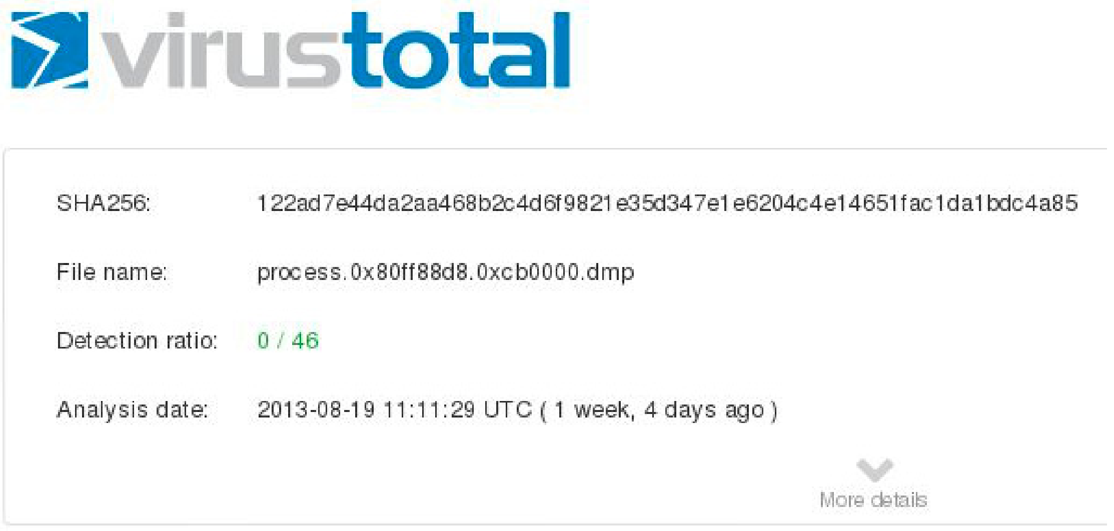
File name: process.0x80ff88d8.0xb70000.dmp

Detection ratio: 35 / 46

Analysis date: 2013-08-26 17:32:29 UTC (4 days, 15 hours ago)

Figure 27. *Dump detect as malware*

The second one seems to don't hold malicious code.



SHA256: 122ad7e44da2aa468b2c4d6f9821e35d347e1e6204c4e14651fac1da1bdc4a85

File name: process.0x80ff88d8.0xcb0000.dmp

Detection ratio: 0 / 46

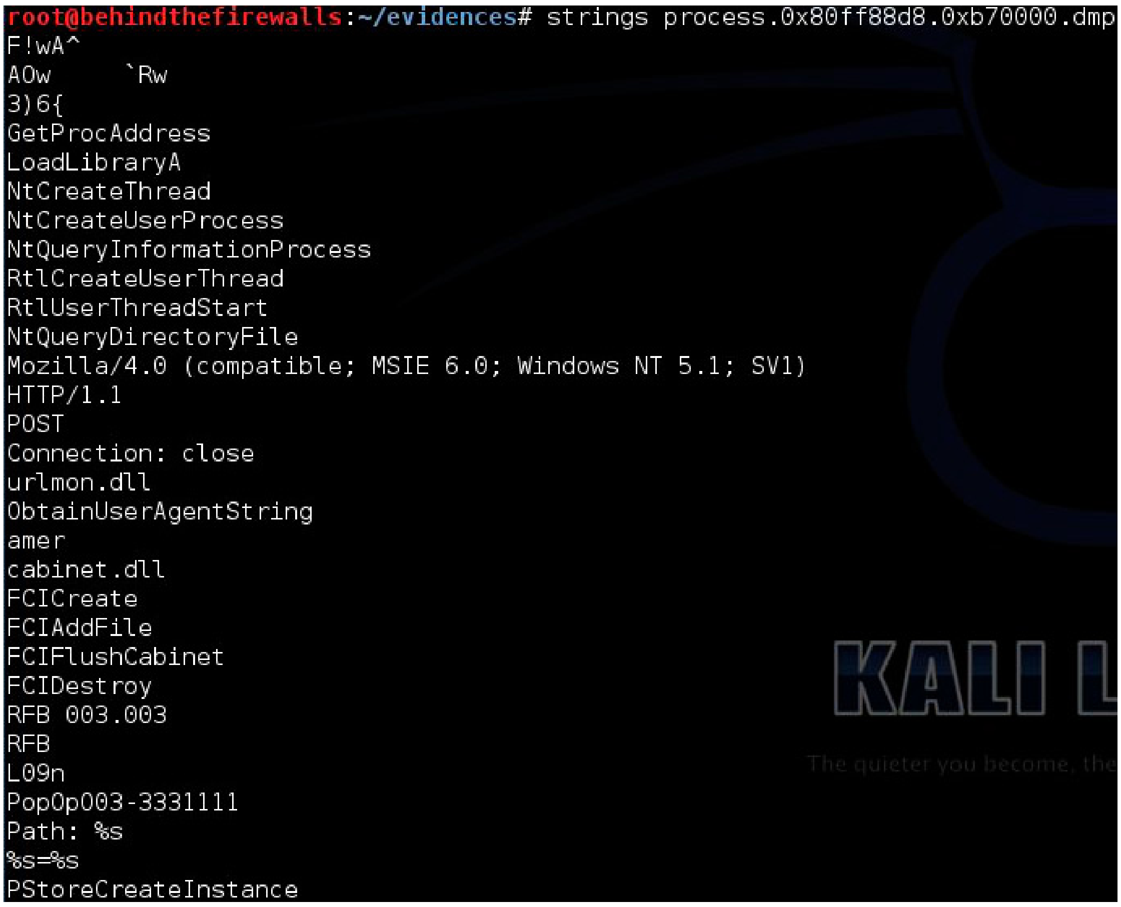
Analysis date: 2013-08-19 11:11:29 UTC (1 week, 4 days ago)

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.

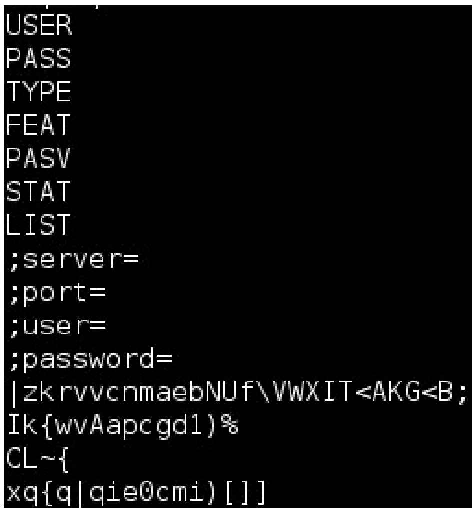
```
strings process.0x80ff88d8.0xb70000.dmp
```



```
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*

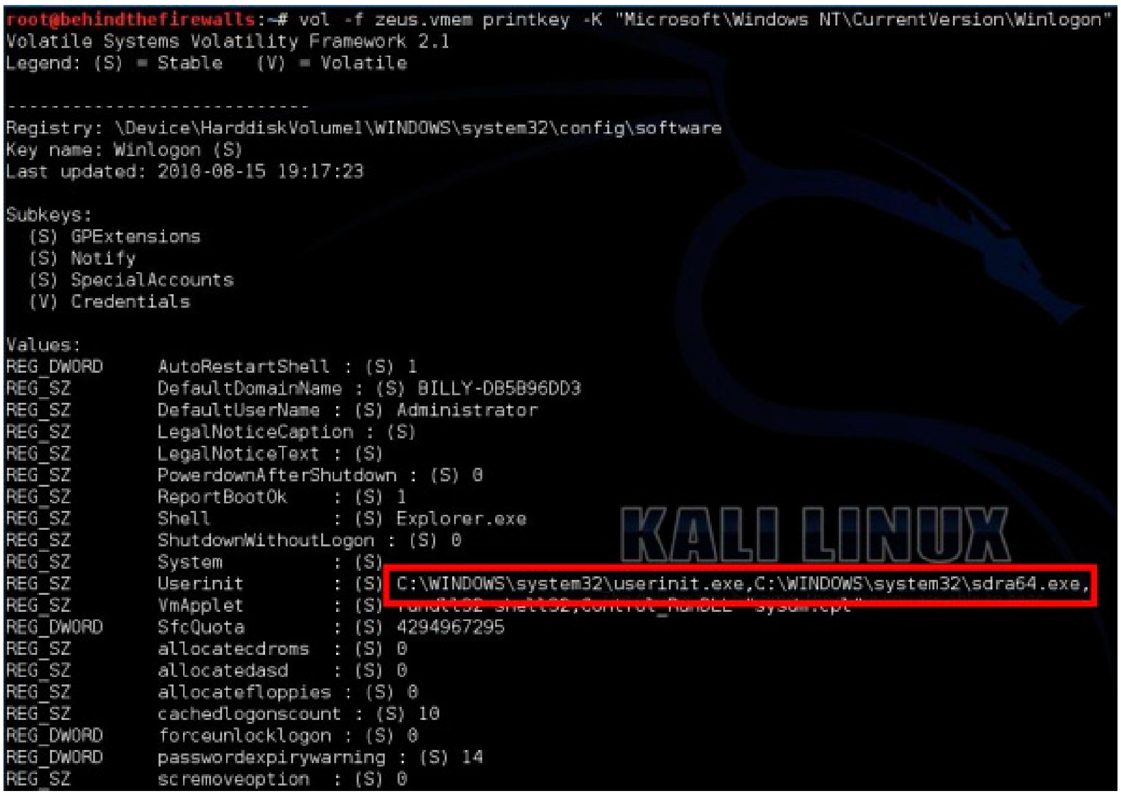
DETECTING REGISTRY KEYS

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""
```



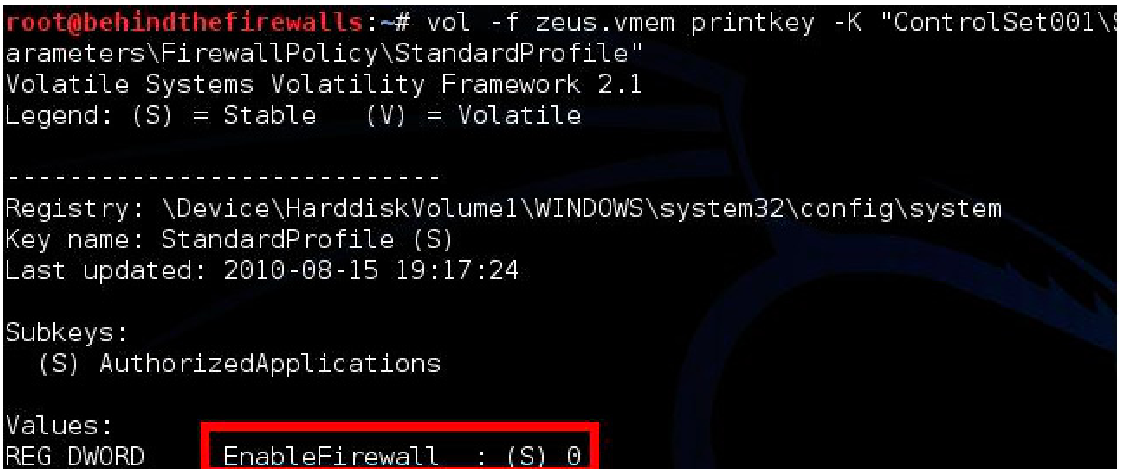
```
root@behindthefirewalls:~# vol -f zeus.vmem printkey -K "Microsoft\Windows NT\CurrentVersion\Winlogon"
Volatile Systems Volatility Framework 2.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\software
Key name: Winlogon (S)
Last updated: 2010-08-15 19:17:23

Subkeys:
  (S) GPExtensions
  (S) Notify
  (S) SpecialAccounts
  (V) Credentials

Values:
REG_DWORD     AutoRestartShell : (S) 1
REG_SZ        DefaultDomainName : (S) BILLY-DB5B96DD3
REG_SZ        DefaultUserName : (S) Administrator
REG_SZ        LegalNoticeCaption : (S)
REG_SZ        LegalNoticeText : (S)
REG_SZ        PowerdownAfterShutdown : (S) 0
REG_SZ        ReportBootOk     : (S) 1
REG_SZ        Shell            : (S) Explorer.exe
REG_SZ        ShutdownWithoutLogon : (S) 0
REG_SZ        System           : (S)
REG_SZ        Userinit         : (S) C:\WINDOWS\system32\userinit.exe,C:\WINDOWS\system32\sdra64.exe,
REG_SZ        VmApplet         : (S) rundll32 shell32,Control_RunDLL "sysdm.cpl"
REG_DWORD     SfcQuota         : (S) 4294967295
REG_SZ        allocatecdroms   : (S) 0
REG_SZ        allocatedasd     : (S) 0
REG_SZ        allocatefloppies : (S) 0
REG_SZ        cachedlogonscount : (S) 10
REG_DWORD     forceunlocklogon : (S) 0
REG_DWORD     passwordexpirywarning : (S) 14
REG_SZ        scremoveoption   : (S) 0
```

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"
```



```
root@behindthefirewalls:~# vol -f zeus.vmem printkey -K "ControlSet001\S
arameters\FirewallPolicy\StandardProfile"
Volatile Systems Volatility Framework 2.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\system
Key name: StandardProfile (S)
Last updated: 2010-08-15 19:17:24

Subkeys:
  (S) AuthorizedApplications

Values:
REG_DWORD     EnableFirewall  : (S) 0
```

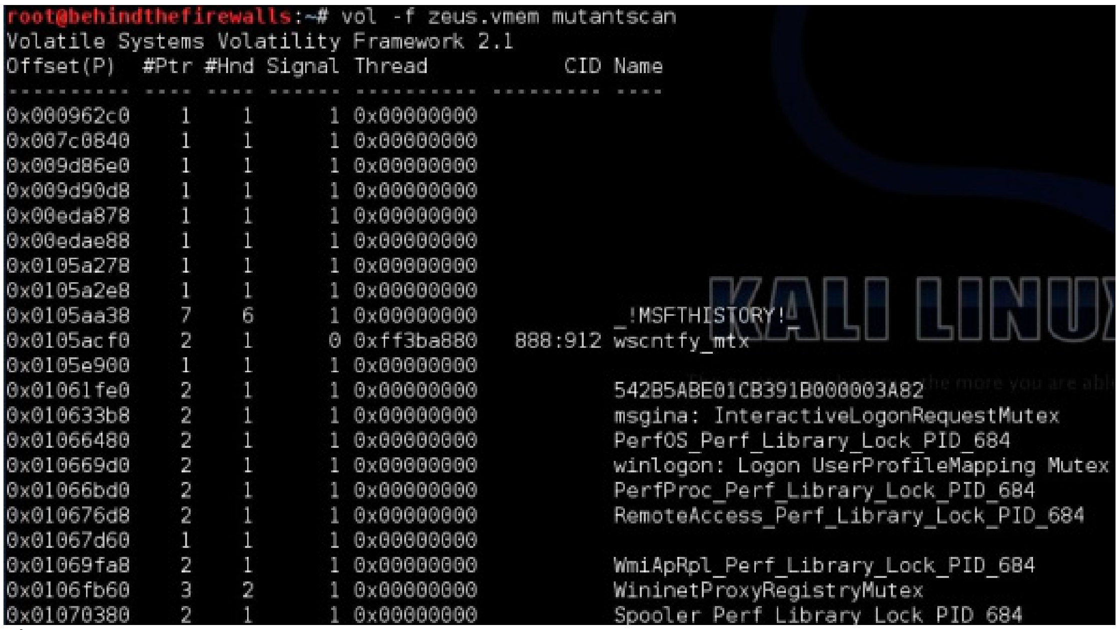
Figure 32. *Firewall Registry key value*

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
```



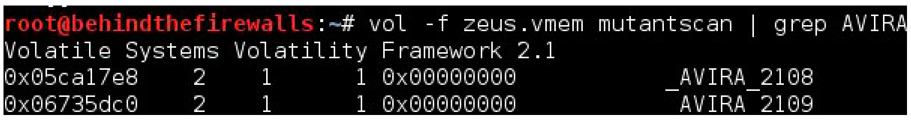
```
root@behindthefirewalls:~# vol -f zeus.vmem mutantscan
Volatile Systems Volatility Framework 2.1
Offset(P)  #Ptr #Hnd Signal Thread           CID Name
---------- ---- ---- ------ ---------- --------- ----
0x000962c0    1    1      1 0x00000000
0x007c0840    1    1      1 0x00000000
0x009d86e0    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.

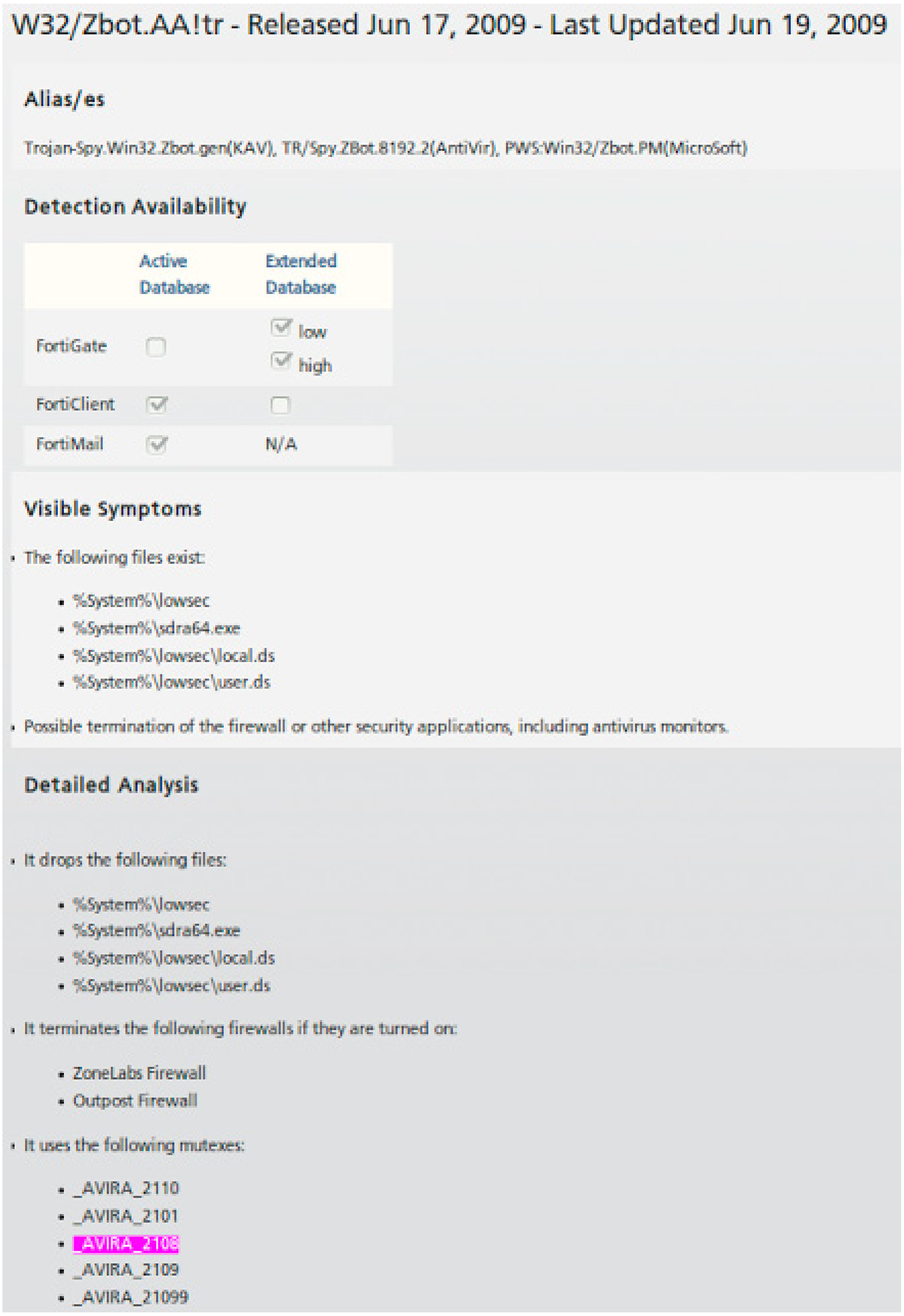
```
vol -f  zeus.vmem mutantscan | grep AVIRA
```



```
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*.



W32/Zbot.AA!tr - Released Jun 17, 2009 - Last Updated Jun 19, 2009

Alias/es

Trojan-Spy.Win32.Zbot.gen(KAV), TR/Spy.ZBot.8192.2(AntiVir), PWS:Win32/Zbot.PM(MicroSoft)

Detection Availability

	Active Database	Extended Database
FortiGate	☐	✓ low ✓ high
FortiClient	✓	☐
FortiMail	✓	N/A

Visible Symptoms

- The following files exist:
  - %System%\lowsec
  - %System%\sdra64.exe
  - %System%\lowsec\local.ds
  - %System%\lowsec\user.ds
- Possible termination of the firewall or other security applications, including antivirus monitors.

Detailed Analysis

- It drops the following files:
  - %System%\lowsec
  - %System%\sdra64.exe
  - %System%\lowsec\local.ds
  - %System%\lowsec\user.ds
- It terminates the following firewalls if they are turned on:
  - ZoneLabs Firewall
  - Outpost Firewall
- It uses the following mutexes:
  - _AVIRA_2110
  - _AVIRA_2101
  - _AVIRA_2108
  - _AVIRA_2109
  - _AVIRA_21099

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.

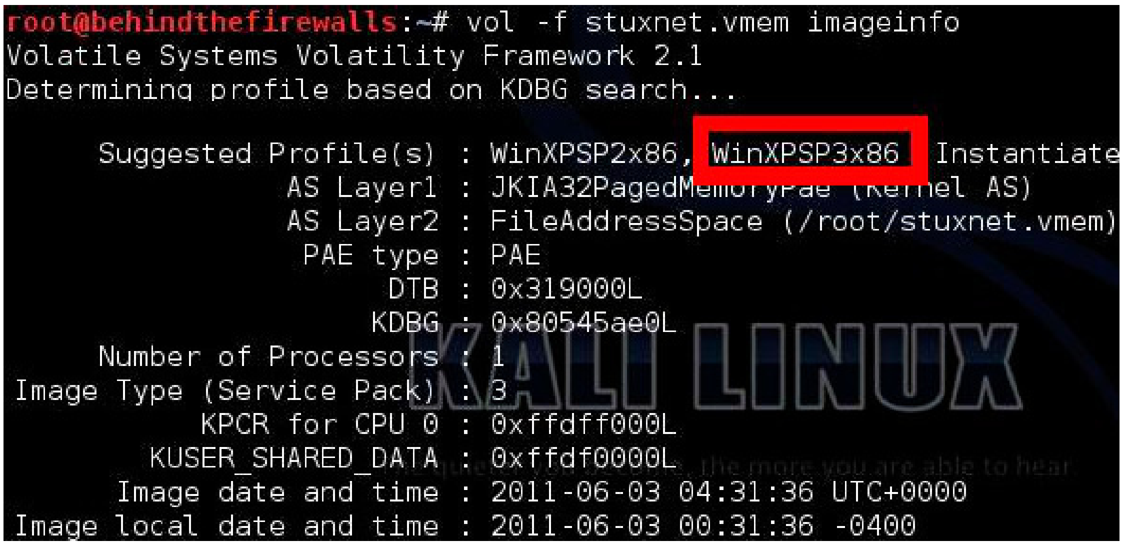
Thanks Malware Cookbook we can download a memory dump from an infected host with this malware in the URL: *http://malwarecookbook.googlecode.com/svn/trunk/stuxnet.vmem.zip*.

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
```

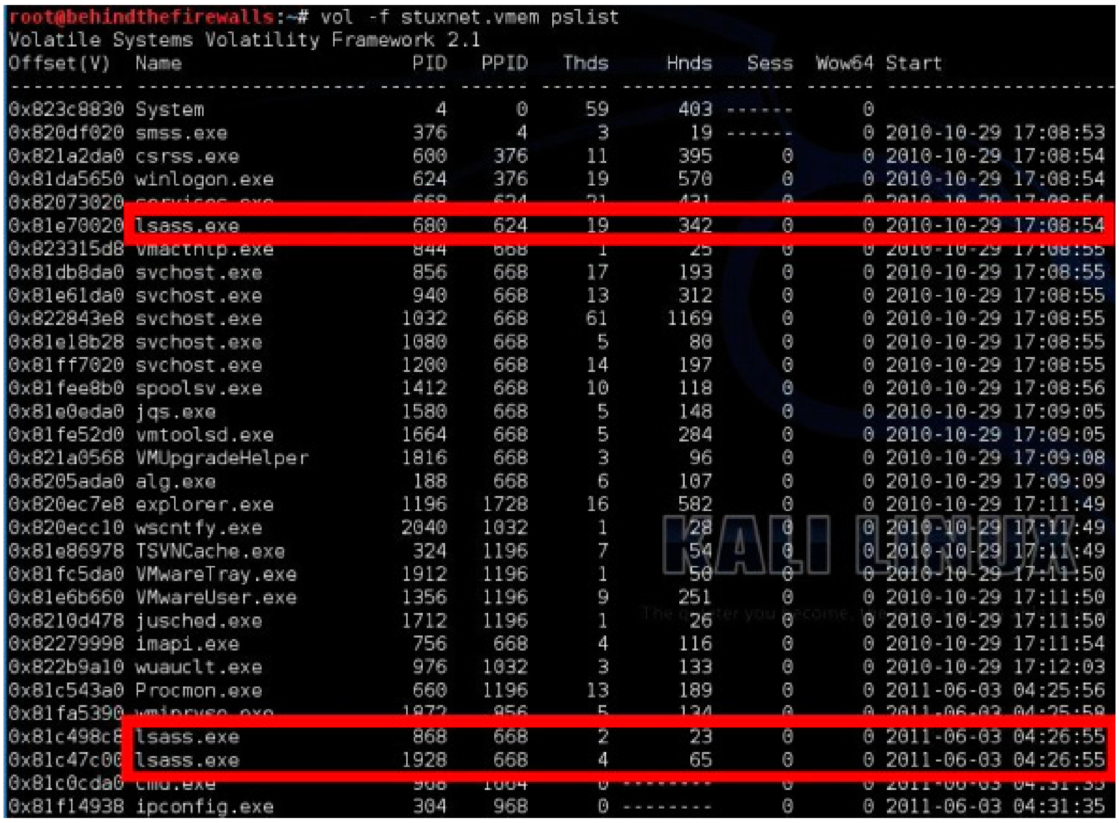
DETECTING MALICIOIUS PROCCESS

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
0x81f14938 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:

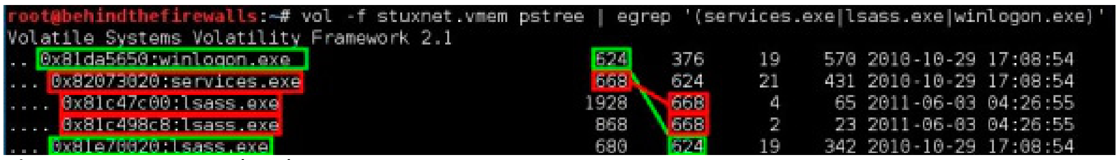
- Pid 680 started at 2010-10-29 17:08:54
- Pid 868 started at 2011-06-03 04:26:55
- Pid 1928 started at 2011-06-03 04:26:55

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
```

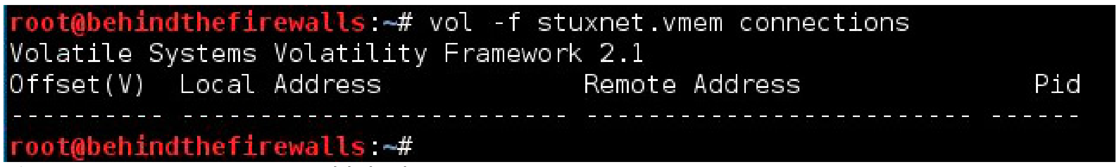
Figure 38. *Lssas.exe details*

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:~#
```

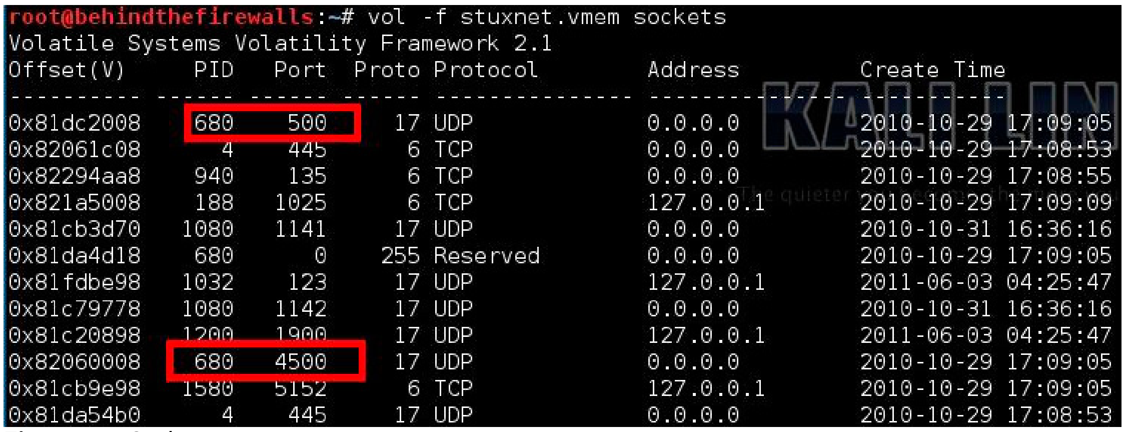
Figure 39. *No connection established*

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
```

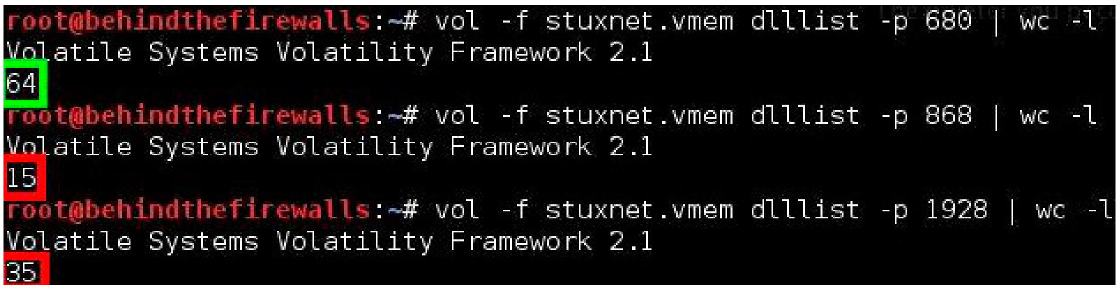
Figure 40. *Sockets*

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
```

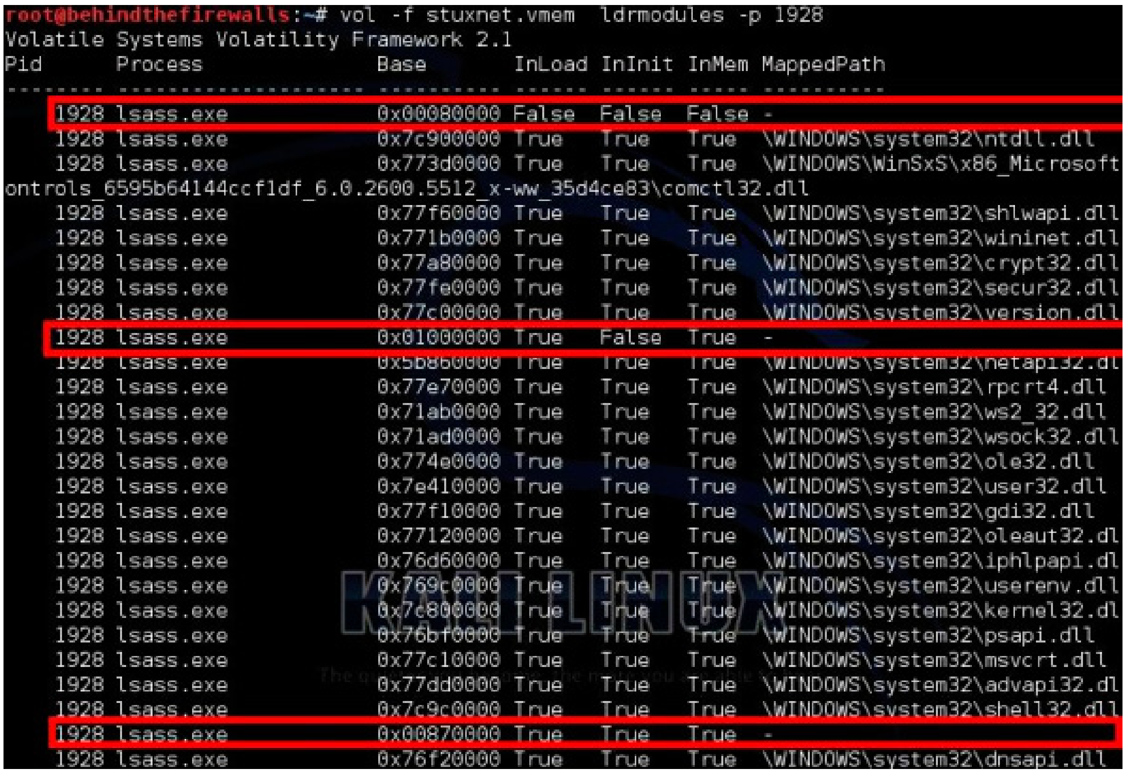


```
root@behindthefirewalls:~# vol -f stuxnet.vmem dlllist -p 680 | wc -l
Volatile Systems Volatility Framework 2.1
64
root@behindthefirewalls:~# vol -f stuxnet.vmem dlllist -p 868 | wc -l
Volatile Systems Volatility Framework 2.1
15
root@behindthefirewalls:~# vol -f stuxnet.vmem dlllist -p 1928 | wc -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
```

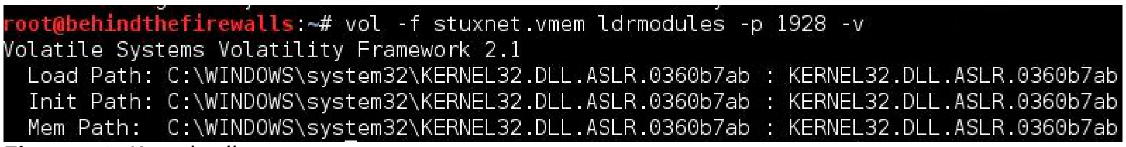


```
root@behindthefirewalls:~# vol -f stuxnet.vmem  ldrmodules -p 1928
Volatile Systems Volatility Framework 2.1
Pid      Process              Base       InLoad InInit InMem MappedPath
-------- -------------------- ---------- ------ ------ ----- ----------
    1928 lsass.exe            0x00080000 False  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*.

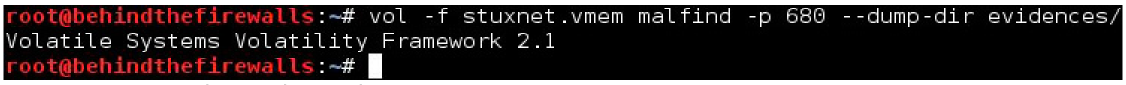
DETECTING INJECTED CODE

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/
```

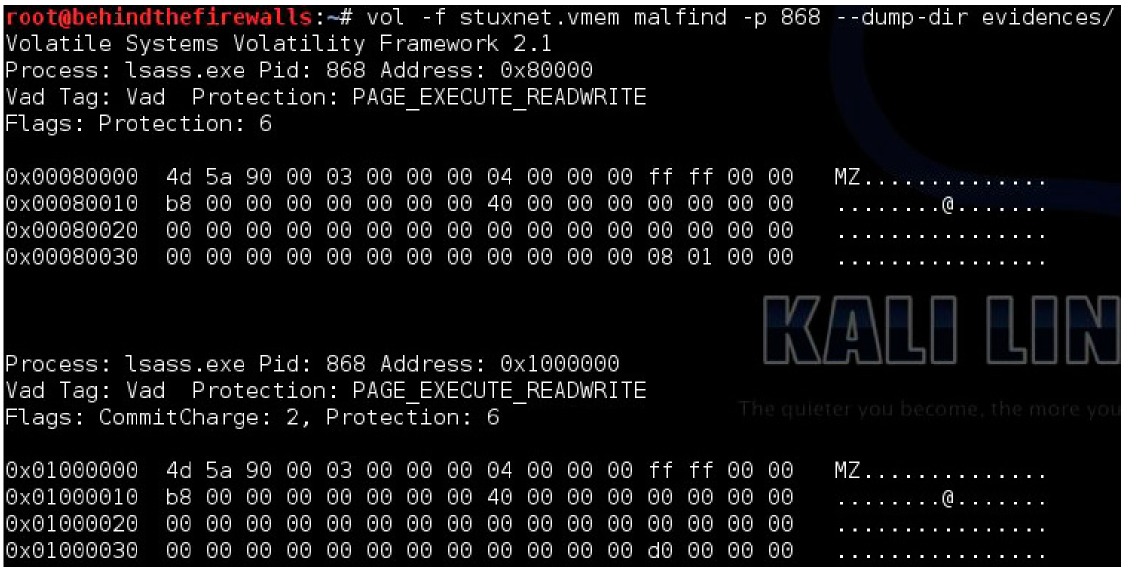


```
root@behindthefirewalls:~# vol -f stuxnet.vmem malfind -p 680 --dump-dir evidences/
Volatile Systems Volatility Framework 2.1
root@behindthefirewalls:~#
```

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/
```

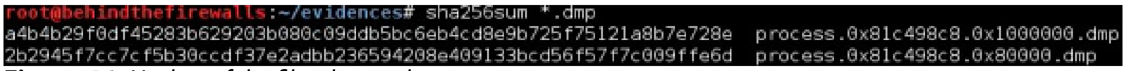


```
root@behindthefirewalls:~# vol -f stuxnet.vmem malfind -p 868 --dump-dir evidences/
Volatile Systems Volatility Framework 2.1
Process: lsass.exe Pid: 868 Address: 0x80000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

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.

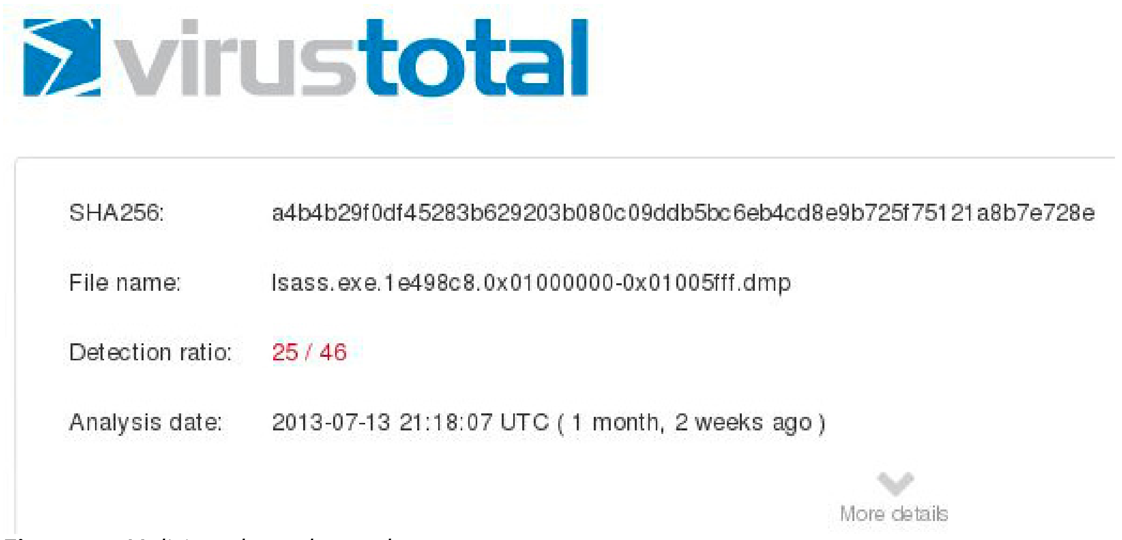
```
Sha256 *.dmp
```



```
root@behindthefirewalls:~/evidences# sha256sum *.dmp
a4b4b29f0df45283b629203b080c09ddb5bc6eb4cd8e9b725f75121a8b7e728e  process.0x81c498c8.0x1000000.dmp
2b2945f7cc7cf5b30ccdf37e2adbb236594208e409133bcd56f57f7c009ffe6d  process.0x81c498c8.0x80000.dmp
```

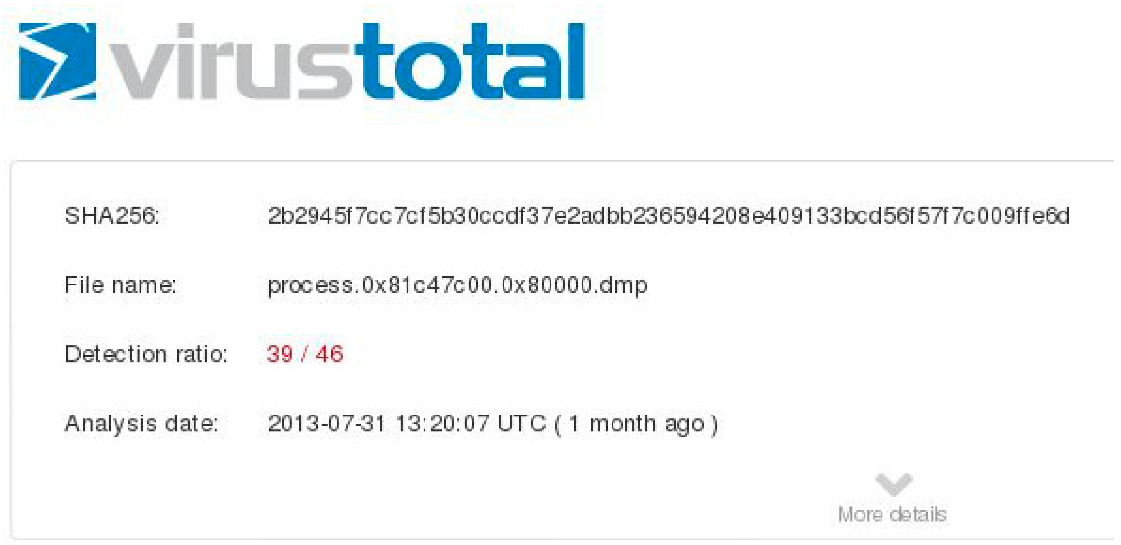
Figure 46. *Hashes of the files dumped*

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*.



SHA256:	a4b4b29f0df45283b629203b080c09ddb5bc6eb4cd8e9b725f75121a8b7e728e
File name:	lsass.exe.1e498c8.0x01000000-0x01005fff.dmp
Detection ratio:	25 / 46
Analysis date:	2013-07-13 21:18:07 UTC (1 month, 2 weeks ago)

Figure 47. *Malicious dump detected*



SHA256: 2b2945f7cc7cf5b30ccdf37e2adbb236594208e409133bcd56f57f7c009ffe6d

File name: process.0x81c47c00.0x80000.dmp

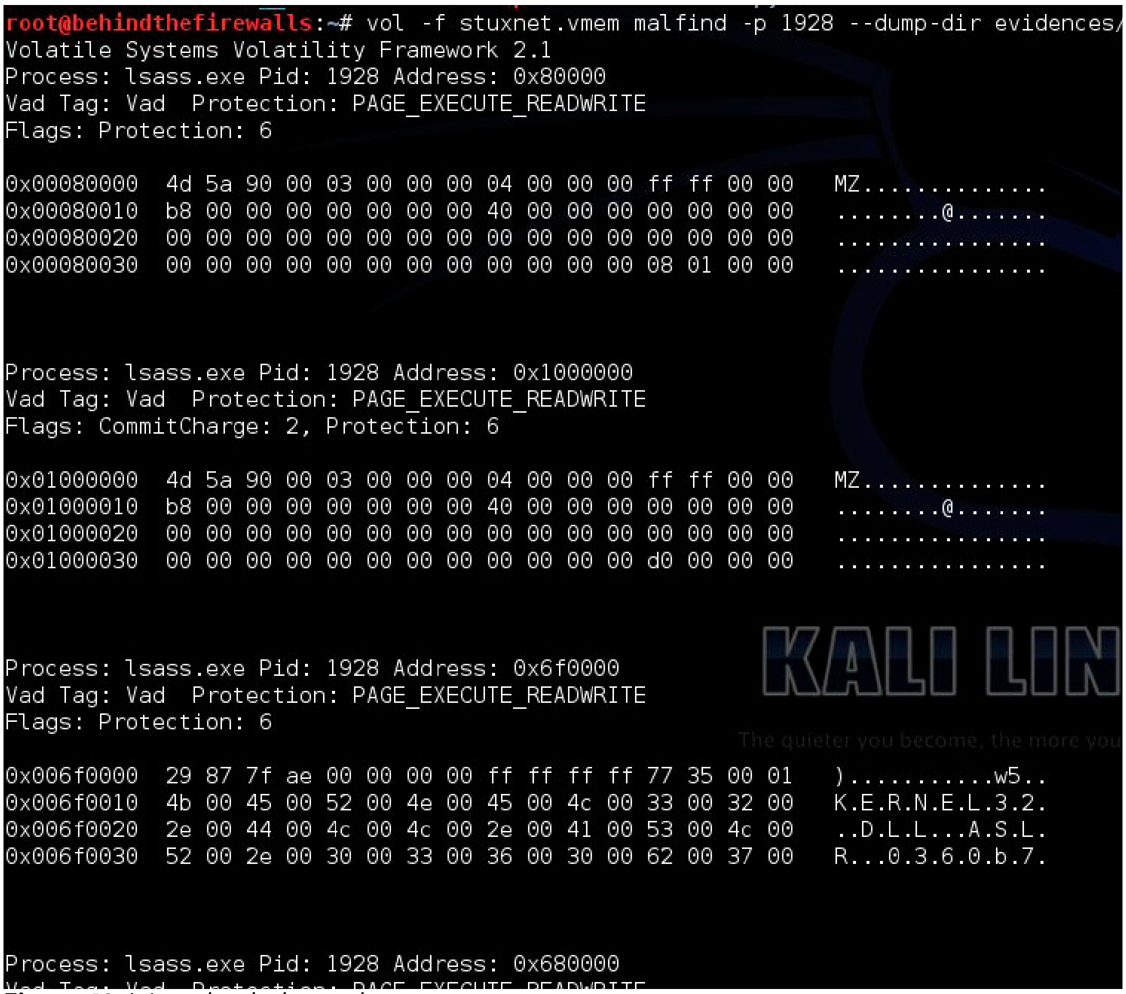
Detection ratio: 39 / 46

Analysis date: 2013-07-31 13:20:07 UTC (1 month ago)

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/
```

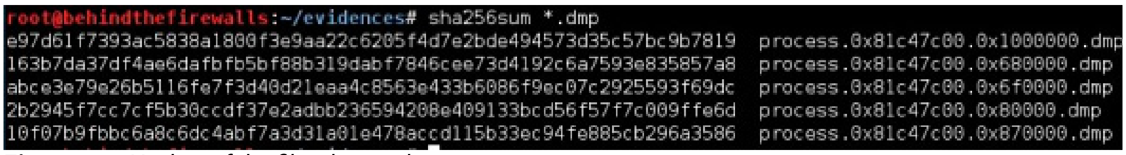


```
root@behindthefirewalls:~# vol -f stuxnet.vmem malfind -p 1928 --dump-dir evidences/
Volatile Systems Volatility Framework 2.1
Process: lsass.exe Pid: 1928 Address: 0x80000
Vad Tag: Vad  Protection: PAGE_EXECUTE_READWRITE
Flags: Protection: 6

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
```

Figure 49. *Injected code detected*

Now, we are going to get the sha256 checksums.

```
Sha256sum *.dmp
```

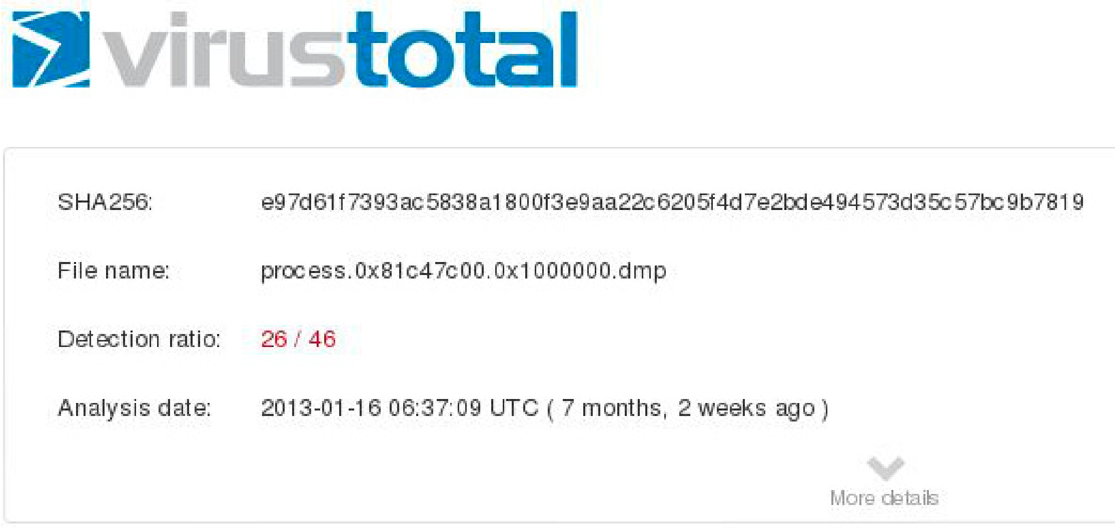


```
root@behindthefirewalls:~/evidences# sha256sum *.dmp
e97d61f7393ac5838a1800f3e9aa22c6205f4d7e2bde494573d35c57bc9b7819  process.0x81c47c00.0x1000000.dmp
163b7da37df4ae6dafbfb5bf88b319dabf7846cee73d4192c6a7593e835857a8  process.0x81c47c00.0x680000.dmp
abce3e79e26b5116fe7f3d40d21eaa4c8563e433b6086f9ec07c2925593f69dc  process.0x81c47c00.0x6f0000.dmp
2b2945f7cc7cf5b30ccdf37e2adbb236594208e409133bcd56f57f7c009ffe6d  process.0x81c47c00.0x80000.dmp
10f07b9fbbc6a8c6dc4abf7a3d31a01e478accd115b33ec94fe885cb296a3586  process.0x81c47c00.0x870000.dmp
```

Figure 50. *Hashes of the files dumped*

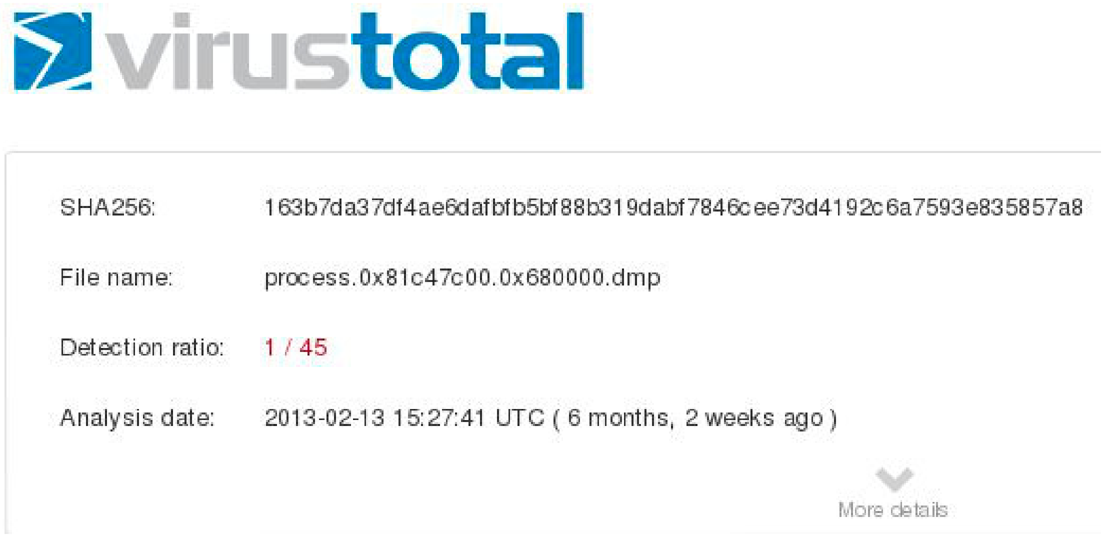
Then, we are going to search this hashes in the same URL: *https://www.virustotal.com/es/#search*.

Notice in the Figure 54, the file is classified as Stuxnet worm.



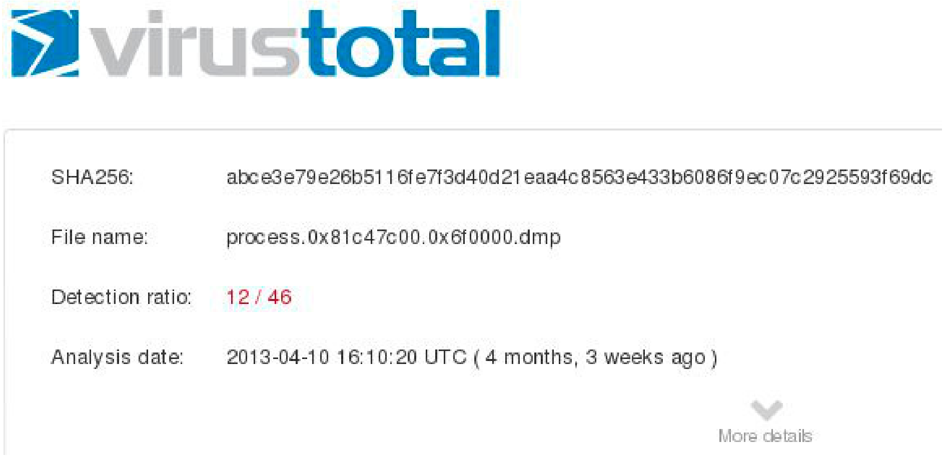
SHA256:	e97d61f7393ac5838a1800f3e9aa22c6205f4d7e2bde494573d35c57bc9b7819
File name:	process.0x81c47c00.0x1000000.dmp
Detection ratio:	26 / 46
Analysis date:	2013-01-16 06:37:09 UTC (7 months, 2 weeks ago)

Figure 51. *Malicious dump detected*



SHA256:	163b7da37df4ae6dafbfb5bf88b319dabf7846cee73d4192c6a7593e835857a8
File name:	process.0x81c47c00.0x680000.dmp
Detection ratio:	1 / 45
Analysis date:	2013-02-13 15:27:41 UTC (6 months, 2 weeks ago)

Figure 52. *Malicious dump detected*



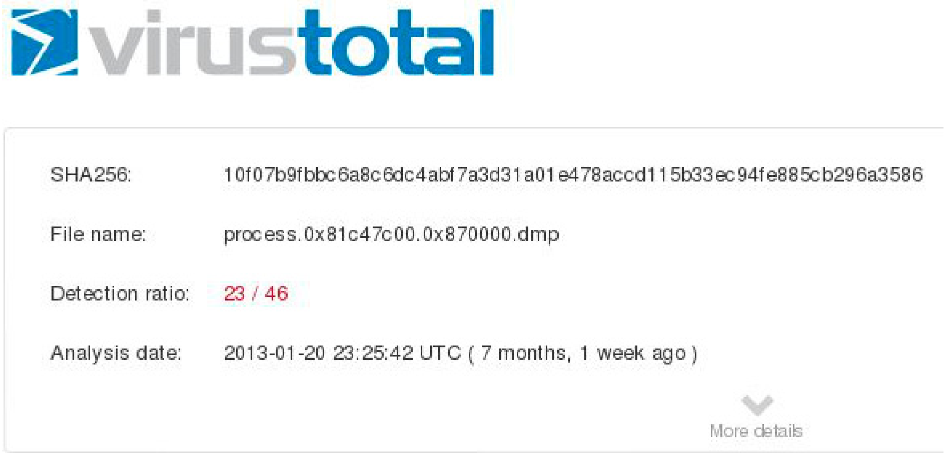
SHA256: abce3e79e26b5116fe7f3d40d21eaa4c8563e433b6086f9ec07c2925593f69dc

File name: process.0x81c47c00.0x6f0000.dmp

Detection ratio: 12 / 46

Analysis date: 2013-04-10 16:10:20 UTC (4 months, 3 weeks ago)

Figure 53. *Malicious dump detected*



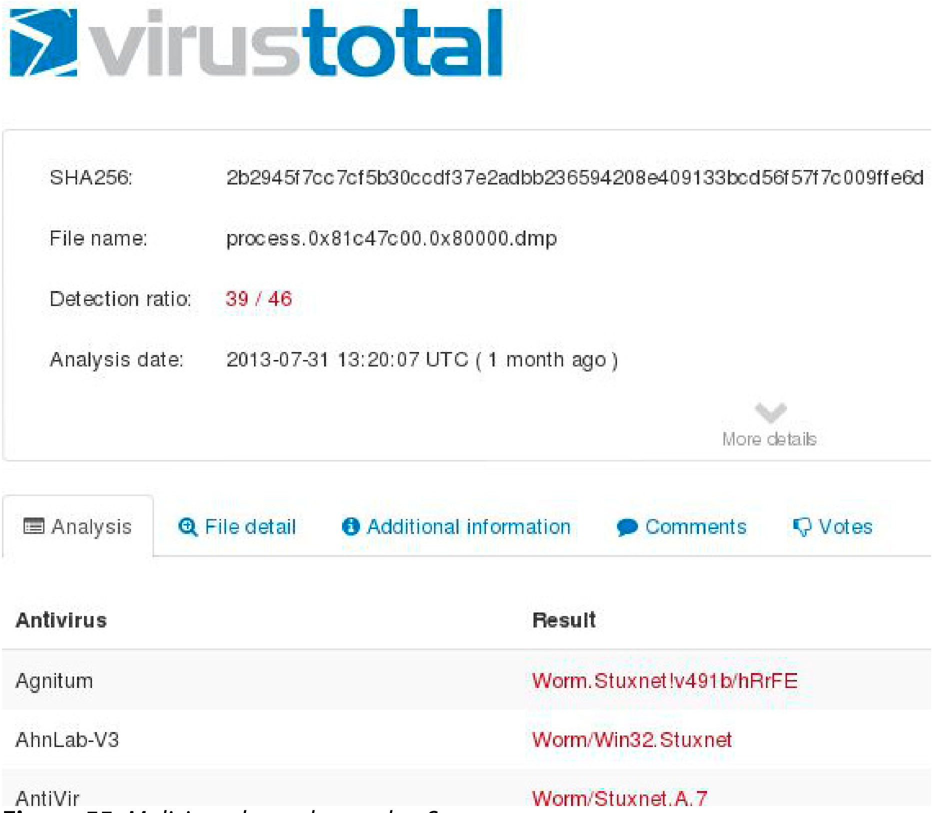
SHA256: 10f07b9fbbc6a8c6dc4abf7a3d31a01e478accd115b33ec94fe885cb296a3586

File name: process.0x81c47c00.0x870000.dmp

Detection ratio: 23 / 46

Analysis date: 2013-01-20 23:25:42 UTC (7 months, 1 week ago)

Figure 54. *Malicious dump detected*



SHA256: 2b2945f7cc7cf5b30ccdf37e2adbb236594208e409133bcd56f57f7c009ffe6d

File name: process.0x81c47c00.0x80000.dmp

Detection ratio: 39 / 46

Analysis date: 2013-07-31 13:20:07 UTC (1 month ago)

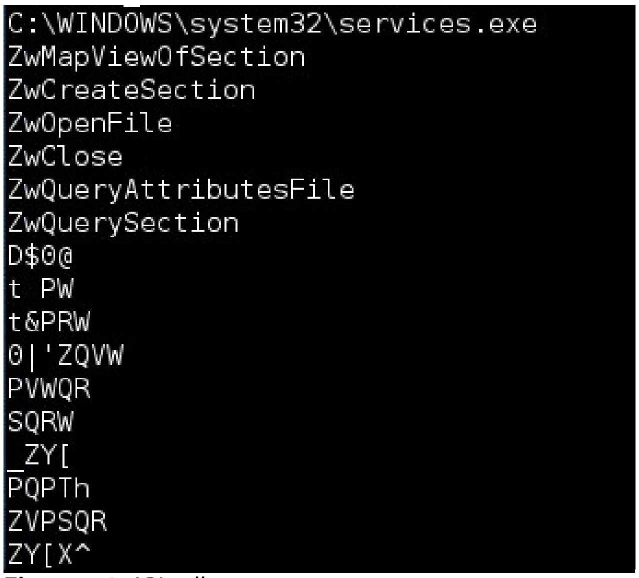
Antivirus	Result
Agnitum	Worm.Stuxnet!v491b/hRrFE
AhnLab-V3	Worm/Win32.Stuxnet
AntiVir	Worm/Stuxnet.A.7

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.*
```

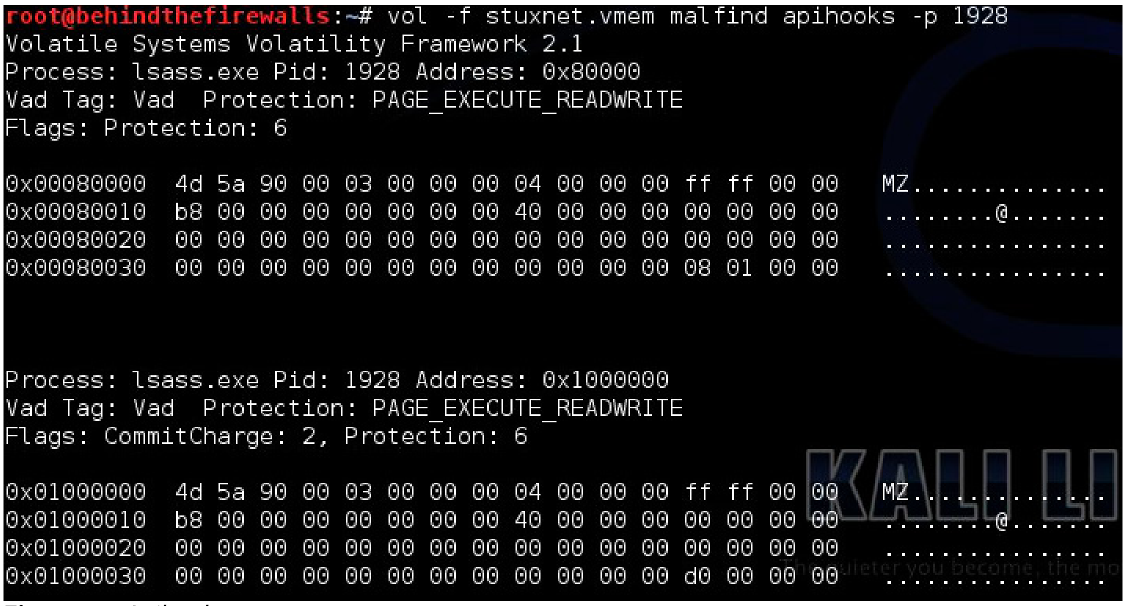


```
C:\WINDOWS\system32\services.exe
ZwMapViewOfSection
ZwCreateSection
ZwOpenFile
ZwClose
ZwQueryAttributesFile
ZwQuerySection
D$0@
t PW
t&PRW
0|'ZQVW
PVWQR
SQRW
_ZY[
PQPTh
ZVPSQR
ZY[X^
```

Figure 56. *API calls*

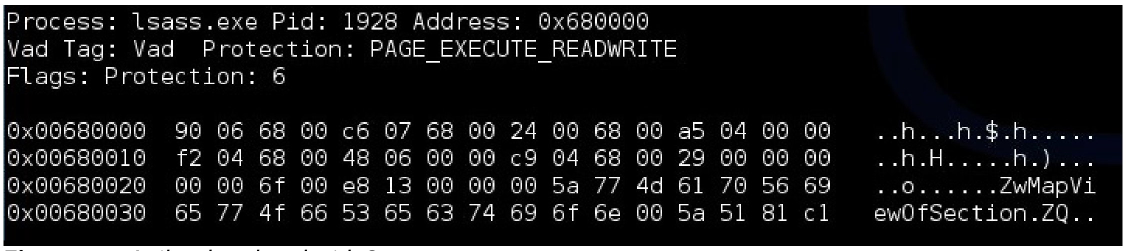
We can get the same information thank Volatility. We can use the command below:

```
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..
```

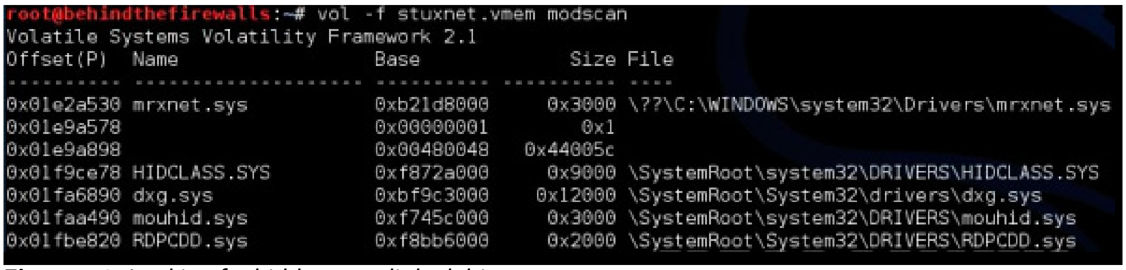
Figure 58. *Apihooks related with Stuxnet*

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.

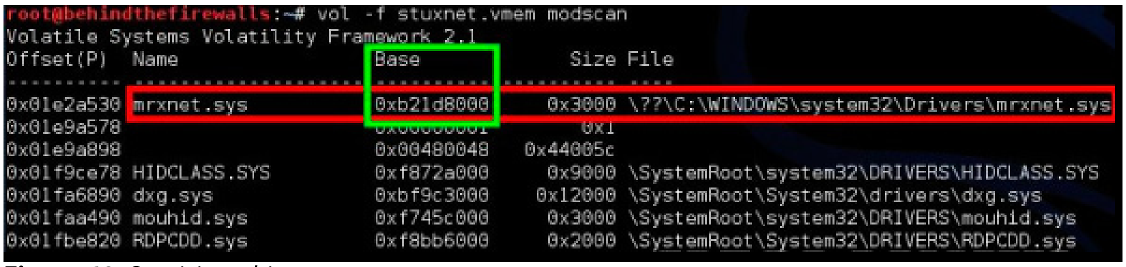
```
vol -f stuxnet.vmem modscan.
```



```
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.

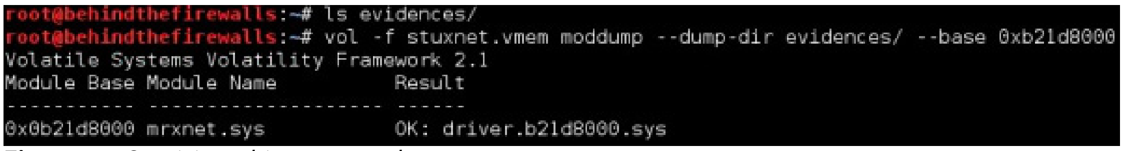


```
root@behindthefirewalls:~# vol -f stuxnet.vmem modscan
Volatile Systems Volatility Framework 2.1
Offset(P)  Name                 Base            Size File
---------- -------------------- ---------- ---------- ----
0x01e2a530 mrxnet.sys           0xb21d8000     0x3000 \??\C:\WINDOWS\system32\Drivers\mrxnet.sys
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
```

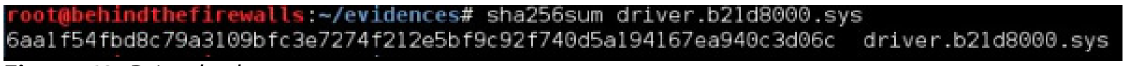


```
root@behindthefirewalls:~# ls evidences/
root@behindthefirewalls:~# vol -f stuxnet.vmem moddump --dump-dir evidences/ --base 0xb21d8000
Volatile Systems Volatility Framework 2.1
Module Base Module Name          Result
----------- -------------------- ------
0x0b21d8000 mrxnet.sys           OK: driver.b21d8000.sys
```

Figure 61. *Suspicious driver exported*

Then, we are going to get the sha256 hash of this driver.

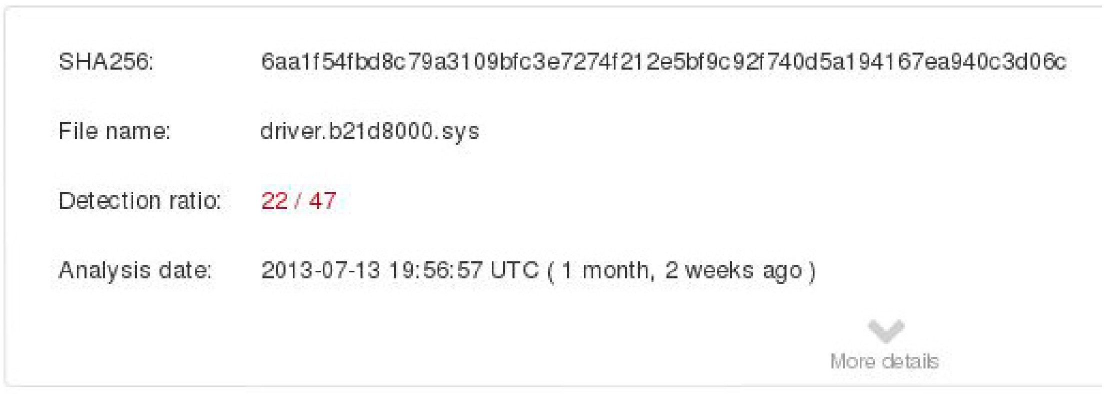
```
sha256sum driver.b21d80000.sys
```



```
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*.

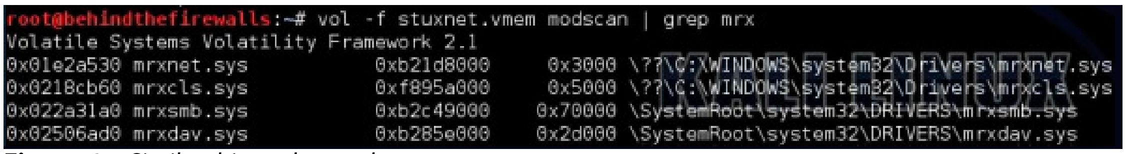


```
SHA256:           6aa1f54fbd8c79a3109bfc3e7274f212e5bf9c92f740d5a194167ea940c3d06c
File name:        driver.b21d8000.sys
Detection ratio:  22 / 47
Analysis date:    2013-07-13 19:56:57 UTC ( 1 month, 2 weeks ago )
```

Figure 63. *Malware detected as malicious*

Ok. I think that it's necessary to look more malicious drivers with a similar name.

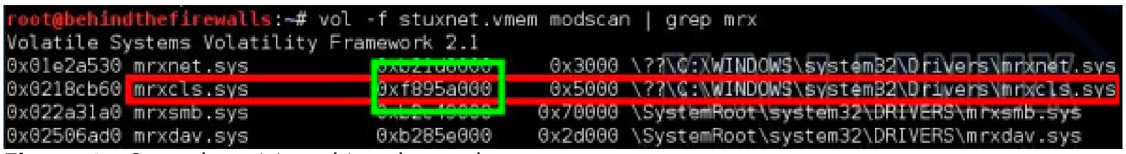
```
vol -f stuxnet.vmem modscan | grep mrx
```



```
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*

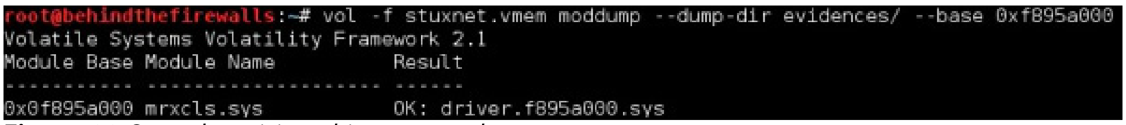
Ok. Let's go to export the second suspicious driver and follow the same sequence.



```
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
```

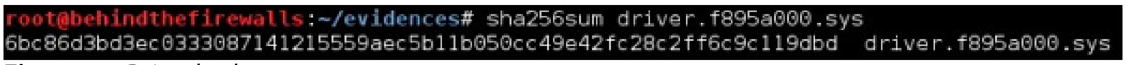


```
root@behindthefirewalls:~# vol -f stuxnet.vmem moddump --dump-dir evidences/ --base 0xf895a000
Volatile Systems Volatility Framework 2.1
Module Base Module Name          Result
----------- -------------------- ------
0x0f895a000 mrxcls.sys           OK: driver.f895a000.sys
```

Figure 66. *Second suspicious driver exported*

Now we are going to get the SHA256 hash.

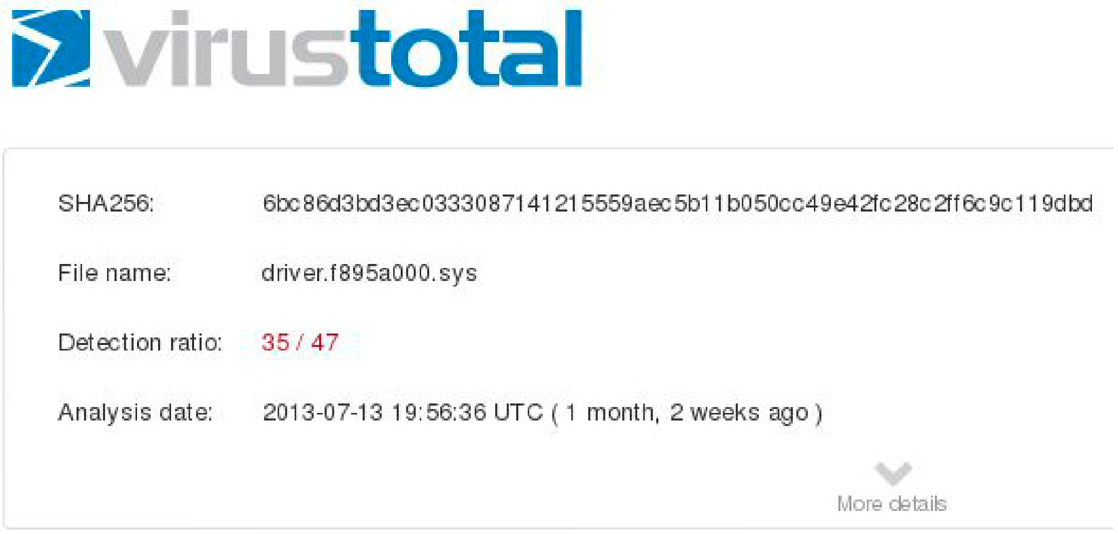
```
sha256sum driver.f895a000.sys
```



```
root@behindthefirewalls:~/evidences# sha256sum driver.f895a000.sys
6bc86d3bd3ec0333087141215559aec5b11b050cc49e42fc28c2ff6c9c119dbd  driver.f895a000.sys
```

Figure 67. *Driver hash*

Finally we are going to check out with Virustotal.


virustotal

SHA256: 6bc86d3bd3ec0333087141215559aec5b11b050cc49e42fc28c2ff6c9c119dbd

File name: driver.f895a000.sys

Detection ratio: 35 / 47

Analysis date: 2013-07-13 19:56:36 UTC (1 month, 2 weeks ago)

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
```

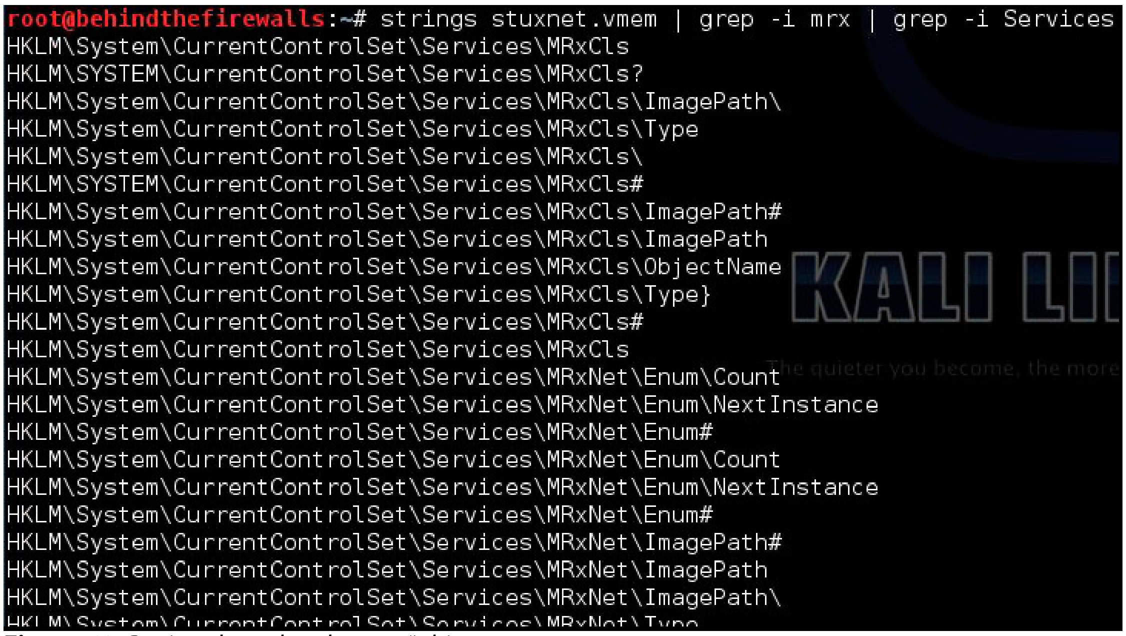
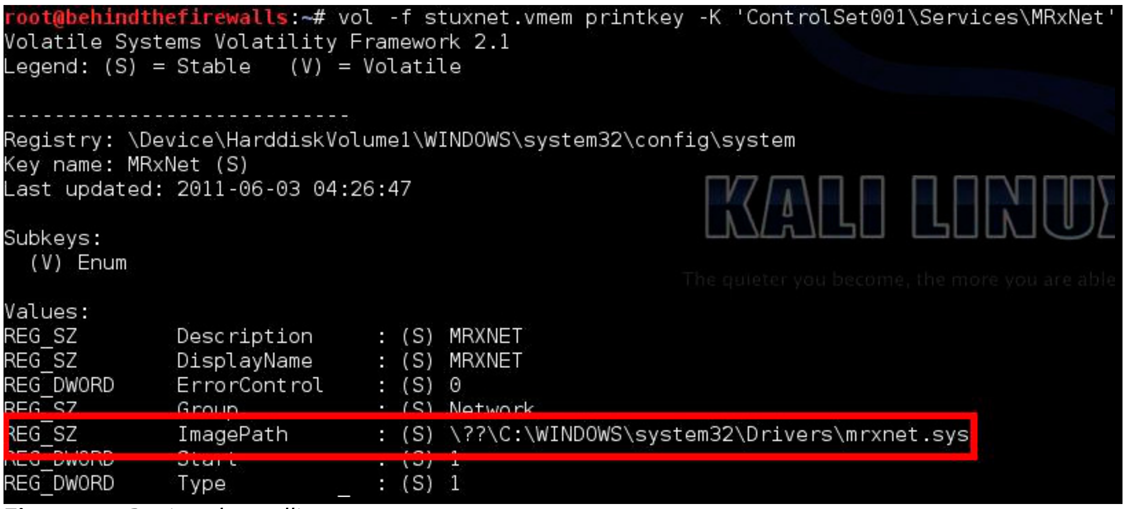

```
root@behindthefirewalls:~# strings stuxnet.vmem | grep -i mrx | grep -i Services
HKLM\System\CurrentControlSet\Services\MRxCls
HKLM\SYSTEM\CurrentControlSet\Services\MRxCls?
HKLM\System\CurrentControlSet\Services\MRxCls\ImagePath\
HKLM\System\CurrentControlSet\Services\MRxCls\Type
HKLM\System\CurrentControlSet\Services\MRxCls\
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\
HKLM\System\CurrentControlSet\Services\MRxNet\Type
```

Figure 69. *Registry key related to mrx* drivers*

Now, we are going to see the details of some of them with the next commands.

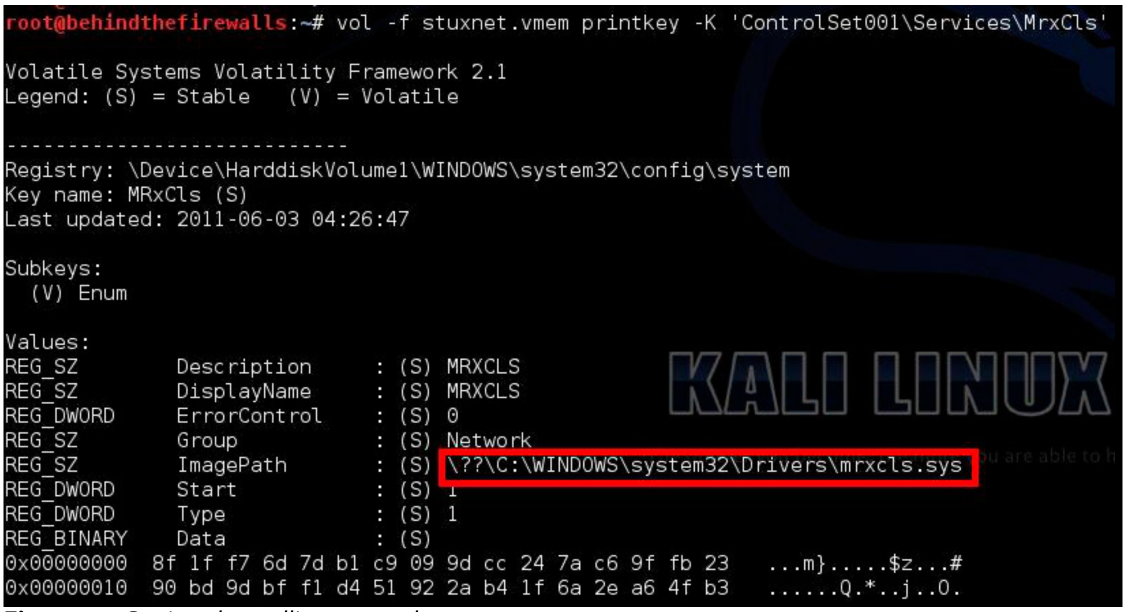
```
vol -f stuxnet.vmem printkey -K 'ControlSet001\Services\MrxNet'
```



```
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

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:

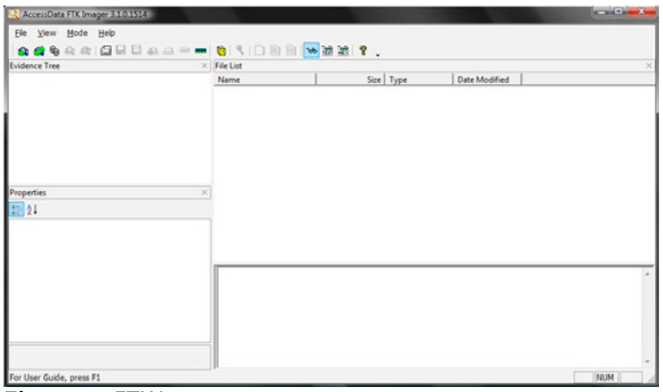
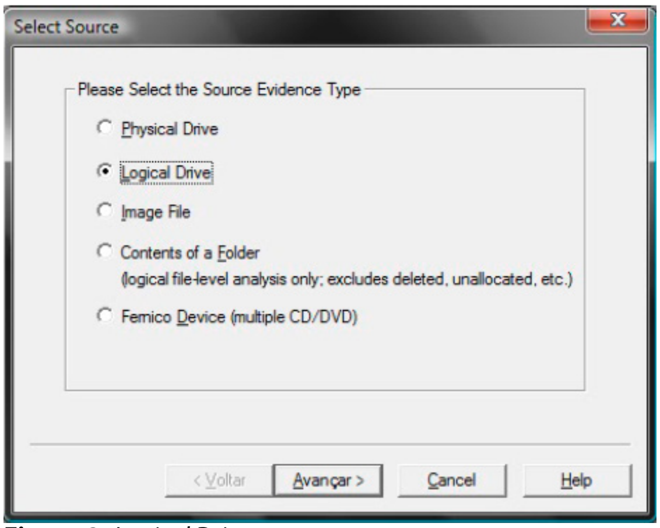


Figure 1. *FTK Imager*

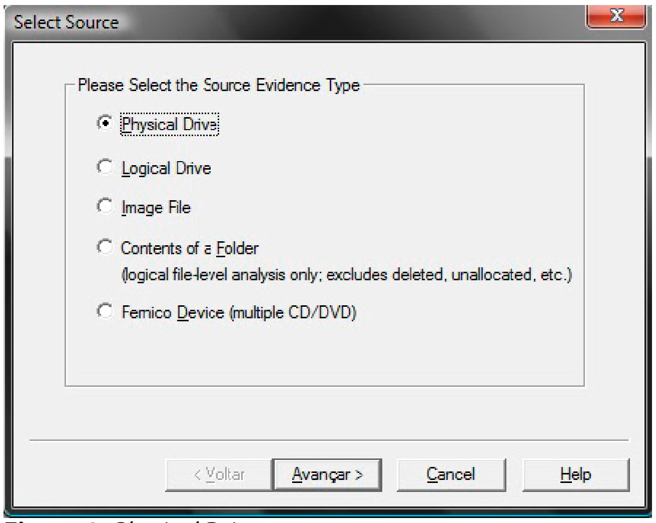


Select Source

Please Select the Source Evidence Type

- Physical Drive
- [✓ Logical Drive]
- Image File
- Contents of a Folder
  (logical file-level analysis only; excludes deleted, unallocated, etc.)
- Femico Device (multiple CD/DVD)

Figure 2. *Logical Drive*



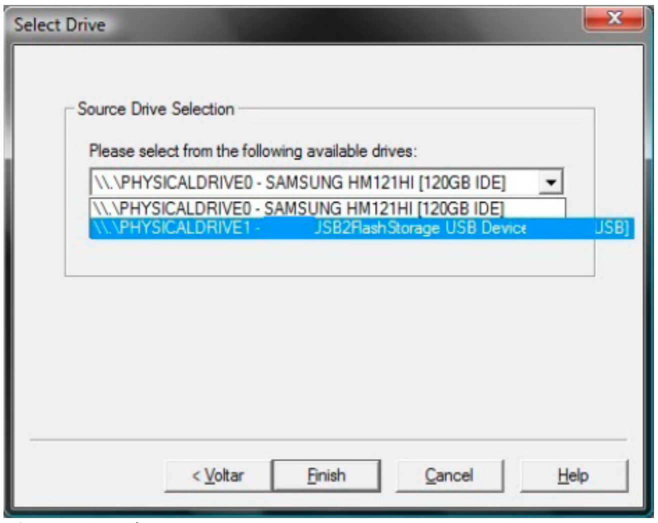
Select Source

Please Select the Source Evidence Type

- [✓ Physical Drive]
- Logical Drive
- Image File
- Contents of a Folder
  (logical file-level analysis only; excludes deleted, unallocated, etc.)
- Femico Device (multiple CD/DVD)

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.



Select Drive

Source Drive Selection

Please select from the following available drives:

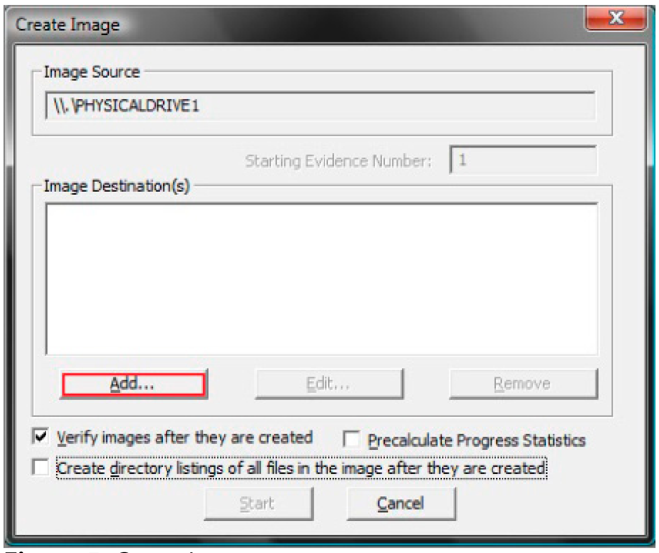
\\.\PHYSICALDRIVE0 - SAMSUNG HM121HI [120GB IDE]

\\.\PHYSICALDRIVE0 - SAMSUNG HM121HI [120GB IDE]

\\.\PHYSICALDRIVE1 - JSB2FlashStorage USB Device JSB]

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.



Create Image

Image Source

\\.\PHYSICALDRIVE1

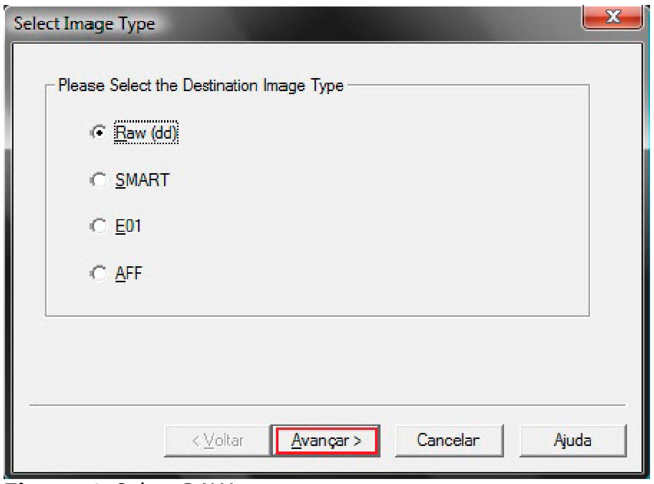
Starting Evidence Number: 1

Image Destination(s)

- [✓] Verify images after they are created
- [ ] Precalculate Progress Statistics
- [ ] Create directory listings of all files in the image after they are created

Figure 5. *Create Image*

Let's select "RAW", to perform forensic image format which is the tool of "DD" and click "Next".



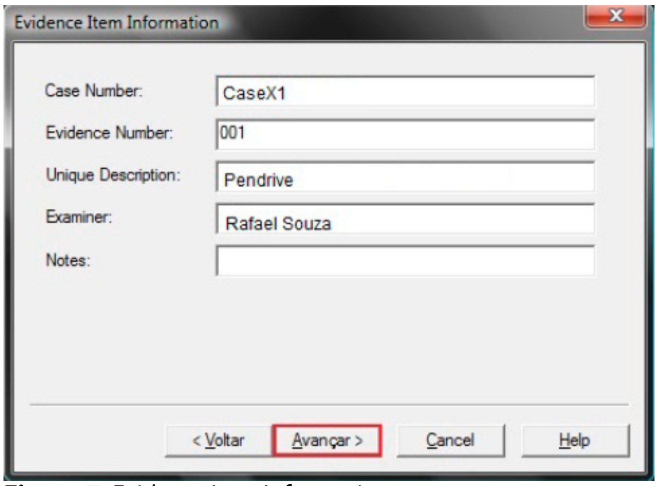
Select Image Type

Please Select the Destination Image Type

- [✓ Raw (dd)]
- SMART
- E01
- AFF

Figure 6. *Select RAW*

Will request some information on evidência. We can fill these information. After that, click on "Next".


Evidence Item Information

Case Number:	CaseX1
Evidence Number:	001
Unique Description:	Pendrive
Examiner:	Rafael Souza
Notes:	

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.

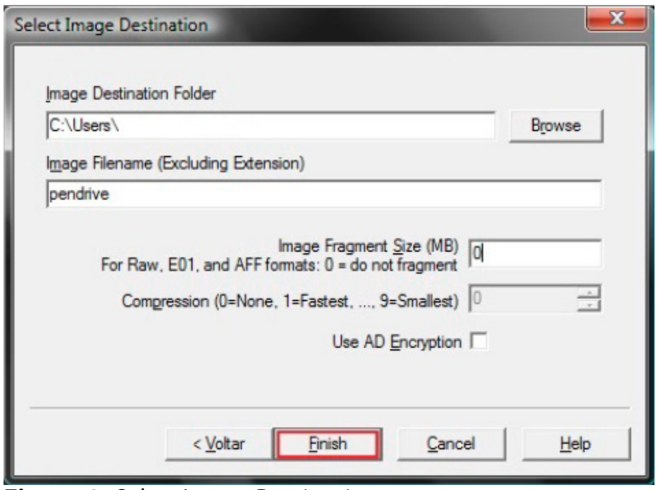

Select Image Destination

Image Destination Folder: C:\Users\
Image Filename (Excluding Extension): pendrive
Image Fragment Size (MB) For Raw, E01, and AFF formats: 0 = do not fragment: 0
Compression (0=None, 1=Fastest, ..., 9=Smallest): 0
Use AD Encryption: ☐

Figure 8. *Select Image Destination*

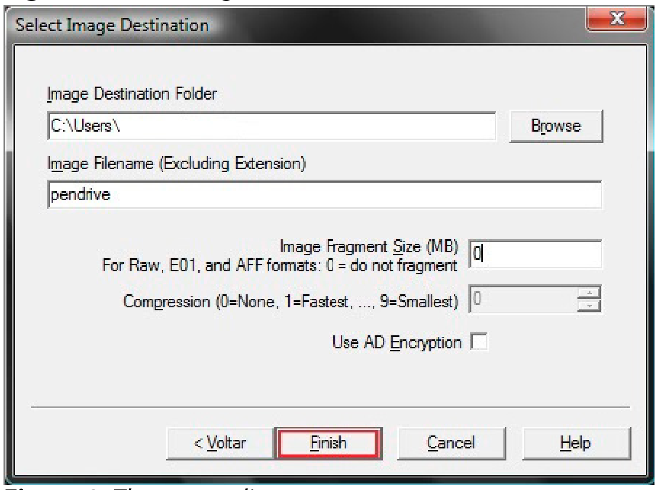

Select Image Destination

Image Destination Folder: C:\Users\
Image Filename (Excluding Extension): pendrive
Image Fragment Size (MB) For Raw, E01, and AFF formats: 0 = do not fragment: 0
Compression (0=None, 1=Fastest, ..., 9=Smallest): 0
Use AD Encryption: ☐

Figure 9. *The output directory*

Just click on the "Start" button.

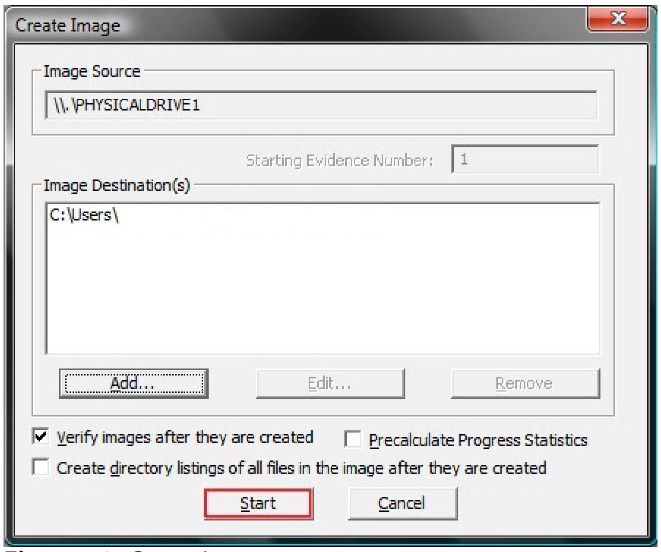

Create Image

Image Source: \\.\PHYSICALDRIVE1
Starting Evidence Number: 1
Image Destination(s): C:\Users\
☑ Verify images after they are created
☐ Precalculate Progress Statistics
☐ Create directory listings of all files in the image after they are created

Figure 10. *Create Image*

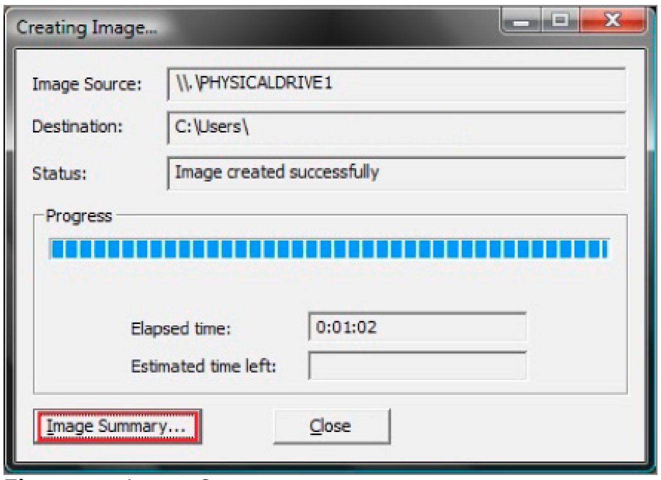

Creating Image...

Image Source: \\.\PHYSICALDRIVE1
Destination: C:\Users\
Status: Image created successfully
Progress
Elapsed time: 0:01:02
Estimated time left:

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.

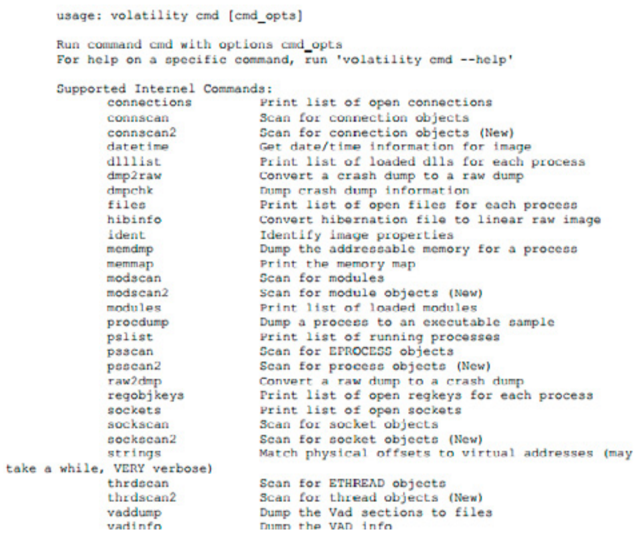
EXTRACTION OF DIGITA ARTIFACTS WITH VOLATILITY

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.

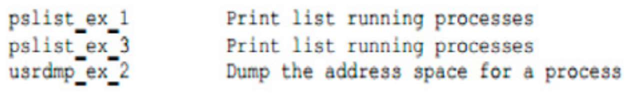
```
C:\Volatility>python volatility
```



```
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`

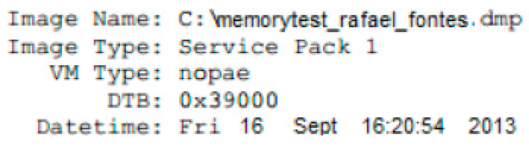


```
pslist_ex_1         Print list running processes
pslist_ex_3         Print list running processes
usrdmp_ex_2         Dump the address space for a process
```

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
```

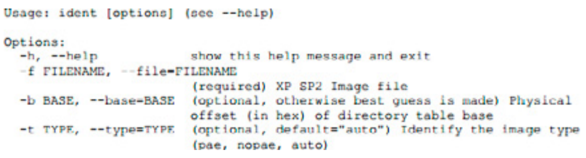


```
   Image Name: C:\memorytest_rafael_fontes.dmp
   Image Type: Service Pack 1
      VM Type: nopae
          DTB: 0x39000
     Datetime: Fri 16  Sept  16:20:54  2013
```

Figure 14. *Command ident*

You can use the `--help` option with any command to get help:

```
C:\Volatility>python volatility ident --help
```

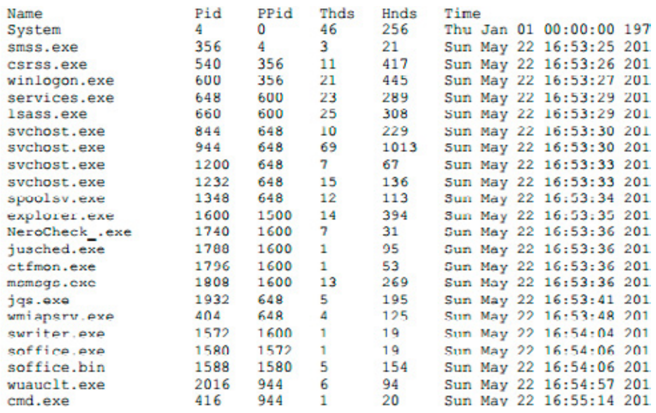


```
Usage: ident [options] (see --help)

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
```

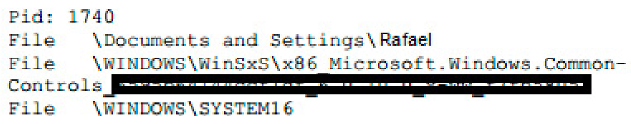


```
Name                 Pid    PPid   Thds   Hnds   Time
System               4      0      46     256    Thu Jan 01 00:00:00 197
smss.exe             356    4      3      21     Sun May 22 16:53:25 201
csrss.exe            540    356    11     417    Sun May 22 16:53:26 201
winlogon.exe         600    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
```

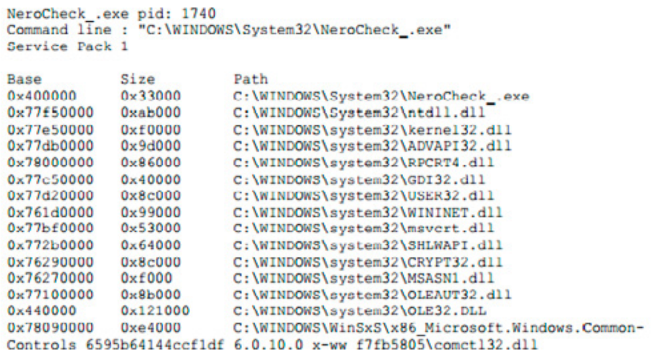


```
Pid: 1740
File   \Documents and Settings\Rafael
File   \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_[redacted]
File   \WINDOWS\SYSTEM16
```

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.

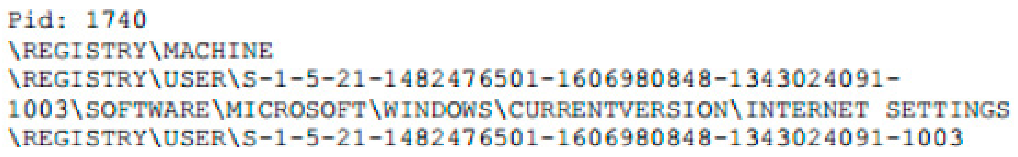
```
C:\Volatility>python volatility dlllist -p 1740 -f
   C:\memorytest_rafael_fontes.dmp
```



```
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
```

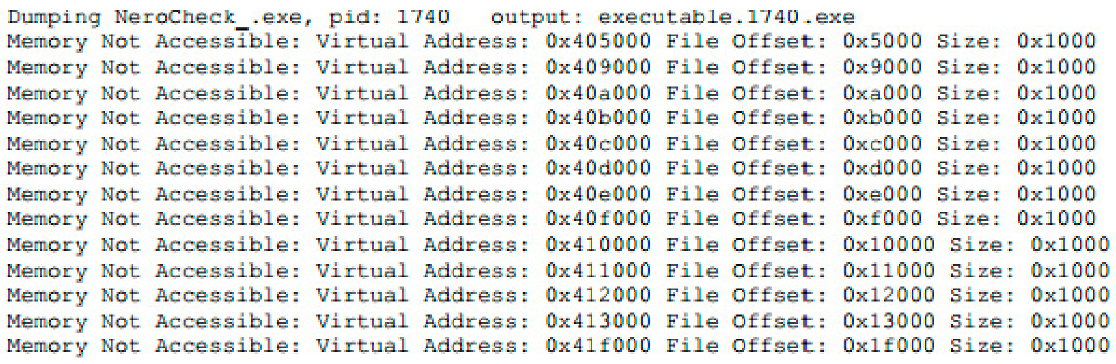


```
Pid: 1740
\REGISTRY\MACHINE
\REGISTRY\USER\S-1-5-21-1482476501-1606980848-1343024091-
1003\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\INTERNET SETTINGS
\REGISTRY\USER\S-1-5-21-1482476501-1606980848-1343024091-1003
```

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:

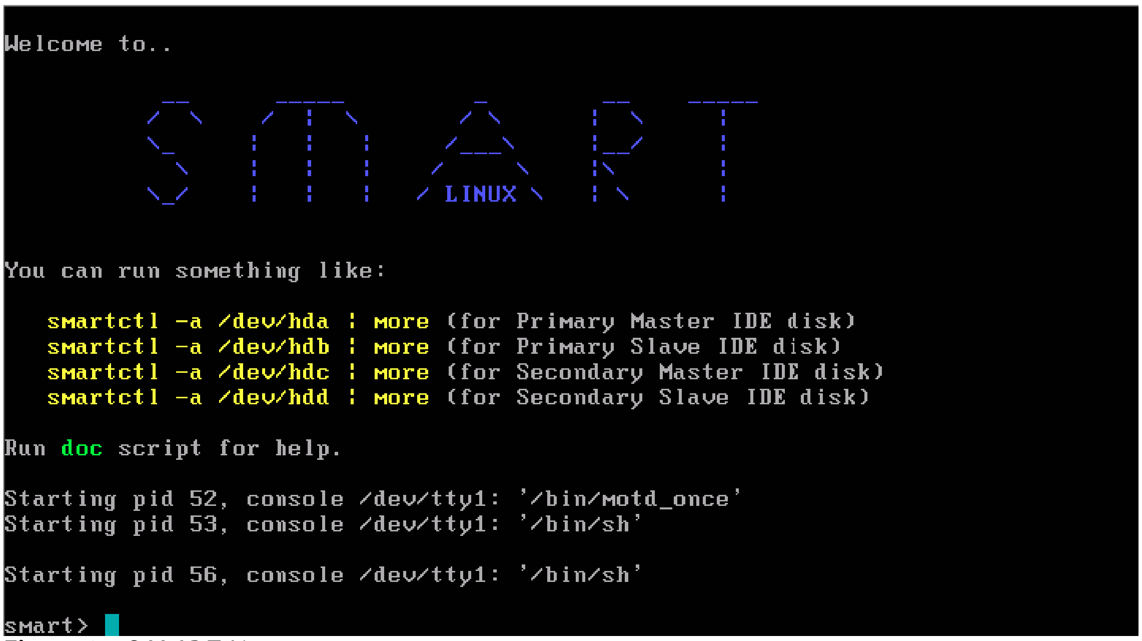
```
 >>> Python volatility pslist -f target-2013-10-10.img
Name Pid PPID THDs HNDs Time
lsass.exe 536 480 20 369 Mon Oct 10 16:22:18 2013
```

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/*

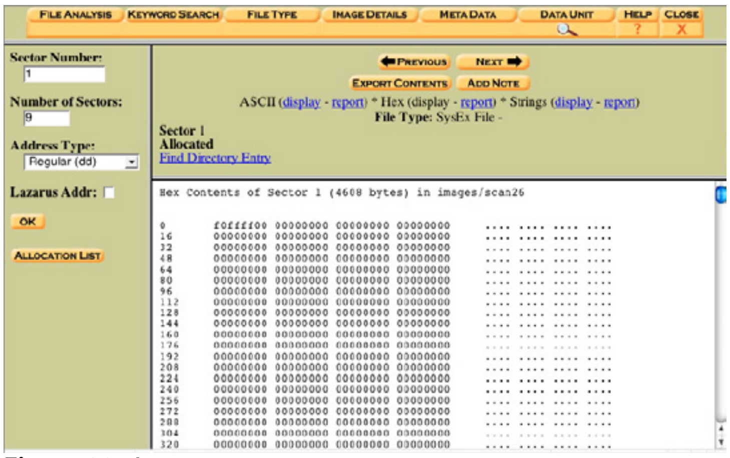


```
Welcome to..

        LINUX

You can run something like:

   smartctl -a /dev/hda | more (for Primary Master IDE disk)
   smartctl -a /dev/hdb | more (for Primary Slave IDE disk)
   smartctl -a /dev/hdc | more (for Secondary Master IDE disk)
   smartctl -a /dev/hdd | more (for Secondary Slave IDE disk)

Run doc script for help.

Starting pid 52, console /dev/tty1: '/bin/motd_once'
Starting pid 53, console /dev/tty1: '/bin/sh'

Starting pid 56, console /dev/tty1: '/bin/sh'

smart>
```

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/*.



Sector Number: 1

Number of Sectors: 9

Address Type: Regular (dd)

Lazarus Addr:

ASCII (display - report) * Hex (display - report) * Strings (display - report)
File Type: SysEx File -

Sector 1
Allocated
Find Directory Entry

```
Hex Contents of Sector 1 (4608 bytes) in images/scan26

0       f0ffff00 00000000 00000000 00000000     .... .... .... ....
16      00000000 00000000 00000000 00000000     .... .... .... ....
32      00000000 00000000 00000000 00000000     .... .... .... ....
48      00000000 00000000 00000000 00000000     .... .... .... ....
64      00000000 00000000 00000000 00000000     .... .... .... ....
80      00000000 00000000 00000000 00000000     .... .... .... ....
96      00000000 00000000 00000000 00000000     .... .... .... ....
112     00000000 00000000 00000000 00000000     .... .... .... ....
128     00000000 00000000 00000000 00000000     .... .... .... ....
144     00000000 00000000 00000000 00000000     .... .... .... ....
160     00000000 00000000 00000000 00000000     .... .... .... ....
176     00000000 00000000 00000000 00000000     .... .... .... ....
192     00000000 00000000 00000000 00000000     .... .... .... ....
208     00000000 00000000 00000000 00000000     .... .... .... ....
224     00000000 00000000 00000000 00000000     .... .... .... ....
240     00000000 00000000 00000000 00000000     .... .... .... ....
256     00000000 00000000 00000000 00000000     .... .... .... ....
272     00000000 00000000 00000000 00000000     .... .... .... ....
288     00000000 00000000 00000000 00000000     .... .... .... ....
304     00000000 00000000 00000000 00000000     .... .... .... ....
320     00000000 00000000 00000000 00000000     .... .... .... ....
```

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.

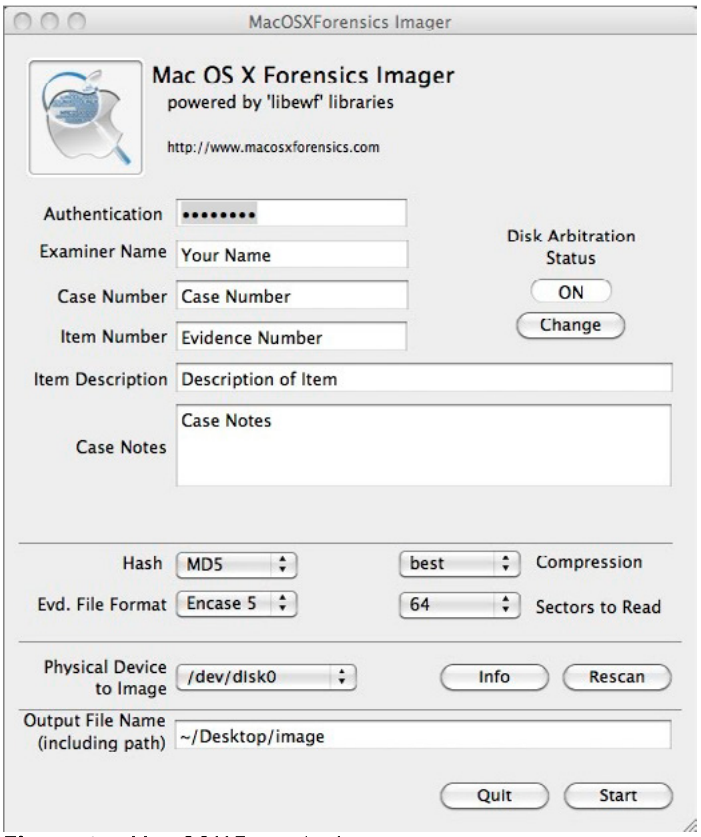
The main design objectives that CAINE aims to guarantee are the following:

- 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*.


MacOSXForensics Imager

Mac OS X Forensics Imager
powered by 'libewf' libraries
http://www.macosxforensics.com

Authentication: ••••••••
Examiner Name: Your Name
Case Number: Case Number
Item Number: Evidence Number
Item Description: Description of Item
Case Notes: Case Notes

Disk Arbitration Status: ON
Change

Hash: MD5
Compression: best
Evd. File Format: Encase 5
Sectors to Read: 64

Physical Device to Image: /dev/disk0
Info
Rescan

Output File Name (including path): ~/Desktop/image

Quit
Start

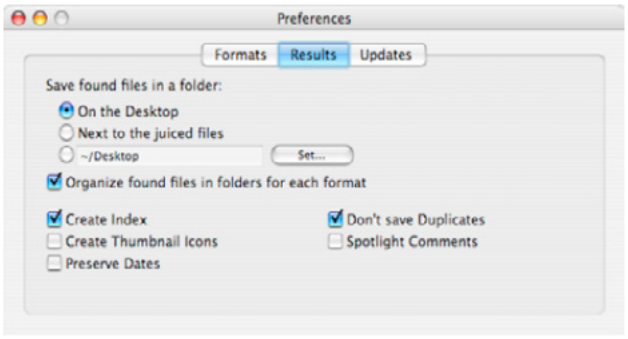
Figure 25. *Mac OS X Forensics Imager*

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/*.

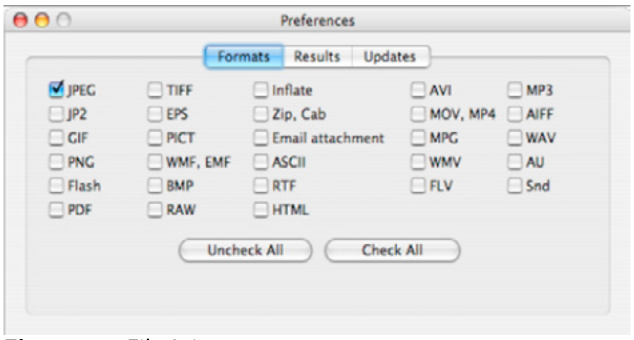

Preferences

Formats | Results | Updates

Save found files in a folder:
[✓ On the Desktop]
Next to the juiced files
~/Desktop Set...
[✓] Organize found files in folders for each format

[✓] Create Index
[✓] Don't save Duplicates
[ ] Create Thumbnail Icons
[ ] Spotlight Comments
[ ] Preserve Dates

Figure 26. *File Juicer 1*


Preferences

Formats | Results | Updates

[✓] JPEG
[ ] JP2
[ ] GIF
[ ] PNG
[ ] Flash
[ ] PDF
[ ] TIFF
[ ] EPS
[ ] PICT
[ ] WMF, EMF
[ ] BMP
[ ] RAW
[ ] Inflate
[ ] Zip, Cab
[ ] Email attachment
[ ] ASCII
[ ] RTF
[ ] HTML
[ ] AVI
[ ] MOV, MP4
[ ] MPG
[ ] WMV
[ ] FLV
[ ] MP3
[ ] AIFF
[ ] WAV
[ ] AU
[ ] Snd

Uncheck All
Check All

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

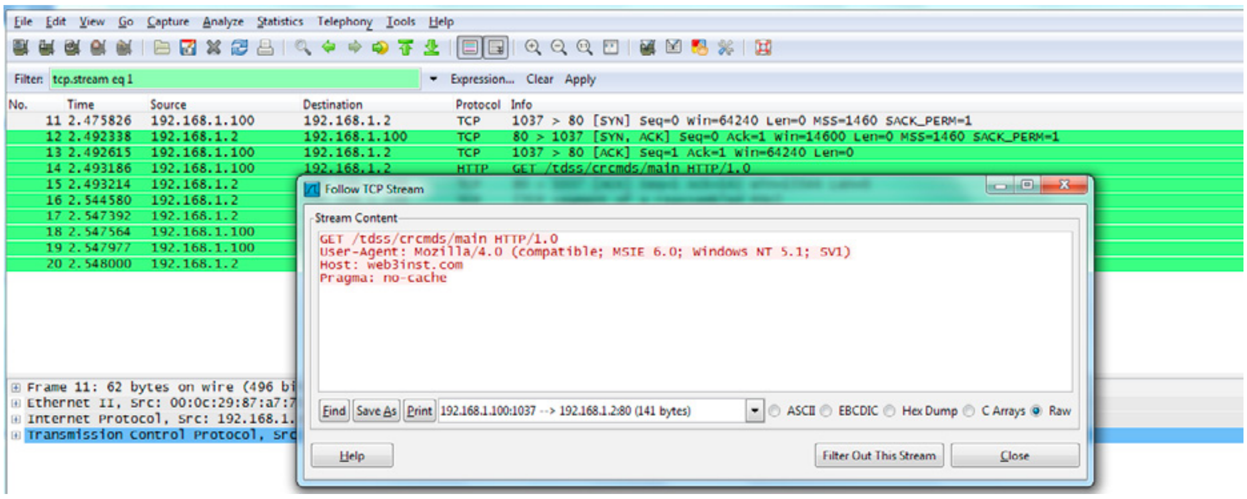
- Using `-h` or `--help` option will display help options and list of a available plugins
  Example: `python vol.py -h`
- Use `-f <filename>` and `--profile` to indicate the memory dump you are analyzing
  Example: `python vol.py -f mem.dmp --profile=WinXPSP3x86`
- To know the `--profile` info use below 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.


File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: tcp.stream eq 1
Expression... Clear Apply

```
No.   Time      Source         Destination    Protocol Info
   11 2.475826  192.168.1.100  192.168.1.2    TCP      1037 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
   12 2.492338  192.168.1.2    192.168.1.100  TCP      80 > 1037 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1
   13 2.492615  192.168.1.100  192.168.1.2    TCP      1037 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
   14 2.493186  192.168.1.100  192.168.1.2    HTTP     GET /tdss/crcmds/main HTTP/1.0
   15 2.493214  192.168.1.2
   16 2.544580  192.168.1.2
   17 2.547392  192.168.1.2
   18 2.547564  192.168.1.100
   19 2.547977  192.168.1.100
   20 2.548000  192.168.1.2

⊞ Frame 11: 62 bytes on wire (496 bi
⊞ Ethernet II, Src: 00:0c:29:87:a7:7
⊞ Internet Protocol, Src: 192.168.1.
⊞ Transmission Control Protocol, Src
```

```
GET /tdss/crcmds/main HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: web3inst.com
Pragma: no-cache
```

Find Save As Print
192.168.1.100:1037 --> 192.168.1.2:80 (141 bytes)
ASCII EBCDIC Hex Dump C Arrays [✓ Raw]
Help Filter Out This Stream Close

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".

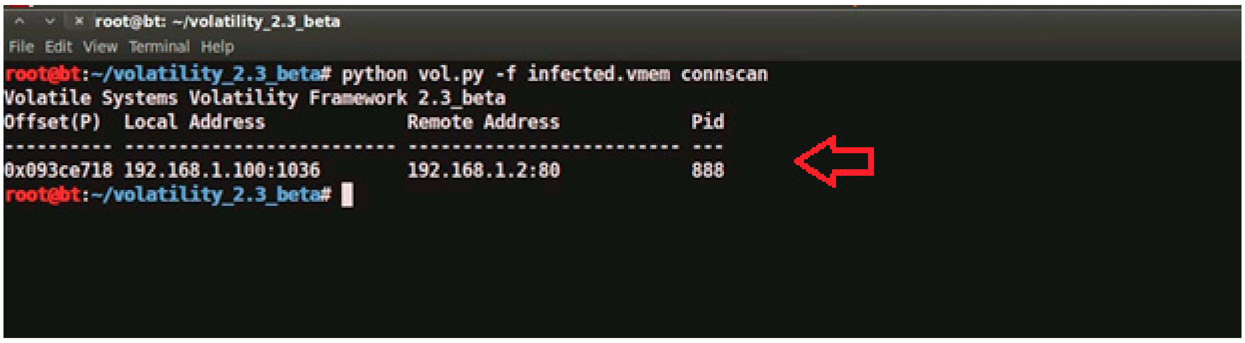
ANALYSIS

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.

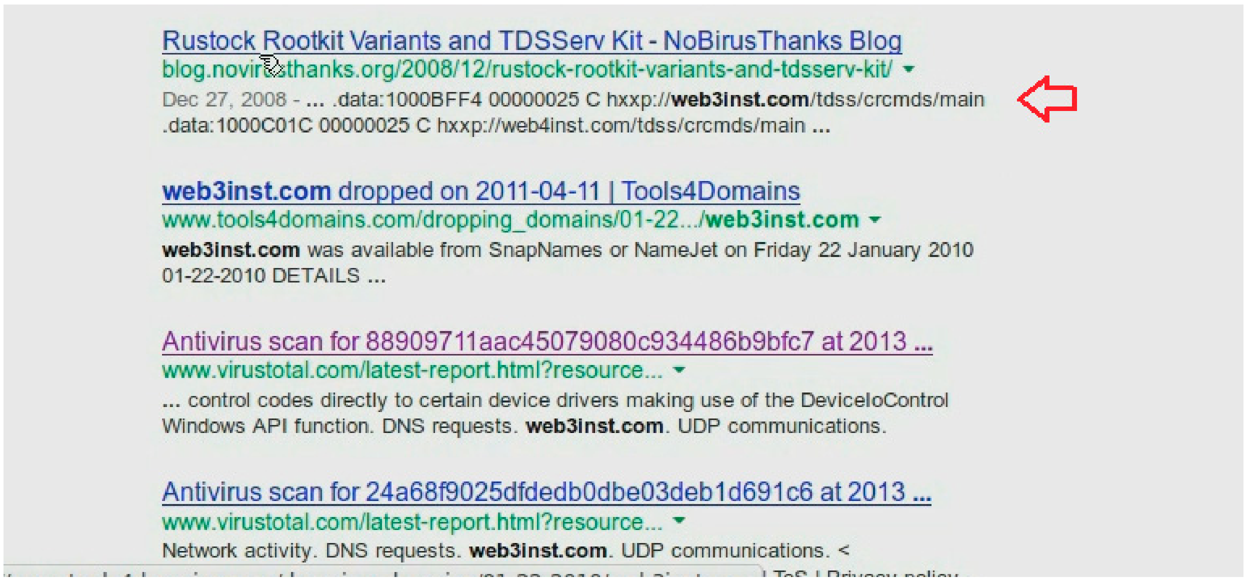
Volatility's connscan module, shows connection to the malicious ip made by process (with pid 888)



```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem connscan
Volatile Systems Volatility Framework 2.3_beta
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x093ce718 192.168.1.100:1036        192.168.1.2:80            888
root@bt:~/volatility_2.3_beta#
```

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.



Rustock Rootkit Variants and TDSServ Kit - NoBirusThanks Blog
blog.novirusthanks.org/2008/12/rustock-rootkit-variants-and-tdsserv-kit/
Dec 27, 2008 - ... .data:1000BFF4 00000025 C hxxp://**web3inst.com**/tdss/crcmds/main .data:1000C01C 00000025 C hxxp://web4inst.com/tdss/crcmds/main ...

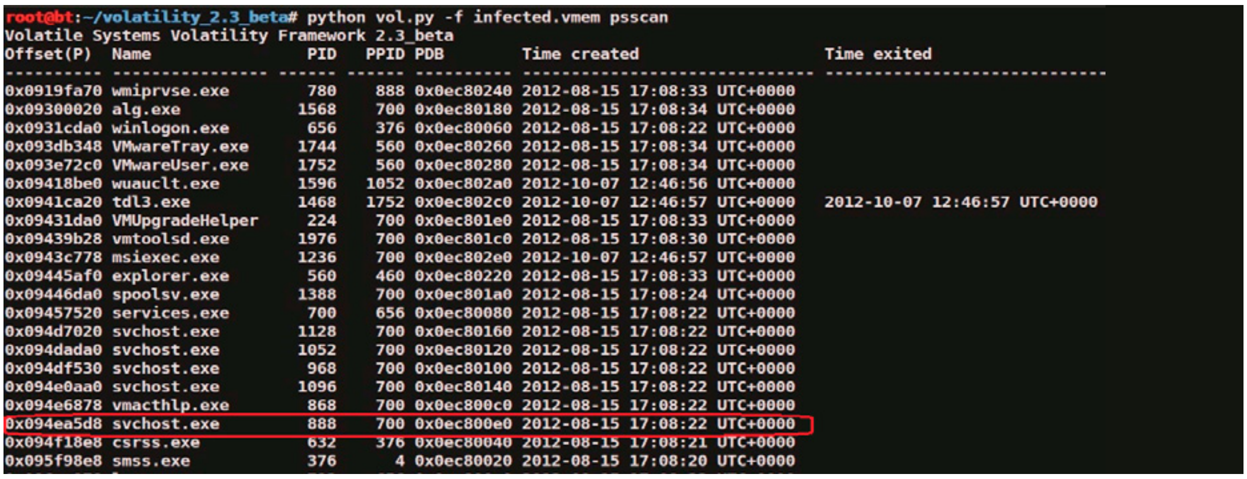
web3inst.com dropped on 2011-04-11 | Tools4Domains
www.tools4domains.com/dropping_domains/01-22.../**web3inst.com**
web3inst.com was available from SnapNames or NameJet on Friday 22 January 2010 01-22-2010 DETAILS ...

Antivirus scan for 88909711aac45079080c934486b9bfc7 at 2013 ...
www.virustotal.com/latest-report.html?resource...
... control codes directly to certain device drivers making use of the DeviceIoControl Windows API function. DNS requests. **web3inst.com**. UDP communications.

Antivirus scan for 24a68f9025dfdedb0dbe03deb1d691c6 at 2013 ...
www.virustotal.com/latest-report.html?resource...
Network activity. DNS requests. **web3inst.com**. UDP communications. <

STEP 3: WHAT IS PID 888?

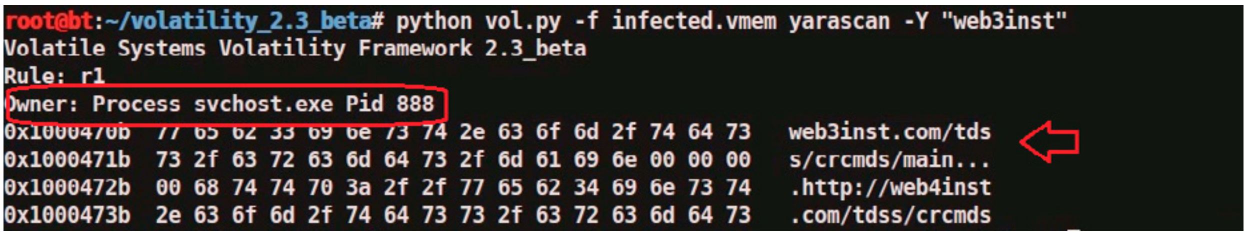
Since the network connection to the ip 192.168.1.2 was made by pid 888, we need to determine which process is associated with pid 888. "psscan" shows pid 888 belongs to svchost.exe.



```
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

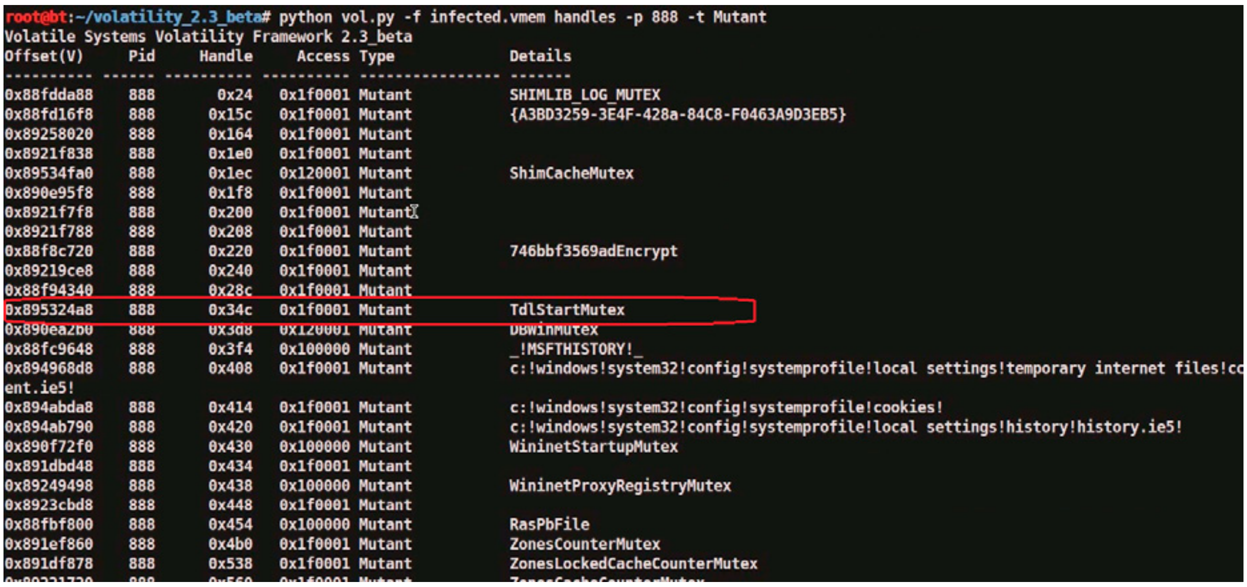
Running the YARA scan on the memory dump for the string "web3inst" confirms that this domain (web3inst.com) is present in the address space of svchost.exe (pid 888). This confirms that svchost.exe was making connections to the malicious domain "web3inst.com"



```
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

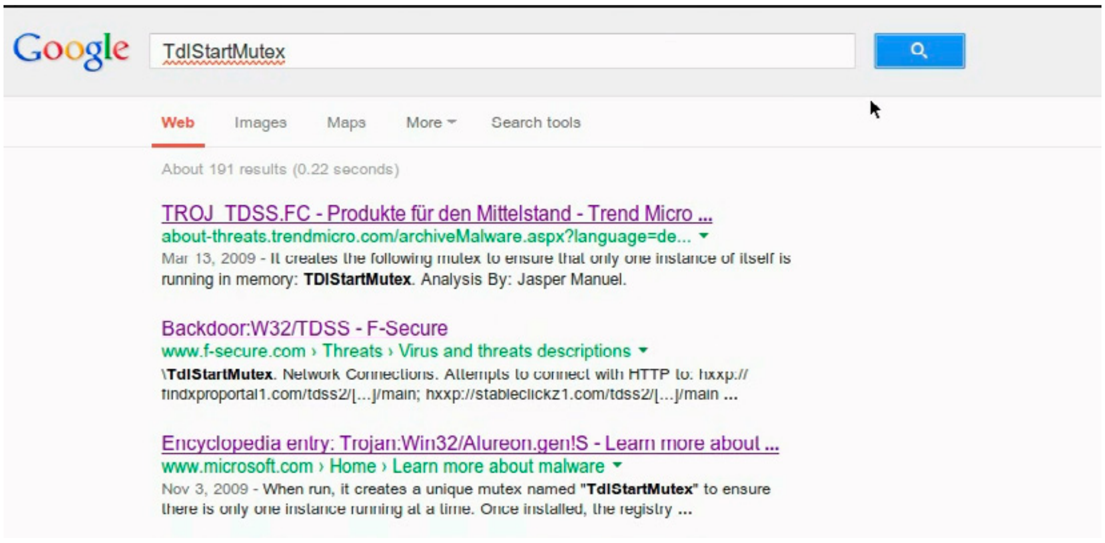
Now we know that svchost.exe process (pid 888) was making connections to the domain "web3inst.com", lets focus on this process. Checking for the mutex created by svchost.exe shows a suspicious mutex "TdlStartMutex"



```
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.



About 191 results (0.22 seconds)

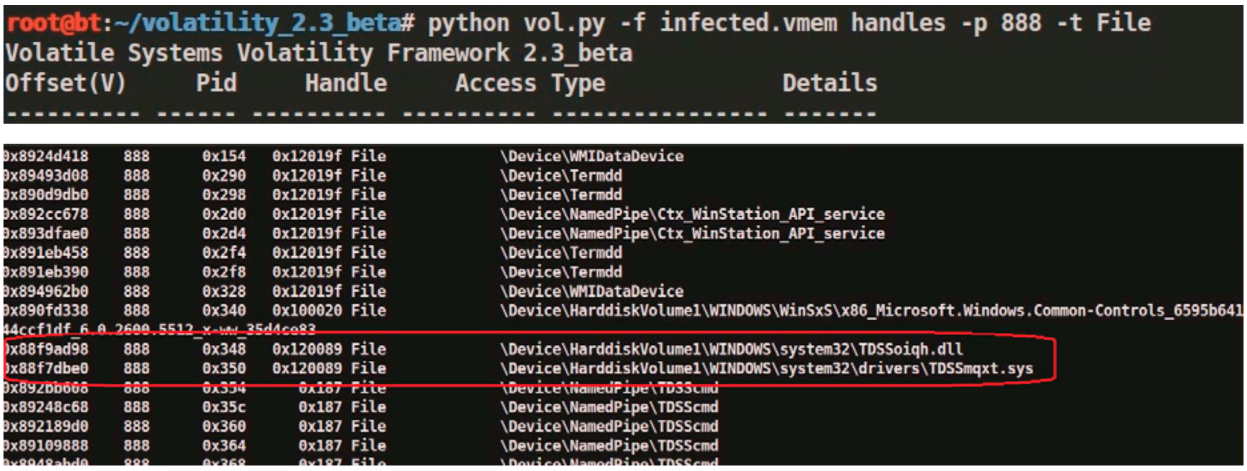
TROJ_TDSS.FC - Produkte für den Mittelstand - Trend Micro ...
about-threats.trendmicro.com/archiveMalware.aspx?language=de...
Mar 13, 2009 - It creates the following mutex to ensure that only one instance of itself is running in memory: **TDlStartMutex**. Analysis By: Jasper Manuel.

Backdoor:W32/TDSS - F-Secure
www.f-secure.com › Threats › Virus and threats descriptions
TdlStartMutex. Network Connections. Attempts to connect with HTTP to: hxxp://findxproportal1.com/tdss2/[...]/main; hxxp://stableclickz1.com/tdss2/[...]/main ...

Encyclopedia entry: Trojan:Win32/Alureon.gen!S - Learn more about ...
www.microsoft.com › Home › Learn more about malware
Nov 3, 2009 - When run, it creates a unique mutex named "**TdlStartMutex**" to ensure there is only one instance running at a time. Once installed, the registry ...

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"

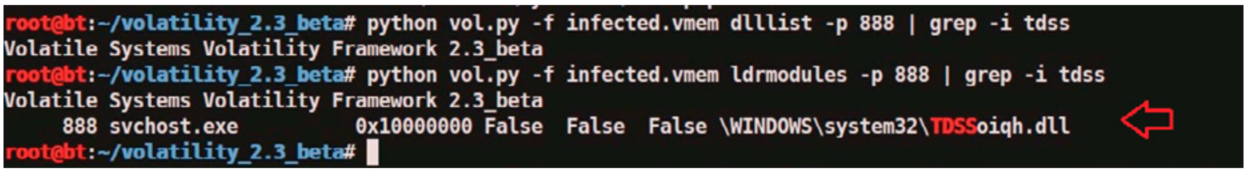


```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem handles -p 888 -t File
Volatile Systems Volatility Framework 2.3_beta
Offset(V)     Pid     Handle     Access Type             Details
---------- ------ ---------- ---------- ---------------- -------
```

```
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
0x892bb608    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

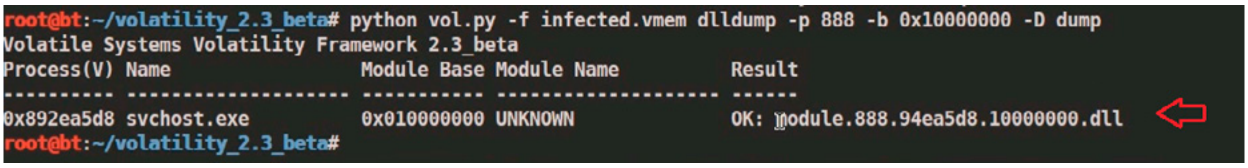
Volatility's dlllist module couldn't find the DLL starting with "TDSS" whereas ldrmodules plugin was able to find it. This confirms that the DLL (TDSSoiqh.dll) was hidden. malware hides the DLL by unlinking from the 3 PEB lists (operating sytem keeps track of the DLL's in these lists)



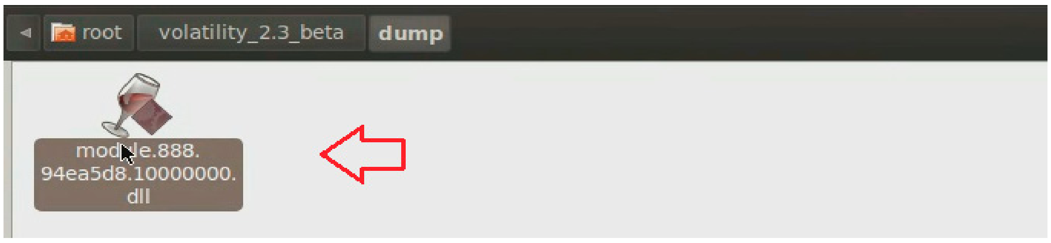
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlllist -p 888 | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem ldrmodules -p 888 | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
     888 svchost.exe          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

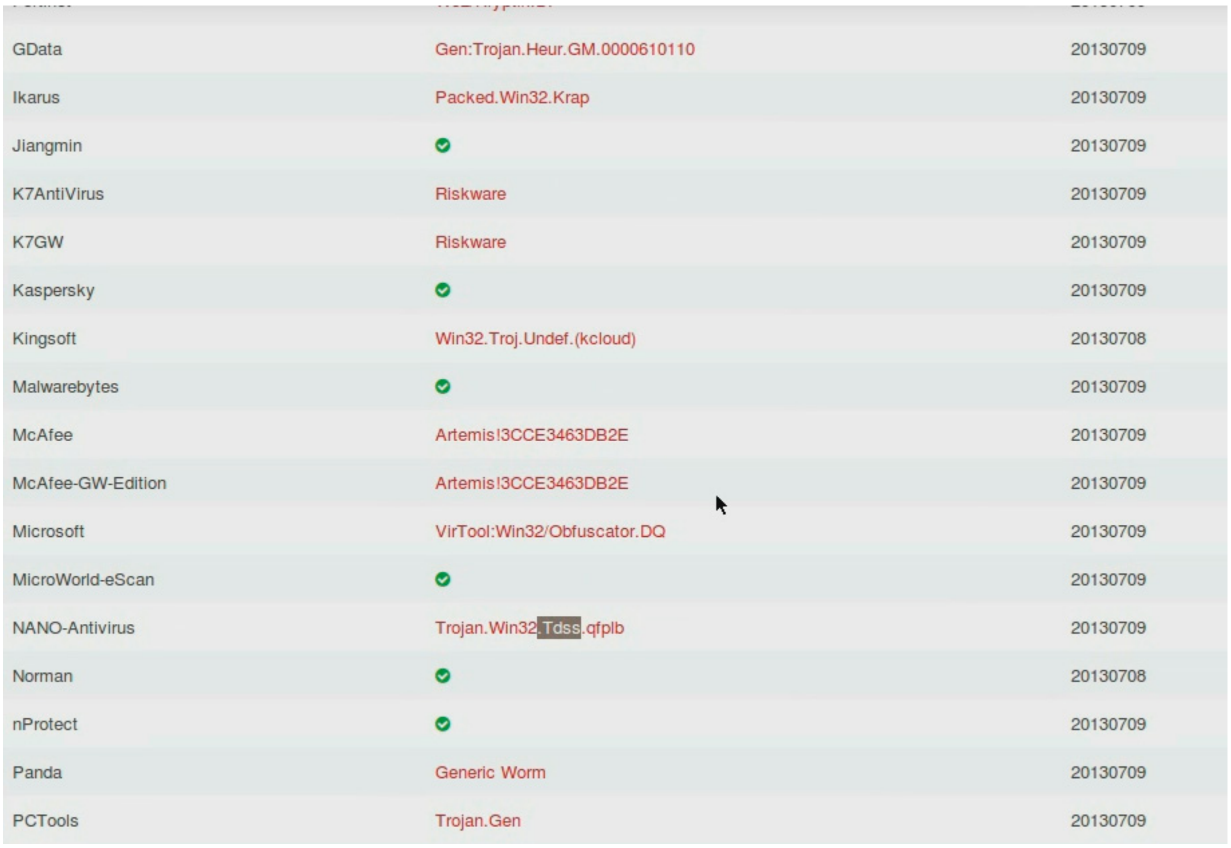


```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlldump -p 888 -b 0x10000000 -D dump
Volatile Systems Volatility Framework 2.3_beta
Process(V) Name                Module Base Module Name          Result
---------- ------------------- ----------- -------------------- ------
0x892ea5d8 svchost.exe         0x010000000 UNKNOWN              OK: module.888.94ea5d8.10000000.dll
root@bt:~/volatility_2.3_beta#
```



STEP 10: VIRUSTOTAL SUBMISSION OF DUMPED DLL

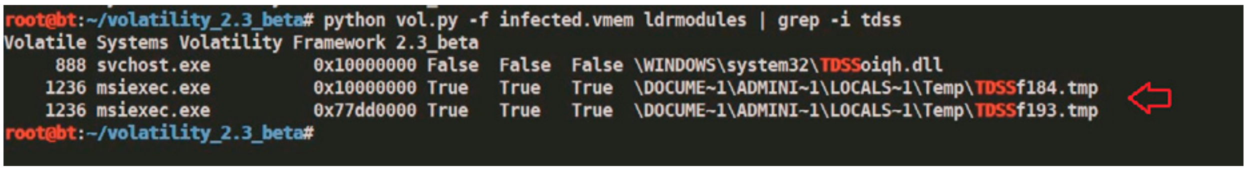
Submitting the dumped dll to VirusTotal confirms that it is malicious



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

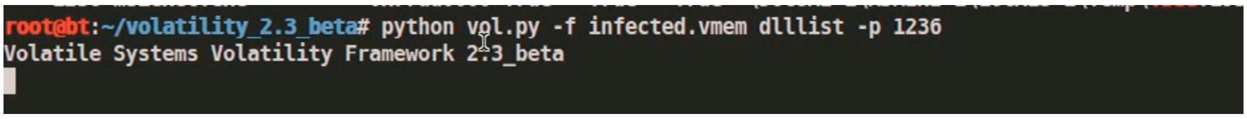
Looking for the modules in all the processes that start with "TDSS" shows that msiexec.exe process (pid 1236) has reference to a temp file (which is starting with TDSS) which is suspicous.



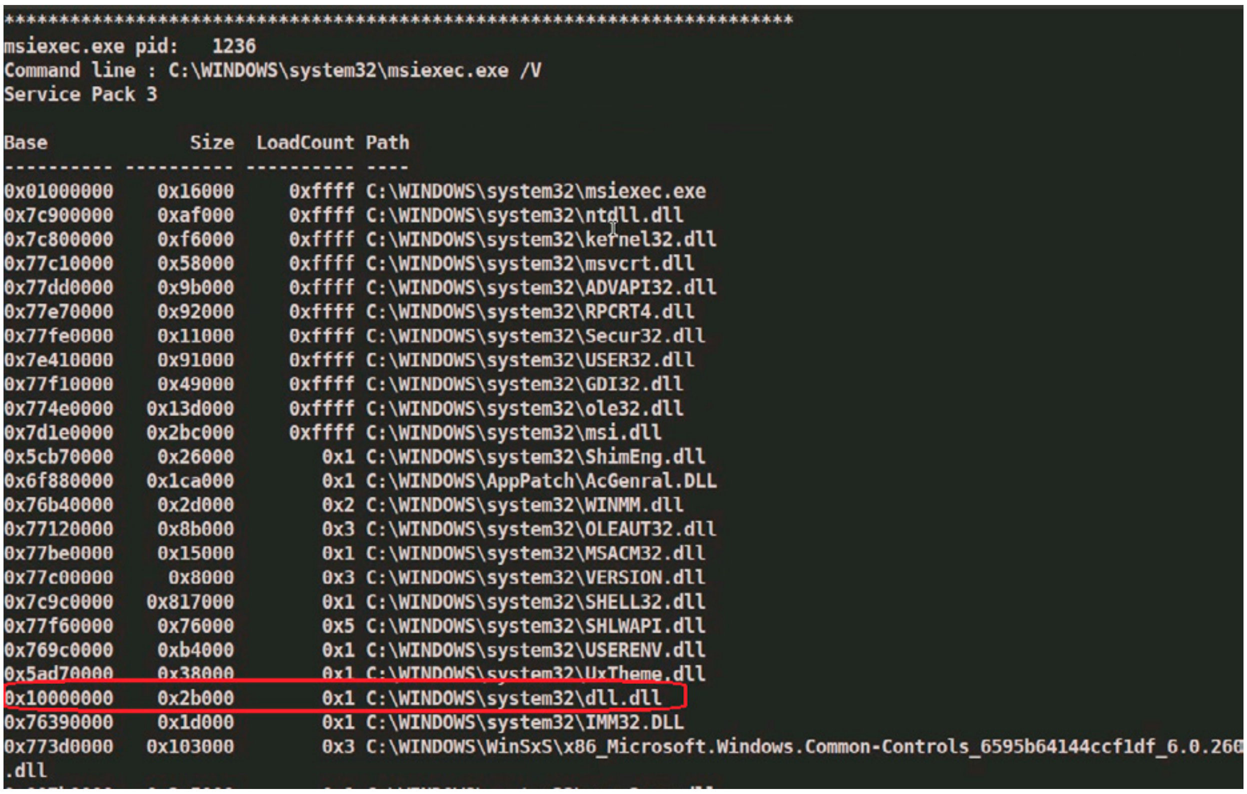
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem ldrmodules | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
     888 svchost.exe          0x10000000 False  False  False \WINDOWS\system32\TDSSoiqh.dll
    1236 msiexec.exe          0x10000000 True   True   True  \DOCUME~1\ADMINI~1\LOCALS~1\Temp\TDSSf184.tmp
    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.



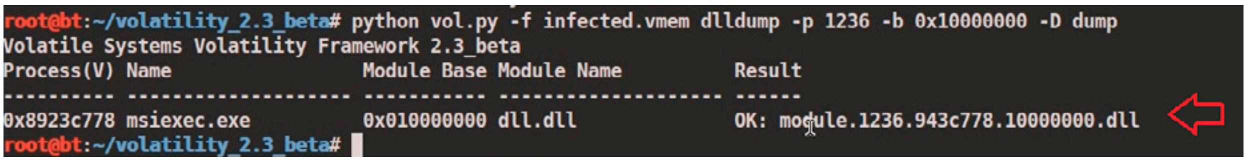
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlllist -p 1236
Volatile Systems Volatility Framework 2.3_beta
```



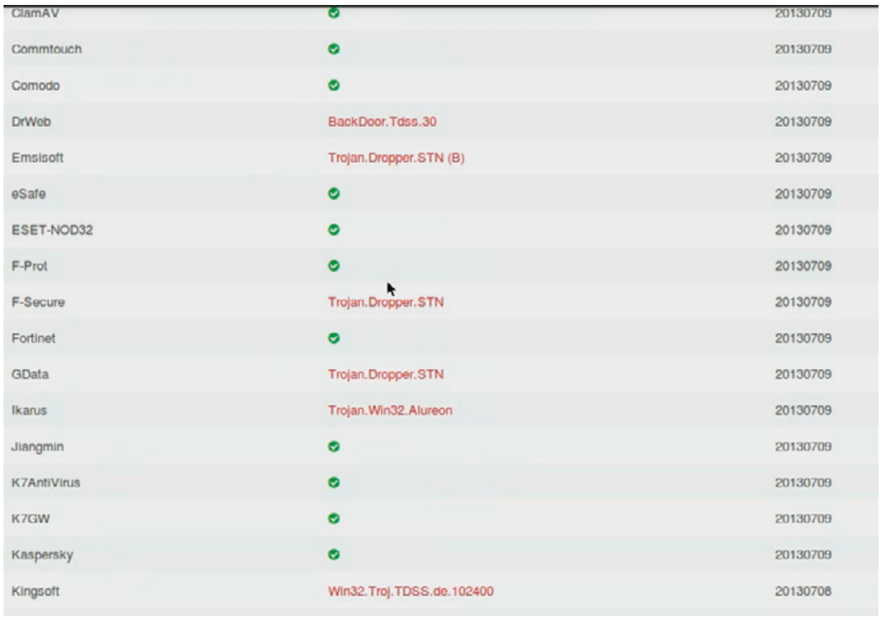
```
************************************************************************
msiexec.exe pid:   1236
Command line : C:\WINDOWS\system32\msiexec.exe /V
Service Pack 3

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



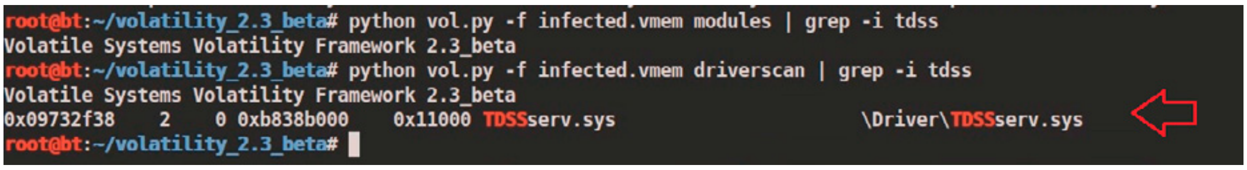
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem dlldump -p 1236 -b 0x10000000 -D dump
Volatile Systems Volatility Framework 2.3_beta
Process(V) Name                 Module Base Module Name          Result
---------- -------------------- ----------- -------------------- ------
0x8923c778 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

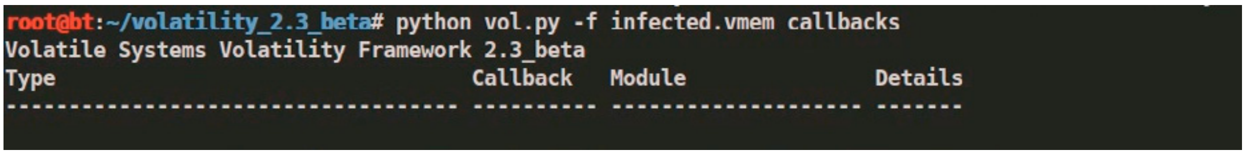
In step 7 we also saw reference to a driver file (starting with "TDSS"). Searching for the driver file using Volatility's modules plugin couldn't find the driver that starts with "TDSS" whereas Volatility's driverscan plugin was able to find it. This confirms that the kernel driver (TDSSserv.sys) was hidden. The below screenshot also shows that the base address of the driver is "0xb838b000" and the size is "0x11000"



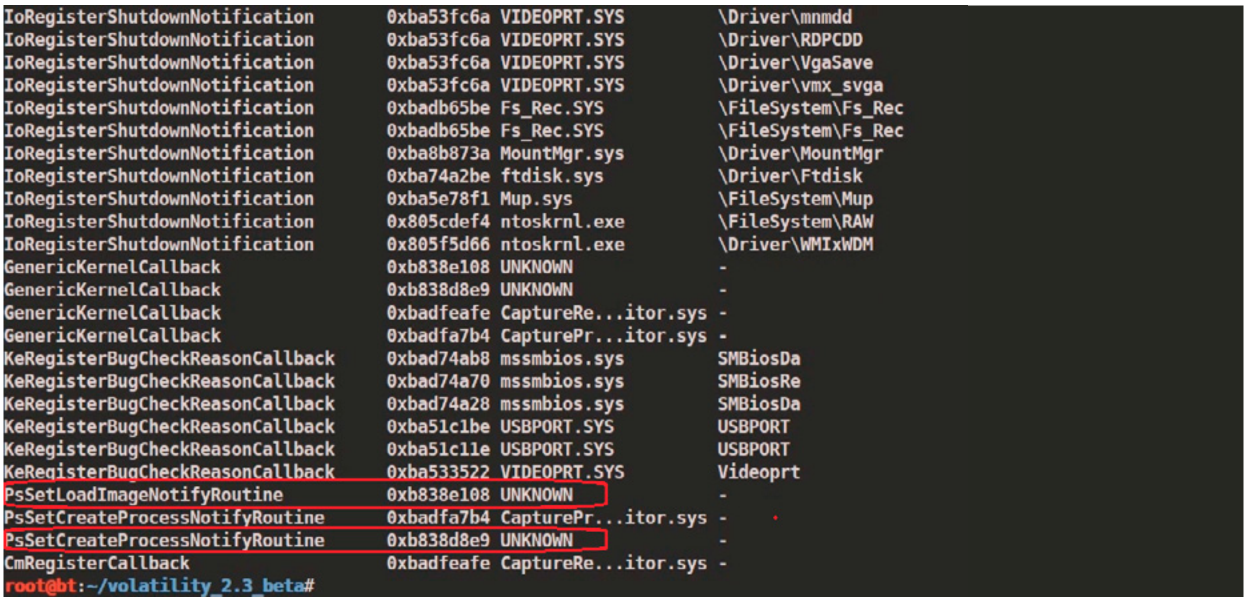
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem modules | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem driverscan | grep -i tdss
Volatile Systems Volatility Framework 2.3_beta
0x09732f38    2    0 0xb838b000    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



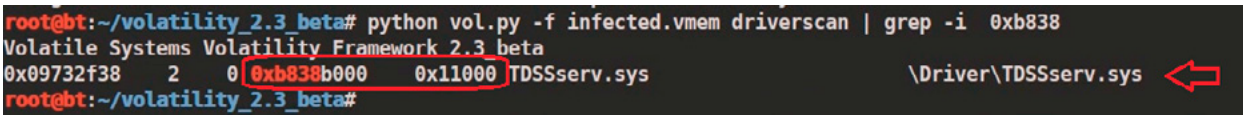
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem callbacks
Volatile Systems Volatility Framework 2.3_beta
Type                                 Callback   Module               Details
------------------------------------ ---------- -------------------- -------
```



```
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

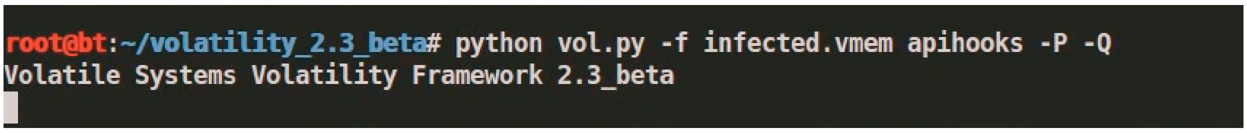
The below screenshot shows that this unknown driver falls under the address range of TDSSserv.sys. This confirms that unknown driver is "TDSSserv.sys"



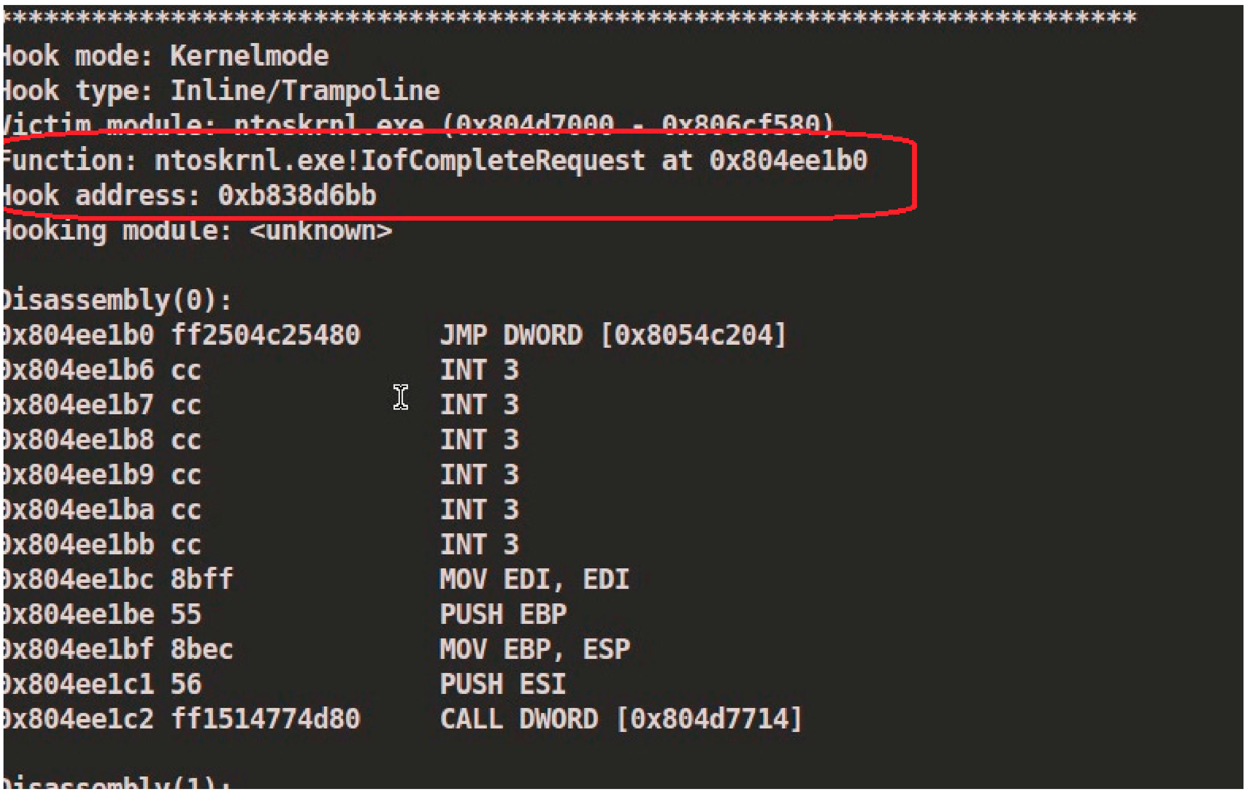
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem driverscan | grep -i  0xb838
Volatile Systems Volatility Framework 2.3_beta
0x09732f38    2    0 0xb838b000    0x11000 TDSSserv.sys                  \Driver\TDSSserv.sys
root@bt:~/volatility_2.3_beta#
```

STEP 17: KERNEL API HOOKS

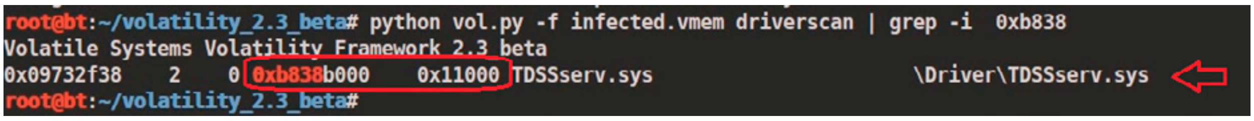
Malware hooks the Kernel API and the hook address falls under the address range of TDSSserv.sys (as shown in the below screenshots)



```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem apihooks -P -Q
Volatile Systems Volatility Framework 2.3_beta
```



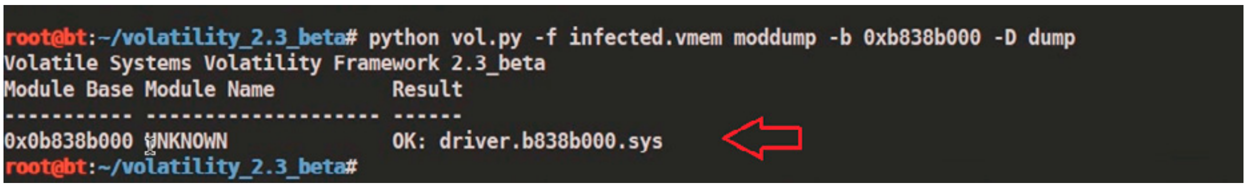
```
*****************************************************************************
Hook mode: Kernelmode
Hook type: Inline/Trampoline
Victim module: ntoskrnl.exe (0x804d7000 - 0x806cf580)
Function: ntoskrnl.exe!IofCompleteRequest at 0x804ee1b0
Hook address: 0xb838d6bb
Hooking module: <unknown>

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):
```



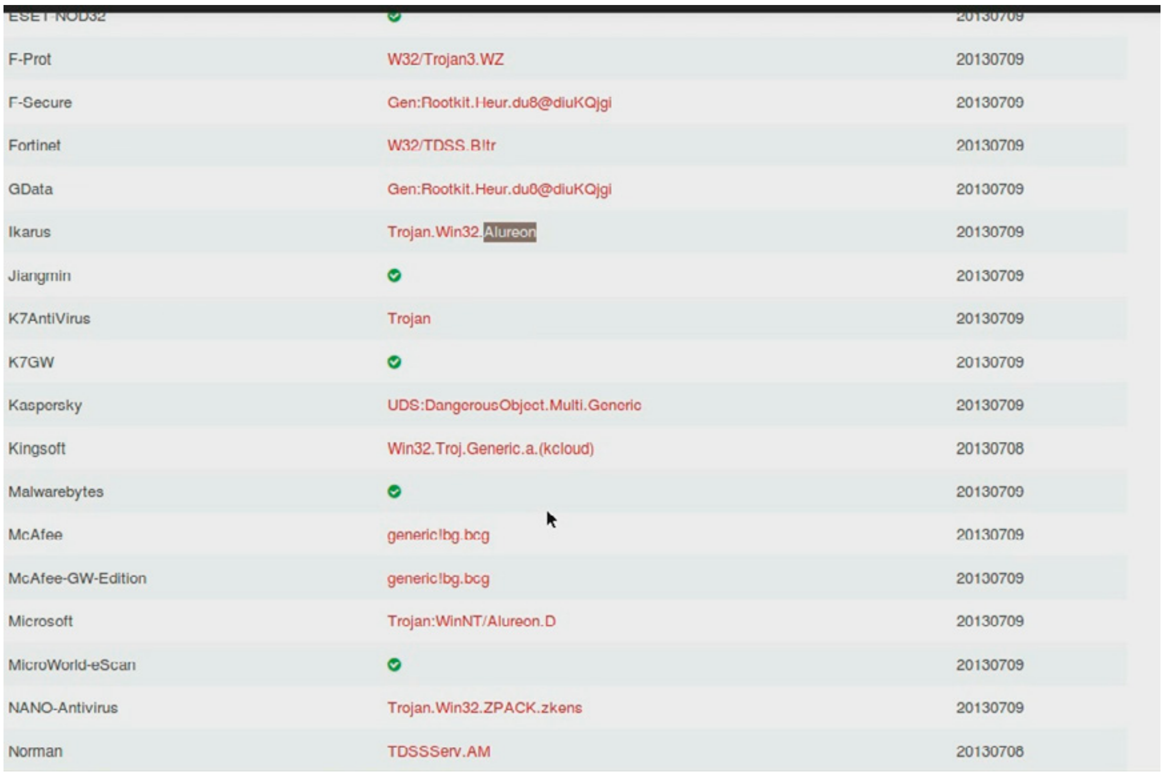
```
root@bt:~/volatility_2.3_beta# python vol.py -f infected.vmem driverscan | grep -i  0xb838
Volatile Systems Volatility Framework 2.3_beta
0x09732f38    2    0 0xb838b000    0x11000 TDSSserv.sys                  \Driver\TDSSserv.sys
root@bt:~/volatility_2.3_beta#
```

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.B!tr	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	20130708

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.

Cyber Security and Digital Forensics 2013

3-5 December 2013, Kuala Lumpur, Malaysia

ib·consultancy

**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

MEMORY FORENSICS: INTEGRATING DIGITAL FORENSICS WITH ARCHIVAL SCIENCE FOR TRUSTING RECORDS AND DATA

by Luciana Duranti University of British Columbia **and Corinne Rogers** University of British Columbia

Both archival and digital forensics methods and principles evolved out of practice and grew into established professional disciplines by developing theoretical foundations, which then returned to inform and standardize that practice. Digital Records Forensics is a new discipline arising from the intersection of digital forensics and archival science to identify records in digital systems, assess their authenticity, and establish the requirements for their long-term preservation.

What you will learn:

- Readers will be introduced to the concepts of archival diplomatics made explicit in the Chain of Preservation model of record creation
- Maintenance,
- Preservation,
- Links these concepts to concepts of digital forensic investigation.

What you should know:

- Basic familiarity with archival theory would be helpful, but is not essential.
- Readers should know the principles of digital forensics.

This paper introduces areas of convergence between digital forensics and archival preservation activities in order to understand moments in which digital records as understood by archival science, and digital evidence as understood by digital forensics, may be identified, their authenticity assessed, their reliability and integrity managed and preserved. The paper shows how digital forensics can enhance archival science and practice, and how the integration of archival theory of records and archives can further develop digital forensics as a discipline and help it in accomplishing its purposes.

Trusting the reliability and assessing the authenticity of digital files, records, documents and data is critical for law enforcement and security professionals, archivists and records managers, and all kinds of organizations and individuals. Digital forensics practitioners are tasked with finding, securing, and analyzing such material in an increasing variety of contexts. While digital forensics excels at collecting, preserving, transporting, and storing digital material, other functions of the digital forensics process are not as highly developed, for example, the ability to attribute authorship and provenance, or to analyze trustworthiness (Carrier, 2003; Cohen, 2012). The "single largest gap" in digital forensics practice has been identified as the explicit identification of information flows in investigations, for example how identity is tracked, how evidence is authenticated, or how chain of custody is maintained (Ciardhuáin, 2004). The knowledge of the digital archivist can help. A significant challenge to both forensic and archival fields is the identification of records (archival focus) and evidence (digital forensics focus) in digital systems, and establishing their contexts, provenance, relationships, and meaning. The authors present areas of cross-fertilization and introduce a new research project that will take this work further.

While the tools and technological challenges of digital forensics are determined by the medium that is the subject of analysis (for example, forensic techniques specific to mobile devices, hard drives, or networks), and those of archival science are determined by the nature and characteristics of the information that it is meant to control, preserve and make accessible, there are common theoretical underpinnings. The digital forensics specialist is concerned with identifying digital objects and traces that may serve as evidence of criminal or other activity, and analyzing those objects for their evidentiary capacity, that is, for their attribution, integrity, and verifiability. Privileged or confidential information must also be identified and protected from unauthorized disclosure. The digital archivist is concerned with identifying digital objects that have been created as records of actions and transactions, facts and events, and assessing their reliability, authenticity, and accuracy in order to guarantee a trustworthy memory and historical accountability (The archivist defines 'record' with distinct specificity: in archival science a record is a document (i.e. recorded information) made or received in the course of a practical activity, and saved for future action or reference. Records serve accountability – both legal-administrative and historical; are the basis of future decision-making, are evidence of past events, actions and transactions etc., and must satisfy admissibility requirements when submitted as evidence at trial). When an archivist acquires material from a digital storage device or network for appraisal and accessioning into a trusted repository, it is critical that s/he be able to uniquely identify the records, analyze them to ascertain their provenance, assess their authenticity and accuracy, establish existing issues regarding intellectual property, copyright, legal privilege, or track personal information that will be subject to redaction, data privacy protection, or access restrictions.

In assessing the identity and integrity of records stored in a variety of digital media, attesting to their accuracy, locating and protecting sensitive information, and acquiring them without alteration, archivists are required to act as forensic investigators. Digital forensics experts are similarly called to act as archivists when identifying, describing and preserving digital documentary evidence. Each domain possesses skills necessary and relevant to the other. Digital forensics has already claimed its place among the tools of archival processing of digital cultural heritage holdings (John, 2008; Rogers & John, 2013). Memory forensics tools are being used in digital archives and libraries for image capture, analysis, and reporting with increasing sophistication (c.f. The British Library, Stanford University Library, Emory University to name but a few sites employing these tools).

The theoretical connections between archival science and digital forensics are still being explored. The principle of provenance is generally considered the foundational principle or theory of archival science. Through analysis and description of the provenance of records archivists can assess "the source, authority, accuracy, and value of the information which [the records] contained for administrative, legal (including access to information), research and cultural uses" (Abukhanfusa & Sydbeck, 1994). The application of the principle of provenance is widely discussed in the archival community, and the complexity of digital records and data, and of the digital information systems containing them has encouraged archivists to explore how definitions and uses of provenance are employed in related disciplines (Niu, 2013). In the digital environment, provenance information (also known as data lineage) has a wide range of critical application areas, and generally involves ownership information and process history documentation. However, the issue of "secure provenance," that is, providing assurances of integrity, confidentiality, and availability to the provenance records themselves is lacking. Furthermore, secure provenance "is the essential bread and butter of digital forensics and post-incident investigation of intrusions" (Hasan, Sion, & Winslett, 2007). Archival theory about records' provenance and provenance analysis can bolster digital forensics practice in this area – to give one example.

The following discussion presents one experiment of integration of knowledge between archival science and digital forensics. The Digital Records Forensics Project (The Digital Records Forensics (DRF) project was initiated in 2008 by Luciana Duranti, Principal Investigator, School of Library, Archival and Information Studies and Anthony Sheppard, Co-Investigator, Faculty of Law (see *www.digitalrecordsforensics.ca*). DRF was a three-year research collaboration between the School of Library, Archival and Information Studies and the Faculty of Law at the University of British Columbia, and the Computer Forensics Division of the Vancouver Police Department funded by the Social Sciences and Humanities Research Council (SSHRC) of Canada), whose purpose was to adapt digital forensics methods for the archival purpose of assessing and maintaining the trustworthiness of digital records, developed a draft integrated model of an archival-forensic process (DRF model). In particular, the goal of the model was to identify points at which complementary knowledge might aid the investigative process, whether the investigation were archival in nature, aiming to preserve trustworthy sources of societal memory, or forensic, aiming to solve a crime or cybersecurity event. The DRF model, particularly well-suited to explain a workflow of "memory forensics," is descriptive and retrospective in nature, in that it seeks to abstract a workflow or process from observation of existing situations. It can then be used prospectively to identify points of weakness in the design of new processes, and to propose solutions to specific problems or issues.

Many models have been put forward to explain the digital forensic investigative response process (Beebe & Clark, 2005; Blackwell, 2011; Carrier & Spafford, 2003; Ciardhuáin, 2004; Ieong, 2006; Kahvedžić & Kechadi, 2009; Reith, Carr, & Gunsch, 2002; Selamat, Yusof, & Sahib, 2008). However, for the purposes of the Digital Records Forensics project, the researchers sought an abstracted model that included the basic elements of a digital investigation in a general framework. Carrier and Spafford have presented such a model, approaching the problem from a point of view of the computer as a crime scene, subject to crime scene investigative techniques (Carrier & Spafford, 2003). The investigation of this digital crime scene is broken down into six phases: preservation, survey for digital evidence, documentation of the evidence and the scene, search for digital evidence, reconstruction of events, and presentation of the reconstruction theory. Documentation that reports on provenance, that is, where the evidence originated and how it was handled, is key to a forensically sound case. "In addition to characteristics of the evidence source, such as a computer hardware clock or the number of sectors of a hard drive, an audit log and chain of custody enable an independent examiner to authenticate the evidence and assess its integrity and completeness" (Casey, 2007). Forensic soundness, expressed through reporting of secure provenance, provides accountability.

Digital forensics specialists are bound by the demands of the scientific method to justify their tools and techniques in identifying and authenticating digital evidence. Scientific testimony presented at trial may be tested for credibility against four criteria:

- Has the theory or technique been reliably tested?
- Has the theory or technique been subject to peer review?
- What are the theories' or techniques' known or potential error rates?
- Has the theory or technique been generally accepted as a standard in its scientific community? (Marsico, 2005)

Digital objects are therefore not examined as documentary residue of business activity—as is the case when archivists conduct such examination, but as latent trace evidence of digital processes. They are bound not by business rules and procedures, but by "the physics of digital information," which governs "the artificial digital world of bits and machines that operate on them" (Cohen, 2011). It is the physics of digital information that is the scientific grounding of the digital forensic examiner and the source of expanded understanding of provenance and other information for the digital archivist.

The roots of authority conferred upon archival and digital forensics professionals derive from the particular ontological view each has of the evidence they seek to authenticate. Despite their different perspectives on analysis of digital material, however, their investigative goals are the same: to identify and authenticate digital evidence of actions and events. To that end, examiners from either profession must establish, document, and be prepared to justify, or account for, the identity, integrity, and context of the evidence, and their role in discovering and describing it.

The archival model of management of digital resources throughout their existence is termed the Chain of Preservation (CoP) (The Chain of Preservation was developed by InterPARES to model all the functions and activities required from the moment of record creation throughout the record's life cycle necessary to ensure that records are created reliable and maintained authentic over time and across technological change), and was developed by the InterPARES (International Research on the Preservation of Authentic Records in Electronic Systems) Project (1998-2012) (Duranti & Preston, 2008). The purpose of the DRF model is to unify in one model the concepts of digital forensics practice that have been previously captured in several process models and incorporate in it the InterPARES Chain of Preservation (CoP) model for managing records throughout their life cycle. The Digital Records Forensics research team modeled the process of conducting a digital forensics investigation in order to assess the moments in which records, as understood by archival science and laws of evidence, could be identified, their authenticity assessed, and their reliability and integrity managed and preserved. The intention was to integrate the core requirements of digital forensics of establishing, documenting, and protecting the chain of custody, with the CoP model for preservation of digital records that can be presumed authentic and maintained reliable. The activities represented in the model are intended to ensure the creation and/or collection/acquisition of trustworthy digital records to be used as evidence, their maintenance throughout the judicial process, and their preservation over the long term for accountability, reference, further action, or societal memory.

The DRF model has within its scope all the phases or stages in the lifecycle of digital material that may be subject to forensic analysis in the process of investigation of a crime or security incident. It situates this material in the context of a juridical system and considers the whole process of investigation as a balance among available inputs, constraints or controls on the investigation, mechanisms used in the investigation, and desired outcomes or outputs from the investigation. As well, this model will seamlessly adapt to apply digital forensics knowledge to archival processing of digital material, thereby showing the integration of knowledge in both directions.

OVERVIEW OF THE DIGITAL RECORDS FORENSICS PROCESS MODEL

Both the DRF model and the CoP model use the IDEFØ function modeling method,
a graphical representation of the decisions, actions, and activities of an organization or system in order to analyze and communicate the functional perspective of that organization or system. Released by the National Institute of Standards and Technology (NIST) in 1993 as a standard for Function Modeling [34], it proceeds in a top-down, general-to-specific modeling approach which results in a hierarchical series of diagrams that gradually increase the level of detail in describing functions or activities and their interfaces within the context of a system. The most general features come first in the hierarchy, as the whole top-level activity is decomposed into sub-activities comprised in it. Those sub-activities may be further decomposed until all the relevant details of the system being modeled are adequately exposed and described. Each box represents a single function or activity to be modeled. For each function or activity, the inputs, controls, outputs, and mechanisms are identified. Inputs are information, materials, objects, or data that are consumed, or transformed by the activity to produce outputs. Controls are conditions required to produce the correct output. Controls impose rules that regulate the performance of an activity. Mechanisms are the physical resources or means used to perform or facilitate the activity. They may be people, infrastructure, or equipment. Outputs are information, materials, objects, or data that are produced by the activity. If an activity does not produce any outputs, it should not be modeled.

A-0: Conduct Digital Forensics. This top-level diagram delineates the subject of the model and its overall context. The bounding arrows represent the primary inputs, controls, mechanisms, and outputs. The activities represented in this diagram and all subsequent decompositions are intended to ensure the creation and/or collection of trustworthy digital records to be used as evidence, their maintenance throughout the judicial process, and their preservation over the long term for accountability, reference, or further action (see Figure 1).

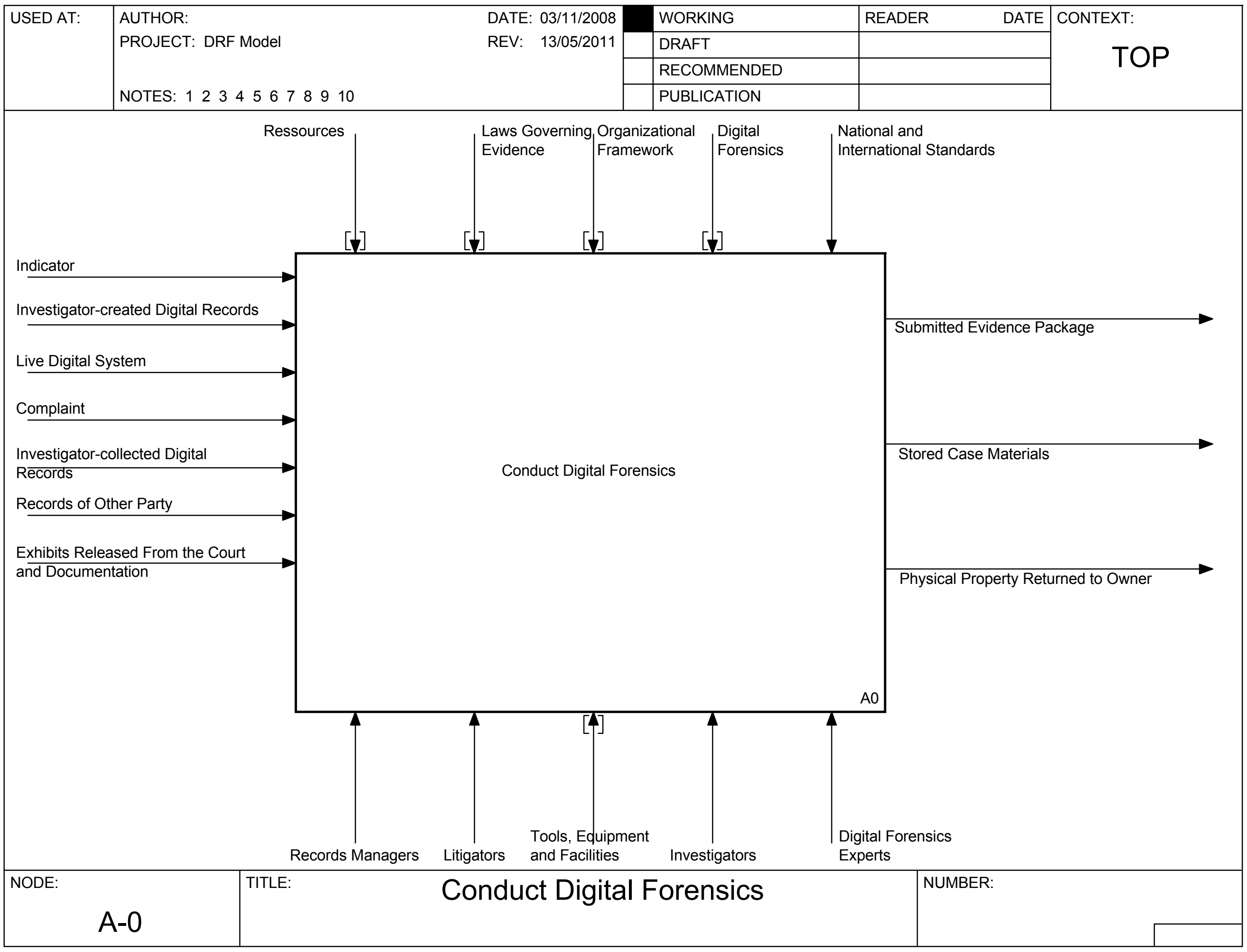

USED AT:
AUTHOR:
PROJECT: DRF Model
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 03/11/2008
REV: 13/05/2011
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
TOP
Ressources
Laws Governing Evidence
Organizational Framework
Digital Forensics
National and International Standards
Indicator
Investigator-created Digital Records
Live Digital System
Complaint
Investigator-collected Digital Records
Records of Other Party
Exhibits Released From the Court and Documentation
Conduct Digital Forensics
A0
Submitted Evidence Package
Stored Case Materials
Physical Property Returned to Owner
Records Managers
Litigators
Tools, Equipment and Facilities
Investigators
Digital Forensics Experts
NODE:
A-0
TITLE:
Conduct Digital Forensics
NUMBER:

Figure 1. *Conduct Digital Forensics A-0*

WHAT ARE THE CONSTRAINTS ON THE DIGITAL RECORDS FORENSICS PROCESS?

Digital forensics is always conducted within the context of constraints and controls imposed by the juridical system in which the investigation takes place, the resources available to undertake the investigation, and the principles of digital forensics that are recognized through methodological and theoretical development of the discipline.

Resources available to the investigator include personnel, financial support, tools and technology, and specialized, or domain knowledge.

Digital Forensics Principles have developed to support the purpose of digital forensics investigations. They have been summarized by a variety of domain experts, and include under the guiding principle: "Action taken to secure and collect electronic evidence should not change that evidence" (US Department of Justice, 2001) concepts of

- Integrity
- Authentication
- Reproducibility
- Non-interference
- Minimalization

These principles are themselves governed by the laws of evidence, and relevant national and international standards. Finally, the investigation will also be constrained by the organizational framework within which it directly takes place.

WHAT ARE THE MECHANISMS INSTRUMENTAL TO THE DIGITAL RECORDS FORENSICS PROCESS?

Many resources are required to conduct a successful digital forensic investigation. Most commonly, these will include litigators, investigators, digital forensics experts, and the tools, equipment and facilities they use. The model recognizes that, in a digital records forensic process, records managers also play an important role in identifying records in context and offering domain expertise in records related issues such as privacy, assessment of authenticity, reliability, and accuracy, and requirements for preservation and access.

WHAT ARE THE IMPORTANT INPUTS TO THE DIGITAL RECORDS FORENSICS PROCESS?

By definition, the inputs at the top level of the model represent information or objects that originate outside of the activity being modeled. In a digital records forensics investigation of a crime or system breach, an indicator is required – some information about unusual, suspicious, or criminal activity. The indicator may result in a complaint – a written or oral request to investigate. The activity may be conducted on a live digital system, on digital materials collected by an investigator, or materials produced by the other party. The activity may also be supported by records authored by the investigator, or by exhibits released from the court with their accompanying documentation.

WHAT ARE THE KEY OUTPUTS OF THE DIGITAL RECORDS FORENSICS PROCESS?

Many different outputs may proceed from the top level activity, but all can be categorized as evidence submitted to counsel or to court, materials stored or preserved from the case and its investigation, and physical property that may be returned to its rightful owner at the completion of the investigation or trial.

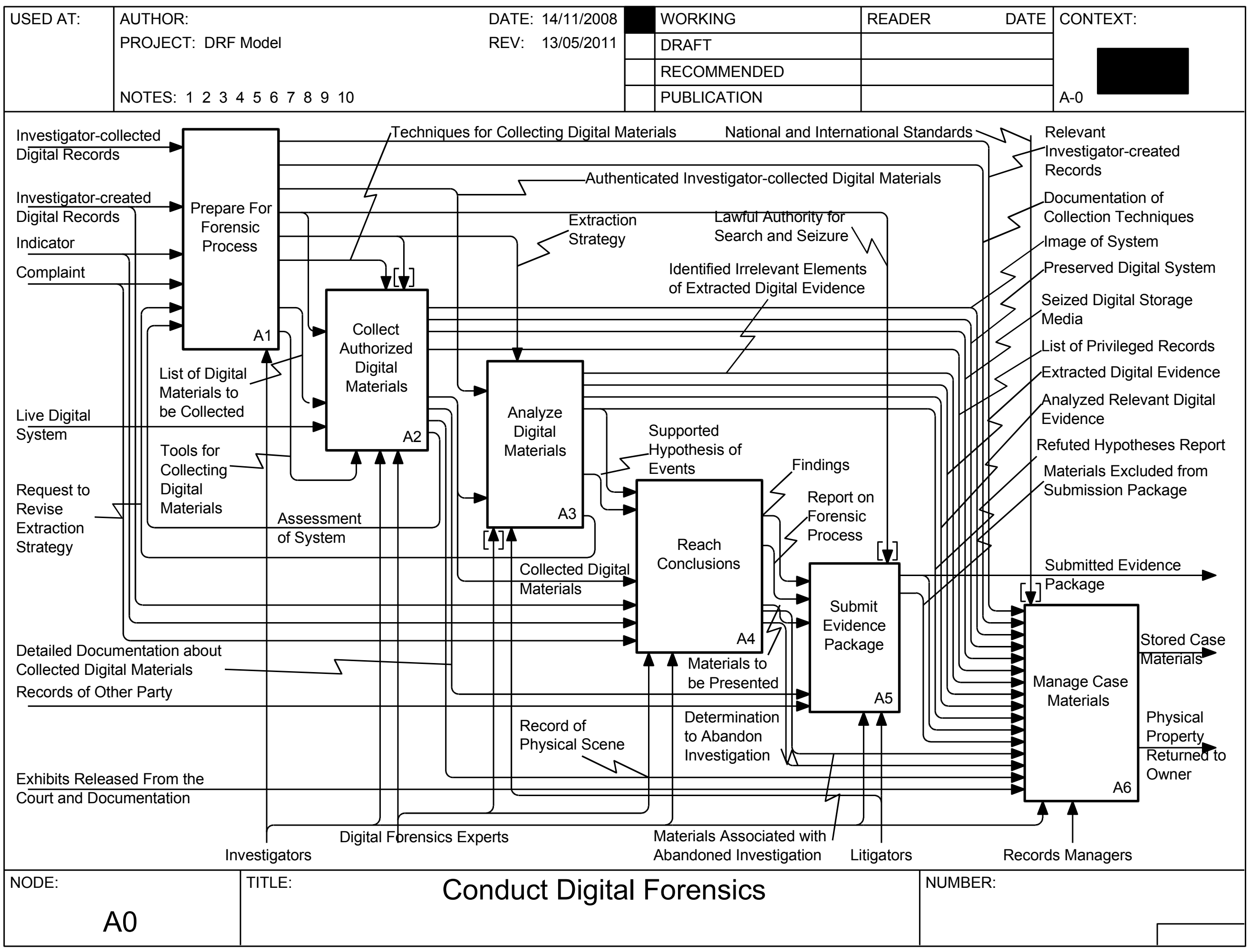

USED AT:
AUTHOR:
PROJECT: DRF Model
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 14/11/2008
REV: 13/05/2011
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A-0
Investigator-collected Digital Records
Investigator-created Digital Records
Indicator
Complaint
Prepare For Forensic Process
A1
Techniques for Collecting Digital Materials
National and International Standards
Authenticated Investigator-collected Digital Materials
Extraction Strategy
Lawful Authority for Search and Seizure
Identified Irrelevant Elements of Extracted Digital Evidence
Collect Authorized Digital Materials
A2
List of Digital Materials to be Collected
Live Digital System
Tools for Collecting Digital Materials
Request to Revise Extraction Strategy
Assessment of System
Analyze Digital Materials
A3
Supported Hypothesis of Events
Findings
Report on Forensic Process
Reach Conclusions
A4
Collected Digital Materials
Submit Evidence Package
A5
Detailed Documentation about Collected Digital Materials
Records of Other Party
Materials to be Presented
Determination to Abandon Investigation
Record of Physical Scene
Exhibits Released From the Court and Documentation
Relevant Investigator-created Records
Documentation of Collection Techniques
Image of System
Preserved Digital System
Seized Digital Storage Media
List of Privileged Records
Extracted Digital Evidence
Analyzed Relevant Digital Evidence
Refuted Hypotheses Report
Materials Excluded from Submission Package
Submitted Evidence Package
Manage Case Materials
A6
Stored Case Materials
Physical Property Returned to Owner
Investigators
Digital Forensics Experts
Materials Associated with Abandoned Investigation
Litigators
Records Managers
NODE:
A0
TITLE:
Conduct Digital Forensics
NUMBER:

Figure 2. *Conduct Digital Forensics A0*

The model distinguishes six main activities (Figure 2): 1) Prepare for forensic analysis; 2) Collect authorized digital materials; 3) Analyze digital materials; 4) Reach Conclusions; 5) Submit evidence package; and 6) Manage case materials.

INTEGRATING DIGITAL FORENSIC AND ARCHIVAL PRACTICE MODELS

The traditional archival practice of ensuring the authenticity of records over time through evidence of an unbroken chain of custody alone is inadequate for digital records. The creation, maintenance, and preservation of digital records that can be proven reliable and presumed authentic over time and across technological change rely on processes and controls that protect them from corruption and maintain their identity and integrity (Duranti & Preston, 2008). Mapping activities in the DRF model to the CoP model allows for greater cross-disciplinary understanding of common terminology, and identifies moments at which domain knowledge may be shared to enhance the process of analysis of digital material.

The following example illustrates this process, and the possibilities for further research. The CoP model distinguishes four main record activities: (1) managing the framework for the chain of preservation, (2) managing the process of records creation, (3) managing records in a recordkeeping system, and (4) preserving selected records (Duranti & Preston, 2008). This example compares activity decomposition from the fourth activity of the CoP model (A4 Manage Records in a Permanent Preservation System) with the activity decomposition from the third activity area of the DRF model (A3 Analyze Digital Materials).

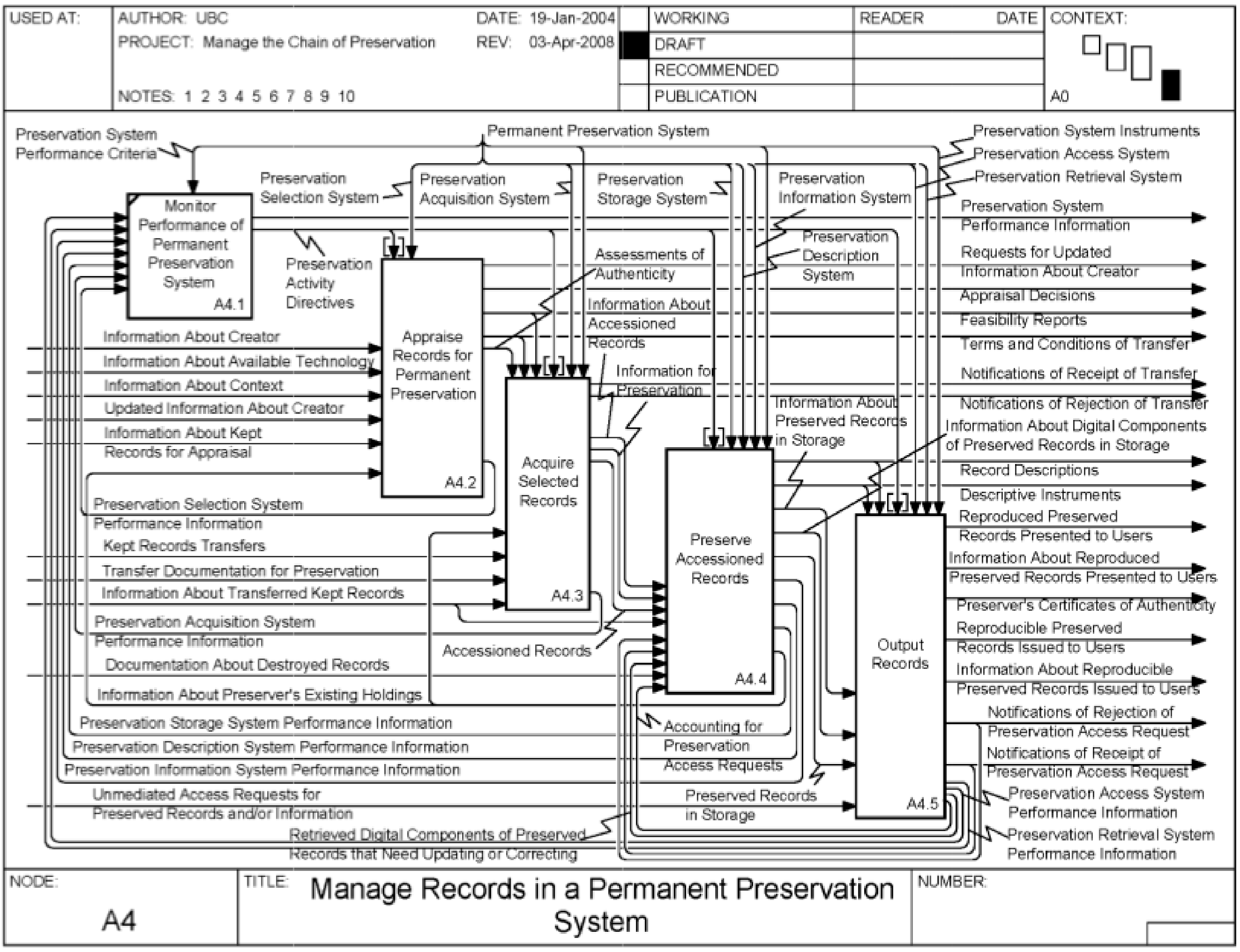

USED AT:
AUTHOR: UBC
PROJECT: Manage the Chain of Preservation
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 19-Jan-2004
REV: 03-Apr-2008
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
Preservation System Performance Criteria
Permanent Preservation System
Preservation System Instruments
Preservation Access System
Preservation Retrieval System
Preservation Selection System
Preservation Acquisition System
Preservation Storage System
Preservation Information System
Monitor Performance of Permanent Preservation System
A4.1
Preservation Activity Directives
Preservation Description System
Preservation System Performance Information
Requests for Updated Information About Creator
Appraisal Decisions
Feasibility Reports
Terms and Conditions of Transfer
Notifications of Receipt of Transfer
Notifications of Rejection of Transfer
Assessments of Authenticity
Information About Accessioned Records
Information About Creator
Information About Available Technology
Information About Context
Updated Information About Creator
Information About Kept Records for Appraisal
Appraise Records for Permanent Preservation
A4.2
Information for Preservation
Information About Preserved Records in Storage
Information About Digital Components of Preserved Records in Storage
Record Descriptions
Descriptive Instruments
Reproduced Preserved Records Presented to Users
Information About Reproduced Preserved Records Presented to Users
Preserver's Certificates of Authenticity
Reproducible Preserved Records Issued to Users
Information About Reproducible Preserved Records Issued to Users
Notifications of Rejection of Preservation Access Request
Notifications of Receipt of Preservation Access Request
Preservation Access System Performance Information
Preservation Retrieval System Performance Information
Acquire Selected Records
A4.3
Preservation Selection System Performance Information
Kept Records Transfers
Transfer Documentation for Preservation
Information About Transferred Kept Records
Preservation Acquisition System Performance Information
Documentation About Destroyed Records
Information About Preserver's Existing Holdings
Accessioned Records
Preserve Accessioned Records
A4.4
Output Records
A4.5
Preservation Storage System Performance Information
Preservation Description System Performance Information
Preservation Information System Performance Information
Accounting for Preservation Access Requests
Unmediated Access Requests for Preserved Records and/or Information
Preserved Records in Storage
Retrieved Digital Components of Preserved Records that Need Updating or Correcting
NODE:
A4
TITLE:
Manage Records in a Permanent Preservation System
NUMBER:

Figure 3. *Manage Records in a Permanent Preservation System (CoP)*

Figure 3 shows the overview of the activity, *Manage Records in a Permanent Preservation System*, from the CoP model. This activity involves actions associated with preserving records to ensure their continuing authenticity while in the custody of the designated preserver. Key activities include appraisal and selection of records of permanent value, capture, preservation, description, and output of selected records. These activities may be mapped to the activities involved in analyzing digital materials – A3 of the DRF model. For example, the process of record appraisal and acquisition (CoP A4.2-4.3) shares features and purposes of preparation and extraction of digital material (DRF A3.1-3.2) (see Fig. 4). By integrating the archival principles embedded in the CoP model that ensure records' authenticity and reliability into the principles of digital forensics that guarantee integrity, authentication, and verifiability of digital material, patterns may be developed that solve issues challenging both domains.

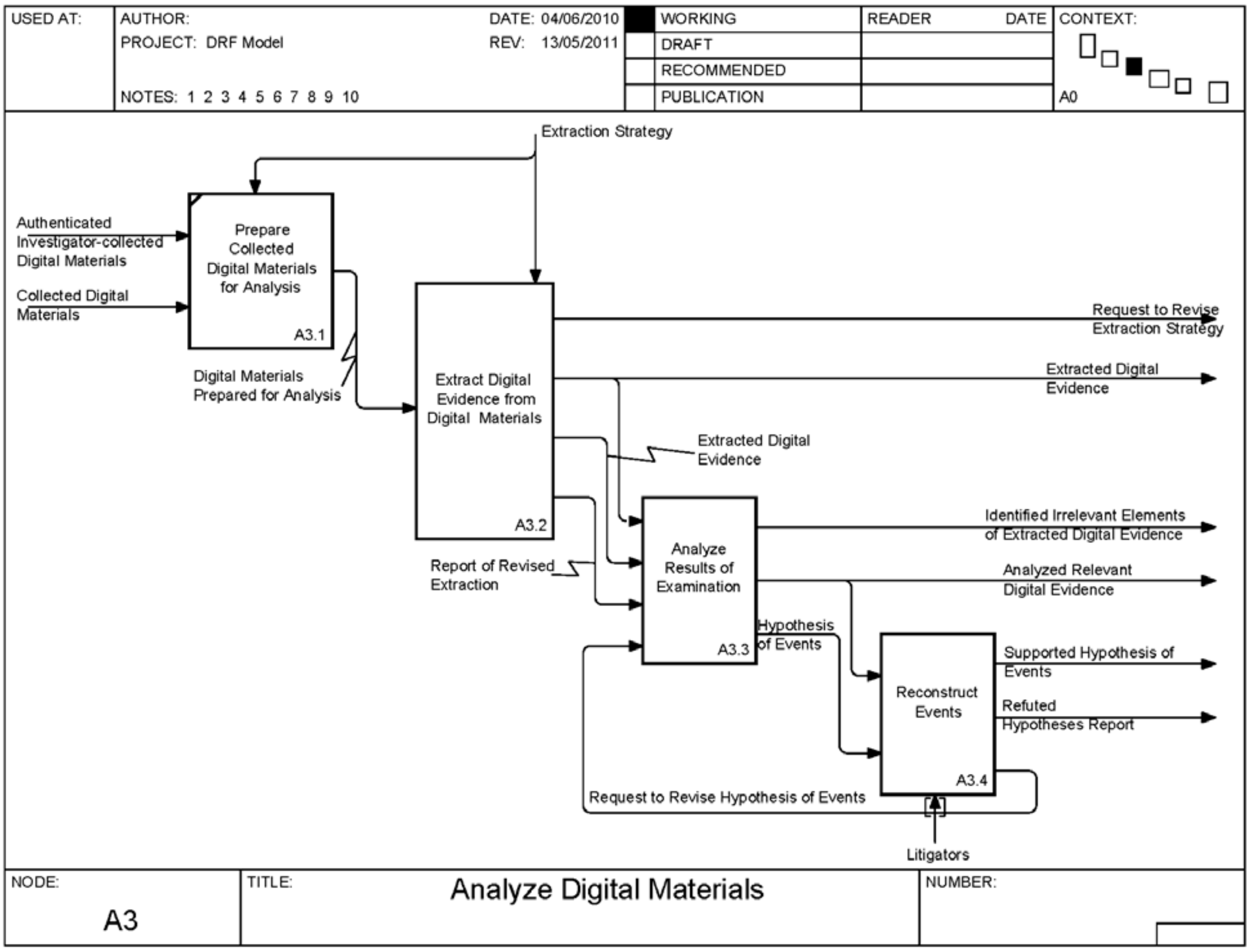

USED AT:
AUTHOR:
PROJECT: DRF Model
DATE: 04/06/2010
REV: 13/05/2011
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
Extraction Strategy
Authenticated Investigator-collected Digital Materials
Collected Digital Materials
Prepare Collected Digital Materials for Analysis
A3.1
Digital Materials Prepared for Analysis
Extract Digital Evidence from Digital Materials
A3.2
Request to Revise Extraction Strategy
Extracted Digital Evidence
Extracted Digital Evidence
Report of Revised Extraction
Analyze Results of Examination
A3.3
Identified Irrelevant Elements of Extracted Digital Evidence
Analyzed Relevant Digital Evidence
Hypothesis of Events
Reconstruct Events
A3.4
Supported Hypothesis of Events
Refuted Hypotheses Report
Request to Revise Hypothesis of Events
Litigators
NODE:
A3
TITLE:
Analyze Digital Materials
NUMBER:

Figure 4. *Analyze Digital Materials (DRF)*

NEXT STEPS – INTERPARES TRUST

The CoP model used to map to the general model of the digital records forensics process is undergoing review and revision as a result of the rapid adoption of cloud services in the creation, management, storage and preservation of digital records. Modeling the Chain of Preservation specifically for records in the cloud is necessary to address concerns over issues related to jurisdiction, privacy, security, authenticity, validity, integrity, and completeness. Thus, in April 2013, a new international multidisciplinary project involving universities, governments, businesses, and cultural heritage institutions in six continents and thirty countries began its research activities aimed to address such issues. This project, which is funded until 2019 by the Social Sciences and Humanities Research Council of Canada and by all participating partners, is the fourth phase of InterPARES research, and addresses the issue of trust in records and archives created, used, maintained and/or permanently preserved online, thereby taking the name of InterPARES Trust (ITrust).

The goal of ITrust is to generate the theoretical and methodological frameworks that will support the development of integrated and consistent local, national and international networks of policies, procedures, regulations, standards and legislation concerning digital records entrusted to the Internet, to ensure public trust grounded on evidence of good governance, a strong digital economy, and a persistent digital memory.

Reviewing the Chain of Preservation model and revising it for records and data hosted by third party service providers addresses the following questions:

- Are requirements for the preservation of digital records identified in InterPARES 1 and 2 applicable to records in the cloud?
- What additional requirements does forensic readiness impose on preservation of records in the cloud?

- How can these requirements be satisfied when records are stored by third party service providers?
- Are there special requirements for records that are discovered and delivered via the internet?
- How can such requirements be implemented?

Because records and archives entrusted to third party providers must satisfy the requirements of reliability, authenticity (i.e. identity and integrity), accuracy, usability, accessibility, and preservability, so that transparency and accountability (legal, administrative, and historical) are ensured, and documentary evidence is protected together with the documentary sources for history, the collaboration of all disciplines concerned with these qualities of records and archives is necessary to the success of InterPARES Trust, and digital forensics has a special role in determining the outcome of this research project.

The theory and methods identified to reach the objectives of InterPARES Trust are those of archival science, resource management, policy design, textual analysis, visual analytics, risk management, and modeling. Although digital forensics is not expressly indicated in the research proposal, it is not an absentee; rather, digital forensics is a stone guest. The ancient expression "stone guest" refers to a looming but invisible presence, silent and therefore disturbing and unpredictable, of which everyone is aware but which no one mentions. While it is clear that digital forensics practices and procedures would be useful, if not outright necessary, in carrying out this research project, it is difficult when outlining a theoretical and methodological approach to research to refer to specific activities or processes rather than to the body of knowledge of a recognized discipline. And digital forensics is hardly perceived as an autonomous discipline.

There is a vast literature on the concept of discipline that proposes very different definitions and interpretations. Liles et al. (1995) build upon the analysis of the existing definitions and suggest that a discipline must have "six basic characteristics: *(1) a focus of study, (2) a world view or paradigm, (3) a set of reference disciplines used to establish the discipline, (4) principles and practices associated with the discipline, (5) an active research or theory development agenda, and (6) the deployment of education and promotion of professionalism*" [italics in the original text]. Digital forensics has some of these characteristics but what separate it from a full-fledged discipline are its reactive approach, and its retrospective outlook, which confines it to the examination of what exists.

Several decades after its recognition as an established practice, digital forensics has accumulated a large body of knowledge that can allow it to identify recurring concepts, ideas, and principles capable of guiding the design of systems for data, records and archives created and/or kept by third party service providers, systems that do not have to trade transparency for safety, or control for economy. Much has still to be done to ensure that digital forensics knowledge can be used to prevent rather than to detect cybercrime, but the key is active collaboration with allied disciplines in the context of multidisciplinary projects like InterPARES Trust. Just as archival science is expanding its body of theory to incorporate knowledge from digital forensics, digital forensics experts can benefit from the study of concepts, laws and models from the other fields involved with the InterPARES research project to foster useful transfers to their own field, to encourage the development of a digital forensic theory in emerging areas of endeavor and investigation, to eliminate the duplication of theoretical efforts in different fields, and to promote consistency of scientific knowledge.

However, in order to develop the knowledge of digital forensics, when experts bring those extraneous concepts, laws and models into their body of knowledge, they have to make them consistent with all of its parts (i.e., confront them with forensics concepts, principles, practice and scholarship), subject them to a feedback process, and insert them into the fundamental structure of their knowledge system. Only in this way will they be able to build up digital forensics as a discipline, maintaining its integrity and continuity while at the same time fostering its enrichment and growth. This paper is an invitation to start this process of growth and change and to do it by helping records professionals to ensure that records and archives in the cloud can be protected without renouncing transparency, accountability, and accessibility…in a word, democracy.

REFERENCES

- Abukhanfusa, K., & Sydbeck, J. (Eds.). (1994). The Principle of Provenance: Report from the First Stockholm Conference on Archival Theory and the Principle of Provenance, 2-3 September 1993. Stockholm.
- Beebe, N. L., & Clark, J. G. (2005). A hierarchical, objectives-based framework for the digital investigations process. Digital Investigation, 2(2), 147–167.
- Blackwell, C. (2011). A Framework for Investigating Questioning in Incident Analysis and Response. Oxford, UK.
- Carrier, B. (2003). Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers. International Journal of Digital Evidence, 1(4), 1–12.
- Carrier, B., & Spafford, E. (2003). Getting Physical with the Digital Investigation Process. International Journal of Digital Evidence, 2(2), 1–20.
- Casey, E. (2007). What does "forensically sound" really mean? Digital Investigation, 4(2), 49–50.
- Ciardhuáin, S. Ó. (2004). An Extended Model of Cybercrime Investigations. International Journal of Digital Evidence, 3(1), 1–22.
- Cohen, F. (2011). The State of the Science of Digital Evidence Examination. Presented at the IFIP 11.9, Orlando, FL: unpublished.
- Cohen, F. (2012, September 24). The Future of Digital Forensics. Presented at the Trust and Conflicting Rights in the Digital Environment, Vancouver, BC.
- Duranti, L., & Preston, R. (2008). Research on Permanent Authentic Records in Electronic Systems (InterPARES) 2: Experiential. Interactive and Dynamic Records. Padova: Associazione Nazionale Archivistica Italiana.
- Hasan, R., Sion, R., & Winslett, M. (2007). Introducing secure provenance (p. 13). ACM Press. doi:10.1145/1314313.1314318 Ieong, R. S. C. (2006). FORZA – Digital forensics investigation framework that incorporate legal issues. Digital Investigation, 3(Supplement 1), 29–36.
- John, J. (2008). Adapting Existing Technologies for Digitally Archiving Personal Lives: Digital Forensics, Ancestral Computing, and Evolutionary Perspectives and Tools. In Proceedings of The Fifth International Conference on Preservation of Digital Objects (pp. 46–55). Presented at the Joined Up and Working: Tools and Methods for Digital Preservation, London, England: The British Library. Retrieved from *http://www.bl.uk/ipres2008/ipres2008-proceedings.pdf*
- Kahvedžić, D., & Kechadi, T. (2009). DIALOG: A framework for modeling, analysis and reuse of digital forensic knowledge. Digital Investigation, 6(Supplement 1), S23–S33.
- Liles, D.H., Johnson, M. E., Meade, L. M., and Ryan, D.: Underdown. Enterprise Engineering: A Discipline? (1995), *http://webs.twsu.edu/enteng/ENTENG1.html*
- Marsico, C. V. (2005). Computer Evidence v. Daubert: The Coming Conflict. Purdue University. Retrieved from *https://www.cerias.purdue.edu/apps/reports_and_papers/view/2819/*
- Niu, J. (2013). Provenance: crossing boundaries. Archives and Manuscripts, 41(2), 105–115. doi:10.1080/01576895.2013.811426
- Reith, M., Carr, C., & Gunsch, G. (2002). An Examination of Digital Forensic Models. International Journal of Digital Evidence, 1(3). Retrieved from *http://www.worldcat.org/wcpa/oclc/223384589?page=frame&url=http%3A%2F%2Fwww.ijde.org%2F%26checksum%3D85756f448e9f3f33b58f16d99aa26bcf&title=&linktype=digitalObject&detail=*
- Rogers, C., & John, J. (2013). Shared Perspectives, Common Challenges: A History of Digital Forensics & Ancestral Computing for Digital Heritage. In The Memory of the World in the Digital Age: Digitization and Preservation (pp. 314–336). Presented at the The Memory of the World in the Digital Age: Digitization and Preservation, Vancouver, BC: UNESCO. Retrieved from *http://www.unesco.org/webworld/download/mow/mow_vancouver_proceedings_en.pdf*
- Selamat, S. R., Yusof, R., & Sahib, S. (2008). Mapping Process of Digital Forensic Investigation Framework. IJCSNS Internation Journal of Computer Science and network Security, 8(10), 163–169.
- US Department of Justice. (2001). Electronic Crime Scene Investigation: A Guide for First Responders. Washington, DC. Retrieved from *www.ojp.usdoj.gov/nij*

ABOUT THE AUTHOR

Luciana Duranti is Chair of the Master of Archival Studies at the School of Library, Archival and Information Studies of the University of British Columbia, and a Professor of archival theory, diplomatics, and the management of digital records in both its master's and doctoral archival programs. Dr. Duranti is Project Director of InterPARES Trust (2013-2018), an multidisciplinary, international research project studying issues of trust in data and records in online environments. She is active nationally and internationally in several archival associations and in boards and committees, including Italy's National Commission for Archives (2007-2013) and the UNESCO International Advisory Committee of the Memory of the World Program (2007-2013), and has been the President of the Society of American Archivists (1998-99), of which she is a Fellow. She publishes widely on archival history and theory, and on diplomatics.

Corinne Rogers and a sessional instructor (diplomatics, digital records forensics) and doctoral candidate (researching concepts of authenticity of digital records, documents, and data and authenticity metadata) at the University of British Columbia. She is Project Coordinator of InterPARES Trust – international multidisciplinary research into issues of trust in digital objects in online environments, and a researcher with the Law of Evidence in the Digital Environment Project (Faculty of Law, UBC).

MEMORY FORENSIC VS PULLING THE PLUG

by Kyriakos Loizou

From right in the beginning of Computer Forensic a set of rules, methods and protocols where established and embedded deep in to the law enforcement community. Traditionally a forensic examiner would enter the crime scene and with compliance to a court warrant he would seize a live computer systems in order to deliver it back to the laboratory for an in depth analysis. This methodology know as "pulling the plug" with in the forensic community, was highly accepted and thought to be forensically sound and applicable in court.

What you will learn:

- General understanding of importance of Memory forensics
- Techniques used in memory forensic actions
- Benefits of being able to use memory forensics
- Identification, preservation, collection, examination, analysis and presentation.

What you should know:

- Basic knowledge in digital forensics field

This protocol was established in the attempt to preserve data found on the hard disk drive(HDD) from being overwritten, modified or deleted. It has been proven by experts in the field of forensic, that even if the live system is left idling or a normal should down is attempted it is highly likely that alteration to the data on the HDD will occur.

The recent changes of the last decade in computer software and architecture brought with them a boost in application requirements and a large increase in computer memory. These new changes reshaped the entire software and hardware interaction layer and completely altered they way application use and store date on a computer system, in turn this started to give a new perspective to some forensic experts and a number of doubts where created about the way an investigation was performed.

Someone once said *With great change comes great opportunity*, although this is very true, what he forgot to mention is the fact that great change in some case can lead to greater problems. Forensic investigators these days are face with such problems when seizing a computer system from a crime scene, despite the benefits of the rapid changes in computer architecture and software these changes also oppose a grate threat to the traditional rules, method and protocols established by the forensic community. With the large increase in system memory, pulling the plug from a running system could in fact prove to be more harmful than beneficial. When a computer system losses power all volatile information on memory and else where at that point in time will be gone for every. A decade ago, when system memory was only a few megabytes in size, pulling the plug from a live computer was not thought to be of a great issue, because of the fact that not much information could be kept in memory it was considered to much of a risk to try and extract data from it, there for pulling the plug was the best option. However, in today's world this is not the case, memory usage and size has change completely, the capacity of memory found on computer systems are dramatically larger than previous years, consequently mush more volatile data can be stored in RAM at any give point in time. In turn this means that if an investigator chose to pull the plug from a live system at the time on an investigation, the amount of data(potential evidence) that could be lost is tremendous, thus creating a dilemma for the investigator whether to pull the plug or not.

In the area of forensic computing, the meaning of the word investigations is defined as the act of identifying, extracting and analysing evidence from a given computer device in order to recreate and understand actions performed from the specific system or systems thus identifying any criminal activity that in turn can be used in a civil or criminal court.

As soon as a new case opens the forensic examiner performing the investigation will be dealing with one of two types of investigations, a “live investigation” also known as live analysis, or a “dead investigation” commonly know as off the scene analysis or traditional investigation.

Both investigation types have there own seperate set of protocols, methodology and steps they follow according to the given case, the major difference between these two types of investigations is the fact the in a live investigation the examination of a devices is done on the scene while the system is running. In other words, at any give crime scene, if the computer system is found running and the case requires a live investigation then the forensic investigator will proceed to an examination while the devices is running with out powering it at any time.

On the other hand, a traditional(dead) investigation of a computer system is typical performed in a laboratory and deals with powered-off(“dead”) devices found on the suspect or the crime scene. However in some cases, when entering a crime scene, the investigator could pull the plug powering-off a running devices before transferring it to a laboratory.

In order to satisfy the requirements of all case, both types of investigations where established with unique advantages and disadvantages.

The traditional dead investigation was established where time was not highly relevant to the case, this allowed the forensic examiner to have a more in depth look at the device making sure that all the evidence will be fully identified, analysed and documented before being presented in court. Typically the procedures of a “dead investigation” consist of 6 main steps and follow a waterfall cycle as shown below in “Figure 1”. These 6 steps are, identification, preservation, collection, examination, analysis and presentation.

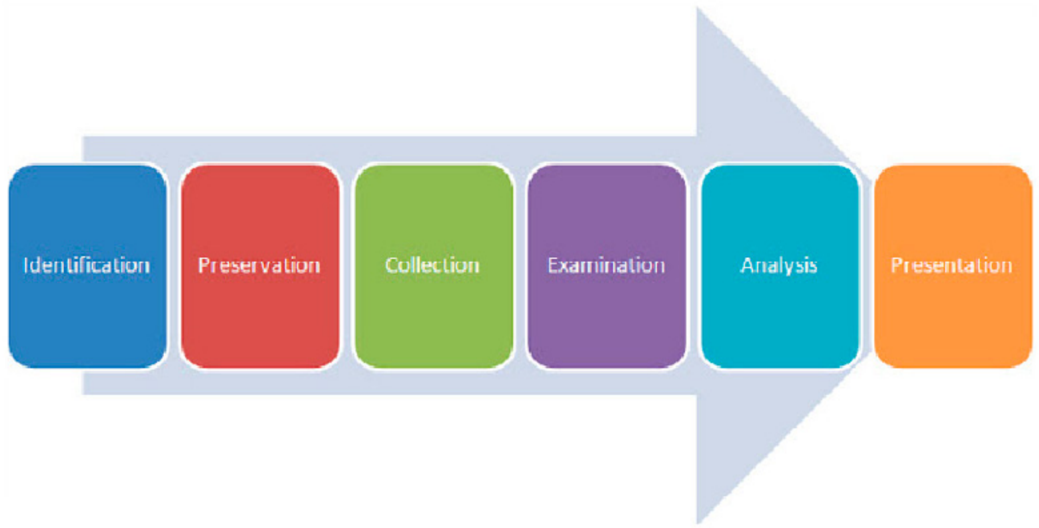


Figure 1. *Typically the procedures of a “dead investigation” consist of 6 main steps and follow a waterfall cycle*

As mentioned earlier, one of the major benefits of this procedure is that it gives the investigator a large amount of time in order for him to perform a thorough examination of the computer system in question. Another benefits of a dead investigation is the write-blockers found in the lab and used by the examiner in order to make sure that no information is written on the computer system, this preserves data integrity and makes it admissible in court.

Although "dead investigations" as we mentioned earlier have many benefits, they also come with some major drawbacks. Two of the biggest drawbacks a forensic examiner will be faced with when dealing with this type of investigation are firstly, the large capacity of hard drive found on computer system these day that have proven to be extremely time consuming when fully examined in order to identify and analysed evidence. Secondly and maybe most importantly, all of the volatile information that was located on a computer system is not recoverable because of the fact that the system lost power, this is the case even if the devices was off at the time of the investigation or the plug was pulled by the investigator after entering the crime scene.

Because of the problems created by the traditional investigation procedures, a new examination method was created and the live investigation procedures was established.

The aim of a live investigation is to try and collect evidence from a running computer system where time is of an essence for the case or where the devices can't be powered of because of the required volatile data sorted in RAM or temporary files.

As seen in "Figure 2." below, when dealing with a live investigation the procedure states that the first step to take is to collect the evidence from the running computer system, after performing the collection of evidence the plug is pulled and the devices is transferred to the laboratory where "Figure 1." procedure is then applied.

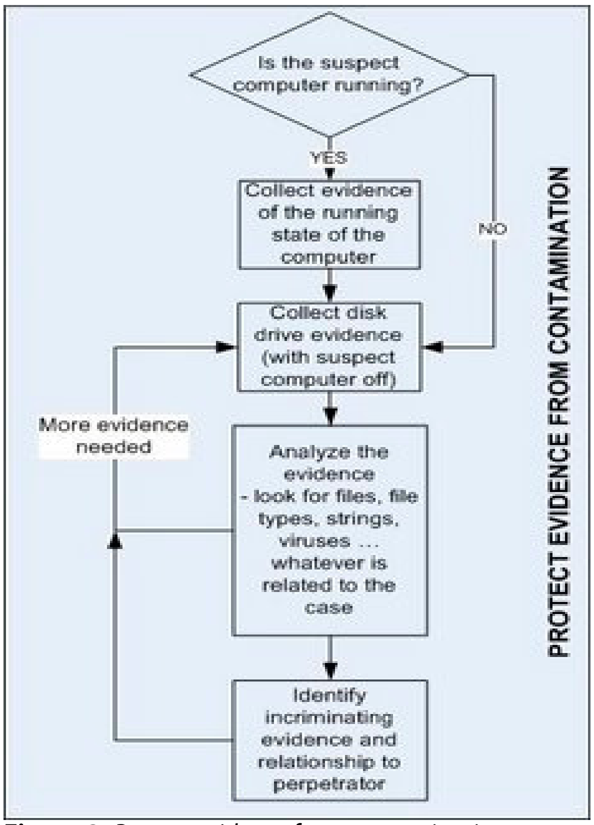

Is the suspect computer running?
YES
NO
Collect evidence of the running state of the computer
Collect disk drive evidence (with suspect computer off)
More evidence needed
Analyze the evidence - look for files, file types, strings, viruses ... whatever is related to the case
Identify incriminating evidence and relationship to perpetrator
PROTECT EVIDENCE FROM CONTAMINATION

Figure 2. *Protect evidence from contamination*

A live investigation has some very important benefits, imagine for example a case involving a kidnapping where important information found on the computer system may reveal the location of the victim, a live examination would then be applied and the data collected from the devices. Another example for when a live investigation is required would be when volatile information such as, running processes or records of recently opened documents are needed for the nature of the case, there for the devices can't be power-off.

Although a live investigations can prove to be highly beneficial, the biggest drawback when dealing with such scenario is the fact that with this type of examination certain changes will be made to the computer system, if the forensic examiner is not 100% sure of his actions, he may end-up altering the evidence found on the devices there for making it inadmissible in court.

Now that we have made ourselves familiar with the benefits of both dead and live investigation and after having had a brief overview of the procedures uses to perform each individual examination, we will continue on by having a more in-depth look at the method of "pulling the plug" that we have mentioned earlier.

In the computer forensic community, especially when talking about methods and procedures of an investigation we will surely come across the term "pulling the plug". This term is used to describe the act of pulling the power cable of a running computer system from the socket shortly after the investigator enters the crime scene thus forcing the devices to lose power and power off.

When entering a crime scene, if faced with a running devices that had to be transferred back to the forensic laboratory for an in depth examination, the forensic expert performing the investigation would typically pull the plug from the wall socket and power off the devices. This method was established back in the beginning of computer forensic in the attempt to try and preserve the data found on the give computer system from being modified or removed by running process, application or the conventional shout-down procedure.

This procedure was traditionally performed shortly after the investigator enter the crime scene and identified the computer system/systems in question. Depending on the requirements of the case.

Because most forensic investigators are aware of the fact that a conventional shout-down of a computer system may potentially lead to loss or alteration of valuable information located on the devices, there for the method of pulling the plug was accepted throughout the forensic community to be the most sufficient. It was the experts general belief that an investigator should not go to extraordinary effort in order to extract volatile data stored in RAM or else where on the devices.

When pulling the plug from a running devices any malicious applications and scripts written or install by the suspect don't get executed. Because of the fact that all running processes on the computer system are stopped unexpectedly, when forcing the devices to power of by pulling the plug, these application will not be executed therefore protecting potential evidence from being lost or altered.

Apart from the risks of malicious applications being executed by conventionally powering of the computer system, another reason that was thought to justify pulling the plug of a running devices was the fact that, because some applications write temporary files to the hard disk in order to improve functionality while in use, when these applications finish there processes or when the system is shout-down these files are removed in order to free space from the hard disk. Although these files are removed when the devices is shout-down conventionally, this is not the case when pulling the plug an forcing the system to power off.

By taking under consideration the potential benefits of pulling the plug from a given computer system, we can clearly see what made the forensic experts establish and use this method when dealing with cases where a computer system was found running at the scene of the crime.

As we have see, because of the advantaged of pulling the plug, this method is still widely used in today's investigation. Although this is one of the first steps a forensic investigator is trained to follow when face with a live system, the recent changes of the last decade in computer software and hardware have started to make forensic expert question this process.

About a decade ago when computers started to become popular in almost all homes around the world, a typical HDD was only a couple of Gb in capacity and memory was at most 1 Gb in size. As computer architecture advanced over the last years, along with it came a dramatic increase in HDD and RAM capacity.

Memory capacity these days of average computer systems found in every day homes and workplaces can start from a minimum of 512Mb going up to 64Gb and in some case maybe even.

Because of the changes in hardware that we mentioned earlier, a change is application development also came with it. Software developers are now creating application that take advantage of the large amount of memory capacity. For various reasons, manly because of security and performance, most applications these day store information in memory while running or in some cases even after the have been closed.

By looking closely at these changes both in computer hardware and software, we should now be able to see why these advances in technology start to create a problem for the forensic investigators and make experts question previously established methods and procedures.

We have seen that up till now, the preferred method used when dealing with a live system during an investigation was to pull the plug and proceed with transferring the device back to a forensic laboratory for further analysis. If we take under consideration the changes in software development in regards to the way they take advantage of RAM and look at the fact that memory on any devices these day could go up to 64Gb, then it should now be clearly that we have a problem with the method of pulling the plug from a running devices.

Pulling the plug was a well justified method back in the day where memory capacity could store only a couple of Mb. Back then if the forensic investigator made the choice of pulling the plug an a running device, because of the size of memory and because of the way applications where developed not much data would be lost. However, in today's world this is not the case. With such a large amount of memory capacity, the amount of information that could be stored in memory at any given point in time may be massive, therefore by pulling the plug from a running device could lead to evidence being lost and may potentially prove to be devastating for a case. When we start putting together all of what we have mentioned earlier, we will soon start asking the questions "*When dose an investigation pull the plug* ?", "*How should an investigator make the decision whether to pull the plug or not ?*". In order to deal with this dilemma an investigator must perform a computer triage right from the beginning of the case.

In "Figure 3." we can see a diagram of an investigation procedure using triage before start the extortion of information from the given devices.

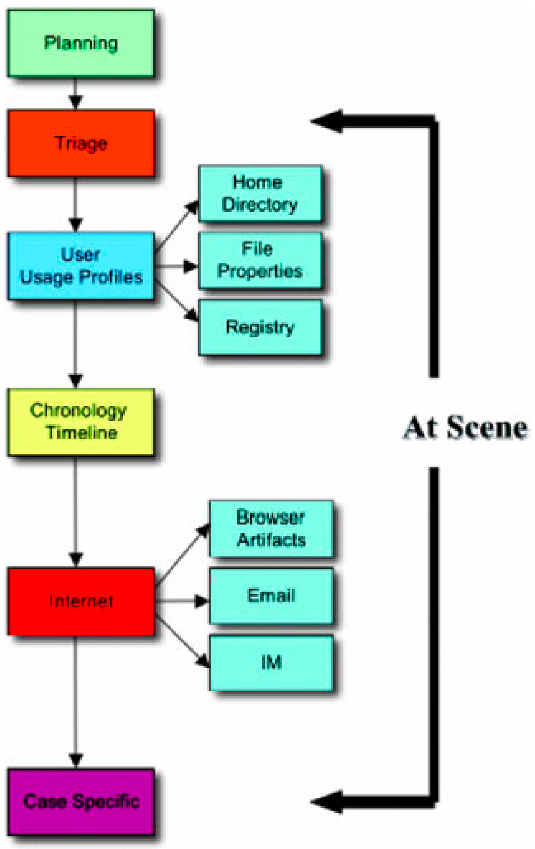


Figure 3. *Investigation procedure using triage before start the extortion of information from the given devices*

Triaging a computer system has be proven to be a successful methodology for avoiding the problems and dilemmas created with pulling the plug.

Keeping in mind that each case has its own requirements, in most situations triaging a computer system would be a better alternative that just going ahead an pulling the plug form a dive. Although triage goes against one of the most well establish rules in forensic, the rule of not alter the evidence, if the investigator performing the examination is competent to do so and know what exactly will be change with each step of the procedure and keeps everything well documented there is now reason not to use triage in the first step of an investigation.

By talking about both methods of live and dead investigations and by having a more in depth look at the process of pulling the plug from a running devices, we start to draw our conclusions about the methodologies and procedures an investigator should consider before starting an examination in a crime scene.

The fist thing a forensic examiner need to have in mined is the nature of the investigation, a clear understanding of what is being investigated will shape the entire procedure used to identify evidence. Whether the investigation is on a live system or a dead system the examiner must be 100% sure that every decision he takes will not affect the integrity of the data found on the give computer system. If the case requires pulling the plug, then the investigator must have a good understanding of what could potentially be lost and there for document the reasons behind his decision. On the other hand, if volatile data in require for the nature of the case the by performing computer triaging the forensic examiner will potentially collect all necessary information with out changing the admissibility of the evidence collected.

REFERENCES

- John J. Barbara. (2010). Before You Pull the Plug. Available: *http://www.forensicmag.com/articles/2010/04/you-pull-plug#.Uk09VV0W38s*. Last accessed 25th Sep 2013.
- Priscilla Oppenheimer. Computer Forensics: Seizing a Computer.Available: *http://www.priscilla.com/forensics/computerseizure.html*. Last accessed 25th Sep 2013.
- Diana J. Michaud. (2001). Adventures in Computer Forensics.Available: *http://www.sans.org/reading-room/whitepapers/incident/adventures-computer-forensics-638?show=adventures-computer-forensics-638&cat=incident*. Last accessed 24th Sep 2013.
- John J. Barbara. (2010). Triage A Computer. Available: *http://www.forensicmag.com/articles/2010/06/triage-computer#.Uk1y_F0W38s*. Last accessed 24th Sep 2013.

ABOUT THE AUTHOR

Kyriakos Loizou is a forensic investigator and currently lives in Paphos, Cyprus. In 2005 he completed a three year course at the Technical School of Paphos where he earned a diploma in Computer Engendering – Theory of Computing and Peripheral Devices. After a two year service in the national guard air force, Kyriakos was then accepted at the Frederick University of Cyprus(FUC) where he followed a 4 year course in Computer Science. When completing his 3rd year of studies at FUC he was offered a position at the University of Central Lancashire where he graduated and earned a Bachelor of Science (Hons) degree in Forensic Computing. Kyriakos now works as an individual forensic examiner for various organizations across the inland of Cyprus.

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.

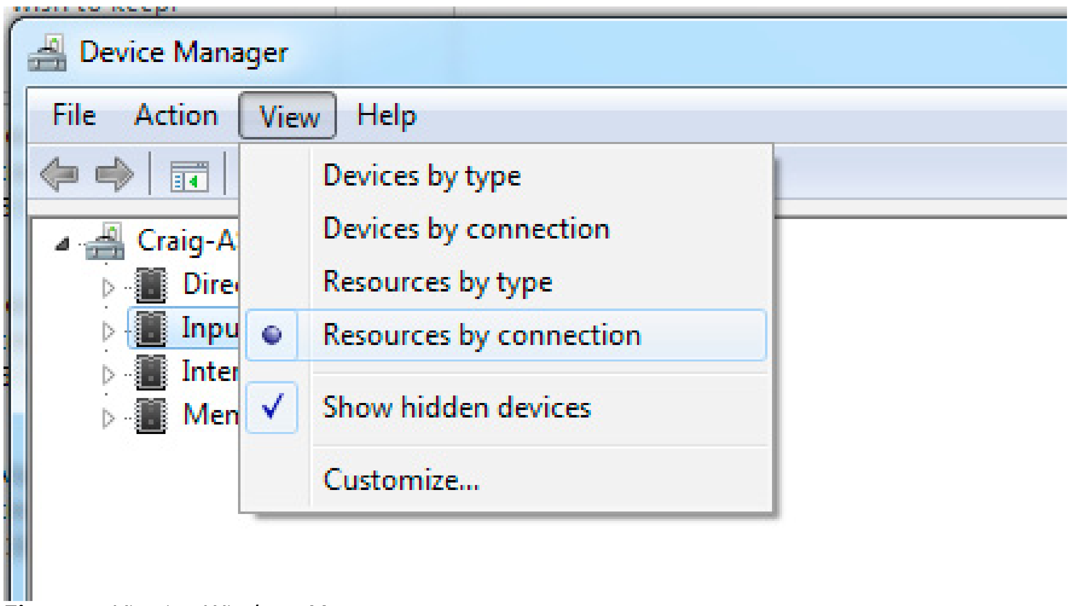
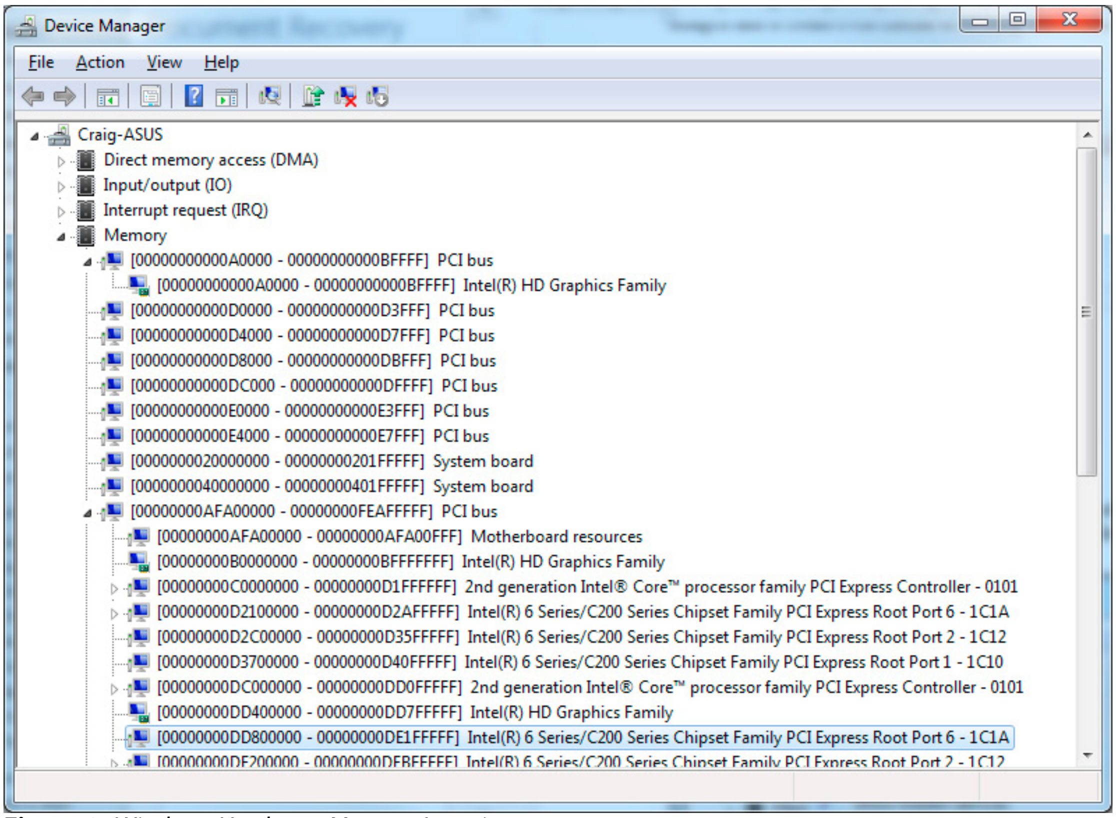


Figure 1. *Viewing Windows Memory*

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.



```
Craig-ASUS
  Direct memory access (DMA)
  Input/output (IO)
  Interrupt request (IRQ)
  Memory
    [00000000000A0000 - 00000000000BFFFF]  PCI bus
      [00000000000A0000 - 00000000000BFFFF]  Intel(R) HD Graphics Family
    [00000000000D0000 - 00000000000D3FFF]  PCI bus
    [00000000000D4000 - 00000000000D7FFF]  PCI bus
    [00000000000D8000 - 00000000000DBFFF]  PCI bus
    [00000000000DC000 - 00000000000DFFFF]  PCI bus
    [00000000000E0000 - 00000000000E3FFF]  PCI bus
    [00000000000E4000 - 00000000000E7FFF]  PCI bus
    [0000000020000000 - 00000000201FFFFF]  System board
    [0000000040000000 - 00000000401FFFFF]  System board
    [00000000AFA00000 - 00000000FEAFFFFF]  PCI bus
      [00000000AFA00000 - 00000000AFA00FFF]  Motherboard resources
      [00000000B0000000 - 00000000BFFFFFFF]  Intel(R) HD Graphics Family
      [00000000C0000000 - 00000000D1FFFFFF]  2nd generation Intel® Core™ processor family PCI Express Controller - 0101
      [00000000D2100000 - 00000000D2AFFFFF]  Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 6 - 1C1A
      [00000000D2C00000 - 00000000D35FFFFF]  Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 2 - 1C12
      [00000000D3700000 - 00000000D40FFFFF]  Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 1 - 1C10
      [00000000DC000000 - 00000000DD0FFFFF]  2nd generation Intel® Core™ processor family PCI Express Controller - 0101
      [00000000DD400000 - 00000000DD7FFFFF]  Intel(R) HD Graphics Family
      [00000000DD800000 - 00000000DE1FFFFF]  Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 6 - 1C1A
      [00000000DE200000 - 00000000DEBFFFFF]  Intel(R) 6 Series/C200 Series Chipset Family PCI Express Root Port 2 - 1C12
```

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.

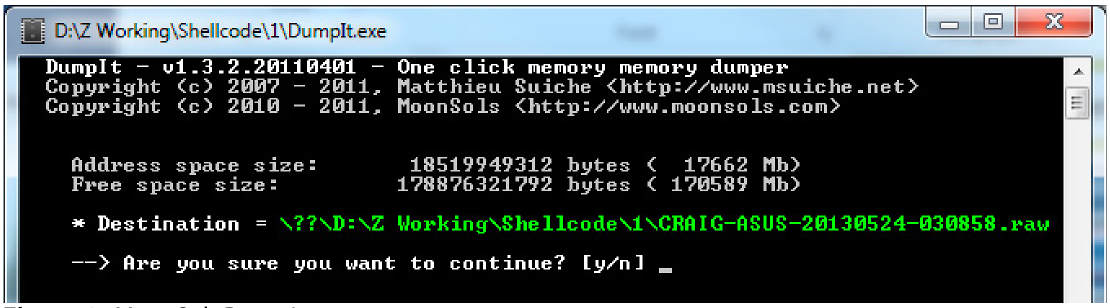
MOONSOLS DUMPIT

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.

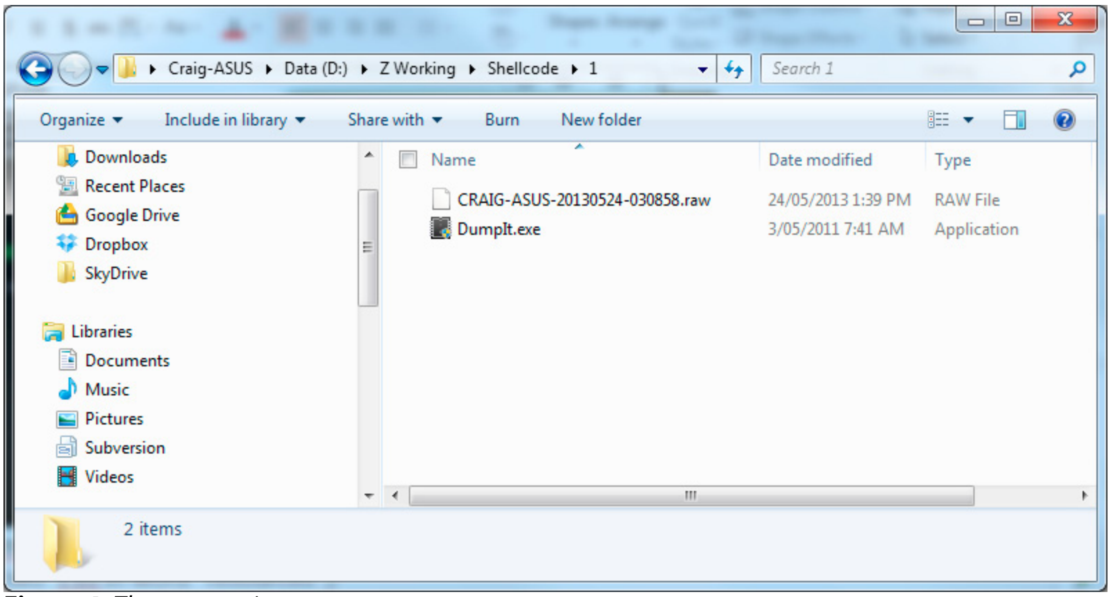


```
D:\Z Working\Shellcode\1\DumpIt.exe

DumpIt - v1.3.2.20110401 - One click memory memory dumper
Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>


  Address space size:       18519949312 bytes (  17662 Mb)
  Free space size:         178876321792 bytes ( 170589 Mb)

  * Destination = \??\D:\Z Working\Shellcode\1\CRAIG-ASUS-20130524-030858.raw

  --> Are you sure you want to continue? [y/n] _
```

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).



Craig-ASUS ▸ Data (D:) ▸ Z Working ▸ Shellcode ▸ 1

Name	Date modified	Type
CRAIG-ASUS-20130524-030858.raw	24/05/2013 1:39 PM	RAW File
DumpIt.exe	3/05/2011 7:41 AM	Application

2 items

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.

D.I.M.
DIGITAL INVESTIGATION MANAGER

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

DFLabs
Managing Infosecurity Risks

SPECIAL PROMO 15% OFF

single user perpetual license

http://www.dimmodule.com

promo code E-FORNCS13

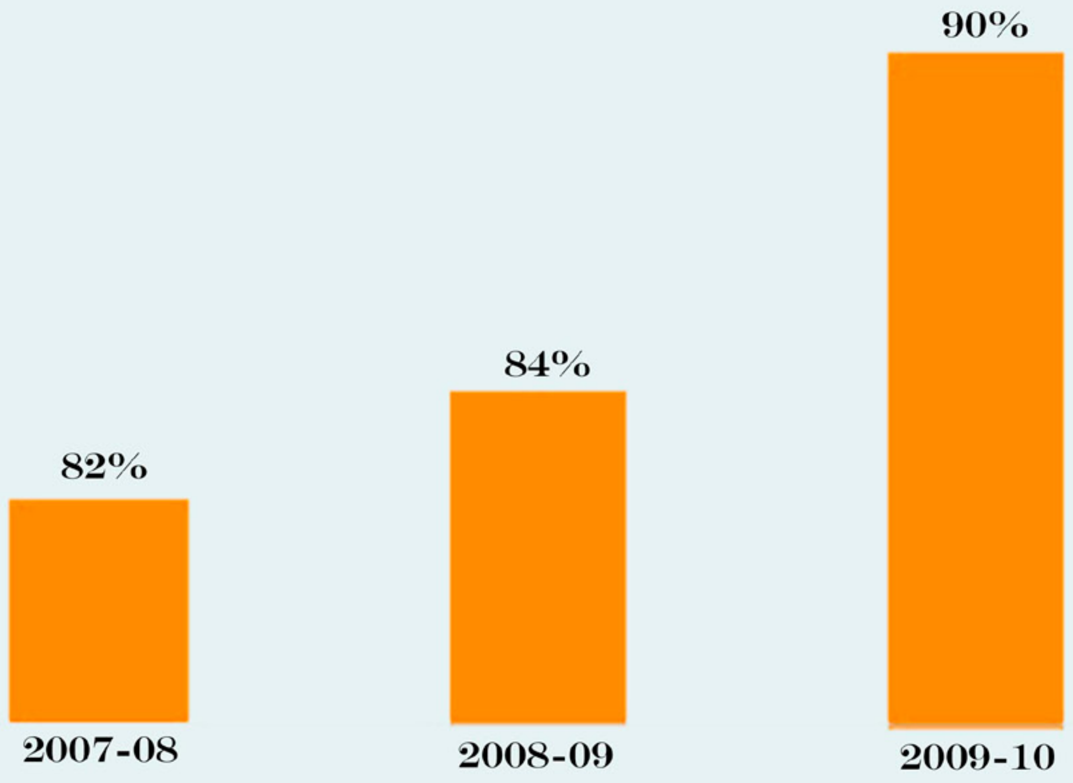
DFLabs DIM is a forensic case management software that coherently manages cases, data input and modifications carried out by the different operators during Digital Evidence Tracking and Forensics Investigations.

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

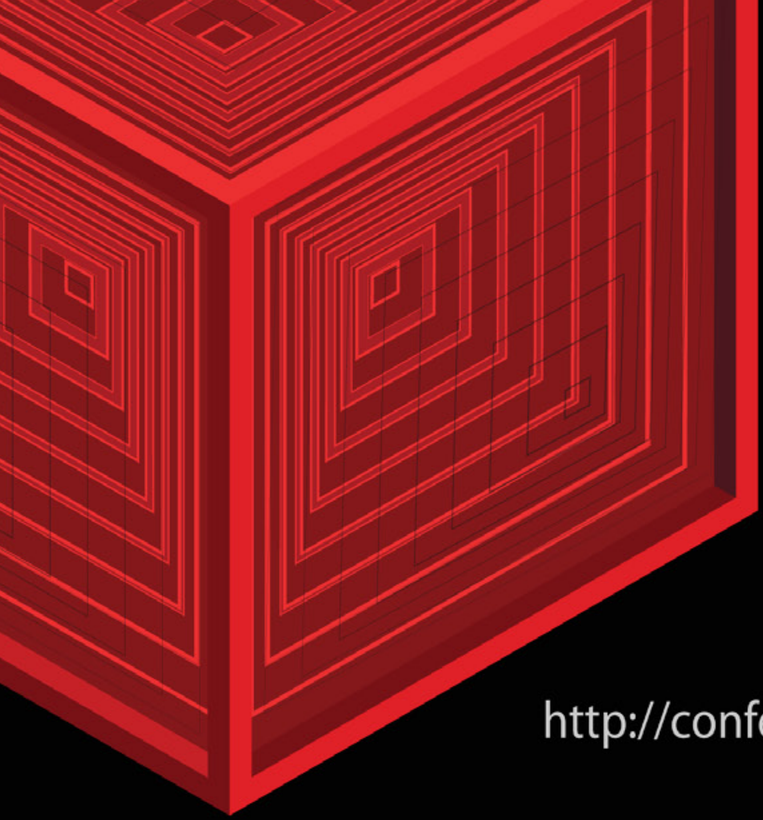
www.dflabs.com email: info@dflabs.com phone: +1 202 4702688 DFLabs

Insider incidents as a percentage of total security incidents



90% of Security incidents happen from inside the organization: "The threat within A study on insider threat by DSCI in collaboration with PwC"

Get to know who in your organization ?
Sempersol Consultancy (P) Ltd.
The Forensicating Troop
Mail to: boonlia@sempersol.in



ELEVENTH ANNUAL
HITB SECURITY
CONFERENCE
IN ASIA

REGISTER ONLINE
http://conference.hitb.org/hitbsecconf2013kul/

HITBSECCONF2013
MALAYSIA

October 14th - 17th 2013 @ InterContinental Kuala Lumpur

8 NEW TRAINING COURSES (14th - 15th October)

- Extreme Web Hacking
- Windows Kernel Internals
- Blackbelt Penetration Testing
- The Art of Exploiting Injection Flaws
- The Android Exploit Lab
- Advanced iOS Exploitation
- Introduction to iOS Exploitation
- Building Secure Web & Mobile Applications

CONFERENCE KEYNOTE SPEAKERS (16th - 17th October)

ANDY ELLIS (Chief Security Officer, Akamai)

JOE SULLIVAN (Chief Security Officer, Facebook)