Hakin9 ON DEMAND

Vol.3 No.1
Issue 01/2014(16) ISSN: 1733-7186

WIRESHARK

WIRESHARK TIPS AND TRICKS

SNIFFING AND RECOVERING NETWORK INFORMATION USING WIRESHARK

TRAFFIC ANALYSIS AND CAPTURE PASSWORDS DETECT/ANALYZE SCANNING TRAFFIC USING WIRESHARK

Dr.Web 9.0 for Windows — the rapid response anti–virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search

www.drweb.com

Free 30-day trial: https://download.drweb.com

New features in Dr.Web 9.0 for Windows: http://products.drweb.com/9

FREE bonus — Dr.Web Mobile Security:
https://download.drweb.com/android

Wireshark

Table of Contents

advertisement

better safe than sorry
www.demyo.com

Dear Readers,

We are happy to present you completlely new issue dedicated to the most known sniffer – Wireshark. We are sure all of you know this special toll. You can use it to analyze network traffic, intrusion detection, or communication protocols development.

This issue is a guidbook for all those who wants to learn step-by-step how to use this sniffing tool. With this issue you will get basic knowledge on how to start an amazing adventure with Wireshark, but you will also dive into deep waters of hacking knowledge. Except of BASICS section you will also find TRAFIC ANALYSIS and INTRUSION DETECTION sections, full of our expert's tutorials.

We would also thank to our friends from PenTest Magazine for sharing their great articles. We appreciate their work which helped us to create this great issue.

Enjoy!

Regards,

Ewelina Nazarczuk

Hakin9 Magazine Junior Product Manager

and Hakin9 Team

Editor in Chief: Ewelina Nazarczuk
ewelina.nazarczuk@hakin9.org

Editorial Advisory Board: Bamidele Ajayi, Kishore PV, Gilles Lami, Rodrigo Comegno, Tim Singletary, Jarvis Simpson, Dallas Moore, Hammad Arshed, Gregory Chrysanthou, Elia Pinto, Jeff Smith, Arnoud Tijssen, Tom Updegrove and others.

Proofreaders: Jeff Smith, Krzysztof Samborski

Special thanks to our Beta testers and Proofreaders who helped us with this issue. Our magazine would not exist without your assistance and expertise.

Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic.@hakin9.org

Product Manager: Ewa Duranc
ewa.duranc@hakin9.org

Production Director: Andrzej Kuca
andrzej.kuca@hakin9.org

Art. Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@hakin9.org
DTP: Ireneusz Pogroszewski

Marketing Director: Ewelina Nazarczuk
ewelina.nazarczuk@hakin9.org

Publisher: Hakin9 Media sp. z o.o. SK
02-676 Warszawa, ul. Postępu 17D
NIP 95123253396
www.hakin9.org/en

[GEEKED AT BIRTH]

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

LEARN:
Advancing Computer Science
Artificial Life Programming
Digital Media
Digital Video
Enterprise Software Development
Game Art and Animation
Game Design
Game Programming
Human-Computer Interaction
Network Engineering
Network Security
Open Source Technologies
Robotics and Embedded Systems
Serious Game and Simulation
Strategic Technology Development
Technology Forensics
Technology Product Design
Technology Studies
Virtual Modeling and Design
Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

Please see www.uat.edu/fastfacts for the latest information about degree program performance, placement and costs.

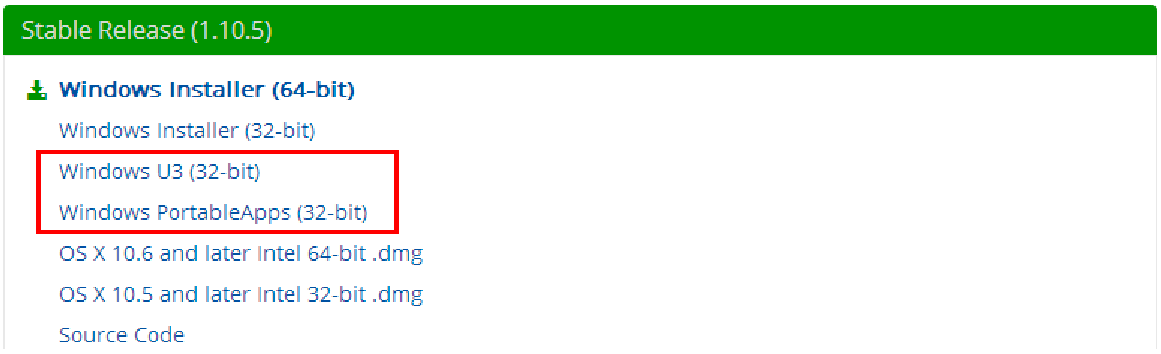
Wireshark Tips and Tricks

by Tony Lee and Jason Bevis

If you were tasked to put together a forensic toolkit with 25 tools or less, chances are Wireshark would be one of those tools--especially if you planned on dealing with packet captures. Because it is free, open source, and cross-platform, Wireshark makes a great packet capture and analysis tool for just about any forensic toolkit. Never the less, this staple tool has been around for so long (think back to the days of Ethereal) that we sometimes take it for granted. In this article we will explore a few tips and tricks that highlight why we like this tool so much.

Obtaining the software

This seems easy enough, right? Many Linux distributions come with Wireshark installed as a default package and Windows has an easy point and click install package. But did you know there is a PortableApps release of Wireshark? How about a U3 release as well?



Stable Release (1.10.5)

Windows Installer (64-bit)
Windows Installer (32-bit)
Windows U3 (32-bit)
Windows PortableApps (32-bit)
OS X 10.6 and later Intel 64-bit .dmg
OS X 10.5 and later Intel 32-bit .dmg
Source Code

Figure 1. Install options – http://www.wireshark.org/download.html

The PortableApps and U3 downloads allow you to run Wireshark from a USB stick without the needing to install the software on the listening machine. Instead, when you insert the USB stick, you are good to go. There are some caveats that exist so be sure to read the fine print. In any event, this provides a flexible and portable option for running Wireshark on other machines.

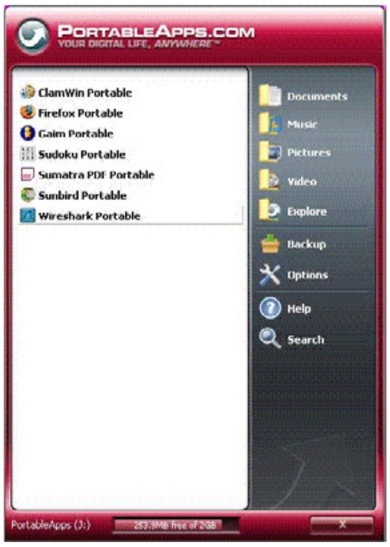
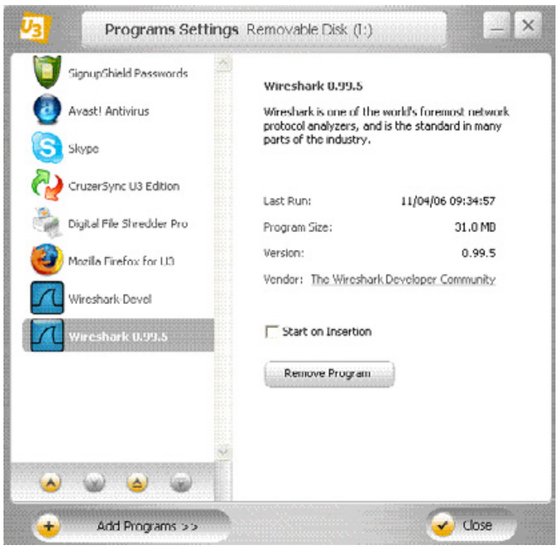


Figure 2. PortableApps Wireshark



Wireshark 0.99.5

Wireshark is one of the world's foremost network protocol analyzers, and is the standard in many parts of the industry.

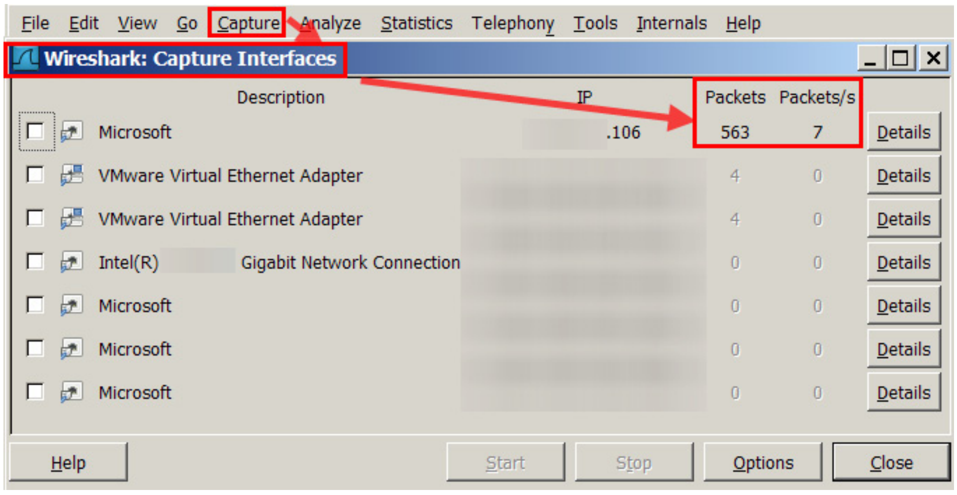
Last Run: 11/04/06 09:34:57
Program Size: 31.0 MB
Version: 0.99.5
Vendor: The Wireshark Developer Community

☐ Start on Insertion

Figure 3. U3 Wireshark

Working with pcaps

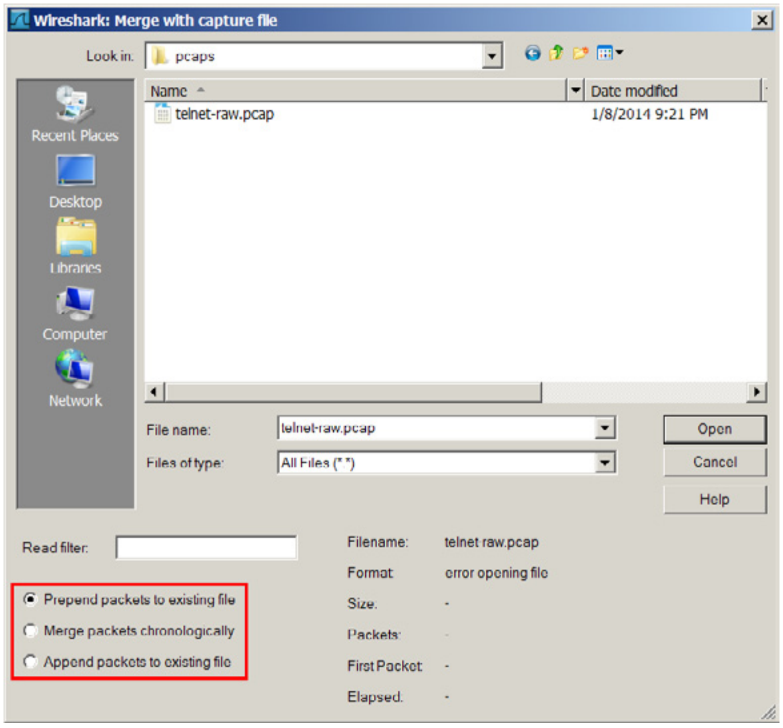
Again, this is another topic that seems self-explanatory. However, when we teach classes on capturing and analyzing traffic we seem to get this question: "I have several interfaces, how do I know which interface in which to listen?" The reason for this question stems from the main screen, which allows users to select an interface, but it does not show which interface(s) are seeing traffic. The work around is to click capture → Interfaces. This menu option shows you the interfaces in real time so you can see which are live and receiving traffic.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Wireshark: Capture Interfaces
Help
Start
Stop
Options
Close

	Description	IP	Packets	Packets/s	
☐	Microsoft	[redacted].106	563	7	Details
☐	VMware Virtual Ethernet Adapter	[redacted]	4	0	Details
☐	VMware Virtual Ethernet Adapter	[redacted]	4	0	Details
☐	Intel(R) [redacted] Gigabit Network Connection	[redacted]	0	0	Details
☐	Microsoft	[redacted]	0	0	Details
☐	Microsoft	[redacted]	0	0	Details
☐	Microsoft	[redacted]	0	0	Details

Figure 4. Capture Interface screen showing live traffic

Opening pcaps that were saved or created in other programs is as easy as dragging and dropping them into Wireshark, but did you know you can easily merge pcaps by doing the same thing? By default dragging and dropping multiple pcaps into Wireshark will cause it to merge the pcaps chronologically. You can also merge pcaps by going to File à Merge where you can select different options to merge the pcaps (prepend, append, or chronologically). This is useful if you collect from multiple sensors or interfaces and want to see the complete picture.



Look in: pcaps

Name	Date modified
telnet-raw.pcap	1/8/2014 9:21 PM

File name: telnet-raw.pcap
Files of type: All Files (*.*)

Read filter:

(●) Prepend packets to existing file
( ) Merge packets chronologically
( ) Append packets to existing file

Filename:	telnet raw.pcap
Format:	error opening file
Size:	-
Packets:	-
First Packet:	-
Elapsed:	-

Figure 5. Merging pcap options

Need a few pcaps to work with? How about viewing some more interesting protocols than what is most likely found on your home network? The wiki at Wireshark.org has lots of pcaps to pick and choose from: *http://wiki.wireshark.org/SampleCaptures*.

Display filters

Since traffic on a busy pipe can be overwhelming, Wireshark provides the capability to use capture and display filters. However, in most cases, for troubleshooting or quick analysis it is best to capture unfiltered traffic and then identify packets of interest using display filters. It is inevitable though that sooner or later you will have to learn a few basic display filters. Start off with easy ones such as the following:

Display only certain protocols:	Display only certain addresses:	Combine Display Filters
Examples: http telnet ftp	Examples: ip.addr == 172.16.100.11 ip.src == 172.16.100.11 ip.dst == 172.16.100.11	Examples: ip.addr == 172.16.100.11 && http dns || http

The simple filters above should be enough to meet most basic requirements, however if a more complex display filter is needed, the Wireshark Expression button is very helpful. It is located right next to the display filter field and acts as a sort of a wizard for building display filters.

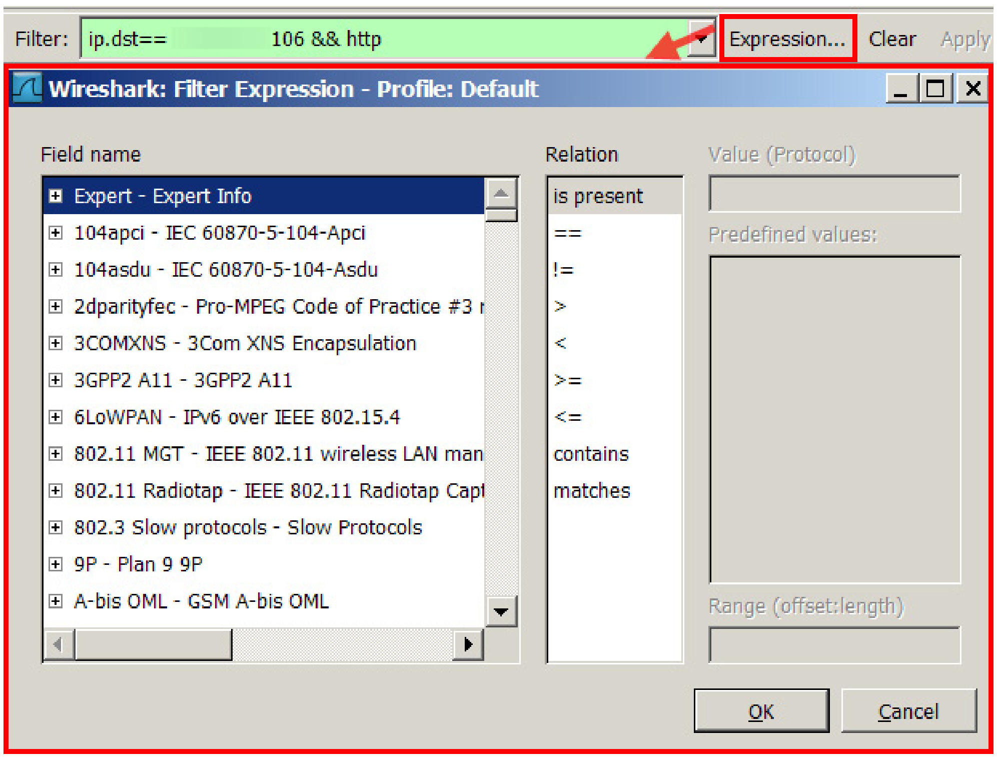
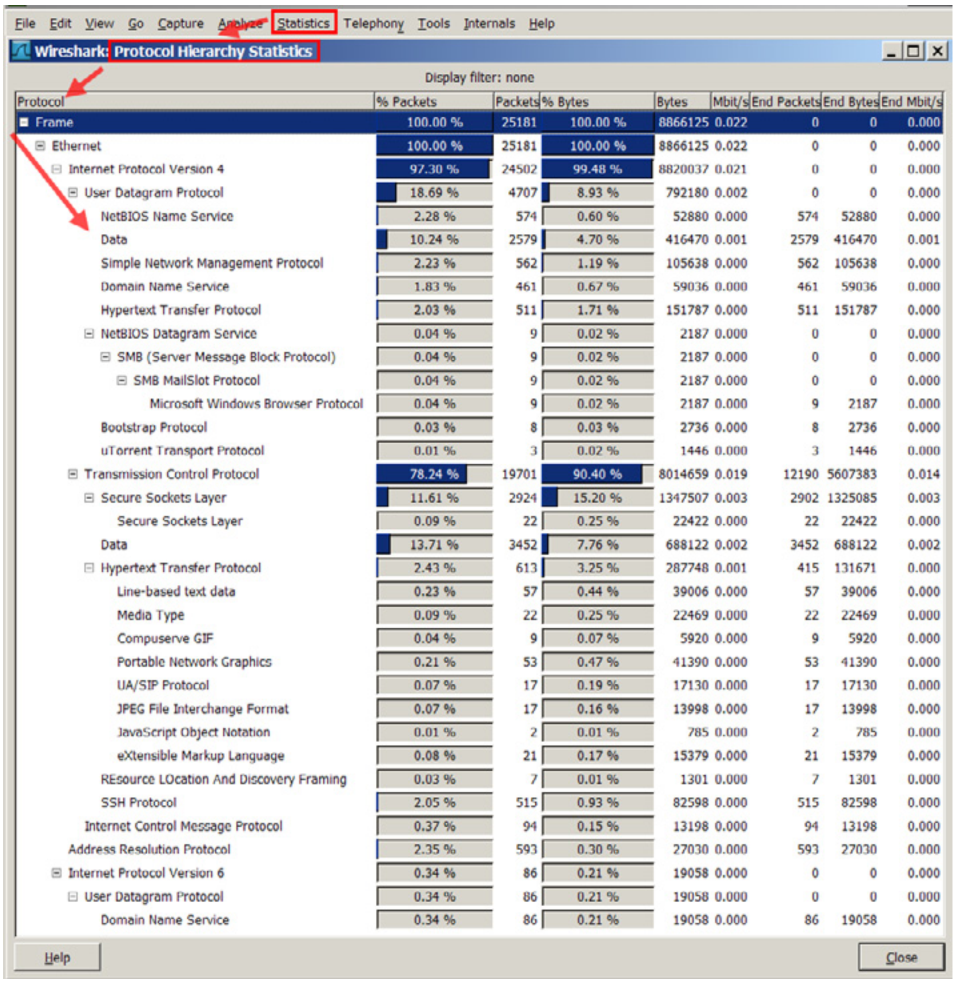

Filter: ip.dst== 106 && http
Expression...
Clear
Apply
Wireshark: Filter Expression - Profile: Default
Field name
Expert - Expert Info
104apci - IEC 60870-5-104-Apci
104asdu - IEC 60870-5-104-Asdu
2dparityfec - Pro-MPEG Code of Practice #3 r
3COMXNS - 3Com XNS Encapsulation
3GPP2 A11 - 3GPP2 A11
6LoWPAN - IPv6 over IEEE 802.15.4
802.11 MGT - IEEE 802.11 wireless LAN man
802.11 Radiotap - IEEE 802.11 Radiotap Capt
802.3 Slow protocols - Slow Protocols
9P - Plan 9 9P
A-bis OML - GSM A-bis OML
Relation
is present
==
!=
>
<
>=
<=
contains
matches
Value (Protocol)
Predefined values:
Range (offset:length)
OK
Cancel

Figure 6. Display filter Expression button

You can also check out this page for more good ideas: *http://wiki.wireshark.org/DisplayFilters*.

High to low level

At times you may be given a pcap with no background knowledge about the protocols or data captured. One of the easiest ways to gain a quick understanding of the situation is by using Wireshark's statistics features. Our favorite option for drilling down on protocols is Statistics → Protocol Hierarchy.



Display filter: none

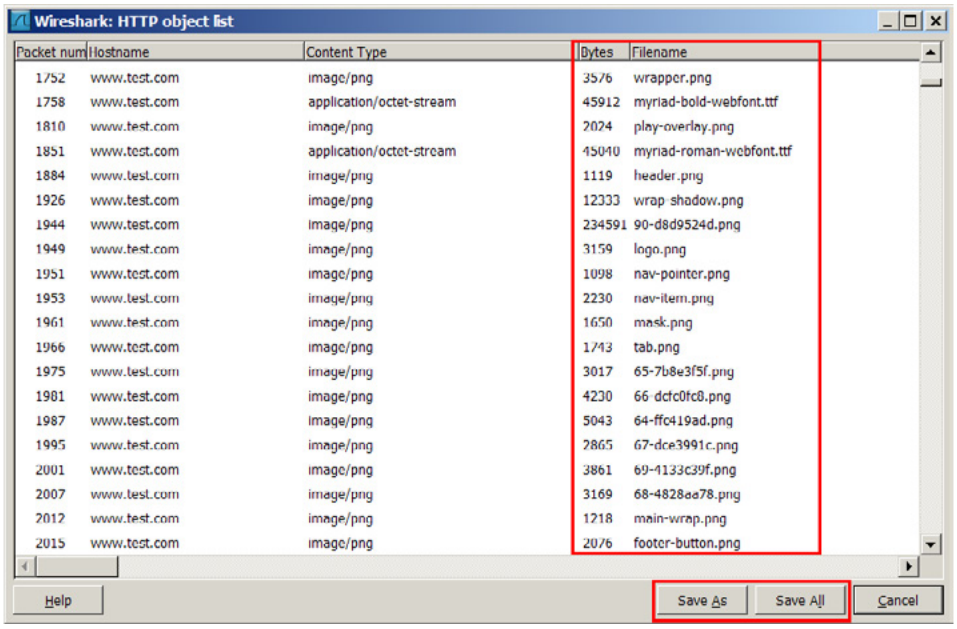
Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00 %	25181	100.00 %	8866125	0.022	0	0	0.000
Ethernet	100.00 %	25181	100.00 %	8866125	0.022	0	0	0.000
Internet Protocol Version 4	97.30 %	24502	99.48 %	8820037	0.021	0	0	0.000
User Datagram Protocol	18.69 %	4707	8.93 %	792180	0.002	0	0	0.000
NetBIOS Name Service	2.28 %	574	0.60 %	52880	0.000	574	52880	0.000
Data	10.24 %	2579	4.70 %	416470	0.001	2579	416470	0.001
Simple Network Management Protocol	2.23 %	562	1.19 %	105638	0.000	562	105638	0.000
Domain Name Service	1.83 %	461	0.67 %	59036	0.000	461	59036	0.000
Hypertext Transfer Protocol	2.03 %	511	1.71 %	151787	0.000	511	151787	0.000
NetBIOS Datagram Service	0.04 %	9	0.02 %	2187	0.000	0	0	0.000
SMB (Server Message Block Protocol)	0.04 %	9	0.02 %	2187	0.000	0	0	0.000
SMB MailSlot Protocol	0.04 %	9	0.02 %	2187	0.000	0	0	0.000
Microsoft Windows Browser Protocol	0.04 %	9	0.02 %	2187	0.000	9	2187	0.000
Bootstrap Protocol	0.03 %	8	0.03 %	2736	0.000	8	2736	0.000
uTorrent Transport Protocol	0.01 %	3	0.02 %	1446	0.000	3	1446	0.000
Transmission Control Protocol	78.24 %	19701	90.40 %	8014659	0.019	12190	5607383	0.014
Secure Sockets Layer	11.61 %	2924	15.20 %	1347507	0.003	2902	1325085	0.003
Secure Sockets Layer	0.09 %	22	0.25 %	22422	0.000	22	22422	0.000
Data	13.71 %	3452	7.76 %	688122	0.002	3452	688122	0.002
Hypertext Transfer Protocol	2.43 %	613	3.25 %	287748	0.001	415	131671	0.000
Line-based text data	0.23 %	57	0.44 %	39006	0.000	57	39006	0.000
Media Type	0.09 %	22	0.25 %	22469	0.000	22	22469	0.000
Compuserve GIF	0.04 %	9	0.07 %	5920	0.000	9	5920	0.000
Portable Network Graphics	0.21 %	53	0.47 %	41390	0.000	53	41390	0.000
UA/SIP Protocol	0.07 %	17	0.19 %	17130	0.000	17	17130	0.000
JPEG File Interchange Format	0.07 %	17	0.16 %	13998	0.000	17	13998	0.000
JavaScript Object Notation	0.01 %	2	0.01 %	785	0.000	2	785	0.000
eXtensible Markup Language	0.08 %	21	0.17 %	15379	0.000	21	15379	0.000
REsource LOcation And Discovery Framing	0.03 %	7	0.01 %	1301	0.000	7	1301	0.000
SSH Protocol	2.05 %	515	0.93 %	82598	0.000	515	82598	0.000
Internet Control Message Protocol	0.37 %	94	0.15 %	13198	0.000	94	13198	0.000
Address Resolution Protocol	2.35 %	593	0.30 %	27030	0.000	593	27030	0.000
Internet Protocol Version 6	0.34 %	86	0.21 %	19058	0.000	0	0	0.000
User Datagram Protocol	0.34 %	86	0.21 %	19058	0.000	0	0	0.000
Domain Name Service	0.34 %	86	0.21 %	19058	0.000	86	19058	0.000

Figure 7. Statistics → Protocol Hierarchy displays an OSI breakdown

Not only does this give you an excellent OSI breakdown, but you can right click on a protocol to apply a filter to look at only those packets of interest.

Extracting files

There are plenty of methods/tools to extract files from a pcap, including: foremost (*http://foremost.sourceforge.net*) and Network Miner (*http://sourceforge.net/projects/networkminer*) – however, Wireshark can also be used to extract files. Some would say that you have to follow the stream and export raw bytes to extract a file. While this is one possible method, depending on the protocol, you may still have to use a hex editor to clean up the resulting file. Wireshark can extract objects from supported protocols by using: File à Export Objects → <Protocol>. In our example, we are exporting from HTTP – which is very common. It makes exporting binaries, zip files, images, and even JavaScript and applets easy.



Packet num	Hostname	Content Type	Bytes	Filename
1752	www.test.com	image/png	3576	wrapper.png
1758	www.test.com	application/octet-stream	45912	myriad-bold-webfont.ttf
1810	www.test.com	image/png	2024	play-overlay.png
1851	www.test.com	application/octet-stream	45040	myriad-roman-webfont.ttf
1884	www.test.com	image/png	1119	header.png
1926	www.test.com	image/png	12333	wrap-shadow.png
1944	www.test.com	image/png	234591	90-d8d9524d.png
1949	www.test.com	image/png	3159	logo.png
1951	www.test.com	image/png	1098	nav-pointer.png
1953	www.test.com	image/png	2230	nav-item.png
1961	www.test.com	image/png	1650	mask.png
1966	www.test.com	image/png	1743	tab.png
1975	www.test.com	image/png	3017	65-7b8e3f5f.png
1981	www.test.com	image/png	4230	66-dcfc0fc8.png
1987	www.test.com	image/png	5043	64-ffc419ad.png
1995	www.test.com	image/png	2865	67-dce3991c.png
2001	www.test.com	image/png	3861	69-4133c39f.png
2007	www.test.com	image/png	3169	68-4828aa78.png
2012	www.test.com	image/png	1218	main-wrap.png
2015	www.test.com	image/png	2076	footer-button.png

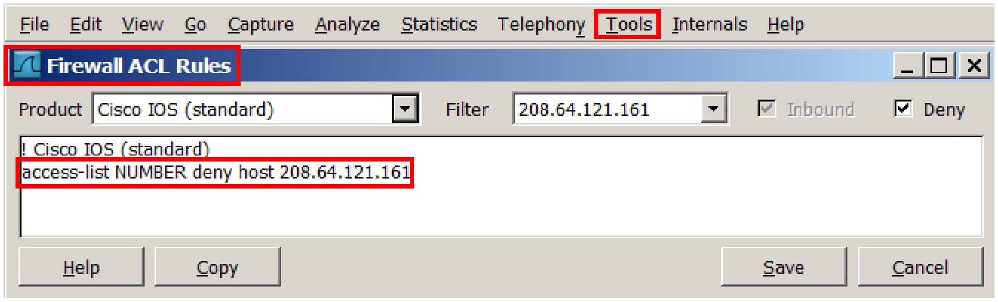
Figure 8. Exporting HTTP objects from Wireshark

Generating firewall rules

Lastly, an interesting feature of Wireshark is that it can generate firewall rules (for different vendors) so you can prevent further unwanted traffic from traversing your network boundary. Just select a packet of interest and click Tools → Firewall ACL Rules. The product drop down allows you to select from the following vendors:

- Cisco IOS (standard)
- Cisco IOS (extended)
- IP Filter (ipfilter)
- IPFirewall (ipfw)
- Netfilter (iptables)
- Packet Filter (pf)
- Windows Firewall (netsh)

Then select the IP of interest and decide which IP address, inbound or outbound, and if you want to deny or permit the traffic.



Product: Cisco IOS (standard) Filter: 208.64.121.161 Inbound [✓ Deny]

```
! Cisco IOS (standard)
access-list NUMBER deny host 208.64.121.161
```

Figure 9. Tools → Firewall ACL Rules

Final thoughts

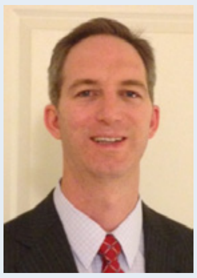
Wireshark is one of those tools that has been around so long we may take it for granted at times, but when you need something to capture or carve packets it is freely available, reliable, and capable. Hopefully this article has either cleared some cobwebs for you or possibly served as a brief introduction to some of the key (and sometimes under advertised) features of the tool. Feedback is always appreciated using our contact information below. Thanks for reading--now go carve some packets.

About the Author

Tony Lee has more than nine years of professional experience pursuing his passion in all areas of information security. He is currently a principal security consultant at FireEye Labs, in charge of advancing many of the network penetration testing service lines. His interests of late are Citrix and kiosk hacking, post exploitation tactics, and malware research. As an avid educator, Tony has instructed thousands of students at many venues worldwide, including government, universities, corporations, and conferences such as Black Hat. He takes every opportunity to share knowledge as a contributing author to Hacking Exposed 7, frequent blogger, and a lead instructor for a series of classes. He holds a Bachelor of Science degree in computer engineering from Virginia Polytechnic Institute and State University and a Master of Science degree in security informatics from The Johns Hopkins University.

Email: Tony.Lee -at- FireEye.com
Linked-in: http://www.linkedin.com/in/tonyleevt

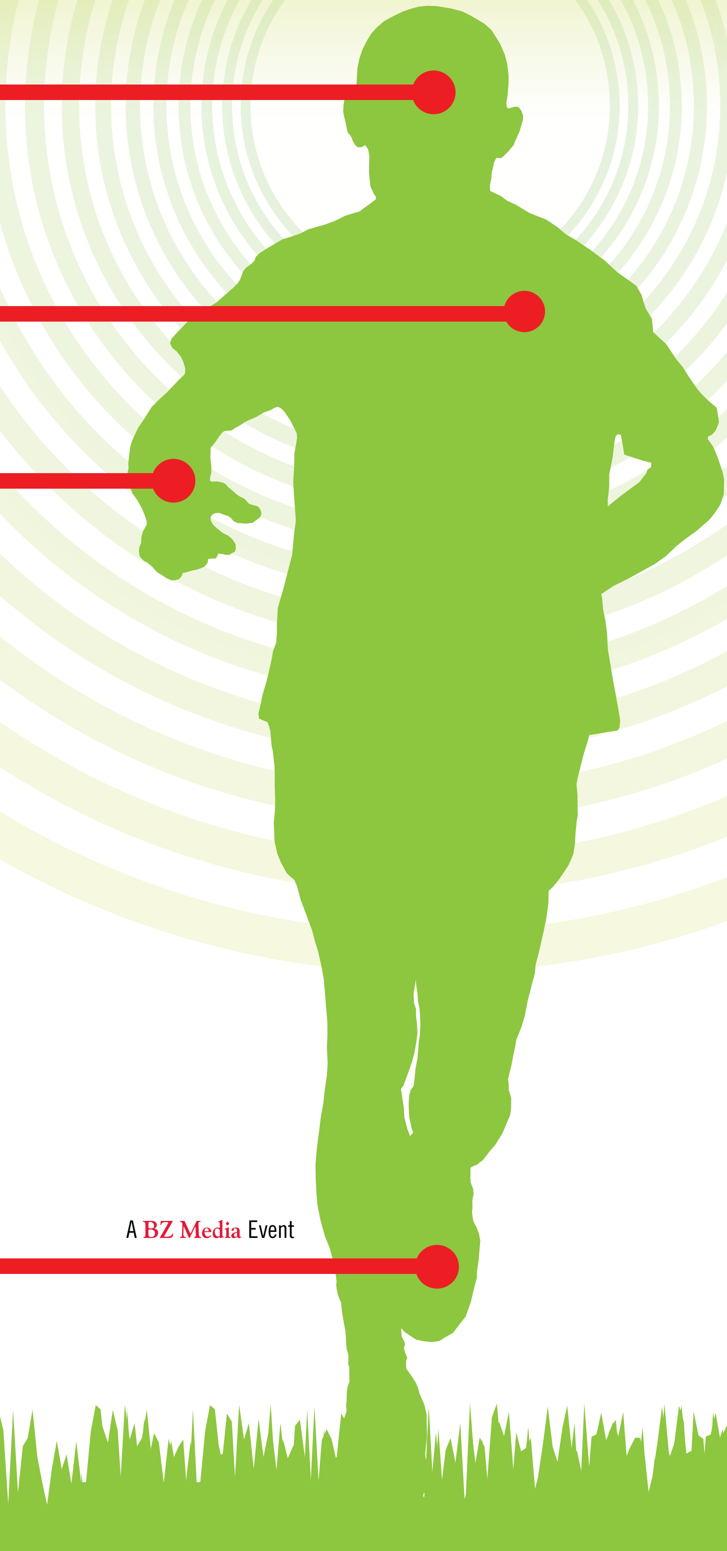
About the Author



Mr. Bevis has an outstanding track record in consulting services, strategic security solutions, incident response, and client management. Jason is a results driven leader with over 17 years of experience including more than six years of "Big X" Consulting experience. He is an expert in the field of information security, incident management, and risk management with a strong technical background in incident response, forensics, security strategy, and network and system architecture. He runs his personal security blog and his interested of late are within the maker community playing around with Arduino projects and researching sensor communications.

Email: Jason.Bevis -at- FireEye.com
Linked-in: http://www.linkedin.com/in/jasonbevis

Join the Wearables Revolution!



Wearables DevCon

A conference for Designers, Builders and Developers of Wearable Computing Devices

Wearable computing devices are the Next Big Wave in technology. And the winning developers in the next decade are going to be the ones who take advantage of these new technologies EARLY and build the next generation of red-hot apps.

Choose from over 35 classes and tutorials!

- Learn how to develop apps for the coolest gadgets like Google Glass, FitBit, Pebble, the SmartWatch 2, Jawbone, and the Galaxy Gear SmartWatch
- Get practical answers to real problems, learn tangible steps to real-world implementation of the next generation of computing devices

March 5-7, 2014
San Francisco
WearablesDevCon.com

A BZ Media Event

Getting Started with Wireshark

by Sebastian Perez

As a pentester, I always get involved in different projects from different clients and no matter what the objective is, having the knowledge and the proper tool to perform the task will save a lot of time, and avoid some headaches. This article will try to aid for those scenarios where a network analysis should be performed. We will focus in one of the most important tools for a pentester: Wireshark.

For most of the engagements a pen tester could perform, there is always a network component, and being able to see, analyze and store all the network transactions is essential to understand network behaviors and evidence all the performed tasks. For such objectives, Wireshark is what was promised, and more. Looking for a formal definition of Wireshark, as stated in the official website (http://www.wireshark.org/faq.htm), it is a free open-source network protocol analyzer. What does this mean? It means that Wireshark will capture all the traffic it can hear from the selected network interface, parse it and present it to the user in a friendly manner. Within the captures packets, Wireshark will allow us to perform several tasks, such as analyze network problems, get network statistics, and even detect network intrusion attempts. Wireshark runs in most of the operating system available in the market, including Windows, OS X, Linux and UNIX, and it has two different interfaces, allowing users to adapt it to their own requirements; it could be executed in a GUI (Graphic User Interface) or CLI (Command Line Interface). The installation process is really simple, no matter if it's being performed on Windows, or *nix based systems. Besides Wireshark, the Winpcap library (libpcap in *nix) will be needed to be installed also. If the Windows GUI version is executed, the following screen will be presented:

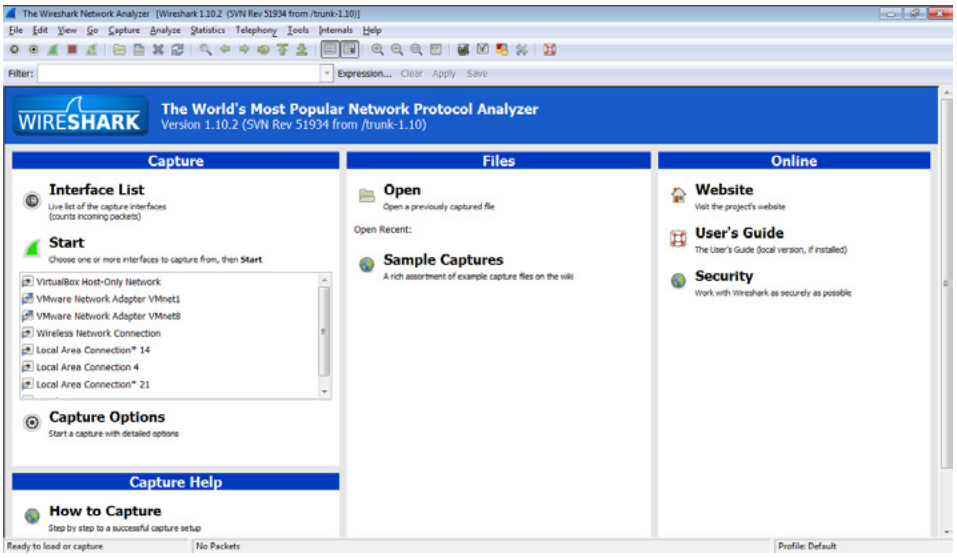
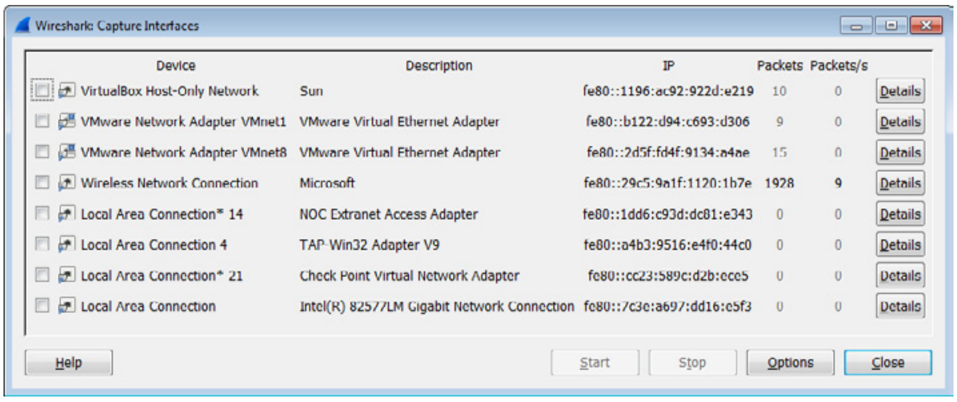

The Wireshark Network Analyzer [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Version 1.10.2 (SVN Rev 51934 from /trunk-1.10)
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
VirtualBox Host-Only Network
VMware Network Adapter VMnet1
VMware Network Adapter VMnet8
Wireless Network Connection
Local Area Connection* 14
Local Area Connection 4
Local Area Connection* 21
Capture Options
Start a capture with detailed options
Capture Help
How to Capture
Step by step to a successful capture setup
Files
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files on the wiki
Online
Website
Visit the project's website
User's Guide
The User's Guide (local version, if installed)
Security
Work with Wireshark as securely as possible
Ready to load or capture
No Packets
Profile: Default

Figure 1. Wireshark main window

The main screen is divided in 3 tabs. The one in the left side is related to the capture options, and the list of interfaces in which Wireshark may be able to listen and capture traffic. The middle tab shows the saved Wireshark sessions; and the right one is for the online content, such as the user guide and official website. We will focus on the Capture tab, as the other two are self-explanatory.

In order to capture traffic, we need to specify to Wireshark which network interface(s) we would like to listen. Currently, most computers have more than one network interface, so in case we are unsure of which is the proper one to listen, Wireshark provides an interface list. This option will show all the network interfaces in the computer and the packet count on each of them (The count starts when we access this option). This will make easier to identify the active interfaces, and probably the one with the most count of packets is the one we would like to capture. Clicking on "Details" button, will provide even more information of each network interfaces.



Device	Description	IP	Packets	Packets/s	
VirtualBox Host-Only Network	Sun	fe80::1196:ac92:922d:e219	10	0	Details
VMware Network Adapter VMnet1	VMware Virtual Ethernet Adapter	fe80::b122:d94:c693:d306	9	0	Details
VMware Network Adapter VMnet8	VMware Virtual Ethernet Adapter	fe80::2d5f:fd4f:9134:a4ae	15	0	Details
Wireless Network Connection	Microsoft	fe80::29c5:9a1f:1120:1b7e	1928	9	Details
Local Area Connection* 14	NOC Extranet Access Adapter	fe80::1dd6:c93d:dc81:e343	0	0	Details
Local Area Connection 4	TAP-Win32 Adapter V9	fe80::a4b3:9516:e4f0:44c0	0	0	Details
Local Area Connection* 21	Check Point Virtual Network Adapter	fe80::cc23:589c:d2b:ece5	0	0	Details
Local Area Connection	Intel(R) 82577LM Gigabit Network Connection	fe80::7c3e:a697:dd16:e5f3	0	0	Details

Figure 2. Example of the list of interfaces to capture

Once proper network interface is identified, just select it in the main screen and click on the "*Start*" button. Once the Wireshark starts to capture, will show a screen like the following: Figure 3.

This screen is divided in 3 rows, each of this showing different information. The top row is the packet list pane; the middle row is the packet details pane and the bottom one is the packet bytes pane. Let's talk about each one of them:

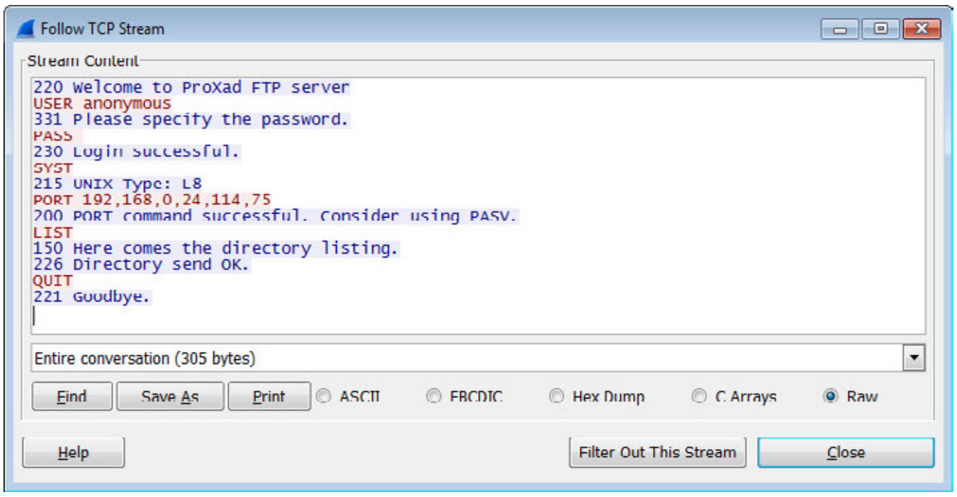
Packet list pane (1)

This pane displays all the packets that are being captured during the current session; in case of a previously stored session, it will display all the packets that were saved on that session. Each of the lines within this pane corresponds to one packet that was captured. By selecting one of them, the packet details pane and packet bytes pane will be updated to reflect the content of the selected packet. The default pane configuration contains 7 columns, displaying the following attributes:

- *No.:* The number of the packet in the capture file. This number is related to the current session only, and is incremented by one for each packet.
- *Time*: The timestamp of the packet. The default configuration shows the amount of seconds since the beginning of the current session. This value could be changed to reflect the proper date and time instead, such as the UTC time.
- *Source*: The IP address from where this packet came from. In case that no IP address was available (ARP packets, for example), the name and part of the device's MAC address is displayed.
- *Destination*: The IP address where this packet is going to. In case of Broadcast packets, the legend "*Broadcast*" is displayed. And similar to "Source" field, if no IP address was available, the name and part of the device's MAC address is displayed.
- *Protocol*: Display the protocol used, in a short name version, or abbreviation, if a short name is not available
- *Length*: Display the packet length
- *Info*: Provides additional information about the packet content, such as TCP header fields

More columns could be added as required, depending on the information the user may need to gather, as Wireshark provides the functionality to add custom columns, using the packet fields available. In order to add a column, just right-click on a column name and select the option "*Column preferences*". A dialog window will show up with a drop-down list, containing a list of predefined columns. Within this list, a "*custom*" option is available to define personal filters.

Going back to the packet list pane, by right-clicking on this window, it is possible to access a menu of tasks to perform, including options to filter packets, follow streams, and copy the packets in TXT or CSV formats. One of the functions that can be very helpful if the user is working with TCP or UDP protocols is the *Follow TCP Stream* or *Follow UDP Steam*. This function will display the data from a TCP or UDP stream in the way that the application layer understands it. By selecting this function, an automated filter will be applied within the packet list, and a dialog window will show a reconstruction of the messages that were sent and received within this stream.



```
220 Welcome to ProXad FTP server
USER anonymous
331 Please specify the password.
PASS
230 Login successful.
SYST
215 UNIX Type: L8
PORT 192,168,0,24,114,75
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
QUIT
221 Goodbye.
```

Entire conversation (305 bytes)

ASCII EBCDIC Hex Dump C Arrays [✓ Raw]

Figure 4. Following a Telnet stream

This window will display the requests to the server in red color, and the responses from the server in blue color. This function is very helpful if the user wants to find some clear text within a particular connection, such as user credentials, or even website cookies. Within this window, there is a search button, to find a particular string within the stream. Also, it is possible to change the character representation between ASCII, EBDCID, hexadecimal, C arrays, or even the raw data

Packet details pane (2)

This pane shows the protocols and protocol fields of the selected packet using a tree structure, which can be expanded and collapsed. Every tree parent is related to a different network layer. This pane will disassemble the packet and display the content of the different layers that compose it. It will parse and show the values of the different fields in each of the protocols involved. In this structure, information like the MAC address of the devices involved and the source and destination port could be observed. By right-clicking on this pane, it is possible to access a menu of tasks similar to the one displayed in the packet list pane.

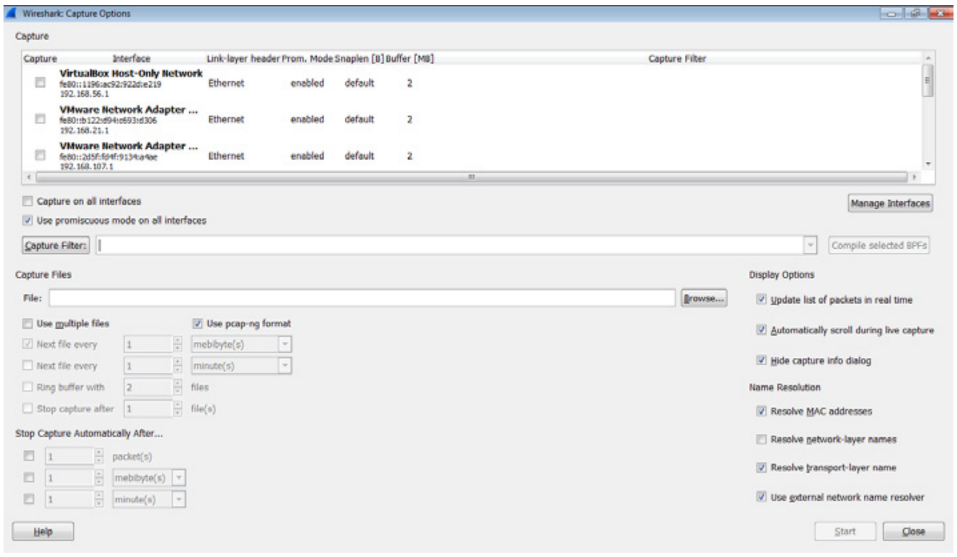
Packet bytes pane (3)

This pane is divided in 3 different columns to show the raw data of the selected packet, using a hexadecimal notation:

- The left column displays the offset in the packet data;
- The middle column displays the packet data in hexadecimal representation
- The right column displays the corresponding ASCII codification, or the dot "." character, if there is no ASCII codification to display.

By right-clicking on this pane, it is possible to change the codification from hexadecimal to binary.

Before starting capturing packets, there are two things to analyze.


Wireshark: Capture Options

Capture

Capture	Interface	Link-layer header	Prom. Mode	Snaplen [B]	Buffer [MB]	Capture Filter
☐	**VirtualBox Host-Only Network** fe80::1196:ac92:922d:e219 192.168.56.1	Ethernet	enabled	default	2	
☐	**VMware Network Adapter ...** fe80::b122:d94:c693:d306 192.168.21.1	Ethernet	enabled	default	2	
☐	**VMware Network Adapter ...** fe80::2d5f:fd4f:9134:a4ae 192.168.107.1	Ethernet	enabled	default	2	

☐ Capture on all interfaces
☑ Use promiscuous mode on all interfaces
Capture Filter:

Capture Files
File:
☐ Use multiple files ☑ Use pcap-ng format
☑ Next file every 1 mebibyte(s)
☐ Next file every 1 minute(s)
☐ Ring buffer with 2 files
☐ Stop capture after 1 file(s)

Stop Capture Automatically After...
☐ 1 packet(s)
☐ 1 mebibyte(s)
☐ 1 minute(s)

Display Options
☑ Update list of packets in real time
☑ Automatically scroll during live capture
☑ Hide capture info dialog

Name Resolution
☑ Resolve MAC addresses
☐ Resolve network-layer names
☑ Resolve transport-layer name
☑ Use external network name resolver

Figure 5. Capture options

The first one I would like to talk about is the capture options, as this allows to specify the behavior that Wireshark will have related to the captures. Let's enumerate the most important options available:

- *Capture on all interfaces*: allows Wireshark to capture on all network interfaces, or to select multiple network interfaces to listen, using the interfaces list that appears above

- *Use promiscuous mode on all interfaces*: this option allows specifying if Wireshark should activate the promiscuous mode setting in all the network interfaces. We will review this setting later.

- *Capture Filter:* this field could be used to specify a filter to be applied in all the selected interfaces to listen. This filter will prevent unwanted packets to be captured and stored in the current session; if this field is not completed, no filters will be applying to the interfaces. By clicking in the "Capture filter" button, a dialog will open and show a list of saved filters, giving to the user the alternative to add or delete more filters. This filter is not the same filter that appears in the Wireshark main screen, while a session is in progress. That filter will be explained later.

- *File*: this field could be used to specify the filename where the session will be stored; if this field is not completed, the session will be stored in a temporary file. By clicking on "Browse" button, a dialog will appear, allowing the user to browse the file system and select the destination of the file.

- *Use multiple files*: Wireshark has the ability to store the session across multiple files, depending on the criteria provided. By selecting this, four new options will appear as shown below:

  - *Next file every n mebibyte(s)*: Specify the maximum size in MiB of the capture file. Once this size is reached, a new file is created.

  - *Next file every n minute(s):* Specify the maximum time a capture file will be used. After this time is reached, a new file is created.

  - *Ring buffer with n files:* Specify the number of files that will be part of a ring buffer. This ring buffer allows to rotate the captures within this number of files.

  - *Stop capture after n file(s):* Specify the maximum number of files that Wireshark will use to store the current session. If this value is reached, the capture will stop.

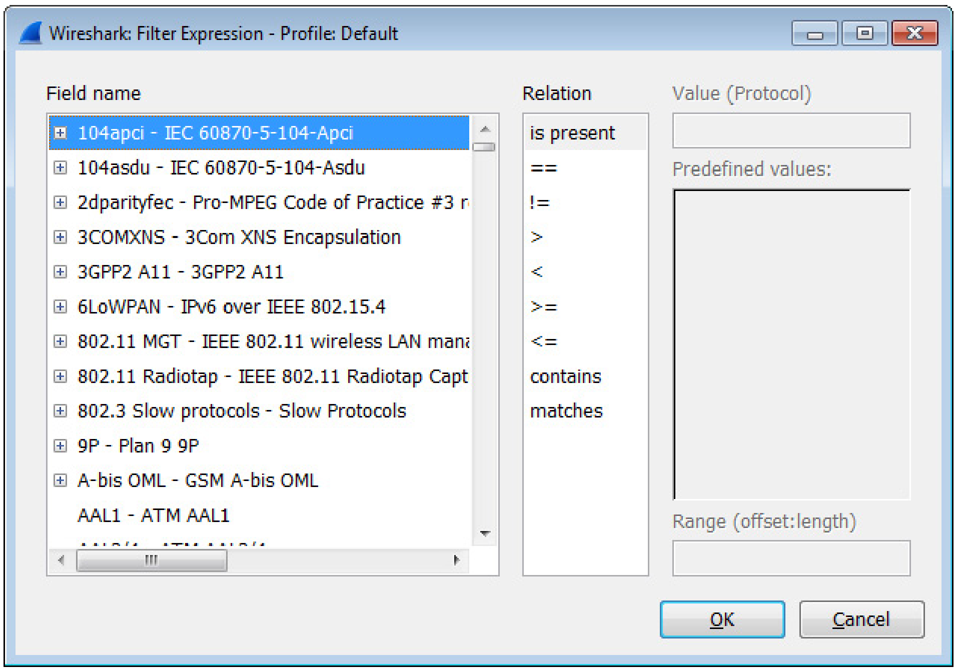
- *Stop Capture Automatically After:* this section contains 3 different options to automatically stop the packets captures.

    - *n packets:* Specify the maximum amount of packets that Wireshark will capture before stopping.
    - *n mebibyte (s):* Specify the maximum amount of MiB that Wireshark will capture before stopping.
    - *n minute (s):* Specify the maximum amount of time that Wireshark will capture packets before stopping.

- *Update list of packets in real time:* this option allows the user to specify if Wireshark will update the packet list pane while the capture is still active, or to not display the packets until the capture is stopped. If this option is enabled, Wireshark will use one process to capture the packets and a different process to display the packets in the packet list pane.
- *Automatically scroll in live capture:* this option allows the user to specify if Wireshark will perform an automated scroll while new packets are being captured. If this is not enabled, the new packets will not be shown in the packet list pane until the user scroll down.
- *Hide capture info dialog:* this option allows the user to specify if the info dialog will be displayed meanwhile a capture is in progress. The info dialog will show statistics of the protocols captured, and the time since the capture started
- *Resolve MAC Addresses:* this option allows the user to specify if the devices MAC address captured should be resolved into names.
- *Resolve network-layer names:* this option allows the user to specify if the network-layer names captured should be resolved into names.
- *Resolve transport-layer name:* this option allows the user to specify if the transport-layer names captured should be translated into protocols.
- *Use external network name resolver:* this option allows the user to specify if the name resolution will be performed through DNS lookups or not.

The second thing I would like to analyze before starting to capture packets is the necessity of configuring the network interface in promiscuous mode. Let's start with a short review of what is promiscuous mode and then identify whether this is needed or not. If we consider a network environment connected through a hub device, all the packets that came from one host will be sent to all the other hosts within the same network (similar to what happens in a wireless network environment). Every host that receives the packet compares the destination MAC address of the packet to the MAC address of its own network interface. If both MAC addresses match, then the network interface captures the packet and processes it. If the MAC addresses do not match, then the packet is discarded. If the promiscuous mode is enabled, every packet that arrives to the network interface, no matter what its MAC address is, it will be captured from the network. This allows us to capture all traffic that travel in the network. Now that we defined the promiscuous mode, let's use a more realistic scenario. Today it is not common to find hubs connecting networks; even it is getting difficult if you want to buy a hub in a computer store. Today's network uses switches to transmit messages only to the destination, avoiding flooding the network with unnecessary packets, and making it more difficult to sniff traffic. With a switching network, enabling the promiscuous mode will not have any effect, as we are not receiving traffic that is not intended to us. Considering this scenario, if we want to capture all the packets that travel across the network, we probably need to do one of the following: perform and ARP poisoning or connects to a mirroring port. These topics are out of the scope of this article, but I wanted to give a real world scenario before continuing.

Going back to our topic, the main concern is if it is required to listen in promiscuous mode, or not. There is not correct answer, as it depends of the nature of the tasks to be performed. In case of a network administration that wants to inspect all the traffic that travel across the network, then yes, the promiscuous mode will be necessary. But, if we consider a penetration tester that only wants to identify the traffic between his computer and a website, or a particular server, then this mode is not required, and enabling it will provide a lot of unnecessary traffic.


Wireshark: Filter Expression - Profile: Default

Field name
- 104apci - IEC 60870-5-104-Apci
- 104asdu - IEC 60870-5-104-Asdu
- 2dparityfec - Pro-MPEG Code of Practice #3 r
- 3COMXNS - 3Com XNS Encapsulation
- 3GPP2 A11 - 3GPP2 A11
- 6LoWPAN - IPv6 over IEEE 802.15.4
- 802.11 MGT - IEEE 802.11 wireless LAN mana
- 802.11 Radiotap - IEEE 802.11 Radiotap Capt
- 802.3 Slow protocols - Slow Protocols
- 9P - Plan 9 9P
- A-bis OML - GSM A-bis OML
- AAL1 - ATM AAL1

Relation
- is present
- ==
- !=
- >
- <
- >=
- <=
- contains
- matches

Value (Protocol)

Predefined values:

Range (offset:length)

Figure 6. Filter expression

Now that we have enough information to adapt the Wireshark to our necessities, it is time to start the sniffing. Once the capturing process is running, the packet list pane will display a lot of traffic, probably more than the desired one. This is when another of the big features of Wireshark comes to play. This tool provides an extensive set of filter options, allowing the user to display only the packets that he wants to see. The filter toolbar is above the packet list pane. By clicking on the "*Expression*" button, a dialog box will open with a complete list of filters that Wireshark is able to manage.

This list is a very good point to start for those who do not have expertise using Wireshark, as the user can easily select the fields he wants to use in the filters; and it is an excellent way to learn how to write those filters. This dialog is composed of 5 fields, as follows:

- *Field Name*: this field contains a list of protocols in a tree structure. Every protocol that has fields that could apply as filters are displayed at the top level of the tree. By expanding the protocol tree, the user will be able to get a list of the field names available for that protocol.

- *Relation*: this is the operator that will apply to this filter. The operators could be logical or comparison. Once a relation is chosen, the user will be able to complete one of the following 3 fields, related to the values.

- *Value*: this field contains the value that should be compared within the one in the packet, in order to Wireshark be able to match it. Between will appear the type of value (Character, string, number, etc.).

- *Predefined values:* for particular protocols and fields, there is a list of predefined values from where the user must select which is the desired one.

- *Range*: for some protocols it is possible to add a range of values to match with.

Within the filter toolbar there are 3 buttons that perform the following:

- The "*Clear*" button removes all the filters applied to the packet list pane.

- The "*Apply*" button use the filter typed in the text box.

- The "*Save*" button store the filter for a future use.

The text box in the left side of the toolbar could be used to write and apply filters. Instead of looking through the list of filters, the user could easily write the desired ones, if the proper syntax and names are known.
Be aware that these filters are case sensitive; for instance, it is not the same thing writing "*ip*", than "*IP*".
The filter contains 3 different items that compose the syntax. There is one field (usually a packet field), one operator and one value. One exception for this syntax is that the operator "is present", does not require a value to be compare with, as it returns true if the field is present within the packet. For example, if we want to filter only packets from or to the IP address 192.168.0.1, I will use the following filter

```
ip == 192.168.0.1
```

There is a list of the most common operators used in the filters

Comparison operators

Operator	C-like	Description
eq	==	equal to
ne	!=	not equal to
ge	>=	greater than or equal to
gt	>	greater than
le	<=	less than or equal to
lt	<	less than
matches		matches to
contain		contains
is present		is present

Logical operators

Operator	C-like	Description
and	&&	AND logical operation
or	||	OR logical operation
xor	^^	XOR logical operation
not	!	NOT logical operation
[...]	[...]	Substring operator

For some popular filters, instead of writing the ports used for that protocol, the user could write the protocol name instead. For example, it can be written *http* in the filter textbox, instead of the following rule (this is assuming that in the current session, there are no other http servers in ports besides 80 and 443)

```
tcp.port == 80 || tcp.port == 443
```

The list of popular protocols that could be used instead of filtering within the ports used is presented:

```
arp, bootp, smtp, pop3, dns, smb, ldap, ftp, icmp, imap, nbss
```

Wireshark also allows the definition of advanced filters that could be used to match specific bytes positions within the required fields. These bytes could be defined by the use of an offset and bytes length. The proper syntax for the advanced filters is shown below, followed by an operator and value to match with:

```
Packet field or protocol[offset:length]
```

For example, if I want to filter all the ip packets that contains in the bytes 10 and 11 the values 0x30 and 0x45, the filter will be

```
ip[10:2] == 30:45
```

The next list contains some examples of useful filters.

- *ip*: displays only ip packets.
- *tcp.port eq 23 or ftp*: displays only packets of ftp and telnet protocols.
- *ip.src==192.168.0.0/24 and ip.dst==192.168.0.0/24:* displays packets sent between hosts in the same network 192.168.0.0
- *ip.addr == 192.168.0.0/24:* displays packets from or to the IP range selected. Be aware that this is not the same than the previous one. This filter will display traffic from the LAN to an external host, and vice versa.
- *smb || nbns || dcerpc || nbss || dns:* displays packets that is usually associated with Windows hosts.
- *ls_ads.opnum==0x09:* displays packets associated with the Sasser worm.
- *eth.addr[0:3]==00:06:5B:* displays those packets where the MAC address starts with 0x00, 0x06 and 0x5B. This is useful to filter packets that came from a particular vendor's network interface.
- *http.request.uri matches "le.com$":* displays HTTP packets where the last characters in the URI field are "le.com".
- *not broadcast and not multicast:* displays all packets, except broadcast and multicast.
- *ether host AA:BB:CC:DD:EE:FF:* displays all Ethernet packets that goes to and from the network interface that has that MAC address.

Now that we know how to capture traffic and filter all the undesired packets, it is time to see what we could do with the traffic captured. Wireshark includes a sophisticated traffic analyzer and decoder. As more devices are being connected to personal and business networks, traffic analysis is becoming more important. From identifying a network misuse, to detect an intrusion, the ability to analyze the traffic is crucial for network administrators and security officers. Wireshark provides several functionalities for a better understanding of the traffic captured. These functionalities could be found within the toolbar, in the "*statistics*" menu. The most important ones will be explained below:

- *Summary*: this function provides general statistics about the current session. This information includes the timestamps of the first and last packets captured, information about the file where the session it is being saved, and statistics about all the traffic captured, such as amount of packets, average packets per second, and average bytes per second.
- *Protocol Hierarchy*: this function displays a tree structure of all the protocols captured in the current session. The hierarchy is based on the OSI network model. It will also provide some statistics of amount of packets and size per protocol. Be aware that usually, each packet has more than one protocol (each of them at a different OSI layer) and there are some situations where the same protocol could appear more than once in the same packet. Wireshark will count each of these occurrences as different packets.
- *Conversations*: this function displays a list of all the conversations captured and statistics for each of one, within the current session. A conversation is the traffic that is sent and received between two specific endpoints. The available tabs within this dialog box will be enabled depending the type of traffic captured and will show the number of conversations captured in the label; if no conversations were captured for a specific protocol, the tab will be greyed out.
- *Endpoints*: this function displays a list of all the endpoints captured and statistics for each of one, within the current session. An endpoint is each end of a conversation, which means that for each conversation we have identified, there are two endpoints. The available tabs within this dialog box will be enabled depending on the type of traffic captured, and will show the number of endpoints captured in the label; if no endpoints were captured for a specific protocol, the tab will be greyed out.
- *IO Graphs:* This function displays a customizable graphic. It has up to five different filters that could be applied to be represented in the graphic with a different color. By clicking on the graph, will show the selected packet in the Wireshark main window.

- *WLAN Traffic:* This function displays statistics of the captured wireless traffic within the current session. In the network overview section, statistics for each of the wireless networks (BSSIDs) will be displayed. By selecting a particular network, this will display information for each client connected to that wireless network and statistics for each of these.

Another good feature of Wireshark is the ability to intercept and reassemble VoIP packets. This functionality is located in the "*Telephony*" menu within the toolbar. This menu contains different VoIP protocols that could be used to filter packets in the packet list pane, and display only the desired one. Wireshark currently supports the following VoIP signaling protocols, including:

- *SIP*: Session Initiation Protocol (SIP) is a signaling protocol used to set up, manage and end VoIP calls, and multimedia conferences. This protocol includes methods such as INVITE and ACCEPT, and responses that indicate if the call was accepted, if it needs to be redirected, or error codes to indicate an abnormal situation.

- *H323*: is a standard that defines the protocol to provide call signaling and control for multimedia communication sessions.

- *ISUP*: ISDN User Part is part of the signaling protocol SS7, used to establish and release phone calls within the PSTN (Public Switched Telephone Network).

- *MGCP*: Media Gateway Control Protocol (MGCP) is a protocol used to control media gateways that lay on IP networks and are connected to the PSTN.

- *Unistim*: Unified Networks IP Stimulus is a telecommunications protocol designed by Nortel, used in communications between IP Phones and IP PBX (Private branch exchange).

Wireshark also supports the RTP protocol (Real-Time Transport Protocol) which is the protocol used to actual transmits packets in real time over an IP network. The type of packets supported by this protocol includes audio, video or data, over multicast or unicast network services. It is commonly used in streaming, telephony and even television services. It is used within RTCP (RTP Control Protocol) which monitor transmission statics and quality of service. Now, going back to the "*Telephony"* menu, the user will see a list of protocols available to select. Once a protocol is selected, a new window will appear, displaying, in most cases, the streams or packet counters depending on the selected protocol. This menu also contains one option that is "VoIP Calls". By selecting this option, a new window will appear, listing all the calls that were found during the captured session. Different attributes of the calls will be displayed, such as:

- *Start Time:* start time of the call.

- *Stop Time:* stop time of the call.

- *Initial Speaker:* the IP address from the device that initiated the call.

- *From*: depending on the signaling protocol used, it will display a telephone number or a terminal ID.

- *To*: depending on the signaling protocol used, it will display a telephone number or a terminal ID.

- *Protocol*: the protocol used to signal the call.

- *Packets*: number of packets that were sent and received during the call.

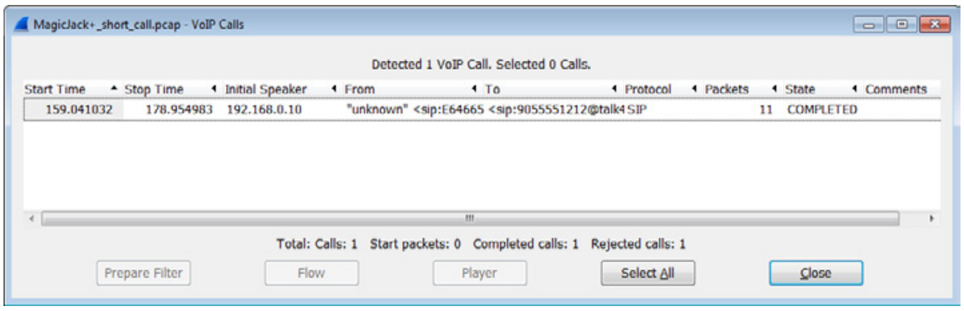
- *State*: the current state of the call. This value could be:

  - CALL SETUP: *indicates that the call is being setup.*

  - *RINGING: only for MGCP protocol*, indicates that the call is ringing.

  - IN CALL: indicates that the call is in place.

  - CANCELLED: indicates that the call was canceled before being connected.
  - COMPLETED: indicates that the call was ended normally.
  - REJECTED: indicates that the call was rejected by the addressee.
  - UNKNOWN: indicates that the call is in unknown state.
- *Comment*: additional comments for the call. If the protocol H323 is being used, this field will indicate if Fast Start or/and H245 Tunneling is being used.



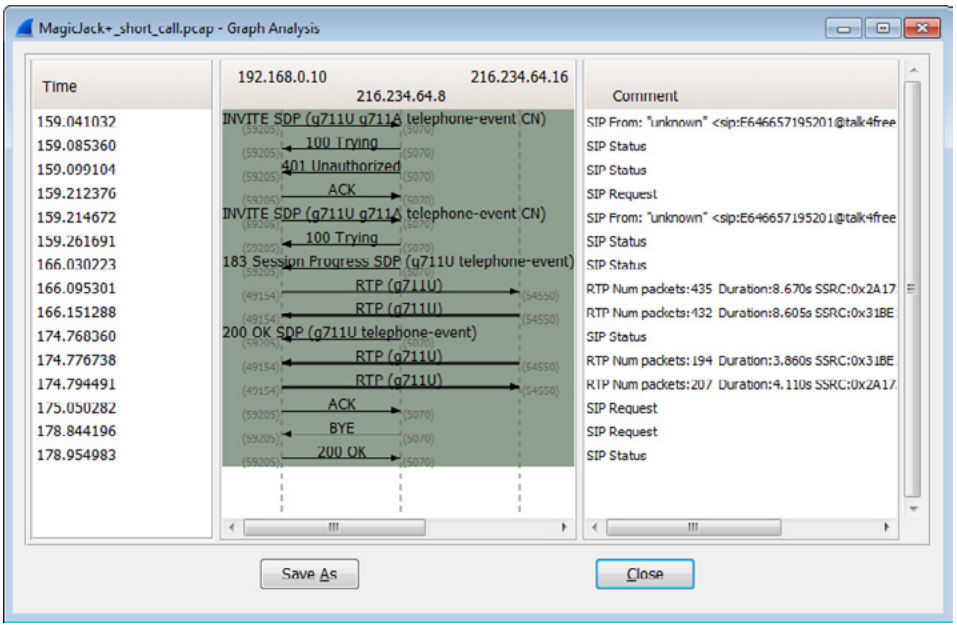
Detected 1 VoIP Call. Selected 0 Calls.

Start Time	Stop Time	Initial Speaker	From	To	Protocol	Packets	State	Comments
159.041032	178.954983	192.168.0.10	"unknown" <sip:E64665	<sip:9055551212@talk4	SIP	11	COMPLETED	

Total: Calls: 1 Start packets: 0 Completed calls: 1 Rejected calls: 1

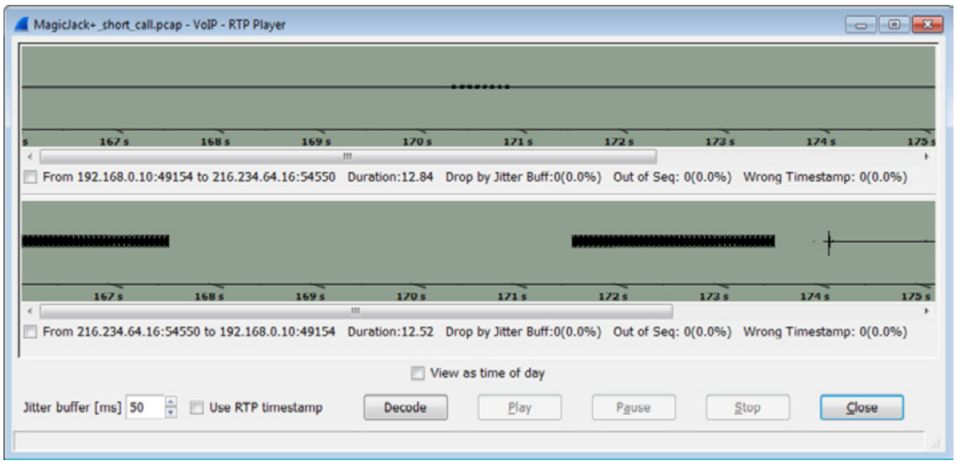
Figure 7. VoIP Call using SIP protocol

It is possible to filter all the packets for a particular call in the Wireshark packet list pane. In order to do so, within the VoIP window, the user just needs to select the desired call and click on the "*Prepare Filter*" button. Wireshark will automatically filter only the traffic that is being involved in that particular call. The "*flow*" button will present a graph analysis of the selected call. This analysis includes which packets were sent and received, at what time this was done, and which protocols were used in each of these packets.



Time	192.168.0.10 / 216.234.64.8 / 216.234.64.16	Comment
159.041032	INVITE SDP (g711U g711A telephone-event CN) (59205) → (5070)	SIP From: "unknown" <sip:E646657195201@talk4free
159.085360	100 Trying (59205) ← (5070)	SIP Status
159.099104	401 Unauthorized (59205) ← (5070)	SIP Status
159.212376	ACK (59205) → (5070)	SIP Request
159.214672	INVITE SDP (g711U g711A telephone-event CN) (59205) → (5070)	SIP From: "unknown" <sip:E646657195201@talk4free
159.261691	100 Trying (59205) ← (5070)	SIP Status
166.030223	183 Session Progress SDP (g711U telephone-event) (59205) ← (5070)	SIP Status
166.095301	RTP (g711U) (49154) → (54550)	RTP Num packets:435 Duration:8.670s SSRC:0x2A17
166.151288	RTP (g711U) (49154) ← (54550)	RTP Num packets:432 Duration:8.605s SSRC:0x31BE
174.768360	200 OK SDP (g711U telephone-event) (59205) ← (5070)	SIP Status
174.776738	RTP (g711U) (49154) ← (54550)	RTP Num packets:194 Duration:3.860s SSRC:0x31BE
174.794491	RTP (g711U) (49154) → (54550)	RTP Num packets:207 Duration:4.110s SSRC:0x2A17
175.050282	ACK (59205) → (5070)	SIP Request
178.844196	BYE (59205) ← (5070)	SIP Request
178.954983	200 OK (59205) → (5070)	SIP Status

Figure 8. VoIP Graph Analysis


MagicJack+_short_call.pcap - VoIP - RTP Player

From 192.168.0.10:49154 to 216.234.64.16:54550 Duration:12.84 Drop by Jitter Buff:0(0.0%) Out of Seq: 0(0.0%) Wrong Timestamp: 0(0.0%)

From 216.234.64.16:54550 to 192.168.0.10:49154 Duration:12.52 Drop by Jitter Buff:0(0.0%) Out of Seq: 0(0.0%) Wrong Timestamp: 0(0.0%)

View as time of day

Jitter buffer [ms] 50 Use RTP timestamp

Figure 9. Playing VoIP captured packets

Wireshark also provides a functionality to play the RTP packets captured, that could be accessed from the "*Player"* button. This option will open a dialog window displaying the actors involved in the call in separate lines and the user could play the reconstruction of the call from each part in a separate way or all the parts together (Figure 9).

If you have captured RTP stream packets, but not the SIP packets, it is probably that Wireshark may not recognize the traffic as RTP. For this scenario, Wireshark provides functionality, allowing the user to try and decode the RTP outside of conversations.

- Finally, but not least important, it's the ability to decrypt SSL (Secure Socket Layer) traffic. As you may know, SSL is a protocol that provides confidentiality and message integrity through a symmetric and asymmetric algorithm, and it is widely used on public and private networks. It can be used to encapsulate application layer protocols such as HTTP, SMTP, FTP, and so on. Let's give a quick overview to the SSL handshake, to understand how Wireshark is able to perform this decryption. The client start a connection to an SSL service by sending a *ClientHello message*, and includes its own SSL client configuration, including the protocol version and cipher settings

- The server receives this message and replies with a *ServerHello* message, including its own SSL server configuration, including the protocol version, cipher settings, and the server certificate

- The client authenticates the server certificate against the certificate authority. It also generates a master key, encrypts it with the server certificate, and sends to it

- The client and the server use the master key to generate a symmetric session key that will be used to encrypt and decrypt the information exchanged, and to verity its integrity. The way the symmetric session key is generated depends on the method used for key exchange (Diffie-Hellman, RSA, ECDH, etc.)

This is a quick explanation of how the SSL handshake is being performed. There are more factors involved, but for the purpose of decrypting traffic with Wireshark, this gives a good overview.

There is one warning I must say before continuing. Wireshark is not able to decrypt ALL SSL traffic that travel across the network. The SSL protocol provides integrity, and there is no way to decrypt its content. There are a few techniques that could be used to crack this protocol, but only works on specific conditions. Wireshark performs the decryption by having the private key that is being used to encrypt the traffic. There are also a couple more of requirements needed to perform this. The communication must use RSA key to encrypt the data, and the capture must include the handshake process (ClientHello and ServerHello requests).

Within this information Wireshark is able to decrypt the SSL traffic. In order to perform this, the user needs to access the Wireshark preferences, under the "*Edit*" menu. A dialog window will appear with a tree structure on the left side. Expand the "*Protocol*" tree, scroll down and select SSL. In the right side of the window, an edit button will appear. By clicking this, a new dialog window will appear; click in "new" within this window; and a new one will be presented, requiring the following information:

- IP address: is the IP address of the server that provides the SSL certificate to the client. Wildcard IP address could be used (0.0.0.0)
- Port: is the port used by the server to provide the encrypted service. Wildcard port could be used (0)
- Protocol: is the protocol that was encrypted by SSL. Considering a web traffic (HTTPs), the protocol encrypted was HTTP
- Key File: requires the location of the key file. The key must be in PEM or PKCS12 formats
- Password: is the password that was used to protect the key file, if any

After this configuration was saved, just start a new capture, and if everything was completed properly, Wireshark will decrypt all the SSL traffic that was encrypted using the selected certificate

As observed, Wireshark is a tool that provides several functionalities to analyze network environments. In this article we just covered the most important ones, but there are a lot more of things to do with this tool. I hope you enjoyed reading this article, as much as I enjoyed writing it. See you in the next article.

About the Author

ar.linkedin.com/in/sebasperez/
Sebastian is a Senior Security Consultant in one of the Big Four, located in Buenos Aires, with an experience of more than seven years providing IT Security services. He is responsible for delivering security solutions; including penetration testing, vulnerability assessment, security audits, security remediation, mobile and web application penetration testing, infrastructure security assessments, network analysis, as well as systems architecture design, review and implementation. He also offers internal trainings related to ATM security and penetration testing Android apps. Prior to joining into his current company, he worked as a systems administrator, security policy manager and IT security consultant. His background gives him a thorough understanding of security controls and their specific weaknesses. Furthermore, he is proficient in multiple security application tools, network technologies and operating systems. Sebastian has a post-degree in Information Security, and is currently working on his master degree thesis related to computer and mobile forensics. He also published a CVE # CVE-2012-4991 related to the Axway Secure Transport software, which was vulnerable to Path Traversal vulnerability.

Become a Big Data Master!

Over 45 HOW-TO, practical classes and tutorials to choose from!

Attend The 3rd Big Data TechCon!

The HOW-TO technical conference for professionals implementing Big Data

Come to Big Data TechCon to learn the best ways to:

- Process and analyze the real-time data pouring into your organization.
- Learn HOW TO integrate data collection technologies with data analytics and predictive analysis tools to produce the kind of workable information and reports your organization needs.
- Understand HOW TO leverage Big Data to help your organization today.
- Master Big Data tools and technologies like Hadoop, MapReduce, HBase, Cassandra, NoSQL databases, and more!
- Looking for Hadoop training? We have several Hadoop tutorials and dozens of Hadoop classes to get you started — or advanced classes to take you to the next level!

A BZ Media Event   Big Data TechCon

BigData TECHCON Boston

March 31-April 2, 2014

www.BigDataTechCon.com

Sniffing and Recovering Network Information Using Wireshark

by Fotis Liatsis

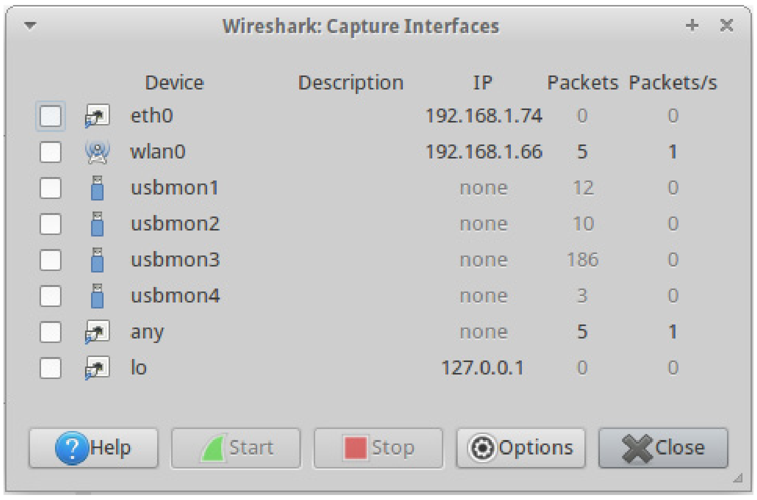
Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark is cross-platform, using the GTK+ widget toolkit to implement its user interface, and using pcap to capture packets, it runs on various Unix-like operating systems including Linux, OS X, BSD, and Solaris, and on Microsoft Windows.

You can download Wireshark for Windows or Mac OS X from the official website (*http://www.wireshark.org/download.html*). Most Linux systems have pre installed Wireshark tool. In the case that Wireshark is not installed you can just follow the bellow documentaiton and run the proper command for each operation system: Building and Installing Wireshark (*http://www.wireshark.org/docs/wsug_html_chunked/ChapterBuildInstall.html*). Wireshark needs to be run as the root user in your system. After Wireshark run, will give you a message – warning that you are running it as root, and that it might be dangerous.

Capture Interfaces

We can get an overview of the available local interfaces by navigating on Capture menu tab and then clicking the Interfaces option as shown Figure 1. By clicking the Option button Wireshark pops up the "Capture Options" dialog box. The table shows the settings for all available interfaces including a lot of information for each one and some checkboxes like:

- Capture on all interfaces – As Wireshark can capture on multiple interfaces, it is possible to choose to capture on all available interfaces.
- Capture all packets in promiscuous mode – This checkbox allows you to specify that Wireshark should put all interfaces in promiscuous mode when capturing.


Wireshark: Capture Interfaces

Device	Description	IP	Packets	Packets/s
☐ eth0		192.168.1.74	0	0
☐ wlan0		192.168.1.66	5	1
☐ usbmon1		none	12	0
☐ usbmon2		none	10	0
☐ usbmon3		none	186	0
☐ usbmon4		none	3	0
☐ any		none	5	1
☐ lo		127.0.0.1	0	0

Figure 1. Wireshark Interfaces

By clicking the Start button we will see a lot of packets start appearing in real time. Wireshark captures each packet sent from (Source) or to (Destination) our system.

User Interface

Before proceed to analyze our traffic network we will explain the basic information we need to know about the packet list pane, the color rules, the packet details pane and the packet bytes pane.

Packet List pane

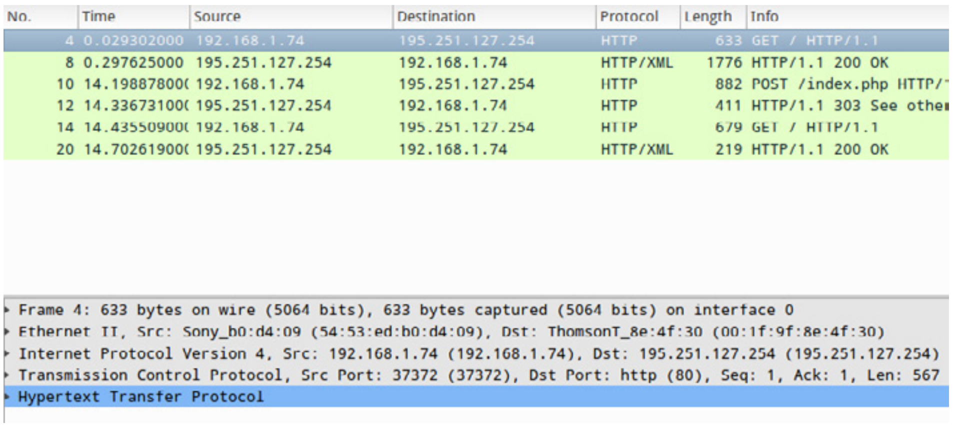
The packet list pane displays all the packets in the current capture file. Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed on Packet Details and Packet Bytes panes.

The default columns will show:

- No. – The number of the packet in the capture file. This number won't change, even if a display filter is used.
- Time – The timestamp of the packet. The presentation format of this timestamp can be changed.
- Source – The address where this packet is coming from.
- Destination – The address where this packet is going to.
- Protocol – The protocol name in a short (perhaps abbreviated) version.
- Info – Additional information about the packet content.

Color Rules

A very useful mechanism available in Wireshark is packet colorization. There are two types of coloring rules in Wireshark; temporary ones that are only used until you quit the program, and permanent ones that will be saved to a preference file so that they are available on a next session. So let's focus to most important name filters. Green Color refers to TCP packets but black identifies corrupted TCP packets. Light Blue refers to UDP packets and dark blue on DNS traffic. For more information or if we would like to edit/add our own color rules we can navigate to *View menu* and click the *Coloring Rules.*



No.	Time	Source	Destination	Protocol	Length	Info
4	0.029302000	192.168.1.74	195.251.127.254	HTTP	633	GET / HTTP/1.1
8	0.297625000	195.251.127.254	192.168.1.74	HTTP/XML	1776	HTTP/1.1 200 OK
10	14.19887800(	192.168.1.74	195.251.127.254	HTTP	882	POST /index.php HTTP/
12	14.33673100(	195.251.127.254	192.168.1.74	HTTP	411	HTTP/1.1 303 See other
14	14.43550900(	192.168.1.74	195.251.127.254	HTTP	679	GET / HTTP/1.1
20	14.70261900(	195.251.127.254	192.168.1.74	HTTP/XML	219	HTTP/1.1 200 OK

```
▸ Frame 4: 633 bytes on wire (5064 bits), 633 bytes captured (5064 bits) on interface 0
▸ Ethernet II, Src: Sony_b0:d4:09 (54:53:ed:b0:d4:09), Dst: ThomsonT_8e:4f:30 (00:1f:9f:8e:4f:30)
▸ Internet Protocol Version 4, Src: 192.168.1.74 (192.168.1.74), Dst: 195.251.127.254 (195.251.127.254)
▸ Transmission Control Protocol, Src Port: 37372 (37372), Dst Port: http (80), Seq: 1, Ack: 1, Len: 567
▸ Hypertext Transfer Protocol
```

Figure 2. List – Details Pane

Packet Details Pane

The packet details pane shows the current packet (selected in the "Packet List" pane) in a more detailed form. This pane shows the protocols and protocol fields of the packet selected in the "Packet List" pane. The protocols and fields of the packet are displayed using a tree, which can be expanded and collapsed.

Packet Bytes Pane

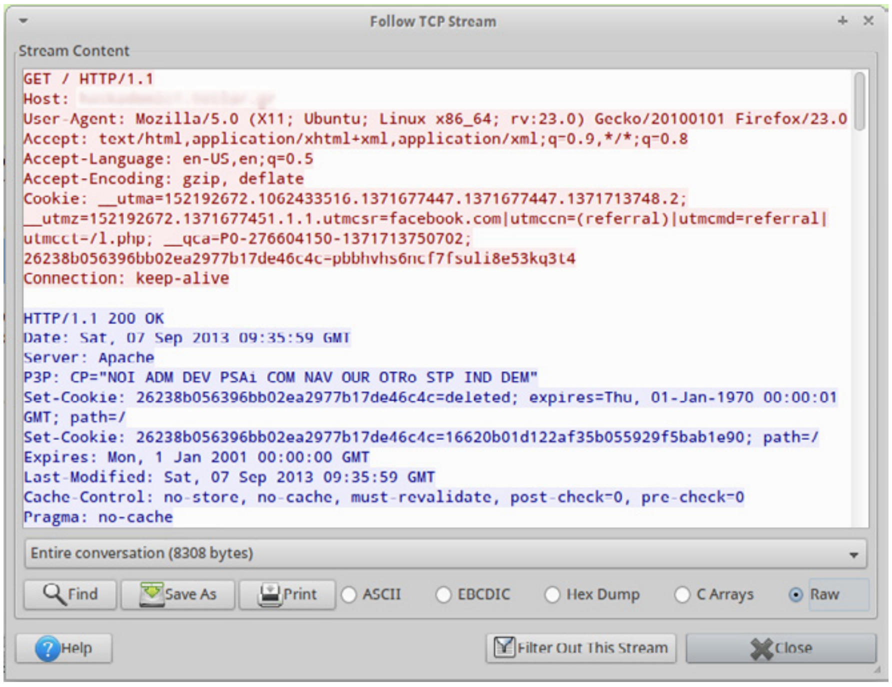
The packet bytes pane shows the data of the current packet in a hexdump style. The left side shows the offset in the packet data, in the middle the packet data is shown in a hexadecimal representation and on the right the corresponding ASCII characters are displayed.

Start Capturing – Analyzing

At this part we will start capturing once more our network, so click from *Capture menu* the *Start* option. Next we will attempt to login to an account and will analyze it into wireshark tool to see if we can find important information. As we can see there are lot of packets that Wireshark appears. A valuable options here is the Filter mechanism which lets us quickly edit and apply-display filters. Let's isolate the http packets by typing http string on filter tab. As we can see the packet list pane shows only HTTP protocols. We need to locate the HTTP protocol and identify the response of the Host which attempted to login. Looking at the highlighted results, we can determine at info tab that there are packages which contain the GET method. Let's focus at this information and explain it.

Note

GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. At the packet list pane click the *Hypertext Transfer Protocol*, as we can see the GET method appears and also a lot of important information such as the request version of the Server, the Host and the User-Agent which contains the browser version and the OS that the user used to login. Next we want to examine the full conversation between the client and the server by accessing the *Follow TCP Stream option* (right click on the packet and then choose Follow TCP Stream). A pop-up window will appear which will contain the entire conversation on stream content. The red words indicates the request and the blue the response of the Host. Also as we can notice choosing the Follow TCP Stream option Wireshark automatically added the property filter in Filter area.



```
GET / HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: __utma=152192672.1062433516.1371677447.1371677447.1371713748.2;
__utmz=152192672.1371677451.1.1.utmcsr=facebook.com|utmccn=(referral)|utmcmd=referral|
utmcct=/l.php; __qca=P0-276604150-1371713750702;
26238b056396bb02ea2977b17de46c4c=pbbhvhs6ncf7fsuli8e53kq3t4
Connection: keep-alive

HTTP/1.1 200 OK
Date: Sat, 07 Sep 2013 09:35:59 GMT
Server: Apache
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Set-Cookie: 26238b056396bb02ea2977b17de46c4c=deleted; expires=Thu, 01-Jan-1970 00:00:01
GMT; path=/
Set-Cookie: 26238b056396bb02ea2977b17de46c4c=16620b01d122af35b055929f5bab1e90; path=/
Expires: Mon, 1 Jan 2001 00:00:00 GMT
Last-Modified: Sat, 07 Sep 2013 09:35:59 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
```

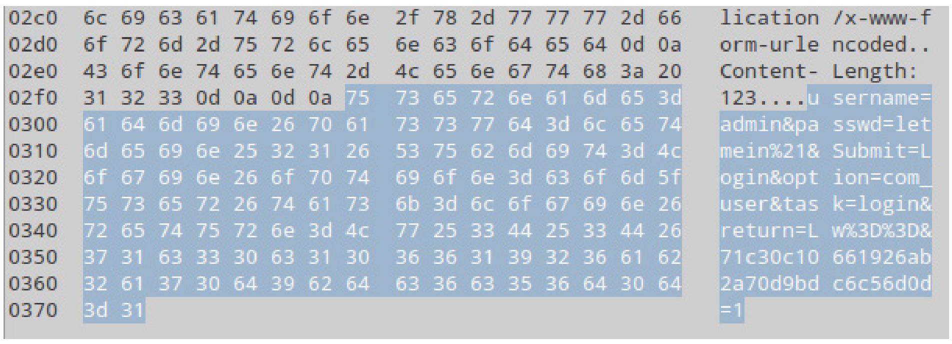
Figure 3. TCP Stream Window

By reviewing the highlighted code closely on Figure 3 we can see that the index.php action has two inputs, the username and the password. We can identify on Packet List pane a POST Request method from our machine to the server using HTTP protocol. Selecting once more the Hypertext Transfer Protocol tree we can verify the request and the method which used to login to the Host.

Note

POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data POSTed might be, as examples, an annotation for existing resources; a message for a bulletin board, newsgroup, mailing list, or comment thread; a block of data that is the result of submitting a web form to a data-handling process; or an item to add to a database.

As we can notice on packet details pane there is also a new tree line named *Line-based text data:* By clicking once it we can see the POST request which contains the username and the password in clear text. Also checking the packet bytes pane we can draw the same information on Hex or Bit View.



```
02c0  6c 69 63 61 74 69 6f 6e  2f 78 2d 77 77 77 2d 66   lication /x-www-f
02d0  6f 72 6d 2d 75 72 6c 65  6e 63 6f 64 65 64 0d 0a   orm-urle ncoded..
02e0  43 6f 6e 74 65 6e 74 2d  4c 65 6e 67 74 68 3a 20   Content- Length:
02f0  31 32 33 0d 0a 0d 0a 75  73 65 72 6e 61 6d 65 3d   123....u sername=
0300  61 64 6d 69 6e 26 70 61  73 73 77 64 3d 6c 65 74   admin&pa sswd=let
0310  6d 65 69 6e 25 32 31 26  53 75 62 6d 69 74 3d 4c   mein%21& Submit=L
0320  6f 67 69 6e 26 6f 70 74  69 6f 6e 3d 63 6f 6d 5f   ogin&opt ion=com_
0330  75 73 65 72 26 74 61 73  6b 3d 6c 6f 67 69 6e 26   user&tas k=login&
0340  72 65 74 75 72 6e 3d 4c  77 25 33 44 25 33 44 26   return=L w%3D%3D&
0350  37 31 63 33 30 63 31 30  36 36 31 39 32 36 61 62   71c30c10 661926ab
0360  32 61 37 30 64 39 62 64  63 36 63 35 36 64 30 64   2a70d9bd c6c56d0d
0370  3d 31                                              =1
```

Figure 4. Bytes Pane

Cracking – Analyzing W-Network

At this part of article we will explain how we can have access to our WLAN network, how to retrieve the wireless password and finally how can use it to analyze the traffic packets into Wireshark.

First we will run the following command to get a list of our network interfaces:

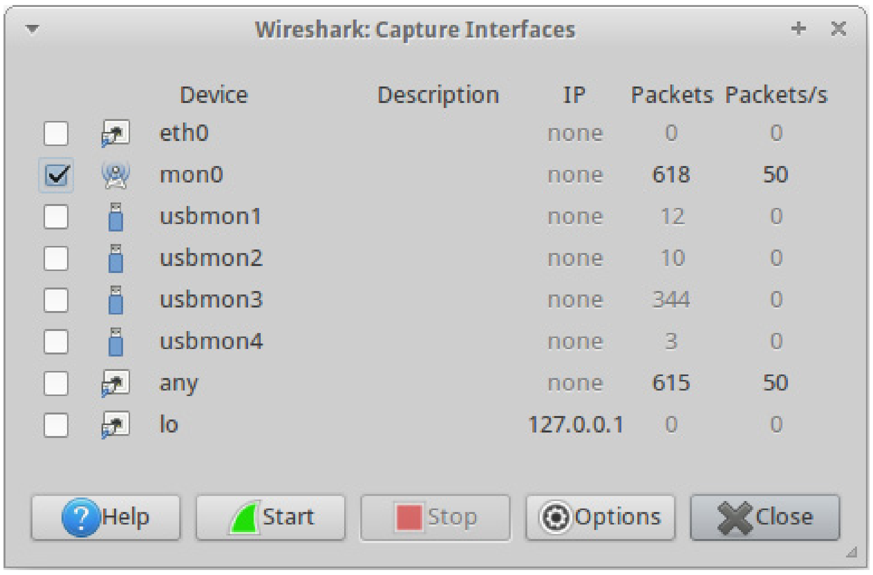
```
wizard32@wizard32:~$ sudo airmon-ng
Interface     Chipset               Driver
wlan0         Unknown       iwlwifi - [phy0]
```

As we can notice the only available interface is the wlan0 adapter. To capture network traffic without being associated with an access point, we need to set the wireless network adapter in monitor mode (Listing 1).

Listing 1. Setting wireless network adapter in monitor mode

```
wizard32@wizard32:~$ sudo airmon-ng start wlan0

Found 4 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!

PID    Name
1103   NetworkManager
1121   avahi-daemon
1125   avahi-daemon
1299   wpa_supplicant


Interface     Chipset               Driver

wlan0         Unknown       iwlwifi - [phy0]
                     (monitor mode enabled on mon0)
```

Next run once more the Wireshark tool and navigate to Capture menu and click Interfaces option. As we mention before monitor mode enabled on mon0 so on wireshark pop-up window select the mon0 as capture interface and click start (Figure 5). After starting the capturing we locate multiple SSID access points. By typing HTTP or DNS on Filter menu, Wireshark doesn't return any result. Looking on packet pist pane we can search our access point or by locating the BSSID (basic service set identification) or the SSID (service set identifier).



	Device	Description	IP	Packets	Packets/s
	eth0		none	0	0
✓	mon0		none	618	50
	usbmon1		none	12	0
	usbmon2		none	10	0
	usbmon3		none	344	0
	usbmon4		none	3	0
	any		none	615	50
	lo		127.0.0.1	0	0

Figure 5. Wireshark Interfaces

- BSSID is the MAC address of the wireless access point (WAP) generated by combining the 24 bit Organization Unique Identifier and the manufacturer's assigned 24-bit identifier for the radio chipset in the WAP.
- SSID is the name of a wireless local area network (WLAN).

As we can notice two new tree lines has been added on packet details pane. Both of them specifies the communication wireless protocol.

Another way to locate out access point is to use the airdump-ng tool.

```
wizard32@wizard32:~$ sudo airodump-ng mon0
BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID
00:11:8F:8E:4E:32  -30       21        0    0   1  54   WEP  WEP         wizard32
```

To capture data into a file using airodump-ng tool once more, we must specify some additional option to target a specific access point.

```
wizard32@wizard32:~$ sudo airodump-ng -c 1 -w ~/Desktop/W-packets --bssid 00:11:8F:8E:4E:32 mon0
```

At this time we can use two different ways to retrieve the password from our network. The first one is to use a tool named aircrack-ng in association with the .pcap packets that we captured using the aiodump-ng tool or using the .pcap file from Wireshark tool and performing a dictionary attack to a specific access point. Let's analyze them.

```
Method: aircrack-ng
```

To recover the WEP key aircrack only requiring collection of enough data. So in terminal we type the following command to retrieve our WEP network key: Listing 2. As we can see aircrack decrypted and found correctly our WEP network key. So let's analyze how we can retrieve it using this time dictionary attack method on .pcap Wireshark file (Listing 3).

```
-w: Identifies our wordlist file
```

Listing 2. Retrieving WEP network key

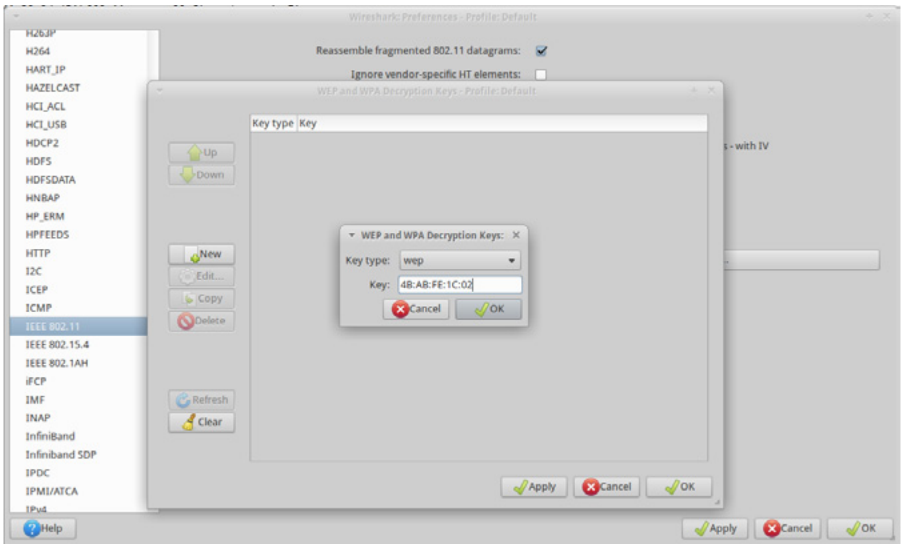
```
wizard32@wizard32:~$ sudo aircrack-ng ~/Desktop/W-packets-01*.cap
Opening /home/wizard32/Desktop/W-packets-01.cap
Read 61960 packets.

   #  BSSID              ESSID                     Encryption

   1  00:11:8F:8E:4E:32  wizard32                  WEP (21124 IVs)

Choosing first network as target.

Opening /home/wizard32/Desktop/W-packets-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 21124 ivs.

                    Aircrack-ng 1.1
                                                  00:00:02] Tested 7 keys (got 21124 IVs)

   KB    depth   byte(vote)
    0    0/  1   4B(29696) E4(28160) 40(27648) C2(27392) D6(26368) 21(26112) 62(25344) A8(25344)
   B3(25344) DB(25344) 2C(25088) 38(25088) A9(25088) 47(24832) C4(24832) CB(24832) CE(24832)
   19(24320) 44(24320)
   [...]
    4    0/  2   C4(29440) 12(28928) 78(28160) 87(27136) 60(26368) 84(26368) 93(25856) 00(25600)
   4C(25600) BD(25344) C5(25344) 03(25088) 68(25088) 7B(25088) F4(25088) 02(24832) 1E(24832)
   28(24832) 54(24832)
   [...]

                         KEY FOUND! [ 4B:AB:FE:1C:02 ]
   Decrypted correctly: 100%
```

Listing 3. Retrieving WEP network key using dictionary attack method

```
wizard32@wizard32:~$ sudo aircrack-ng -w ~/Desktop/mywordlist.txt -b 00:11:8F:8E:4E:32 ~/Desktop/
   W-capture.pcap
Opening /home/wizard32/Desktop/W-capture.pcap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 21096 ivs.

                                                                     Aircrack-ng 1.1
                                                         [00:00:02] Tested 7 keys (got 21096 IVs)

   KB    depth   byte(vote)
    1    0/  1   AB(34816) 32(27904) C6(27648) B0(26624) 12(26112) 16(25600) 28(25600) B1(25600)
   CD(25344) F5(25344) 60(25088) D0(25088) E1(25088) D4(24832) 20(24576) 10(24320) 82(24320)
   21(24064) 4A(24064)
   [...]
    2    2/  3   FE(27648) 4A(26624) B9(25600) EB(25600) 0D(25344) 2A(25344) 3A(25344) 46(25088)
   25(24832) 7B(24832) 8E(24832) 9A(24832) AF(24832) 01(24576) C1(24576) 5E(24320) 78(24320)
   8F(24320) BD(24320)
   [...]

                         KEY FOUND! [ 4B:AB:FE:1C:02 ]
   Decrypted correctly: 100%
```

Note

Some of these tools (airmon-ng) might need to be installed unless if we are using a system which has airmon-ng already installed, such as BackTrack/Kali or BackBox.



Reassemble fragmented 802.11 datagrams: ☑
Ignore vendor-specific HT elements: ☐
Key type: wep
Key: 4B:AB:FE:1C:02

Figure 6. Decryption Keys Pane

On both cases aircrack successfully recovered the WEP key. Now it's time to apply our WEP key into Wireshark tool to enable decryption in order to locate possible sensitive information. Navigate to *Edit menu*, then click on *Preferences* option and on *Protocol* tree line locate the *IEEE 802.11* protocol. Next we mark *Enable decryption* checkbox and then we click the Edit button to add our WEP key.

The Moment of Truth (TMT)

We searching once more for possible http || dns protocols. By reviewing the highlighted code closely on figure 2 we can see multiple http request to a specific host. To eliminate even more results we will create a new filter which will specify only those packages from the specific Host. So we locate the GET request and we apply as filter the selected line. As previously we locate the line which contains the parameters (username/password). Notice that on packet bytes pane the Frame tab and the Decrypted WEP data tab appearing.

Table 1. POST info request

Key	Value
task:	login
username:	Admin
passwd:	l3tmeIn!

Protect from Snooping

All the above examples shows how easy it is to obtain sensitive data from snooping on a connection. The best way to prevent this is to encrypt the data that's being sent. The most known encryption methods are SSL (Secure Sockets Layer) and TLS (Transport Layer Security).

The Secure Socket Layer (SSL) and Transport Layer Security (TLS) are the most widely deployed security protocol used today. It is essentially a protocol that provides a secure channel between two machines operating over the Internet or an internal network. SSL Certificates have a key pair: a public and a private key. These keys work together to establish an encrypted connection. The certificate also contains what is called the "subject," which is the identity of the certificate/website owner.

About the Author

Fotis Liatsis is a member and System/Network Administrator of Greek Student Security Team – CampSec. Hi is also an OWASP member of the Greek Student Chapter and System Administrator of Hackademic. His interests include Ethical Hacking, Penetration Testing, Security/vulnerability research and applied Cryptography methods. He is a security enthusiast and Athcon fanatic. You can follow his Twitter (@liatsisfotis) or his blog for more information (http://www.liatsisfotis.com/).

Traffic Analysis and Capture Passwords

by Rafael Fontes

It is known that Wireshark is a powerful tool that goes far beyond a simple sniffer. What many do not know is that there are several ways to harness the potential of this tool, readers, this article will introduce. Let us learn to sniff the network effectively, create filters to find only the information we want, see it as a black hat would use this tool to steal passwords and finally, how to use Wireshark to diagnose network problems or if a firewall is blocking packets correctly.

Your password is hard to be broken? Has many characters and you trade with a certain regularity and one day you're surprised to receive allegations of invasion. Evidence indicates that the invasions third party accounts departed from your account and you have no idea what is happening. That is, someone may have made use of your account and performed such acts as you. How could this have happened? A strong possibility is that you have been the victim of an attack of "sniffer".

UNDESTAND THE MAIN CONCEPT

What are Sniffers? Well... Are very useful software, so great is the use of them, even the IDS systems are made based sniffers. A sniffer is a program that can capture all traffic passing in a segment of a network.

Programs that allow you to monitor network activity recording names (username and password) each time they access other computers on the network.

These programs are monitoring ("sniffing") network traffic to capture access to network services, such as remote mail service (IMAP, POP), remote access (telnet, rlogin, etc.), file transfer (FTP) etc. Accesses made, captured packets. Always aiming to get identification for access the user's account.

When we called the HUB computer and send information from one computer to another, in reality these data are for all ports of the HUB, and therefore for all machines. It turns out that only the machine on which the information was intended to send the operating system.

If a sniffer was running on other computers, even without these systems send the information travels there for the operating system, the sniffer will intercede at the network layer, data capturing and displaying them to the user, unfriendly way. Generally the data are organized by type of protocol (TCP, UDP, FTP, ICMP, etc...) and each package shown may have read your content.

YOUR PASSWORD CAN BE CAPTURED BY SNIFFERS

Many local area networks (LANs) are configured sharing the same Ethernet segment. Virtually any computer of the network can run a "sniffer" program to "steal" users passwords. "Sniffers" work monitoring the flow of communication between computers on the network to find out when someone uses the network services previously mentioned. Each of these services uses a protocol that defines how a session is established, such as your account is identified and authenticated and how to use the service.

To have access to these services, you first have to have a "log in". Is the login sequence - the part of these authentication protocols, which occurs at the beginning of each session - the "sniffers" are concerned, because it is this part that is your password. Therefore, it is the only filter "strings" keys that the password is obtained.

STEP BY STEP

Currently, almost all environments using switches and not hubs, which makes sniffing a little more difficult because the switches do not send the data to all ports as a hub does, it sends directly to the port where the host destination, so if you try to sniff a network switch you will only hear what is broadcast, or its own connection. To be able to hear everything without being the gateway of the network, an ARP spoof attack is necessary, or burst the CAM table of the switch.

Basic Usage

Now let's put our hands dirty: I'm assuming you already have the program installed, if you do not download. When starting Wireshark, the displayed screen will look something like Figure 1:

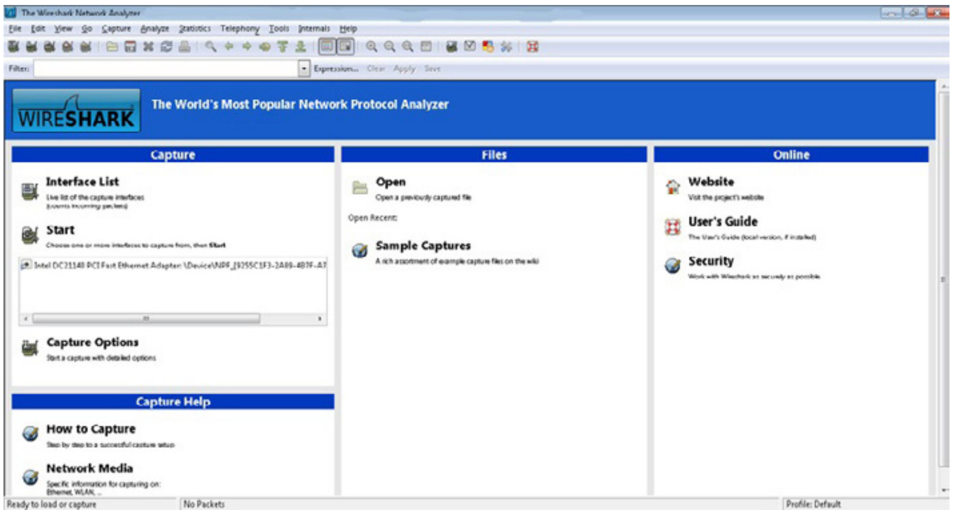

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Capture
Interface List
Start
Intel DC21140 PCI Fast Ethernet Adapter: \Device\NPF_{9255C1F3-3A89-4B7F-A7
Capture Options
Capture Help
How to Capture
Network Media
Files
Open
Open Recent:
Sample Captures
Online
Website
User's Guide
Security
Ready to load or capture
No Packets
Profile: Default

Figure 1. Wireshark

Before you can start capturing packets, we have to define which interface will "listen" to the traffic. Click *Capture > Interfaces*

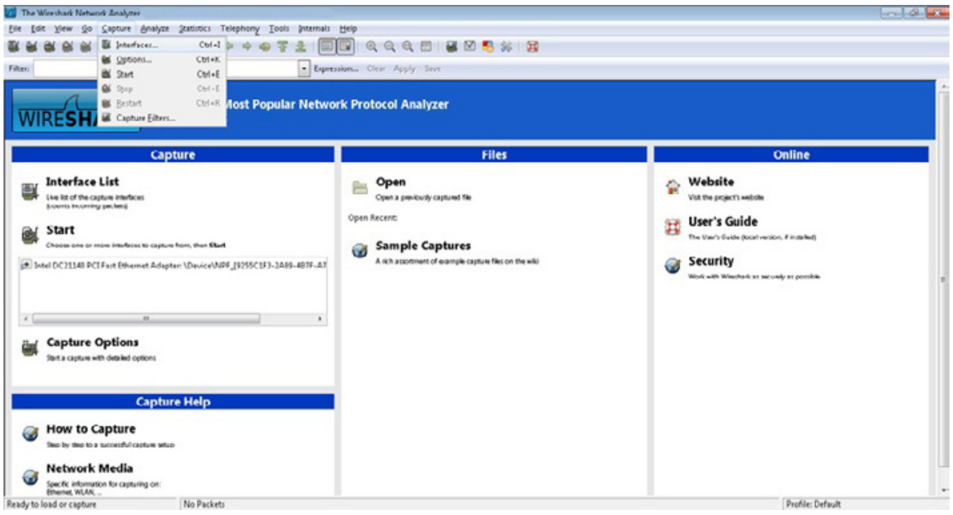
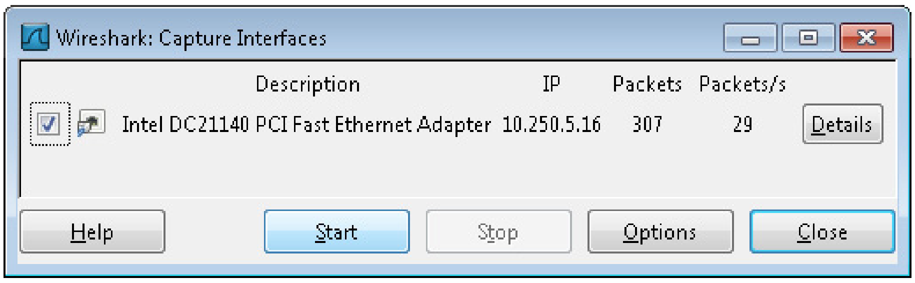


Figure 2. Interfaces

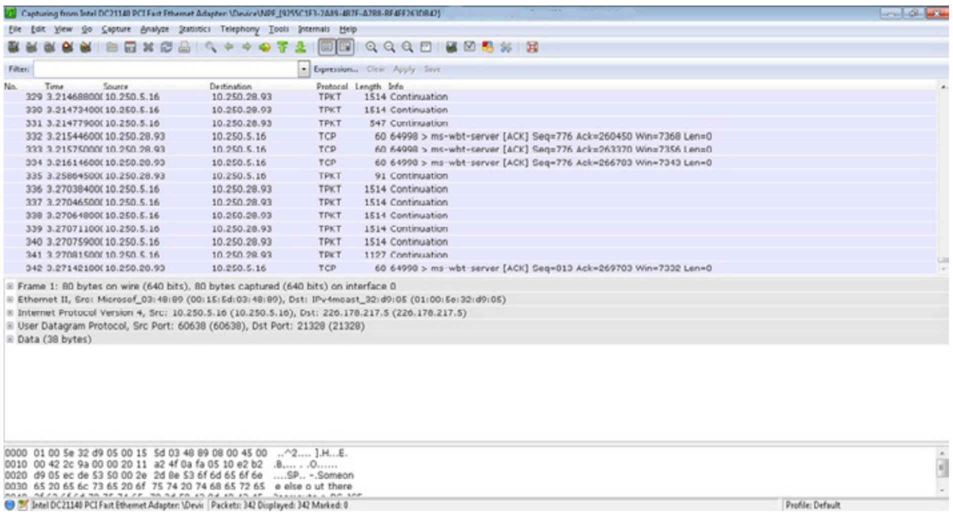
From there, a new window will appear with the list of automatically detected interfaces, simply select the desired interface by clicking the box next to the name of the interface, as in Figure 3:



```
Wireshark: Capture Interfaces

    Description                              IP           Packets  Packets/s
[✓] Intel DC21140 PCI Fast Ethernet Adapter  10.250.5.16  307      29         Details

Help    Start    Stop    Options    Close
```

Figure 3. Capture Interfaces

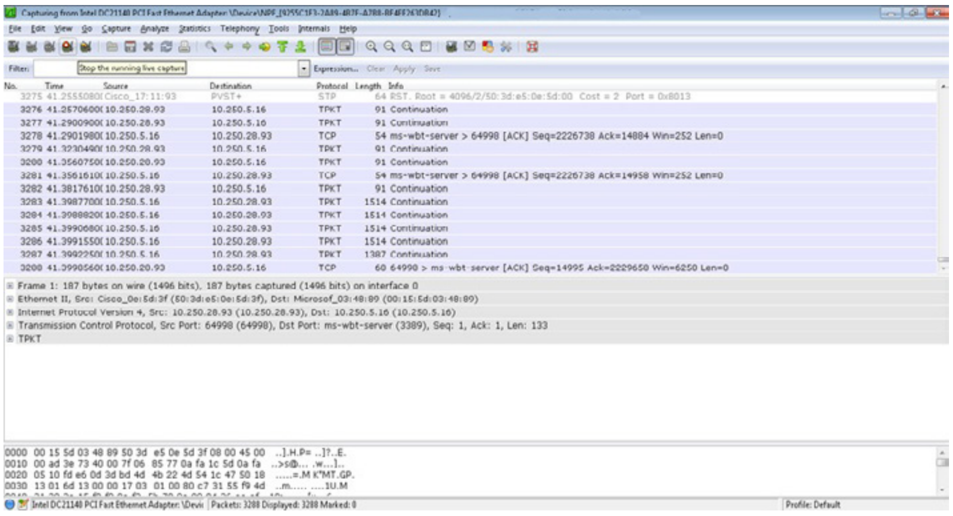
If you click Start, it will begin automatically captures. You can only select the interface and only then start the capture if necessary. When the capture process starts, you will see several packets traversing the screen Wireshark (varying according to the traffic of your machine / network). Will look something like the Figure 4:



```
No.  Time        Source         Destination    Protocol Length Info
 329 3.21468800( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
 330 3.21473400( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
 331 3.21477900( 10.250.5.16    10.250.28.93   TPKT      547 Continuation
 332 3.21544600( 10.250.28.93   10.250.5.16    TCP        60 64998 > ms-wbt-server [ACK] Seq=776 Ack=260450 Win=7368 Len=0
 333 3.21575000( 10.250.28.93   10.250.5.16    TCP        60 64998 > ms-wbt-server [ACK] Seq=776 Ack=263370 Win=7356 Len=0
 334 3.21614600( 10.250.28.93   10.250.5.16    TCP        60 64998 > ms-wbt-server [ACK] Seq=776 Ack=266703 Win=7343 Len=0
 335 3.25864500( 10.250.28.93   10.250.5.16    TPKT       91 Continuation
 336 3.27038400( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
 337 3.27046500( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
 338 3.27064800( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
 339 3.27071100( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
 340 3.27075900( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
 341 3.27081500( 10.250.5.16    10.250.28.93   TPKT     1127 Continuation
 342 3.27142100( 10.250.28.93   10.250.5.16    TCP        60 64998 > ms-wbt-server [ACK] Seq=813 Ack=269703 Win=7332 Len=0

⊞ Frame 1: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
⊞ Ethernet II, Src: Microsof_03:48:89 (00:15:5d:03:48:89), Dst: IPv4mcast_32:d9:05 (01:00:5e:32:d9:05)
⊞ Internet Protocol Version 4, Src: 10.250.5.16 (10.250.5.16), Dst: 226.178.217.5 (226.178.217.5)
⊞ User Datagram Protocol, Src Port: 60638 (60638), Dst Port: 21328 (21328)
⊞ Data (38 bytes)

0000  01 00 5e 32 d9 05 00 15  5d 03 48 89 08 00 45 00   ..^2.... ].H...E.
0010  00 42 2c 9a 00 00 20 11  a2 4f 0a fa 05 10 e2 b2   .B.... . .O......
0020  d9 05 ec de 53 50 00 2e  2d 8e 53 6f 6d 65 6f 6e   ....SP.. -.Someon
0030  65 20 65 6c 73 65 20 6f  75 74 20 74 68 65 72 65   e else o ut there

Intel DC21140 PCI Fast Ethernet Adapter: \Devi | Packets: 342 Displayed: 342 Marked: 0 | Profile: Default
```

Figure 4. Capturing

To stop the capture, simply click the button, "Stop the running live capture".



```
Filter: Stop the running live capture

No.  Time        Source         Destination    Protocol Length Info
3275 41.2555080( Cisco_17:11:93 PVST+          STP        64 RST. Root = 4096/2/50:3d:e5:0e:5d:00  Cost = 2  Port = 0x8013
3276 41.2570600( 10.250.28.93   10.250.5.16    TPKT       91 Continuation
3277 41.2900900( 10.250.28.93   10.250.5.16    TPKT       91 Continuation
3278 41.2901980( 10.250.5.16    10.250.28.93   TCP        54 ms-wbt-server > 64998 [ACK] Seq=2226738 Ack=14884 Win=252 Len=0
3279 41.3230490( 10.250.28.93   10.250.5.16    TPKT       91 Continuation
3280 41.3560750( 10.250.28.93   10.250.5.16    TPKT       91 Continuation
3281 41.3561610( 10.250.5.16    10.250.28.93   TCP        54 ms-wbt-server > 64998 [ACK] Seq=2226738 Ack=14958 Win=252 Len=0
3282 41.3817610( 10.250.28.93   10.250.5.16    TPKT       91 Continuation
3283 41.3987700( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
3284 41.3988820( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
3285 41.3990680( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
3286 41.3991550( 10.250.5.16    10.250.28.93   TPKT     1514 Continuation
3287 41.3992250( 10.250.5.16    10.250.28.93   TPKT     1387 Continuation
3288 41.3990560( 10.250.28.93   10.250.5.16    TCP        60 64998 > ms-wbt-server [ACK] Seq=14995 Ack=2229650 Win=6250 Len=0

⊞ Frame 1: 187 bytes on wire (1496 bits), 187 bytes captured (1496 bits) on interface 0
⊞ Ethernet II, Src: Cisco_0e:5d:3f (50:3d:e5:0e:5d:3f), Dst: Microsof_03:48:89 (00:15:5d:03:48:89)
⊞ Internet Protocol Version 4, Src: 10.250.28.93 (10.250.28.93), Dst: 10.250.5.16 (10.250.5.16)
⊞ Transmission Control Protocol, Src Port: 64998 (64998), Dst Port: ms-wbt-server (3389), Seq: 1, Ack: 1, Len: 133
⊞ TPKT

0000  00 15 5d 03 48 89 50 3d  e5 0e 5d 3f 08 00 45 00   ..].H.P= ..]?..E.
0010  00 ad 3e 73 40 00 7f 06  85 77 0a fa 1c 5d 0a fa   ..>s@... .w...]..
0020  05 10 fd e6 0d 3d bd 4d  4b 22 4d 54 1c 47 50 18   .....=.M K"MT.GP.
0030  13 01 6d 13 00 00 17 03  01 00 80 c7 31 55 f9 4d   ..m..... ....1U.M

Intel DC21140 PCI Fast Ethernet Adapter: \Devi | Packets: 3288 Displayed: 3288 Marked: 0 | Profile: Default
```

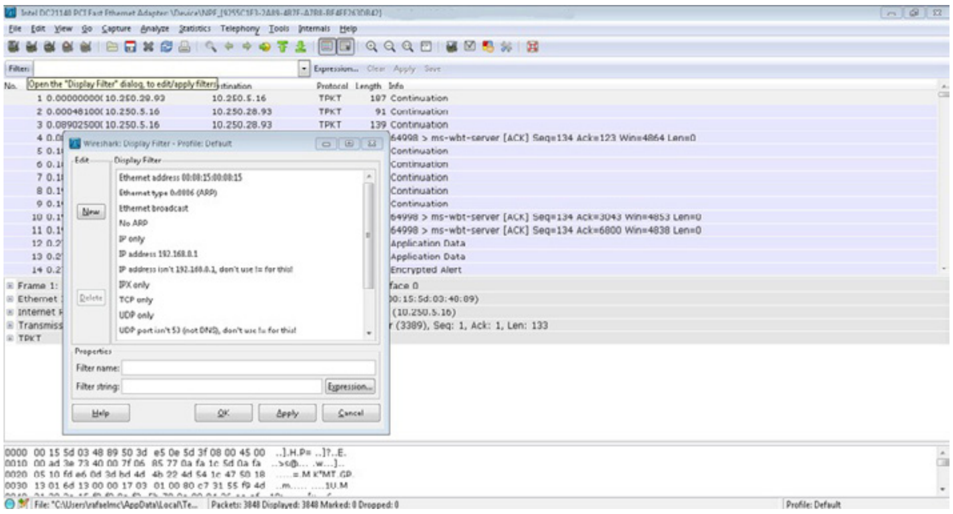
Figure 5. Stop

It is important to remember that you must take care if your network is busy, the data stream may even lock your machine, then it is not advisable to leave the Wireshark capture for a long time, as we will see, we will leave it running only during the process debug a connection. The greater the amount of packets, the longer it takes to apply a filter, find a package, etc.

With this we have the basics of the program, we can set the capture interface, start and stop the capture. The next step is to identify what interests among many packages. For this, we will start using filters.

Using Filters

There is a plethora of possible filters, but at this moment we will see just how to filter by IP address, port and protocol. The filters can be constructed by clicking on “Filter”, then selecting the desired filter (there is a short list of pre-defined filters), or by typing directly into the text box. After you create your filter, just click “Apply”, if you wanted to see the entire list of packages again just click “Clear”, this will remove the filter previously applied.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Open the "Display Filter" dialog, to edit/apply filters
Wireshark: Display Filter - Profile: Default
Edit
Display Filter
New
Delete
Properties
Filter name:
Filter string:
Expression...
Help
OK
Apply
Cancel

```
Ethernet address 00:08:15:00:08:15
Ethernet type 0x0806 (ARP)
Ethernet broadcast
No ARP
IP only
IP address 192.168.0.1
IP address isn't 192.168.0.1, don't use != for this!
IPX only
TCP only
UDP only
UDP port isn't 53 (not DNS), don't use != for this!
```

```
No.  ...                  Destination    Protocol Length Info
  1 0.00000000( 10.250.28.93  10.250.5.16    TPKT    107 Continuation
  2 0.00048100( 10.250.5.16   10.250.28.93   TPKT     91 Continuation
  3 0.08902500( 10.250.5.16   10.250.28.93   TPKT    139 Continuation
  4 0.0                        64998 > ms-wbt-server [ACK] Seq=134 Ack=123 Win=4864 Len=0
  5 0.1                        Continuation
  6 0.1                        Continuation
  7 0.1                        Continuation
  8 0.1                        Continuation
  9 0.1                        Continuation
 10 0.1                        64998 > ms-wbt-server [ACK] Seq=134 Ack=3043 Win=4853 Len=0
 11 0.1                        64998 > ms-wbt-server [ACK] Seq=134 Ack=6800 Win=4838 Len=0
 12 0.2                        Application Data
 13 0.2                        Application Data
 14 0.2                        Encrypted Alert
```

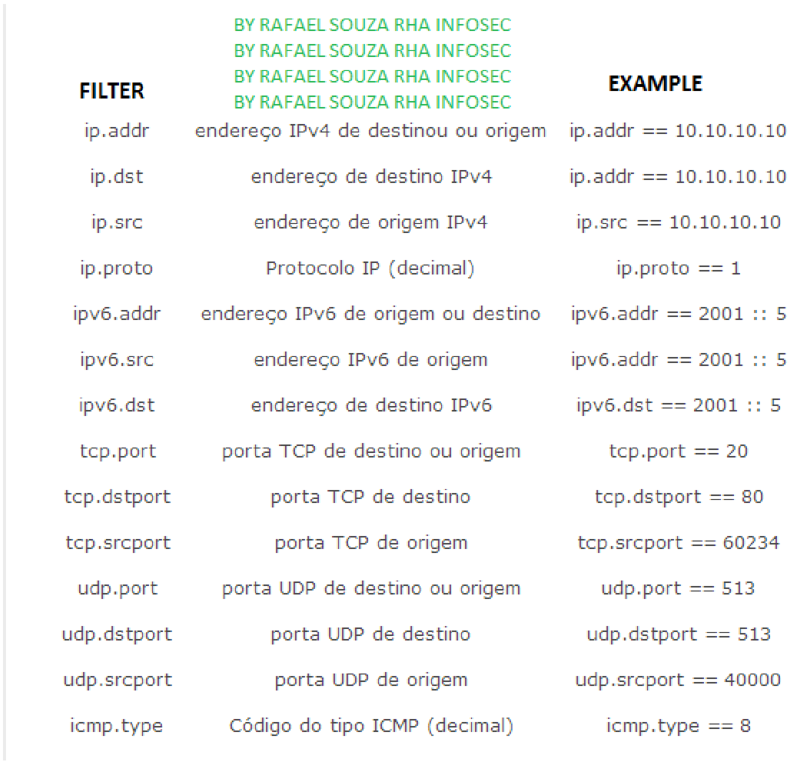
```
Frame 1:                       face 0
Ethernet                       00:15:5d:03:48:89)
Internet P                     (10.250.5.16)
Transmiss                      r (3389), Seq: 1, Ack: 1, Len: 133
TPKT
```

```
0000  00 15 5d 03 48 89 50 3d  e5 0e 5d 3f 08 00 45 00   ..].H.P= ..]?..E.
0010  00 ad 3e 73 40 00 7f 06  85 77 0a fa 1c 5d 0a fa   ..>s@... .w...]..
0020  05 10 fd e6 0d 3d bd 4d  4b 22 4d 54 1c 47 50 18   .....=.M K"MT.GP.
0030  13 01 6d 13 00 00 17 03  01 00 80 c7 31 55 f9 4d   ..m..... ....1U.M
```

```
File: "C:\Users\rafaelmc\AppData\Local\Te...   Packets: 3848 Displayed: 3848 Marked: 0 Dropped: 0   Profile: Default
```

Figure 6. Filter

I will use a small filter list as an example:



FILTER	BY RAFAEL SOUZA RHA INFOSEC BY RAFAEL SOUZA RHA INFOSEC BY RAFAEL SOUZA RHA INFOSEC BY RAFAEL SOUZA RHA INFOSEC	EXAMPLE
ip.addr	endereço IPv4 de destinou ou origem	ip.addr == 10.10.10.10
ip.dst	endereço de destino IPv4	ip.addr == 10.10.10.10
ip.src	endereço de origem IPv4	ip.src == 10.10.10.10
ip.proto	Protocolo IP (decimal)	ip.proto == 1
ipv6.addr	endereço IPv6 de origem ou destino	ipv6.addr == 2001 :: 5
ipv6.src	endereço IPv6 de origem	ipv6.addr == 2001 :: 5
ipv6.dst	endereço de destino IPv6	ipv6.dst == 2001 :: 5
tcp.port	porta TCP de destino ou origem	tcp.port == 20
tcp.dstport	porta TCP de destino	tcp.dstport == 80
tcp.srcport	porta TCP de origem	tcp.srcport == 60234
udp.port	porta UDP de destino ou origem	udp.port == 513
udp.dstport	porta UDP de destino	udp.dstport == 513
udp.srcport	porta UDP de origem	udp.srcport == 40000
icmp.type	Código do tipo ICMP (decimal)	icmp.type == 8

Figure 7. Example by Rafael Souza (RHA Infosec)

It is also possible to group the filters, for example:

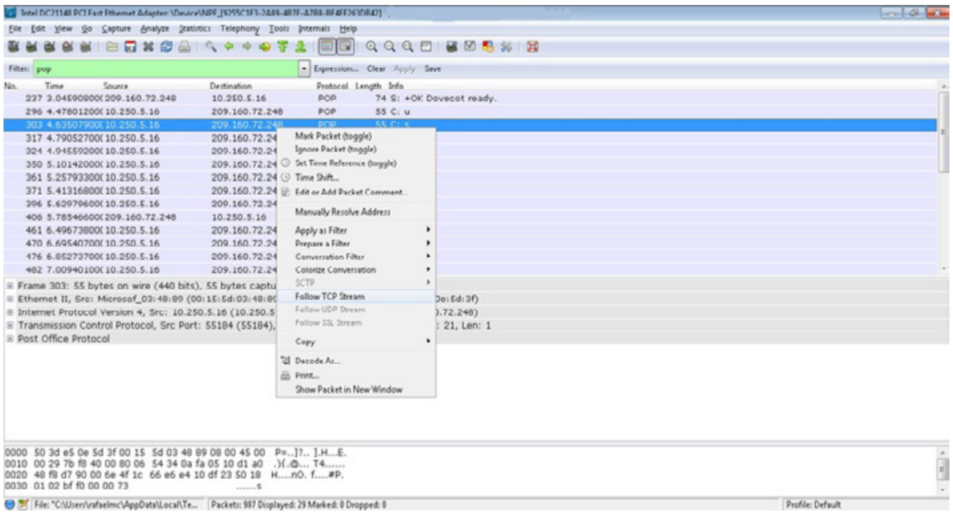
```
ip.src == 10.10.10.1 && tcp.dstport==80 OR ip.src == 10.10.10.1 and tcp.dstport==80

Source address 10.10.10.1
And destination port 80
```

CAPTURING PASSWORDS

Now we will see how you can capture passwords easily, just by listening to traffic. For this example we will use the POP3 protocol, which sends the data in clear text over the network. To do this, start capturing packets normally and start a session with your server pop3 email. If you use a safer as imaps or pop3s and I just wanted to see the functioning of the mechanism, protocol is possible to connect via telnet pop3 without having to add / modify your account, simply run the following:

```
telnet serveremail.com 110
user user@rhainfosec.com
pass rhainfosecpasswd
```

Now stop the capture, filter and put “pop” and then click “Apply”. That done, you see only the packets of pop3 connection. Now click on any of them right, and then click “Follow TCP Stream”.

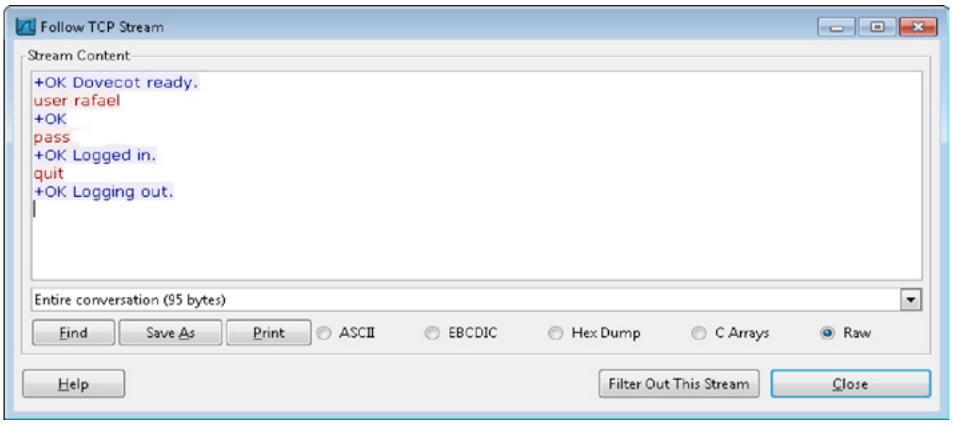


Filter: pop

```
No.   Time         Source           Destination      Protocol  Length  Info
 237 2.04590000( 209.160.72.248   10.250.5.16       POP        74 S: +OK Dovecot ready.
 296 4.47801200( 10.250.5.16      209.160.72.248    POP        55 C: u
 303 4.63507900( 10.250.5.16      209.160.72.248    POP        55 C: s
 317 4.79052700( 10.250.5.16      209.160.72.24
 324 4.94559200( 10.250.5.16      209.160.72.24
 350 5.10142000( 10.250.5.16      209.160.72.24
 361 5.25793300( 10.250.5.16      209.160.72.24
 371 5.41316800( 10.250.5.16      209.160.72.24
 396 5.62979600( 10.250.5.16      209.160.72.24
 406 5.78546600( 209.160.72.248   10.250.5.16
 461 6.49673800( 10.250.5.16      209.160.72.24
 470 6.69540700( 10.250.5.16      209.160.72.24
 476 6.85273700( 10.250.5.16      209.160.72.24
 482 7.00940100( 10.250.5.16      209.160.72.24

⊞ Frame 303: 55 bytes on wire (440 bits), 55 bytes captu
⊞ Ethernet II, Src: Microsof_03:48:89 (00:15:5d:03:48:89                 0e:5d:3f)
⊞ Internet Protocol Version 4, Src: 10.250.5.16 (10.250.5                 .72.248)
⊞ Transmission Control Protocol, Src Port: 55184 (55184),                 : 21, Len: 1
⊞ Post Office Protocol

0000  50 3d e5 0e 5d 3f 00 15  5d 03 48 89 08 00 45 00   P=..]?.. ].H...E.
0010  00 29 7b f8 40 00 80 06  54 34 0a fa 05 10 d1 a0   .){.@... T4......
0020  48 f8 d7 90 00 6e 4f 1c  66 e6 e4 10 df 23 50 18   H....nO. f....#P.
0030  01 02 bf f0 00 00 73                               ......s

File: "C:\Users\rafaelmc\AppData\Local\Te...  Packets: 987 Displayed: 29 Marked: 0 Dropped: 0
```

Figure 8. POP3

With this he will open a new window with the entire contents of the ASCII connection. As the pop3 protocol sends everything in plain text, you can see all the commands executed, including the password.



```
+OK Dovecot ready.
user rafael
+OK
pass
+OK Logged in.
quit
+OK Logging out.
```

Entire conversation (95 bytes)

ASCII EBCDIC Hex Dump C Arrays [✓ Raw]

Figure 9. Pass

This can be transported to any connection in plain text, such as ftp, telnet, http, etc.. Just to let you change the filter and examine the contents of the connection.

Importing External Captures

Usually in servers, there is no graphical environment installed and with that you cannot use Wireshark directly. If you want to analyze traffic on this server and you cannot install Wireshark, or if you do not have to capture this traffic elsewhere, the best one can do is write traffic with tcpdump locally and then copy this dump to a machine with Wireshark for a more detailed analysis is made.

We will capture everything that comes or goes to the host 10.10.10.1 with destination port 80 and save content in `capturerafaelsouzarhainfosec.pcap` file from the local folder where the command was executed. Run the server:

```
tcpdump -i eth0 host 10.10.10.1 and dst port 80 -w capturerafaelsouzarhainfosec.pcap
```

Once you're finished capturing, simply use CTRL + C to copy the file to the machine Wireshark capture and import by clicking on File -> Import. Once imported, you can use the program normally as if the capture had occurred locally.

EVOLUTION OF THINKING

Why steal your password?

There are various reasons that lead people to steal passwords from simply to annoy someone (sending email as you) up to perform illegal activities (invasion on other computers, theft of information, etc.) An attractive to crackers is the ability to use the identity of others in these activities.

One of the main reasons that attackers try to break systems and install "sniffers" is able to quickly capture the maximum possible accounts. Thus, the more accounts this attacker has, the easier it is to hide your stash.

How can you protect yourself?

Do not be thinking that "sniffers" can make all the insecure Internet. Not so. You need to be aware of where the risk is, when you're at risk and what to do to be safe.

When you have your stolen credit card or suspect that someone may be using it improperly, you cancel the card and asks another. Likewise, as passwords can be stolen, it's critical that you replace regularly.
This precaution limited the amount of time that a stolen password can be used by an attacker.

Never share your password with others. This sharing makes it difficult to know where your password is being used (exposed) and is harder to detect unauthorized use.

Never give your password to anyone claiming access your account needs to fix some problem or want to investigate a breach of the system. This trick is one of the most effective methods of hacking, known as "social engineering."

Use networks you can trust

Another aspect you should take into consideration is what network you can trust and which cannot. If you're traveling and need to access their computers remotely organization. For example, pick any file in your home directory and you have available is a "LanHouse" or network of another organization. Are you sure you can trust the network?

If you have no alternative for secure remote access and only have available resources such as telnet, for example, you can "mitigate" this effect by changing the password at the end of each session. Remember that only the first packet (200-300 bytes)of each session carry information from your "login".

Therefore, to always change your password before logging out, this will not be captured and password before it was exposed to the network is no longer valid. Of course it is possible to capture everything going across the network, but has no intention of attacking fill the file system quickly and so easily discovered.

Why networks remain vulnerable to "sniffers" long?

There are several reasons and there is no quick solution to the problem.

Part of the problem is that companies tend to invest in more new features than add security. New security features can leave the most difficult systems to configure and less convenient to use.

Another part of the problem is related to added costs for Ethernet switches, hubs, network interfaces that do not support the particular "promiscuous" that sniffers can use.

CONCLUSION

The question that remains is how can we protect ourselves from this threat...

- Network cards that cannot be put into "promiscuous" mode. Thus, computers cannot be mastered and transformed into "sniffers".
- Typically, the Ethernet interface only passes packets to the highest level protocol that are intended for local machine. This interface into promiscuous mode allows all packets are accepted and passed to the higher layer of the protocol stack. This allows the selection you want.
- Packages that encrypt data in transit over the network, thus avoiding to flow passwords "in the clear".

I would remind you that the safest is to adopt and encourage the use of software which enable remote access encrypted sessions, help much to make your environment more secure.

One fairly common encryption technology currently in secure communication between remote machines SSH (Secure Shell). SSH is available for different platforms. Its use does not prevent the password captured, but as this is not encrypted serve to the attacker. SSH negotiates connections using RSA algorithm. Once the service is authenticated, all subsequent traffic is encrypted using IDEA technology. This type of encryption is very strong.

In the future, security will increasingly intrinsic to the systems and infrastructure networks. No use having all the "apparatus" of security you need, but do not use them. Security is not something that can be completely secure. Remember, no one is 100% secure.

About the Author

Over the years, acquiring knowledge of Webmaster programmer (HTML5, CSS, XML, ActionScript), developer in languages like Python, Shell Script, Perl, Pascal, Ruby, Object Pascal, C and Java. I started studying with thirteen (SQL database), i have extensive experience in operating systems such as Linux, UNIX, and Windows. I am maintainer of the "project backtrack team brazilian", I am also a member of the "French Backtrack Team" and made partnerships with groups from Indonesia and Algeria; prepared a collection of video lessons and made them available on the website. I have good communication in groups and the general public, attended college projects with a focus on business organization, I am currently seeking for a work experience outside of Brazil".
http://sourceforge.net/projects/cypherpunks/

Thank you readers, I hope I have contributed in knowledge of you... I just want to remind that this article was written for educational purposes and for security experts, I want to make clear that password stealing is a crime, and I hope you also have to apply a coincidence methods encrypted connection to protect themselves.

UPDATE
NOW WITH
STIG
AUDITING

"IN SOME CASES
nipper studio
HAS VIRTUALLY
REMOVED
the NEED FOR a
MANUAL AUDIT"

CISCO SYSTEMS INC.

Titania's award winning Nipper Studio configuration auditing tool is helping security consultants and end-user organizations worldwide improve their network security. Its reports are more detailed than those typically produced by scanners, enabling you to maintain a higher level of vulnerability analysis in the intervals between penetration tests.

Now used in over 45 countries, Nipper Studio provides a thorough, fast & cost effective way to securely audit over 100 different types of network device. The NSA, FBI, DoD & U.S. Treasury already use it, so why not try it for free at www.titania.com

2012 Computing Security Awards
WINNER
Enterprise Security Solution of the Year

Security Products' GOVIES
2013 Government Security Awards

2012 Computing Security Awards
WINNER
Network Security Solution of the Year

www.titania.com

Detect/Analyze Scanning Traffic Using Wireshark

by Santosh Kumar

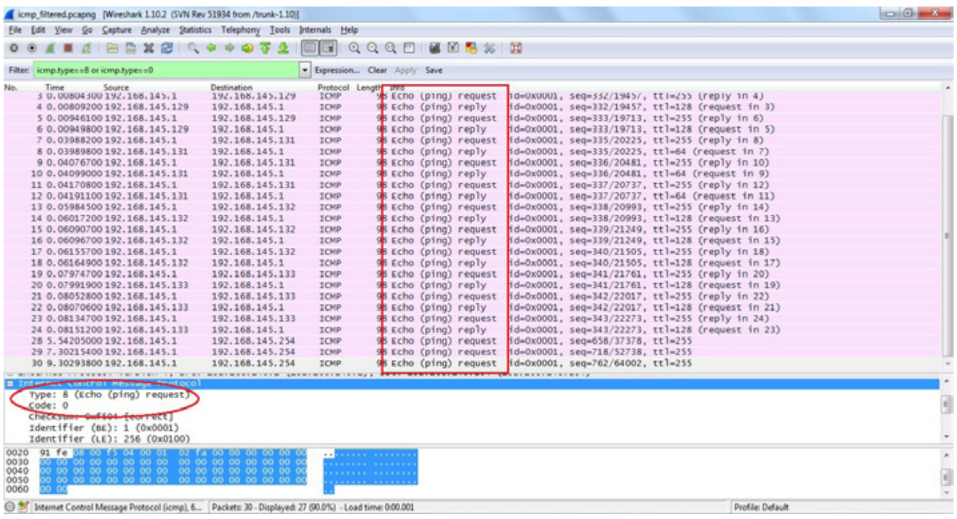
"Wireshark", the world's most popular Network Protocol Analyzer is a multipurpose tool. It can be used as a Packet Sniffer, Network Analyser, Protocol Analyser & Forensic tool. Through this article my focus is on how to use Wireshark to detect/analyze any scanning & suspect traffic.

Let's start with Scanning first. As a thief studies surroundings before stealing something from a target, similarly attackers or hackers also perform foot printing and scanning before the actual attack. In this phase, they want to collect all possible information about the target so that they can plan their attack accordingly. If we talk about scanning here they want to collect details like:

- Which IP addresses are in use?
- Which port/services are active on those IPs?
- Which platform (Operating System) is in use?
- What are the vulnerabilities & other similar kinds of information.
- Now I am moving to some popular scan methods and how to detect them in Wireshark.

Ping Sweep

This scan is helpful to find out which IPs are active in the network. Ping Sweep can be performed using ICMP, TCP or UDP, the most popular one is ICMP Ping Sweep. In this ICMP type 8, ECHO request is followed by ICMP type 0, ECHO reply packets are being used while in TCP/UDP ping sweep packets are destined to TCP/UDP port 7, The ECHO port. If that target host doesn't support ECHO service then this TCP/UDP ping sweep will not work. Thus ICMP ping sweep is mostly used, but if there is a firewall in between which is configured to block ICMP packet then even ICMP ping sweep is useless. In this situation, ARP scan/ARP sweep can be used which is discussed next (Figure 1).


icmp_filtered.pcapng [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: icmp.type==8 or icmp.type==0
Expression... Clear Apply Save

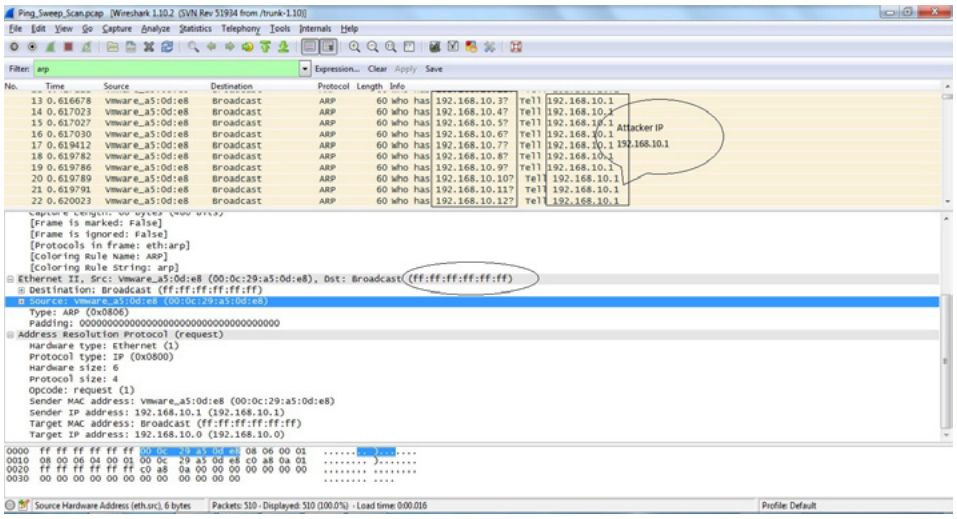
```
No.   Time        Source           Destination      Protocol Length Info
 3 0.00804300 192.168.145.1   192.168.145.129 ICMP   98 Echo (ping) request  id=0x0001, seq=332/19457, ttl=255 (reply in 4)
 4 0.00809200 192.168.145.129 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=332/19457, ttl=128 (request in 3)
 5 0.00946100 192.168.145.1   192.168.145.129 ICMP   98 Echo (ping) request  id=0x0001, seq=333/19713, ttl=255 (reply in 6)
 6 0.00949800 192.168.145.129 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=333/19713, ttl=128 (request in 5)
 7 0.03988200 192.168.145.1   192.168.145.131 ICMP   98 Echo (ping) request  id=0x0001, seq=335/20225, ttl=255 (reply in 8)
 8 0.03989800 192.168.145.131 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=335/20225, ttl=64 (request in 7)
 9 0.04076700 192.168.145.1   192.168.145.131 ICMP   98 Echo (ping) request  id=0x0001, seq=336/20481, ttl=255 (reply in 10)
10 0.04099000 192.168.145.131 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=336/20481, ttl=64 (request in 9)
11 0.04170800 192.168.145.1   192.168.145.131 ICMP   98 Echo (ping) request  id=0x0001, seq=337/20737, ttl=255 (reply in 12)
12 0.04191100 192.168.145.131 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=337/20737, ttl=64 (request in 11)
13 0.05984500 192.168.145.1   192.168.145.132 ICMP   98 Echo (ping) request  id=0x0001, seq=338/20993, ttl=255 (reply in 14)
14 0.06017200 192.168.145.132 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=338/20993, ttl=128 (request in 13)
15 0.06090700 192.168.145.1   192.168.145.132 ICMP   98 Echo (ping) request  id=0x0001, seq=339/21249, ttl=255 (reply in 16)
16 0.06096700 192.168.145.132 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=339/21249, ttl=128 (request in 15)
17 0.06155700 192.168.145.1   192.168.145.132 ICMP   98 Echo (ping) request  id=0x0001, seq=340/21505, ttl=255 (reply in 18)
18 0.06164900 192.168.145.132 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=340/21505, ttl=128 (request in 17)
19 0.07974700 192.168.145.1   192.168.145.133 ICMP   98 Echo (ping) request  id=0x0001, seq=341/21761, ttl=255 (reply in 20)
20 0.07991900 192.168.145.133 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=341/21761, ttl=128 (request in 19)
21 0.08052800 192.168.145.1   192.168.145.133 ICMP   98 Echo (ping) request  id=0x0001, seq=342/22017, ttl=255 (reply in 22)
22 0.08070600 192.168.145.133 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=342/22017, ttl=128 (request in 21)
23 0.08134700 192.168.145.1   192.168.145.133 ICMP   98 Echo (ping) request  id=0x0001, seq=343/22273, ttl=255 (reply in 24)
24 0.08151200 192.168.145.133 192.168.145.1   ICMP   98 Echo (ping) reply    id=0x0001, seq=343/22273, ttl=128 (request in 23)
28 5.54205000 192.168.145.1   192.168.145.254 ICMP   98 Echo (ping) request  id=0x0001, seq=658/37378, ttl=255
29 7.30215400 192.168.145.1   192.168.145.254 ICMP   98 Echo (ping) request  id=0x0001, seq=718/52738, ttl=255
30 9.30293800 192.168.145.1   192.168.145.254 ICMP   98 Echo (ping) request  id=0x0001, seq=762/64002, ttl=255

⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf504 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)

0020  91 fe 08 00 f5 04 00 01  02 fa 00 00 00 00 00 00   ..........
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
0060  00 00                                              ..

Internet Control Message Protocol (icmp), 6...  Packets: 30 · Displayed: 27 (90.0%) · Load time: 0:00.001   Profile: Default
```

Figure 1. Ping Sweep

To detect ICMP ping sweep in Wireshark apply simple filter `icmp.type==8 or icmp.type==0`. TCP ping sweep can be detected with `tcp.dstport==7` filter and for UDP ping sweep `udp.dstport==7` filter can be used. After applying these filters if we are getting more than expected packets then it's possible that ping sweep is going on in our network. We need to be careful about the volume of such traffic as it might be normal ping traffic. It should be considered as a scan signature only if you are getting unexpected increase in ICMP traffic.

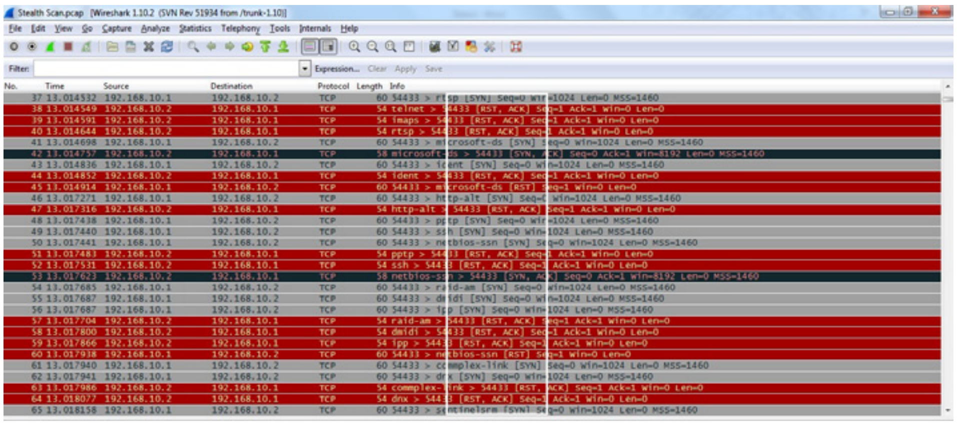
ARP Sweep/ARP Scan

As discussed in previous scan that if a firewall is implemented in between and ICMP is blocked then we can't use ICMP ping sweep. In such a situation, ARP scan is helpful to find out active IPs in the network. Here, attacker sends ARP broadcast (for broadcast, destination MAC will be `0xff:ff:ff:ff:ff:ff`) for each and every possible IP in selected subnet and if he gets ARP response then it shows that IP is active. Advantage of this scan is that ARP communication can't be filtered or disabled because all TCP/IP communication is based on it. Blocking or disabling ARP communication will break TCP/IP communication or it will force static ARP entries and disadvantage of this scan is that it can't cross layer 3 Devices. This scan can be easily detected with filter ARP. After applying this filter if we are getting unexpected no. of ARP queries as shown in the picture, it is a sign for ARP scan or ARP sweep (FIgure 2).


Ping_Sweep_Scan.pcap [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: arp
Expression... Clear Apply Save
Attacker IP 192.168.10.1
Profile: Default

```
No.   Time       Source             Destination  Protocol Length Info
   13 0.616678   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.3?  Tell 192.168.10.1
   14 0.617023   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.4?  Tell 192.168.10.1
   15 0.617027   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.5?  Tell 192.168.10.1
   16 0.617030   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.6?  Tell 192.168.10.1
   17 0.619412   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.7?  Tell 192.168.10.1
   18 0.619782   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.8?  Tell 192.168.10.1
   19 0.619786   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.9?  Tell 192.168.10.1
   20 0.619789   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.10? Tell 192.168.10.1
   21 0.619791   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.11? Tell 192.168.10.1
   22 0.620023   Vmware_a5:0d:e8    Broadcast    ARP      60 Who has 192.168.10.12? Tell 192.168.10.1

    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:arp]
    [Coloring Rule Name: ARP]
    [Coloring Rule String: arp]
⊟ Ethernet II, Src: Vmware_a5:0d:e8 (00:0c:29:a5:0d:e8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊞ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊞ Source: Vmware_a5:0d:e8 (00:0c:29:a5:0d:e8)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Vmware_a5:0d:e8 (00:0c:29:a5:0d:e8)
    Sender IP address: 192.168.10.1 (192.168.10.1)
    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
    Target IP address: 192.168.10.0 (192.168.10.0)

0000  ff ff ff ff ff ff 00 0c  29 a5 0d e8 08 06 00 01   ........ )......
0010  08 00 06 04 00 01 00 0c  29 a5 0d e8 c0 a8 0a 01   ........ ).......
0020  ff ff ff ff ff ff c0 a8  0a 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

Source Hardware Address (eth.src), 6 bytes | Packets: 510 · Displayed: 510 (100.0%) · Load time: 0:00.016
```

Figure 2. ARP Scan/ARP sweep

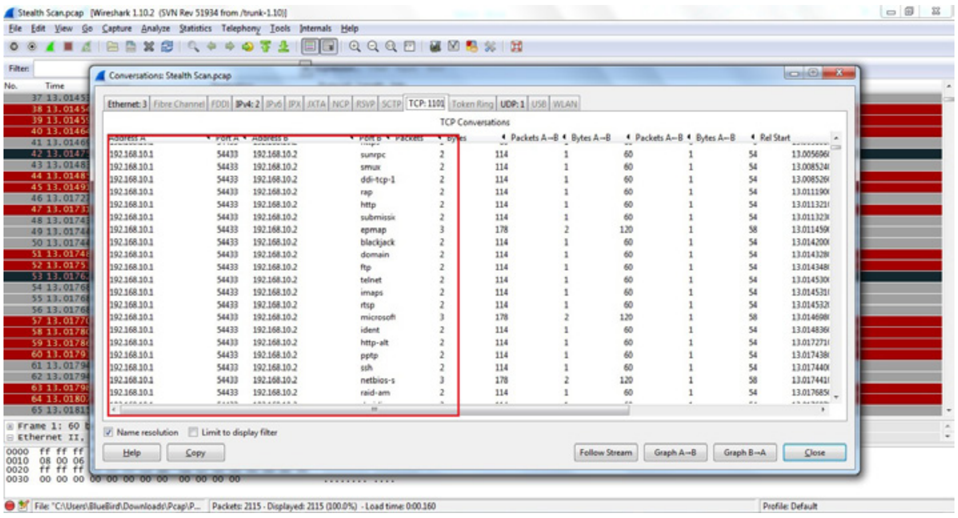

Stealth Scan.pcap [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time        Source        Destination   Protocol Length Info
   37 13.014532   192.168.10.1  192.168.10.2  TCP      60 54433 > rtsp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   38 13.014549   192.168.10.2  192.168.10.1  TCP      54 telnet > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   39 13.014591   192.168.10.2  192.168.10.1  TCP      54 imaps > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   40 13.014644   192.168.10.2  192.168.10.1  TCP      54 rtsp > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   41 13.014698   192.168.10.1  192.168.10.2  TCP      60 54433 > microsoft-ds [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   42 13.014757   192.168.10.2  192.168.10.1  TCP      58 microsoft-ds > 54433 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
   43 13.014836   192.168.10.1  192.168.10.2  TCP      60 54433 > ident [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   44 13.014852   192.168.10.2  192.168.10.1  TCP      54 ident > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   45 13.014914   192.168.10.1  192.168.10.2  TCP      60 54433 > microsoft-ds [RST] Seq=1 Win=0 Len=0
   46 13.017271   192.168.10.1  192.168.10.2  TCP      60 54433 > http-alt [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   47 13.017316   192.168.10.2  192.168.10.1  TCP      54 http-alt > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   48 13.017438   192.168.10.1  192.168.10.2  TCP      60 54433 > pptp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   49 13.017440   192.168.10.1  192.168.10.2  TCP      60 54433 > ssh [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   50 13.017441   192.168.10.1  192.168.10.2  TCP      60 54433 > netbios-ssn [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   51 13.017483   192.168.10.2  192.168.10.1  TCP      54 pptp > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   52 13.017531   192.168.10.2  192.168.10.1  TCP      54 ssh > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   53 13.017623   192.168.10.2  192.168.10.1  TCP      58 netbios-ssn > 54433 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
   54 13.017685   192.168.10.1  192.168.10.2  TCP      60 54433 > raid-am [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   55 13.017687   192.168.10.1  192.168.10.2  TCP      60 54433 > dmidi [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   56 13.017687   192.168.10.1  192.168.10.2  TCP      60 54433 > ipp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   57 13.017704   192.168.10.2  192.168.10.1  TCP      54 raid-am > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   58 13.017800   192.168.10.2  192.168.10.1  TCP      54 dmidi > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   59 13.017866   192.168.10.2  192.168.10.1  TCP      54 ipp > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   60 13.017938   192.168.10.1  192.168.10.2  TCP      60 54433 > netbios-ssn [RST] Seq=1 Win=0 Len=0
   61 13.017940   192.168.10.1  192.168.10.2  TCP      60 54433 > commplex-link [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   62 13.017941   192.168.10.1  192.168.10.2  TCP      60 54433 > dnx [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   63 13.017986   192.168.10.2  192.168.10.1  TCP      54 commplex-link > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   64 13.018077   192.168.10.2  192.168.10.1  TCP      54 dnx > 54433 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   65 13.018158   192.168.10.1  192.168.10.2  TCP      60 54433 > sentinelsrm [SYN] Seq=0 Win=1024 Len=0 MSS=1460
```

Figure 3. Stealth Scan

TCP Half Open/Stealth Scan

To detect open or close TCP port on target system, Stealth scan is the most often used method. In this scan, attacker sends a SYN packet on the target port like a normal TCP communication. If the port is open, he will get SYN+ACK and RST or RST+ACK if the port is closed. After getting SYN+ACK on the open port as a response, attacker will send RST because attacker doesn't want to open TCP session with a target. If that target port is firewalled then expected response is ICMP type 3 Packet with Code 1,2,3,9,10, or 13. So in Wireshark if we are getting a lot of RST packets or ICMP type 3 packets, it can be a sign for Stealth Scan or TCP Full Connect Scan. As we can see that in the above picture a lot of SYN & RST packets are moving back and forth, but there is no data communication between these hosts. To get a quick view in the above capture we can go to top menu Statistics -> Conversations and then go to TCP tab. There we can see multiple TCP sessions, but all are having less than 4 packet communications which is a sign for TCP port Scan (Figure 4).


Conversations: Stealth Scan.pcap

TCP Conversations

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B	Rel Start
192.168.10.1	54433	192.168.10.2	sunrpc	2	114	1	60	1	54	13.005696
192.168.10.1	54433	192.168.10.2	smux	2	114	1	60	1	54	13.008524
192.168.10.1	54433	192.168.10.2	ddi-tcp-1	2	114	1	60	1	54	13.008526
192.168.10.1	54433	192.168.10.2	rap	2	114	1	60	1	54	13.011190
192.168.10.1	54433	192.168.10.2	http	2	114	1	60	1	54	13.011321
192.168.10.1	54433	192.168.10.2	submissi	2	114	1	60	1	54	13.011323
192.168.10.1	54433	192.168.10.2	epmap	3	178	2	120	1	58	13.011459
192.168.10.1	54433	192.168.10.2	blackjack	2	114	1	60	1	54	13.014200
192.168.10.1	54433	192.168.10.2	domain	2	114	1	60	1	54	13.014328
192.168.10.1	54433	192.168.10.2	ftp	2	114	1	60	1	54	13.014348
192.168.10.1	54433	192.168.10.2	telnet	2	114	1	60	1	54	13.014530
192.168.10.1	54433	192.168.10.2	imaps	2	114	1	60	1	54	13.014531
192.168.10.1	54433	192.168.10.2	rtsp	2	114	1	60	1	54	13.014532
192.168.10.1	54433	192.168.10.2	microsoft	3	178	2	120	1	58	13.014698
192.168.10.1	54433	192.168.10.2	ident	2	114	1	60	1	54	13.014836
192.168.10.1	54433	192.168.10.2	http-alt	2	114	1	60	1	54	13.017271
192.168.10.1	54433	192.168.10.2	pptp	2	114	1	60	1	54	13.017438
192.168.10.1	54433	192.168.10.2	ssh	2	114	1	60	1	54	13.017440
192.168.10.1	54433	192.168.10.2	netbios-s	3	178	2	120	1	58	13.017441
192.168.10.1	54433	192.168.10.2	raid-am	2	114	1	60	1	54	13.017685

Packets: 2115 · Displayed: 2115 (100.0%) · Load time: 0:00.160

Figure 4. Statistics->Conversation_ TCP tab

TCP Full Connect Scan

In this scan attacker is going to perform complete three way hand shake to find out if the port is open or close. Attacker will send SYN on target port, if the port opens, he will get SYN+ACK and RST+ACK on the closed port. After getting SYN+ACK, the attacker will send ACK and try to establish TCP session and then terminate it. In Wireshark, we can use a similar method like TCP Half open scan to detect TCP full connect as well. If that target port is firewalled then here also we will get the same response which is ICMP type 3 Packet with Code 1,2,3,9,10, or 13. Following filters can be used in Wireshark to detect TCP scan packet quickly (TCP Half open & TCP Full Connect)

- To get SYN, SYN+ACK, RST & RST+ACK packet

```
tcp.flags==0x002 or tcp.flags==0x012 or tcp.flags==0x004 or tcp.flags==0x014
```

- To get ICMP type 3 Packet with Code 1,2,3,9,10, or 13 Packet

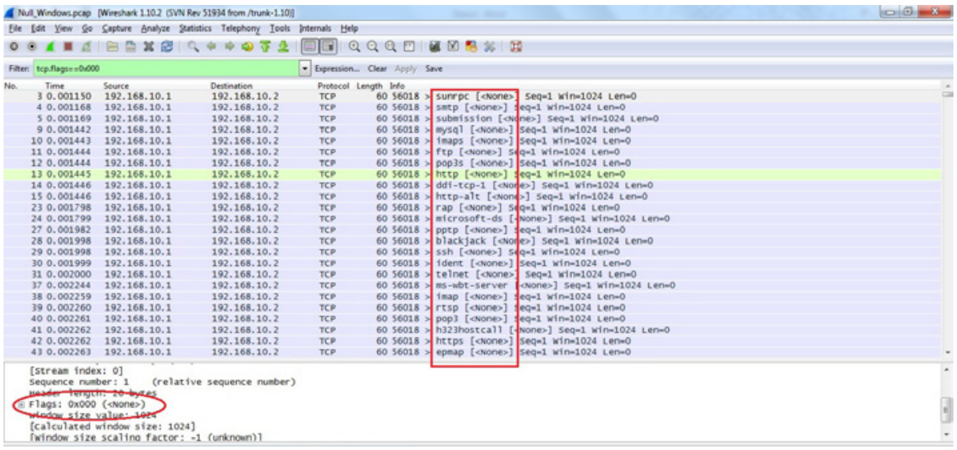
```
icmp.type==3 and (icmp.code==1 or icmp.code==2 or icmp.code==3 or icmp.code==9 or icmp.code==10
or icmp.code==13)
```

- To get SYN, SYN+ACK, RST & RST+ACK packet along with ICMP type 3 Packet with Code 1,2,3,9,10, or 13 Packet

```
tcp.flags==0x002 or tcp.flags==0x012 or tcp.flags==0x004 or tcp.flags==0x014 or (icmp.type==3 and
(icmp.code==1 or icmp.code==2 or icmp.code==3 or icmp.code==9 or icmp.code==10 or icmp.code==13))
```

Null Scan

In this scan attacker sends a TCP packet without setting any flag on it and as a response if he is getting RST packet it means the port is closed. There will be no response to null scan if the port is open or filtered and if he is getting ICMP Type 3 Code 1,2,3,9,10 or 13 packet then *port seems to be firewalled.To detect Null Scan in Wireshark, we can use a simple filter* `TCP.flags==0x000`. It will filter all TCP packets moving without Flag (Figure 5).

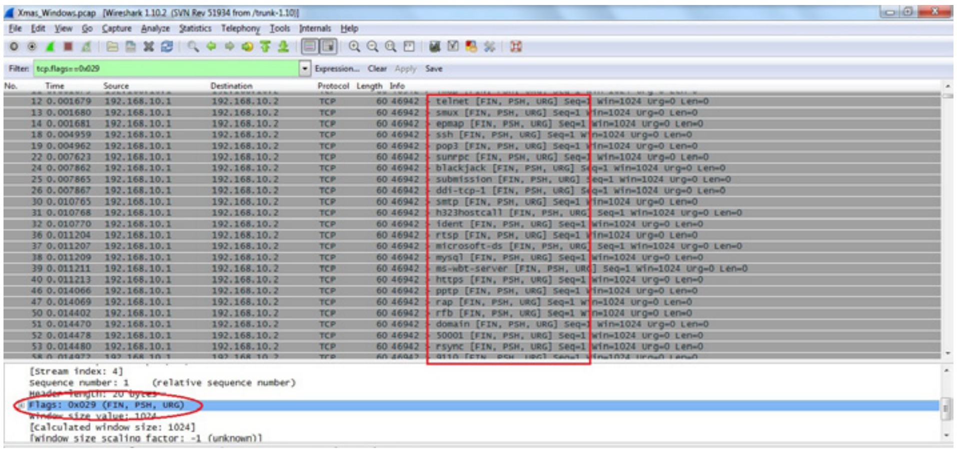


```
Null_Windows.pcap  [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
Filter: tcp.flags==0x000
No.   Time       Source         Destination    Protocol Length Info
  3 0.001150   192.168.10.1   192.168.10.2   TCP      60 56018 > sunrpc [<None>] Seq=1 Win=1024 Len=0
  4 0.001168   192.168.10.1   192.168.10.2   TCP      60 56018 > smtp [<None>] Seq=1 Win=1024 Len=0
  5 0.001169   192.168.10.1   192.168.10.2   TCP      60 56018 > submission [<None>] Seq=1 Win=1024 Len=0
  9 0.001442   192.168.10.1   192.168.10.2   TCP      60 56018 > mysql [<None>] Seq=1 Win=1024 Len=0
 10 0.001443   192.168.10.1   192.168.10.2   TCP      60 56018 > imaps [<None>] Seq=1 Win=1024 Len=0
 11 0.001444   192.168.10.1   192.168.10.2   TCP      60 56018 > ftp [<None>] Seq=1 Win=1024 Len=0
 12 0.001444   192.168.10.1   192.168.10.2   TCP      60 56018 > pop3s [<None>] Seq=1 Win=1024 Len=0
 13 0.001445   192.168.10.1   192.168.10.2   TCP      60 56018 > http [<None>] Seq=1 Win=1024 Len=0
 14 0.001446   192.168.10.1   192.168.10.2   TCP      60 56018 > ddi-tcp-1 [<None>] Seq=1 Win=1024 Len=0
 15 0.001446   192.168.10.1   192.168.10.2   TCP      60 56018 > http-alt [<None>] Seq=1 Win=1024 Len=0
 23 0.001798   192.168.10.1   192.168.10.2   TCP      60 56018 > rap [<None>] Seq=1 Win=1024 Len=0
 24 0.001799   192.168.10.1   192.168.10.2   TCP      60 56018 > microsoft-ds [<None>] Seq=1 Win=1024 Len=0
 27 0.001982   192.168.10.1   192.168.10.2   TCP      60 56018 > pptp [<None>] Seq=1 Win=1024 Len=0
 28 0.001998   192.168.10.1   192.168.10.2   TCP      60 56018 > blackjack [<None>] Seq=1 Win=1024 Len=0
 29 0.001998   192.168.10.1   192.168.10.2   TCP      60 56018 > ssh [<None>] Seq=1 Win=1024 Len=0
 30 0.001999   192.168.10.1   192.168.10.2   TCP      60 56018 > ident [<None>] Seq=1 Win=1024 Len=0
 31 0.002000   192.168.10.1   192.168.10.2   TCP      60 56018 > telnet [<None>] Seq=1 Win=1024 Len=0
 37 0.002244   192.168.10.1   192.168.10.2   TCP      60 56018 > ms-wbt-server [<None>] Seq=1 Win=1024 Len=0
 38 0.002259   192.168.10.1   192.168.10.2   TCP      60 56018 > imap [<None>] Seq=1 Win=1024 Len=0
 39 0.002260   192.168.10.1   192.168.10.2   TCP      60 56018 > rtsp [<None>] Seq=1 Win=1024 Len=0
 40 0.002261   192.168.10.1   192.168.10.2   TCP      60 56018 > pop3 [<None>] Seq=1 Win=1024 Len=0
 41 0.002262   192.168.10.1   192.168.10.2   TCP      60 56018 > h323hostcall [<None>] Seq=1 Win=1024 Len=0
 42 0.002262   192.168.10.1   192.168.10.2   TCP      60 56018 > https [<None>] Seq=1 Win=1024 Len=0
 43 0.002263   192.168.10.1   192.168.10.2   TCP      60 56018 > epmap [<None>] Seq=1 Win=1024 Len=0

    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    Header length: 20 bytes
  ⊞ Flags: 0x000 (<None>)
    Window size value: 1024
    [Calculated window size: 1024]
    [Window size scaling factor: -1 (unknown)]
```

Figure 5. TCP Null Scan

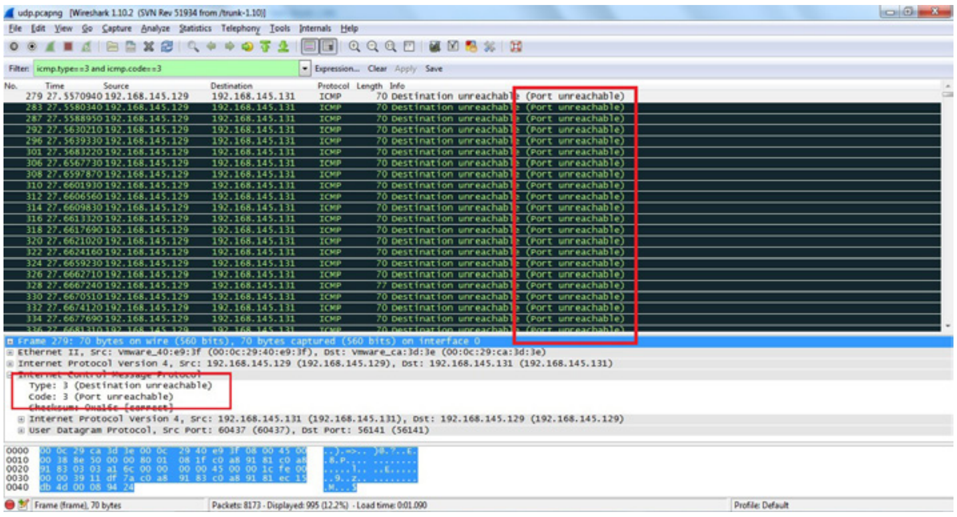
Xmas Scan

Here the attacker sends packet with FIN, PSH & URG TCP flags and response is exactly the same like Null Scan. To detect this type of scan in Wireshark we can use filter "`tcp.flags==0X029` (Figure 6).



```
Xmas_Windows.pcap  [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
Filter: tcp.flags==0x029
No.   Time       Source         Destination    Protocol Length Info
 12 0.001679   192.168.10.1   192.168.10.2   TCP      60 46942 > telnet [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 13 0.001680   192.168.10.1   192.168.10.2   TCP      60 46942 > smux [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 14 0.001681   192.168.10.1   192.168.10.2   TCP      60 46942 > epmap [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 18 0.004959   192.168.10.1   192.168.10.2   TCP      60 46942 > ssh [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 19 0.004962   192.168.10.1   192.168.10.2   TCP      60 46942 > pop3 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 22 0.007623   192.168.10.1   192.168.10.2   TCP      60 46942 > sunrpc [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 24 0.007862   192.168.10.1   192.168.10.2   TCP      60 46942 > blackjack [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 25 0.007865   192.168.10.1   192.168.10.2   TCP      60 46942 > submission [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 26 0.007867   192.168.10.1   192.168.10.2   TCP      60 46942 > ddi-tcp-1 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 30 0.010765   192.168.10.1   192.168.10.2   TCP      60 46942 > smtp [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 31 0.010768   192.168.10.1   192.168.10.2   TCP      60 46942 > h323hostcall [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 32 0.010770   192.168.10.1   192.168.10.2   TCP      60 46942 > ident [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 36 0.011204   192.168.10.1   192.168.10.2   TCP      60 46942 > rtsp [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 37 0.011207   192.168.10.1   192.168.10.2   TCP      60 46942 > microsoft-ds [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 38 0.011209   192.168.10.1   192.168.10.2   TCP      60 46942 > mysql [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 39 0.011211   192.168.10.1   192.168.10.2   TCP      60 46942 > ms-wbt-server [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 40 0.011213   192.168.10.1   192.168.10.2   TCP      60 46942 > https [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 46 0.014066   192.168.10.1   192.168.10.2   TCP      60 46942 > pptp [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 47 0.014069   192.168.10.1   192.168.10.2   TCP      60 46942 > rap [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 50 0.014402   192.168.10.1   192.168.10.2   TCP      60 46942 > rfb [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 51 0.014470   192.168.10.1   192.168.10.2   TCP      60 46942 > domain [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 52 0.014478   192.168.10.1   192.168.10.2   TCP      60 46942 > 50001 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 53 0.014480   192.168.10.1   192.168.10.2   TCP      60 46942 > rsync [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
 58 0.014972   192.168.10.1   192.168.10.2   TCP      60 46942 > 9110 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0

    [Stream index: 4]
    Sequence number: 1    (relative sequence number)
    Header length: 20 bytes
  ⊞ Flags: 0x029 (FIN, PSH, URG)
    Window size value: 1024
    [Calculated window size: 1024]
    [Window size scaling factor: -1 (unknown)]
```

Figure 6. TCP Xmas Scan



```
udp.pcapng  [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
Filter: icmp.type==3 and icmp.code==3
No.   Time        Source           Destination      Protocol Length Info
279 27.5570940 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
283 27.5580340 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
287 27.5588950 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
292 27.5630210 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
296 27.5639330 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
301 27.5683220 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
306 27.6567730 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
308 27.6597870 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
310 27.6601930 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
312 27.6606560 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
314 27.6609830 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
316 27.6613320 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
318 27.6617690 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
320 27.6621020 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
322 27.6624160 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
324 27.6659230 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
326 27.6662710 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
328 27.6667240 192.168.145.129 192.168.145.131 ICMP     77 Destination unreachable (Port unreachable)
330 27.6670510 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
332 27.6674120 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)
334 27.6677690 192.168.145.129 192.168.145.131 ICMP     70 Destination unreachable (Port unreachable)

⊟ Frame 279: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: Vmware_40:e9:3f (00:0c:29:40:e9:3f), Dst: Vmware_ca:3d:3e (00:0c:29:ca:3d:3e)
⊞ Internet Protocol Version 4, Src: 192.168.145.129 (192.168.145.129), Dst: 192.168.145.131 (192.168.145.131)
⊟ Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 3 (Port unreachable)
    Checksum: 0xa16c [correct]
  ⊞ Internet Protocol Version 4, Src: 192.168.145.131 (192.168.145.131), Dst: 192.168.145.129 (192.168.145.129)
  ⊞ User Datagram Protocol, Src Port: 60437 (60437), Dst Port: 56141 (56141)

0000  00 0c 29 ca 3d 3e 00 0c  29 40 e9 3f 08 00 45 00   ..).=>.. )@.?..E.
0010  00 38 8e 50 00 00 80 01  08 1f c0 a8 91 81 c0 a8   .8.P.... ........
0020  91 83 03 03 a1 6c 00 00  00 00 45 00 00 1c fe 00   .....l.. ..E.....
0030  00 00 39 11 df 7a c0 a8  91 83 c0 a8 91 81 ec 15   ..9..z.. ........
0040  db 4d 00 08 94 24                                  .M...$

Frame (frame), 70 bytes    Packets: 8173 · Displayed: 995 (12.2%) · Load time: 0:01.090    Profile: Default
```

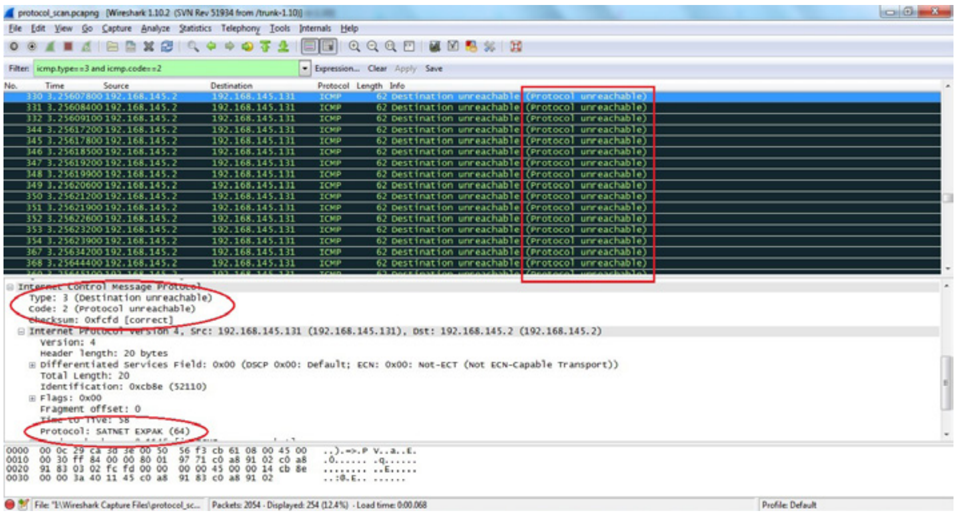
Figure 7. UDP Scan

UDP Scan

In UDP scan attacker sends a UDP packet (contains no meaningful data) on the target port and if that target responds with ICMP Type 3 Code 3 port is unavailable but if there is no response then it might be open or filtered. After capturing packets in Wireshark if you are getting high no. of packets with ICMP type 3 Code 3, it is a sign of UDP Scan. We can use filter `icmp.type==3 and icmp.code==3` to detect UDP scan in Wireshark.

IP Protocol Scan

IP Protocol Scan is helpful in finding out protocols running over IP. To detect this attacker sends packet with different protocol nos., if he gets ICMP type 3 Code 2 Packet as a response then it means that this protocol is not running on the target system while no response means protocol is there or filtered. To detect this scan in Wireshark, we can apply `icmp.type==3` and `icmp.code==2` as a filter (Figure 8).


protocol_scan.pcapng [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: icmp.type==3 and icmp.code==2

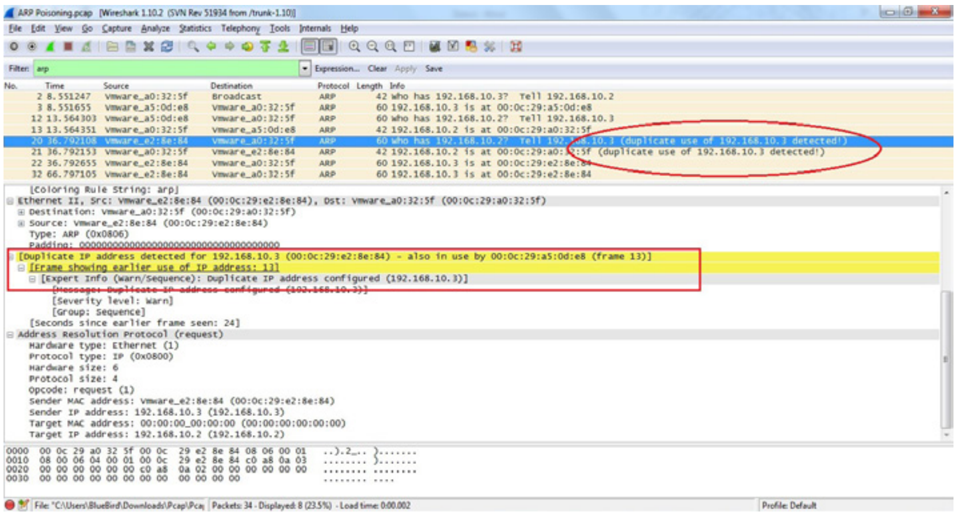
```
No.  Time        Source          Destination      Protocol Length Info
330 3.25607800 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
331 3.25608400 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
332 3.25609100 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
344 3.25617200 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
345 3.25617800 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
346 3.25618500 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
347 3.25619200 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
348 3.25619900 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
349 3.25620600 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
350 3.25621200 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
351 3.25621900 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
352 3.25622600 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
353 3.25623200 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
354 3.25623900 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
367 3.25634200 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)
368 3.25644400 192.168.145.2   192.168.145.131   ICMP     62 Destination unreachable (Protocol unreachable)

Internet Control Message Protocol
  Type: 3 (Destination unreachable)
  Code: 2 (Protocol unreachable)
  Checksum: 0xfcfd [correct]
  Internet Protocol Version 4, Src: 192.168.145.131 (192.168.145.131), Dst: 192.168.145.2 (192.168.145.2)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 20
    Identification: 0xcb8e (52110)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 58
    Protocol: SATNET EXPAK (64)

0000  00 0c 29 ca 3d 3e 00 50  56 f3 cb 61 08 00 45 00   ..).=>.P V..a..E.
0010  00 30 ff 84 00 00 80 01  97 71 c0 a8 91 02 c0 a8   .0...... .q......
0020  91 83 03 02 fc fd 00 00  00 00 45 00 00 14 cb 8e   ........ ..E.....
0030  00 00 3a 40 11 45 c0 a8  91 83 c0 a8 91 02         ..:@.E.. ......
```

File: "I:\Wireshark Capture Files\protocol_sc... | Packets: 2054 · Displayed: 254 (12.4%) · Load time: 0:00.068 | Profile: Default

Figure 8. Protocol Scan

ARP Poisoning

ARP poisoning is a layer 2 redirection technique which can be easily identified by Wireshark. If more than one MAC addresses claim to have the same IP address it will highlight that packet as *Duplicate IP Address Detected* (Figure 9).


ARP Poisoning.pcap [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

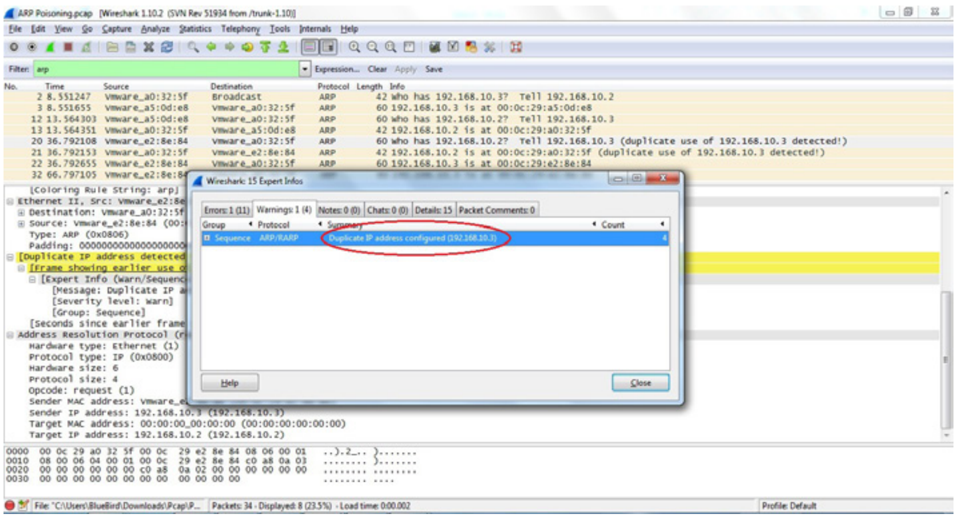
Filter: arp

```
No.  Time       Source             Destination        Protocol Length Info
  2 8.551247   Vmware_a0:32:5f    Broadcast          ARP      42 Who has 192.168.10.3?  Tell 192.168.10.2
  3 8.551655   Vmware_a5:0d:e8    Vmware_a0:32:5f    ARP      60 192.168.10.3 is at 00:0c:29:a5:0d:e8
 12 13.564303  Vmware_a5:0d:e8    Vmware_a0:32:5f    ARP      60 Who has 192.168.10.2?  Tell 192.168.10.3
 13 13.564351  Vmware_a0:32:5f    Vmware_a5:0d:e8    ARP      42 192.168.10.2 is at 00:0c:29:a0:32:5f
 20 36.792108  Vmware_e2:8e:84    Vmware_a0:32:5f    ARP      60 Who has 192.168.10.2?  Tell 192.168.10.3 (duplicate use of 192.168.10.3 detected!)
 21 36.792153  Vmware_a0:32:5f    Vmware_e2:8e:84    ARP      42 192.168.10.2 is at 00:0c:29:a0:32:5f (duplicate use of 192.168.10.3 detected!)
 22 36.792655  Vmware_e2:8e:84    Vmware_a0:32:5f    ARP      60 192.168.10.3 is at 00:0c:29:e2:8e:84
 32 66.797105  Vmware_e2:8e:84    Vmware_a0:32:5f    ARP      60 192.168.10.3 is at 00:0c:29:e2:8e:84

    [Coloring Rule String: arp]
Ethernet II, Src: Vmware_e2:8e:84 (00:0c:29:e2:8e:84), Dst: Vmware_a0:32:5f (00:0c:29:a0:32:5f)
  ⊞ Destination: Vmware_a0:32:5f (00:0c:29:a0:32:5f)
  ⊞ Source: Vmware_e2:8e:84 (00:0c:29:e2:8e:84)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
  [Duplicate IP address detected for 192.168.10.3 (00:0c:29:e2:8e:84) - also in use by 00:0c:29:a5:0d:e8 (frame 13)]
  ⊟ [Frame showing earlier use of IP address: 13]
    ⊟ [Expert Info (Warn/Sequence): Duplicate IP address configured (192.168.10.3)]
        [Message: Duplicate IP address configured (192.168.10.3)]
        [Severity level: Warn]
        [Group: Sequence]
    [Seconds since earlier frame seen: 24]
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Vmware_e2:8e:84 (00:0c:29:e2:8e:84)
    Sender IP address: 192.168.10.3 (192.168.10.3)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.10.2 (192.168.10.2)

0000  00 0c 29 a0 32 5f 00 0c  29 e2 8e 84 08 06 00 01   ..).2_.. ).......
0010  08 00 06 04 00 01 00 0c  29 e2 8e 84 c0 a8 0a 03   ........ ).......
0020  00 00 00 00 00 00 c0 a8  0a 02 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

File: "C:\Users\BlueBird\Downloads\Pcap\Pca | Packets: 34 · Displayed: 8 (23.5%) · Load time: 0:00.002 | Profile: Default

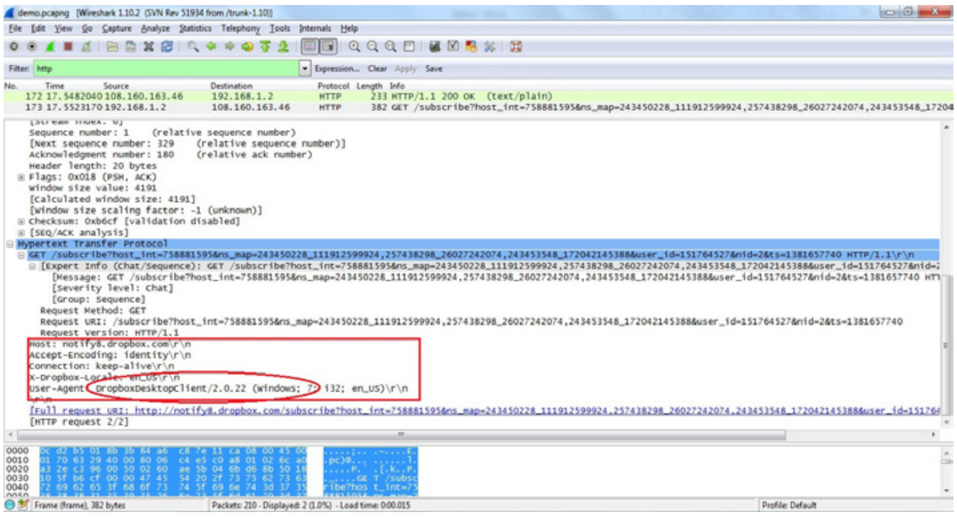
Figure 9. ARP Poisioning

If we find challenge in finding such packets, by reading packet details we can go to top menu Analyze -> ExpertInfo and then Warnings tab as shown in the picture (Figure 10). Here it will display all warning messages related to this capture which will help us to identify problems quickly.



```
ARP Poisoning.pcap  [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
Filter: arp
No.    Time       Source             Destination        Protocol Length Info
     2 8.551247   Vmware_a0:32:5f    Broadcast          ARP      42 Who has 192.168.10.3?  Tell 192.168.10.2
     3 8.551655   Vmware_a5:0d:e8    Vmware_a0:32:5f    ARP      60 192.168.10.3 is at 00:0c:29:a5:0d:e8
    12 13.564303  Vmware_a5:0d:e8    Vmware_a0:32:5f    ARP      60 Who has 192.168.10.2?  Tell 192.168.10.3
    13 13.564351  Vmware_a0:32:5f    Vmware_a5:0d:e8    ARP      42 192.168.10.2 is at 00:0c:29:a0:32:5f
    20 36.792108  Vmware_e2:8e:84    Vmware_a0:32:5f    ARP      60 Who has 192.168.10.2?  Tell 192.168.10.3 (duplicate use of 192.168.10.3 detected!)
    21 36.792153  Vmware_a0:32:5f    Vmware_e2:8e:84    ARP      42 192.168.10.2 is at 00:0c:29:a0:32:5f (duplicate use of 192.168.10.3 detected!)
    22 36.792655  Vmware_e2:8e:84    Vmware_a0:32:5f    ARP      60 192.168.10.3 is at 00:0c:29:e2:8e:84
    32 66.797105  Vmware_e2:8e:84

    [Coloring Rule String: arp]
  Ethernet II, Src: Vmware_e2:8e
    Destination: Vmware_a0:32:5f
    Source: Vmware_e2:8e:84 (00:
    Type: ARP (0x0806)
    Padding: 000000000000000000
  [Duplicate IP address detected
    [Frame showing earlier use o
      [Expert Info (Warn/Sequenc
        [Message: Duplicate IP a
        [Severity level: Warn]
        [Group: Sequence]
    [Seconds since earlier frame
  Address Resolution Protocol (r
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Vmware_e
    Sender IP address: 192.168.10.3 (192.168.10.3)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.10.2 (192.168.10.2)

0000  00 0c 29 a0 32 5f 00 0c  29 e2 8e 84 08 06 00 01   ..).2_.. )...….
0010  08 00 06 04 00 01 00 0c  29 e2 8e 84 c0 a8 0a 03   ........ ).......
0020  00 00 00 00 00 00 c0 a8  0a 02 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

File: "C:\Users\BlueBird\Downloads\Pcap\P...   Packets: 34 · Displayed: 8 (23.5%) · Load time: 0:00.002   Profile: Default

Wireshark: 15 Expert Infos
Errors: 1 (11) | Warnings: 1 (4) | Notes: 0 (0) | Chats: 0 (0) | Details: 15 | Packet Comments: 0
Group     Protocol   Summary                                      Count
Sequence  ARP/RARP   Duplicate IP address configured (192.168.10.3)   4
```

Figure 10. ExpertInfo window



```
demo.pcapng  [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
Filter: http
No.   Time        Source          Destination     Protocol Length Info
  172 17.5482040  108.160.163.46  192.168.1.2     HTTP     233 HTTP/1.1 200 OK  (text/plain)
  173 17.5523170  192.168.1.2     108.160.163.46  HTTP     382 GET /subscribe?host_int=758881595&ns_map=243450228_111912599924,257438298_26027242074,243453548_17204

    Sequence number: 1    (relative sequence number)
    [Next sequence number: 329    (relative sequence number)]
    Acknowledgment number: 180    (relative ack number)
    Header length: 20 bytes
  Flags: 0x018 (PSH, ACK)
    Window size value: 4191
    [Calculated window size: 4191]
    [Window size scaling factor: -1 (unknown)]
  Checksum: 0xb6cf [validation disabled]
  [SEQ/ACK analysis]
Hypertext Transfer Protocol
  GET /subscribe?host_int=758881595&ns_map=243450228_111912599924,257438298_26027242074,243453548_172042145388&user_id=151764527&nid=2&ts=1381657740 HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /subscribe?host_int=758881595&ns_map=243450228_111912599924,257438298_26027242074,243453548_172042145388&user_id=151764527&nid=2
      [Message: GET /subscribe?host_int=758881595&ns_map=243450228_111912599924,257438298_26027242074,243453548_172042145388&user_id=151764527&nid=2&ts=1381657740 HTT
      [Severity level: Chat]
      [Group: Sequence]
    Request Method: GET
    Request URI: /subscribe?host_int=758881595&ns_map=243450228_111912599924,257438298_26027242074,243453548_172042145388&user_id=151764527&nid=2&ts=1381657740
    Request Version: HTTP/1.1
  Host: notify8.dropbox.com\r\n
  Accept-Encoding: identity\r\n
  Connection: keep-alive\r\n
  X-Dropbox-Locale: en_US\r\n
  User-Agent: DropboxDesktopClient/2.0.22 (Windows; 7; i32; en_US)\r\n
  \r\n
  [Full request URI: http://notify8.dropbox.com/subscribe?host_int=758881595&ns_map=243450228_111912599924,257438298_26027242074,243453548_172042145388&user_id=15176
  [HTTP request 2/2]

0000  0c d2 b5 01 8b 3b 84 a6  c8 7e 11 ca 08 00 45 00   ......;.. .~....E.
0010  01 70 63 29 40 00 80 06  c4 e5 c0 a8 01 02 6c a0   .pc)@... ......l.
0020  a3 2e c3 96 00 50 02 60  ae 5b 04 6b d6 8b 50 18   .....P.` .[.k..P.
0030  10 5f b6 cf 00 00 47 45  54 20 2f 73 75 62 73 63   ._....GE T /subsc
0040  72 69 62 65 3f 68 6f 73  74 5f 69 6e 74 3d 37 35   ribe?hos t_int=75

Frame (frame), 382 bytes   Packets: 210 · Displayed: 2 (1.0%) · Load time: 0:00.015   Profile: Default
```

Figure 11. Application Mapping

Application Mapping

Wireshark can be used for application mapping as well, for example, if I am using HTTP communication then start looking for GET packet, within this packet if I will look for user-agent under Hypertext Transfer Protocol section it may reveal application OS and browser information (Figure 11). As we can see in the above picture that host is using Dropbox client tool version 2.0.22 on Windows 7 Operating System. I hope this article was useful and it will help you in understanding how Wireshark can be used to detect/analyze scanning traffic.

About the Author

Santosh Kumar has more than 8 years of experience in IT Security. He is currently working as Technical Manager (IT Security) with Koenig Solutions Ltd. Santosh is proudly certified with Check Point Certified Managed Security Expert (CCMSE), Check Point Certified Security Expert (CCSE), CISCO ASA Specialist, Certified Ethical Hacker (CEH) along with many others. He also has proven track record of streamlining security processes, design and implement efficient security solutions, lead and assist multi-disciplined, multi-national teams in achieving security efficiency.

Discover How The Attack Happened By WireShark

by Basem Helmy

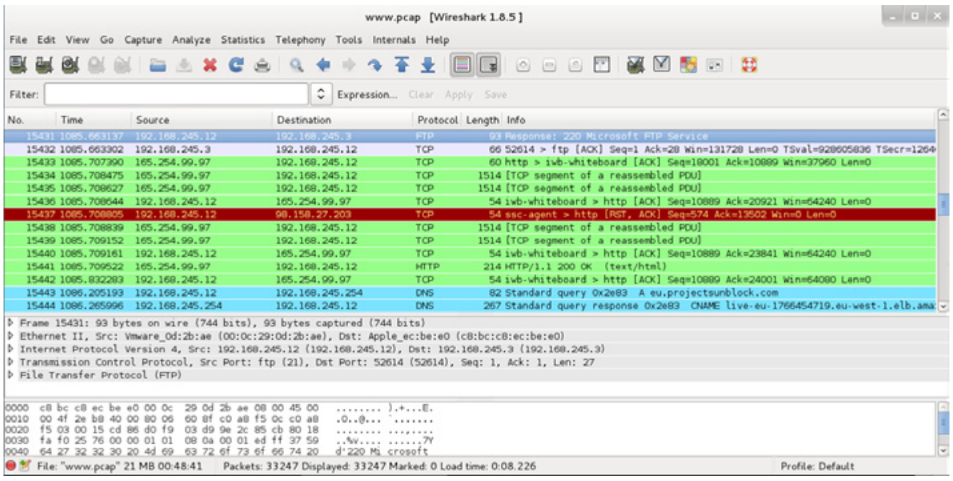
In this article you will learn how to use wireshark effectively to identify how the attack happened and what the attacker do on the compromised machine

Discover How The Attack Happened By WireShark:

In this scenario a pcap file generated by cyberlympics <ref-here> in the 2013 competition will be used to answer the following questions to identify how the attacker get in and how he extract the data from the compromised machine.

The questions are:

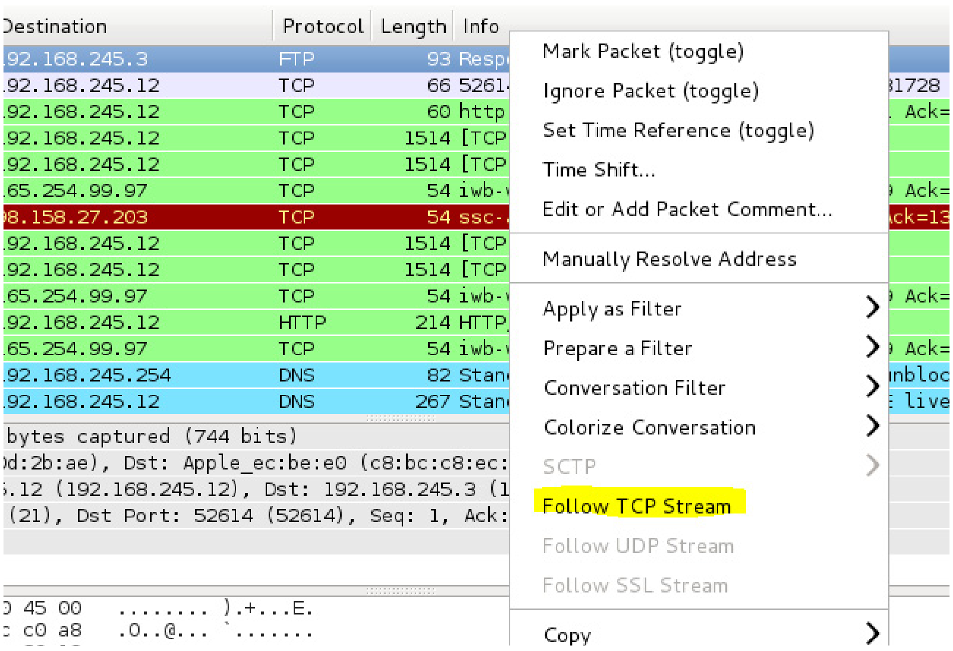
- What files were transferred to/from the victim?
- What malware/unauthorized programs were installed?
- What directory were files transferred to or from?
- What is the router password?
- What were user passwords changed to?
- We will start by loading the pcap file into wireshark


www.pcap [Wireshark 1.8.5]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time         Source            Destination        Protocol Length Info
15431 1085.663137  192.168.245.12    192.168.245.3      FTP        93 Response: 220 Microsoft FTP Service
15432 1085.663302  192.168.245.3     192.168.245.12     TCP        66 52614 > ftp [ACK] Seq=1 Ack=28 Win=131728 Len=0 TSval=928605836 TSecr=1264
15433 1085.707390  165.254.99.97     192.168.245.12     TCP        60 http > iwb-whiteboard [ACK] Seq=18001 Ack=10889 Win=37960 Len=0
15434 1085.708475  165.254.99.97     192.168.245.12     TCP      1514 [TCP segment of a reassembled PDU]
15435 1085.708627  165.254.99.97     192.168.245.12     TCP      1514 [TCP segment of a reassembled PDU]
15436 1085.708644  192.168.245.12    165.254.99.97      TCP        54 iwb-whiteboard > http [ACK] Seq=10889 Ack=20921 Win=64240 Len=0
15437 1085.708805  192.168.245.12    98.158.27.203      TCP        54 ssc-agent > http [RST, ACK] Seq=574 Ack=13502 Win=0 Len=0
15438 1085.708839  165.254.99.97     192.168.245.12     TCP      1514 [TCP segment of a reassembled PDU]
15439 1085.709152  165.254.99.97     192.168.245.12     TCP      1514 [TCP segment of a reassembled PDU]
15440 1085.709161  192.168.245.12    165.254.99.97      TCP        54 iwb-whiteboard > http [ACK] Seq=10889 Ack=23841 Win=64240 Len=0
15441 1085.709522  165.254.99.97     192.168.245.12     HTTP      214 HTTP/1.1 200 OK  (text/html)
15442 1085.832283  192.168.245.12    165.254.99.97      TCP        54 iwb-whiteboard > http [ACK] Seq=10889 Ack=24001 Win=64080 Len=0
15443 1086.205193  192.168.245.12    192.168.245.254    DNS        82 Standard query 0x2e83  A eu.projectsunblock.com
15444 1086.265996  192.168.245.254   192.168.245.12     DNS       267 Standard query response 0x2e83  CNAME live-eu-1766454719.eu-west-1.elb.ama

▷ Frame 15431: 93 bytes on wire (744 bits), 93 bytes captured (744 bits)
▷ Ethernet II, Src: Vmware_0d:2b:ae (00:0c:29:0d:2b:ae), Dst: Apple_ec:be:e0 (c8:bc:c8:ec:be:e0)
▷ Internet Protocol Version 4, Src: 192.168.245.12 (192.168.245.12), Dst: 192.168.245.3 (192.168.245.3)
▷ Transmission Control Protocol, Src Port: ftp (21), Dst Port: 52614 (52614), Seq: 1, Ack: 1, Len: 27
▷ File Transfer Protocol (FTP)

0000  c8 bc c8 ec be e0 00 0c  29 0d 2b ae 08 00 45 00   ........ ).+...E.
0010  00 4f 2e b8 40 00 80 06  60 8f c0 a8 f5 0c c0 a8   .O..@... `.......
0020  f5 03 00 15 cd 86 d0 f9  03 d9 9e 2c 85 cb 80 18   ........ ...,....
0030  fa f0 25 76 00 00 01 01  08 0a 00 01 ed ff 37 59   ..%v.... ......7Y
0040  64 27 32 32 30 20 4d 69  63 72 6f 73 6f 66 74 20   d'220 Mi crosoft

File: "www.pcap" 21 MB 00:48:41   Packets: 33247 Displayed: 33247 Marked: 0 Load time: 0:08.226   Profile: Default
```

Figure 1. Loading pcap file into wireshark

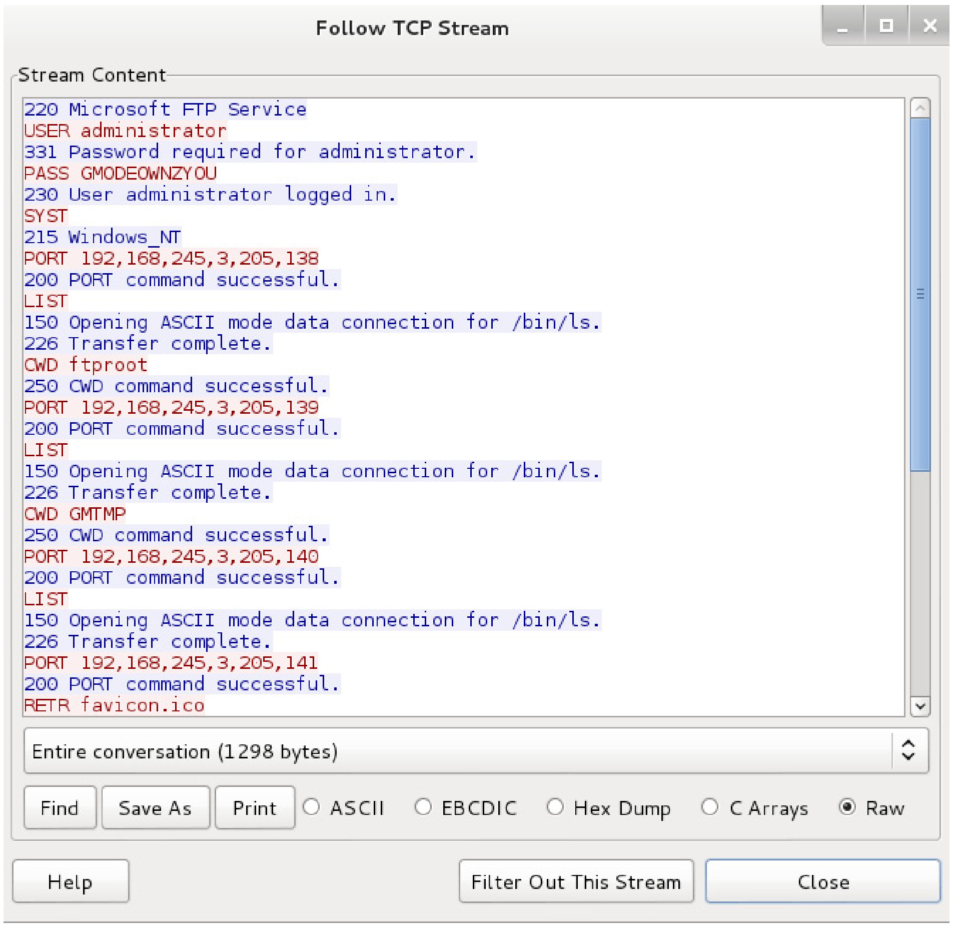
After while navigating throw the packet we identify interesting packets. Those packets are using FTP protocol. We will follow the stream to figure out what have been done.



```
Destination       Protocol Length Info
92.168.245.3      FTP        93 Resp
92.168.245.12     TCP        66 5261                       1728
92.168.245.12     TCP        60 http                       Ack=
92.168.245.12     TCP      1514 [TCP
92.168.245.12     TCP      1514 [TCP
65.254.99.97      TCP        54 iwb-                       Ack=
8.158.27.203      TCP        54 ssc-                       ck=13
92.168.245.12     TCP      1514 [TCP
92.168.245.12     TCP      1514 [TCP
65.254.99.97      TCP        54 iwb-                       Ack=
92.168.245.12     HTTP      214 HTTP
65.254.99.97      TCP        54 iwb-                       Ack=
92.168.245.254    DNS        82 Stan                       nbloc
92.168.245.12     DNS       267 Stan                       live

bytes captured (744 bits)
d:2b:ae), Dst: Apple_ec:be:e0 (c8:bc:c8:ec:
.12 (192.168.245.12), Dst: 192.168.245.3 (1
(21), Dst Port: 52614 (52614), Seq: 1, Ack:

 45 00   ........ ).+...E.
 c0 a8   .0..@... `.......
```

Figure 2. Right click on the packet the select follow tcp stream

We found really interesting data in the stream



Follow TCP Stream

Stream Content

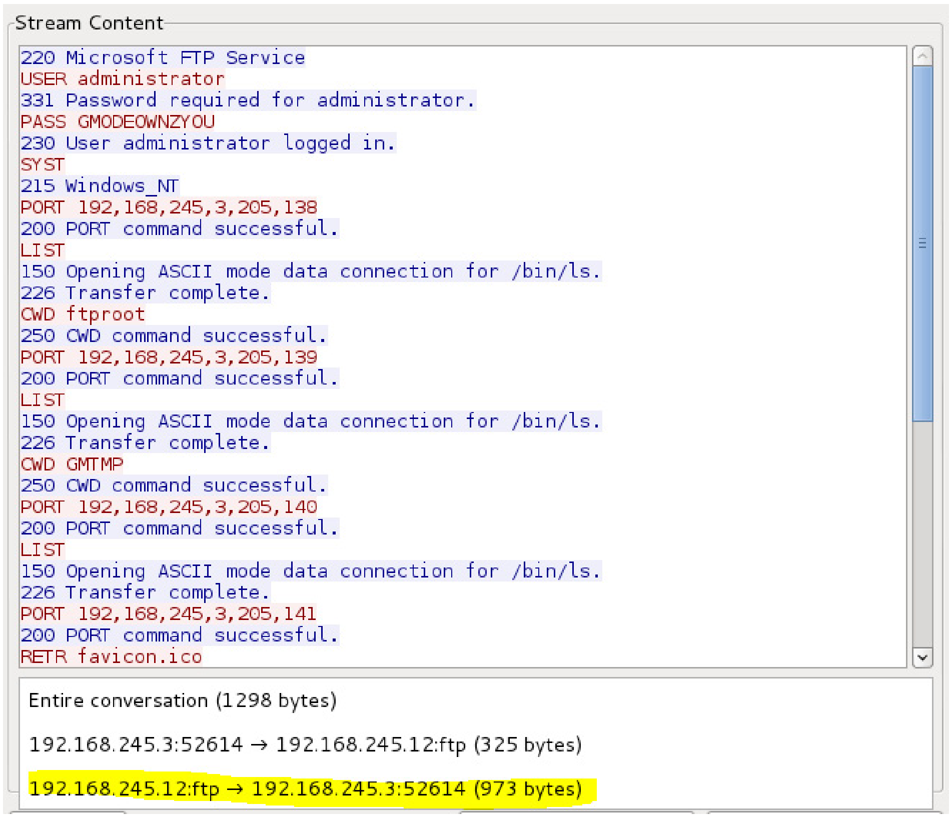
```
220 Microsoft FTP Service
USER administrator
331 Password required for administrator.
PASS GMODEOWNZYOU
230 User administrator logged in.
SYST
215 Windows_NT
PORT 192,168,245,3,205,138
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
CWD ftproot
250 CWD command successful.
PORT 192,168,245,3,205,139
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
CWD GMTMP
250 CWD command successful.
PORT 192,168,245,3,205,140
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
PORT 192,168,245,3,205,141
200 PORT command successful.
RETR favicon.ico
```

Entire conversation (1298 bytes)

ASCII EBCDIC Hex Dump C Arrays [✓ Raw]

Figure 3. Stream content

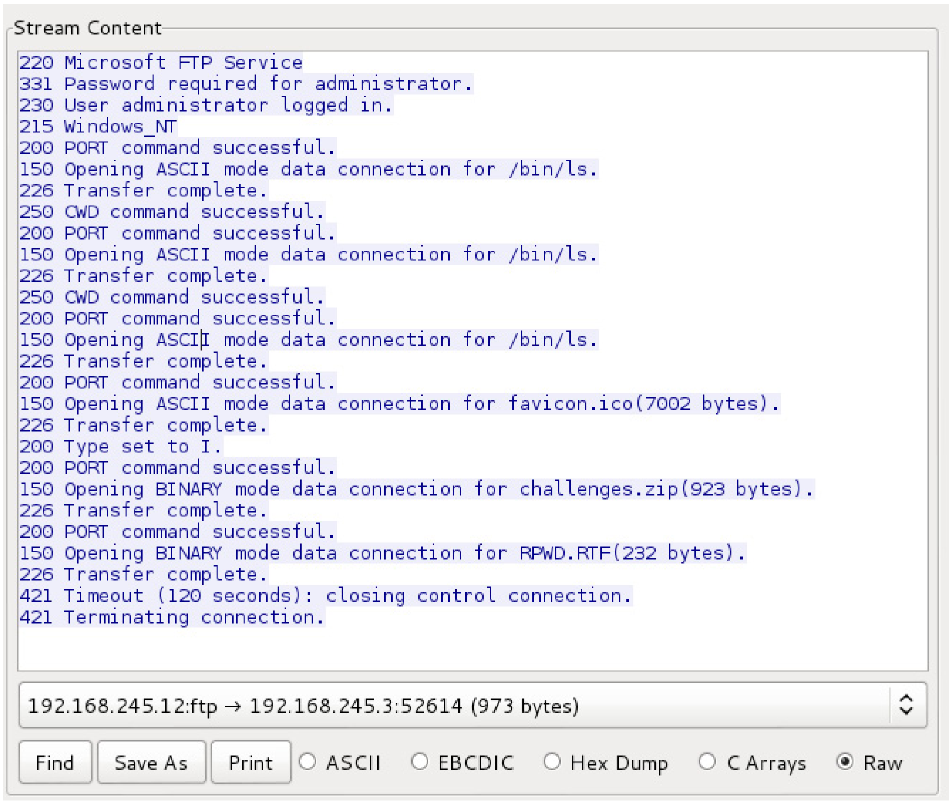
Here we go, a username *Administrator* with password *GMODEOWNZYOU* has been logged in to the victim. To identify the victim ip which is the ftp server we will select stream as follow



```
Stream Content
220 Microsoft FTP Service
USER administrator
331 Password required for administrator.
PASS GMODEOWNZYOU
230 User administrator logged in.
SYST
215 Windows_NT
PORT 192,168,245,3,205,138
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
CWD ftproot
250 CWD command successful.
PORT 192,168,245,3,205,139
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
CWD GMTMP
250 CWD command successful.
PORT 192,168,245,3,205,140
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
PORT 192,168,245,3,205,141
200 PORT command successful.
RETR favicon.ico

Entire conversation (1298 bytes)
192.168.245.3:52614 → 192.168.245.12:ftp (325 bytes)
192.168.245.12:ftp → 192.168.245.3:52614 (973 bytes)
```

Figure 4. Select ftp server stream content

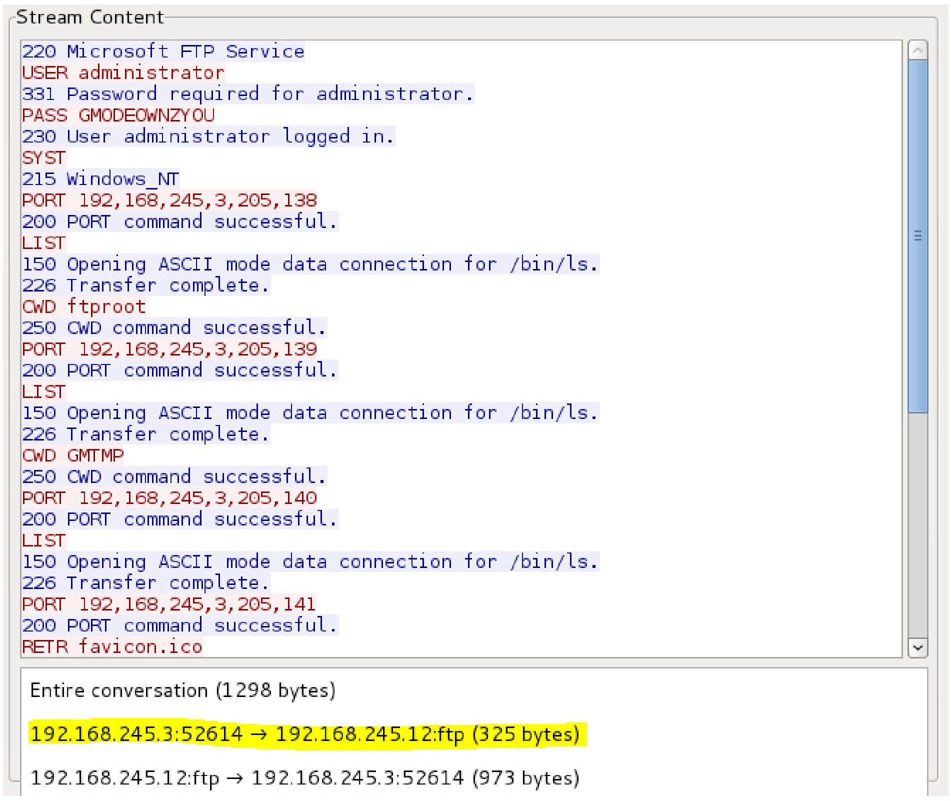
We will identify that the stream is represent a response command from the ftp server. Therefore the victim IP is 192.168.245.12 and the attacker IP 192.168.245.3



```
Stream Content
220 Microsoft FTP Service
331 Password required for administrator.
230 User administrator logged in.
215 Windows_NT
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
250 CWD command successful.
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
250 CWD command successful.
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
200 PORT command successful.
150 Opening ASCII mode data connection for favicon.ico(7002 bytes).
226 Transfer complete.
200 Type set to I.
200 PORT command successful.
150 Opening BINARY mode data connection for challenges.zip(923 bytes).
226 Transfer complete.
200 PORT command successful.
150 Opening BINARY mode data connection for RPWD.RTF(232 bytes).
226 Transfer complete.
421 Timeout (120 seconds): closing control connection.
421 Terminating connection.

192.168.245.12:ftp → 192.168.245.3:52614 (973 bytes)
```

Figure 5. ftp server response stream content

So we need now to identify the attacker stream by select as follow



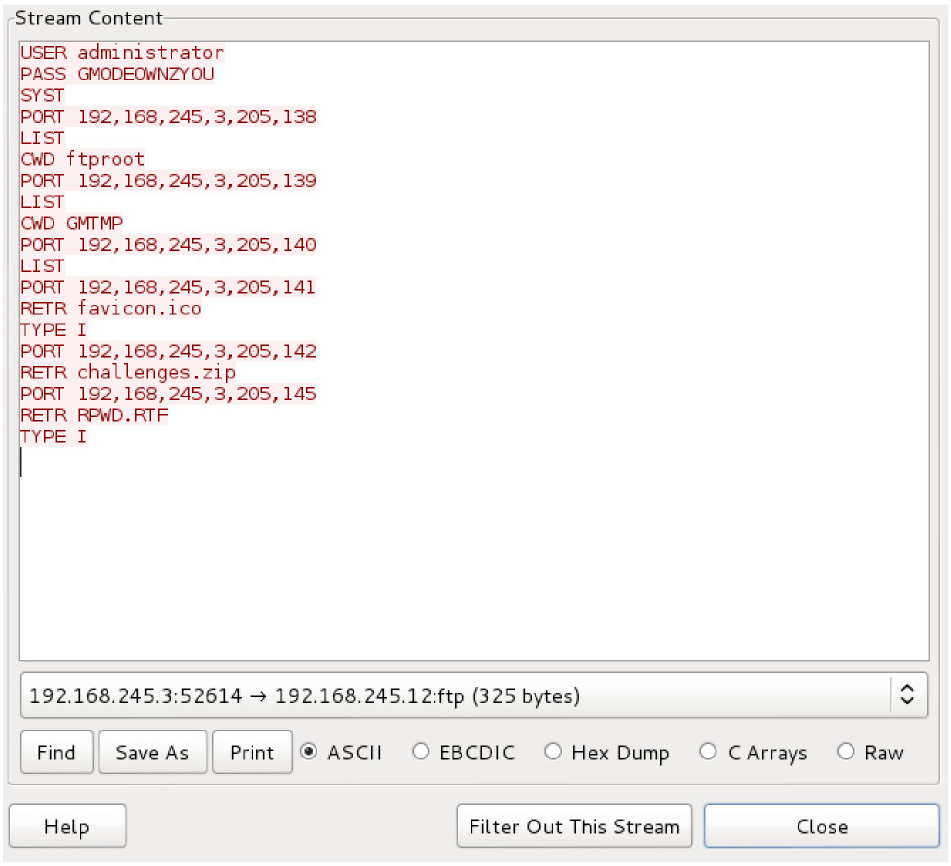
```
220 Microsoft FTP Service
USER administrator
331 Password required for administrator.
PASS GMODEOWNZYOU
230 User administrator logged in.
SYST
215 Windows_NT
PORT 192,168,245,3,205,138
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
CWD ftproot
250 CWD command successful.
PORT 192,168,245,3,205,139
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
CWD GMTMP
250 CWD command successful.
PORT 192,168,245,3,205,140
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Transfer complete.
PORT 192,168,245,3,205,141
200 PORT command successful.
RETR favicon.ico
```

Entire conversation (1298 bytes)

192.168.245.3:52614 → 192.168.245.12:ftp (325 bytes)

192.168.245.12:ftp → 192.168.245.3:52614 (973 bytes)

Figure 6. Select attacker stream content



```
USER administrator
PASS GMODEOWNZYOU
SYST
PORT 192,168,245,3,205,138
LIST
CWD ftproot
PORT 192,168,245,3,205,139
LIST
CWD GMTMP
PORT 192,168,245,3,205,140
LIST
PORT 192,168,245,3,205,141
RETR favicon.ico
TYPE I
PORT 192,168,245,3,205,142
RETR challenges.zip
PORT 192,168,245,3,205,145
RETR RPWD.RTF
TYPE I
```

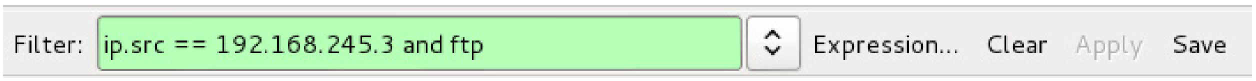
192.168.245.3:52614 → 192.168.245.12:ftp (325 bytes)

Figure 7. Attacker stream content

From the stream we could identify that the attacker use the following ftp commands to retrieve some data from the ftp server

- *CWD ftproot:* this command used to change the working directory to ftproot
- *CWD GMTMP:* this command used to change the working directory to GMTMP
- *LIST:* this command used to list files inside the GMTMP directory
- *RETR favicon.ico:* this command used to download favicon.ico to the attacker machine
- *RETR challenges.zip:* this command used to download challenges.zip to the attacker machine
- *RETR RPWD.RTF:* this command used to download RPWD.RTF to the attacker machine

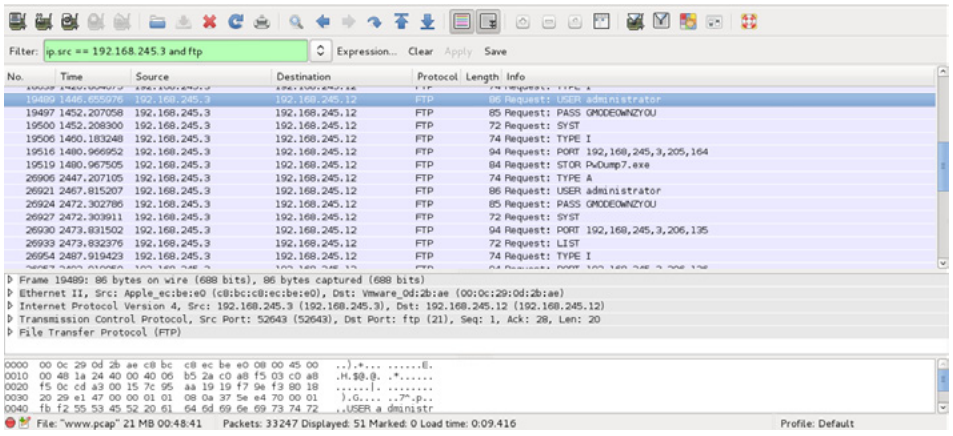
Currently we know that the attacker IP is 192.168.245.3. We will make a filter on the source IP of the attacker as follow



Filter: ip.src == 192.168.245.3 and ftp

Figure 8. Filter on the source IP of the attacker and ftp protocol

We found interesting stream in Figure 9

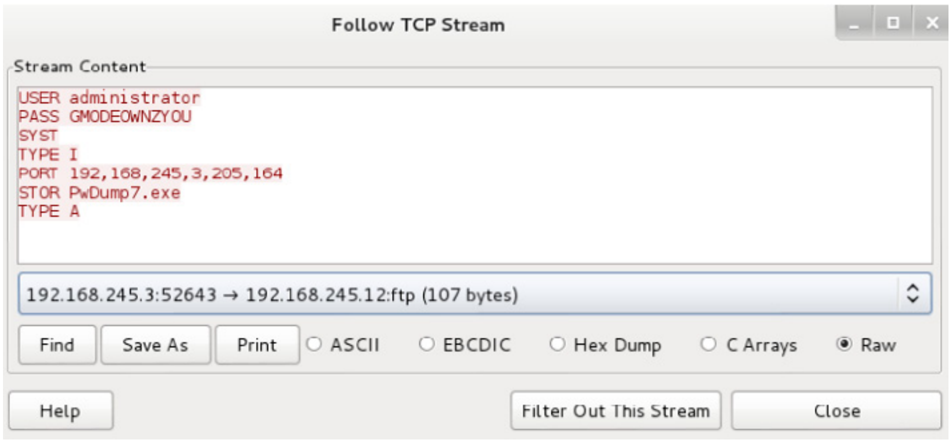


Filter: ip.src == 192.168.245.3 and ftp

```
No.   Time         Source         Destination     Protocol Length Info
19489 1446.655076  192.168.245.3  192.168.245.12  FTP      86 Request: USER administrator
19497 1452.207058  192.168.245.3  192.168.245.12  FTP      85 Request: PASS GMODEOWNZYOU
19500 1452.208300  192.168.245.3  192.168.245.12  FTP      72 Request: SYST
19506 1460.183248  192.168.245.3  192.168.245.12  FTP      74 Request: TYPE I
19516 1480.966952  192.168.245.3  192.168.245.12  FTP      94 Request: PORT 192,168,245,3,205,164
19519 1480.967505  192.168.245.3  192.168.245.12  FTP      84 Request: STOR PwDump7.exe
26906 2447.207105  192.168.245.3  192.168.245.12  FTP      74 Request: TYPE A
26921 2467.815207  192.168.245.3  192.168.245.12  FTP      86 Request: USER administrator
26924 2472.302786  192.168.245.3  192.168.245.12  FTP      85 Request: PASS GMODEOWNZYOU
26927 2472.303911  192.168.245.3  192.168.245.12  FTP      72 Request: SYST
26930 2473.831502  192.168.245.3  192.168.245.12  FTP      94 Request: PORT 192,168,245,3,206,135
26933 2473.832376  192.168.245.3  192.168.245.12  FTP      72 Request: LIST
26954 2487.919423  192.168.245.3  192.168.245.12  FTP      74 Request: TYPE I

▷ Frame 19489: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
▷ Ethernet II, Src: Apple_ec:be:e0 (c8:bc:c8:ec:be:e0), Dst: Vmware_0d:2b:ae (00:0c:29:0d:2b:ae)
▷ Internet Protocol Version 4, Src: 192.168.245.3 (192.168.245.3), Dst: 192.168.245.12 (192.168.245.12)
▷ Transmission Control Protocol, Src Port: 52643 (52643), Dst Port: ftp (21), Seq: 1, Ack: 28, Len: 20
▷ File Transfer Protocol (FTP)

0000  00 0c 29 0d 2b ae c8 bc  c8 ec be e0 08 00 45 00   ..).+... ......E.
0010  00 48 1a 24 40 00 40 06  b5 2a c0 a8 f5 03 c0 a8   .H.$@.@. .*......
0020  f5 0c cd a3 00 15 7c 95  aa 19 19 f7 9e f3 80 18   ......|. ........
0030  20 29 e1 47 00 00 01 01  08 0a 37 5e e4 70 00 01    ).G.... ..7^.p..
0040  fb f2 55 53 45 52 20 61  64 6d 69 6e 69 73 74 72   ..USER a dministr

File: "www.pcap" 21 MB 00:48:41   Packets: 33247 Displayed: 51 Marked: 0 Load time: 0:09.416   Profile: Default
```

Figure 9. Attacker logged in before

We will use the same instructions as shown before in Figure 2 and Figure 6 to get the attacker stream



Follow TCP Stream

Stream Content

```
USER administrator
PASS GMODEOWNZYOU
SYST
TYPE I
PORT 192,168,245,3,205,164
STOR PwDump7.exe
TYPE A
```

192.168.245.3:52643 → 192.168.245.12:ftp (107 bytes)

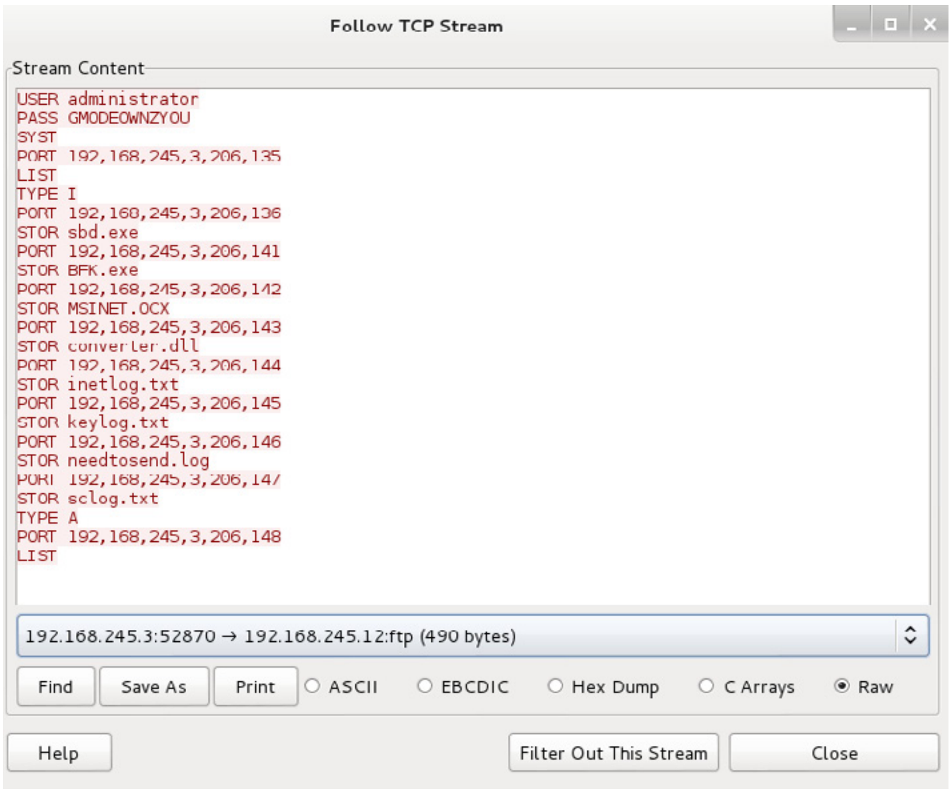
ASCII EBCDIC Hex Dump C Arrays [✓ Raw]

Figure 10. Attacker stream

From this stream we could identify that the attacker use the following ftp commands to store file on the ftp server

- *TYPE I:* this command used to set the type of file to be transferred to binary data or image
- *STOR PwDump7.exe:* this command used upload pwdump7.exe to the ftp server
- *TYPE A:* this command used to set the type of file to be transferred to ASCII text

Also here is another stream



```
USER administrator
PASS GMODEOWNZYOU
SYST
PORT 192,168,245,3,206,135
LIST
TYPE I
PORT 192,168,245,3,206,136
STOR sbd.exe
PORT 192,168,245,3,206,141
STOR BFK.exe
PORT 192,168,245,3,206,142
STOR MSINET.OCX
PORT 192,168,245,3,206,143
STOR converter.dll
PORT 192,168,245,3,206,144
STOR inetlog.txt
PORT 192,168,245,3,206,145
STOR keylog.txt
PORT 192,168,245,3,206,146
STOR needtosend.log
PORT 192,168,245,3,206,147
STOR sclog.txt
TYPE A
PORT 192,168,245,3,206,148
LIST
```

192.168.245.3:52870 → 192.168.245.12:ftp (490 bytes)

○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays [✓ Raw]

Figure 11. Attacker stream

From this stream we could identify that the attacker use the following ftp commands to store file on the ftp server

- *TYPE I:* this command used to set the type of file to be transferred to binary data or image
- *STOR sbd.exe,BFK.exe, MSINET.OCX, convert.dll, inetlog.txt, keylog.txt, needtosend.log and sclog.txt:* this command used upload files to the ftp server
- *TYPE A:* this command used to set the type of file to be transferred to ASCII text
- *LIST:* this command used to list files inside the GMTMP directory

So by the previous techniques you can figure out and answer the previous questions as follow:

What files were transferred to/from the victim?

Using wireshark filter “ftp.request.command==STOR” you find the following applications transferred to the victim 192.168.245.12

- PwDump7.exe
- sbd.exe

- BFK.exe
- MSINET.OCX
- converter.dll
- inetlog.txt
- keylog.txt
- needtosend.log

Using wireshark filter “ftp.request.command==RETR” you find the following applications transferred from the victim 192.168.245.12

- favicon.ico
- challenges.zip
- RPWD.RTF

What malware/unauthorized programs were installed?

According to The Logs and the file transfered to the vivtim The programs are:

- PwDump7.exe
- sbd.exe
- BFK.exe

What directory were files transferred to or from?

- All Files Transferred to a new Directory in C:\Inetpub\ftproot\ with name GMTMP
- Full path C:\Inetpub\ftproot\GMTMP

What is the router password?

- Router Password Found in RPWD.RTF file
- Router Password Encryption Type 7 0139562C753F2E5C067E16
- Router Clear Text Password J0HNTH3GR8

What were user passwords changed to?

Password changed for John, Administrator and nonadmin accounts to be GMODEOWNZYOU

About the Author

Basem Helmy| ECSA/LPT. He is information Security Engineer specialist in offensive security track. He is specialist in penetration testing for network and web applications in highly secured environments for more than 3 years' experience. LinkedIn: https://www.linkedin.com/in/bhelmy. Twitter: https://twitter.com/b4s3mh3lmy

AppDra

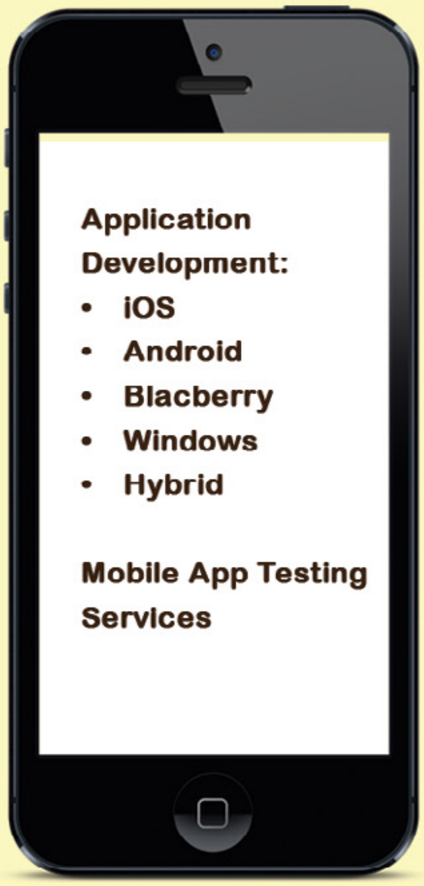
Software Solutions Pvt Ltd.

Pioneers in Mobility and Web Solutions

AppDra Software Solutions Pvt Ltd., headquartered in the IT capital of India, Bangalore.

Our primary focus is to design and deliver innovative, breakthrough and cost effective solutions to not only represent our customers' business, but also to further leverage it.

Our team at AppDra are not limited to just technical brilliance but bring to the fore the most creative prowess, which reflect in our services. We strive to not just building solutions, but further leverage technologies.



Application Development:

- **iOS**
- **Android**
- **Blacberry**
- **Windows**
- **Hybrid**

Mobile App Testing Services

Website Design and Development

Web Application Development

Ecommerce Development

Contact:

Raghav - +91-9886129128

Email: contact@appdra.com

Website: www.appdra.com

Detecting Attacks and Threats in Elastic Cloud Infrastructures: the Case of Side-channel Attacks

by Pasquale Puzio, Sergio Loureiro

Cloud computing adoption is rising fast. Flexibility, pay-per-use and available resources on-demand with the promise of lower ownership costs are a very attractive value proposition.

Virtualization is just a part of Cloud Computing, which leverages virtualization in order to provide resources in an on-demand and pay-per-use fashion.

According to the official NIST definition, Cloud Computing can be classified into three main models:

- IaaS (Infrastructure as a Service): users have access to on-demand virtual machines and storage which are provided from large shared data centers;
- PaaS (Platform as a Service): what is offered by IaaS plus ready-to-use development environment;
- SaaS (Software as a Service): on-demand software hosted on a remote server.

In this article we will focus on the main security concerns arising in IaaS environment, in particular multi-tenancy.

Multi-tenancy

The wide adoption of Cloud technologies has brought enormous advantages but also several new security concerns. One of the main causes for these new threats is multi-tenancy. Cloud Computing widespreads new scenarios in which users' applications and data share the same physical host.

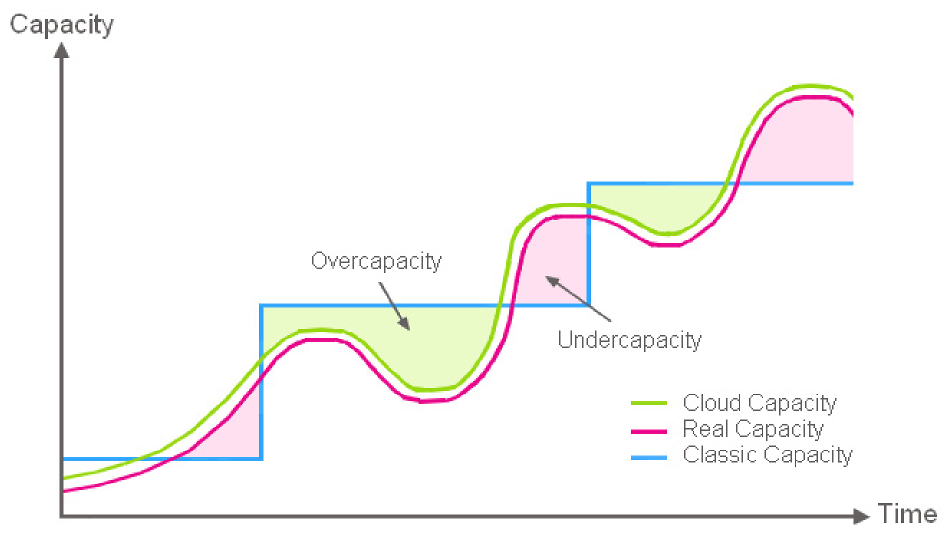


Figure 1. Cloud capacities comparison

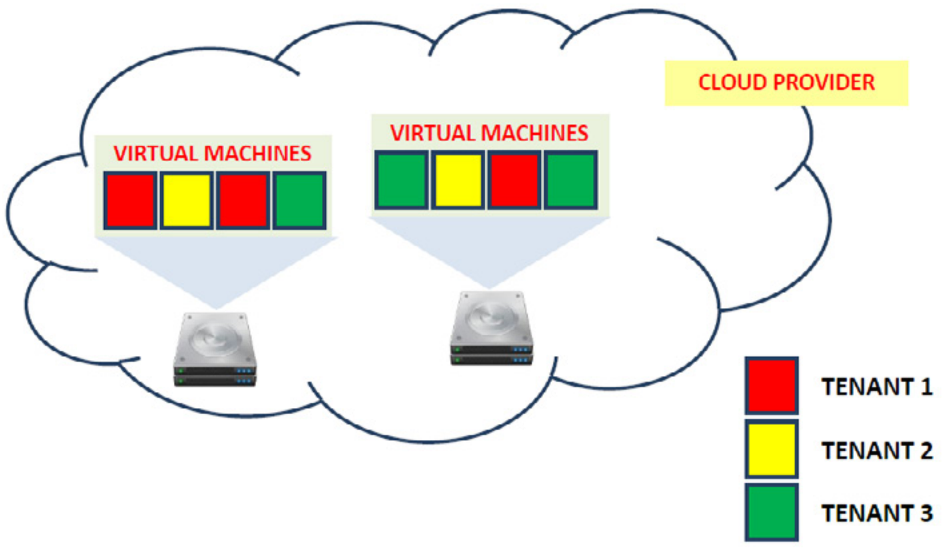


Figure 2. Basic cloud structure

Because of co-residency, an attacker has a new way to access the victim's data: he can leverage the co-residency factor and infer victim's data by observing the activity of a shared component (e.g. the processor cache) on the physical host.

This new class of attacks is called "Side-channel attacks". While side-channel attacks have been developed for years, for example on the smartcard field, their application to cloud computing infrastructures is just starting.

Problem Statement

Side-channel attacks

Before performing a side-channel attack, the attacker needs to create a virtual machine and determine if it is running on the same physical host of the victim. In order to do so, he has to perform the so called "co-residency". The authors of [9] defined three possible ways to detect co-residency:

- matching Dom0 IP address: if the Dom0 IP address of two machines is equal, then co-residency has been achieved;
- measuring the round-trip time: if the round-trip time between two machines is very small, there is a high chance of co-residency;
- observing IP addresses: if two IP addresses are close enough, there is a high chance of co-residency.

On Amazon EC2, check #1 proved to be the most effective with a false-positive rate of zero. This means that check #1 is sufficient to determine co-residency.

If the result of the co-residency check is positive, the attacker can go ahead with the side-channel attack. Many techniques belonging to this class have been shown in the context of Cloud Computing Infrastructures, but until now, just one has been implemented and actually proved [8]. This technique can be classified as an "access-driven" attack since it manages to retrieve a given information which belongs to another user by constantly observing (and accessing) the activity of a shared physical component, the processor cache.

However, even if this kind of attacks has been widely treated in the literature, performing such an attack is very hard and there are several challenges to take into account:

- the attacker-VM needs to frequently monitor the status of the processor cache, so it needs to go in execution often enough in order to make the observation granularity as fine-grained as possible;
- the attacker-VM has to clean the results and reduce the noise introduced by hardware and software sources;
- last but not least, the attacker-VM has to be able to detect when the target VM is not running anymore on the same physical host.

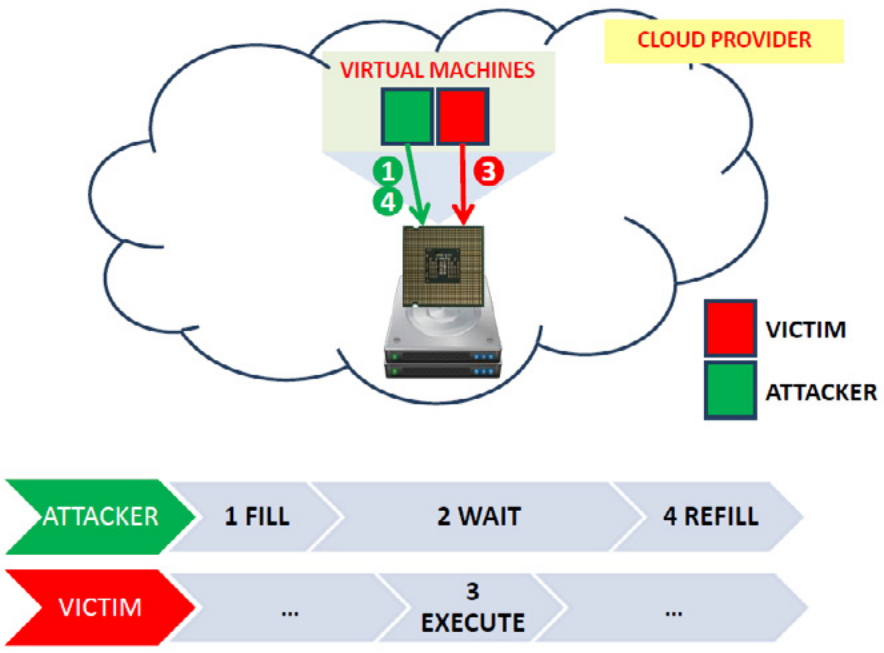


Figure 3. Basic attack scheme

The strategy adopted by the authors [8] to extract the wanted information can be summarized as follows:

- PRIME: the attacker fills the processor cache;
- IDLE: the attacker waits for a pseudo-random interval. During this interval the target VM is supposed to access the cache and thus change the content of some blocks;
- PROBE: when the attacker resumes the execution, it refills the cache in order to learn the activity of the target VM on the cache.

Once the attacker has collected a sufficiently high number of measurements, it can finally analyze these measurements and infer the encryption key used by the target VM.

During the analysis, the measurements achieved during the previous phase are converted to basic operations (operations performed during the execution of the target VM). This phase is called “cache pattern classifier”. Of course, doing this requires to know in advance the algorithm which is being executed on the target VM.

After classifying each operation, a special Markov Model (Hidden Markov Model) is used to remove noises and apply some heuristics (based on knowledge of the algorithm) in order to reconstruct the possible execution paths. Finally, the attacker obtains a set containing all the possible encryption keys. This set is composed by few thousands of keys, so the attacker can use these keys to perform a brute-force attack.

The existence of such an attack is the proof that traditional security systems such as intrusion detection systems, antivirus, firewalls, etc. are not effective anymore in today’s virtualized and multi-tenant environment, that is the Cloud Infrastructure. In order to protect users against Cloud threats, a new approach is required and new techniques need to be developed.

Solution

In this section we will describe in detail all the characteristics of our solution and the rationale behind it. At a high level of abstraction, the architecture we propose is composed by three main components: a cloud provider, Elastic Detector and a SIEM.

The cloud provider provides an API which allows Elastic Detector to retrieve information on the status of the infrastructure and events which are useful for the detection of threats and attacks.

Elastic Detector operates as an intermediary and its duty is to handle elasticity and make it transparent to the SIEM system. Indeed, existing SIEM solutions do not take into account the elasticity of modern cloud infrastructures.

Thanks to Elastic Detector, the SIEM system can work as usual to analyze and correlate logs.

The final goal of such an architecture is to catch security-relevant events in order to detect threats and ongoing attacks.

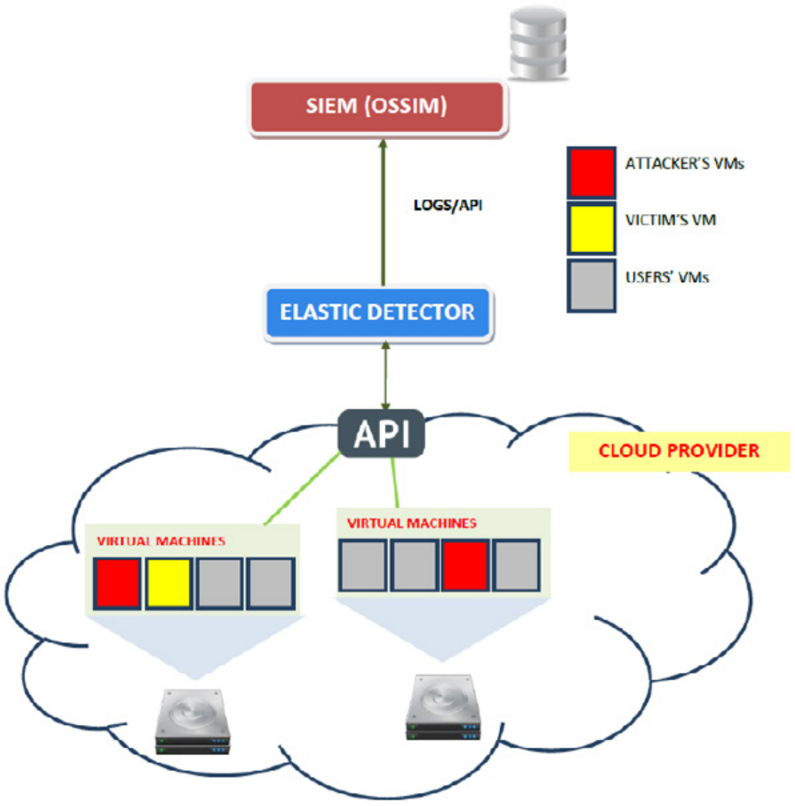


Figure 4. Elastic detector

Elastic Detector

Cloud Computing brought, along with several benefits, a set of problems that changed the way we handle security and need to be addressed in order to meet security needs:

- Lack of visibility. IaaS is more dynamic than classical infrastructures, since servers, network and storage are launched for temporary usage or automatically. This makes it difficult to keep track of the availability of each server, network and storage as well as their security status.
- Security degradation over time. Modifications to an IaaS environment, such as starting new services, tests and starting new machines, generally reduce the level of protection of a system over time, which increases the risk of external and internal attacks.

- Manual configuration errors. Today, due to the complexity and dynamic nature of cloud computing infrastructures security in such environments can no longer be handled manually.
- New attack vectors and threats. The capabilities and the flexibility of IaaS brings as well new threats as the nefarious use of resources by malicious insiders or threats related to the virtualization and APIs technologies.

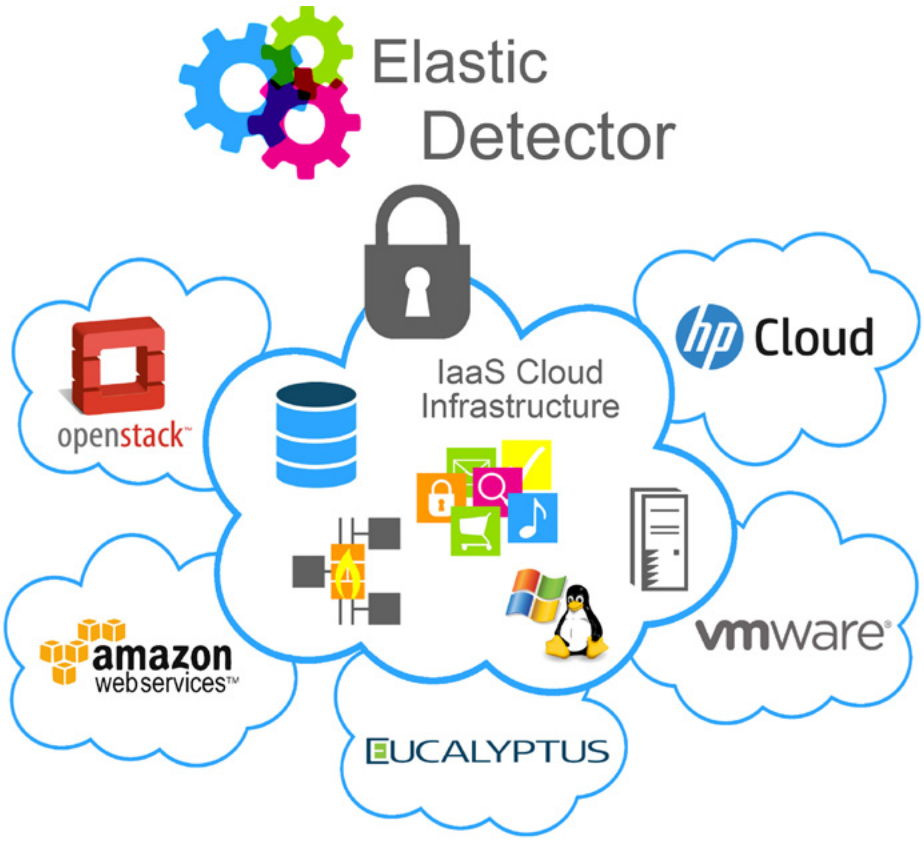


Figure 5. Services that use elastic detector

Moreover, Cloud Computing brings also a new way to build and manage IT infrastructures. Compared to the traditional approaches, thanks to cloud technologies, such as APIs, infrastructures can become highly elastic. This means that the set of active virtual machines, the storage and the network topology can change very fast. As a consequence, the security system needs as well to react to these changes as fast as possible and adapt its configuration to the new scenario.

Due to the highly elastic and easy-to-use nature of Cloud Computing technologies, it's getting easier for attackers to find vulnerable or not properly protected resources. As a proof, researchers [2] managed to retrieve a large amount of private data from public Amazon EC2 AMIs. The same kind of vulnerabilities has been found on Amazon S3 buckets by Rapid7 staff [3].

Our proposed solution to fulfill these requirements can be summarized as "virtual machine cloning". By cloning the virtual machine to be analyzed, we can perform very deep and intrusive security checks without impacting the performance of the applications in production. This way, the vulnerability assessment process can run smoothly and in a completely automatic way. It is worth pointing out that this solution is feasible thanks to the features of Cloud Computing. Indeed, in a traditional environment (e.g. on-the-premises data centers), it would not be possible to clone a machine on the fly and destroy the clone after performing all the required security checks.

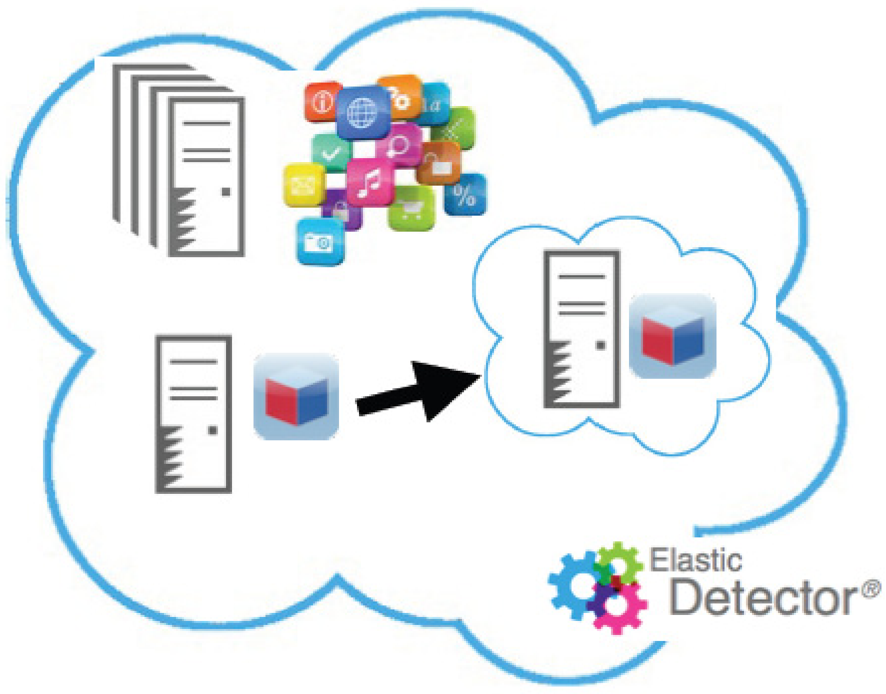


Figure 6. Cloning

From a practical point of view, cloning is the easiest solution to deploy. Indeed, no software (e.g. agent) has to be installed on virtual machines or within your cloud infrastructure. Every action can be performed by taking advantage of the API provided by the IaaS provider.

Cloning is also cost-effective since the cost of an additional virtual machine for a short period of time time is very low within IaaS infrastructures. Moreover, as the whole analysis is performed on a different machine, cloning avoids the risks of breaking applications and losing data.

Furthermore, new elastic and pay per use infrastructures bring higher percentages of stopped servers. These "dormant" servers constitute potential threats to the infrastructure as acknowledged by the Cloud Security Alliance. While stopped, the servers are not surveilled by agents or agentless solutions and they are not patched. They become weak links of your infrastructure when started. That's why we propose to test and raise alerts in case of vulnerabilities in your dormant servers.

Auto-checks should be automatically set in order to monitor your IT cloud infrastructure. This is mandatory on a continuously changing infrastructure. Therefore, while your IT infrastructure evolves to answer your business needs, the right security checks are automatically set.

Our vision is that only a fully automated approach to security can cope with the elastic nature of new cloud infrastructures and their new threats.

SIEM

Detecting attacks in distributed environments, such as cloud infrastructures, often requires the capability of analyzing logs and correlating several events from different sources. Nowadays, the best approach to threat detection in distributed environments is to employ a SIEM (Security Information and Event Management) system.

A SIEM system is much more than an analyzer of logs. A SIEM system takes care of several aspects which can be summarized as follows:

- log and context data collection;
- normalization and categorization;
- correlation;
- notification/alerting;
- prioritization;
- dashboards and visualization;
- reporting and report delivery.

A SIEM system works with different kinds of data coming from different sources within one or more networks. Starting from 2006 [6], when Cloud Computing appeared for the first time, a decentralization process has started and several companies have decided to move their own data and infrastructures to the cloud. Latest trend analysis [7] show that this tendency is not going to stop, so in the next few years, the number of companies adopting cloud computing technologies will be even higher than now. Due to the remote nature of the cloud computing, several new security concerns arise and companies are worried about the protection of their own data and infrastructures.

In this scenario, SIEM plays a very important role and needs to effectively operate in new cloud networks. Unfortunately, adopting SIEM systems in elastic cloud infrastructures is not an easy task. Indeed, SIEM systems were designed to operate in traditional (static) environments. Therefore, we need to adapt existing solutions in order to deal with elasticity. This means that a modern SIEM system should have the capability of automatically detecting the virtual machines running in the cloud, performing security checks on them and logging any security-relevant activity.

Nowadays, a system administrator needs to reconfigure his own SIEM system every time there is a change (e.g. a new virtual machine starts running) in the cloud infrastructure. Configuring a SIEM system is a very difficult task which requires the experience and skills of a security expert. Therefore, in highly dynamic infrastructures, reconfiguring the SIEM system every time there is a change in the cloud infrastructure would not be practical.

A possible alternative could be to install agents on each virtual machine running in the cloud but this approach has various drawbacks affecting especially performance.

Elastic Detector, which is our flagship product, can be an easy and effective solution to make the deployment of a SIEM for the cloud much faster. Moreover, Elastic Detector could easily aggregate all the data retrieved by performing auto-checks in the cloud network and forward it to the SIEM system, where it can be properly processed together with data collected locally.

From a practical point of view, integrating Elastic Detector with any SIEM system would not require any complex integration platform. In particular, we analyzed the integration of Elastic Detector with OSSIM [4], an open-source and widely adopted system for SIEM. One of the benefits brought by OSSIM is the possibility of easily developing and integrating custom components (plugins).

Architecture

As we mentioned above, we want to take advantage of the capabilities of a SIEM system in order to correlate logs and detect a potential side-channel attack. In our experiments, we employed OSSIM [10], a well-known and open-source (with a commercial extension) SIEM system which is easily expandable with custom plugins (Figure 7).

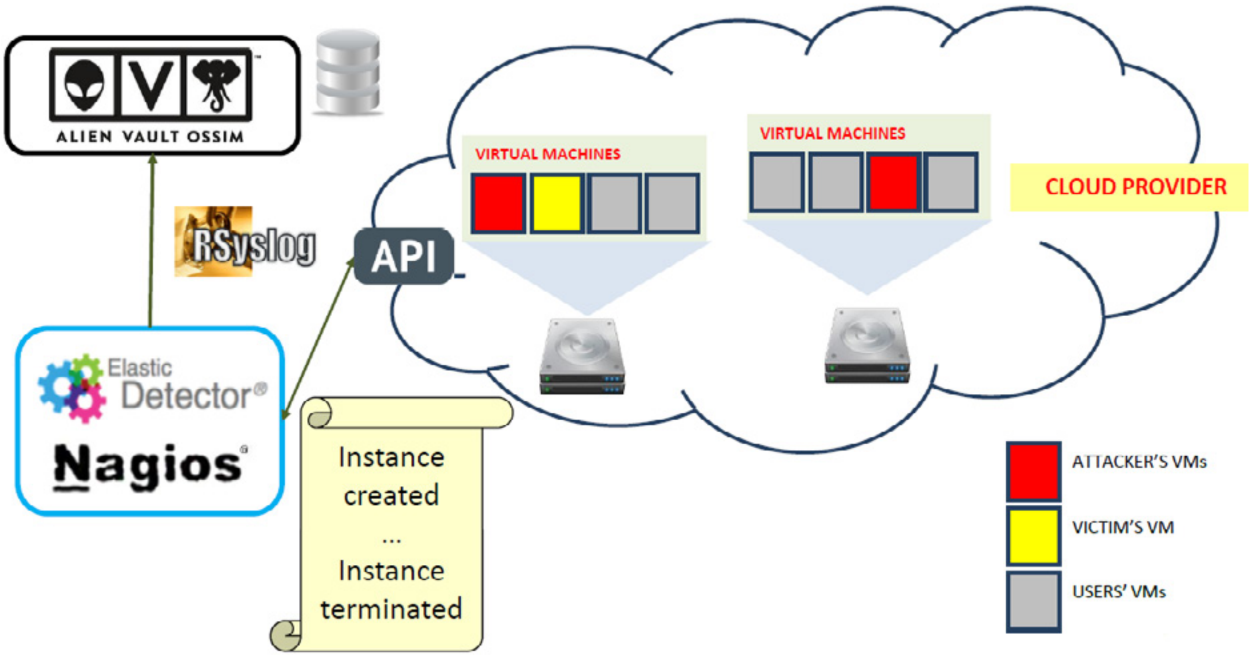


Figure 7. Architecture

Our strategy consist of forwarding all the security-relevant logs to OSSIM, where correlation can take place. In order to collect and forward logs, we used Elastic Detector [11], which is our flagship product. Thanks to Elastic Detector, we can smoothly and automatically detect changes (e.g. a virtual machine has been launched/stopped) in the cloud infrastructure and communicate these changes to OSSIM, by forwarding NAGIOS logs. Elastic Detector employs NAGIOS for performing automatic checks on the cloud infrastructure. This way, every change concerning the status of the user's virtual machines is automatically detected and logged. OSSIM does not provide any remote API, however, every OSSIM installation includes a RSyslog server, which can receive logs from remote machines. For this reason, the best way to forward our logs is to configure configure NAGIOS so it can send logs to a remote server, that is the server on which OSSIM is running.

Detailed Description

In order to test our solution, we developed a Python script which makes use of AWS standard APIs to emulate a side-channel attack. In particular, this script emulates the first phase of a side-channel attack, which is called "Placement". During this phase, the attacker launches a high number of virtual machines until he gets one virtual machine running on the same physical host of the victim. It is worth pointing out that this is not a simulation of the real scenario, this is exactly what an attacker would do.

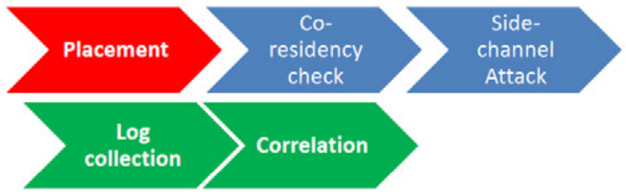


Figure 8. Basic algorithm

We think that our approach is the best one in terms of security, since the potential attacker is stopped before performing the side-channel attack. Indeed, while the attacker creates and destroys virtual machine, logs are collected, analyzed and correlated so the attack can be detected before it takes place.

Once logs have been delivered to OSSIM, thanks to our custom plugin, we can parse and convert them into security-relevant events. After that, we are interested in correlating two kinds of events, the creation and the termination of a virtual machine.

In order to parse logs generated by Elastic Detector, we needed to define a plugin and, as part of this plugin, a regular expression which allows us to extract the specific information we need from a given log line. Below we can see a typical log line generated when a virtual machine is launched and the regular expression used to capture the event and extract the meaningful information.

Log:

```
Aug 19 15:51:32 debian-secludit nagios3: SERVICE NOTIFICATION: event@551;72-us-east-1;722;notify-
service-by-cloutomate;Found new Instance: i-f0ad689c
```

Regular expression:

```
^(?P<date>\w{3}\s\d{1,2}\s\d\d:\d\d:\d\d)\sdebian-secludit\snagios3:\sSERVICE\sNOTIFICATION:\
sevent@\d{3}\;(?P<account>\d{2,3})-(?P<region>\w{2}-\w{4,9}-\d)\;\d{3}\;notify-service-by-
cloutomate\;Found\snew\sInstance:\s(?P<instanceid>i-[a-z,0-9]{8})$
```

where `72` is the account identifier of the user who launched the virtual machine, `us-east-1` is the region in which the virtual machine is running and `i-f0ad689c` is the virtual machine ID.

Thanks to these information, we can proceed to the next stage, that is correlating these events and determining if a user (the potential attacker) is performing a side-channel attack.

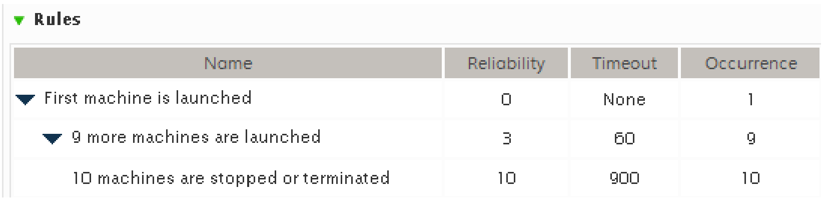
In Listing 1 we can have a closer look at the code we wrote for the plugin.

Listing 1. The code of the plugin

```
;; Elastic Detector
;; plugin_id: 9001
;; type: detector
;;

[DEFAULT]
plugin_id=9001

[config]
type=detector
enable=yes
source=log
location=/var/log/nagios3/nagios.log
create_file=false
process=
start=no
stop=no
startup=
shutdown=

[elastic-detector-found-new-instance]
#Aug 19 15:51:32 debian-secludit nagios3: SERVICE NOTIFICATION : event@551;72-us-east-
   1;722;UNKNOWN;notify-service-by-cloutomate;Found new Instance: i-f0ad698c
event_type=event
regexp=”^(?P<date>\w{3}\s\d{1,2}\s\d\d:d\d:\d\d)\sdebian-secludit\snagios3:\sSERVICE\
   sNOTIFICATION:\sevent@\d{3}\;(?P<account>\d{2,3})-(?P<region>\w{2}-\w{4,9}-\d)\;\
   d{3}\;UNKNOWN\;notify-service-by-cloutomate\;Found\snew\sInstance\:s(?P(instanceid>i-[a-z,0-9]
   {8})$"
date={normalize_date($date)}
#sensor={resolv($sensor)}
plugin_sid=1
#src_ip={$src}
userdata1={$region}
userdata2={$instanceid}
userdata3={$account}

[elastic-detector-instance-not-running]
#Aug 20 15:50:26 debian-secludit nagios3: SERVICE NOTIFICATION : event@551;72-us-east-
   1;722;UNKNOWN;notify-service-by-cloutomate;Instance Terminated: i-7824db12
event_type=event
regexp="^(?P<date>\w{3}\s\d{1,2}\s\d\d:d\d:\d\d)\sdebian-secludit\snagios3:\sSERVICE\
   sNOTIFICATION:\sevent@\d{3}\;(?P<account>\d{2,3})-(?P<region>\w{2}-\w{4,9}-\d)\;\
   d{3}\;UNKNOWN\;notify-service-by-cloutomate\;Instance\sTerminated:\s(?P(instanceid>i-[a-z,0-9]
   {8})$"
date={normalize_date($date)}
#sensor={resolv($sensor)}
plugin_sid=2
#src_ip={$src}
userdata1={$region}
userdata2={$instanceid}
userdata3={$account}
```

As we can see above, defining a plugin means to define a set of events and a set of regular expressions to convert a log line into an event.

Any SIEM system is based on correlation rules. In OSSIM, correlation rules define a set of conditions that, if met, can raise an alarm. In our experiment, we defined a simple and effective correlation rule which can be seen below.



Rules

Name	Reliability	Timeout	Occurrence
First machine is launched	0	None	1
9 more machines are launched	3	60	9
10 machines are stopped or terminated	10	900	10

Figure 9. Correlation rule

When a new virtual machine is created, OSSIM starts the evaluation of the correlation rule. If in the following 60 seconds, 9 more virtual machines are created in the same region, by the same user, then the evaluation proceeds to the next level. Finally, if within 15 minutes (approximation of the time required to create a virtual machine and perform a co-residency check) all those virtual machines are terminated, then an alarm is raised. At this stage, several countermeasures could be taken. For instance, a notification or an email could be sent to the security administrator or a program/script could be executed. As an example, a realistic reaction to such an alarm could be the temporary revocation of the user's account.

In our experiments, we successfully managed to detect several potential side-channel attacks during the placement phase. Potential attacks are detected as soon as virtual machines are terminated so that the proper countermeasures can be taken before the attacker can try to steal any information.

A comprehensive DEMO of the above-mentioned solution is available at *http://youtu.be/3NacQOksyJo*.

Discussion

Our proposed solution does not require any change in the way local information are collected. The main advantage of our approach is that the system administrator does not need anymore to re-configure the SIEM system every time there is a change in the public cloud infrastructure. Therefore, Elastic Detector is able to automatically detects changes in the cloud infrastructure and its configuration. This way, event information and logs can be collected according to the current configuration and settings. The only thing the administrator has to do is to provide his API credentials in order to enable Elastic Detector to connect to the cloud.

When a relevant event occurs or a log file needs to be forwarded to the SIEM system, Elastic Detector can seamlessly for the user communicate with it and deliver the required information. Once the SIEM system has successfully communicated with Elastic Detector, it can finally process the events occurred and perform the typical actions for SIEM: normalization, correlation, notification and alerting, prioritization, visualization and reporting.

Furthermore, our solution does not even need the collaboration of the hypervisor, so it is fully compatible with any existing platform. This is the main reason why we decided to detect side-channel attacks during the placement phase. Indeed, detecting a side-channel attack during the following phases would require the collaboration of other components of the infrastructure such as routers (co-residency check) and hypervisors (side-channel attack), resulting in a more complex integration.

The main benefits of our strategy are:

- Full Automation. Keeping operating costs under control means being able to automate the security management by eliminating the majority of manual setup, security monitoring, and corrective actions.
- Agentless. The performance footprint of agents on servers and potential conflicts with applications are sources of problems. Moreover, agents are OS dependent and have vulnerabilities as well. Through the virtualization layer, and using APIs such as VMware vShield or Amazon EC2 security groups, security solutions can analyze resource information and enforce security with no agents.
- Comprehensive Security Assessment. The traditional layered approach, where each security component takes care of a specific layer such as the network, is not enough. For this reason, there is a need for tools that tackle the new security challenges brought by IaaS, such as multi-tenancy and side-channel attacks.
- No Lock-in. In some scenarios, it is important to have the ability to use different IaaS offerings for reliability and flexibility. However, this should not compromise the effectiveness of security tools and the ability to have a full visibility of the security of your infrastructure.

Conclusion

In this article we have presented a new class of attacks, the so called side-channel attacks, and a prototype for protecting cloud users from these attacks. The main reason why we decided to focus on this use case is that nowadays there is no solution to detect side-channel attacks in the cloud environment.

Deploying our solution would be extremely easy and safe for a cloud provider since it should just allow Elastic Detector to access a very small subset of events (virtual machine creation/termination). Also, this solution would be completely transparent to the user.

Furthermore, we provided also a solid architecture for adapting existing SIEM systems in cloud infrastructures. In this particular case we have presented the integration of our main product, Elastic Detector, and OSSIM, an open-source and widely adopted solution for SIEM.

Cloud infrastructures have the advantage of being highly elastic. Unfortunately, their elasticity also brings several security concerns. Among these concerns, from a SIEM point of view, there is the necessity for the system administrator to constantly monitor the public cloud infrastructure in order to detect changes and re-configure the SIEM system accordingly. In most of the cases, this approach is unfeasible since configuring a SIEM system can be very complex and take a long time.

We strongly believe that our proposed solution can help system administrators and IT professionals to easily and safely manage their infrastructures without having the hassle of re-configuring their SIEM every time there is a change in the public cloud.

Elastic Detector is an optimal and smart solution to overcome this serious issue by delegating the monitoring activity of remote virtual machines to an external, automatic and elastic tool. In addition, deploying Elastic Detector does not require any change in the infrastructure and no software agent has to be installed on virtual machines.

References

[1] Cloud Security Alliance Top Threats *https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf*

[2] A Security Analysis of Amazon's Elastic Compute Cloud Service *http://secludit.com/wp-content/uploads/2012/09/securecloud.pdf*

[3] There's a Hole in 1,951 Amazon S3 Buckets *https://community.rapid7.com/community/infosec/blog/2013/02/14/1951-open-s3-buckets*

[4] OSSIM *http://communities.alienvault.com/*

[5] Amazon Virtual Private Cloud *http://aws.amazon.com/vpc/*

[6] Announcing Amazon Elastic Compute Cloud (Amazon EC2) – beta *http://aws.amazon.com/about-aws/whats--new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/*

[7] Gartner Top 10 Strategic Technology Trends for 2013 *http://apmdigest.com/gartner-top-10-strategic-technology--trends-for-2013-big-data-cloud-analytics-and-mobile*

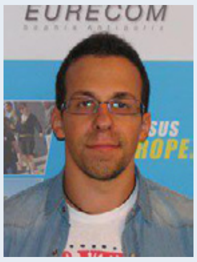
[8] 1 Cross-VM Side Channels and Their Use to Extract Private Keys Y Zhang, A Juels, MK Reiter, T Ristenpart *https://mexico.rsa.com/rsalabs/presentations/cross-vm-side-channels.pdf*

[9] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. 2009. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In Proceedings of the 16th ACM conference on Computer and communications security (CCS '09). ACM, New York, NY, USA, 199-212. DOI=10.1145/1653662.1653687 *http://doi.acm.org/10.1145/1653662.1653687*

[10] OSSIM *http://www.alienvault.com/open-threat-exchange*

[11] Elastic Detector, SecludIT. *http://elastic-detector.secludit.com*

About the Author



Pasquale Puzio is a CIFRE PhD Student at SecludIT and EURECOM, under the supervision of Sergio Loureiro and Refik Molva. He got a Master's Degree in Computer Science from University of Bologna and a Master's Degree in Ubiquitous Computing from University of Nice-Sophia Antipolis. The topic of his PhD thesis is Data Storage Security in Cloud Computing but his research interests include also infrastructure security in cloud computing.

About the Author

Sergio Loureiro, CEO and Co-Founder has worked in network security for more than 15 years. He has occupied top management positions in 2 startups where he was responsible for email security products and services, and security gateways. Before he was the lead architect for a number of security products such as SSL VPNs, log management, web security and SSL crypto accelerators. His career started in several research labs, where he participated in European projects focusing on security. Sergio holds a Ph.D. in computer science from the ENST Paris and MSc and BSc degrees from the University of Porto (Portugal). He is the holder of 3 patents.

Content-Based Intrusion Detection System

by Mark Sitkowski

Nobody ever broke into a bank's IT system by cracking a user's password. It's not cost-effective to waste computer time on such a pursuit, for the sake of the few thousand dollars that may, or may not be in the user's account.

It's far more cost-effective to persuade the bank to let you have access to its database, via a back door. Then, you have access to all of the bank's resources, for the expenditure of a minimum of effort, and without even having to understand how the authentication system works.

On the other side of fence, when your company's product actually is that bank's authentication system, and which it describes as 'Uncrackable', you have to expect this to be like a red rag to a bull, as far as the world's hackers are concerned.

Every day, dozens of them try to break the algorithm, but none ever succeed, so there is some excuse for the complacency which ensues. However, you soon notice that, for every front door attack, there are over a hundred attempts to totally bypass the authentication system, and get in via a back door.

Now, after you've told the world that the authentication system is uncrackable, it would be rather embarrassing to find that the hackers had decided not to bother cracking it, but had broken into your authentication server, instead, and hijacked your database.

You have no control over how the average bank, securities trading company or whoever uses your product, configures their online access server or ATM machine, but you can lead by example, and make sure that your authentication server, at least, can be made hack-proof.

Easy, right? All you need to do is to buy a device which will alert you, as soon as it detects a hack attempt, and prevent it succeeding.

If, after a few weeks of searching on the internet, and talking to prospective suppliers, you find that nothing on the market will do what you want, what do you do?

You write your own, of course...

Defining the problem

When we set up the infrastructure for our authentication server's website, we did all the right things.

The only open port was port 80, there was no GET permission for cgi-bin, no POST permission for htdocs, all other methods like MOVE, DELETE, COPY etc were disabled, and there were no interpreted scripts, like those written in java, perl, shell or ruby.

The only HTML page was index.html, and the other sixty four pages were dynamically created by the CGI – which was an executable, written in a compiled language. That way, if a hacker ran Wget on our site, he'd have no additional clues as to which page called which CGI, or what any of the HTML variables meant.

Bulletproof.

As far as it went, it certainly was. We had many connections each day, from the usual hopeful hackers, who would try to get in by breaking the authentication algorithm, and from the old-timers and incompetents, who would try buffer overflow, not having heard that that particular method didn't work on modern network applications.

Then, after a few months, things changed, as dozens of more determined hackers, with no life of their own, decided that they could combine distributed denial of service attacks with hack attempts. We were inundated with hundreds of queries, each designed to plant or exploit back doors, inject SQL or exploit vulnerabilities in every file whose name ended in '.php'.

We don't use WordPress, cPanel, Joomla, ccmail or any of the other traditionally exploited software packages, so we were immune to all of these attacks, but it was extremely annoying to watch the server logs scrolling like a Las Vegas slot machine, as every unimaginative hack script repeated the same dumb vector anything from two to four hundred times.

Also, it was eating up our network bandwidth, and making the site respond less quickly than we would have liked, and giving perverted pleasure to some hacker, who was watching hundreds of lines of hack script execute.

The last straw came, one day, when we were hit with a DDOS from an address in the Netherlands. It started about 4am, and continued till 11am, during which time the hacker had thrown over twenty thousand vectors at us, at which point, I manually added a firewall rule to block his IP address.

The hacker continued to bang his head against the firewall till around lunch time, on every port from 1024 to 32767, and then gave up. The only positive outcome of this was that, during the attack, all of the other hackers were blocked by the limited remaining bandwidth.

It was obvious that something positive had to be done to stop this nonsense.

We decided to find an intrusion detection system which, everyone agreed, would solve our problem, and made a list of the functions we wanted it to perform.

First, it had to be content-based, so it could identify a hack attempt by the kind of thing the query was trying to do, which implied that such a system would need a certain amount of intelligence.

Second, having identified the hack, it would need to remember the IP address, drop the connection, and make sure that that IP address would never again be allowed to connect to our site.

Last, it would need to do all this in less than one second. The attacks that we faced were not directed from Mum and Dad's Wintel PC, but from high-end Unix servers in data centres. Having seen the speed at which our log monitor scrolled up the screen when we were under attack, we then examined the access log, and noticed that the average zombie hijacked server could shower us with hack vectors at a minimum rate of two or three a second and, sometimes, if they'd hacked a decent machine, up to ten a second.

Our goal was to stop it after the first vector.

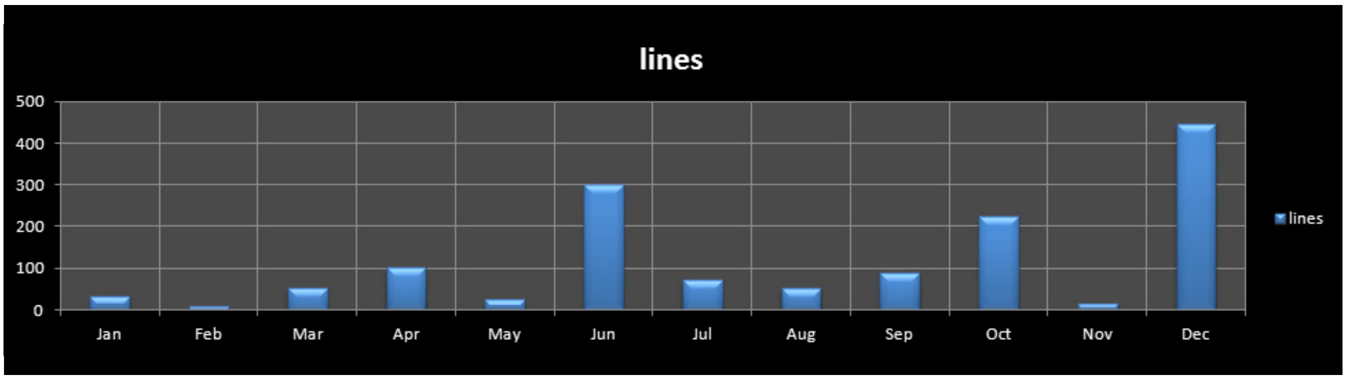


Figure 1. Before deployment of IDS

The search for the product

What we expected was, that we would make a quick list of suitable products, then spend a long period of decision-making, choosing from many suitable candidates. This turned out to be a huge disappointment.

We noticed from the first day, that the vast majority of intrusion detection systems were really no more than fancy java, shell and perl scripts, with a response time similar to that of a whale trying to turn itself around.

Disillusioned with the (not so) cheap end of the market, we decided that something used by banks had to be of the right quality, so we took a look at the professional, so-called 'enterprise level' products.

While researching this kind of product online, the whole thing got off to an unpromising start, when I read the comments of a security consultant to a bank, describing the product they used. During his speech, he declared proudly, that they would be aware of an intrusion within forty-eight hours of its happening.

Forty-eight hours? To us, forty-eight seconds would be too long, never mind forty-eight hours.

Predictably enough, the search of the high end of the market showed that shell scripts could be available at high prices, too.

Worse, most of this stuff only ran on Windows, and we're a Sun Solaris shop. Who, in his right mind, would run a website on Windows?

During the demo of one of these products, the salesman explained that his system took its data via a network connection to the actual web server machine, and it had this absolutely mind-blowing graphical display of how your website was being hacked, minute by minute. This was impressive, and a lot easier than watching lines of text scrolling up the screen.

We asked how it worked, and were told that it counted the number of queries received in a given period and, if that exceeded a given value (which we could preset, of course) it flashed a lot of lights on the panel, and sounded an important alarm bell. Yes, but how did it differentiate between a legitimate connection, which just happened to be from a particularly fast machine, and a hack script? Well, it didn't, but the final decision would be up to its operator. Did that mean that it didn't automatically cut off the incoming connection? That's correct. The system administrator would have to do that.

The salesman explained, rather frostily, that what we wanted was an intrusion protection system, not an intrusion detection system.

Since his product was totally unaware of the content of each query, we rejected it, and took a look at another, which claimed to be content-aware. This was more promising, since it was possible to pre-program the thing with a selection from a set of internally stored, popular hack strings, and have it do the usual light flashing and frantic beeping when it discovered something interesting.

Although it ran on Linux, and a source code licence was available (at an additional cost), so that we could recompile it to run on Solaris, it, too, relied on the system administrator to do something about the hacker. Furthermore, there was no provision for adding new hack strings to the list hard-coded inside it.

Further questioning revealed that the thing ran like a packet sniffer, and reassembled each packet's payload to figure out the query string. This procedure resulted in many false positives, and false negatives, and made its response time less than breathtaking.

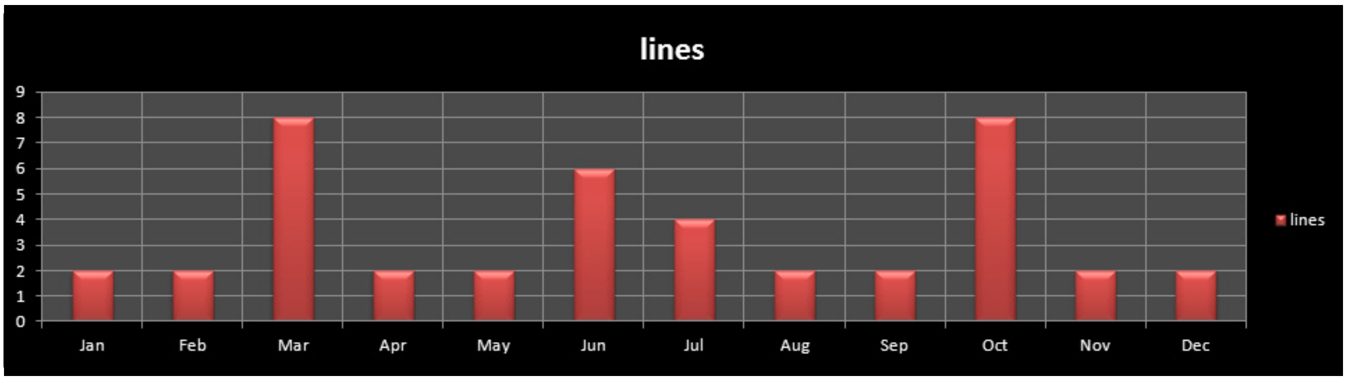


Figure 2. After deployment of IDS

The only product which, apparently, did what we wanted was a proxy. Filled with new enthusiasm, we took a cautious look at a few proxy offerings, only to be further disappointed.

Although a proxy really could do content-based filtering, accessing our web pages through it proved to be virtually impossible. Also, the degree of remote control available was strictly limited, to the point of being unusable.

So, there it was. The market was willing to sell us a few Linux offerings, a huge number of Windows ornaments, but nothing that would examine the content of what was trying to get into our website, and automatically drop the connection, if it saw something it didn't like.

The solution

During the time we were talking to the representatives of the various intrusion detection companies, I was thinking about the various issues which surrounded this problem.

Firstly, we absolutely needed to know the content of each query. However, there was no way that we would accomplish this reliably, by tracking text strings across several hundred packets, and then reassembling the original query. The packet stream contained too much information, some of which was irrelevant, and identifying the query with any degree of certainty was too difficult.

What we needed was a pre-assembled query, which was guaranteed to be a query.

I was sitting in front of the apache access log monitor, in the middle of a botnet attack, watching it scroll enthusiastically up the screen, when it occurred to me, that what happened at packet level was totally irrelevant. Bad things would only happen at the time when apache had the whole query in its buffer, and was about to act on it. Therefore, if we read the access log as it was being created, we would only be one line behind apache.

The hack queries themselves didn't bother us, since they attempted to exploit vulnerabilities in software we didn't use, so allowing one to slip through would be of no consequence.

So, the only criterion was to identify the first in a series of malicious queries from a given address, and do something before the hacker could send a second query.

Now the question arose, as to what constituted a malicious hack?

So, what is a hack vector?

We filtered our access log, and removed all queries which accessed our legitimate web pages and CGI executables. What remained, according to Sherlock Holmes, had to be the truth.

A lengthy and detailed examination of the logs showed a rich selection of attempted hacks.

One that was extremely prolific, was a GET followed by a series of '../../..' of varying lengths, terminating in some significant filename, like /etc/passwd. This would have to be the first on our list, since so many hackers, with no lnowledge of Unix, thought it had some chance of succeeding.

Next, we noticed blocks of up to a thousand hexadecimal characters, each preceded by a percent sign. Decoding these, revealed that they were either IP addresses, or filenames, which some incompetent hacker assumed would slip past the casual observer. This hack's secondary function was an attempted buffer overflow, caused by its sheer length. A definite second choice for blocking.

Almost identical in purpose, was a similar hack, but with the percent sign replaced with '\x'. However, the hexadecimal values weren't ASCII.
This was a puzzle, which took a lot of research, until I recognized one of the hexadecimal values as being the Intel processor opcode for 'CALL subroutine'. Hackers call these things shellcodes, and the

intent is to execute a buffer overflow, so they can place Intel machine code in the system's RAM, take over the CPU's program counter, index it to point to their own code, and execute anything they like on your machine. For any machine running on an Intel CPU, this would be the kiss of death.

With so many '\x' characters, this hack was easy to identify, so we added it to the list.

Then, there was the embedded question mark, usually followed by what looked like a script of some kind. and the embedded exclamation mark, usually in the middle of a lot of different hexadecimal stuff, which was obviously up to no good.

There were also the SQL injection hack attempts. I guess the most original, was one which attempted to overflow the CAPTCHA buffer (which we didn't use) with a script like this:

```
"captcha/img.php?code=1'%20AND%201=0%20UNION%20SELECT%20SUBSTRING(CONCAT(login,0x3a,passwo
rd),1,7)%20FROM%20User%20WHERE%20User_id="
```

We decided against wasting computing time on these, since our other criteria, such as the string '.php' and the percent signs, would easily identify it. Finally, there were quite a few hacks containing an embedded series of plus signs, usually accompanied by a string of hexadecimal, or plain text like 'Result:+no+post+sending+forms+are+found'. Just for the sake of complete coverage, we added a line of code to reject these.

Collating all of the information revealed something even more interesting. It became obvious that approximately ninety percent of all hack attempts of all kinds were aimed at dozens of different PHP files.

The attacks varied from simple GET queries and POST queries, to a pattern, where an initial query would attempt to GET a file like index.php (presumably, to establish its existence) and be followed by a second query, which would try to POST to the same file, and overwrite it with a back door. Then, a third query would try another GET.

In the light of these observations, we decided that another primary candidate for blocking would be any query containing the string '.php'.

Command and Control

What happens once the malware is installed on your computer?

Since Unix, unlike Windows, doesn't permit self-executing executables, the hacker needs to access his malware after it has been installed.

How is he going to do so? Any self-respective server will have all ports closed except port 80, and believe itself to be totally impregnable. Unfortunately, this is not the case, since it is through port 80 that the C&C will wake up and direct the malware.

Almost all security devices concentrate on monitoring and defending TCP traffic through port 80. The C&C, on the other hand, talks to the malware using the UDP protocol, also through port 80, and is invisible to apache, and to many security systems.

It's perfectly reasonable to block UDP traffic, with few resulting issues. However, just to complicate matters, there are other services, which run on UDP. DNS queries and replies, the Unix XDMCP login, and time server data are just a few examples. Any firewall rule which blocks UDP traffic, has to exclude these.

Dropping the connection

When we reached this point in the investigation, I could almost write the code for the content analyzer in my head, and it was beginning to look more and more possible that we could write our own intrusion detection system. Then, I thought about the tricky part: dropping the connection.

The first thing to come to mind was a utility called tcpkill, which will very nicely drop an established TCP connection. However, a moment's reflection showed that this would be inadequate. The average hack script re-sent the same line anything up to four hundred times and, if we invoked tcpkill every time, not only would the network traffic be no lighter, but the CPU would chase its tail trying to keep up with the repeated hack attempts as well, especially when handling an attack from a few dozen servers simultaneously.

The next thought was that we would use the firewall.

Since we expected that our IDS would be a stand-alone process, it would be necessary to use a firewall which was remotely programmable. Almost every supplier that we contacted claimed to have such a device, so things were looking very promising.

Unfortunately, firewalls are very security-conscious animals, and the only way to remotely program them, is to login to them first. The procedure for doing this was either through a gee-whiz graphical user interface, or via a telnet or SSH TCP connection. The GUI was obviously unacceptable, so we wrote piece of code which established a telnet connection to the firewall and sent it a new rule. Ten seconds later, it was back on line.

Most firewalls contain a minimal Linux computer, and every time a new rule is added, this computer is rebooted. Even though ten seconds is a very short time for a reboot, it was just too long for our purposes.

Apart from the huge delay to add another rule, during that ten seconds, the machine would be sitting there with open arms, welcoming all hackers to do their worst, since the firewall was resetting itself, and totally inoperative. Further, that ten seconds would allow several hundred new hack attempts to queue up for processing, resulting in a never-ending shuffle between our content analyzer and the firewall.

Firewalls were abandoned, and we turned our attention to the Unix operating system.

Solaris has am extremely powerful utility, called 'ipf', which is a version of the 'ipfilter' module, which dates back to SunOS 4.1.3, in the good old BSD days.

It has all of the facilities available in stand-alone firewalls, such as NAT, but the filtering is actually performed in the Unix kernel, making it extremely efficient. It gets its rule set from a file, which is a minor drawback, but I decided to try it, anyway.

I wrote another piece of code, which appended a new firewall rule to the file, then told ipf to re-read the file and restart. We ran a few tests, and found that the time delay was almost immeasurable.

This is actually not that surprising. Since the filtering is done in the kernel, there is no actual ipf process. When a user issues a command to re-read the configuration file, the kernel activates a 'read' system call, which is internal to itself, so there isn't even a separate process to re-spawn. The only delay, is the time taken to execute the disk I/O – which is always a high priority task, since the kernel knows it takes a long time.

We decided against including any facility to count the number of queries in a given time interval. If the purpose was to identify a DDOS, then the hardware firewall could adequately cope with it. Also, this would mean repeatedly stopping other processing for the duration of that time interval. This could add an order of magnitude to the response time.

The complete system

We now had all of the building blocks for a complete intrusion detection – or, more accurately, intrusion protection system.

On startup, the IDS would read the ipf configuration file, and store all of the rules in an array of data structures. This would put the IDS in sync with the firewall, which was necessary, so that we didn't try to add a rule for an IP address which was already being blocked.

Next, we called a function which opened the apache access log, and performed a seek to the last line in the file. Having done that, it entered an endless loop, and waited for another line to be added to the file.

The loop contained the code of the content analyser, and had no time delays or pauses built in, so it would execute as fast as the CPU could execute machine code. This is usually extremely bad practice, since it uses 100% of the CPU's processing power. In our case, it didn't matter, since our machine had 32 CPU's, and devoting one of them to the IDS was a good investment.

As soon as apache logged another query, the content analyzer would scan it to see if it contained any of the hack signatures which we had built into it. If a hack attempt was identified, a firewall rule would be automatically created, to make comparison with the stored firewall rules easier, then another function would be called, to see if that IP address was already being blocked.

If this turned out to be a new hack attempt, the ipf configuration file would be opened, the new rule appended, and ipf re-invoked so it could re-read the file. Having performed the most important operations, the IDS would then add the new rule to its internal store.

There is a great temptation, when designing a system like this, to use multi-threading, or parallel processing. Although this would have considerably speeded up part of the processing cycle, the dangers of collision, between threads or processes, in areas such as file reading or writing was too great. Semaphores and mutexes are traditionally used to obviate such problems but, in general, if you need to use a mutex, you're either doing it wrong, or you shouldn't be multi-threading.

Conclusion

After a short period of debugging, the IDS was commissioned, and we monitored its progress over the first week, or so.

The performance was even better than we expected, and there were no false negatives. Anything that was supposed to be stopped, was stopped dead, after apache received just one illegal query.

However, there were a number of false positives.

We examined the logs, and found that some query strings, especially those which were links from some online magazines, and some social media sites contained elements containing the string '.php'. This was enough to trip the content analyzer, and have the IP address blocked.

There were so few of these false positives, that we were willing to write this off as acceptable collateral damage, when compared with the enormous benefit of limiting each hacker to one hack attempt per lifetime. With the possible exception of LinkedIn, the social media sites were unlikely to bring us any significant business, but some of the online magazines were important. Accordingly, we added a few lines of code into the loop, which would cause it to ignore any positives containing the names of chosen sites.

So far, the IDS has been running continuously for over two years, with no modifications, apart from the periodic addition of new rules, as new hacks are discovered.

If this were a commercial product, we would probably have it read a configuration file on startup, instead of having the hacks and exceptions hard-coded. However, since it isn't, we don't mind recompiling it each time there's an update. It keeps it more secure.

About the Author

Design Simulation Systems Ltd
http://www.designsim.com.au
Consultant to Forticom Security
http://www.forticom.com.au

Bridging the gap between business & technology

Ranked among the largest minority-owned IT services firms in the U.S. having Global Delivery capabilities

Operations in the U.S., U.K., India, Singapore and Philippines with over 9000 professionals

Leading mid-tier IT vendor with end-to-end IT capabilities spanning Technology Consulting, Application Outsourcing and Infrastructure Services

Recognized on numerous occasions by GS 100, The International Association for Outsourcing Professionals, FinTech 100 & InformationWeek

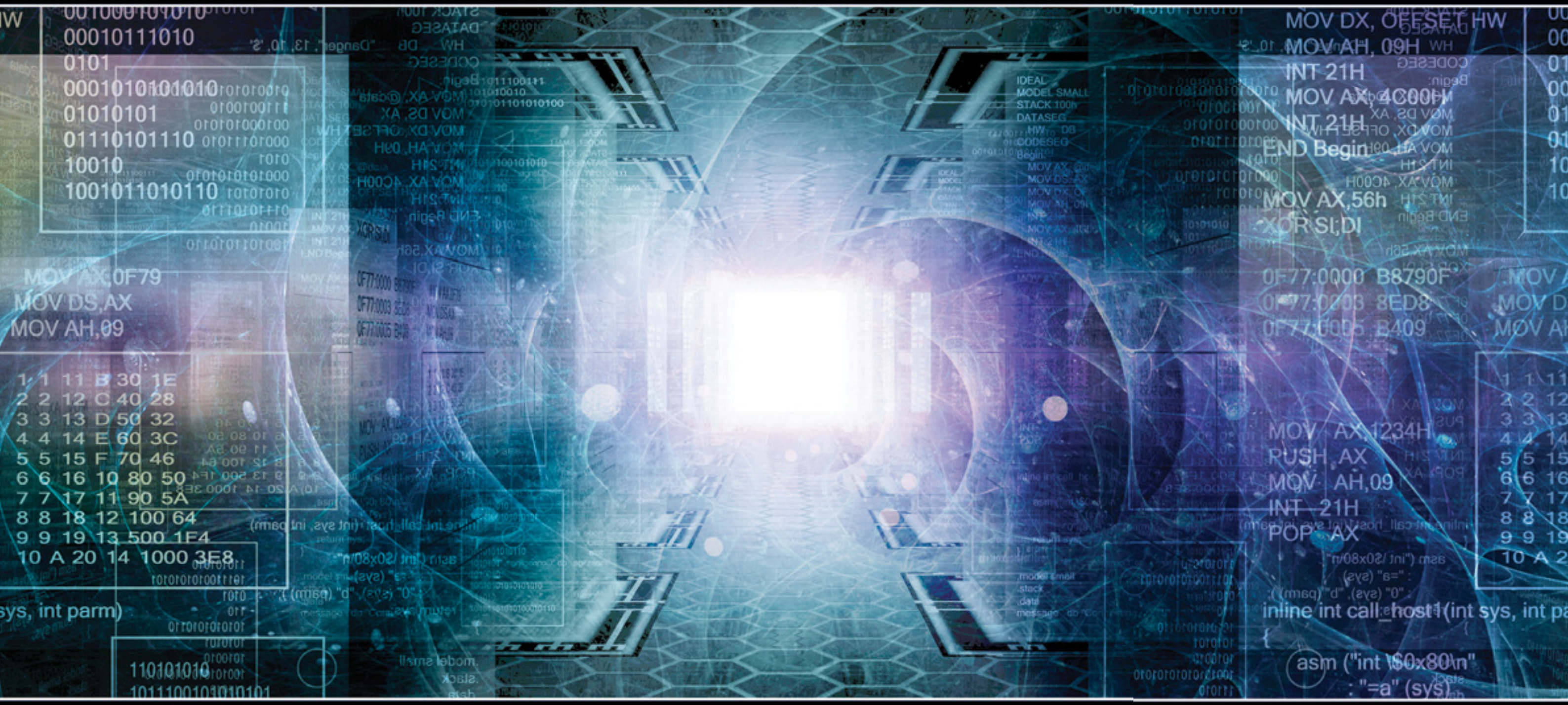
Collabera

www.collabera.com

ADVANCED TARGETED ATTACKS

HAVE PENETRATED **95%** OF ALL NETWORKS*.

THINK YOU'RE IN THE 5%?



You may think your existing security defenses prevent advanced targeted attacks from entering your network and stealing your data. They don't. Advanced attacks easily evade traditional and next generation firewalls, IPS, AV and gateways. Your best defense is **FireEye**. Trusted by the Fortune 500, and over 60 government agencies globally, FireEye is the leader in helping organizations combat advanced malware and targeted APT attacks.

Put a stop to advanced attacks with advanced security. Visit us today at **www.FireEye.com/StopAPTs** and let us help you close the hole in your network.

FireEye™

*Based on FireEye end-user data