

Universidad Nacional Autónoma de México
Facultad de Ciencias

Asignatura: Redes de computadoras
Semestre: 2024-1

Profesor: Javier León Cotonieto

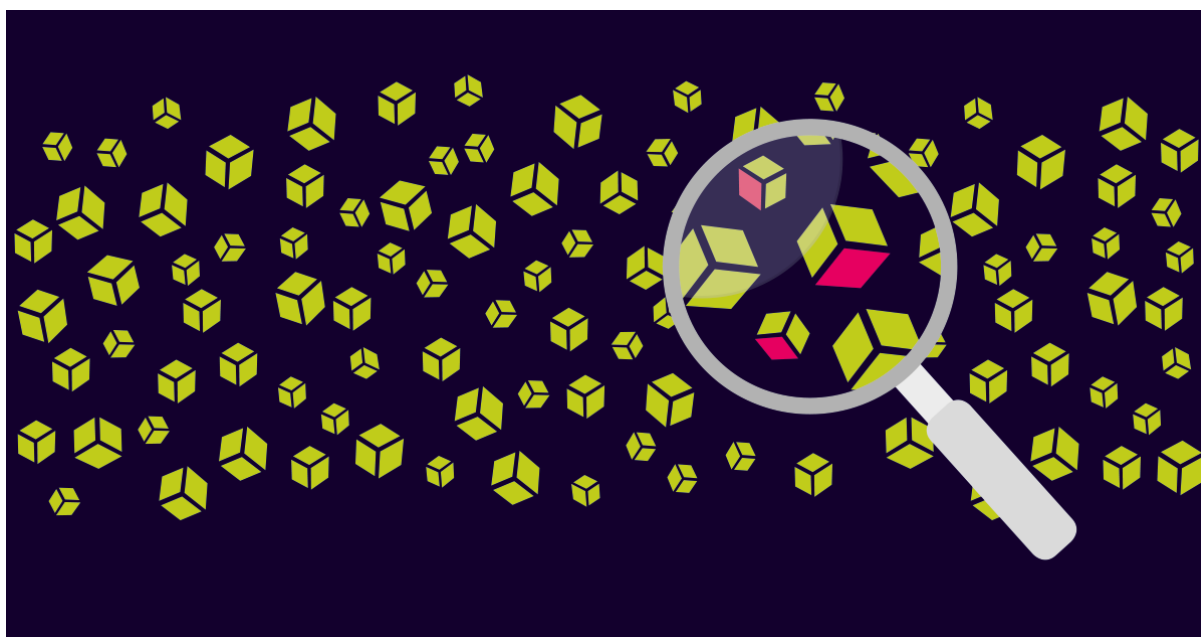
Ayudantes: Magdalena Reyes Granados
Itzel Gómez Muñoz
Sandra Plata Velázquez

Práctica 5. Captura y análisis de paquetes (tcpdump y Wireshark).

Equipo 5

Integrantes:

- **Almanza Torres José Luis**
- **Jimenez Reyes Abraham**
- **Martínez Pardo Esaú**



TCPDUMP

Instalamos tcpdump

```
atjl_jra_mpe@equipo5: ~  
atjl_jra_mpe@equipo5:~$ sudo apt-get install tcpdump  
[sudo] contraseña para atjl_jra_mpe:  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
tcpdump ya está en su versión más reciente (4.99.1-3ubuntu0.1).  
fijado tcpdump como instalado manualmente.  
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 39 no actualizados.
```

Verificamos su instalación

```
atjl_jra_mpe@equipo5:~$ tcpdump --version  
tcpdump version 4.99.1  
libpcap version 1.10.1 (with TPACKET_V3)  
OpenSSL 3.0.2 15 Mar 2022
```

Ingresamos como root

```
atjl_jra_mpe@equipo5:~$ sudo -i  
[sudo] contraseña para atjl_jra_mpe:  
root@equipo5:~#
```

Vemos la lista de interfaces de red disponibles en el sistema y en las que TCPDUMP puede capturar paquetes

```
root@equipo5:~# tcpdump -D  
1.enp0s3 [Up, Running, Connected]  
2.any (Pseudo-device that captures on all interfaces) [Up, Running]  
3.lo [Up, Running, Loopback]  
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]  
5.nflog (Linux netfilter log (NFLOG) interface) [none]  
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]  
7.dbus-system (D-Bus system bus) [none]  
8.dbus-session (D-Bus session bus) [none]  
root@equipo5:~#
```

Para capturar los paquetes que fluyen a través de una interfaz en específico, se utiliza la opción “-i”, seguido del nombre de la interfaz

```
root@equipo5:~# tcpdump -i enp0s3
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:36:44.876171 IP equipo5.35068 > 149.154.174.100.https: Flags [P.], seq 94373732:94373931, ack 3183
7570, win 65535, length 199
16:36:44.877247 IP 149.154.174.100.https > equipo5.35068: Flags [.], ack 199, win 65535, length 0
16:36:44.910865 IP equipo5.34395 > 192.168.100.1.domain: 57383+ [1au] PTR? 100.174.154.149.in-addr.ar
pa. (57)
16:36:44.922490 IP 192.168.100.1.domain > equipo5.34395: 57383 NXDomain 0/1/1 (150)
16:36:44.922773 IP equipo5.34395 > 192.168.100.1.domain: 57383+ PTR? 100.174.154.149.in-addr.arpa. (4
6)
16:36:44.934082 IP 192.168.100.1.domain > equipo5.34395: 57383 NXDomain 0/1/0 (139)
16:36:44.935407 IP equipo5.41729 > 192.168.100.1.domain: 57517+ [1au] PTR? 15.2.0.10.in-addr.arpa. (5
1)
16:36:44.937192 IP 149.154.174.100.https > equipo5.35068: Flags [P.], seq 1:114, ack 199, win 65535,
length 113
16:36:44.937240 IP equipo5.35068 > 149.154.174.100.https: Flags [.], ack 114, win 65535, length 0
16:36:44.946958 IP 192.168.100.1.domain > equipo5.41729: 57517 NXDomain 0/1/1 (128)
16:36:44.947227 IP equipo5.41729 > 192.168.100.1.domain: 57517+ PTR? 15.2.0.10.in-addr.arpa. (40)
16:36:44.959251 IP 192.168.100.1.domain > equipo5.41729: 57517 NXDomain 0/1/0 (117)
16:36:45.014178 IP equipo5.56587 > 192.168.100.1.domain: 21366+ [1au] PTR? 1.100.168.192.in-addr.arpa
. (55)
16:36:45.027331 IP 192.168.100.1.domain > equipo5.56587: 21366 NXDomain 0/1/1 (132)
16:36:45.027713 IP equipo5.56587 > 192.168.100.1.domain: 21366+ PTR? 1.100.168.192.in-addr.arpa. (44)
16:36:45.038640 IP 192.168.100.1.domain > equipo5.56587: 21366 NXDomain 0/1/0 (121)
^C
16 packets captured
16 packets received by filter
0 packets dropped by kernel
root@equipo5:~#
```

Sin la opción -i tcpdump seleccionará la primera interfaz de red que encuentra.

```
root@equipo5:~# tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:39:49.018844 IP equipo5.35068 > 149.154.174.100.https: Flags [P.], seq 94390056:94390239, ack 3184
9793, win 65535, length 183
16:39:49.019633 IP 149.154.174.100.https > equipo5.35068: Flags [.], ack 183, win 65535, length 0
16:39:49.031209 IP equipo5.57635 > 192.168.100.1.domain: 50279+ [1au] PTR? 100.174.154.149.in-addr.ar
pa. (57)
16:39:49.046570 IP 192.168.100.1.domain > equipo5.57635: 50279 NXDomain 0/1/1 (150)
16:39:49.046848 IP equipo5.57635 > 192.168.100.1.domain: 50279+ PTR? 100.174.154.149.in-addr.arpa. (4
6)
16:39:49.057238 IP 192.168.100.1.domain > equipo5.57635: 50279 NXDomain 0/1/0 (139)
16:39:49.058579 IP equipo5.48717 > 192.168.100.1.domain: 43208+ [1au] PTR? 15.2.0.10.in-addr.arpa. (5
1)
16:39:49.069867 IP 192.168.100.1.domain > equipo5.48717: 43208 NXDomain 0/1/1 (128)
16:39:49.070122 IP equipo5.48717 > 192.168.100.1.domain: 43208+ PTR? 15.2.0.10.in-addr.arpa. (40)
16:39:49.080510 IP 192.168.100.1.domain > equipo5.48717: 43208 NXDomain 0/1/0 (117)
16:39:49.105884 IP 149.154.174.100.https > equipo5.35068: Flags [P.], seq 1:114, ack 183, win 65535,
length 113
16:39:49.105936 IP equipo5.35068 > 149.154.174.100.https: Flags [.], ack 114, win 65535, length 0
16:39:49.140950 IP equipo5.39502 > 192.168.100.1.domain: 28603+ [1au] PTR? 1.100.168.192.in-addr.arpa
. (55)
16:39:49.152459 IP 192.168.100.1.domain > equipo5.39502: 28603 NXDomain 0/1/1 (132)
16:39:49.152887 IP equipo5.39502 > 192.168.100.1.domain: 28603+ PTR? 1.100.168.192.in-addr.arpa. (44)
16:39:49.163771 IP 192.168.100.1.domain > equipo5.39502: 28603 NXDomain 0/1/0 (121)
^C
16 packets captured
16 packets received by filter
0 packets dropped by kernel
root@equipo5:~#
```

Investigue y ejemplifique que hacen las opciones -v y -vv

-v

Al analizar e imprimir, produzca (un poco más) una salida detallada. Por ejemplo, se imprimen el tiempo de vida, la identificación, la duración total y las opciones en un paquete IP. También permite verificaciones adicionales de integridad de paquetes, como verificar la suma de verificación del encabezado IP y ICMP.

```
root@equipo5:~# tcpdump -v
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:44:08.407568 IP (tos 0x0, ttl 1, id 24692, offset 0, flags [DF], proto UDP (17), length 201)
  equipo5.33761 > 239.255.255.250.1900: UDP, length 173
16:44:08.455054 IP (tos 0x0, ttl 64, id 40248, offset 0, flags [none], proto UDP (17), length 85)
  equipo5.56471 > 192.168.100.1.domain: 52433+ [1au] PTR? 250.255.255.239.in-addr.arpa. (57)
16:44:08.477942 IP (tos 0x0, ttl 64, id 14970, offset 0, flags [none], proto UDP (17), length 142)
  192.168.100.1.domain > equipo5.56471: 52433 NXDomain 0/1/1 (114)
16:44:08.478203 IP (tos 0x0, ttl 64, id 40249, offset 0, flags [none], proto UDP (17), length 74)
  equipo5.56471 > 192.168.100.1.domain: 52433+ PTR? 250.255.255.239.in-addr.arpa. (46)
16:44:08.488647 IP (tos 0x0, ttl 64, id 14971, offset 0, flags [none], proto UDP (17), length 131)
  192.168.100.1.domain > equipo5.56471: 52433 NXDomain 0/1/0 (103)
16:44:08.489781 IP (tos 0x0, ttl 64, id 17406, offset 0, flags [none], proto UDP (17), length 79)
  equipo5.45489 > 192.168.100.1.domain: 34049+ [1au] PTR? 15.2.0.10.in-addr.arpa. (51)
16:44:08.502194 IP (tos 0x0, ttl 64, id 14972, offset 0, flags [none], proto UDP (17), length 156)
  192.168.100.1.domain > equipo5.45489: 34049 NXDomain 0/1/1 (128)
16:44:08.502529 IP (tos 0x0, ttl 64, id 17407, offset 0, flags [none], proto UDP (17), length 68)
  equipo5.45489 > 192.168.100.1.domain: 34049+ PTR? 15.2.0.10.in-addr.arpa. (40)
16:44:08.513456 IP (tos 0x0, ttl 64, id 14973, offset 0, flags [none], proto UDP (17), length 145)
  192.168.100.1.domain > equipo5.45489: 34049 NXDomain 0/1/0 (117)
16:44:08.561817 IP (tos 0x0, ttl 64, id 47731, offset 0, flags [none], proto UDP (17), length 83)
  equipo5.41617 > 192.168.100.1.domain: 54484+ [1au] PTR? 1.100.168.192.in-addr.arpa. (55)
16:44:08.573294 IP (tos 0x0, ttl 64, id 14974, offset 0, flags [none], proto UDP (17), length 160)
  192.168.100.1.domain > equipo5.41617: 54484 NXDomain 0/1/1 (132)
16:44:08.573460 IP (tos 0x0, ttl 64, id 47732, offset 0, flags [none], proto UDP (17), length 72)
  equipo5.41617 > 192.168.100.1.domain: 54484+ PTR? 1.100.168.192.in-addr.arpa. (44)
16:44:08.584591 IP (tos 0x0, ttl 64, id 14975, offset 0, flags [none], proto UDP (17), length 149)
  192.168.100.1.domain > equipo5.41617: 54484 NXDomain 0/1/0 (121)
^C
13 packets captured
13 packets received by filter
0 packets dropped by kernel
```

-vv

Salida aún más detallada. Por ejemplo, se imprimen campos adicionales a partir de paquetes de respuesta NFS y los paquetes SMB se decodifican por completo.

```
root@equipo5:~# tcpdump -vv
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:46:43.732548 IP (tos 0x0, ttl 64, id 15877, offset 0, flags [none], proto UDP (17), length 107)
  tzgroa-ab-in-f10.1e100.net.https > equipo5.50829: [udp sum ok] UDP, length 79
16:46:43.732809 IP (tos 0x0, ttl 64, id 15878, offset 0, flags [none], proto UDP (17), length 107)
  tzgroa-ab-in-f10.1e100.net.https > equipo5.50829: [udp sum ok] UDP, length 79
16:46:43.733241 IP (tos 0x0, ttl 64, id 15469, offset 0, flags [DF], proto UDP (17), length 62)
  equipo5.50829 > tzgroa-ab-in-f10.1e100.net.https: [bad udp cksum 0x01a7 -> 0x9bc2!] UDP, length 3
4
16:46:43.803446 IP (tos 0x0, ttl 64, id 41632, offset 0, flags [none], proto UDP (17), length 79)
  equipo5.48516 > 192.168.100.1.domain: [bad udp cksum 0x3105 -> 0x7e73!] 51928+ [1au] PTR? 15.2.0.10.in-addr.arpa. ar: . OPT UDPsize=1472 (51)
16:46:43.814738 IP (tos 0x0, ttl 64, id 15879, offset 0, flags [none], proto UDP (17), length 156)
  192.168.100.1.domain > equipo5.48516: [udp sum ok] 51928 NXDomain q: PTR? 15.2.0.10.in-addr.arpa. 0/1/1 ns: 10.in-addr.arpa. SOA prisoner.iana.org. hostmaster.root-servers.org. 1 604800 60 604800 60 4800 ar: . OPT UDPsize=512 (128)
16:46:43.817628 IP (tos 0x0, ttl 64, id 41633, offset 0, flags [none], proto UDP (17), length 68)
  equipo5.48516 > 192.168.100.1.domain: [bad udp cksum 0x30fa -> 0x6790!] 51928+ PTR? 15.2.0.10.in-addr.arpa. (40)
16:46:43.829551 IP (tos 0x0, ttl 64, id 15880, offset 0, flags [none], proto UDP (17), length 145)
  192.168.100.1.domain > equipo5.48516: [udp sum ok] 51928 NXDomain q: PTR? 15.2.0.10.in-addr.arpa. 0/1/0 ns: 10.in-addr.arpa. SOA prisoner.iana.org. hostmaster.root-servers.org. 1 604800 60 604800 60 4800 (117)
16:46:43.910766 IP (tos 0x0, ttl 64, id 26658, offset 0, flags [none], proto UDP (17), length 83)
  equipo5.34525 > 192.168.100.1.domain: [bad udp cksum 0x3109 -> 0x6099!] 16217+ [1au] PTR? 1.100.168.192.in-addr.arpa. ar: . OPT UDPsize=1472 (55)
16:46:43.924132 IP (tos 0x0, ttl 64, id 15881, offset 0, flags [none], proto UDP (17), length 160)
  192.168.100.1.domain > equipo5.34525: [udp sum ok] 16217 NXDomain q: PTR? 1.100.168.192.in-addr.arpa. 0/1/1 ns: 168.192.in-addr.arpa. SOA prisoner.iana.org. hostmaster.root-servers.org. 1 604800 60 604800 60 4800 ar: . OPT UDPsize=512 (132)
16:46:43.924299 IP (tos 0x0, ttl 64, id 26659, offset 0, flags [none], proto UDP (17), length 72)
  equipo5.34525 > 192.168.100.1.domain: [bad udp cksum 0x30fe -> 0x49b6!] 16217+ PTR? 1.100.168.192.in-addr.arpa. (44)
16:46:43.936806 IP (tos 0x0, ttl 64, id 15882, offset 0, flags [none], proto UDP (17), length 149)
  192.168.100.1.domain > equipo5.34525: [udp sum ok] 16217 NXDomain q: PTR? 1.100.168.192.in-addr.arpa. 0/1/0 ns: 168.192.in-addr.arpa. SOA prisoner.iana.org. hostmaster.root-servers.org. 1 604800 60 604800 60 4800 (121)
^C
11 packets captured
11 packets received by filter
0 packets dropped by kernel
```


Se usa la opción “-n” con la cual se muestran directamente las direcciones IP.

```
root@equipo5:~# tcpdump -n
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:01:26.666997 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [P.], seq 32298064:32304007, ack 94815104, win 65535, length 5943
18:01:26.667077 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [..], ack 5943, win 65535, length 0
18:01:26.684090 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], seq 1:200, ack 5943, win 65535, length 199
18:01:26.684486 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [..], ack 200, win 65535, length 0
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
```

Para capturar un conjunto en específico de paquetes se usa la opción “-c” seguido del número de paquetes que se quiera capturar.

```
root@equipo5:~# tcpdump -c 5
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:03:00.678240 IP equipo5.35068 > 149.154.174.100.https: Flags [P.], seq 94824530:94824697, ack 32313259, win 65535, length 167
18:03:00.679121 IP 149.154.174.100.https > equipo5.35068: Flags [..], ack 167, win 65535, length 0
18:03:00.741011 IP 149.154.174.100.https > equipo5.35068: Flags [P.], seq 1:114, ack 167, win 65535, length 113
18:03:00.741072 IP equipo5.35068 > 149.154.174.100.https: Flags [..], ack 114, win 65535, length 0
18:03:00.776510 IP equipo5.34189 > 192.168.100.1.domain: 57585+ [1au] PTR? 100.174.154.149.in-addr.arpa. (57)
5 packets captured
16 packets received by filter
0 packets dropped by kernel
```

La opción “-tttt”, muestra el tiempo con formato de horas, minutos, segundos y fracciones de segundos, seguido de la fecha en cada línea.

```
root@equipo5:~# tcpdump -tttt
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2023-09-28 18:04:09.455373 IP equipo5.52737 > 239.255.255.250.1900: UDP, length 173
2023-09-28 18:04:09.494993 IP equipo5.57834 > 192.168.100.1.domain: 13697+ [1au] PTR? 250.255.255.239.in-addr.arpa. (57)
2023-09-28 18:04:09.518730 IP 192.168.100.1.domain > equipo5.57834: 13697 NXDomain 0/1/1 (114)
2023-09-28 18:04:09.519001 IP equipo5.57834 > 192.168.100.1.domain: 13697+ PTR? 250.255.255.239.in-addr.arpa. (46)
2023-09-28 18:04:09.531503 IP 192.168.100.1.domain > equipo5.57834: 13697 NXDomain 0/1/0 (103)
2023-09-28 18:04:09.532808 IP equipo5.54243 > 192.168.100.1.domain: 49474+ [1au] PTR? 15.2.0.10.in-addr.arpa. (51)
2023-09-28 18:04:09.544749 IP 192.168.100.1.domain > equipo5.54243: 49474 NXDomain 0/1/1 (128)
2023-09-28 18:04:09.545016 IP equipo5.54243 > 192.168.100.1.domain: 49474+ PTR? 15.2.0.10.in-addr.arpa. (40)
2023-09-28 18:04:09.555323 IP 192.168.100.1.domain > equipo5.54243: 49474 NXDomain 0/1/0 (117)
2023-09-28 18:04:09.597887 IP equipo5.37420 > 192.168.100.1.domain: 6660+ [1au] PTR? 1.100.168.192.in-addr.arpa. (55)
2023-09-28 18:04:09.609434 IP 192.168.100.1.domain > equipo5.37420: 6660 NXDomain 0/1/1 (132)
2023-09-28 18:04:09.609711 IP equipo5.37420 > 192.168.100.1.domain: 6660+ PTR? 1.100.168.192.in-addr.arpa. (44)
2023-09-28 18:04:09.623281 IP 192.168.100.1.domain > equipo5.37420: 6660 NXDomain 0/1/0 (121)
2023-09-28 18:04:10.382122 IP mia09s19-in-f10.1e100.net.https > equipo5.54144: UDP, length 32
2023-09-28 18:04:10.382549 IP equipo5.54144 > mia09s19-in-f10.1e100.net.https: UDP, length 34
^C
15 packets captured
18 packets received by filter
0 packets dropped by kernel
```

Implementando en un solo comando todas las opciones anteriores, captura el número de líneas de acuerdo con tu número de equipo.

```
root@equipo5:~# tcpdump -nvc 5 -tttt
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2023-09-28 22:52:20.131062 IP (tos 0x0, ttl 64, id 21188, offset 0, flags [DF], proto UDP (17), length 57)
  10.0.2.15.38407 > 142.250.177.14.443: UDP, length 29
2023-09-28 22:52:20.141929 IP (tos 0x0, ttl 64, id 2409, offset 0, flags [none], proto UDP (17), length 53)
  142.250.177.14.443 > 10.0.2.15.38407: UDP, length 25
2023-09-28 22:52:20.345794 IP (tos 0x0, ttl 64, id 33175, offset 0, flags [DF], proto TCP (6), length 207)
  10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], cksum 0x50cf (incorrect -> 0x4dbf), seq 96333346:96333513, ack 33652
046, win 65535, length 167
2023-09-28 22:52:20.346632 IP (tos 0x0, ttl 64, id 2410, offset 0, flags [none], proto TCP (6), length 40)
  149.154.174.100.443 > 10.0.2.15.35068: Flags [.], cksum 0x6139 (correct), ack 167, win 65535, length 0
2023-09-28 22:52:20.406382 IP (tos 0x0, ttl 64, id 2411, offset 0, flags [none], proto TCP (6), length 153)
  149.154.174.100.443 > 10.0.2.15.35068: Flags [P.], cksum 0xa7b4 (correct), seq 1:114, ack 167, win 65535, length 113
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```

EXPRESIONES DE FILTRO TCPDUMP

Realiza 4 combinaciones con los filtros, cada combinación deberá capturar al menos 3 paquetes.

Ej. La IP tanto de origen como destino es 185.125.190.56 y contine al puerto 123. Deberás agregar captura de pantalla del comando de cada combinación.

Verificamos que IPs tenemos

```
root@equipo5:~# tcpdump -i enp0s3 -n
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
20:28:10.541330 IP 10.0.2.15.47661 > 239.255.255.250.1900: UDP, length 173
20:28:11.432939 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], seq 95584223:95584390, ack 32952222, win 65535, length 167
20:28:11.433315 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [.], ack 167, win 65535, length 0
20:28:11.492940 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [P.], seq 1:114, ack 167, win 65535, length 113
20:28:11.492976 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [.], ack 114, win 65535, length 0
20:28:11.541798 IP 10.0.2.15.47661 > 239.255.255.250.1900: UDP, length 173
20:28:11.543088 IP 172.217.2.138.443 > 10.0.2.15.40767: UDP, length 119
20:28:11.545636 IP 10.0.2.15.40767 > 172.217.2.138.443: UDP, length 33
20:28:12.072960 IP 10.0.2.15.35074 > 149.154.167.99.443: Flags [.], ack 2075788754, win 62780, length 0
20:28:12.073278 IP 149.154.167.99.443 > 10.0.2.15.35074: Flags [.], ack 1, win 65535, length 0
20:28:12.579791 IP 10.0.2.15.44228 > 142.251.34.14.443: UDP, length 29
20:28:12.581115 IP 142.251.34.14.443 > 10.0.2.15.44228: UDP, length 37
20:28:12.584270 IP 10.0.2.15.44228 > 142.251.34.14.443: UDP, length 33
20:28:12.591758 IP 142.251.34.14.443 > 10.0.2.15.44228: UDP, length 26
20:28:13.436831 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], seq 167:300, ack 114, win 65535, length 133
20:28:13.437636 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [.], ack 300, win 65535, length 0
20:28:13.497792 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [P.], seq 114:227, ack 300, win 65535, length 113
20:28:13.497832 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [.], ack 227, win 65535, length 0
20:28:15.435478 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], seq 300:433, ack 227, win 65535, length 133
20:28:15.435856 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [.], ack 433, win 65535, length 0
20:28:15.495316 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [P.], seq 227:340, ack 433, win 65535, length 113
20:28:15.495357 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [.], ack 340, win 65535, length 0
20:28:17.433540 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], seq 433:600, ack 340, win 65535, length 167
20:28:17.433844 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [.], ack 600, win 65535, length 0
20:28:17.494350 IP 149.154.174.100.443 > 10.0.2.15.35068: Flags [P.], seq 340:453, ack 600, win 65535, length 113
20:28:17.494415 IP 10.0.2.15.35068 > 149.154.174.100.443: Flags [.], ack 453, win 65535, length 0
```

La IP tanto de origen como destino es 172.217.2.138 y contiene al puerto 443

```
root@equipo5:~# tcpdump -nvc 5 -tttt host 172.217.2.138 and port 443
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2023-09-28 20:59:12.876020 IP (tos 0x0, ttl 64, id 32241, offset 0, flags [none], proto UDP (17), length 187)
  172.217.2.138.443 > 10.0.2.15.40767: UDP, length 159
2023-09-28 20:59:12.880125 IP (tos 0x0, ttl 64, id 54797, offset 0, flags [DF], proto UDP (17), length 61)
  10.0.2.15.40767 > 172.217.2.138.443: UDP, length 33
2023-09-28 20:59:24.313147 IP (tos 0x0, ttl 64, id 32264, offset 0, flags [none], proto UDP (17), length 108)
  172.217.2.138.443 > 10.0.2.15.40767: UDP, length 80
2023-09-28 20:59:24.316475 IP (tos 0x0, ttl 64, id 54798, offset 0, flags [DF], proto UDP (17), length 61)
  10.0.2.15.40767 > 172.217.2.138.443: UDP, length 33
2023-09-28 20:59:28.680867 IP (tos 0x0, ttl 64, id 32300, offset 0, flags [none], proto UDP (17), length 148)
  172.217.2.138.443 > 10.0.2.15.40767: UDP, length 120
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```

Todos los paquetes cuyo host origen es 142.251.34.14 y contiene al puerto 443

```
root@equipo5:~# tcpdump -nvc 5 -tttt src host 142.251.34.14 and port 443
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2023-09-28 20:53:35.843827 IP (tos 0x0, ttl 64, id 30665, offset 0, flags [none], proto UDP (17), length 54)
  142.251.34.14.443 > 10.0.2.15.44228: UDP, length 26
2023-09-28 20:53:39.847497 IP (tos 0x0, ttl 64, id 30676, offset 0, flags [none], proto UDP (17), length 60)
  142.251.34.14.443 > 10.0.2.15.44228: UDP, length 32
2023-09-28 20:53:39.882526 IP (tos 0x0, ttl 64, id 30685, offset 0, flags [none], proto UDP (17), length 146)
  142.251.34.14.443 > 10.0.2.15.44228: UDP, length 118
2023-09-28 20:53:39.883247 IP (tos 0x0, ttl 64, id 30686, offset 0, flags [none], proto UDP (17), length 136)
  142.251.34.14.443 > 10.0.2.15.44228: UDP, length 108
2023-09-28 20:53:39.900181 IP (tos 0x0, ttl 64, id 30689, offset 0, flags [none], proto UDP (17), length 54)
  142.251.34.14.443 > 10.0.2.15.44228: UDP, length 26
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```

Todos los paquetes cuyo host destino es 149.154.174.100 y contiene al puerto 443

```
root@equipo5:~# tcpdump -nvc 5 -tttt dst host 149.154.174.100 and port 443
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2023-09-28 20:55:49.990629 IP (tos 0x0, ttl 64, id 25985, offset 0, flags [DF], proto TCP (6), length 207)
  10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], cksum 0x50cf (incorrect -> 0x1933), seq 95729886:95730053,
  ack 33060112, win 65535, length 167
2023-09-28 20:55:50.050761 IP (tos 0x0, ttl 64, id 25986, offset 0, flags [DF], proto TCP (6), length 40)
  10.0.2.15.35068 > 149.154.174.100.443: Flags [.], cksum 0x5028 (incorrect -> 0x9e5c), ack 114, win 65535, len
  gth 0
2023-09-28 20:55:51.993942 IP (tos 0x0, ttl 64, id 25987, offset 0, flags [DF], proto TCP (6), length 189)
  10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], cksum 0x50bd (incorrect -> 0x2ff7), seq 167:316, ack 114,
  win 65535, length 149
2023-09-28 20:55:52.056202 IP (tos 0x0, ttl 64, id 25988, offset 0, flags [DF], proto TCP (6), length 40)
  10.0.2.15.35068 > 149.154.174.100.443: Flags [.], cksum 0x5028 (incorrect -> 0x9d56), ack 227, win 65535, len
  gth 0
2023-09-28 20:55:53.992887 IP (tos 0x0, ttl 64, id 25989, offset 0, flags [DF], proto TCP (6), length 239)
  10.0.2.15.35068 > 149.154.174.100.443: Flags [P.], cksum 0x50ef (incorrect -> 0xe2c7), seq 316:515, ack 227,
  win 65535, length 199
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Todos los paquetes que no contienen al puerto 443

```
root@equipo5:~# tcpdump -nvc 5 -tttt not port 443
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2023-09-28 21:13:17.266051 IP6 (flowlabel 0x48d44, hlim 255, next-header UDP (17) payload length: 126) fe80::8f7c
:a124:7017:cf14.5353 > ff02::fb.5353: [bad udp cksum 0x6edb -> 0xeabf!] 0 [7q] PTR (QM)? _ftp._tcp.local. PTR (QM)
)? _nfs._tcp.local. PTR (QM)? _afpovertcp._tcp.local. PTR (QM)? _smb._tcp.local. PTR (QM)? _sftp-ssh._tcp.local.
PTR (QM)? _webdav._tcp.local. PTR (QM)? _webdav._tcp.local. (118)
2023-09-28 21:13:17.266396 IP (tos 0x0, ttl 255, id 27886, offset 0, flags [DF], proto UDP (17), length 146)
  10.0.2.15.5353 > 224.0.0.251.5353: 0 [7q] PTR (QM)? _ftp._tcp.local. PTR (QM)? _nfs._tcp.local. PTR (QM)? _af
  povertcp._tcp.local. PTR (QM)? _smb._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _webdav._tcp.local. PT
  R (QM)? _webdav._tcp.local. (118)
2023-09-28 21:13:19.528898 IP (tos 0x0, ttl 64, id 28000, offset 0, flags [DF], proto TCP (6), length 40)
  10.0.2.15.54838 > 142.250.113.188.5228: Flags [.], cksum 0x0ce0 (incorrect -> 0x284c), ack 2343111753, win 62
  780, length 0
2023-09-28 21:13:19.529325 IP (tos 0x0, ttl 64, id 39893, offset 0, flags [none], proto TCP (6), length 40)
  142.250.113.188.5228 > 10.0.2.15.54838: Flags [.], cksum 0x1d88 (correct), ack 1, win 65535, length 0
2023-09-28 21:13:33.823143 IP (tos 0x0, ttl 64, id 65223, offset 0, flags [none], proto UDP (17), length 76)
  10.0.2.15.39659 > 192.168.100.1.53: 51099+ [1au] A? accounts.google.com. (48)
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Todos los paquetes cuyo tamaño sea mayor a 200 bytes

```
root@equipo5:~# tcpdump -nvc 5 -tttt greater 200
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2023-09-28 21:21:39.880340 IP (tos 0x0, ttl 64, id 27829, offset 0, flags [DF], proto UDP (17), length 633)
  10.0.2.15.51554 > 142.251.34.14.443: UDP, length 605
2023-09-28 21:21:39.881747 IP (tos 0x0, ttl 64, id 31147, offset 0, flags [DF], proto UDP (17), length 1278)
  10.0.2.15.60525 > 142.251.34.3.443: UDP, length 1250
2023-09-28 21:21:39.882332 IP (tos 0x0, ttl 64, id 31149, offset 0, flags [DF], proto UDP (17), length 780)
  10.0.2.15.60525 > 142.251.34.3.443: UDP, length 752
2023-09-28 21:21:39.885201 IP (tos 0x0, ttl 64, id 1712, offset 0, flags [DF], proto UDP (17), length 1278)
  10.0.2.15.47875 > 142.251.218.142.443: UDP, length 1250
2023-09-28 21:21:39.918086 IP (tos 0x0, ttl 64, id 43718, offset 0, flags [none], proto UDP (17), length 1278)
  142.251.34.3.443 > 10.0.2.15.60525: UDP, length 1250
5 packets captured
20 packets received by filter
0 packets dropped by kernel
```

GUARDAR LA INFORMACIÓN EN UN ARCHIVO

Ejercicio 1

Genera una captura de tráfico que cumpla con las siguientes características:

1. Guarde la captura en un archivo *.pcap con el nombre "CapturaEquipoX" (sustituya la X por el número de equipo).
2. La cantidad de paquetes que contendrá el archivo será número de acuerdo con el día que estén realizando la actividad más el número del equipo (ej. 26 de septiembre + equipo 9 = 2609 paquetes)
3. Use por lo menos un filtro y las opciones que considere necesarias para tener una captura detallada.
4. Anexe la captura de pantalla del comando y explicar el filtro usado.
5. Vea el contenido del archivo (-r), seleccione al menos 20 paquetes, analice el contenido de cada paquete y explique su comportamiento.

NOTA: Genere tráfico al momento de realizar su captura (ping, envío de archivos, reproducción multimedia, etc.).

Para guardar la captura de datos en un archivo .cap

```
tcpdump -w nombreArchivo.cap
```

Cantidad de paquetes: 28 de septiembre + equipo 5 = 2805 paquetes

Utilizamos -nvc -tttt para visualizar los datos más detallados.

El filtro greater 300 para que salieran únicamente los paquetes de tamaño mayor a 300 bytes.

Y los paquetes con puerto 443.

```
root@equipo5:~# tcpdump -nvc 2805 -tttt -w CapturaEquipo5.cap greater 300 and port 443
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
2805 packets captured
2875 packets received by filter
0 packets dropped by kernel
```

Observamos el contenido del archivo seleccionando 20 paquetes.


```

root@equipo5:~# tcpdump -c 20 -r CapturaEquipo5.cap
reading from file CapturaEquipo5.cap, link-type EN10MB (Ethernet), snapshot length 262144
01:11:10.819261 IP equipo5.36686 > qro01s27-in-f14.1e100.net.https: UDP, length 400
01:11:10.897435 IP qro01s27-in-f14.1e100.net.https > equipo5.36686: UDP, length 623
01:11:10.898509 IP qro01s27-in-f14.1e100.net.https > equipo5.36686: UDP, length 268
01:11:18.788759 IP equipo5.58685 > kul08s10-in-f3.1e100.net.https: UDP, length 1250
01:11:19.012470 IP kul08s10-in-f3.1e100.net.https > equipo5.58685: UDP, length 1250
01:11:19.013879 IP equipo5.58685 > kul08s10-in-f3.1e100.net.https: UDP, length 1250
01:11:19.021441 IP kul08s10-in-f3.1e100.net.https > equipo5.58685: UDP, length 1250
01:11:19.021993 IP kul08s10-in-f3.1e100.net.https > equipo5.58685: UDP, length 1250
01:11:19.022451 IP kul08s10-in-f3.1e100.net.https > equipo5.58685: UDP, length 1246
01:11:19.024598 IP equipo5.58685 > kul08s10-in-f3.1e100.net.https: UDP, length 379
01:11:19.306237 IP kul08s10-in-f3.1e100.net.https > equipo5.58685: UDP, length 946
01:11:19.309981 IP equipo5.58685 > kul08s10-in-f3.1e100.net.https: UDP, length 497
01:11:19.592263 IP kul08s10-in-f3.1e100.net.https > equipo5.58685: UDP, length 774
01:11:20.442114 IP whatsapp-cdn-shv-01-qro1.fbcdn.net.https > equipo5.41296: Flags [P.], s
eq 250966156:250966424, ack 833066230, win 65535, length 268
01:11:22.085023 IP equipo5.50095 > qro04s04-in-f10.1e100.net.https: UDP, length 1250
01:11:22.090839 IP equipo5.50095 > qro04s04-in-f10.1e100.net.https: UDP, length 511
01:11:22.123839 IP qro04s04-in-f10.1e100.net.https > equipo5.50095: UDP, length 1250
01:11:22.124402 IP qro04s04-in-f10.1e100.net.https > equipo5.50095: UDP, length 804
01:11:22.125837 IP equipo5.50095 > qro04s04-in-f10.1e100.net.https: UDP, length 1250
01:11:22.283176 IP equipo5.51520 > tzqroa-ac-in-f10.1e100.net.https: UDP, length 530

```

Tcpdump lee el archivo .cap y muestra la información de cada paquete, en este caso de 20 paquetes indicado por el comando -c 20.

Para cada paquete se muestra información como la dirección IP de origen y destino, hora de captura, longitud del paquete. Tcpdump intentará interpretar y mostrar la información sobre los protocolos presentes en los paquetes capturados, en este caso identifica el protocolo UDP y llega a mostrar detalles específicos del protocolo.

Está aplicando el filtro por el tamaño de bytes que especificamos en el paso anterior (300) y con puerto 443.

otra opción agregando -nv -tttt

```

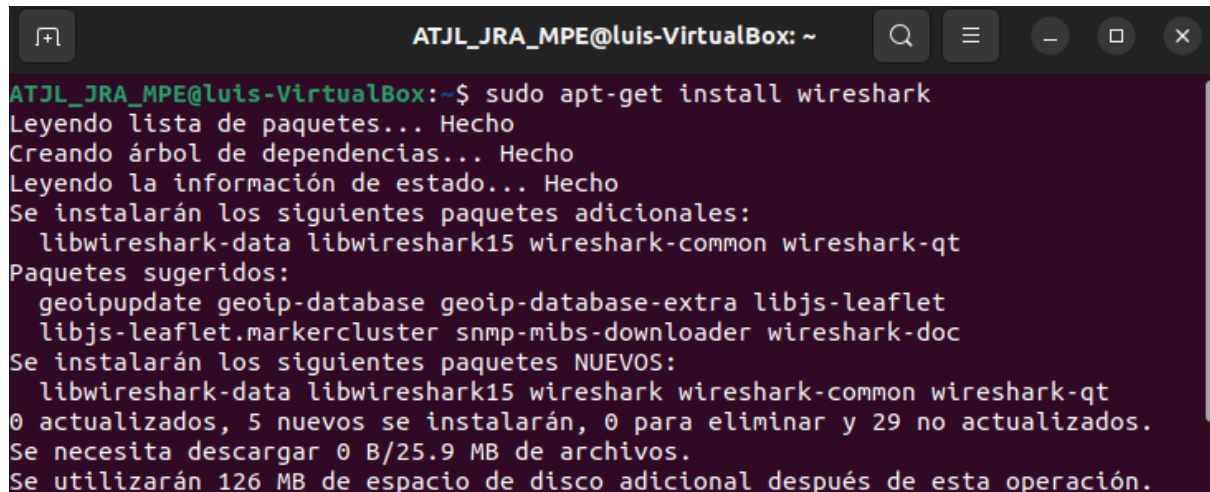
root@equipo5:~# tcpdump -nv -tttt -r CapturaEquipo5.cap
reading from file CapturaEquipo5.cap, link-type EN10MB (Ethernet), snapshot length 262144
2023-09-29 01:11:10.819261 IP (tos 0x0, ttl 64, id 63512, offset 0, flags [DF], proto UDP (17), length 428)
  10.0.2.15.36686 > 142.251.34.14.443: UDP, length 400
2023-09-29 01:11:10.897435 IP (tos 0x0, ttl 64, id 4653, offset 0, flags [none], proto UDP (17), length 651)
  142.251.34.14.443 > 10.0.2.15.36686: UDP, length 623
2023-09-29 01:11:10.898509 IP (tos 0x0, ttl 64, id 4655, offset 0, flags [none], proto UDP (17), length 296)
  142.251.34.14.443 > 10.0.2.15.36686: UDP, length 268
2023-09-29 01:11:18.788759 IP (tos 0x0, ttl 64, id 20277, offset 0, flags [DF], proto UDP (17), length 1278)
  10.0.2.15.58685 > 216.58.221.195.443: UDP, length 1250
2023-09-29 01:11:19.012470 IP (tos 0x0, ttl 64, id 4701, offset 0, flags [none], proto UDP (17), length 1278)
  216.58.221.195.443 > 10.0.2.15.58685: UDP, length 1250
2023-09-29 01:11:19.013879 IP (tos 0x0, ttl 64, id 20280, offset 0, flags [DF], proto UDP (17), length 1278)
  10.0.2.15.58685 > 216.58.221.195.443: UDP, length 1250
2023-09-29 01:11:19.021441 IP (tos 0x0, ttl 64, id 4702, offset 0, flags [none], proto UDP (17), length 1278)
  216.58.221.195.443 > 10.0.2.15.58685: UDP, length 1250
2023-09-29 01:11:19.021993 IP (tos 0x0, ttl 64, id 4703, offset 0, flags [none], proto UDP (17), length 1278)
  216.58.221.195.443 > 10.0.2.15.58685: UDP, length 1250
2023-09-29 01:11:19.022451 IP (tos 0x0, ttl 64, id 4704, offset 0, flags [none], proto UDP (17), length 1274)
  216.58.221.195.443 > 10.0.2.15.58685: UDP, length 1246
2023-09-29 01:11:19.024598 IP (tos 0x0, ttl 64, id 20282, offset 0, flags [DF], proto UDP (17), length 407)
  10.0.2.15.58685 > 216.58.221.195.443: UDP, length 379
2023-09-29 01:11:19.306237 IP (tos 0x0, ttl 64, id 4706, offset 0, flags [none], proto UDP (17), length 974)
  216.58.221.195.443 > 10.0.2.15.58685: UDP, length 946
2023-09-29 01:11:19.309981 IP (tos 0x0, ttl 64, id 20286, offset 0, flags [DF], proto UDP (17), length 525)
  10.0.2.15.58685 > 216.58.221.195.443: UDP, length 497
2023-09-29 01:11:19.592263 IP (tos 0x0, ttl 64, id 4714, offset 0, flags [none], proto UDP (17), length 802)
  216.58.221.195.443 > 10.0.2.15.58685: UDP, length 774
2023-09-29 01:11:20.442114 IP (tos 0x0, ttl 64, id 4718, offset 0, flags [none], proto TCP (6), length 308)
  31.13.89.53.443 > 10.0.2.15.41296: Flags [P.], cksum 0x9f82 (correct), seq 250966156:250966424, ack 833066230, win 65535, length 268
2023-09-29 01:11:22.085023 IP (tos 0x0, ttl 64, id 51083, offset 0, flags [DF], proto UDP (17), length 1278)
  10.0.2.15.50095 > 172.217.4.170.443: UDP, length 1250
2023-09-29 01:11:22.090839 IP (tos 0x0, ttl 64, id 51085, offset 0, flags [DF], proto UDP (17), length 539)
  10.0.2.15.50095 > 172.217.4.170.443: UDP, length 511
2023-09-29 01:11:22.123839 IP (tos 0x0, ttl 64, id 4732, offset 0, flags [none], proto UDP (17), length 1278)
  172.217.4.170.443 > 10.0.2.15.50095: UDP, length 1250
2023-09-29 01:11:22.124402 IP (tos 0x0, ttl 64, id 4733, offset 0, flags [none], proto UDP (17), length 832)
  172.217.4.170.443 > 10.0.2.15.50095: UDP, length 804
2023-09-29 01:11:22.125837 IP (tos 0x0, ttl 64, id 51086, offset 0, flags [DF], proto UDP (17), length 1278)
  10.0.2.15.50095 > 172.217.4.170.443: UDP, length 1250
2023-09-29 01:11:22.283176 IP (tos 0x0, ttl 64, id 33990, offset 0, flags [DF], proto UDP (17), length 558)
  10.0.2.15.51520 > 192.178.52.202.443: UDP, length 530

```

Ejercicio 2

Descargue el archivo *.pcap proporcionado. Mediante la herramienta wireshark analiza los paquetes obtenidos e identifica los datos que se le solicitan a continuación agregando una muy breve explicación de cómo se obtuvo y captura de pantalla del resultado.

Instalación de Wireshark.



```
ATJL_JRA_MPE@luis-VirtualBox: ~  
ATJL_JRA_MPE@luis-VirtualBox:~$ sudo apt-get install wireshark  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Se instalarán los siguientes paquetes adicionales:  
  libwireshark-data libwireshark15 wireshark-common wireshark-qt  
Paquetes sugeridos:  
  geoipupdate geoip-database geoip-database-extra libjs-leaflet  
  libjs-leaflet.markercluster snmp-mibs-downloader wireshark-doc  
Se instalarán los siguientes paquetes NUEVOS:  
  libwireshark-data libwireshark15 wireshark wireshark-common wireshark-qt  
0 actualizados, 5 nuevos se instalarán, 0 para eliminar y 29 no actualizados.  
Se necesita descargar 0 B/25.9 MB de archivos.  
Se utilizarán 126 MB de espacio de disco adicional después de esta operación.
```

1. Nombre de la red.

Primero vamos a aplicar un filtro para mostrar solo paquetes de tipo Beacon:

```
wlan.fc.type_subtype == 0x08
```

Este filtro selecciona tramas de gestión que son del tipo "Beacon". Los paquetes Beacon son emitidos periódicamente por los puntos de acceso para anunciar la existencia de la red y proporcionar información sobre la red, como el SSID, la velocidad de datos y, en algunos casos, el canal utilizado.

Para poder encontrar el SSID que es básicamente el nombre de la red Wi-Fi, seleccionamos un paquete de tipo "Beacon" ya que son transmitidos por puntos de acceso para anunciar la existencia de la red. Una vez encontrado uno hacemos doble clic sobre el paquete para que nos despliegue el panel de detalles. Una vez dentro, seleccionamos el apartado "IEEE 802.11 Wireless Management", este a su vez nos despliega dos opciones, de las que vamos a dar clic en "Tagged Parameters" y vemos el apartado "Tag: SSID parameter set: Coherer".

Por lo que el nombre de la red es “Coherer”.

The screenshot shows the Wireshark interface with a packet capture filter `wlan.fc.type_subtype == 0x08` applied. The packet list shows five packets, all of which are 802.11 Beacon frames from source `Cisco-Li_82:b2:55` to destination `Broadcast`. The packet details pane for the first packet (No. 1) is expanded, showing the following structure:

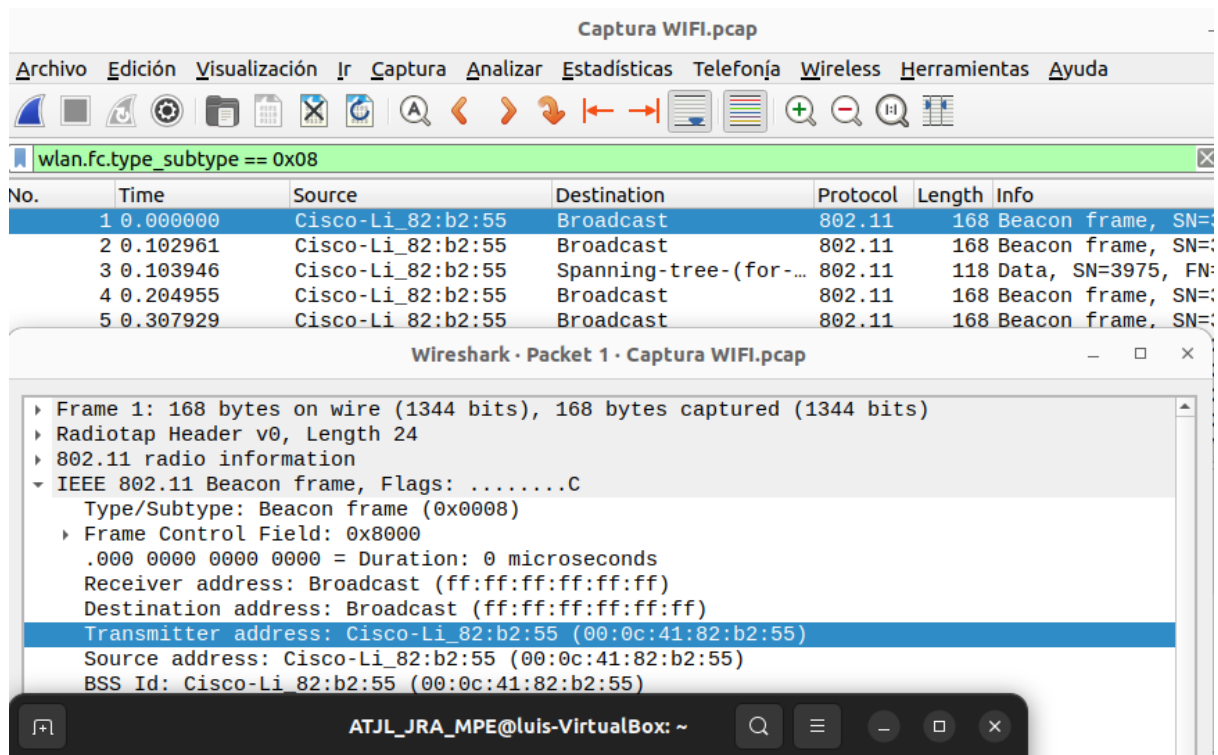
- Frame 1: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits)
- Radiotap Header v0, Length 24
- 802.11 radio information
- IEEE 802.11 Beacon frame, Flags:C
- IEEE 802.11 Wireless Management
 - Fixed parameters (12 bytes)
 - Tagged parameters (104 bytes)
 - Tag: SSID parameter set: Coherer
 - Tag: Supported Rates 1(B), 2(B), 5.5(B), 11(B), 18, 24, 36, 54, [Mbit/sec]
 - Tag: DS Parameter set: Current Channel: 1

2. BSSID.

Primero vamos a aplicar un filtro para mostrar solo paquetes de tipo Beacon:

`wlan.fc.type_subtype == 0x08`

Para poder encontrar el BSSID (Basic Service Set Identifier) que es la dirección MAC del punto de acceso Wi-Fi. Primero seleccionamos un paquete de tipo "Beacon" ya que son transmitidos por puntos de acceso para anunciar la existencia de la red. Una vez encontrado uno hacemos doble clic sobre el paquete para que nos despliegue el panel de detalles. Una vez dentro, seleccionamos el apartado "IEEE 802.11 Beacon frame, Flags:C", a continuación podemos observar que se despliega un apartado que se llama "Transmitter address" en la cual es la dirección MAC del punto de acceso, que es el BSSID. Por lo que la BSSID es: 00:0c:41:82:b2:55



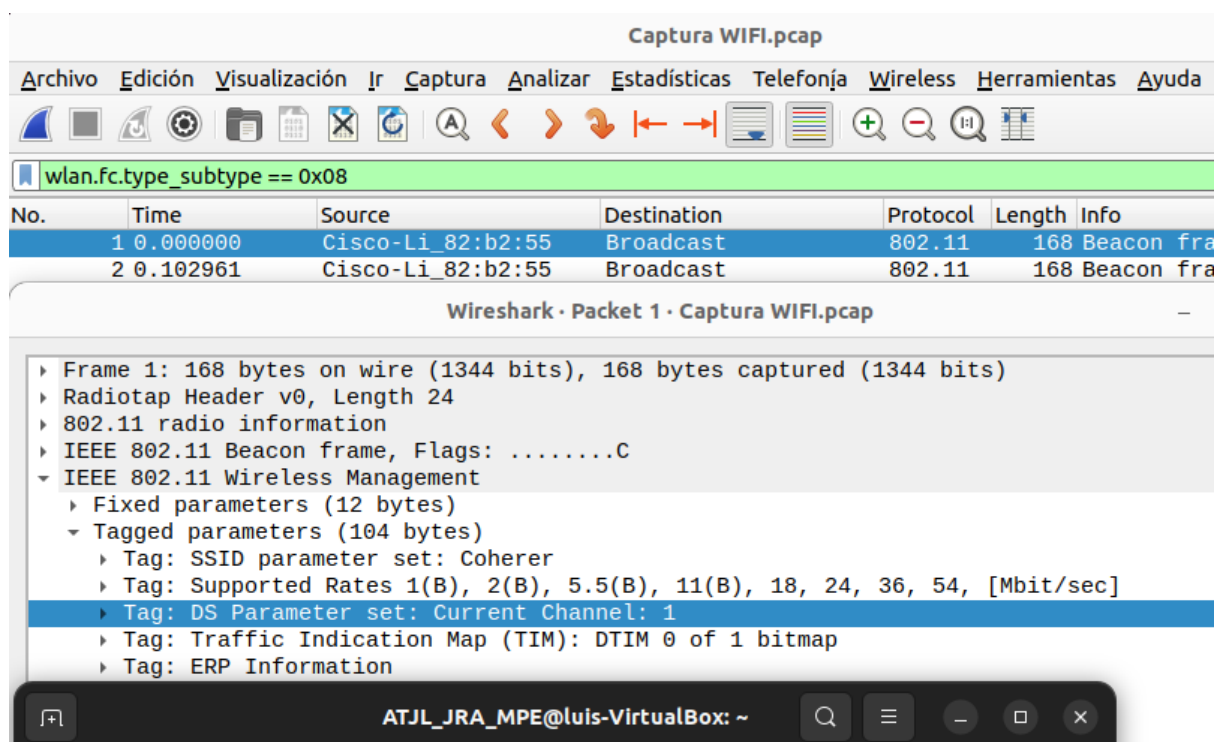
3. Canal utilizado.

Primero vamos a aplicar un filtro para mostrar solo paquetes de tipo Beacon:

`wlan.fc.type_subtype == 0x08`

Posteriormente, seleccionamos un paquete Beacon y hacemos doble clic, a continuación se va a desplegar una nueva ventana con los detalles. Ahora, seleccionamos la opción: "IEEE 802.11 Wireless Management", después "Tagged Parameters", y buscamos el apartado "DS Parameter Set" en la que encontraremos el valor del canal.

Por lo que, el valor del canal en este caso es: 1.



4. Clientes conectados.

Primero, vamos a aplicar un filtro para seleccionar paquetes de gestión de tipo "Association Request" (solicitud de asociación) y "Association Response" (respuesta de asociación):

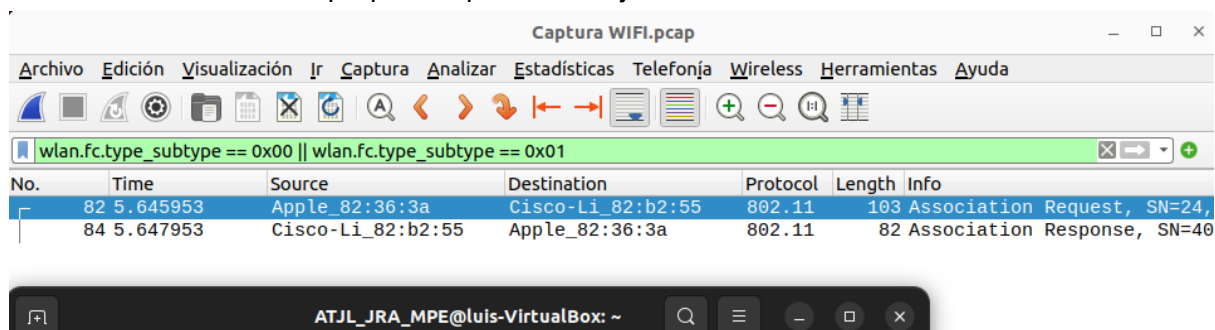
wlan.fc.type_subtype == 0x00 || wlan.fc.type_subtype == 0x01

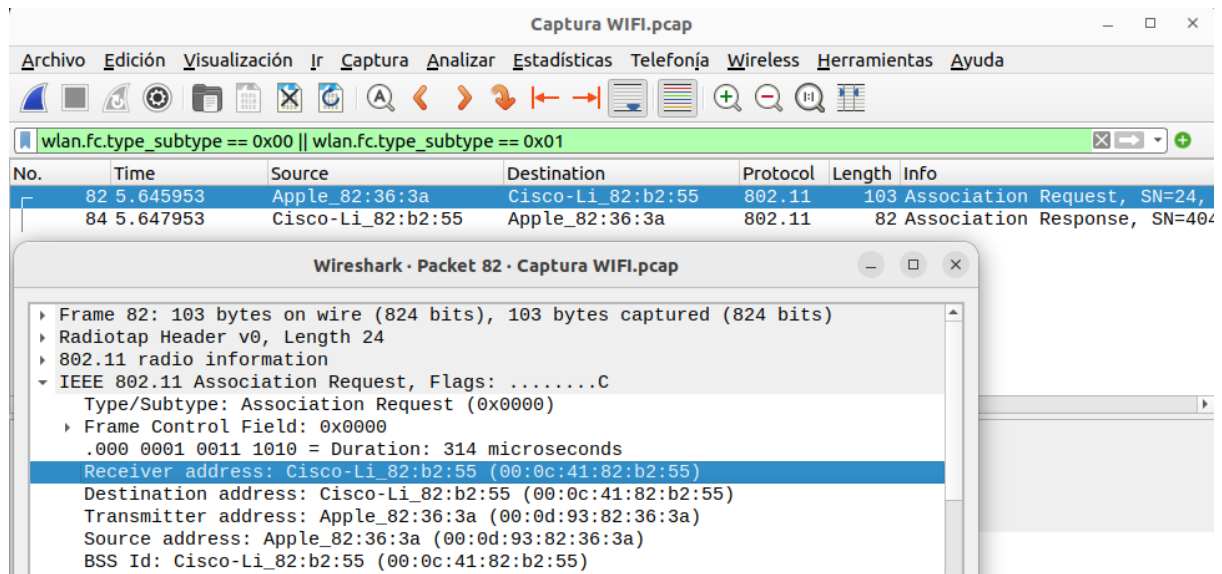
Esto nos mostrará los paquetes relacionados con la asociación y desasociación. Veamos que los paquetes de asociación (Association Request y Association Response) y los paquetes de desasociación (Disassociation) son tipos de tramas de gestión en el estándar 802.11 que se utilizan para gestionar la conexión y desconexión de estaciones (clientes) con un punto de acceso en una red Wi-Fi.

Ahora, vamos a analizar las direcciones MAC para poder identificar los clientes conectados.

Nota: notemos que dado una dirección MAC también podemos hacer un filtro para ver todos los paquetes relacionados.

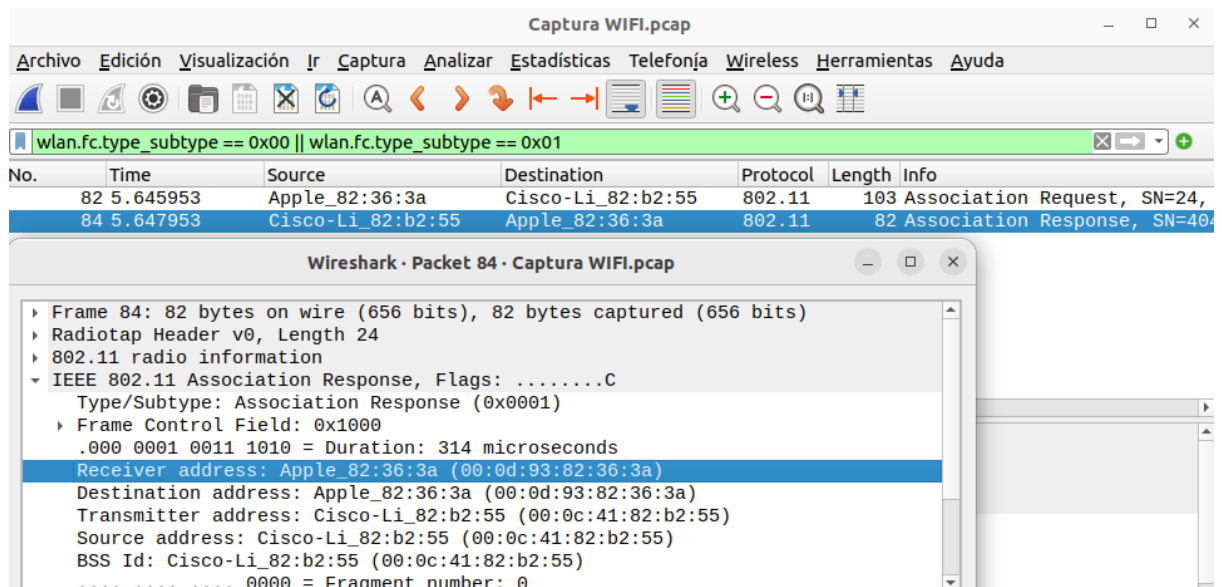
En este caso tenemos 2 paquetes que nos arroja este filtro.





Clientes asociados:

- Receiver address: 00:0c:41:82:b2:55
- Destination address: 00:0c:41:82:b2:55



Clientes asociados:

- Receiver address: 00:0d:93:82:36:3a
- Destination address: 00:0d:93:82:36:3a

5. Tipo de seguridad.

Primero vamos a aplicar un filtro para mostrar solo paquetes de tipo Beacon:

wlan.fc.type_subtype == 0x08

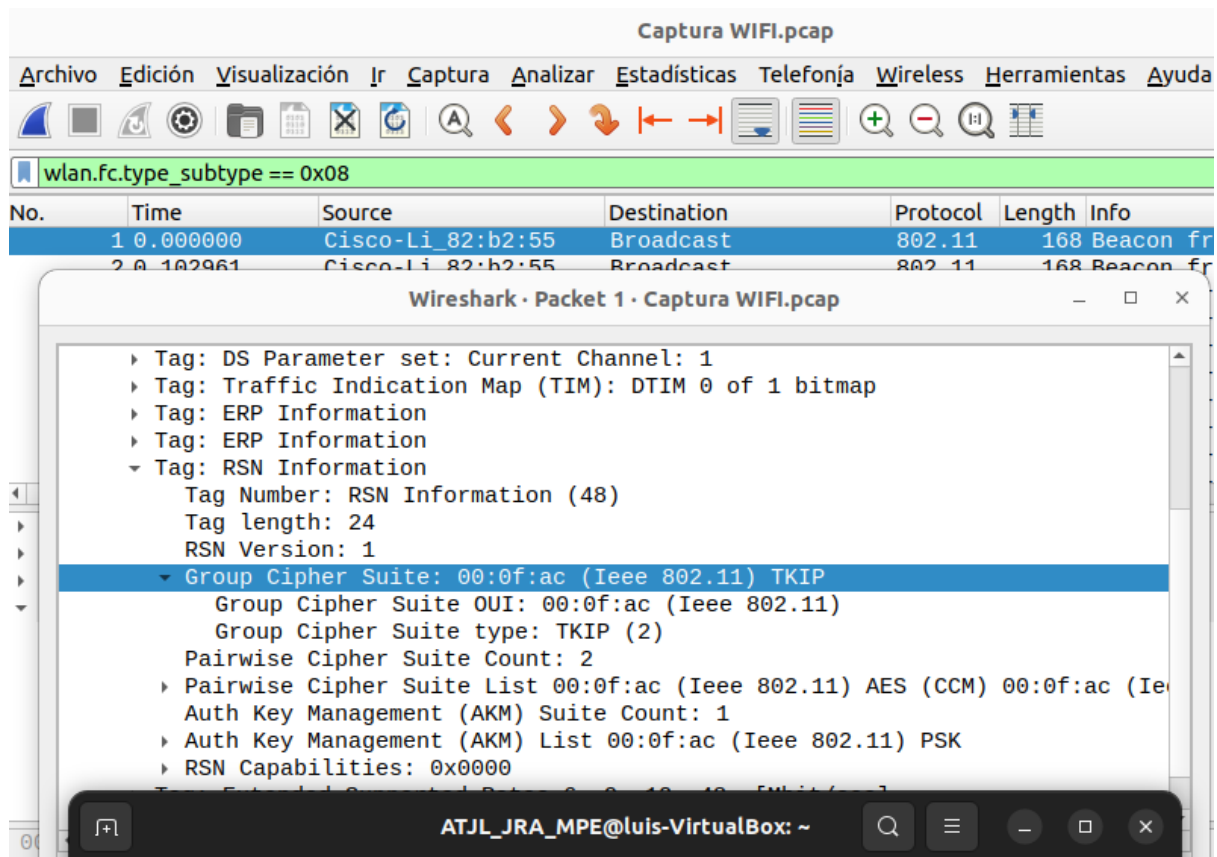
Posteriormente, seleccionamos un paquete que resultó de aplicar el filtro, y hacemos doble clic sobre el paquete para desplegar la ventana de detalles. Dentro de esta ventana, buscamos el elemento SSID (Service Set Identifier) que indica el nombre de la red y

buscamos el elemento RSN (Robust Security Network) que indica el tipo de seguridad utilizado. En particular, vamos a identificar el campo "Pairwise Cipher Suite" y "Group Cipher Suite" para determinar el tipo de cifrado utilizado. Estos campos indicarán el tipo de cifrado utilizado, como WPA2-CCMP (AES) o WPA-TKIP.

Nota: En las redes seguras, la información sobre el tipo de seguridad puede estar cifrada para proteger la privacidad y la seguridad de la red.

Los pasos a seguir en la ventana de detalles son IEEE 802.11 Wireless Management, Tagged parameters, RSN Information.

En este caso es la siguiente:



TKIP

6. Credenciales de acceso.

7. IP (origen y destino) y sitio web al que accedió.

NOTA: Los puntos 6 y 7 son opcionales. En caso de decidir realizarlos y obtener la respuesta correcta tienen un valor adicional de 20 pts.

PREGUNTAS ADICIONALES

a) En un párrafo intente explicar el comportamiento de los paquetes capturados:

¿Qué fue lo que pasó?

Se aplicaron los filtros indicados para que solo se muestren los paquetes restringidos por esa condición, mostrando sus horas de captura, IPs y puertos de origen y destino, longitud y protocolos que utilizan los paquetes.

b) ¿En qué casos utilizaría tcpdump y en cuáles wireshark?

Tcpdump es una utilidad de línea de comandos, mientras que Wireshark tiene una potente interfaz gráfica. Si bien tcpdump entiende algunos protocolos de la capa de aplicación, Wireshark comprende una cantidad mucho mayor de protocolos. Wireshark, emplea la misma librería de captura de paquetes (libpcap) que otros sniffers conocidos, como tcpdump, aunque es capaz de leer muchos otros tipos de formato de captura.

Básicamente tcpdump es ideal para capturar y filtrar paquetes en entornos de línea de comandos, especialmente en situaciones donde no hay una interfaz gráfica disponible. Wireshark hace un análisis detallado de paquetes a través de su interfaz gráfica, proporcionando herramientas visuales para entender y diagnosticar problemas en el tráfico de red. En muchos casos, ambos se utilizan de manera complementaria, con tcpdump para la captura inicial y Wireshark para un análisis más profundo.

c) ¿Cuáles son las ventajas y desventajas de ambas herramientas?

TCPDUMP	
Ventajas	Desventajas
<ul style="list-style-type: none">• Tcpdump es una herramienta de línea de comandos popular y ligera para capturar paquetes y analizar el tráfico de red.• Es una utilidad de línea de comandos.• Disponible para múltiples plataformas.	<ul style="list-style-type: none">• Tcpdump puede capturar y analizar paquetes, pero no encripta los datos de dichos paquetes.• Si bien tcpdump entiende algunos protocolos de la capa de aplicación, Wireshark comprende una cantidad mucho mayor de protocolos• No es recomendable tenerlo en una red con mucho tráfico
WIRESHARK	
Ventajas	Desventajas
<ul style="list-style-type: none">• Entiende más protocolos de nivel de aplicación.• Documentación extensa.• Captura todo tipo de paquetes al analizar la red.• Muestra errores y problemas en niveles por debajo del protocolo HTTP.• Guarda y restaura los datos empaquetados capturados, en ficheros pcap.	<ul style="list-style-type: none">• Al analizar la red no se pueden modificar datos de los paquetes, solo mediante ficheros de red, sus pcap.• La interfaz que usa es funcional, pero es poco intuitiva, a comparación de otras interfaces en la actualidad.

CONCLUSIONES

José Luis:

Al realizar esta práctica utilizamos dos herramientas muy importantes TCPDUMP y Wireshark, note que esta última es una herramienta muy útil para analizar y solucionar problemas en redes, ofreciendo detalles muy buenos del tráfico de paquetes. Su capacidad para filtrar y decodificar protocolos permite identificar eficazmente problemas de red y optimizar su rendimiento. Sin embargo, tiene limitaciones en entornos cifrados, donde la privacidad y ética deben ser prioritarias, ya que no puede revelar información detallada de tráfico encriptado. Sin embargo, Wireshark se destaca como una herramienta indispensable para analizar el tráfico de redes.

Abraham: TCPDump lo hace adecuado para entornos de servidor o sistemas con recursos limitados ya que al usarlo es fácil y podemos ver el diagnóstico de problemas en redes mientras que Wireshark ofrece capacidades avanzadas y una interfaz gráfica intuitiva es ideal para un diagnóstico profundo. Con lo anterior puedo decir que comprendimos la importancia de aplicar filtros personalizados durante la captura de paquetes y con esto una visión detallada del comportamiento de la red. Y pues nada quiero agradecer a mis compañeros Jose Luis y Esau por realizar la instalación.

Esaú:

En esta práctica observamos el tráfico de paquetes a través de dos herramientas: Tcpcdump y Wireshark. En estas herramientas nos muestran la captura de paquetes (puede ser la cantidad que tu desees) y analizan el tráfico de red, mostrando información de estos, como la hora, ip, puerto, banderas, protocolo UDP, etc. Ambas herramientas son similares, solo que Wireshark utiliza una interfaz gráfica lo que te puede dar una idea más intuitiva o especializada del análisis. Probamos filtros para que nos aparecieran paquetes únicamente con las características que indicamos, como que ip es origen y cuál destino, con cierta cantidad de bytes o con cierto puerto. Reconocimos sus diferencias entre Tcpcdump y Wireshark, así como sus ventajas y desventajas, teniendo que hacer consultas en distintas páginas confiables para recopilar toda esta información, concluyendo que ambas se pueden complementar.

REFERENCIAS

- Altube, R. (2021). Wireshark: Qué es y ejemplos de uso | OpenWebinars. Recuperado el 27 de septiembre de 2023, de <https://openwebinars.net/blog/wireshark-que-es-y-ejemplos-de-uso/#wireshark:-ventajas-y-desventajas-de-uso>
- Departamento de Automática y Computación. (--). Analizadores de red: Wireshark y tcpdump| tlm. Recuperado el 27 de septiembre de 2023, de https://www.tlm.unavarra.es/~daniel/docencia/arss/arss10_11/practicas/practica3.pdf
- Rojas, J. (2007). TCP Dump | scribd. Recuperado el 27 de septiembre de 2023, de <https://es.scribd.com/document/332350409/Tcp-Dump>
- Studocu. (2022). Redes II | Studocu. Recuperado el 27 de septiembre de 2023, de <https://www.studocu.com/es/document/colegio-la-salle-plasencia/informatica/redes-ii/41016448>

- TCPDUMP. (2023). PÁGINA DE MANUAL TCPDUMP(1) | tcpdump. Recuperado el 27 de septiembre de 2023, de <https://www.tcpdump.org/manpages/tcpdump.1.html>
- Universidad Complutense Madrid. (2013). WireShark | ucm. Recuperado el 27 de septiembre de 2023, de <https://www.ucm.es/pimcd2014-free-software/wireshark#:~:text=Se%20utiliza%20para%20realizar%20an%C3%A1lisis,protocolos%20de%20forma%20%C3%BAnicamente%20hueca.>