Curso de Hacking ético Master. D

Ejercicio 14

ENVENENAR Y/O SUPLANTAR SERVICIOS

Alumno: Julián Gordon

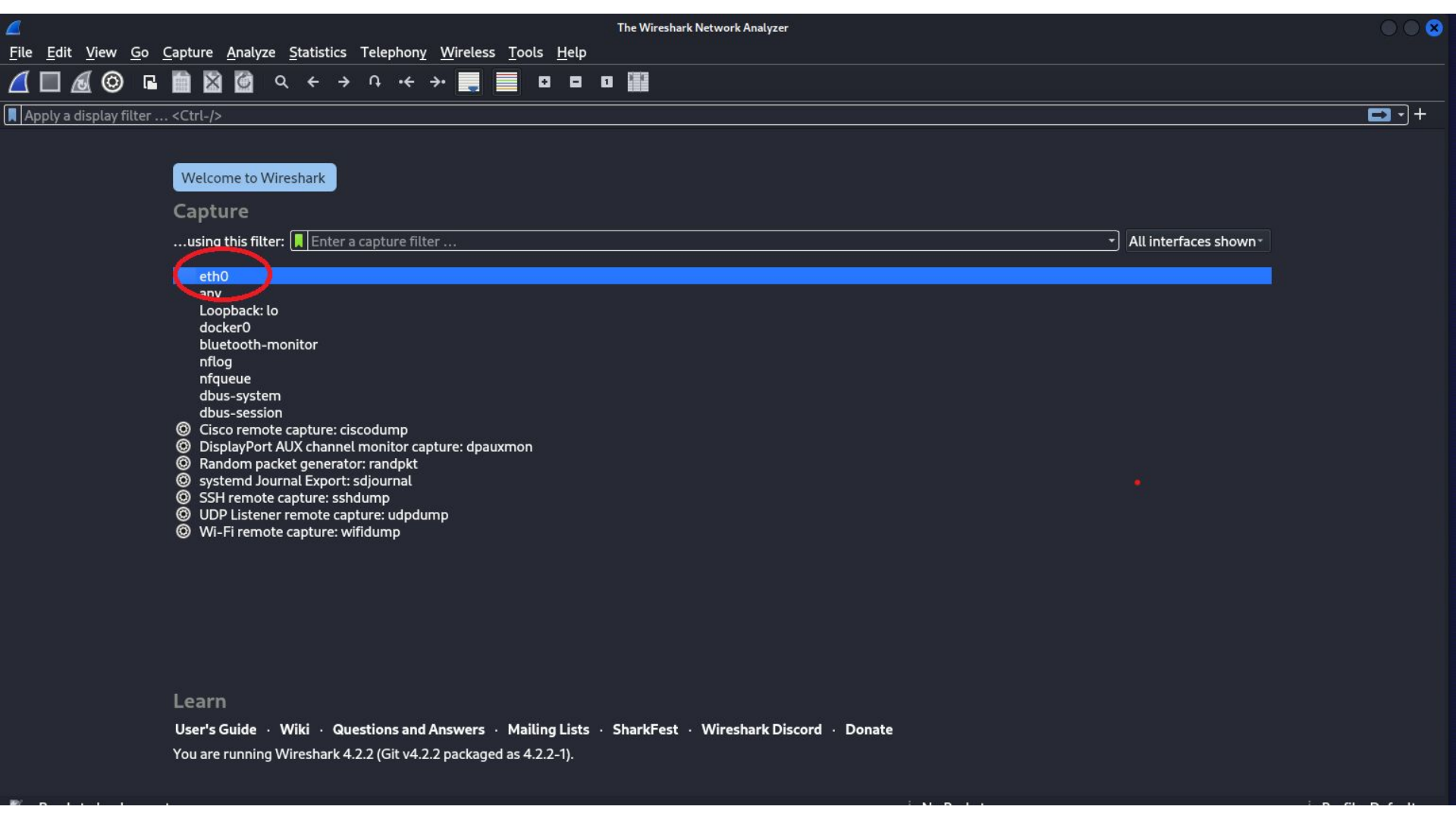
Índice

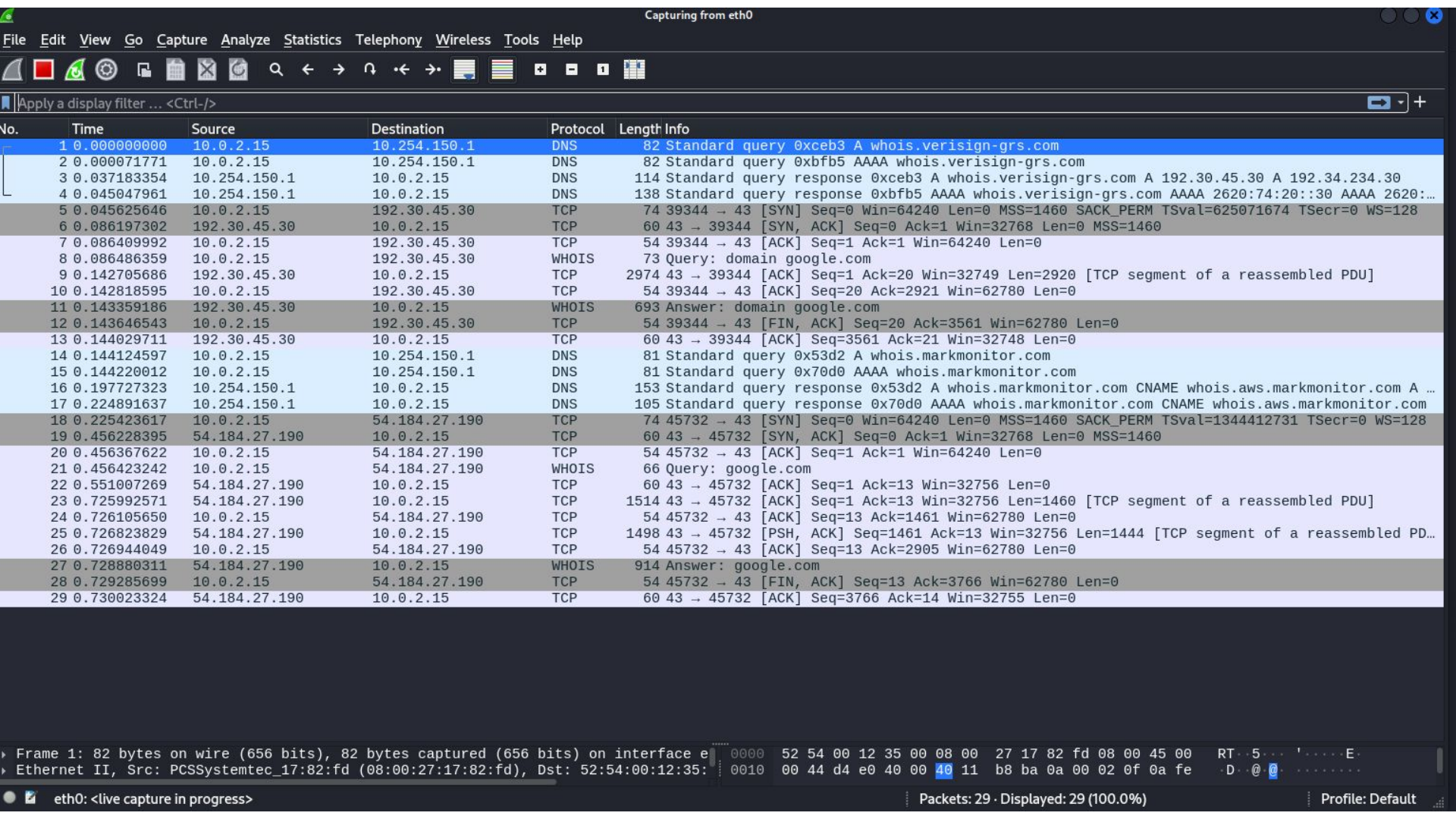
Introducción

En este ejercicio, trabajaremos con WireShark. Es una herramienta de análisis de protocolos de red, que nos permite capturar y analizar el tráfico de red en tiempo real. Funciona capturando los paquetes de datos que pasan a través de una red, y luego nos muestra estos datos en un formato comprensible. Podemos filtrar y examinar los paquetes según diversos criterios, como dirección IP, protocolo, tipo de paquete, etc. Esto nos permite diagnosticar problemas de red, detectar actividades sospechosas, y comprender cómo funcionan los diferentes protocolos de red.

Iniciar esnifado de red con WireShark

Para iniciar Wireshark, lo primero que debemos hacer es elegir la red que vamos a querer capturar el tráfico, en este caso usaremos la eth0. Para empezar a generar tráfico y que lo pueda captar Wireshark, haremos un ‘whois google.com’ . Podemos observar en las siguientes imágenes este proceso, y como se muestra en la interfaz de Wireshark el envío y recepción de los distintos paquetes.

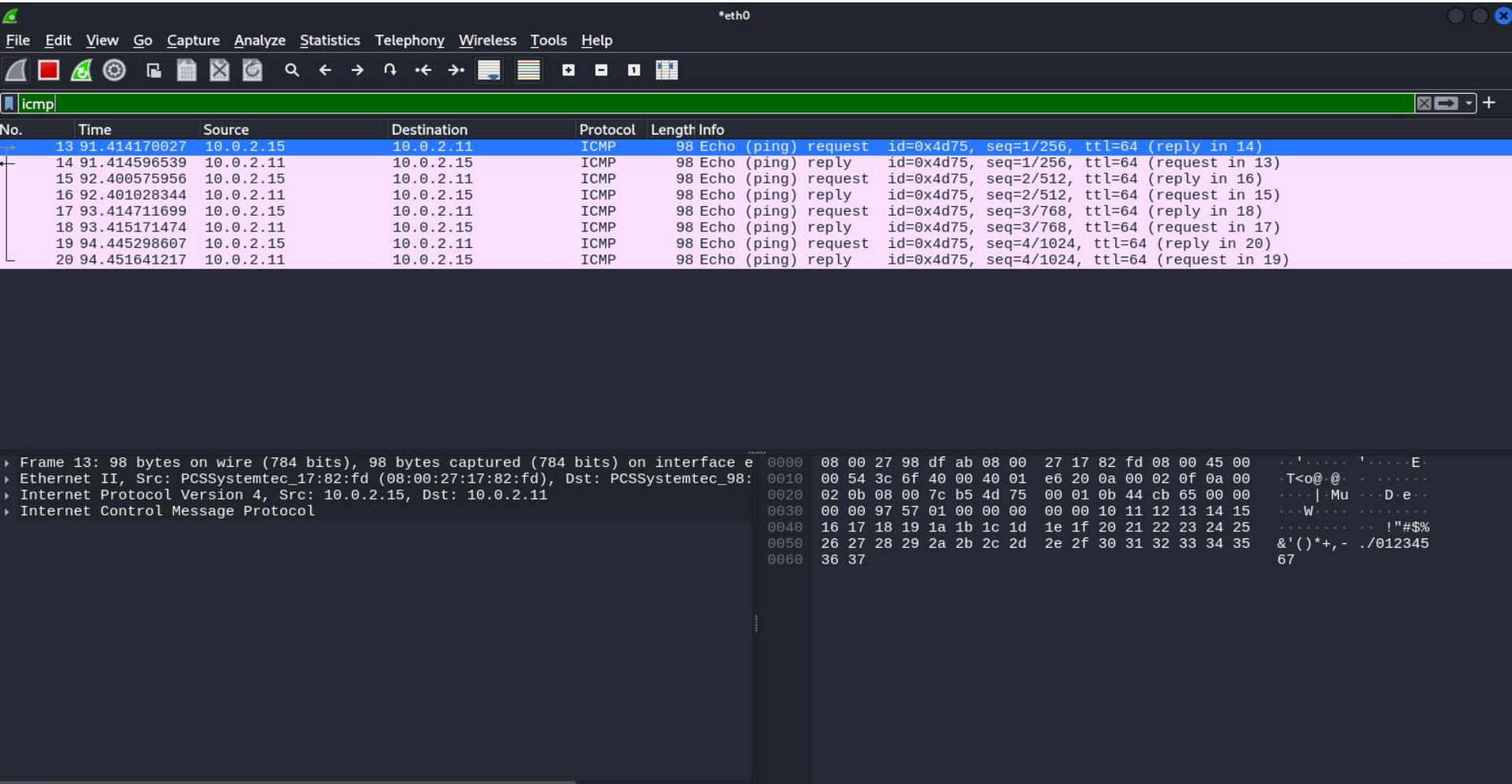

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter: Enter a capture filter ...
All interfaces shown
eth0
any
Loopback: lo
docker0
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
Wi-Fi remote capture: wifidump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate
You are running Wireshark 4.2.2 (Git v4.2.2 packaged as 4.2.2-1).


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 10.0.2.15 10.254.150.1 DNS 82 Standard query 0xceb3 A whois.verisign-grs.com
2 0.000071771 10.0.2.15 10.254.150.1 DNS 82 Standard query 0xbfb5 AAAA whois.verisign-grs.com
3 0.037183354 10.254.150.1 10.0.2.15 DNS 114 Standard query response 0xceb3 A whois.verisign-grs.com A 192.30.45.30 A 192.34.234.30
4 0.045047961 10.254.150.1 10.0.2.15 DNS 138 Standard query response 0xbfb5 AAAA whois.verisign-grs.com AAAA 2620:74:20::30 AAAA 2620:...
5 0.045625646 10.0.2.15 192.30.45.30 TCP 74 39344 → 43 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=625071674 TSecr=0 WS=128
6 0.086197302 192.30.45.30 10.0.2.15 TCP 60 43 → 39344 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
7 0.086409992 10.0.2.15 192.30.45.30 TCP 54 39344 → 43 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8 0.086486359 10.0.2.15 192.30.45.30 WHOIS 73 Query: domain google.com
9 0.142705686 192.30.45.30 10.0.2.15 TCP 2974 43 → 39344 [ACK] Seq=1 Ack=20 Win=32749 Len=2920 [TCP segment of a reassembled PDU]
10 0.142818595 10.0.2.15 192.30.45.30 TCP 54 39344 → 43 [ACK] Seq=20 Ack=2921 Win=62780 Len=0
11 0.143359186 192.30.45.30 10.0.2.15 WHOIS 693 Answer: domain google.com
12 0.143646543 10.0.2.15 192.30.45.30 TCP 54 39344 → 43 [FIN, ACK] Seq=20 Ack=3561 Win=62780 Len=0
13 0.144029711 192.30.45.30 10.0.2.15 TCP 60 43 → 39344 [ACK] Seq=3561 Ack=21 Win=32748 Len=0
14 0.144124597 10.0.2.15 10.254.150.1 DNS 81 Standard query 0x53d2 A whois.markmonitor.com
15 0.144220012 10.0.2.15 10.254.150.1 DNS 81 Standard query 0x70d0 AAAA whois.markmonitor.com
16 0.197727323 10.254.150.1 10.0.2.15 DNS 153 Standard query response 0x53d2 A whois.markmonitor.com CNAME whois.aws.markmonitor.com A ...
17 0.224891637 10.254.150.1 10.0.2.15 DNS 105 Standard query response 0x70d0 AAAA whois.markmonitor.com CNAME whois.aws.markmonitor.com
18 0.225423617 10.0.2.15 54.184.27.190 TCP 74 45732 → 43 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1344412731 TSecr=0 WS=128
19 0.456228395 54.184.27.190 10.0.2.15 TCP 60 43 → 45732 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
20 0.456367622 10.0.2.15 54.184.27.190 TCP 54 45732 → 43 [ACK] Seq=1 Ack=1 Win=64240 Len=0
21 0.456423242 10.0.2.15 54.184.27.190 WHOIS 66 Query: google.com
22 0.551007269 54.184.27.190 10.0.2.15 TCP 60 43 → 45732 [ACK] Seq=1 Ack=13 Win=32756 Len=0
23 0.725992571 54.184.27.190 10.0.2.15 TCP 1514 43 → 45732 [ACK] Seq=1 Ack=13 Win=32756 Len=1460 [TCP segment of a reassembled PDU]
24 0.726105650 10.0.2.15 54.184.27.190 TCP 54 45732 → 43 [ACK] Seq=13 Ack=1461 Win=62780 Len=0
25 0.726823829 54.184.27.190 10.0.2.15 TCP 1498 43 → 45732 [PSH, ACK] Seq=1461 Ack=13 Win=32756 Len=1444 [TCP segment of a reassembled PD...
26 0.726944049 10.0.2.15 54.184.27.190 TCP 54 45732 → 43 [ACK] Seq=13 Ack=2905 Win=62780 Len=0
27 0.728880311 54.184.27.190 10.0.2.15 WHOIS 914 Answer: google.com
28 0.729285699 10.0.2.15 54.184.27.190 TCP 54 45732 → 43 [FIN, ACK] Seq=13 Ack=3766 Win=62780 Len=0
29 0.730023324 54.184.27.190 10.0.2.15 TCP 60 43 → 45732 [ACK] Seq=3766 Ack=14 Win=32755 Len=0
Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface e
Ethernet II, Src: PCSSystemtec_17:82:fd (08:00:27:17:82:fd), Dst: 52:54:00:12:35:
0000 52 54 00 12 35 00 08 00 27 17 82 fd 08 00 45 00
0010 00 44 d4 e0 40 00 40 11 b8 ba 0a 00 02 0f 0a fe
eth0: <live capture in progress>
Packets: 29 · Displayed: 29 (100.0%)
Profile: Default

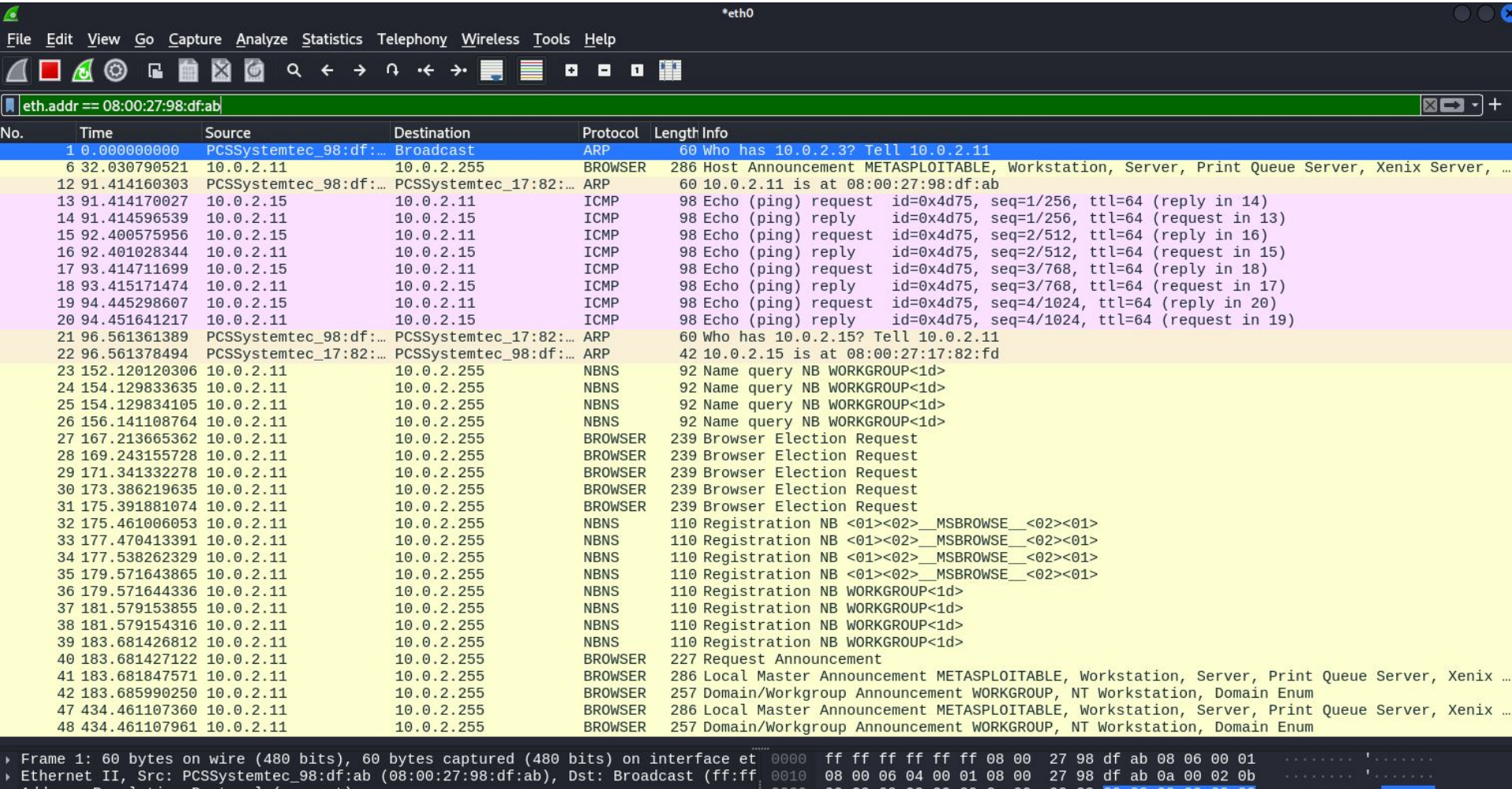
Uso de filtros con WireShark

Para poder mostrar el uso de los filtros de WireShark, en este ejercicio, generamos distintos tipos de tráfico en nuestra red. Empezamos haciendo un ping desde nuestra máquina de Kali Linux, sobre nuestro objetivo, que será la máquina de Metasploitable2. El comando ping, utiliza el Protocolo de Control de Mensajes de Internet (ICMP) para enviar paquetes de solicitud y recibir respuestas de un dispositivo. Para filtrar este proceso, usaremos el filtro por protocolo 'ICMP' . Luego usamos otro filtro, que será por dirección MAC, para ello se necesita este comando 'eth.addr == {MAC de nuestro objetivo}' . Para saber la MAC de nuestro objetivo, podemos usar el comando 'arp -a' y nos dirá la MAC asociada a la IP. Ahora filtraremos por puerto destino(en este caso telnet que es el 23) con el comando 'tcp.port == 23' , para generar este tráfico, solamente con usar el comando 'telnet + ip' con la contraseña que ya obtuvimos en ejercicios anteriores. También podemos filtrar por dirección IP de origen o de destino, con los comandos 'ip.src == x.x.x.x' y 'ip.dst == x.x.x.x. Además también podemos combinar dichos filtros con 'or' ó 'and' ó negarlo con 'not'.
A continuación se muestran las imágenes de todos los procesos.

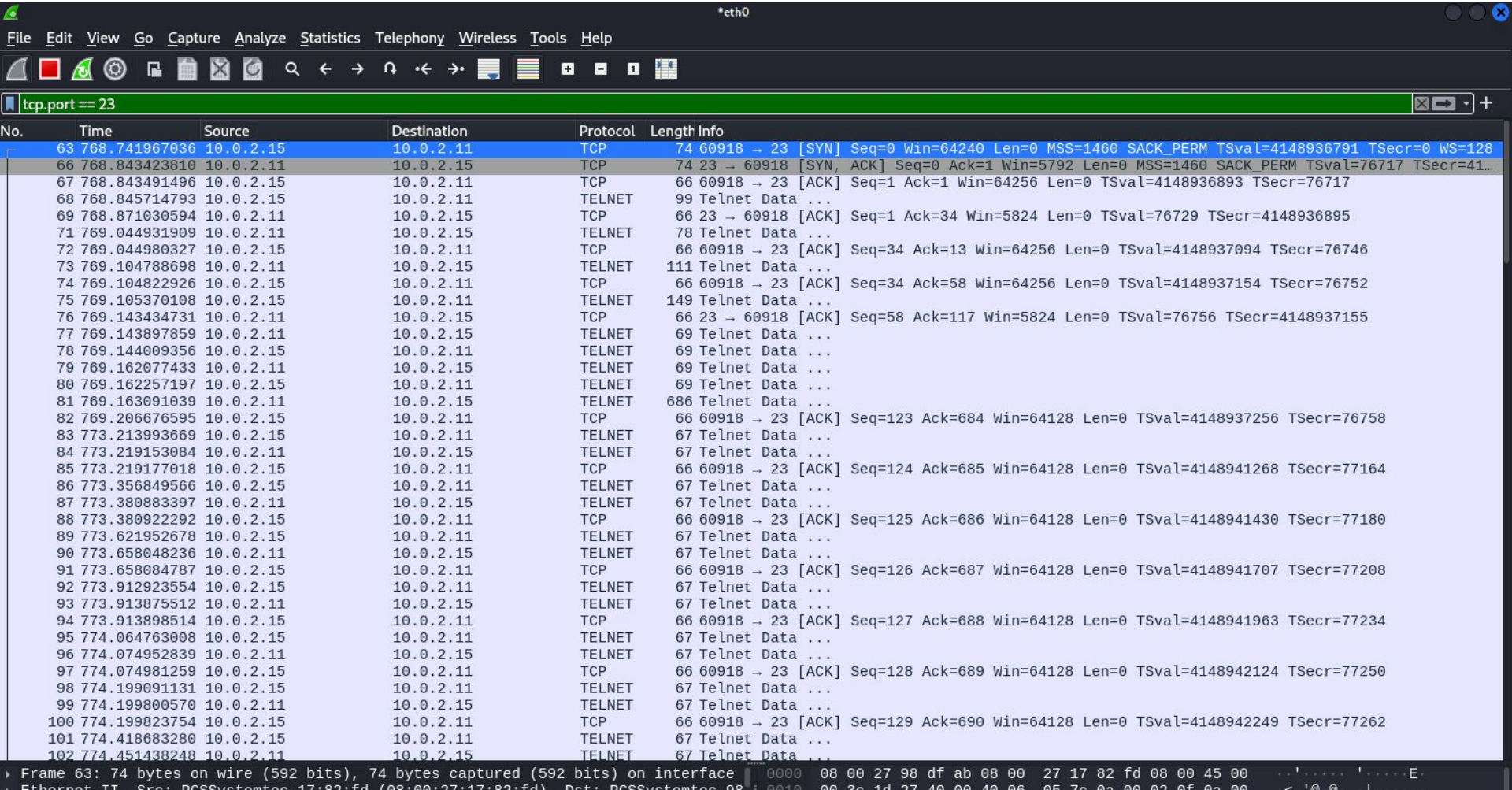
Filtro por protocolo



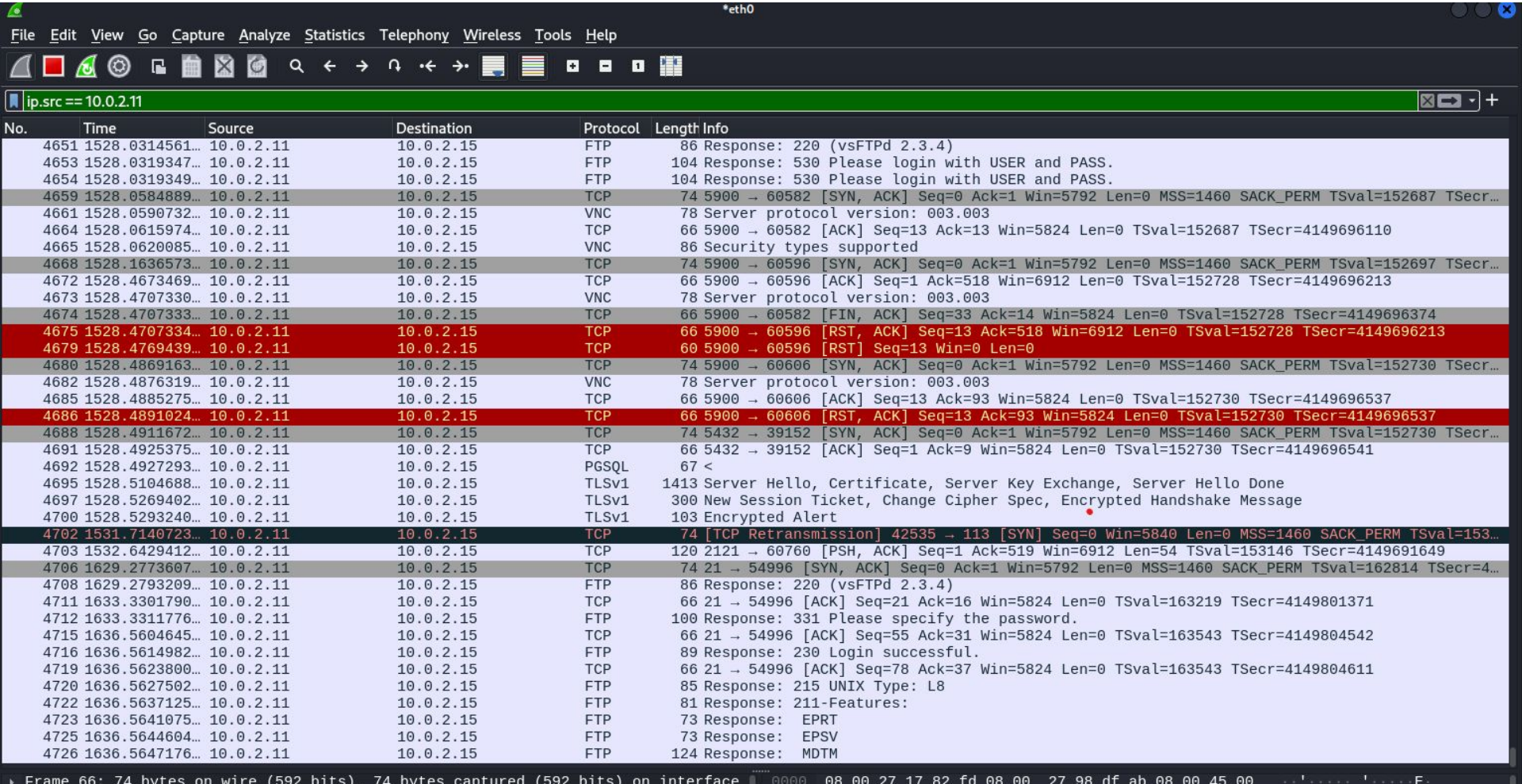
Filtro por dirección MAC



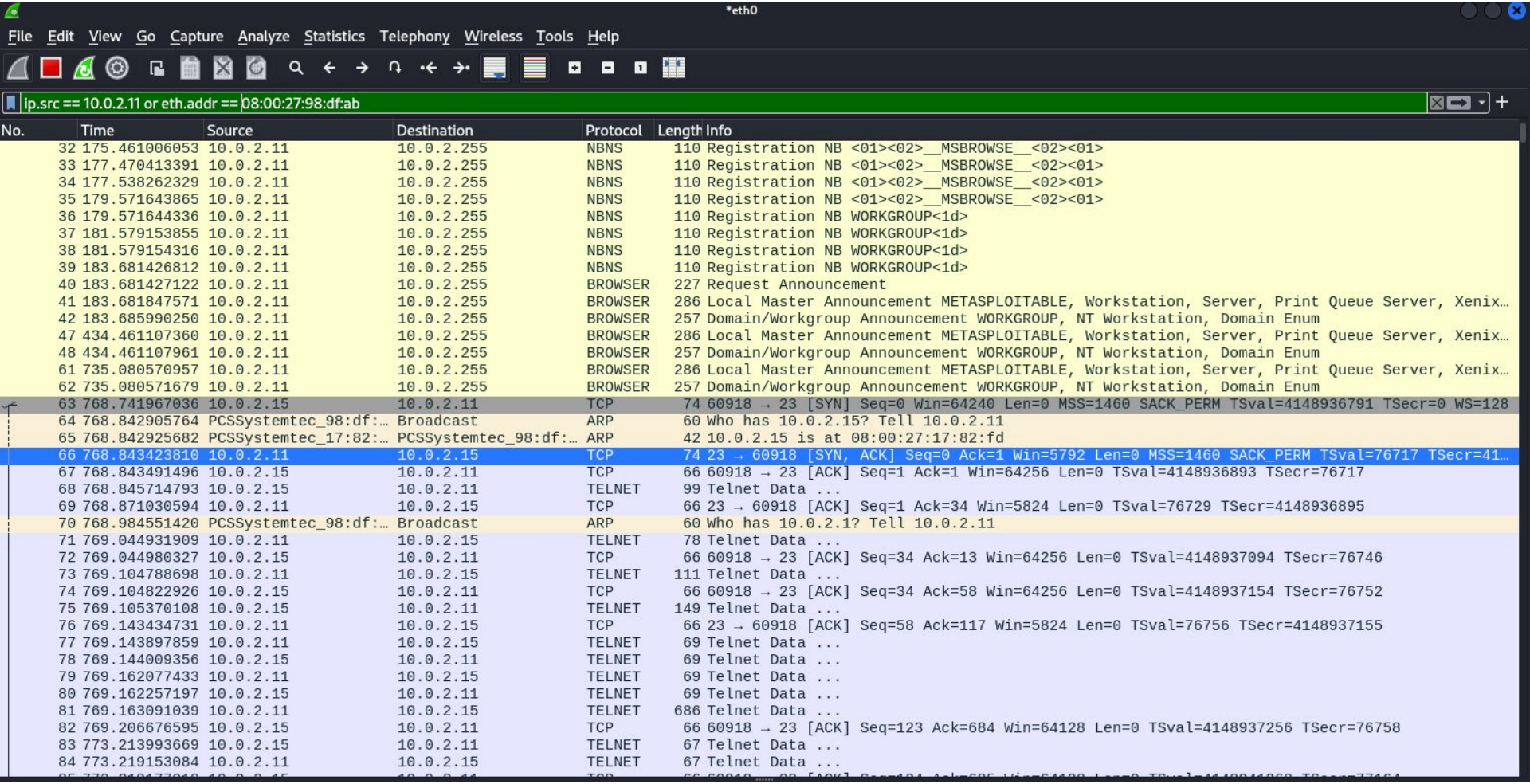
Filtro por puerto



Filtro por dirección IP



Filtro por dirección IP y dirección MAC combinados



Conclusiones

En este ejercicio, Wireshark demostró ser una herramienta esencial para analizar el tráfico de red en tiempo real, vimos la capacidad que tiene de filtrar paquetes según diversos criterios y obtuvimos una visión profunda de nuestra red. Podríamos detectar actividades sospechosas, al filtrar por dirección MAC y puerto destino, en distintas auditorías que podríamos llegar a realizar en el futuro. Podemos también, identificar rápidamente problemas como latencia y pérdida de paquetes. Wireshark es una herramienta imprescindible para, analizar y diagnosticar problemas, y mejorar la seguridad en redes.