

SOC (Security Operations Center)

Módulo 3



Networking – Wireshark Ejercicios

Sheila Fernández Cisneros – 11/06/2024

Ejercicios de redes

Wireshark es una herramienta de análisis de protocolos de red muy utilizada en el ámbito de las redes. Permite capturar y examinar el tráfico que pasa por una red en tiempo real. En el mundo de la ciberseguridad este tipo de herramientas son muy interesantes, debido a que permiten detectar intentos de intrusión, analizar todo el tráfico que se considere sospechoso y actividades maliciosas. También permite identificar problemas en las redes, ya sean de rendimiento o de fallos y es muy útil a nivel educativo para ver el funcionamiento de algunos protocolos.

Esta herramienta viene instalada por defecto en algunas distribuciones de Linux, como es el caso de Kali. Por ello, se recomienda realizar los siguientes ejercicios desde la máquina virtual “KaliLinux”. Para abrir dicha aplicación, se puede escribir en la terminal el comando “wireshark”.

Tarea 1:

Como se ha visto de manera teórica, los dos protocolos principales de transporte son UDP y TCP. La principal característica de TCP es el Three-Way Handshake que se produce al principio de la sesión entre dos dispositivos. Este procedimiento asegura que ambos estén listos para comunicarse y acuerden parámetros importantes de la conexión. Para analizar tráfico TCP, descargar el siguiente archivo con extensión .pcapng que se encuentra en el enlace que se proporciona al final del enunciado. Una vez descargado el archivo, hay que abrirlo en Wireshark. Para ello se le da clic en “File > Open” y se selecciona el archivo que se acaba de descargar. La tarea pide identificar los mensajes intercambiados en el Three-Way Handshake. Aportar una captura de pantalla de los mensajes correspondientes, con la dirección IP de origen y destino en cada uno de estos mensajes, además de los puertos y las direcciones MAC de origen y destino.

https://wiki.wireshark.org/uploads/_moin_import_/attachments/SampleCaptures/200722_with_scale_examples_anon.pcapng

El **Three-Way Handshake** es un proceso utilizado por el protocolo TCP (Transmission Control Protocol) para establecer una conexión entre un cliente y un servidor. Durante este proceso, se intercambian tres tipos de paquetes: **SYN**, **SYN-ACK**, y **ACK**. Aquí está el significado de cada uno:

SYN (Synchronize)

- **Función:** El paquete SYN (Synchronize) es enviado por el cliente al servidor para iniciar una conexión. Este paquete incluye un número de secuencia inicial, que el cliente utilizará para identificar esta conexión.

- **Propósito:** Indica que el cliente desea establecer una conexión TCP y sincronizar los números de secuencia con el servidor.

SYN-ACK (Synchronize-Acknowledge)

- **Función:** El paquete SYN-ACK es enviado por el servidor en respuesta al paquete SYN del cliente. Este paquete incluye el número de secuencia inicial del servidor y un número de acuse de recibo (ACK) para confirmar la recepción del SYN del cliente.
- **Propósito:** Indica que el servidor ha recibido el paquete SYN del cliente y está dispuesto a establecer la conexión. También sincroniza los números de secuencia del servidor con el cliente.

ACK (Acknowledge)

- **Función:** El paquete ACK es enviado por el cliente en respuesta al paquete SYN-ACK del servidor. Este paquete incluye un número de acuse de recibo (ACK) para confirmar la recepción del SYN-ACK del servidor.
- **Propósito:** Completa el proceso de establecimiento de la conexión, confirmando que ambos, cliente y servidor, han recibido y reconocido los números de secuencia iniciales del otro.

Visualización en Wireshark

En Wireshark, los paquetes correspondientes al Three-Way Handshake pueden identificarse con las siguientes etiquetas:

1. **SYN:** Verás un paquete TCP con la bandera SYN activada.
2. **SYN-ACK:** Verás un paquete TCP con las banderas SYN y ACK activadas.
3. **ACK:** Verás un paquete TCP con la bandera ACK activada.

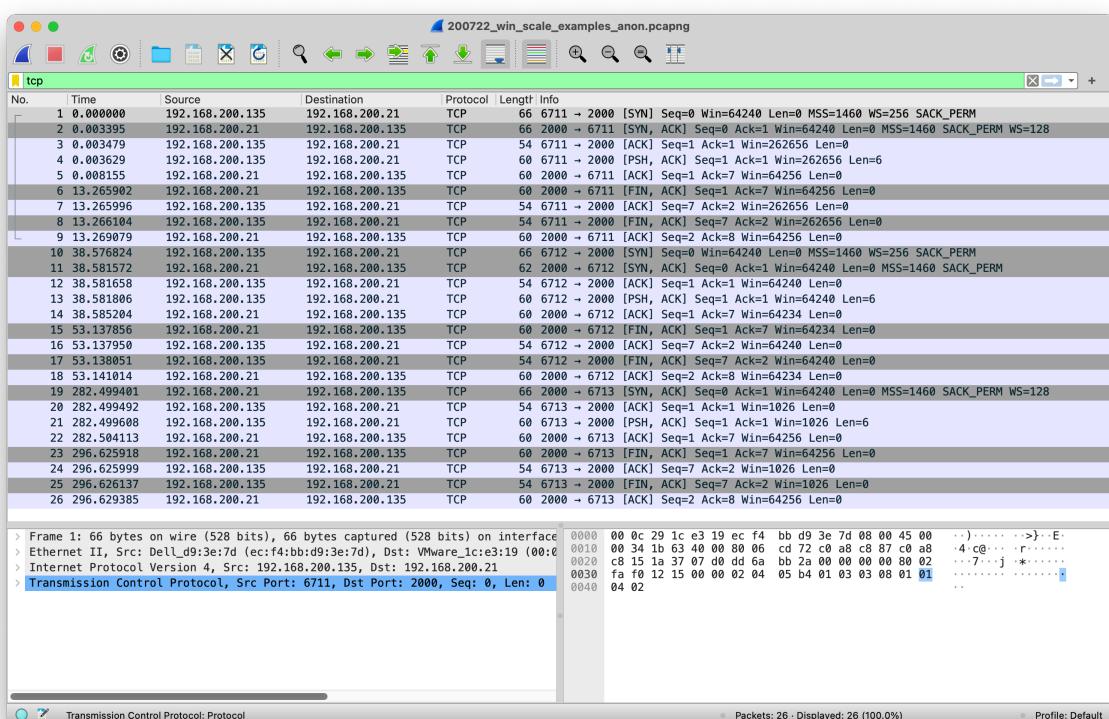
El Three-Way Handshake es esencial para establecer una conexión TCP fiable, asegurando que tanto el cliente como el servidor estén sincronizados y listos para intercambiar datos. Las banderas SYN y ACK en los paquetes juegan un papel crucial en este proceso, permitiendo la sincronización de los números de secuencia y la confirmación de la recepción de los mensajes.

Pasos para realizar la tarea:

- **Abrir Wireshark y cargar el archivo .pcapng como se indica en el enunciado:**
 - Abrir Wireshark.
 - Nos dirigimos al menú superior para seleccionar “File > Open” para abrir el archivo descargado. Seleccionar el archivo .pcapng descargado.
- **Usar la barra de filtros:**

- La barra de filtros está ubicada en la parte superior de la ventana principal de Wireshark, justo debajo de las pestañas de menú (File, Edit, View, etc.).
 - Hacer clic en la barra de filtros y escribir “tcp” para filtrar solo el tráfico TCP. Luego, presionar Enter.
- **Identificar los mensajes del Three-Way Handshake:**
- Nos desplazamos por la lista de paquetes filtrados hasta encontrar los paquetes que forman parte del Three-Way Handshake. Son los tres primeros paquetes.
 - Los mensajes son etiquetados como [SYN], [SYN, ACK], y [ACK].
- **Captura de pantalla de cada mensaje:**
- Hacer clic en el primer paquete que contiene el flag [SYN].
 - En la parte superior de la ventana, encontramos los detalles del paquete, donde podemos ver las direcciones IP de origen y destino, los puertos y las direcciones MAC.
 - Luego repetimos el proceso para los paquetes [SYN, ACK] y [ACK].

Para realizar este ejercicio se ha usado la aplicación de escritorio de wireshark para MacOS.

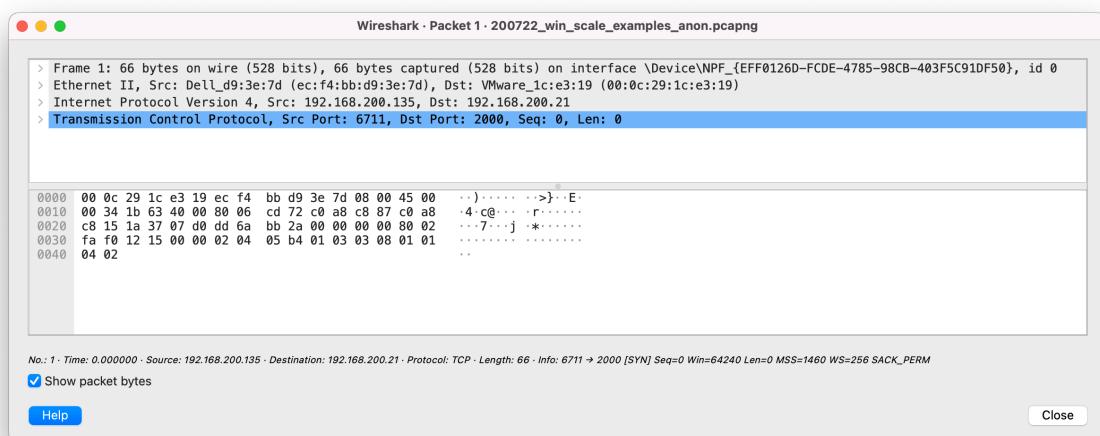


Identificación de los Paquetes del Three-Way Handshake

Para poder obtener la información requerida, hacemos doble click en los 3 primeros paquetes que aparecen en el listado y obtenemos así los detalles.

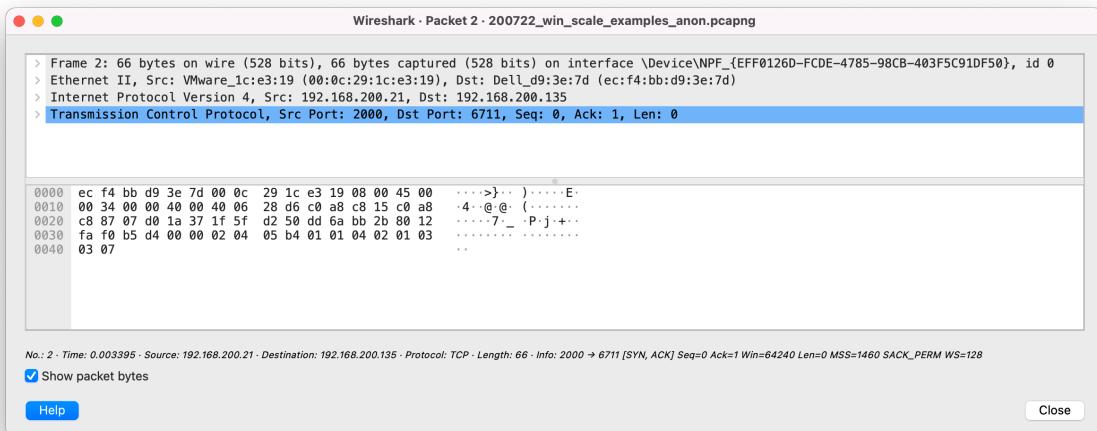
1. **Paquete SYN:** Información del paquete SYN, que es el primer paso del Three-Way Handshake.

- **Dirección IP de origen:** 192.168.200.135
- **Dirección IP de destino:** 192.168.200.21
- **Puerto de origen:** 6711
- **Puerto de destino:** 2000
- **Dirección MAC de origen:** Dell_d9:3e:7d (ec:f4:bb:d9:3e:7d)
- **Dirección MAC de destino:** VMWare_1c:e3:19 (00:0c:29:1c:e3:19)
- **Descripción:** El cliente (192.168.200.135) envía un paquete SYN al servidor (192.168.200.21) para iniciar la conexión. Este paquete incluye el número de secuencia inicial del cliente.



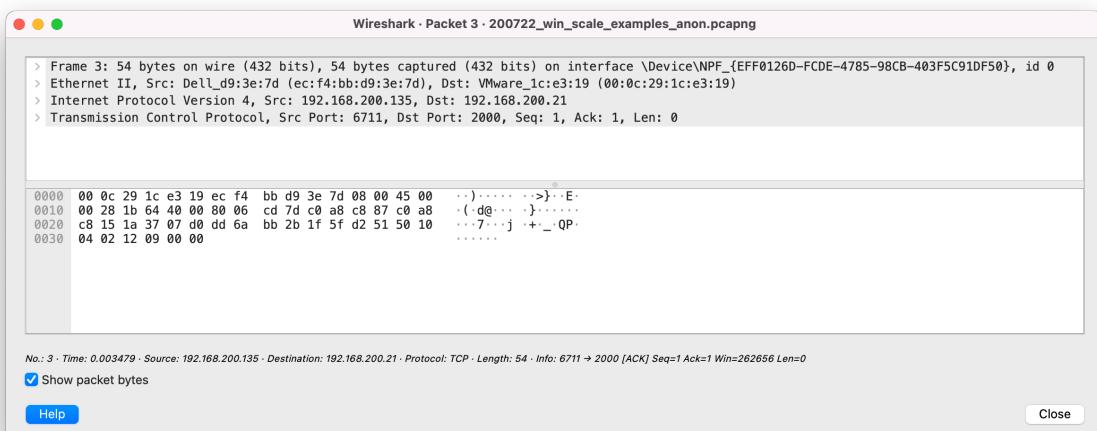
2. **Paquete SYN-ACK:** Información del paquete SYN-ACK, que es el segundo paso del Three-Way Handshake.

- **Dirección IP de origen:** 192.168.200.21
- **Dirección IP de destino:** 192.168.200.135
- **Puerto de origen:** 2000
- **Puerto de destino:** 6711
- **Dirección MAC de origen:** VMWare_1c:e3:19 (00:0c:29:1c:e3:19)
- **Dirección MAC de destino:** Dell_d9:3e:7d (ec:f4:bb:d9:3e:7d)
- **Descripción:** El servidor (192.168.200.21) responde con un paquete SYN-ACK, que incluye su propio número de secuencia inicial y un acuse de recibo para el número de secuencia del cliente.



3. Paquete ACK: Información del paquete ACK, que es el tercer paso del Three-Way Handshake.

- **Dirección IP de origen:** 192.168.200.135
- **Dirección IP de destino:** 192.168.200.21
- **Puerto de origen:** 6711
- **Puerto de destino:** 2000
- **Dirección MAC de origen:** Dell_d9:3e:7d (ec:f4:bb:d9:3e:7d)
- **Dirección MAC de destino:** VMware_1c:e3:19 (00:0c:29:1c:e3:19)
- **Descripción:** El cliente (192.168.200.135) responde con un paquete ACK para confirmar la recepción del SYN-ACK del servidor. Este paquete completa el proceso de establecimiento de la conexión.



Descripción del Proceso Completo

1. Cliente -> Servidor: SYN

- El cliente inicia la conexión enviando un paquete SYN al servidor. Este paquete incluye el número de secuencia inicial del cliente, lo que indica su disposición para comenzar la comunicación.
- **Paquete 1:** SYN, Seq=0

2. Servidor -> Cliente: SYN-ACK

- El servidor recibe el paquete SYN y responde con un paquete SYN-ACK. Este paquete contiene el número de secuencia inicial del servidor y un acuse de recibo para el número de secuencia del cliente.
- **Paquete 2:** SYN-ACK, Seq=0, Ack=1

3. Cliente -> Servidor: ACK

- El cliente recibe el paquete SYN-ACK y responde con un paquete ACK. Este paquete confirma la recepción del SYN-ACK del servidor y completa el establecimiento de la conexión.
- **Paquete 3:** ACK, Seq=1, Ack=1

El Three-Way Handshake observado en Wireshark sigue el proceso estándar de establecimiento de una conexión TCP. Los paquetes SYN, SYN-ACK, y ACK se intercambian correctamente entre el cliente y el servidor, lo que permite una conexión confiable y ordenada.

Tarea 2:

Una vez vistos los mensajes intercambiados en el Three-Way Hanshake, se pide comprobar en la misma captura de tráfico que en el ejercicio anterior, cuáles son los mensajes de cierre de sesión que se producen, indicando también cuál manda cada uno (dirección IP, MAC y puerto de origen) y quién lo recibe (dirección IP, MAC y puerto de destino). Adjuntar captura de pantalla en la que se muestren estos datos.

El cierre de una conexión TCP también se realiza mediante un proceso de intercambio de mensajes, conocido como el **Four-Way Handshake** o **Four-Way Termination**. Este proceso utiliza los paquetes **FIN** y **ACK** para cerrar una conexión de manera ordenada entre dos dispositivos. Aquí se detallan los pasos y el significado de cada tipo de paquete:

FIN (Finish)

- **Función:** El paquete FIN (Finish) es enviado por una de las partes (cliente o servidor) para indicar que ya no tiene más datos que enviar y desea cerrar la conexión.
- **Propósito:** Inicia el proceso de cierre de la conexión TCP.

ACK (Acknowledge)

- **Función:** El paquete ACK (Acknowledge) es enviado por la parte receptora en respuesta a un paquete FIN para confirmar su recepción.
- **Propósito:** Confirma que el paquete FIN ha sido recibido.

Visualización del Proceso en Wireshark

En Wireshark, el cierre de la conexión TCP puede identificarse con las siguientes etiquetas:

1. **FIN**: Verás un paquete TCP con la bandera FIN activada.
2. **ACK**: Verás un paquete TCP con la bandera ACK activada.
3. **FIN-ACK**: En algunos casos, verás un paquete combinado FIN-ACK cuando una parte envía un FIN y ACK en el mismo paquete.

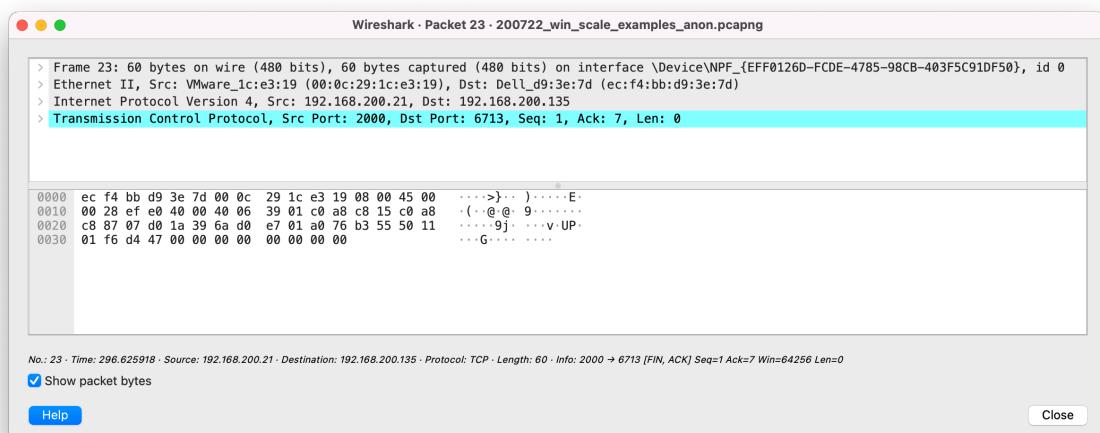
A veces, en lugar de ver paquetes separados para FIN y ACK, puedes ver un **paquete combinado FIN-ACK**. Esto puede ocurrir para optimizar la transmisión.

Realizamos lo mismo que en el ejercicio anterior, pero en este caso, los cuatro últimos paquetes.

Análisis del Cierre de Sesión en TCP

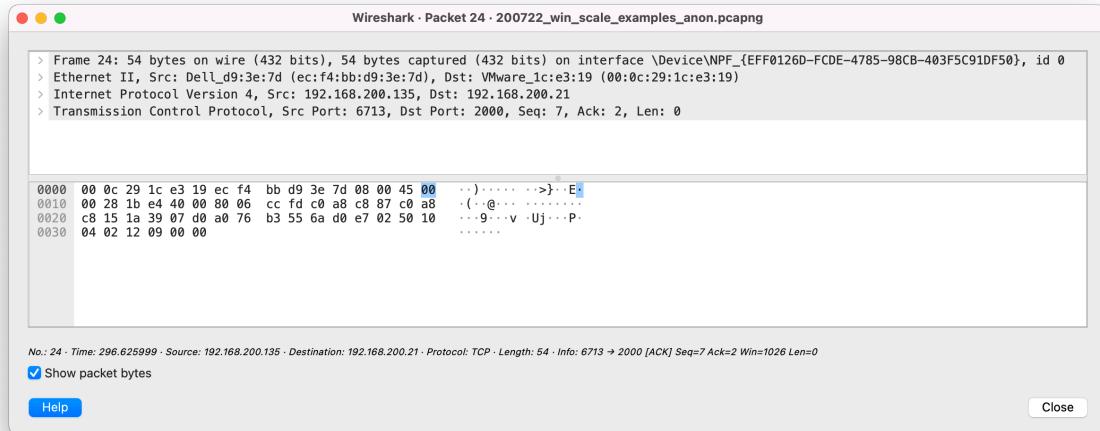
1. Paquete FIN, ACK Inicial: paquete 17

- **Dirección IP de origen**: 192.168.200.21
- **Dirección IP de destino**: 192.168.200.135
- **Puerto de origen**: 2000
- **Puerto de destino**: 6713
- **Dirección MAC de origen**: VMware_1c:e3:19 (00:0c:29:1c:e3:19)
- **Dirección MAC de destino**: Dell_d9:3e:7d (ec:f4:bb:d9:3e:7d)
- **Descripción**: El cliente (192.168.200.21) envía un paquete FIN, ACK indicando que ha terminado de enviar datos.



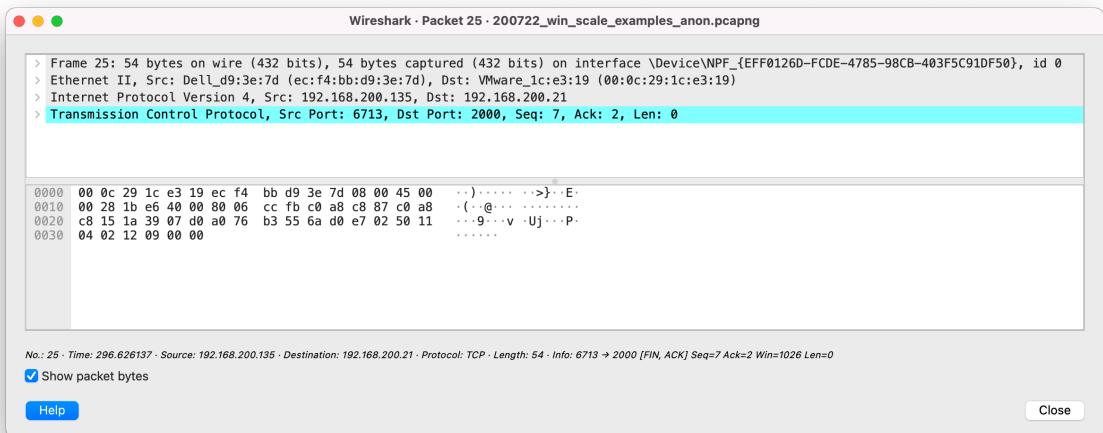
2. Paquete ACK de Confirmación: paquete 18

- **Dirección IP de origen:** 192.168.200.135
- **Dirección IP de destino:** 192.168.200.21
- **Puerto de origen:** 6713
- **Puerto de destino:** 2000
- **Dirección MAC de origen:** Dell_d9:3e:7d (ec:f4:bb:d9:3e:7d)
- **Dirección MAC de destino:** VMWare_1c:e3:19 (00:0c:29:1c:e3:19)
- **Descripción:** El servidor (192.168.200.135) responde con un ACK para confirmar la recepción del FIN del cliente.



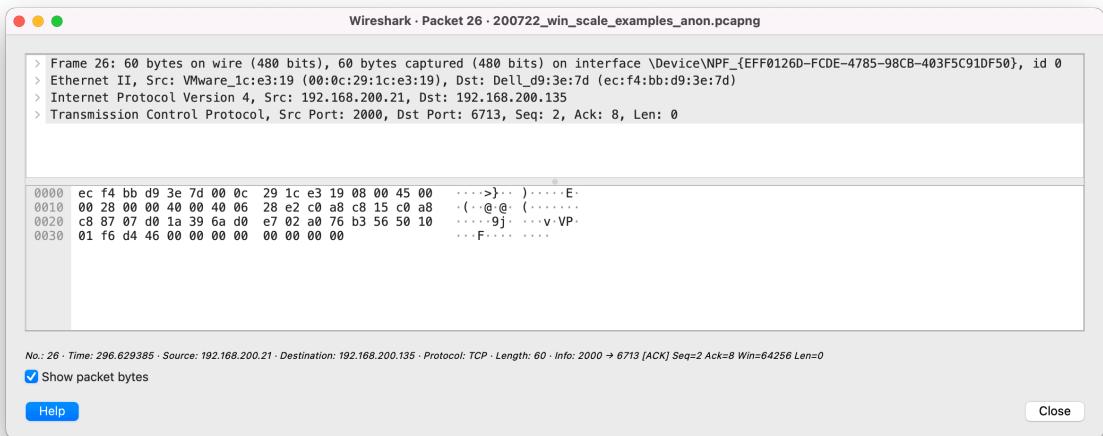
3. Paquete FIN, ACK del Segundo Extremo: paquete 19

- **Dirección IP de origen:** 192.168.200.135
- **Dirección IP de destino:** 192.168.200.21
- **Puerto de origen:** 6713
- **Puerto de destino:** 2000
- **Dirección MAC de origen:** Dell_d9:3e:7d (ec:f4:bb:d9:3e:7d)
- **Dirección MAC de destino:** VMWare_1c:e3:19 (00:0c:29:1c:e3:19)
- **Descripción:** El servidor (192.168.200.135) envía un FIN, ACK indicando que también ha terminado de enviar datos.



4. Paquete ACK Final: paquete 20

- **Dirección IP de origen:** 192.168.200.21
- **Dirección IP de destino:** 192.168.200.135
- **Puerto de origen:** 2000
- **Puerto de destino:** 6713
- **Dirección MAC de origen:** VMware_1c:e3:19 (00:0c:29:1c:e3:19)
- **Dirección MAC de destino:** Dell_d9:3e:7d (ec:f4:bb:d9:3e:7d)
- **Descripción:** El cliente (192.168.200.21) responde con un ACK para confirmar la recepción del FIN del servidor, completando así el proceso de cierre de la conexión.



Protocolo Usado: TCP

Secuencia de Cierre:

- **FIN-ACK:** Cliente -> Servidor
- **ACK:** Servidor -> Cliente

- **FIN-ACK:** Servidor -> Cliente
- **ACK:** Cliente -> Servidor

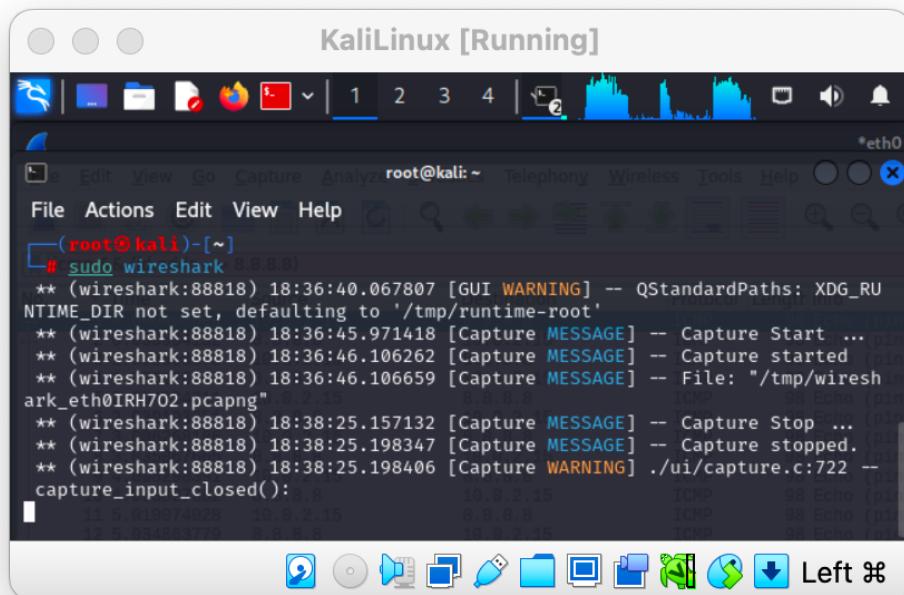
El cierre de la conexión TCP mostrado en Wireshark es correcto. Utiliza paquetes FIN-ACK y ACK para asegurar que tanto el cliente como el servidor han terminado de enviar y recibir datos de manera ordenada. La secuencia que se observa es una implementación válida del protocolo TCP para cerrar una conexión.

Tarea 3:

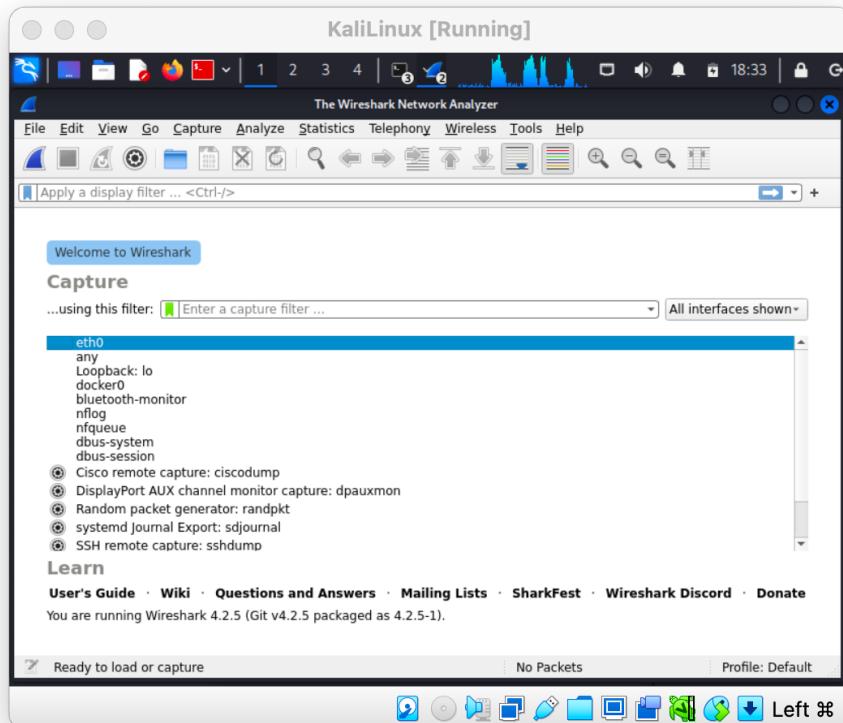
En los ejercicios anteriores se ha utilizado una captura de tráfico proporcionada por Wireshark, preparada para tener únicamente el tráfico requerido. Sin embargo, también es posible analizar el tráfico de nuestra propia red. No obstante, de esta manera es necesario utilizar filtros que permitan mostrar únicamente el tráfico que queremos analizar. Estos nos permiten filtrar por direcciones IP de origen o destino, por protocolos, puertos, etc. La presente tarea requiere que se abra Wireshark y se comience a capturar el tráfico de la interfaz “eth0”. Tras esto, hay que realizar las siguientes tareas:

- a) Abrir una terminal y realizar un ping a la dirección 8.8.8.8.
- b) Anotar la cantidad de mensajes que se envían con el ping (para parar de enviar mensajes se puede cortar con la combinación Ctrl + C) que se hacen a esa dirección.
- c) Dentro de Wireshark, detener la captura con el botón “stop”.
- d) Realizar un filtro que capture únicamente el tráfico que pertenezca a la dirección IP 8.8.8.8 y que además sea del protocolo ICMP.

Para resolver este ejercicio, lo hemos hecho en Kali Linux para practicar con el otro sistema operativo, en esta ocasión no tenemos una aplicación de escritorio sino que accedemos a través de terminal con privilegios root. Usamos el comando “sudo wireshark” para acceder.



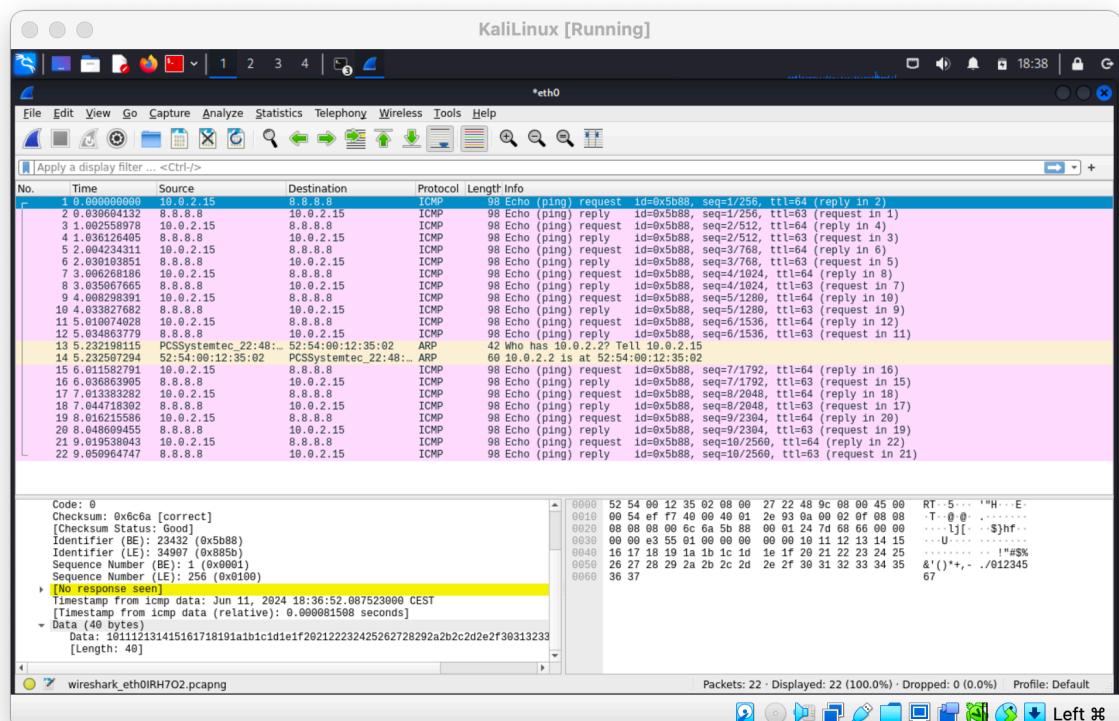
Al hacer sudo accedemos a wireshark con privilegios de root, a continuación seleccionamos la interfaz, en este caso “eth0”.



Realizamos “ping 8.8.8.8” en otra terminal y observamos, que tanto en la terminal como en wireshark, que son 10 los mensajes que se envían tras el ping a 8.8.8.8.

```
(kali㉿kali)-[~]
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=30.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=33.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=26.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=28.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=63 time=25.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=63 time=24.8 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=63 time=25.4 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=63 time=31.4 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=63 time=32.5 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=63 time=31.5 ms
^C
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9020ms
rtt min/avg/max/mdev = 24.823/29.058/33.637/3.140 ms

(kali㉿kali)-[~]
```

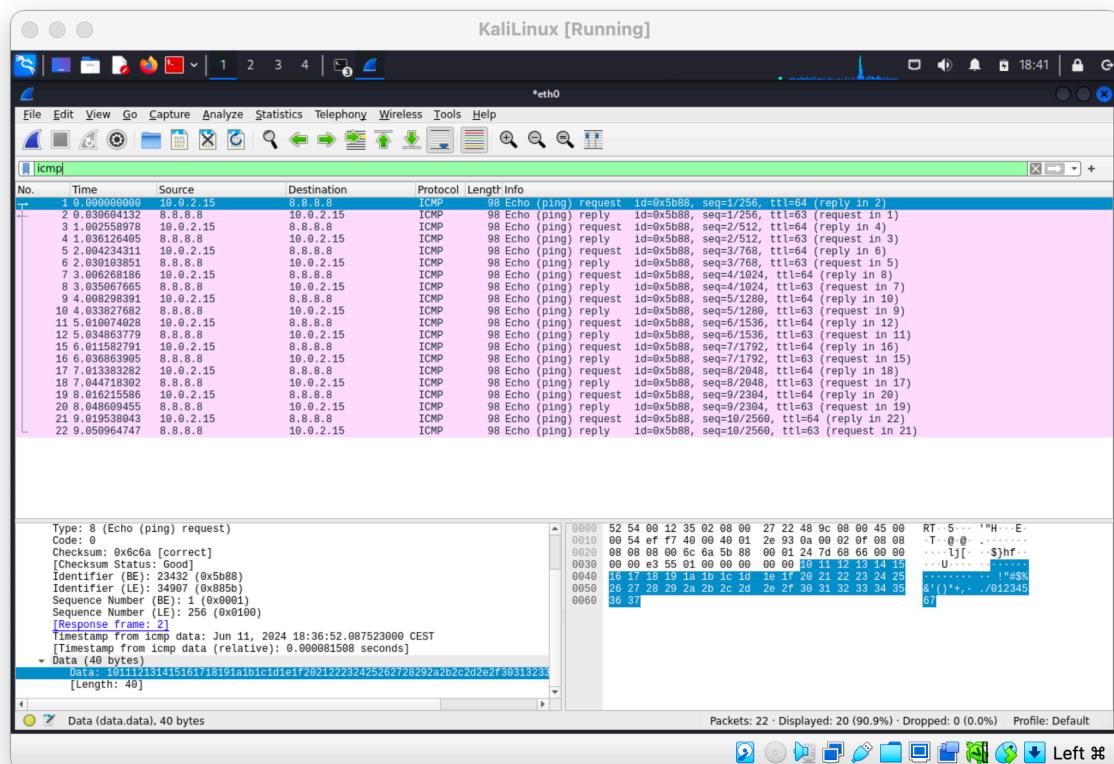


A continuación, lo filtramos por el protocolo ICMP. El Protocolo de Mensajes de Control de Internet, es un protocolo de la suite de protocolos de Internet que se utiliza principalmente para enviar mensajes de control y errores en la red. ICMP es fundamental para la gestión y operación de redes IP.

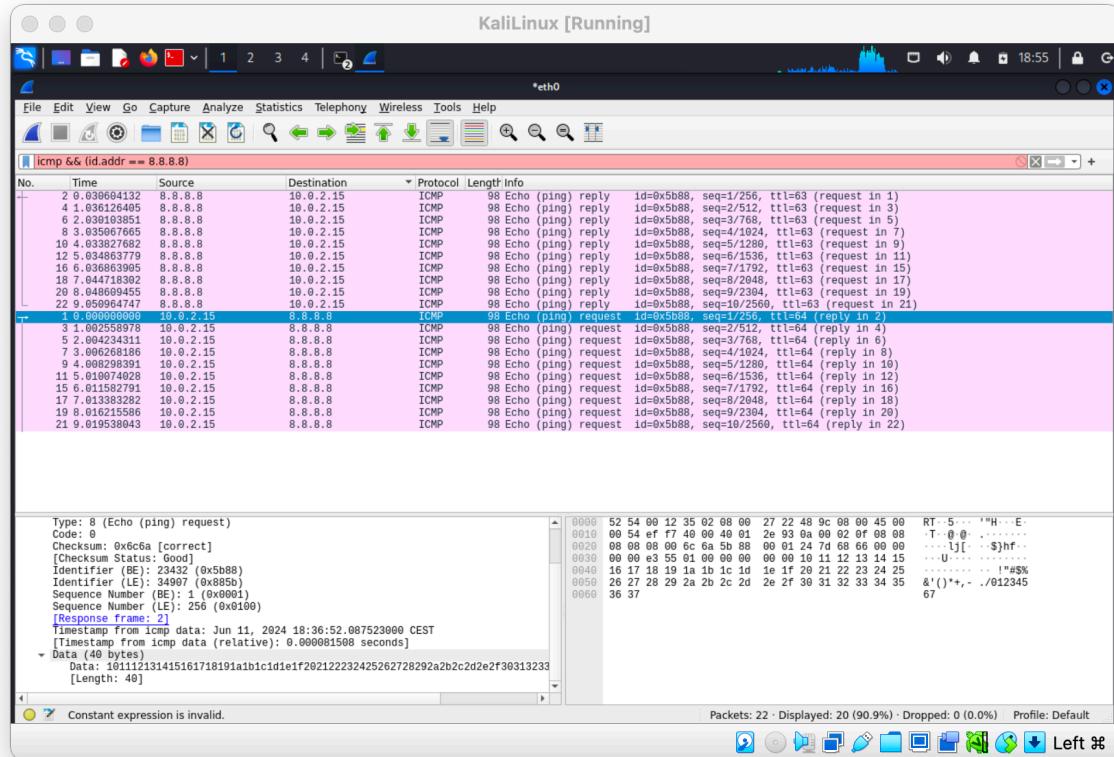
Funciones y características principales:

1. **Diagnóstico de red:** ICMP se utiliza para diagnosticar problemas de red. Una de las herramientas más conocidas que utiliza ICMP es el comando ping, que envía paquetes ICMP Echo Request y espera recibir ICMP Echo Reply, permitiendo verificar la conectividad entre dispositivos.
2. **Mensajes de error:** ICMP informa sobre errores en la entrega de paquetes IP. Por ejemplo, si un router no puede reenviar un paquete debido a una ruta inalcanzable, enviará un mensaje ICMP Destination Unreachable al remitente del paquete.
3. **Gestión de tráfico:** ICMP puede ayudar a gestionar el tráfico en una red. Por ejemplo, los mensajes ICMP Redirect informan a un host que utilice una ruta diferente para un destino específico.
4. **Control de flujo:** ICMP también incluye mensajes de control de flujo como el Time Exceeded, que indica que el TTL (Time to Live) de un paquete ha expirado, y el Source Quench, que solía indicar que un remitente debía reducir la velocidad de envío de datos (aunque este mensaje está obsoleto).
5. **Estructura de mensajes ICMP:** Los mensajes ICMP tienen una estructura simple que incluye un tipo de mensaje, un código de mensaje y una suma de comprobación, junto con información específica del tipo de mensaje.

ICMP es esencial para la operación eficiente y el diagnóstico de redes IP, proporcionando mecanismos para la detección y solución de problemas de comunicación en la red.

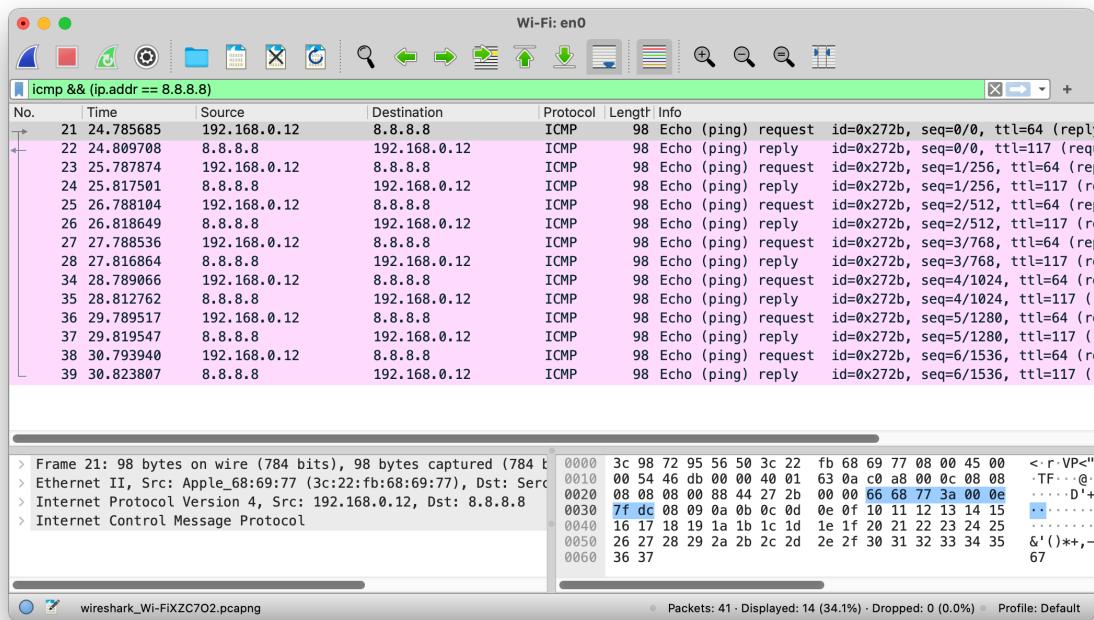


Podemos aplicarle un filtro doble (`icmp && id.addr == 8.8.8.8`) y ordenarlo si queremos seleccionando las columnas.

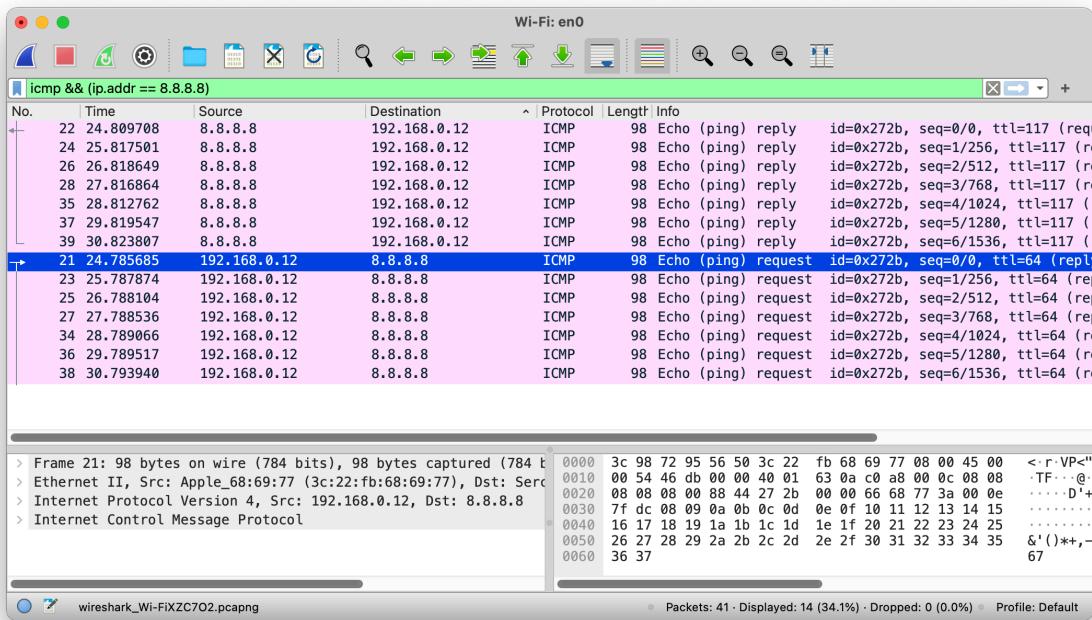


He querido comparar los resultados con la aplicación de wireshark en MacOS. En el mac he usado la interfaz en0.

```
Last login: Tue Jun 11 17:31:19 on ttys000
→ ~ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=117 time=24.191 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=29.774 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=30.715 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=28.463 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=23.825 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=30.199 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=30.006 ms
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 7 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 23.825/28.168/30.715/2.708 ms
→ ~
```



Lo he ordenado igual para visualizarlo como el anterior.



e) Responder a las siguientes preguntas: ¿Cuántos paquetes te aparecen en la pantalla? ¿Se corresponde con el número de mensajes enviados en el ping? Aporta en la resolución del ejercicio una captura de pantalla con los paquetes filtrados en Wireshark.

¿Cuántos paquetes te aparecen en la pantalla?

En Wireshark aparecen **20 paquetes ICMP**.

¿Se corresponde con el número de mensajes enviados en el ping?

Sí. 10 mensajes de ping en la terminal y Wireshark muestra **20 paquetes ICMP**, lo cual corresponde a los 10 mensajes de solicitud (Echo Request) y los 10 mensajes de respuesta (Echo Reply).

Por tanto, el análisis del tráfico ICMP capturado en Wireshark muestra que el número de paquetes ICMP coincide exactamente con el número de mensajes de ping enviados y recibidos, confirmando que la herramienta Wireshark está capturando y mostrando correctamente el tráfico de red generado por los comandos de ping.

Tarea 4:

En este caso, al igual que en el ejercicio anterior, se va a capturar tráfico de nuestra red. Para ello, si aún se tiene el tráfico capturado en el ejercicio anterior, se recomienda empezar una nueva captura. Tras esto:

- a) Abrir el navegador y visitar una página web. Con esta acción se generará tráfico HTTP y/o HTTPS.
- b) Usar una terminal para ejecutar el siguiente comando: nslookup google.com, con el cual se realizará una consulta DNS.
- c) Ejecutar el comando: ping google.com, lo que generará tráfico ICMP.
- d) Aplica los filtros correspondientes para filtrar paquetes en los protocolos HTTP/HTTPS, DNS e ICMP. Aportar capturas de pantalla de estos paquetes. ¿De qué colores se representa por defecto el tráfico de cada protocolo? ¿Habrá otra manera de filtrar el tráfico HTTP/HTTPS?

NOTA: Wireshark es una herramienta muy interesante, con la que se pueden aprender cómo funcionan los protocolos, etc. Por tanto, es interesante realizar más tareas aparte de las propuestas, con las que se profundice más en el análisis de los mensajes intercambiados, cabeceras, etc.

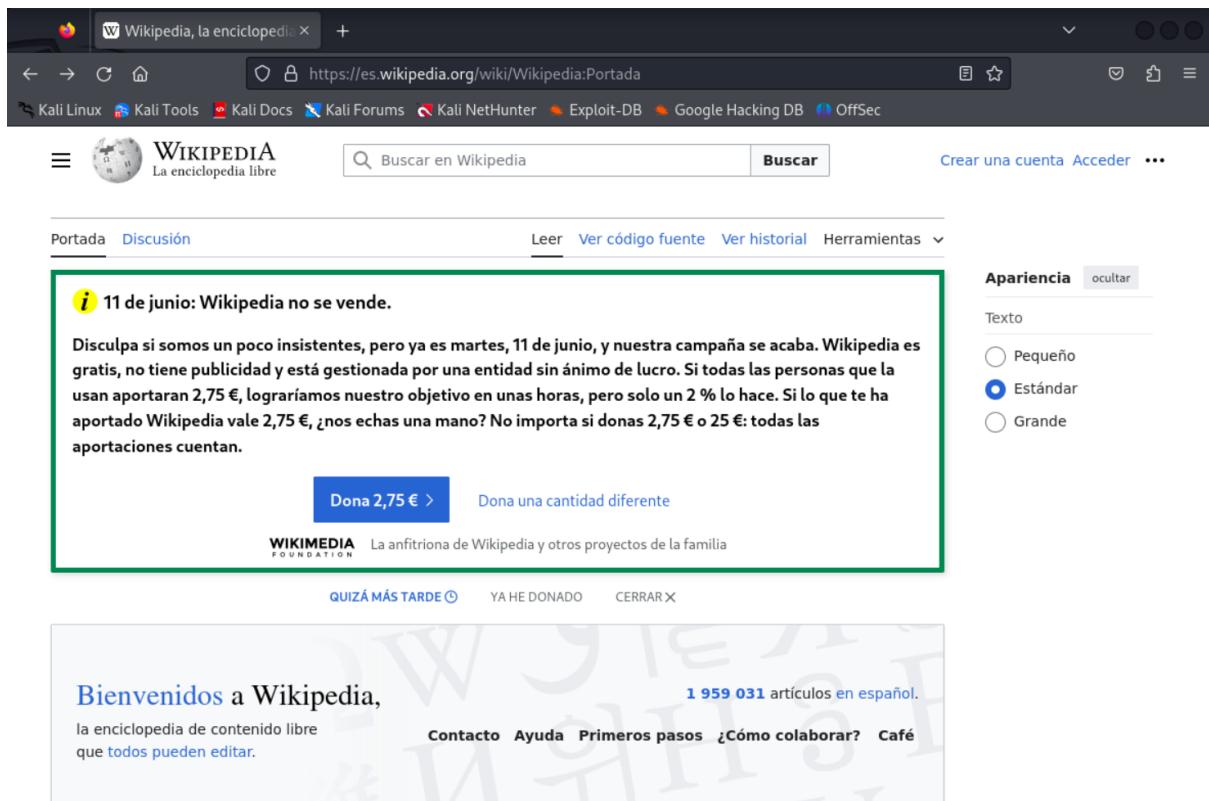
Para realizar este ejercicio en Wireshark, seguiremos los pasos sugeridos para a continuación capturar el tráfico de red y aplicar los filtros correspondientes para HTTP/HTTPS, DNS e ICMP.

Iniciar Captura:

- Abrimos de nuevo Wireshark desde la terminal con el comando “sudo wireshark” y seleccionamos la interfaz de red eth0.
- Hacemos clic en el botón “Start” (ícono de aleta de un tiburón) para comenzar una nueva captura de tráfico.

Visitamos una Página Web:

- Abrimos el navegador web, en este caso firefox y visitamos varias páginas webs. Esto generará tráfico HTTP y/o HTTPS.
- Mostramos una de las webs visitadas a modo ejemplo.



Realizar una Consulta DNS:

- Abrimos una terminal y ejecutamos el siguiente comando para realizar una consulta DNS: “nslookup google.com”

Enviar un Ping a google.com:

- En la misma terminal, ejecutamos el siguiente comando para enviar un ping a google.com: “ping google.com”.
- Dejamos que el ping envíe varios paquetes y luego lo detenemos presionando Ctrl + C.

```

kali㉿kali:[~]
$ nslookup google.com
Server:      192.168.0.1
Address:     192.168.0.1#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.185.14
3 (request in 2088)
4 (reply in 2089)
kali㉿kali:[~]
$ ping google.com
PING google.com (142.250.185.14) 56(84) bytes of data.
64 bytes from mad41s11-in-f14.1e100.net (142.250.185.14): icmp_seq=1 ttl=63 time=29.9 ms
64 bytes from mad41s11-in-f14.1e100.net (142.250.185.14): icmp_seq=2 ttl=63 time=29.9 ms
64 bytes from mad41s11-in-f14.1e100.net (142.250.185.14): icmp_seq=3 ttl=63 time=32.6 ms
64 bytes from mad41s11-in-f14.1e100.net (142.250.185.14): icmp_seq=4 ttl=63 time=32.3 ms
64 bytes from mad41s11-in-f14.1e100.net (142.250.185.14): icmp_seq=5 ttl=63 time=33.5 ms
^C
--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4010ms
rtt min/avg/max/mdev = 29.861/31.615/33.516/1.489 ms
Profile:Default [GUI WARNING] -- QStandardPaths: XDG_RUNTIME_DIR set to /tmp/root'
kali㉿kali:[~]
[23] 19:25:40.692686 [Capture Stop] -- Capture Start ...

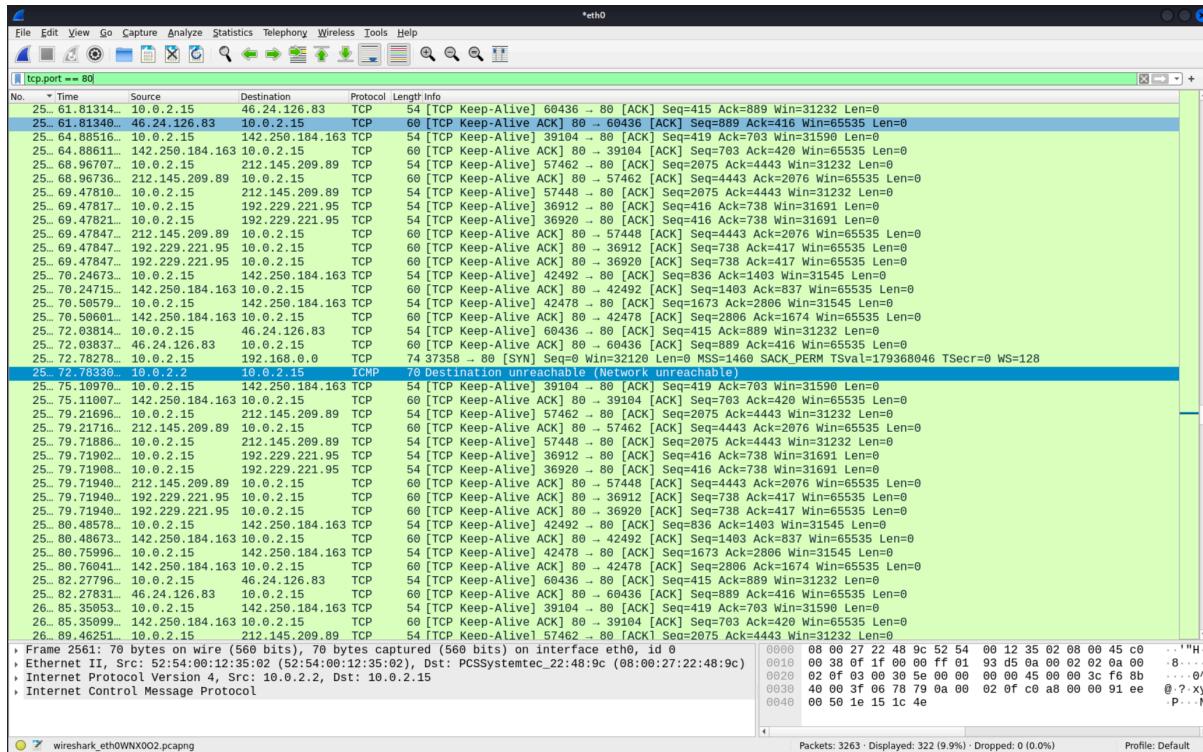
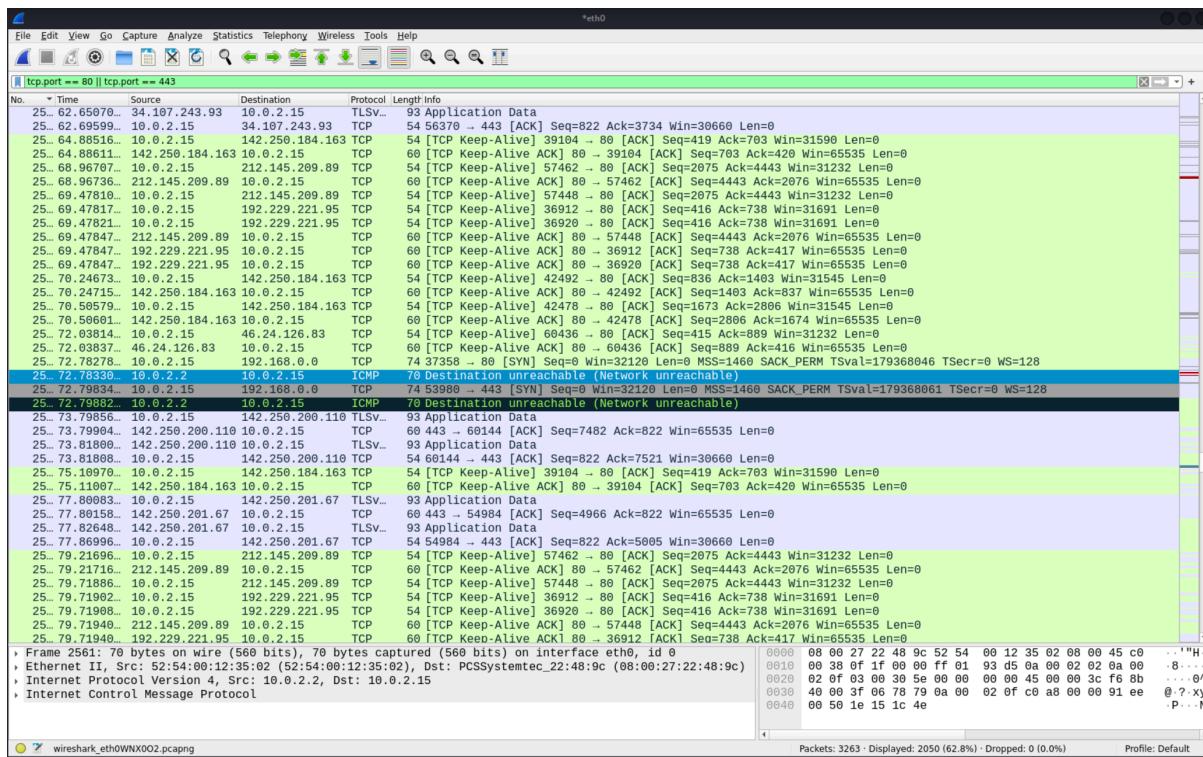
```

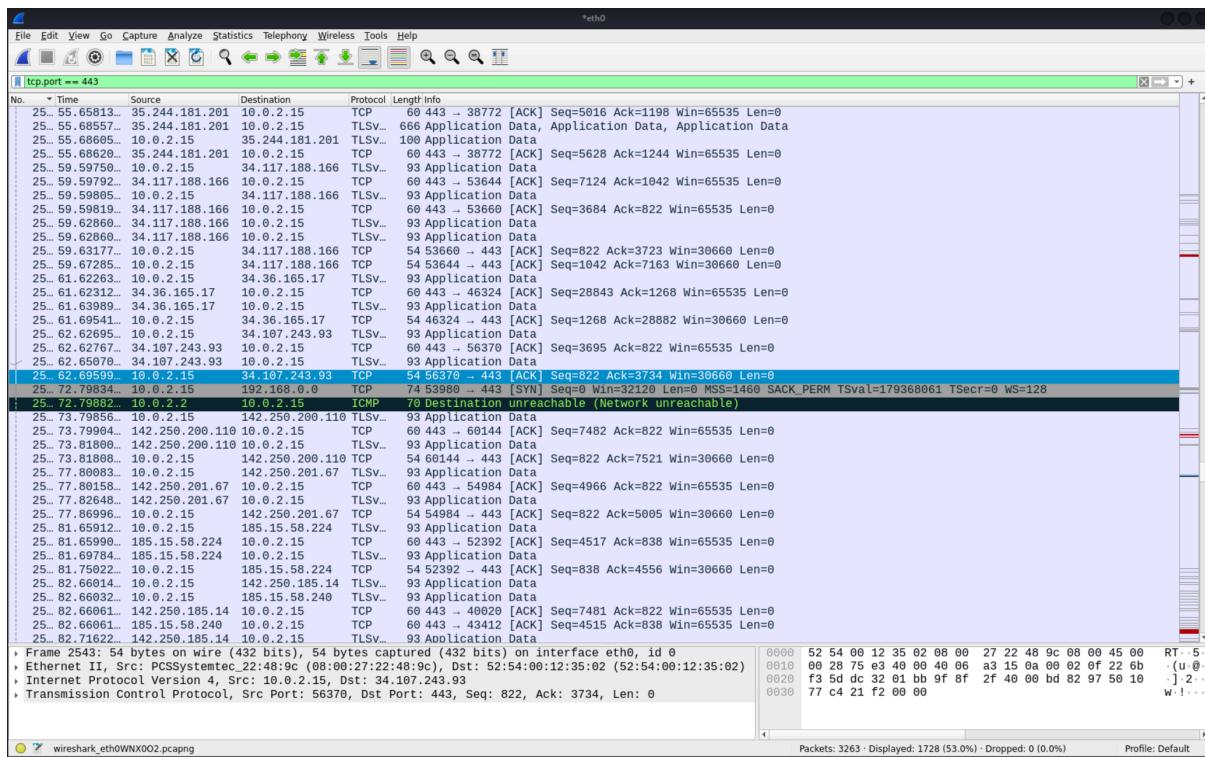
Detener la Captura:

- Volvemos a Wireshark y hacemos clic en el botón "Stop" (ícono con un cuadrado rojo) para detener la captura de tráfico.

Filtrar Tráfico HTTP/HTTPS:

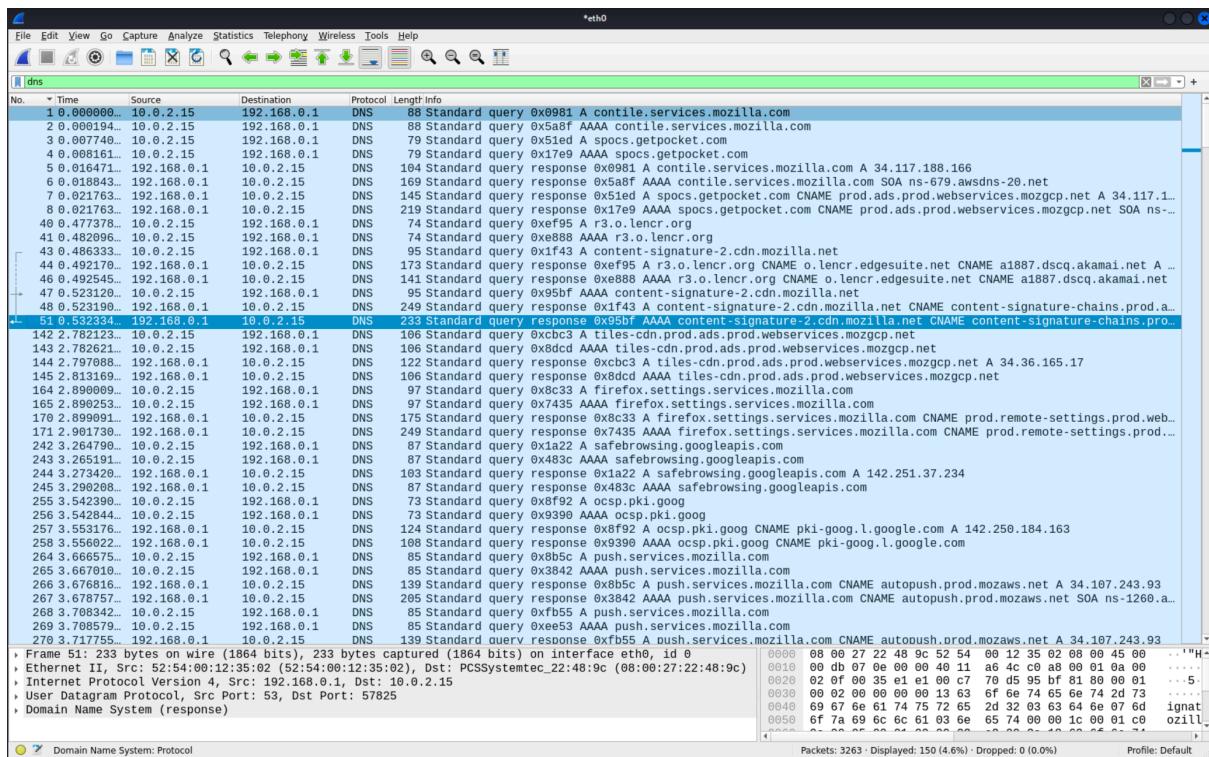
- En la barra de filtros de Wireshark, escribimos el siguiente filtro para mostrar solo el tráfico HTTP y HTTPS: “tcp.port == 80 || tcp.port == 443”
- Presionamos Enter para aplicar el filtro.
- **Añadimos captura de Pantalla:** del tráfico HTTP/HTTPS filtrado.





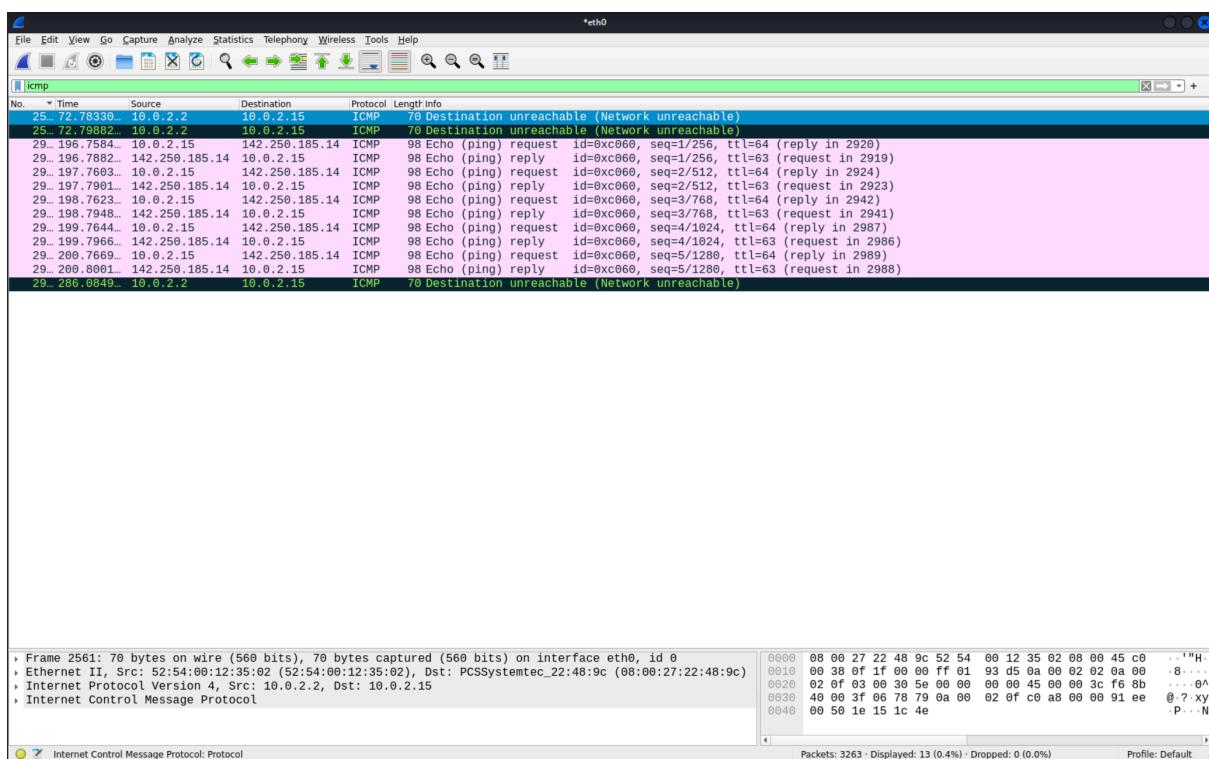
Filtrar Tráfico DNS:

- En la barra de filtros de Wireshark, escribimos el siguiente filtro para mostrar solo el tráfico DNS: “dns”
- Presionamos Enter para aplicar el filtro.
- **Añadimos captura de Pantalla:** del tráfico DNS filtrado.



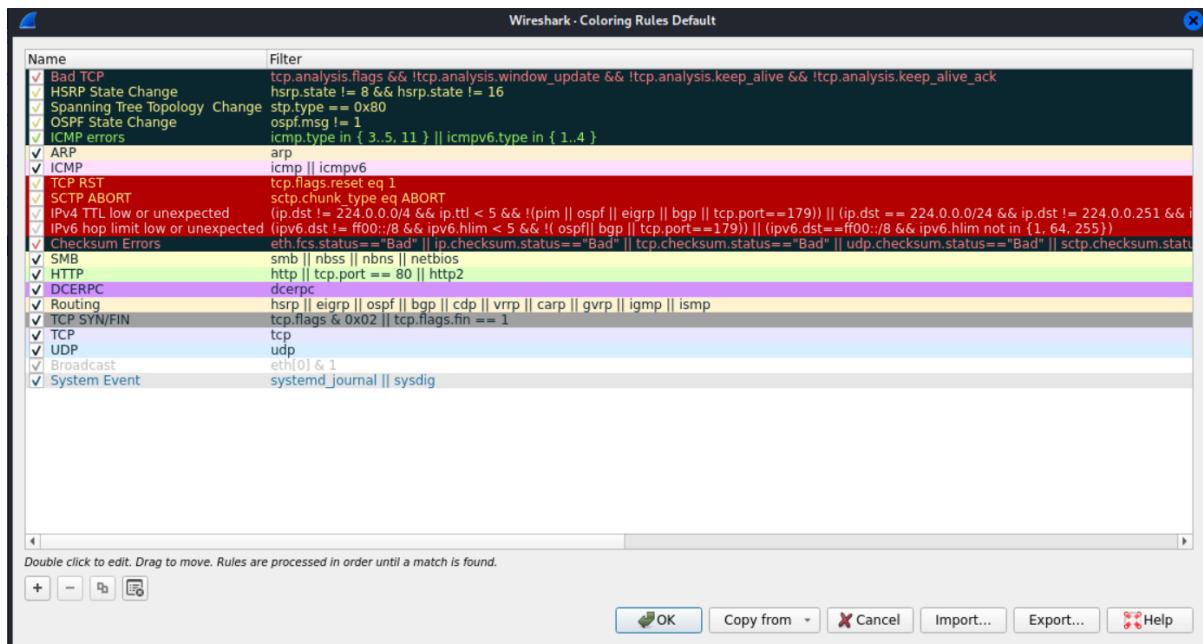
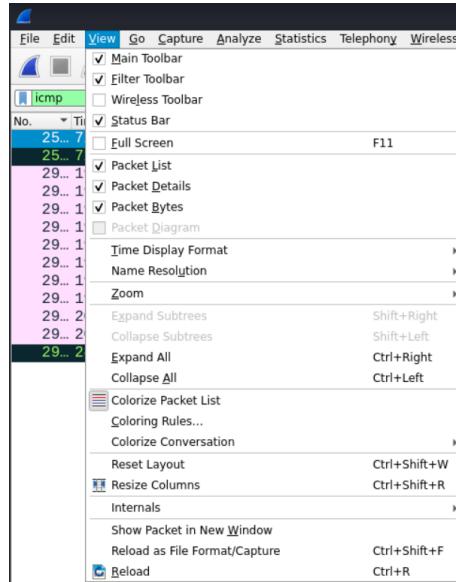
Filtrar Tráfico ICMP:

- En la barra de filtros de Wireshark, escribimos el siguiente filtro para mostrar solo el tráfico ICMP: "icmp"
- Presionamos Enter para aplicar el filtro.
- Añadimos captura de Pantalla:** del tráfico ICMP filtrado.



Análisis de Resultados

- **Colores de los Protocolos:** para poder ver las reglas de los colores, seleccionamos en “view” y a continuación “coloring rules”.



Estas reglas de coloración en Wireshark ayudan a identificar rápidamente diferentes tipos de tráfico y eventos importantes en una captura de red. Los colores hacen más fácil destacar y analizar problemas específicos como errores de checksum, cambios de estado en protocolos de enrutamiento, paquetes ICMP, tráfico ARP, y más.

¿De qué colores se representa por defecto el tráfico de cada protocolo?

- **HTTP/HTTPS:** Verde claro (HTTP) y morado muy claro (HTTPS/TLS).
- **DNS:** Azul claro.
- **ICMP:** Rosa claro.

¿Habrá otra manera de filtrar el tráfico HTTP/HTTPS?

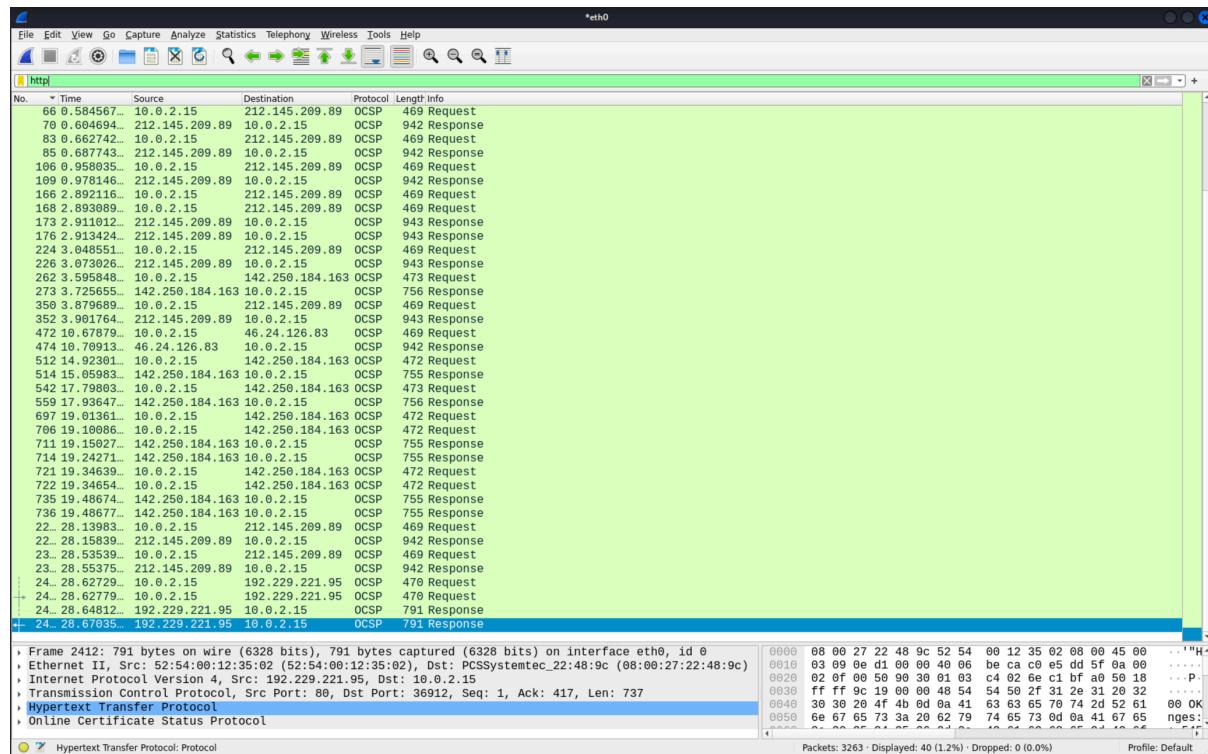
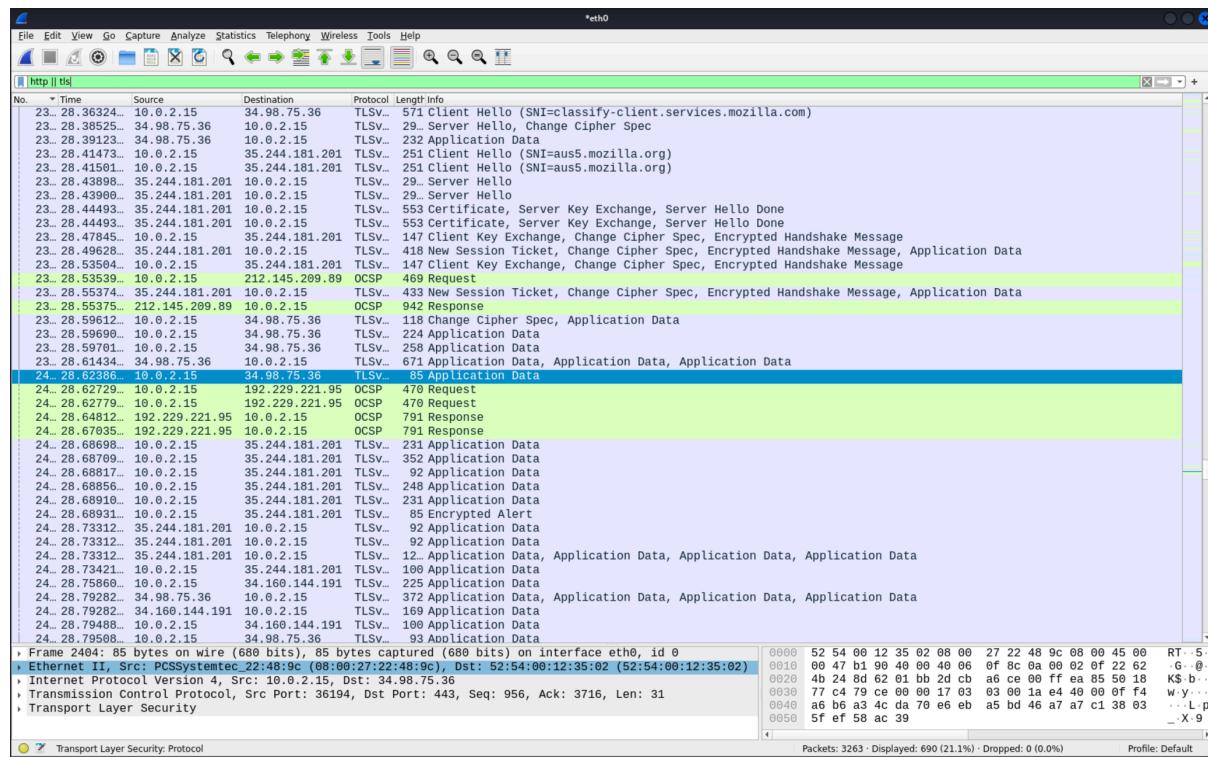
- Sí, se puede usar http también como filtro pero no se puede usar https, en su lugar se podría usar TLS. No son exactamente lo mismo, es por ello que en este caso es mejor usar el puerto ya que mayormente se usa el puerto 443 para https.

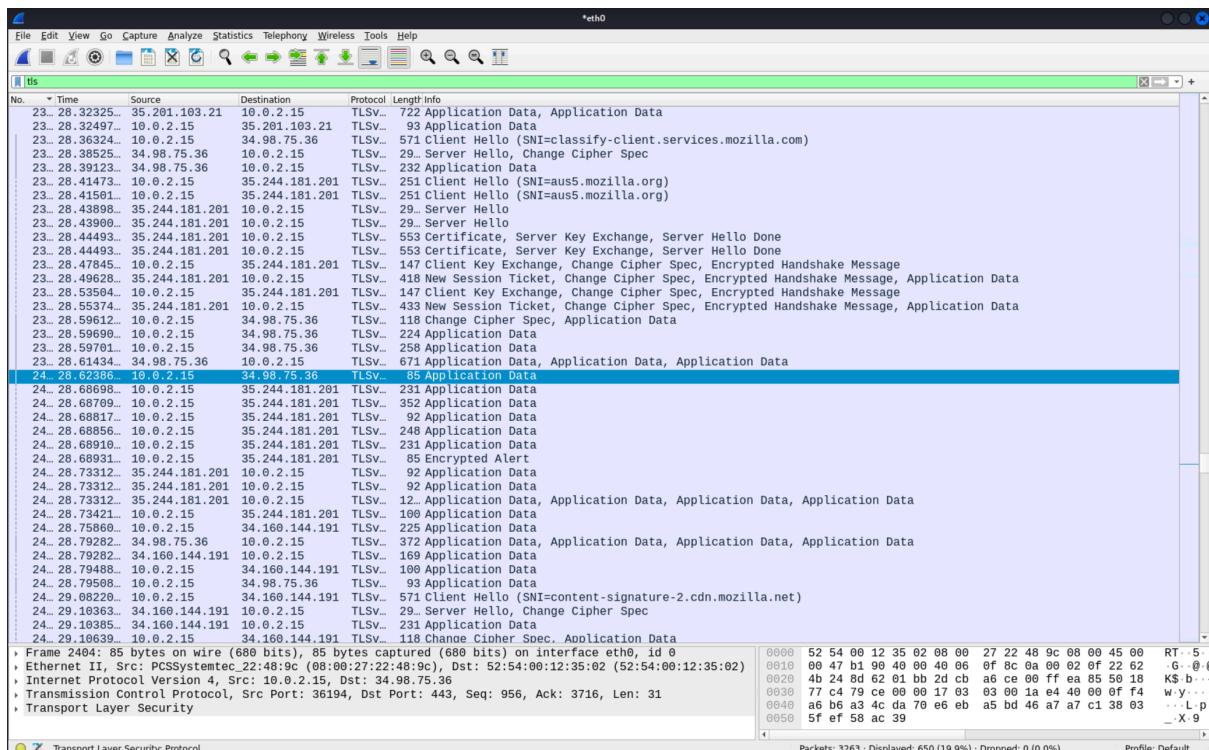
TLS (Transport Layer Security) es un protocolo criptográfico diseñado para proporcionar seguridad en las comunicaciones a través de una red. TLS cifra los datos para asegurar que la comunicación entre dos sistemas (por ejemplo, entre un cliente y un servidor web) sea privada e íntegra.

- **Uso:** TLS se utiliza no solo para HTTPS, sino también para asegurar otros protocolos de red como SMTP (para correo electrónico), FTP (para transferencias de archivos) y otros.
- **Versiónes:** TLS ha evolucionado a lo largo del tiempo, con versiones más seguras como TLS 1.2 y TLS 1.3.

Cifrado HTTPS: Cuando se establece una conexión HTTPS, se utiliza el protocolo TLS para cifrar los datos que se envían y reciben. Por lo tanto, HTTPS no podría existir sin TLS.

Proceso: Al acceder a un sitio web a través de HTTPS, el navegador y el servidor realizan un “handshake” TLS para establecer una conexión segura. Durante este proceso, se autentican entre sí y acuerdan los parámetros de cifrado que se utilizarán.





Tarea 5:

En este ejercicio se va a utilizar “Netcat”, una herramienta de red versátil que permite a los usuarios leer y escribir datos a través de conexiones de red utilizando los protocolos de transporte TCP y UDP. Para la realización de esta tarea, es necesario abrir dos terminales y el Wireshark.

Netcat, también conocido como **nc**, es una herramienta de red versátil utilizada para leer y escribir datos a través de conexiones de red utilizando los protocolos **TCP** o **UDP**. Sus principales características y usos incluyen:

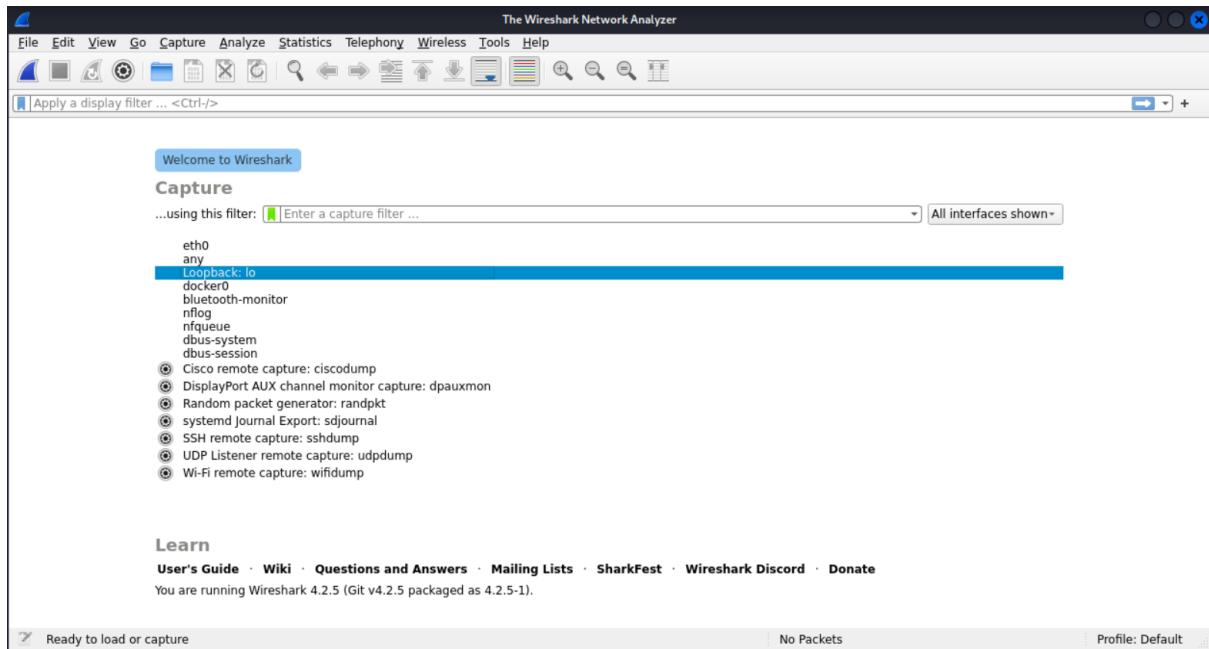
1. **Transferencia de archivos:** Netcat puede enviar y recibir archivos a través de la red.
2. **Exploración de puertos:** Puede escanear puertos para ver cuáles están abiertos en un host remoto.
3. **Depuración y prueba de red:** Permite probar y depurar redes, servicios y aplicaciones.
4. **Servidor y cliente:** Puede actuar como un servidor simple para escuchar conexiones entrantes o como un cliente para iniciar conexiones.

Recordamos los siguientes conceptos.

- **UDP (User Datagram Protocol):** Es un protocolo de comunicación sin conexión que permite enviar datagramas sin necesidad de establecer una conexión previa. Es rápido pero no garantiza la entrega, el orden ni la integridad de los datos.

- **TCP (Transmission Control Protocol):** Es un protocolo de comunicación orientado a conexión que garantiza la entrega fiable y ordenada de datos entre aplicaciones. Incluye mecanismos de control de flujo, corrección de errores y reenvío de paquetes perdidos.

a) Abrir Wireshark y comenzar a capturar el tráfico en la interfaz de Loopback (lo).



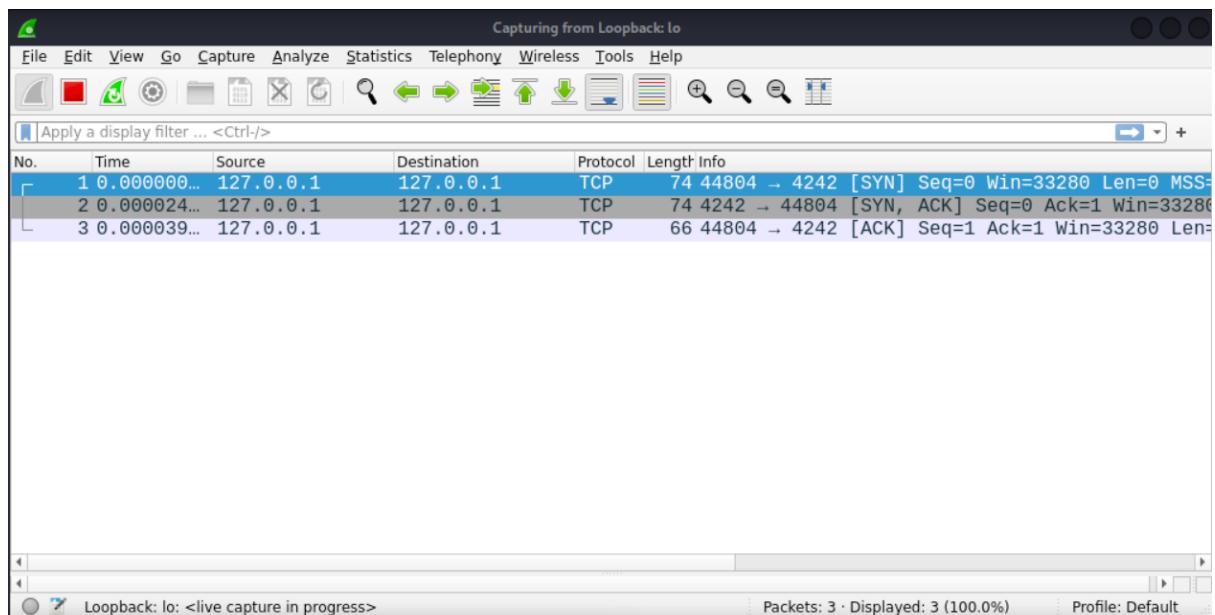
b) En la primera terminal hay que poner a escuchar a Netcat (con el comando “nc”) en un determinado puerto. Para ello, se recomienda ver la ayuda de este comando (“nc -h”).

```

(kali㉿kali)-[~]
$ nc -h
[1.10-48.1]
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound: nc -l -p port [-options] [hostname] [port]
options:
  -c shell commands      as '-e'; use /bin/sh to exec [dangerous!!]
  -e filename            program to exec after connect [dangerous !!]
  -b                   allow broadcasts
  -g gateway             source-routing hop point[s], up to 8
  -G num                source-routing pointer: 4, 8, 12, ...
  -h                   this cruft
  -i secs               delay interval for lines sent, ports scanned
  -k                   set keepalive option on socket
  -l                   listen mode, for inbound connects
  -n                   numeric-only IP addresses, no DNS
  -o file               hex dump of traffic
  -p port               local port number
  -r                   randomize local and remote ports
  -q secs               quit after EOF on stdin and delay of secs
  -s addr               local source address
  -T tos                set Type Of Service
  -t                   answer TELNET negotiation
  -u                   UDP mode
  -v                   verbose [use twice to be more verbose]
  -w secs               timeout for connects and final net reads
  -C                   Send CRLF as line-ending
  -z                   zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-\data').

```

c) En la segunda terminal hay que conectar Netcat a un servidor en un puerto específico. Como el servidor que hemos creado en la primera terminal es en el mismo equipo, habrá que indicarle que la dirección IP es 127.0.0.1 (o localhost), y el puerto al que hay que conectarse, es en el que se ha puesto en escucha al servidor en la primera terminal.



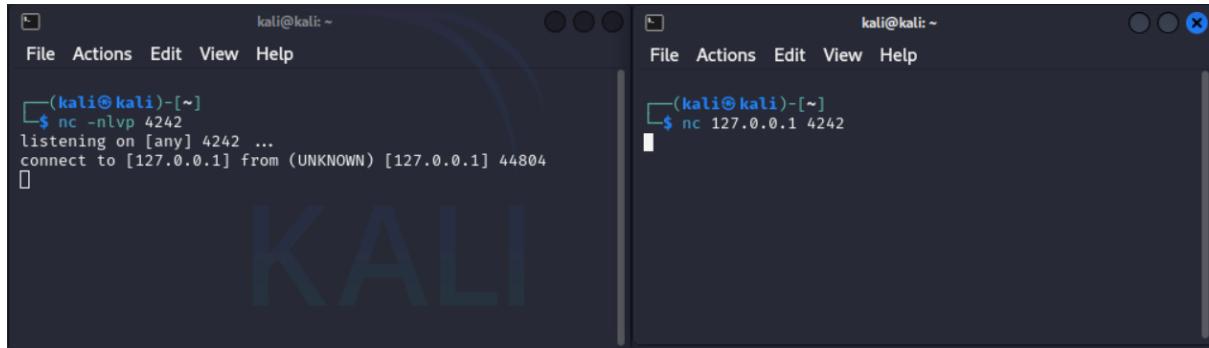
Usamos el comando **nc** con las siguientes flags, **-nlvp** que significan lo siguiente.

-n: No realizar resolución DNS.

-l: Escuchar conexiones entrantes.

-v: Modo detallado (verbose).

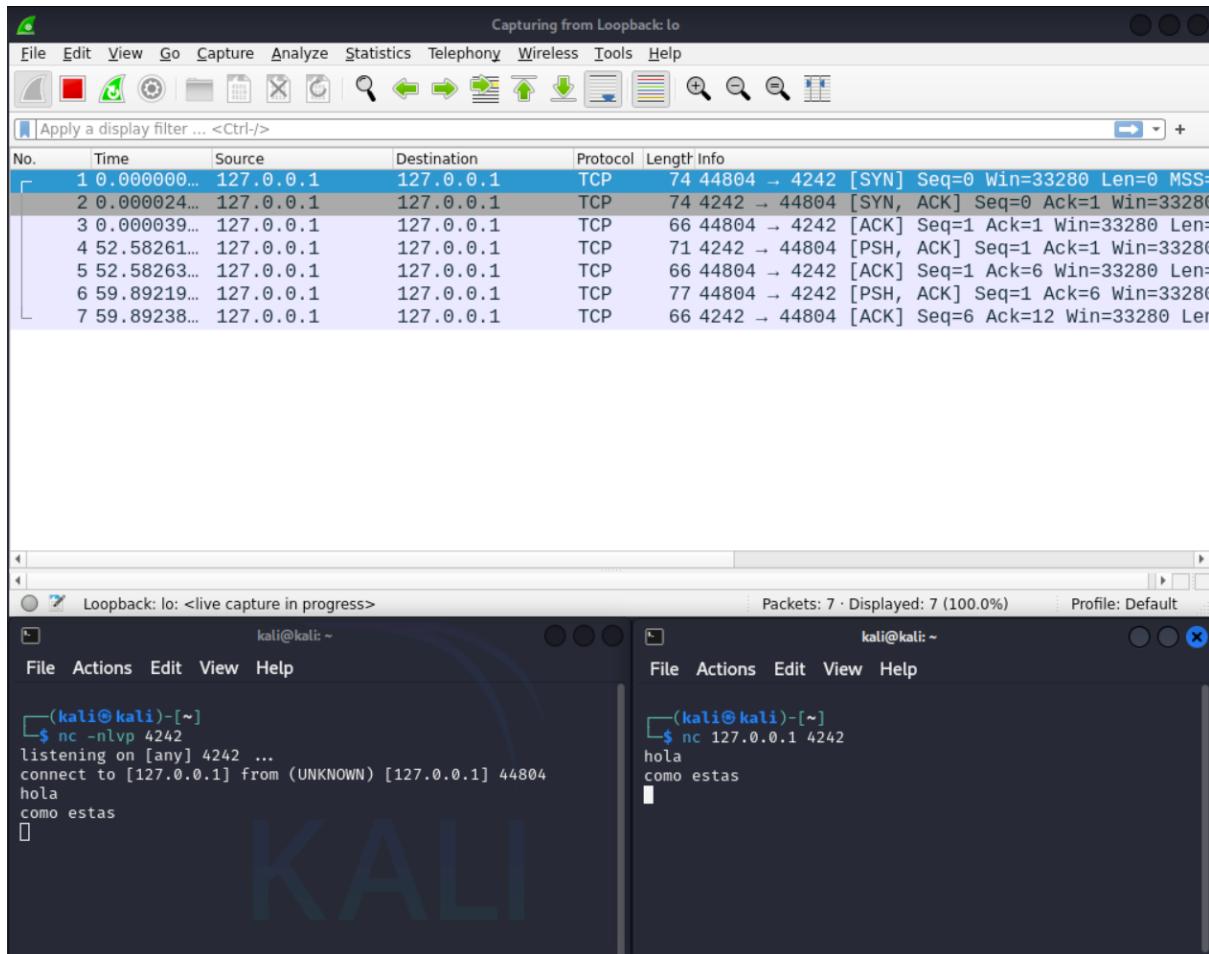
-p: Especificar el puerto.



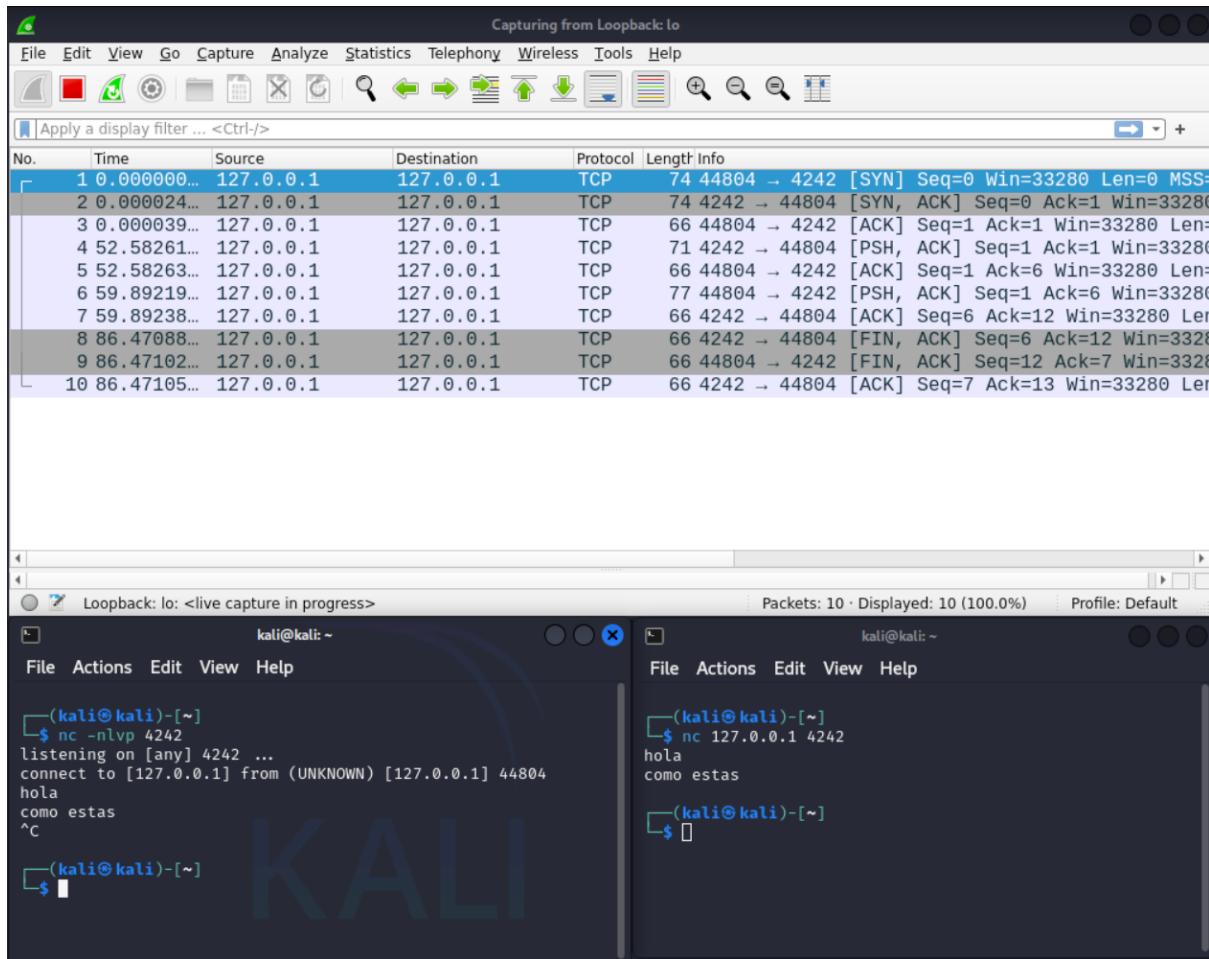
The image shows two terminal windows side-by-side. The left window shows a netcat listener running on port 4242, with the command \$ nc -nlvp 4242 and the message "listening on [any] 4242 ... connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 44804". The right window shows a netcat connection to the listener, with the command \$ nc 127.0.0.1 4242 and the message "hola como estas".

Observamos que la conexión ya se ha establecido y recogido en wireshark, donde podemos observar los paquetes SYN, SYN-ACK y ACK.

d) Una vez establecida la conexión entre ambos (se quedan en espera las terminales), escribir mensajes entre ambas y comprobar que aparecen en el otro extremo de la comunicación también. Tras esto, cerrar la conexión haciendo Ctrl + C.



The image shows a Wireshark capture window at the top and two terminal windows below it. The Wireshark window shows a live capture from the loopback interface (Loopback: lo) with 7 displayed packets. The captured traffic consists of TCP segments between 127.0.0.1 and 127.0.0.1, illustrating the handshake and data exchange. The terminal windows show the netcat listener on the left and the client on the right. The client sends "hola" and "como estas" to the listener, which responds with "hola" and "como estas".



Podemos observar claramente que los mensajes que se emiten en una terminal son recibidos en la otra y viceversa.

e) Por último, analizar los mensajes que se han capturado en el Wireshark. ¿Qué protocolo usa esta comunicación? ¿Se puede ver el contenido de los mensajes intercambiados? ¿Cómo y dónde?

¿Qué protocolo usa esta comunicación?

La comunicación utiliza el **protocolo TCP (Transmission Control Protocol)**. Esto se puede determinar observando la columna “Protocol” en Wireshark, donde todos los paquetes están etiquetados como “TCP”.

¿Se puede ver el contenido de los mensajes intercambiados? ¿Cómo y dónde?

Sí, se puede ver el contenido de los mensajes intercambiados en la captura de Wireshark. Para ver el contenido de los mensajes, seguimos estos pasos:

1. Seleccionamos el Paquete:

- En la lista de paquetes capturados en Wireshark, seleccionamos un paquete de datos. Los paquetes con los mensajes intercambiados suelen ser aquellos que tienen la etiqueta “PSH, ACK” en la columna “Info”.

2. Examinamos los Detalles del Paquete:

- Una vez que seleccionamos un paquete, en el panel inferior de Wireshark, se puede ver los detalles del mismo.
- Dentro de los detalles del paquete, expandimos las secciones “Transmission Control Protocol” y “Data”. La sección “Data” contiene el contenido del mensaje intercambiado.
- Ambas secciones contiene la misma información pero expresada de un modo mas legible o menos para el humano.

Tras los mensajes del Three HandShake, aparece el primer mensaje de transmisión de datos, en este caso, se puede observar en la parte de abajo el contenido del mensaje “hola”. En la siguiente captura se puede ver arriba el tráfico de wireshark, en azul la línea del mensaje y al clicar sobre ésta se muestra abajo la ventana de ese mensaje con la palabra hola a la derecha.

The screenshot shows two windows of the Wireshark application. The main window at the top displays a list of network packets captured from the 'Loopback' interface. The packet list includes columns for No., Time, Source, Destination, Protocol, Length, and Info. Several TCP handshake packets are visible, followed by a data packet at index 4. The detailed view window at the bottom, titled 'Wireshark - Packet 4 - Loopback: lo', shows the selected packet's structure. It includes a summary pane with protocol details (Frame 4: 71 bytes on wire, 71 bytes captured, Ethernet II, IP Version 4, TCP Port 4242 to 44804), a bytes pane showing hex and ASCII representations of the data, and a notes pane. The ASCII dump shows the word 'hola' in the fourth byte position. A status bar at the bottom provides packet information and a checkbox for 'Show packet bytes'.

El siguiente es el mensaje de “acknowledgement”.

Capturing from Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000...	127.0.0.1	127.0.0.1	TCP	74	44804 → 4242 [SYN] Seq=0 Win=33280 Len=0 MSS=6
2	0.000024...	127.0.0.1	127.0.0.1	TCP	74	4242 → 44804 [SYN, ACK] Seq=0 Ack=1 Win=33280
3	0.000039...	127.0.0.1	127.0.0.1	TCP	66	44804 → 4242 [ACK] Seq=1 Ack=1 Win=33280 Len=0
4	52.58261...	127.0.0.1	127.0.0.1	TCP	71	4242 → 44804 [PSH, ACK] Seq=1 Ack=1 Win=33280
5	52.58263...	127.0.0.1	127.0.0.1	TCP	66	44804 → 4242 [ACK] Seq=1 Ack=6 Win=33280 Len=0
6	59.89219...	127.0.0.1	127.0.0.1	TCP	77	44804 → 4242 [PSH, ACK] Seq=1 Ack=6 Win=33280
7	59.89238...	127.0.0.1	127.0.0.1	TCP	66	4242 → 44804 [ACK] Seq=6 Ack=12 Win=33280 Len=0
8	86.47088...	127.0.0.1	127.0.0.1	TCP	66	4242 → 44804 [FIN, ACK] Seq=6 Ack=12 Win=33280
9	86.47102...	127.0.0.1	127.0.0.1	TCP	66	44804 → 4242 [FIN, ACK] Seq=12 Ack=7 Win=33280
10	86.47105...	127.0.0.1	127.0.0.1	TCP	66	4242 → 44804 [ACK] Seq=7 Ack=13 Win=33280 Len=0

Wireshark - Packet 5 - Loopback: lo

```

Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 44804, Dst Port: 4242, Seq: 1, Ack: 6, Len: 0

No.: 5 · Time: 52.582633884 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · L... 66 · Info: 44804 → 4242 [ACK] Seq=1 Ack=6 Win=33280 Len=0 TSval=1595986932 TSecr=1595986932
 Show packet bytes

```

Close Help

Le sigue el otro mensaje donde se puede apreciar el segundo mensaje “como estas”.

Capturing from Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000...	127.0.0.1	127.0.0.1	TCP	74	44804 → 4242 [SYN] Seq=0 Win=33280 Len=0 MSS=6
2	0.000024...	127.0.0.1	127.0.0.1	TCP	74	4242 → 44804 [SYN, ACK] Seq=0 Ack=1 Win=33280
3	0.000039...	127.0.0.1	127.0.0.1	TCP	66	44804 → 4242 [ACK] Seq=1 Ack=1 Win=33280 Len=0
4	52.58261...	127.0.0.1	127.0.0.1	TCP	71	4242 → 44804 [PSH, ACK] Seq=1 Ack=1 Win=33280
5	52.58263...	127.0.0.1	127.0.0.1	TCP	66	44804 → 4242 [ACK] Seq=1 Ack=6 Win=33280 Len=0
6	59.89219...	127.0.0.1	127.0.0.1	TCP	77	44804 → 4242 [PSH, ACK] Seq=1 Ack=6 Win=33280
7	59.89238...	127.0.0.1	127.0.0.1	TCP	66	4242 → 44804 [ACK] Seq=6 Ack=12 Win=33280 Len=0
8	86.47088...	127.0.0.1	127.0.0.1	TCP	66	4242 → 44804 [FIN, ACK] Seq=6 Ack=12 Win=33280
9	86.47102...	127.0.0.1	127.0.0.1	TCP	66	44804 → 4242 [FIN, ACK] Seq=12 Ack=7 Win=33280
10	86.47105...	127.0.0.1	127.0.0.1	TCP	66	4242 → 44804 [ACK] Seq=7 Ack=13 Win=33280 Len=0

Wireshark - Packet 6 - Loopback: lo

```

> Frame 6: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 44804, Dst Port: 4242, Seq: 1, Ack: 6, Len: 11
> Data (11 bytes)

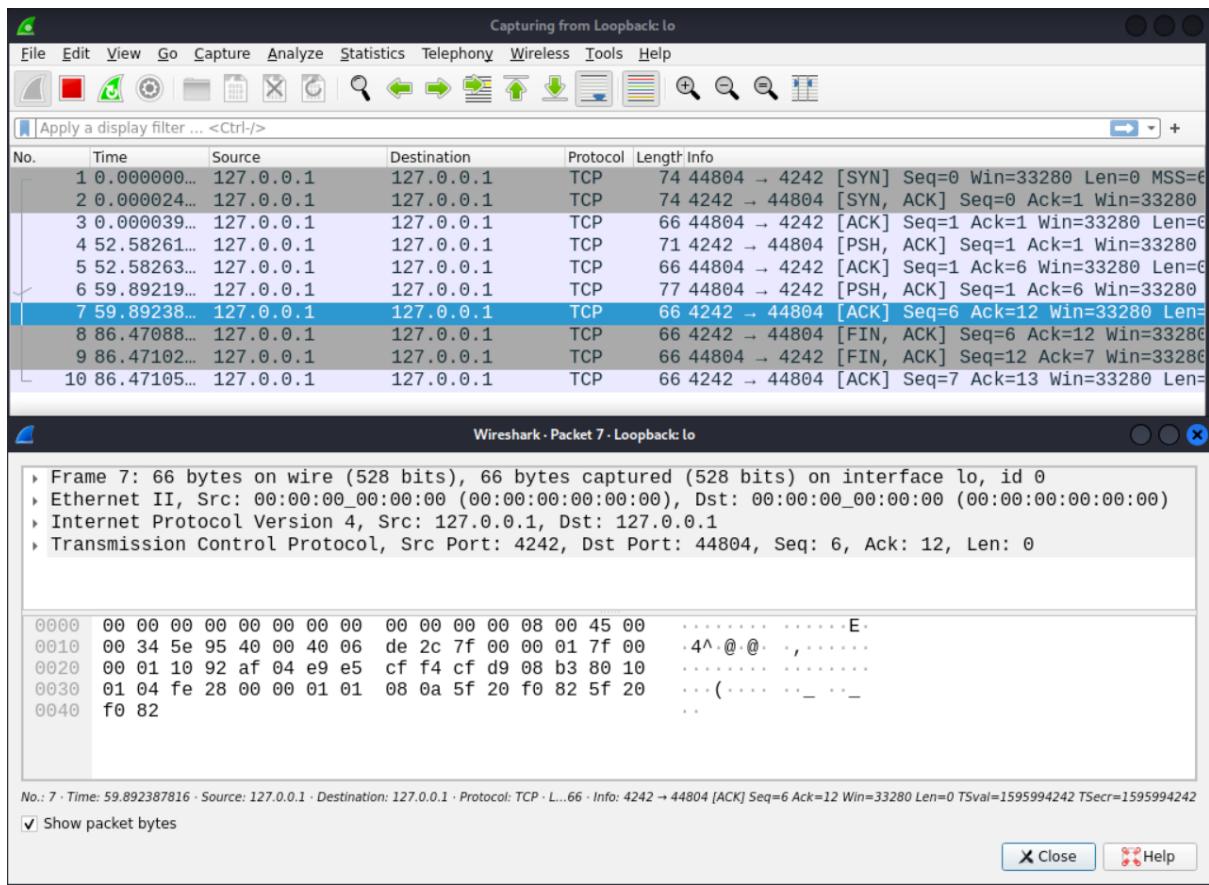
0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00  ....E.
0010  00 3f 25 45 40 00 40 06 17 72 7f 00 00 01 7f 00  .?%E@.r....
0020  00 01 af 04 10 92 cf d9 08 a8 e9 e5 cf f4 80 18  .....
0030  01 04 fe 33 00 00 01 01 08 0a 5f 20 f0 82 5f 20  ..3.....
0040  d3 f4 63 6f 6d 6f 20 65 73 74 61 73 0a  ..como e stas.

No.: 6 · Time: 59.892197172 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · ...info: 44804 → 4242 [PSH, ACK] Seq=1 Ack=6 Win=33280 Len=11 TSval=1595994242 TSecr=1595986932
 Show packet bytes

```

X Close Help

Último mensaje de acknowledgement.



f) Mostrar capturas de pantalla que validen la realización del ejercicio.

Las capturas de pantalla se han incluido en cada apartado del ejercicio para facilitar su seguimiento.