

# TPL 2 - Aplicaciones 1: Cliente/Servidor – Telnet

**Fecha de Entrega:** 31/03/2025

**Alumno:** Joaquin Dominguez -182901

**URL de Entrega:** <https://tinyurl.com/TyR-2025-TP2>

**Objetivo:** Familiarizar el trabajo con aplicaciones cliente/servidor y, como primer ejemplo de aplicación de la pila TCP/IP, conocer el propósito y funcionamiento del protocolo telnet.

Para el estudio previo de los temas se sugiere la siguiente lectura:

Modelo cliente/servidor: Capítulo 17 (hasta pagina 546) [FOR09]

Multiplexación en capas de transporte de la pila TCP/IP: “TCP/IP Processes, Multiplexing and Client/Server Application Roles”, y temas siguientes hasta “Common TCP/IP Applications and Assigned Well-Known and Registered Port Numbers”

Establecimiento de conexiones TCP, Negociación de 3 Vías (three way handshake): <https://www.youtube.com/watch?v=ixBAG8apsFM> (minuto 2:0 en adelante)

Protocolo Telnet: Capítulo 20.1 [FOR09]

## **Primer parte: Creación de un modelo simple Cliente/Servidor**

Utilice para esta parte de la práctica el laboratorio de práctica kathara-lab\_conf\_inicial y configure las interfaces de pc1 y pc2 tal como lo hizo en el primer trabajo práctico de laboratorio. Verifique conectividad entre ambos hosts.

Defina un número de puerto para el proceso servidor (superior a 1024).

En el dispositivo capturador inicie la captura utilizando el comando tcpdump o tshark sobre la interfaz eth0 y redirigir la salida a un archivo en el directorio /shared para su posterior análisis. (Ej. “ tshark -i eth0 -w - > /shared/captura\_nc.pcap ”)

En el host pc1 deberá ejecutar la utilidad nc actuando como servidor, indicando como parámetro el número de puerto elegido. Una vez iniciado, este servicio quedará en modo de escucha o listening. En el otro host (pc2) ejecute la utilidad nc como cliente indicando como parámetros la IP del servidor y número de puerto. En el otro host (pc2) ejecute la utilidad nc como cliente indicando como parámetros la IP del servidor y número de puerto.

Si generó correctamente los procesos servidor y cliente, debería poder ver una especie de “chat”. Intercambie varios mensajes con el otro dispositivo y finalice la conexión (en cualquiera de los host presione CTRL+C). Luego detenga la captura en el dispositivo capturador (CTRL+C).

Analice la captura almacenada en el archivo utilizando tshark y diversos parámetros de visualización (consulte la guía de comandos provista por la materia).

- a) “Extraiga” de la captura solamente los datos intercambiados a nivel aplicación y remítalos.
- b) Realice un diagrama representando el intercambio de tramas indicando las que corresponden al establecimiento de la conexión TCP, a las de transmisión de datos a nivel aplicación, y a las del cierre de la conexión TCP.
- c) ¿Todas las tramas en las que identifica el protocolo TCP transportan datos de aplicación?. ¿Si no es así puede explicar el porqué?

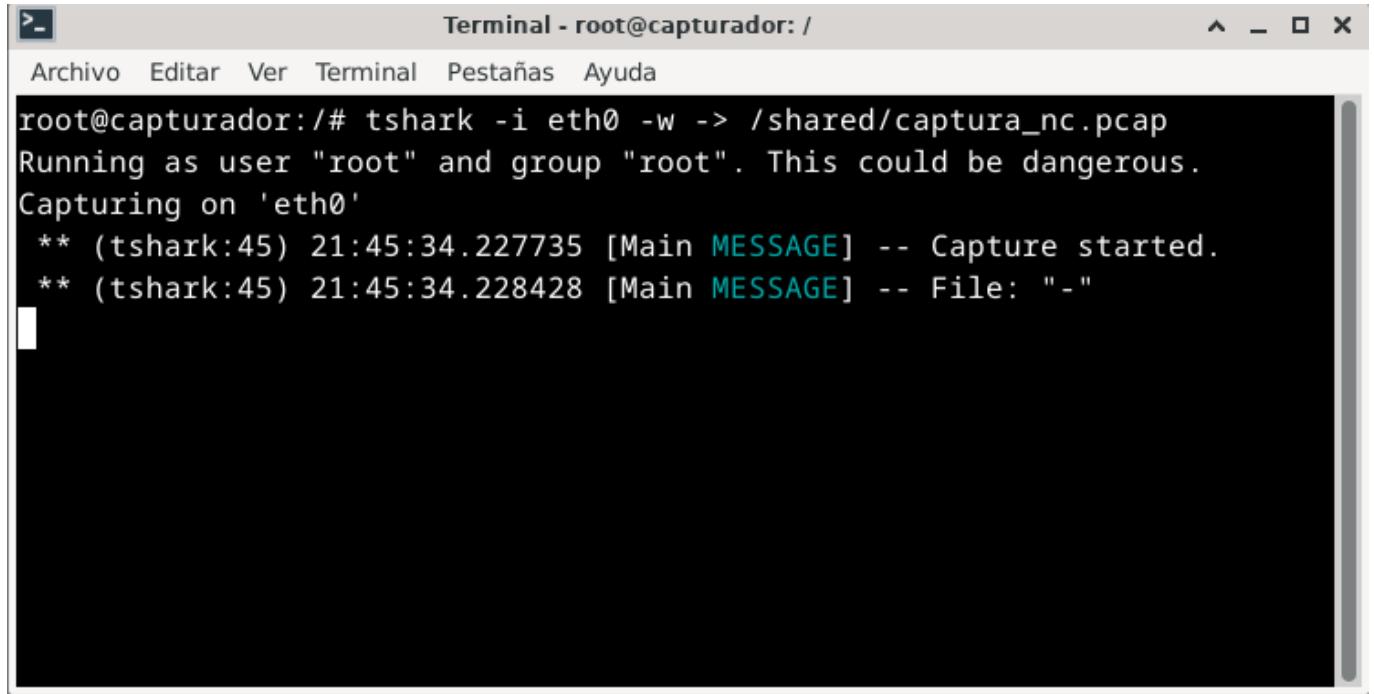
Verifico conectividad entre pc1 y pc2:

```
Terminal - root@tyr12: /  
Archivo Editar Ver Terminal Pestañas Ayuda  
++ ip route add default via 10.4.11.30  
--- End Startup Commands Log  
root@tyr12:/# ping -c 3 try11  
ping: try11: Temporary failure in name resolution  
root@tyr12:/# ping -c 3 tyr11  
PING tyr11 (10.4.11.11) 56(84) bytes of data.  
64 bytes from tyr11 (10.4.11.11): icmp_seq=1 ttl=64 time=0.156 ms  
64 bytes from tyr11 (10.4.11.11): icmp_seq=2 ttl=64 time=0.196 ms  
64 bytes from tyr11 (10.4.11.11): icmp_seq=3 ttl=64 time=0.192 ms  
  
--- tyr11 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2027ms  
rtt min/avg/max/mdev = 0.156/0.181/0.196/0.018 ms  
root@tyr12:/#
```

```
Terminal - root@tyr11: /  
Archivo Editar Ver Terminal Pestañas Ayuda  
++ ip addr add 10.4.11.11/24 dev eth0  
++ echo '10.4.11.12 tyr12'  
++ hostname tyr11  
++ ip route add default via 10.4.11.30  
--- End Startup Commands Log  
root@tyr11:/# ping -c 3 tyr12  
PING tyr12 (10.4.11.12) 56(84) bytes of data.  
64 bytes from tyr12 (10.4.11.12): icmp_seq=1 ttl=64 time=0.411 ms  
64 bytes from tyr12 (10.4.11.12): icmp_seq=2 ttl=64 time=0.165 ms  
64 bytes from tyr12 (10.4.11.12): icmp_seq=3 ttl=64 time=0.169 ms  
  
--- tyr12 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2030ms  
rtt min/avg/max/mdev = 0.165/0.248/0.411/0.115 ms  
root@tyr11:/#
```

Iniciamos la captura desde capturador con el comando:

```
tshark -i eth0 -w - > /shared/captura.pcap
```

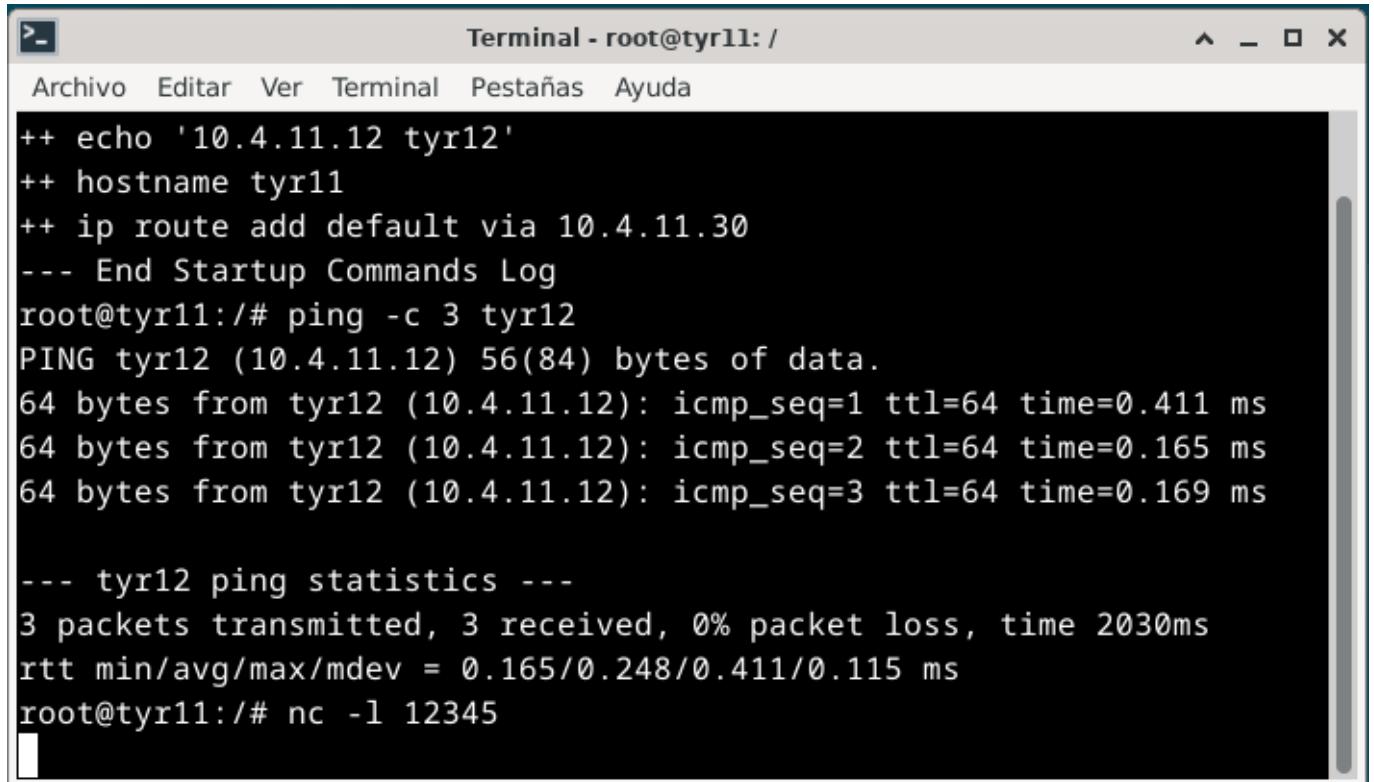


A terminal window titled "Terminal - root@capturador: /". The menu bar includes Archivo, Editar, Ver, Terminal, Pestañas, and Ayuda. The command "tshark -i eth0 -w - > /shared/captura\_nc.pcap" is run, followed by a warning about running as user "root". The capture starts on interface "eth0". Log messages show the capture starting at 21:45:34.227735 and the file being set to "-".

```
root@capturador:/# tshark -i eth0 -w - > /shared/captura_nc.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
** (tshark:45) 21:45:34.227735 [Main MESSAGE] -- Capture started.
** (tshark:45) 21:45:34.228428 [Main MESSAGE] -- File: "-"
```

Ejecutamos la utilidad nc desde pc1 actuando como servidor, para lo que definimos el puerto 12345 y usamos el siguiente comando:

```
Nc -l 12345
```



A terminal window titled "Terminal - root@tyr11: /". The menu bar includes Archivo, Editar, Ver, Terminal, Pestañas, and Ayuda. The command "Nc -l 12345" is run. The system logs startup commands like "echo '10.4.11.12 tyr12'", "hostname tyr11", and "ip route add default via 10.4.11.30". A "ping" command is run to "tyr12" (10.4.11.12), showing three successful packets. Finally, "nc -l 12345" is run to listen for incoming connections.

```
++ echo '10.4.11.12 tyr12'
++ hostname tyr11
++ ip route add default via 10.4.11.30
--- End Startup Commands Log
root@tyr11:/# ping -c 3 tyr12
PING tyr12 (10.4.11.12) 56(84) bytes of data.
64 bytes from tyr12 (10.4.11.12): icmp_seq=1 ttl=64 time=0.411 ms
64 bytes from tyr12 (10.4.11.12): icmp_seq=2 ttl=64 time=0.165 ms
64 bytes from tyr12 (10.4.11.12): icmp_seq=3 ttl=64 time=0.169 ms

--- tyr12 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2030ms
rtt min/avg/max/mdev = 0.165/0.248/0.411/0.115 ms
root@tyr11:/# nc -l 12345
```

Usamos el comando: `nc 10.4.11.11 12345` para establecer la conexión



A terminal window titled "Terminal - root@tyr12: /". The menu bar includes Archivo, Editar, Ver, Terminal, Pestañas, and Ayuda. The terminal content shows a conversation between two hosts:

```
me resolution
root@tyr12:/# nc 10.4.11.11 12345
Hola
todo bien?
chau
nos vemos
^C
root@tyr12:/# nc 10.4.11.11 12345
Hola
Hola, como estas?
todo bien
me alegro, chau
nos vemos
```

Ahora vemos que tenemos una especie de chat compartido entre ambos equipos

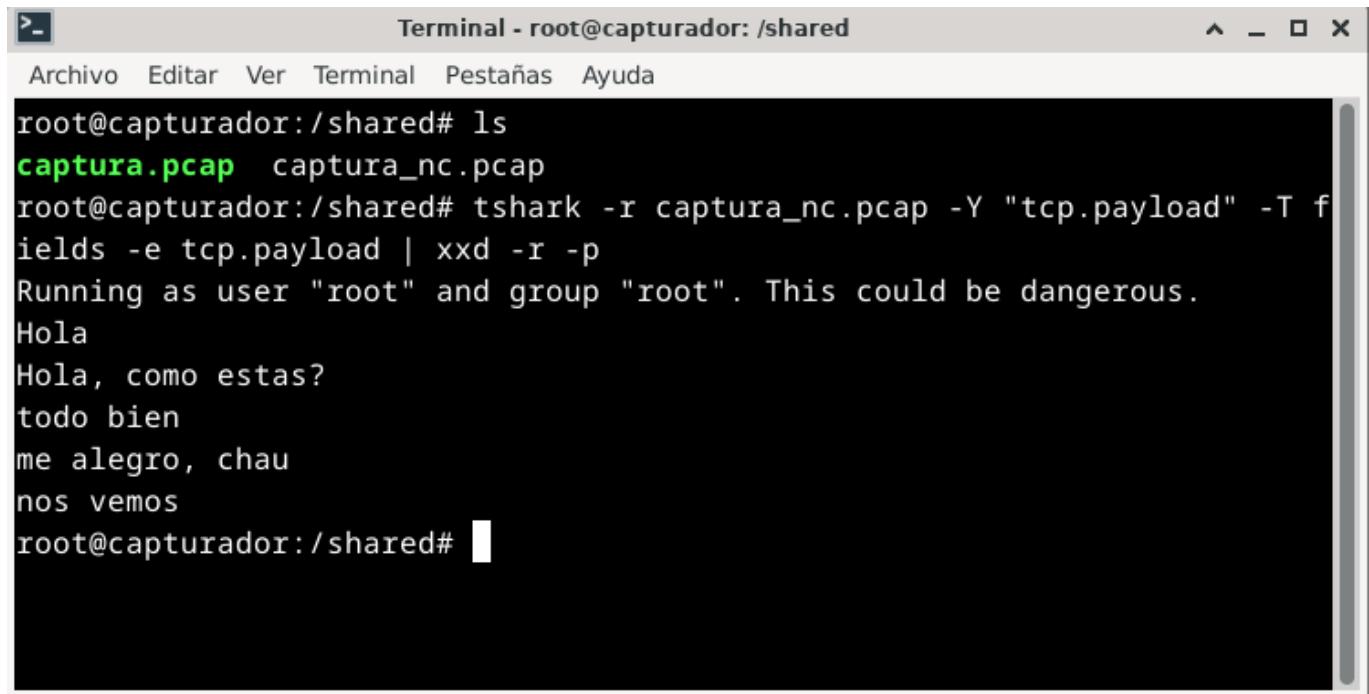
Finalizamos el chat con `ctrl +C` y terminamos la captura desde el host anfitrión. Ahora vamos a ver los datos intercambiados a nivel aplicación. Usamos el siguiente comando:

Analizamos la captura:

- Extraemos los datos intercambiados por la aplicación utilizando el comando:

```
tshark -r /shared/captura_nc.pcap -Y "tcp.payload" -T fields -e tcp.payload | xxd -r -p
```

y obtenemos:



A terminal window titled "Terminal - root@capturador: /shared". The menu bar includes Archivo, Editar, Ver, Terminal, Pestañas, and Ayuda. The terminal content shows the results of the tshark command:

```
root@capturador:/shared# ls
captura.pcap  captura_nc.pcap
root@capturador:/shared# tshark -r captura_nc.pcap -Y "tcp.payload" -T f
ields -e tcp.payload | xxd -r -p
Running as user "root" and group "root". This could be dangerous.
Hola
Hola, como estas?
todo bien
me alegro, chau
nos vemos
root@capturador:/shared#
```

Se puede ver que viaja el texto plano

Explicación del comando:

**tshark -r /shared/captura\_nc.pcap**

- tshark → Es la herramienta de línea de comandos de Wireshark para analizar capturas de red.
- -r /shared/captura\_nc.pcap → Indica que queremos **leer** (-r) el archivo de captura captura\_nc.pcap, ubicado en /shared/.

**-Y "tcp.payload"**

- -Y → Aplica un **filtro de visualización** (diferente de los filtros de captura).
- "tcp.payload" → Filtra solo los paquetes TCP que contienen **datos de aplicación**.
  - No incluye paquetes **sin datos** (como los SYN, ACK, FIN).
  - Solo se muestran paquetes con información útil intercambiada entre el cliente y el servidor.

**-T fields -e tcp.payload**

- -T fields → Cambia el formato de salida para mostrar solo **ciertos campos** (en este caso, tcp.payload).
- -e tcp.payload → Extrae **solo el campo** tcp.payload, que contiene la carga útil en hexadecimal.

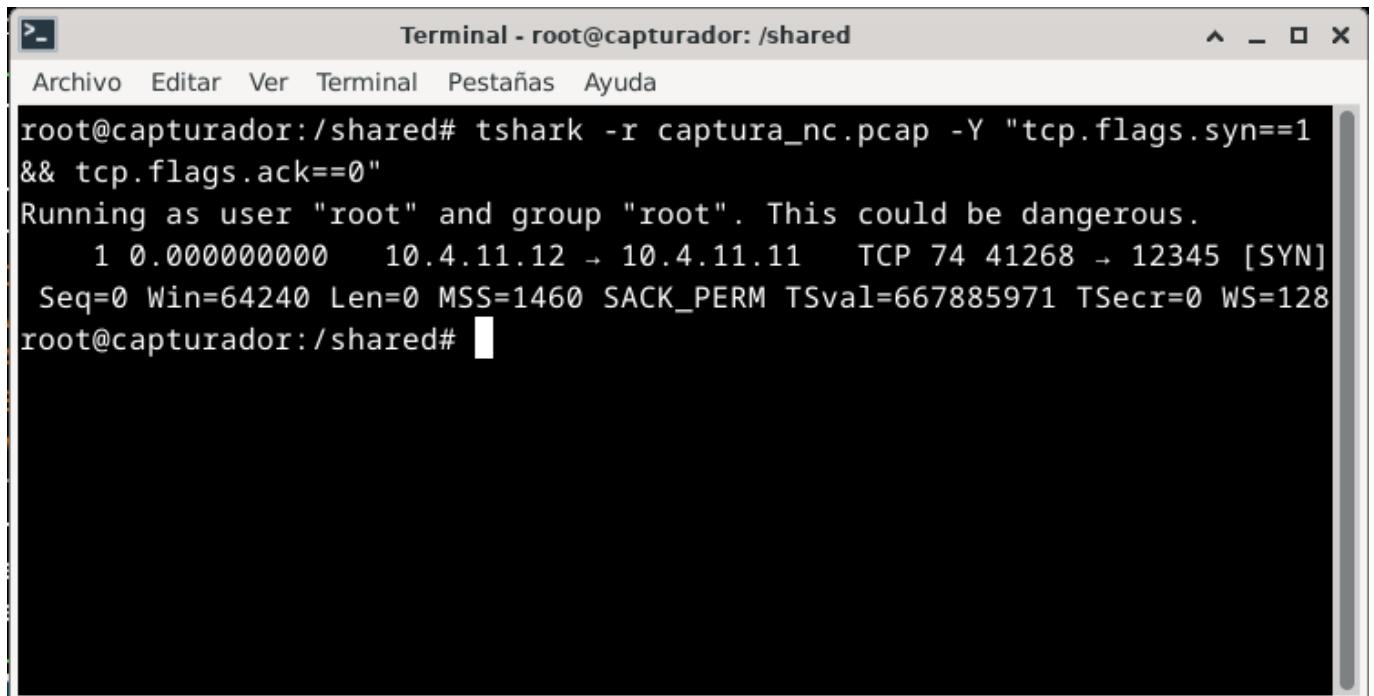
| **xxd -r -p**

- | → Pasa la salida de tshark al siguiente comando (xxd).
- xxd -r -p → Convierte los datos de **hexadecimal a texto ASCII legible**.
  - -r → Modo "reversa": convierte desde hexadecimal a texto normal.
  - -p → Lee la entrada como una secuencia de bytes en formato "plain" (sin direcciones ni espacios extra).

Gráfico con las entradas y salidas de los paquetes

Usamos el siguiente comando para filtrar los paquetes SYN enviados por el cliente, que inician la conexión TCP:

```
tshark -r captura_nc.pcap -Y "tcp.flags.syn==1 && tcp.flags.ack==0"
```

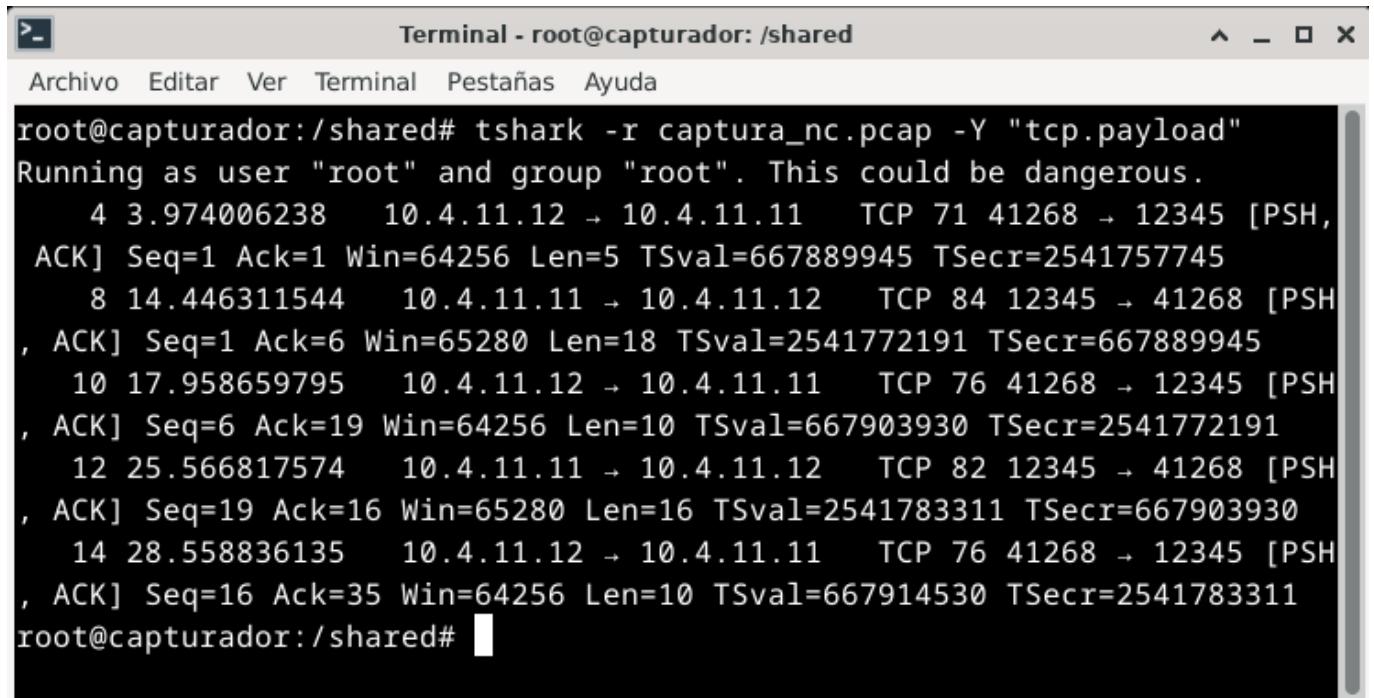


```
Terminal - root@capturador: /shared
Archivo Editar Ver Terminal Pestañas Ayuda
root@capturador:/shared# tshark -r captura_nc.pcap -Y "tcp.flags.syn==1 && tcp.flags.ack==0"
Running as user "root" and group "root". This could be dangerous.
 1 0.000000000 10.4.11.12 → 10.4.11.11  TCP 74 41268 → 12345 [SYN]
 Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=667885971 TSecr=0 WS=128
root@capturador:/shared#
```

*Establecimiento de la conexión TCP: Utiliza un filtro para seleccionar los paquetes que tienen la bandera SYN establecida y ACK desactivada.*

Para ver la transmisión de datos usamos el siguiente comando, para ver los paquetes que contienen datos de aplicación transmitidos por tcp:

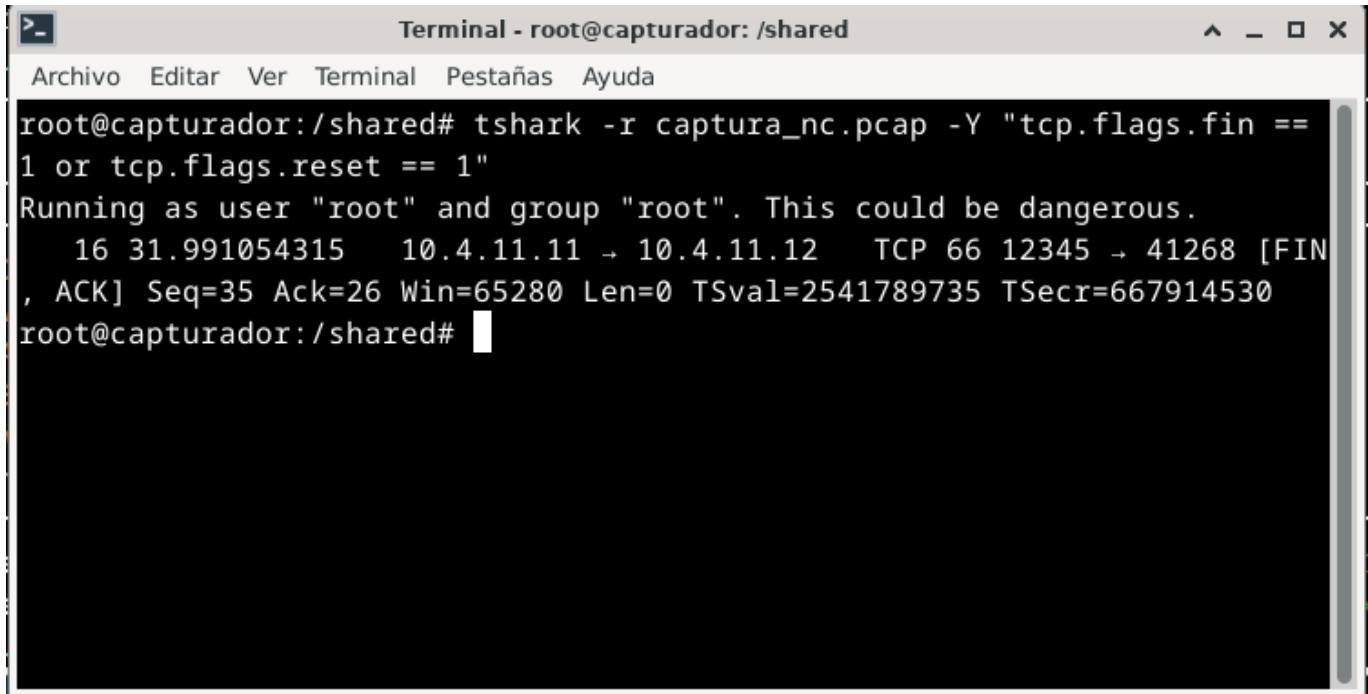
```
tshark -r captura_nc.pcap -Y "tcp.payload"
```



```
Terminal - root@capturador: /shared
Archivo Editar Ver Terminal Pestañas Ayuda
root@capturador:/shared# tshark -r captura_nc.pcap -Y "tcp.payload"
Running as user "root" and group "root". This could be dangerous.
 4 3.974006238 10.4.11.12 → 10.4.11.11  TCP 71 41268 → 12345 [PSH,
 ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=667889945 TSecr=2541757745
 8 14.446311544 10.4.11.11 → 10.4.11.12  TCP 84 12345 → 41268 [PSH
 , ACK] Seq=1 Ack=6 Win=65280 Len=18 TSval=2541772191 TSecr=667889945
 10 17.958659795 10.4.11.12 → 10.4.11.11  TCP 76 41268 → 12345 [PSH
 , ACK] Seq=6 Ack=19 Win=64256 Len=10 TSval=667903930 TSecr=2541772191
 12 25.566817574 10.4.11.11 → 10.4.11.12  TCP 82 12345 → 41268 [PSH
 , ACK] Seq=19 Ack=16 Win=65280 Len=16 TSval=2541783311 TSecr=667903930
 14 28.558836135 10.4.11.12 → 10.4.11.11  TCP 76 41268 → 12345 [PSH
 , ACK] Seq=16 Ack=35 Win=64256 Len=10 TSval=667914530 TSecr=2541783311
root@capturador:/shared#
```

Para ver el cierre de conexión usamos el siguiente comando:

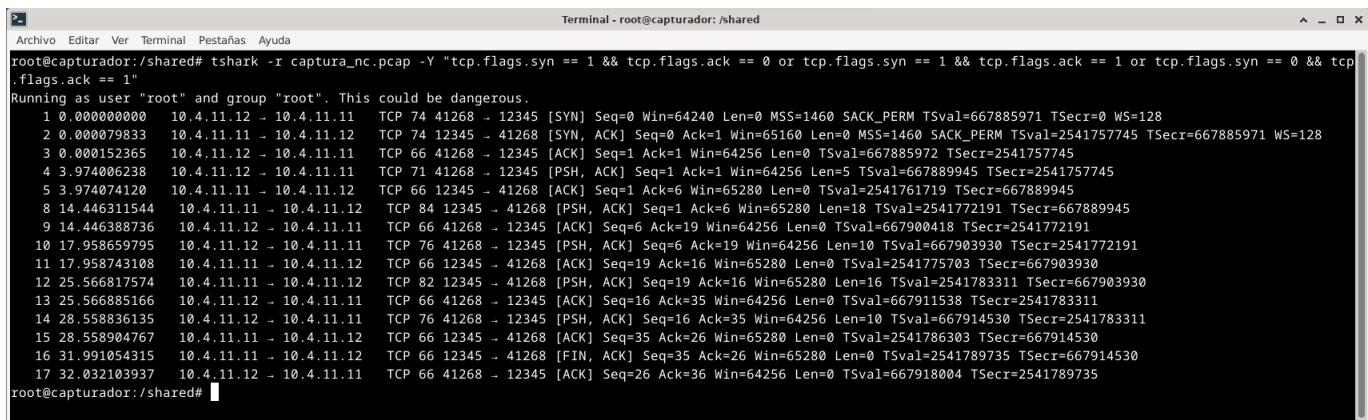
```
tshark -r captura_nc.pcap -Y "tcp.flags.fin == 1 or tcp.flags.reset == 1"
```



```
Terminal - root@capturador: /shared
Archivo Editar Ver Terminal Pestañas Ayuda
root@capturador:/shared# tshark -r captura_nc.pcap -Y "tcp.flags.fin == 1 or tcp.flags.reset == 1"
Running as user "root" and group "root". This could be dangerous.
16 31.991054315 10.4.11.11 -> 10.4.11.12 TCP 66 12345 -> 41268 [FIN, ACK] Seq=35 Ack=26 Win=65280 Len=0 TSval=2541789735 TSecr=667914530
root@capturador:/shared#
```

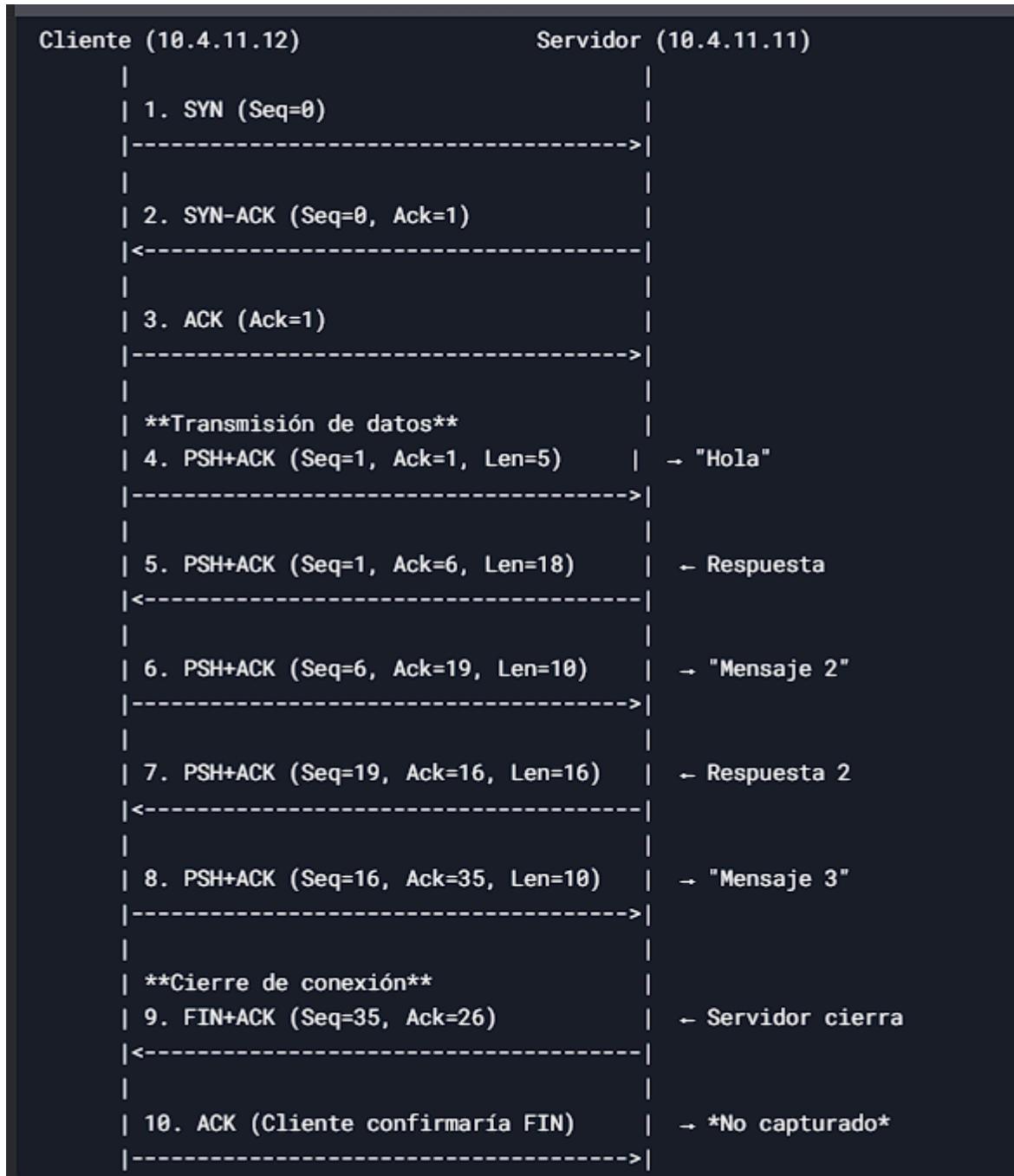
Con este comando veo el 3 way handshake:

```
tshark -r captura_nc.pcap -Y "tcp.flags.syn == 1 && tcp.flags.ack == 0 or tcp.flags.syn == 1 && tcp.flags.ack == 1 or tcp.flags.syn == 0 && tcp.flags.ack == 1"
```



```
Terminal - root@capturador: /shared
Archivo Editar Ver Terminal Pestañas Ayuda
root@capturador:/shared# tshark -r captura_nc.pcap -Y "tcp.flags.syn == 1 && tcp.flags.ack == 0 or tcp.flags.syn == 1 && tcp.flags.ack == 1 or tcp.flags.syn == 0 && tcp.flags.ack == 1"
Running as user "root" and group "root". This could be dangerous.
1 0.000000000 10.4.11.12 -> 10.4.11.11 TCP 74 41268 -> 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=667885971 TSecr=0 WS=128
2 0.000079833 10.4.11.11 -> 10.4.11.12 TCP 74 12345 -> 41268 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2541757745 TSecr=667885971 WS=128
3 0.000152365 10.4.11.12 -> 10.4.11.11 TCP 66 41268 -> 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=667885972 TSecr=2541757745
4 3.974006238 10.4.11.12 -> 10.4.11.11 TCP 71 41268 -> 12345 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=667889945 TSecr=2541757745
5 3.974074120 10.4.11.11 -> 10.4.11.12 TCP 66 12345 -> 41268 [ACK] Seq=1 Ack=6 Win=65280 Len=0 TSval=2541761719 TSecr=667889945
8 14.446311544 10.4.11.11 -> 10.4.11.12 TCP 84 12345 -> 41268 [PSH, ACK] Seq=1 Ack=6 Win=65280 Len=18 TSval=2541772191 TSecr=667889945
9 14.446388736 10.4.11.12 -> 10.4.11.11 TCP 66 41268 -> 12345 [ACK] Seq=6 Ack=19 Win=64256 Len=0 TSval=667900418 TSecr=2541772191
10 17.958659795 10.4.11.12 -> 10.4.11.11 TCP 76 41268 -> 12345 [PSH, ACK] Seq=6 Ack=19 Win=64256 Len=10 TSval=667903930 TSecr=2541772191
11 17.958743108 10.4.11.11 -> 10.4.11.12 TCP 66 12345 -> 41268 [ACK] Seq=19 Ack=10 Win=65280 Len=0 TSval=2541775703 TSecr=667903930
12 25.566817574 10.4.11.11 -> 10.4.11.12 TCP 82 12345 -> 41268 [PSH, ACK] Seq=19 Ack=16 Win=65280 Len=16 TSval=2541783311 TSecr=667903930
13 25.566885166 10.4.11.12 -> 10.4.11.11 TCP 66 41268 -> 12345 [ACK] Seq=16 Ack=35 Win=64256 Len=0 TSval=667911538 TSecr=2541783311
14 28.558836135 10.4.11.12 -> 10.4.11.11 TCP 76 41268 -> 12345 [PSH, ACK] Seq=16 Ack=35 Win=64256 Len=10 TSval=667914530 TSecr=2541783311
15 28.558904767 10.4.11.11 -> 10.4.11.12 TCP 66 12345 -> 41268 [ACK] Seq=35 Ack=26 Win=65280 Len=0 TSval=2541786303 TSecr=667914530
16 31.991054315 10.4.11.11 -> 10.4.11.12 TCP 66 12345 -> 41268 [FIN, ACK] Seq=35 Ack=26 Win=65280 Len=0 TSval=2541789735 TSecr=667914530
17 32.032103937 10.4.11.12 -> 10.4.11.11 TCP 66 41268 -> 12345 [ACK] Seq=26 Ack=36 Win=64256 Len=0 TSval=667918004 TSecr=2541789735
root@capturador:/shared#
```

Diagrama de secuencia:



c) ¿Todas las tramas TCP transportan datos de aplicación?

Para comprobar esto usamos el siguiente comando:

```
tshark -r captura_nc.pcap -Y "tcp and not tcp.payload"
```

```
Terminal - root@capturador: /shared
Archivo Editar Ver Terminal Pestañas Ayuda
root@capturador:/shared# tshark -r captura_nc.pcap -y "tcp and not tcp.payload"
Running as user "root" and group "root". This could be dangerous.
tshark: The specified data link type "tcp and not tcp.payload" isn't valid
root@capturador:/shared# tshark -r captura_nc.pcap -Y "tcp and not tcp.payload"
Running as user "root" and group "root". This could be dangerous.
  1 0.000000000  10.4.11.12 → 10.4.11.11  TCP 74 41268 → 12345 [SYN] Seq=0 Win=64
240 Len=0 MSS=1460 SACK_PERM TSecr=0 TSval=667885971 WS=128
  2 0.000079833  10.4.11.11 → 10.4.11.12  TCP 74 12345 → 41268 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2541757745 TSecr=667885971 WS=128
  3 0.000152365  10.4.11.12 → 10.4.11.11  TCP 66 41268 → 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=667885972 TSecr=2541757745
  5 3.974074120  10.4.11.11 → 10.4.11.12  TCP 66 12345 → 41268 [ACK] Seq=1 Ack=6 Win=65280 Len=0 TSval=2541761719 TSecr=667889945
  9 14.446388736  10.4.11.12 → 10.4.11.11  TCP 66 41268 → 12345 [ACK] Seq=6 Ack=19 Win=64256 Len=0 TSval=667900418 TSecr=2541772191
  11 17.958743108  10.4.11.11 → 10.4.11.12  TCP 66 12345 → 41268 [ACK] Seq=19 Ack=16 Win=65280 Len=0 TSval=2541775703 TSecr=667903930
  13 25.566885166  10.4.11.12 → 10.4.11.11  TCP 66 41268 → 12345 [ACK] Seq=16 Ack=35 Win=64256 Len=0 TSval=667911538 TSecr=2541783311
  15 28.558904767  10.4.11.11 → 10.4.11.12  TCP 66 12345 → 41268 [ACK] Seq=35 Ack=26 Win=65280 Len=0 TSval=2541786303 TSecr=667914530
  16 31.991054315  10.4.11.11 → 10.4.11.12  TCP 66 12345 → 41268 [FIN, ACK] Seq=35 Ack=26 Win=65280 Len=0 TSval=2541789735 TSecr=667914530
  17 32.032103937  10.4.11.12 → 10.4.11.11  TCP 66 41268 → 12345 [ACK] Seq=26 Ack=36 Win=64256 Len=0 TSval=667918004 TSecr=2541789735
root@capturador:/shared#
```

Esto muestra paquetes TCP **sin datos de aplicación**, como:

- **Handshake (SYN, SYN-ACK, ACK)**
- **ACK de confirmación de recepción**
- **FIN / RST (cierre de conexión)**

💡 **Conclusión:** No todas las tramas TCP transportan datos de aplicación, ya que muchas de ellas son de control (handshake, ACKs, cierre de conexión).

Las tramas TCP transportan datos de aplicación. TCP también se utiliza para el control de flujo y la gestión de la conexión. Por lo tanto, existen tramas TCP que no contienen datos de aplicación, como paquetes de control de conexión (SYN, ACK) o paquetes de cierre de conexión (FIN, FIN-ACK).

## Segunda parte: Protocolo de acceso remoto TELNET

Instale e inicie en Kathará el laboratorio de Telnet provisto por los docentes, disponible en [https://github.com/redesunlu/kathara-labs/blob/main/tarballs/kathara-lab\\_telnet.tar.gz](https://github.com/redesunlu/kathara-labs/blob/main/tarballs/kathara-lab_telnet.tar.gz)

El laboratorio cuenta con dos hosts. El primer host actuará como cliente telnet (client), mientras que el segundo host actuará como servidor remoto de telnet (remote).

Asigne una dirección IP al host cliente dentro de la red 172.16.0.0/24 . (puede elegir cualquiera del rango 172.16.0.1-254 excepto 172.16.0.10 )

En el dispositivo capturador inicie la captura utilizando el comando tcpdump o tshark sobre la interfaz eth0 y redirigir la salida a un archivo en el directorio /shared para su posterior análisis. (Ej. “ tshark -i eth0 -w - > /shared/captura\_telnet.pcap ”)

En la terminal del host cliente, conéctese mediante telnet al host remoto, cuya dirección IP es 172.16.0.10 . Utilice el nombre de usuario alumno y la clave ultrasecreta .

Con la sesión iniciada en remoto, ejecute el siguiente comando respetando la sintaxis.

```
who && who | openssl dgst
```

Copie la salida de dicho comando como resolución de este ejercicio (como texto). Añada además todos los comandos que ejecutó para lograr dicho resultado.

Salga del host remoto escribiendo el comando

```
exit
```

Luego detenga la captura en el dispositivo capturador. Remítala en formato pcap como parte de la tarea.

Analice la captura:

- a) Identifique e indique las tramas que corresponden a la transmisión de datos a nivel aplicación, cuáles a protocolos auxiliares (si existen) y al establecimiento y cierre de la conexión TCP. (referenciando por número de trama en la captura)
- b) Comente las características de la información en tránsito con respecto a la confidencialidad.

## Bibliografía

[FOR09] Capítulo 17: “Introduction to the Application Layer” (hasta pagina 546)

[FOR09] Capítulo 20: “Remote Login: TELNET and SSH”

## Recursos en internet

- Para cada uno de los protocolos (TELNET, HTTP, DNS, FTP, etc.) a desarrollar a lo largo de la cursada, busque cuales son los Request For Comments (RFC) o Internet Draft que los describen, siguiendo la cadena de actualizaciones. Recurra a <http://www.faqs.org>, <http://www.rfc-editor.org>, y <http://www.ietf.org>

## Referencia

[FOR09] FOROUZAN, B.A. TCP IP Protocol Suite. McGraw-Hill Higher Education, 2009

## Preguntas (guía de lectura)

En la capa de aplicación ¿a qué se denomina cliente y a qué servidor?.

En el stack TCP/IP, ¿cómo es posible que un protocolo de transporte brinde servicio a n procesos ejecutándose en un mismo host?.

¿Cuáles son las características y usos del protocolo telnet?.

¿Qué problemática resuelve NVT?

Asignamos una dirección ip al host cliente : 172.16.0.11 con el comando:

```
Ip addr add dev eth0 172.16.0.11/24
```

```
tyr-2024 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Aplicaciones Lugares Sistema sábado 6 de abr, 21:33
root@client:/#
root@client:/# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
6: eth0@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 0a:af:21:ef:b9:35 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.16.0.11/24 scope global eth0
        valid_lft forever preferred_lft forever
root@client:/#
```

Iniciamos captura

```
tshark -i kt-6d19f3ba623d -w capTelnet.pcap
```

```
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$ tshark -i kt-6d19f3ba623d -w capTelnet.pcap
Capturing on 'kt-6d19f3ba623d'
** (tshark:1531) 21:41:16.406029 [Main MESSAGE] -- Capture started.
** (tshark:1531) 21:41:16.406201 [Main MESSAGE] -- File: "capTelnet.pcap"
```

Nos conectamos mediante telnet al host remoto:

```
telnet 172.16.0.10
```

```
root@client:/ 
Archivo Editar Ver Buscar Terminal Ayuda
root@client:/# telnet 172.16.0.10
Trying 172.16.0.10...
Connected to 172.16.0.10.
Escape character is '^]'.

Linux 6.1.0-18-amd64 (remote) (pts/2)

remote login: alumno
Password:
Linux remote 6.1.0-18-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.76-1 (2024-02-01
) x86_64
=====
Bienvenido al servidor en la Antartida.

Continue el ejercicio ejecutando el comando:

    who && who | openssl dgst

Copie y pegue la salida de este en la resolucion de su practica.
=====

No directory, logging in with HOME=/
```

Ejecutamos el comando indicado:

```
who && who | openssl dgst
```

```
$ who && who | openssl dgst
SHA256(stdin)= e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85
5
$ who && who | openssl dgst
SHA256(stdin)= e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85
5
$ █
```

```
SHA256(stdin) = e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Para ver las tramas que contienen protocolos arp o icmp, protocolos auxiliares

```
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$ tshark -r capTelnet.pcap -Y 'arp || icmp'
 2 0.781812998 0a:af:21:ef:b9:35 → Broadcast      ARP 42 Who has 172.16.0.10?
Tell 172.16.0.11
 3 0.981883403 92:2d:ea:8e:6b:da → 0a:af:21:ef:b9:35 ARP 42 172.16.0.10 is at 92:2d:ea:8e:6b:da
```

No hubo intercambio de información:

```
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$ tshark -r capTelnet.pcap -Y 'telnet && !tcp.flags.syn && !tcp.flags.fin'
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$
```

- 
- a) Las tramas que corresponden a la transmisión de datos a nivel de aplicación son desde el número 7 a la 124 (no son consecutivos en todos los casos, lo cual es normal).

Se usa el siguiente comando:

```
tshark -r capTelnet.pcap -Y "telnet"
```

```
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$ tshark -r capTelnet.pcap -Y 'telnet'
```

```
 7 1.186977732 172.16.0.11 → 172.16.0.10  TELNET 96 Telnet Data ...
 9 1.399609378 172.16.0.10 → 172.16.0.11  TELNET 87 Telnet Data ...
11 1.399823315 172.16.0.11 → 172.16.0.10  TELNET 81 Telnet Data ...
12 1.600022703 172.16.0.10 → 172.16.0.11  TELNET 98 Telnet Data ...
15 1.643844904 172.16.0.11 → 172.16.0.10  TELNET 139 Telnet Data ...
16 1.843910453 172.16.0.10 → 172.16.0.11  TELNET 84 Telnet Data ...
19 1.844118119 172.16.0.11 → 172.16.0.10  TELNET 100 Telnet Data ...
21 2.044600591 172.16.0.10 → 172.16.0.11  TELNET 69 Telnet Data ...
22 2.044652973 172.16.0.11 → 172.16.0.10  TELNET 69 Telnet Data ...
23 2.245497470 172.16.0.10 → 172.16.0.11  TELNET 82 Telnet Data ...
24 2.245689195 172.16.0.11 → 172.16.0.10  TELNET 75 Telnet Data ...
25 2.445935420 172.16.0.10 → 172.16.0.11  TELNET 165 Telnet Data ...
27 12.015628776 172.16.0.11 → 172.16.0.10  TELNET 67 Telnet Data ...
28 12.215787519 172.16.0.10 → 172.16.0.11  TELNET 67 Telnet Data ...
29 12.215812890 172.16.0.11 → 172.16.0.10  TELNET 67 Telnet Data ...
30 12.415990858 172.16.0.10 → 172.16.0.11  TELNET 67 Telnet Data ...
31 12.416014378 172.16.0.11 → 172.16.0.10  TELNET 67 Telnet Data ...
```

para ver las tramas de protocolos auxiliares uso el siguiente comando:

```
tshark -r capTelnet.pcap -Y "arp or icmp"
```

```
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$ tshark -r capTelnet.pcap -Y 'arp or icmp'
 2 0.781812998 0a:af:21:ef:b9:35 → Broadcast      ARP 42 Who has 172.16.0.10?
Tell 172.16.0.11
 3 0.981883403 92:2d:ea:8e:6b:da → 0a:af:21:ef:b9:35 ARP 42 172.16.0.10 is at 92:2d:ea:8e:6b:da
```

Para ver las tramas de establecimiento y cierre de conexión TCP (SYN, FIN, etc.)

```
Tshark -r capTelnet.pcap -Y "tcp.flags.syn or tcp.flags.fin"
```

```
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$ tshark -r capTelnet.pcap -Y 'tcp.flags.syn or tcp.flags.fin'
    4 0.981927374 172.16.0.11 → 172.16.0.10 TCP 74 57410 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2088897137 TSecr=0 WS=128
    5 1.182762006 172.16.0.10 → 172.16.0.11 TCP 74 23 → 57410 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3662339754 TSecr=2088897137 WS=128
```

Las de establecimiento de conexión (SYN y SYN-ACK)

```
125 591.152807660 172.16.0.10 → 172.16.0.11 TCP 66 23 → 57410 [FIN, ACK] Seq=960 Ack=286 Win=65152 Len=0 TSval=3662929725 TSecr=2089487307
126 591.152866232 172.16.0.11 → 172.16.0.10 TCP 66 57410 → 23 [FIN, ACK] Seq=286 Ack=961 Win=64128 Len=0 TSval=2089487508 TSecr=3662929725
127 591.353617096 172.16.0.10 → 172.16.0.11 TCP 66 23 → 57410 [ACK] Seq=961 Ack=287 Win=65152 Len=0 TSval=3662929925 TSecr=2089487508
alumno@tyr-2024:~/workspace/kathara-labs/kathara-lab_telnet$ █
```

Las de fin de conexión

- b) Dado que la captura muestra tramas de datos Telnet sin cifrar, se confirma la falta de confidencialidad en la comunicación. Los datos transmitidos, como se puede ver en las tramas identificadas, son legibles y están expuestos a cualquier persona que pueda interceptar el tráfico de red. Esta exposición directa de los datos puede poner en riesgo la confidencialidad de la información sensible transmitida a través de Telnet, ya que cualquier actor malintencionado con acceso a la red puede leer fácilmente el contenido de las comunicaciones. En resumen tiene 3 problemas graves:
- Exposición a la interceptación, ya que transmite datos en texto claro
  - Al transmitirse contraseñas y usuarios (como es el caso aca) no están cifrados
  - Posibilidad de ataques de intermediarios, ya que no autentica la identidad del servidor

Se puede ver que los datos no son cifrados con el siguiente comando:

```
tshark -r capTelnet.pcap -nqz follow,tcp,hex,0
```

```
000000A4 61
000000BD 61
000000A5 6c
000000BE 6c
000000A6 75
000000BF 75
000000A7 6d
000000C0 6d
000000A8 6e
000000C1 6e
000000A9 6f
000000C2 6f
000000AA 0d
000000C3 0d 0a
000000C5 50 61 73 73 77 6f 72 64 3a 20
000000AB 75
000000AC 6c
000000AD 74
000000AE 72
000000AF 61
000000B0 73
000000B1 65
000000B2 63
000000B3 72
000000B4 65
000000B5 74
000000B6 61
000000B7 0d
000000CF 0d 0a
000000D1 4c 69 6e 75 78 20 72 65 6d 6f 74 65 20 36 2e 31 .. Linux re mote 6.1
```

