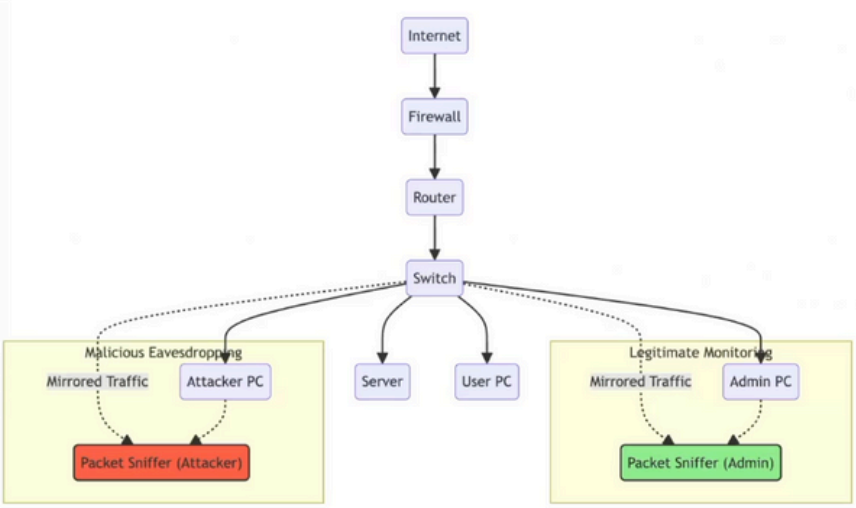
Packet Sniffing

Packet sniffing in networks and security refers to the practice of monitoring and capturing data packets as they traverse a network. This technique allows administrators or malicious actors to examine the content of packets, including source and destination addresses, protocols used, and potentially sensitive data like passwords or personal information.

Packet Sniffing

Promiscuous mode enables network interface cards (NICs) to capture all network traffic passing through the network segment they are connected to, regardless of whether the traffic is addressed to them. This mode is crucial for packet sniffing tools like tcpdump and Wireshark, allowing comprehensive network monitoring and analysis by capturing all packets on the network. It is essential for network troubleshooting, security analysis, and application development, but its use should adhere to privacy regulations to protect sensitive information transmitted over the network.

Packet Sniffing



Picture source: own creation

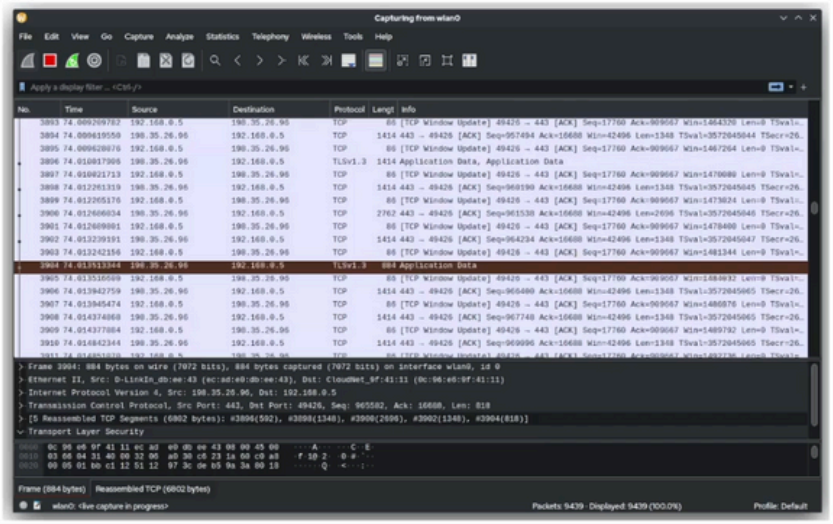
Packet Sniffing

TCP Header (packet)

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Data  | Rese|NS|CWR|ECE|URG|ACK|PSH|RST|SYN|FIN|    Window     |
| Offset| rved|  |   |   |   |   |   |   |   |   |               |
|       | Bit |  |   |   |   |   |   |   |   |   |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |        Urgent Pointer         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Data (if any)                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Packet Sniffing

Wireshark



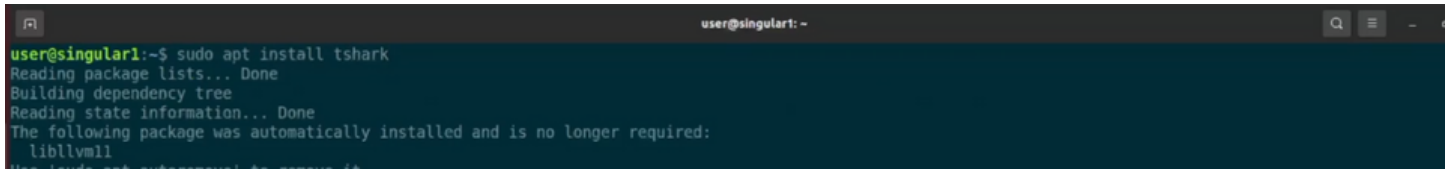
Wireshark: Wikipedia

tcpdump and tshark

TCPdump: Command-line packet analysis tool for capturing, filtering, and inspecting network packets in UNIX-like operating systems.

TShark: Command-line version of Wireshark, offering packet capturing and analysis features without a graphical user interface, useful for automated tasks or remote servers.

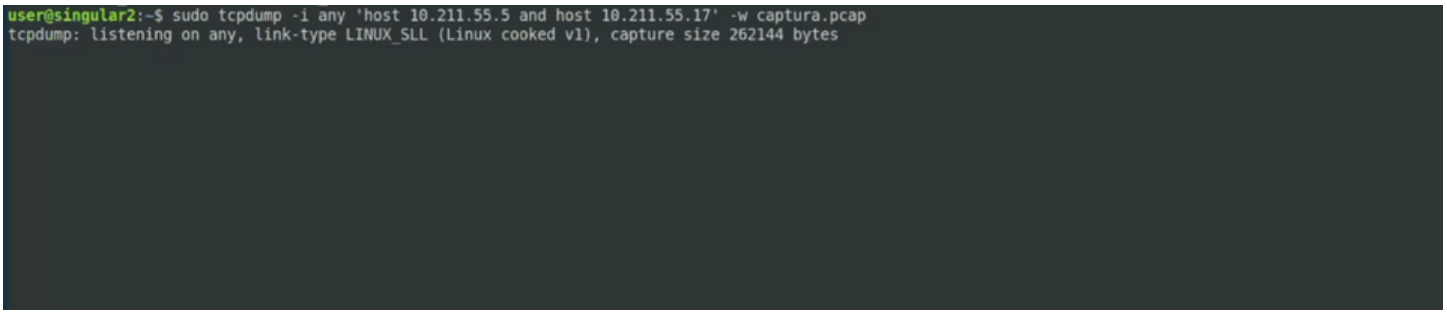
Wireshark: Wikipedia



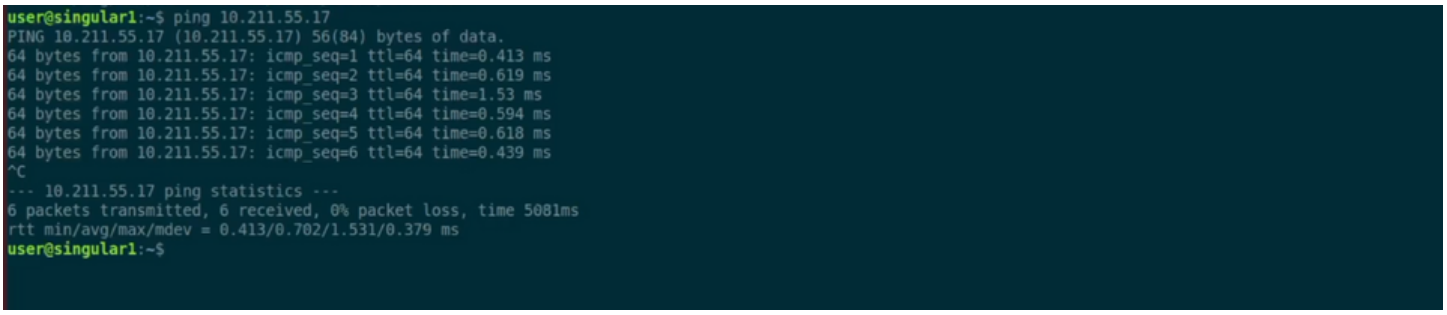
Para probar tcpdump y tshark vamos a generar un poco de tráfico entre dos máquinas con un ping, para ver que hay conectividad entre ambas máquinas.

Vamos a la maquina 2 que es la que va a interceptar toda la información, ya tiene tcpdump y tshark.

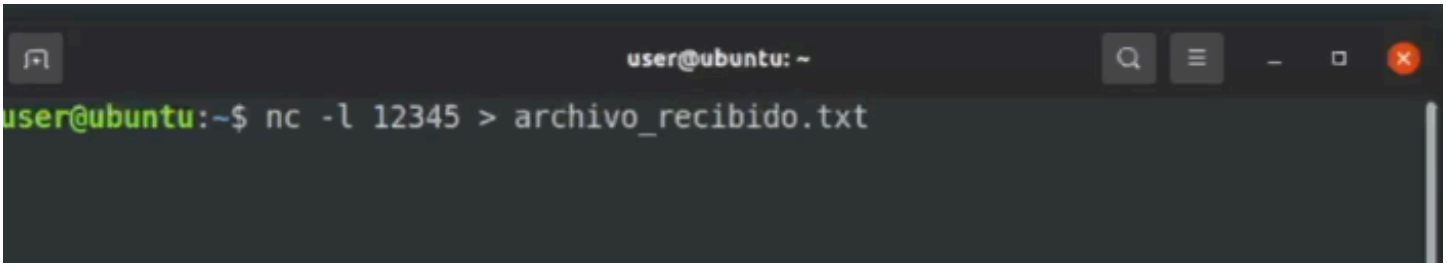
Empezamos a capturar la información con tcpdump:



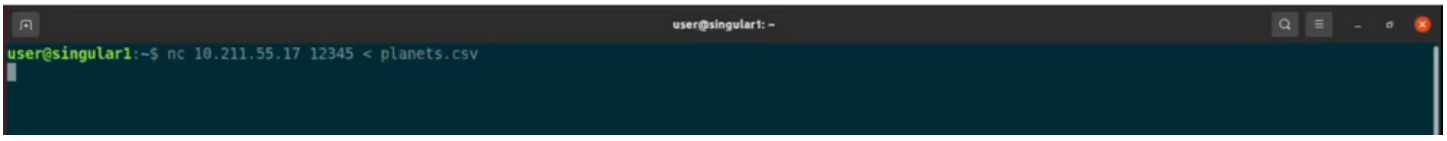
vamos a la otra máquina para lanzar un ping y dejarlo en marcha, para que tcpdump lo capture:



en la maquina que tenemos tcpdump escuchando vamos a lanzar otra terminal con netcap escuchando en un puerto para recibir un fichero (netcap sirve para la transferencia de archivos), para que tcpdump también capture dicha información:



volvemos a la otra máquina para enviar al fichero con netcap y recibirla en la máquina a la escucha:



Volvemos a la otra máquina, paramos el netcap y hacemos un cat, y vemos que hemos recibido el archivo exitosamente, aunque netcap a veces se queda colgado y hay que cerrarlo

manualmente.

Ahora vamos a analizar los paquetes que hemos interceptado con tcpdump.

Entramos en el fichero dónde hemos guardado todo el tráfico escuchado con tcpdump, para hacerlo tenemos que utilizar el siguiente comando con tshark:

```
user@singular2:~$ tshark -r captura.pcap -Y "ip.addr == 10.211.55.5 and ip.addr == 10.211.55.17"
```

Se ven todos los paquetes capturados:

```
user@singular2: ~
user@singular2:~$ tshark -r captura.pcap -Y "ip.addr == 10.211.55.5 and ip.addr == 10.211.55.17"
    1   0.000000  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=1/256, ttl=64
    2   0.000039 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=1/256, ttl=64 (request in 1)
    3   1.018579  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=2/512, ttl=64
    4   1.018628 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=2/512, ttl=64 (request in 3)
    5   2.044871  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=3/768, ttl=64
    6   2.044937 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=3/768, ttl=64 (request in 5)
    7   3.045083  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=4/1024, ttl=64
    8   3.045131 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=4/1024, ttl=64 (request in 7)
    9   4.056106  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=5/1280, ttl=64
   10   4.056155 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=5/1280, ttl=64 (request in 9)
   13   5.080883  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=6/1536, ttl=64
   14   5.080925 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=6/1536, ttl=64 (request in 13)
   17 247.997412  10.211.55.5 → 10.211.55.17 TCP 76 54500 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3897757879 TSecr=0 WS=128
   18 247.997444 10.211.55.17 → 10.211.55.5  TCP 76 12345 → 54500 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3925449672 TSecr=3897757879 WS
=128
   19 247.997574  10.211.55.5 → 10.211.55.17 TCP 68 54500 → 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3897757879 TSecr=3925449672
   20 247.998177  10.211.55.5 → 10.211.55.17 TCP 7308 54500 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=7240 TSval=3897757879 TSecr=3925449672
   21 247.998177  10.211.55.5 → 10.211.55.17 TCP 7308 54500 → 12345 [PSH, ACK] Seq=7241 Ack=1 Win=64256 Len=7240 TSval=3897757879 TSecr=3925449672
   22 247.998202 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 54500 [ACK] Seq=1 Ack=7241 Win=61312 Len=0 TSval=3925449673 TSecr=3897757879
   23 247.998215 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 54500 [ACK] Seq=1 Ack=14481 Win=57472 Len=0 TSval=3925449673 TSecr=3897757879
   24 247.998369  10.211.55.5 → 10.211.55.17 TCP 10204 54500 → 12345 [PSH, ACK] Seq=14481 Ack=1 Win=64256 Len=10136 TSval=3897757880 TSecr=3925449673
   25 247.998371 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 54500 [ACK] Seq=1 Ack=24617 Win=59776 Len=0 TSval=3925449673 TSecr=3897757880
   26 247.998451  10.211.55.5 → 10.211.55.17 TCP 11715 54500 → 12345 [PSH, ACK] Seq=24617 Ack=1 Win=64256 Len=11647 TSval=3897757880 TSecr=3925449673
   27 247.998454 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 54500 [ACK] Seq=1 Ack=36264 Win=57984 Len=0 TSval=3925449673 TSecr=3897757880
   32 383.727201 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 54500 [FIN, ACK] Seq=1 Ack=36264 Win=64128 Len=0 TSval=3925585402 TSecr=3897757880
   33 383.727594  10.211.55.5 → 10.211.55.17 TCP 68 54500 → 12345 [FIN, ACK] Seq=36264 Ack=2 Win=64256 Len=0 TSval=3897893607 TSecr=3925585402
   34 383.727606 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 54500 [ACK] Seq=2 Ack=36265 Win=64128 Len=0 TSval=3925585403 TSecr=3897893607
   39 474.283621  10.211.55.5 → 10.211.55.17 TCP 76 47064 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3897984161 TSecr=0 WS=128
   40 474.283676 10.211.55.17 → 10.211.55.5  TCP 76 12345 → 47064 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3925675959 TSecr=3897984161 WS
=128
   41 474.285047  10.211.55.5 → 10.211.55.17 TCP 68 47064 → 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3897984162 TSecr=3925675959
   42 474.285047  10.211.55.5 → 10.211.55.17 TCP 7308 47064 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=7240 TSval=3897984162 TSecr=3925675959
   43 474.285047  10.211.55.5 → 10.211.55.17 TCP 7308 47064 → 12345 [PSH, ACK] Seq=7241 Ack=1 Win=64256 Len=7240 TSval=3897984162 TSecr=3925675959
   44 474.285188 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 47064 [ACK] Seq=1 Ack=7241 Win=61312 Len=0 TSval=3925675960 TSecr=3897984162
   45 474.285232 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 47064 [ACK] Seq=1 Ack=14481 Win=57472 Len=0 TSval=3925675960 TSecr=3897984162
   46 474.285404  10.211.55.5 → 10.211.55.17 TCP 10204 47064 → 12345 [PSH, ACK] Seq=14481 Ack=1 Win=64256 Len=10136 TSval=3897984163 TSecr=3925675960
   47 474.285404  10.211.55.5 → 10.211.55.17 TCP 11715 47064 → 12345 [PSH, ACK] Seq=24617 Ack=1 Win=64256 Len=11647 TSval=3897984163 TSecr=3925675960
   48 474.285407 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 47064 [ACK] Seq=1 Ack=24617 Win=59520 Len=0 TSval=3925675960 TSecr=3897984163
   49 474.285418 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 47064 [ACK] Seq=1 Ack=36264 Win=52608 Len=0 TSval=3925675960 TSecr=3897984163
   54 518.069330 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 47064 [FIN, ACK] Seq=1 Ack=36264 Win=64128 Len=0 TSval=3925719744 TSecr=3897984163
   55 518.070669  10.211.55.5 → 10.211.55.17 TCP 68 47064 → 12345 [FIN, ACK] Seq=36264 Ack=2 Win=64256 Len=0 TSval=3898027949 TSecr=3925719744
   56 518.070694 10.211.55.17 → 10.211.55.5  TCP 68 12345 → 47064 [ACK] Seq=2 Ack=36265 Win=64128 Len=0 TSval=3925719746 TSecr=3898027949
user@singular2:~$
```

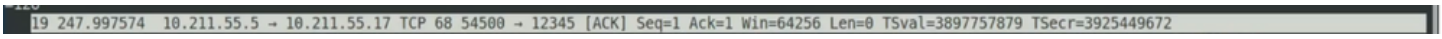
Los paquetes seleccionados son los del ping:

```
user@singular2: ~
user@singular2:~$ tshark -r captura.pcap -Y "ip.addr == 10.211.55.5 and ip.addr == 10.211.55.17"
    1   0.000000  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=1/256, ttl=64
    2   0.000039 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=1/256, ttl=64 (request in 1)
    3   1.018579  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=2/512, ttl=64
    4   1.018628 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=2/512, ttl=64 (request in 3)
    5   2.044871  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=3/768, ttl=64
    6   2.044937 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=3/768, ttl=64 (request in 5)
    7   3.045083  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=4/1024, ttl=64
    8   3.045131 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=4/1024, ttl=64 (request in 7)
    9   4.056106  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=5/1280, ttl=64
   10   4.056155 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=5/1280, ttl=64 (request in 9)
   13   5.080883  10.211.55.5 → 10.211.55.17 ICMP 100 Echo (ping) request  id=0x0003, seq=6/1536, ttl=64
   14   5.080925 10.211.55.17 → 10.211.55.5  ICMP 100 Echo (ping) reply    id=0x0003, seq=6/1536, ttl=64 (request in 13)
   17 247.997412  10.211.55.5 → 10.211.55.17 TCP 76 54500 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3897757879 TSecr=0 WS=128
```

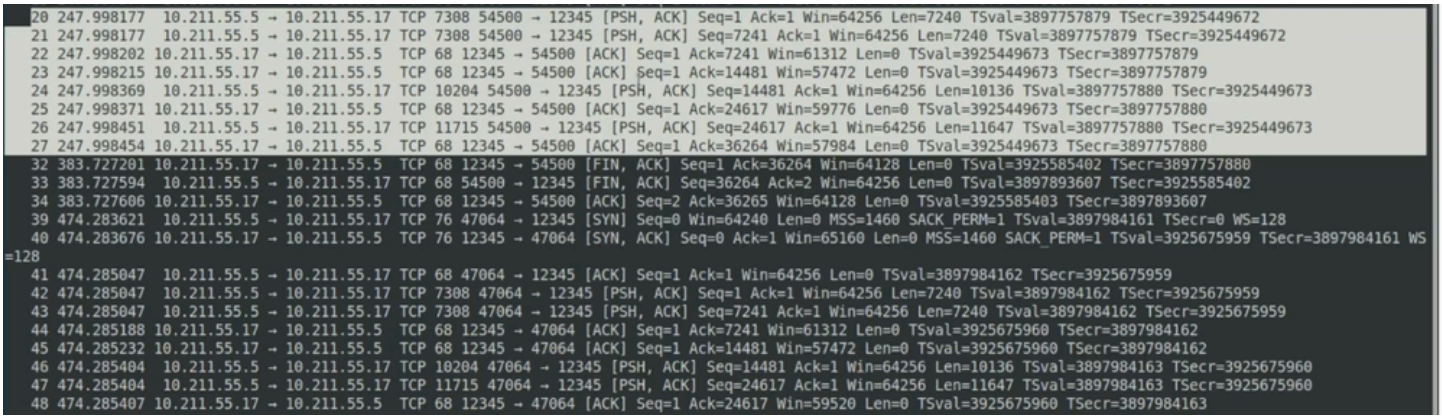
Éstos dos logs son los del netcap:

```
   17 247.997412  10.211.55.5 → 10.211.55.17 TCP 76 54500 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3897757879 TSecr=0 WS=128
   18 247.997444 10.211.55.17 → 10.211.55.5  TCP 76 12345 → 54500 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3925449672 TSecr=3897757879 WS
=128
```

El paquete 19 confirma que se ha completado el handshake del protocolo tcp:



Los paquetes del 20 al 27 y del 40 al 47 son los datos que se han enviado (con la flag PUSH y ACK)



Los paquetes del 32 al 34 indican la finalización del tcp:

Más en detalle, para ver la información que hemos capturado, el xxd es para poder leerlo en formato ASCII sino estará todo en formato hexadecimal: