Document de travail

ENSEA

Beyond Engineering

Architecture & Protocols

Lab 2

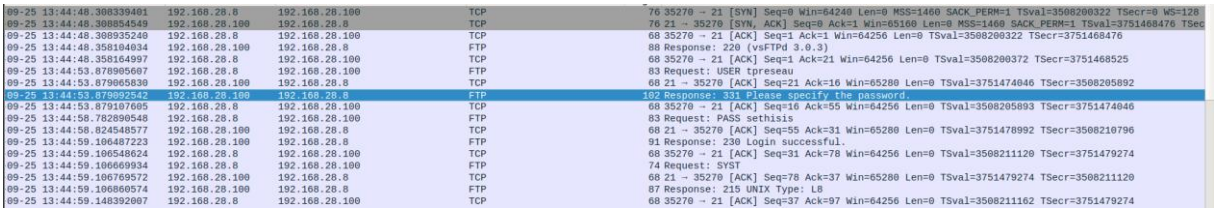
Report Practical Work

3ème année - RTS

PETIT Alexandre
Arthur LAUMY

TP2 REPORT

I – Capture FTP Session

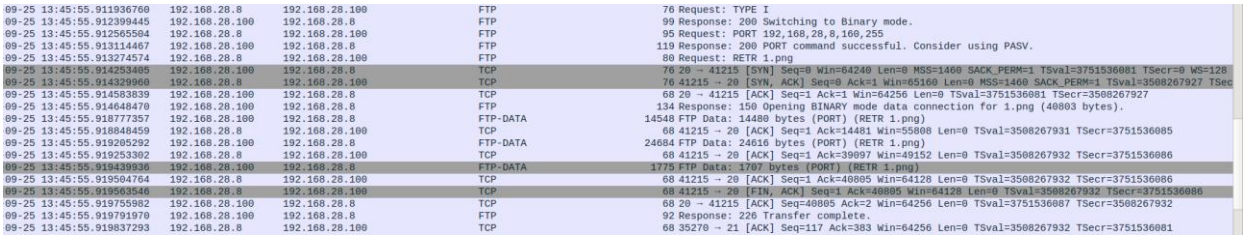


09-25 13:44:48.308339401	192.168.28.8	192.168.28.100	TCP	76 35270 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3508200322 TSecr=0 WS=128
09-25 13:44:48.308854549	192.168.28.100	192.168.28.8	TCP	76 21 → 35270 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3751468476 TSec
09-25 13:44:48.308935240	192.168.28.8	192.168.28.100	TCP	68 35270 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3508200322 TSecr=3751468476
09-25 13:44:48.358104034	192.168.28.100	192.168.28.8	FTP	88 Response: 220 (vsFTPd 3.0.3)
09-25 13:44:48.358164997	192.168.28.8	192.168.28.100	TCP	68 35270 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=3508200372 TSecr=3751468525
09-25 13:44:53.878905607	192.168.28.8	192.168.28.100	FTP	83 Request: USER tpreseau
09-25 13:44:53.879065830	192.168.28.100	192.168.28.8	TCP	68 21 → 35270 [ACK] Seq=21 Ack=16 Win=65280 Len=0 TSval=3751474046 TSecr=3508205892
09-25 13:44:53.879092542	192.168.28.100	192.168.28.8	FTP	102 Response: 331 Please specify the password.
09-25 13:44:53.879107605	192.168.28.8	192.168.28.100	TCP	68 35270 → 21 [ACK] Seq=16 Ack=55 Win=64256 Len=0 TSval=3508205893 TSecr=3751474046
09-25 13:44:58.782890548	192.168.28.8	192.168.28.100	FTP	83 Request: PASS sethisis
09-25 13:44:58.824548577	192.168.28.100	192.168.28.8	TCP	68 21 → 35270 [ACK] Seq=55 Ack=31 Win=65280 Len=0 TSval=3751478992 TSecr=3508210796
09-25 13:44:59.106487223	192.168.28.100	192.168.28.8	FTP	91 Response: 230 Login successful.
09-25 13:44:59.106548624	192.168.28.8	192.168.28.100	TCP	68 35270 → 21 [ACK] Seq=31 Ack=78 Win=64256 Len=0 TSval=3508211120 TSecr=3751479274
09-25 13:44:59.106669934	192.168.28.8	192.168.28.100	FTP	74 Request: SYST
09-25 13:44:59.106769572	192.168.28.100	192.168.28.8	TCP	68 21 → 35270 [ACK] Seq=78 Ack=37 Win=65280 Len=0 TSval=3751479274 TSecr=3508211120
09-25 13:44:59.106860574	192.168.28.100	192.168.28.8	FTP	87 Response: 215 UNIX Type: L8
09-25 13:44:59.148392007	192.168.28.8	192.168.28.100	TCP	68 35270 → 21 [ACK] Seq=37 Ack=97 Win=64256 Len=0 TSval=3508211162 TSecr=3751479274

Figure 1 – Capture Connection FTP

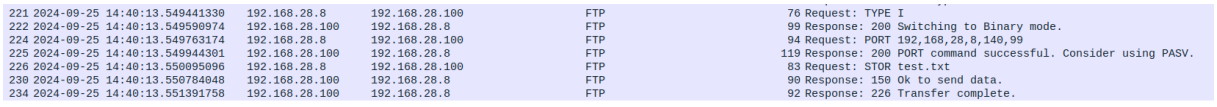
The connection between our computer and the computer with the IP address 192.168.28.100 is established using the TCP protocol. After this connection, we see an alternation between the TCP and FTP protocols. We can see that all these exchanges are not encrypted (the appearance of the password and the username). In fact, this protocol is so old that network security was not a real problem at the time.

The TCP protocol is used to manage requests to the storage server, identification management (user and password) and network connection management (ACK and synchronization). The FTP protocol is used for file transfer and listing.



09-25 13:45:55.911936760	192.168.28.8	192.168.28.100	FTP	76 Request: TYPE I
09-25 13:45:55.912399445	192.168.28.100	192.168.28.8	FTP	99 Response: 200 Switching to Binary mode.
09-25 13:45:55.912565504	192.168.28.8	192.168.28.100	FTP	95 Request: PORT 192,168,28,8,160,255
09-25 13:45:55.913114467	192.168.28.100	192.168.28.8	FTP	119 Response: 200 PORT command successful. Consider using PASV.
09-25 13:45:55.913274574	192.168.28.8	192.168.28.100	FTP	80 Request: RETR 1.png
09-25 13:45:55.914253405	192.168.28.100	192.168.28.8	TCP	76 20 → 41215 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3751536081 TSecr=0 WS=128
09-25 13:45:55.914329960	192.168.28.8	192.168.28.100	TCP	76 41215 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3508267927 TSec
09-25 13:45:55.914583839	192.168.28.100	192.168.28.8	TCP	68 20 → 41215 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3751536081 TSecr=3508267927
09-25 13:45:55.914648470	192.168.28.100	192.168.28.8	FTP	134 Response: 150 Opening BINARY mode data connection for 1.png (40803 bytes).
09-25 13:45:55.918777357	192.168.28.100	192.168.28.8	FTP-DATA	14548 FTP Data: 14480 bytes (PORT) (RETR 1.png)
09-25 13:45:55.918848459	192.168.28.8	192.168.28.100	TCP	68 41215 → 20 [ACK] Seq=1 Ack=14481 Win=55808 Len=0 TSval=3508267931 TSecr=3751536085
09-25 13:45:55.919205292	192.168.28.100	192.168.28.8	FTP-DATA	24684 FTP Data: 24616 bytes (PORT) (RETR 1.png)
09-25 13:45:55.919253302	192.168.28.8	192.168.28.100	TCP	68 41215 → 20 [ACK] Seq=1 Ack=39097 Win=49152 Len=0 TSval=3508267932 TSecr=3751536086
09-25 13:45:55.919439936	192.168.28.100	192.168.28.8	FTP-DATA	1775 FTP Data: 1707 bytes (PORT) (RETR 1.png)
09-25 13:45:55.919504764	192.168.28.8	192.168.28.100	TCP	68 41215 → 20 [ACK] Seq=1 Ack=40805 Win=64128 Len=0 TSval=3508267932 TSecr=3751536086
09-25 13:45:55.919563546	192.168.28.8	192.168.28.100	TCP	68 41215 → 20 [FIN, ACK] Seq=1 Ack=40805 Win=64128 Len=0 TSval=3508267932 TSecr=3751536086
09-25 13:45:55.919755982	192.168.28.100	192.168.28.8	TCP	68 20 → 41215 [ACK] Seq=40805 Ack=2 Win=64256 Len=0 TSval=3751536087 TSecr=3508267932
09-25 13:45:55.919791970	192.168.28.100	192.168.28.8	FTP	92 Response: 226 Transfer complete.
09-25 13:45:55.919837293	192.168.28.8	192.168.28.100	TCP	68 35270 → 21 [ACK] Seq=117 Ack=383 Win=64256 Len=0 TSval=3508267932 TSecr=3751536081

Figure 2 – Capture Get FTP



221 2024-09-25 14:40:13.549441330	192.168.28.8	192.168.28.100	FTP	76 Request: TYPE I
222 2024-09-25 14:40:13.549590974	192.168.28.100	192.168.28.8	FTP	99 Response: 200 Switching to Binary mode.
224 2024-09-25 14:40:13.549763174	192.168.28.8	192.168.28.100	FTP	94 Request: PORT 192,168,28,8,140,99
225 2024-09-25 14:40:13.549944301	192.168.28.100	192.168.28.8	FTP	119 Response: 200 PORT command successful. Consider using PASV.
226 2024-09-25 14:40:13.550095096	192.168.28.8	192.168.28.100	FTP	83 Request: STOR test.txt
230 2024-09-25 14:40:13.550784048	192.168.28.100	192.168.28.8	FTP	90 Response: 150 Ok to send data.
234 2024-09-25 14:40:13.551391758	192.168.28.100	192.168.28.8	FTP	92 Response: 226 Transfer complete.

Figure 3 – Capture Send FTP

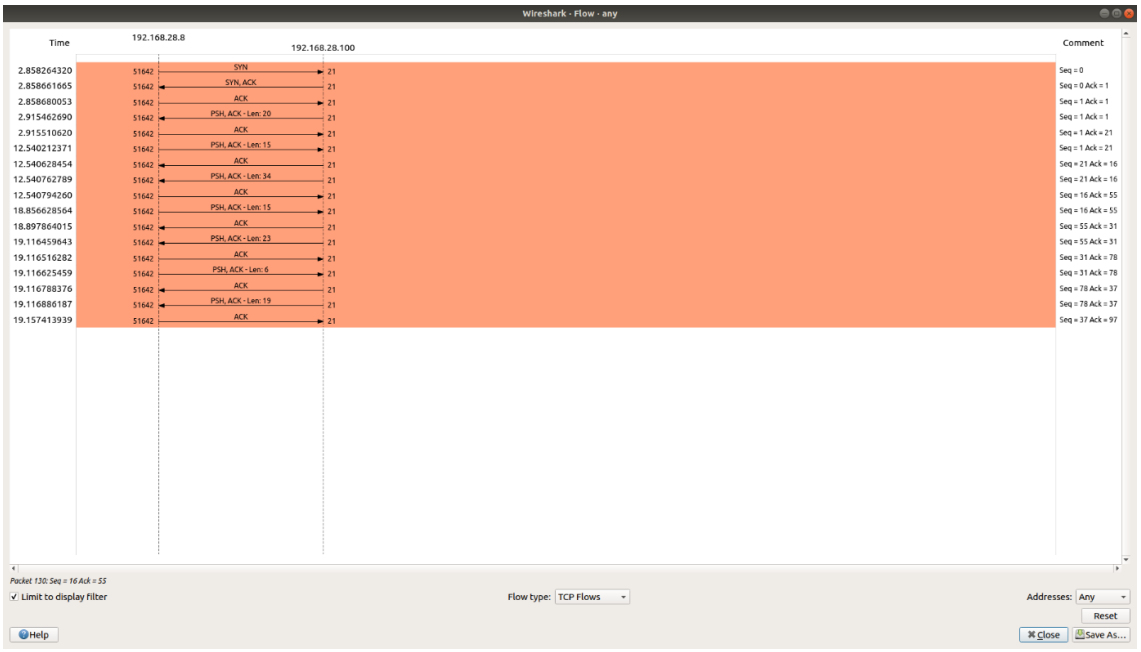


Figure 4 - Flow chart

You can see from the graph (obtained with Statistics > Flow Graph > TCP Flow) that the seq becomes the old ACK, clearly showing the next steps.

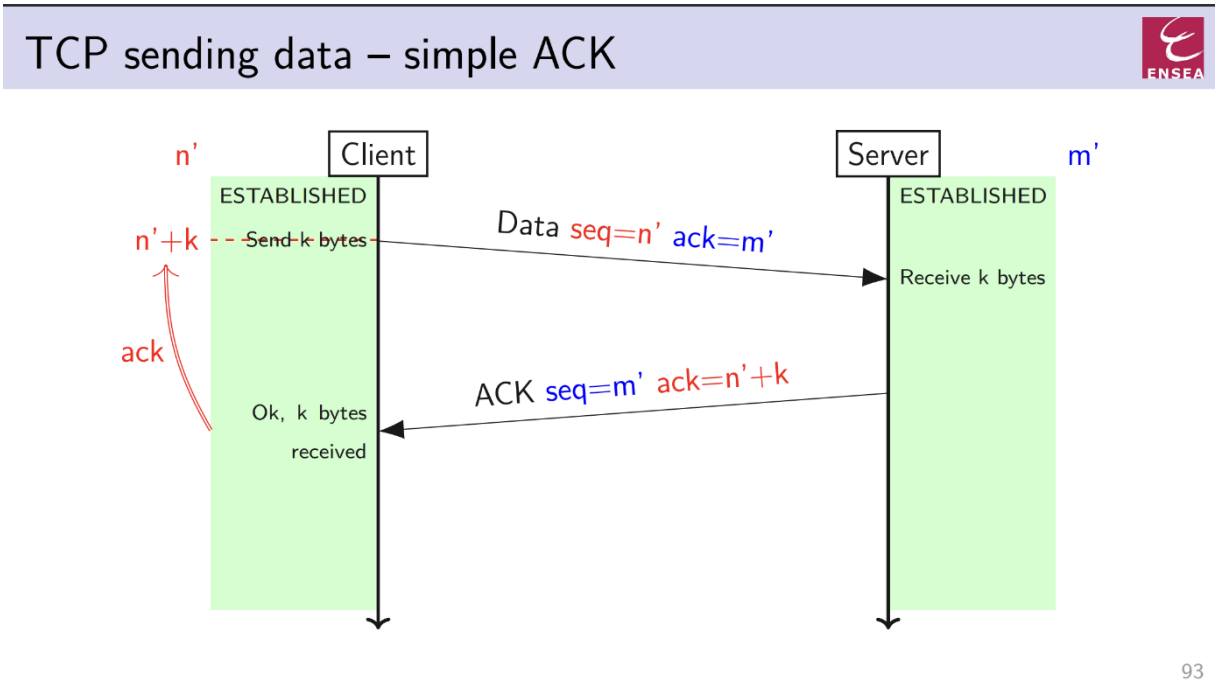


Figure 5 – Course Scheme

In fact, we can see that when ack=m'=21, the following seq is equal to 21. Furthermore, len in the flowchart corresponds to k in the graph. Therefore, we see that seq=21 and the length of the file to be sent is 34, so ack becomes m'+k=55. There is therefore a sequence of steps in this protocol that respects the TCP protocol.

II – MTU (Maximum Transfer Unit)

We use the ifconfig command to determine the MTU.

```
enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.28.8  netmask 255.255.255.0  broadcast 192.168.28.255
        inet6 fe80::56bf:64ff:fe64:a688  prefixlen 64  scopeid 0x20<link>
        ether 54:bf:64:64:a6:88  txqueuelen 1000  (Ethernet)
        RX packets 22465  bytes 21124303 (21.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12072  bytes 1666085 (1.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xef400000-ef420000
```

Figure 6 – MTU=1500

The MTU is now 1500. We then change it to 100 with the following command. We can see that the MTU has changed. We can note that the command need the administration rights.

```
tpreseau@d055-pc8:~$ ifconfig enp0s31f6 mtu 100
SIOCSIFMTU: Operation not permitted
tpreseau@d055-pc8:~$ sudo ifconfig enp0s31f6 mtu 100
[sudo] password for tpreseau:
tpreseau@d055-pc8:~$ ifconfig enp0s31f6
enp0s31f6: flags=4099<UP,BROADCAST,MULTICAST>  mtu 100
        inet 192.168.28.8  netmask 255.255.255.0  broadcast 192.168.28.255
        ether 54:bf:64:64:a6:88  txqueuelen 1000  (Ethernet)
        RX packets 23534  bytes 21275715 (21.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13127  bytes 1787456 (1.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xef400000-ef420000
```

Figure 7 – MTU=100

MTU stands for Maximum Transmission Unit. It is the maximum size of a packet that can be transmitted without being truncated. If it exceeds this size, the packet will be fragmented.

```
 32 2024-09-25 15:07:55.276368894   192.168.28.8    192.168.28.7    ICMP    92 Echo (ping) request  id=0x2012, seq=2/512, ttl=64 (reply in 33)
 33 2024-09-25 15:07:55.276525850   192.168.28.7    192.168.28.8    ICMP    92 Echo (ping) reply    id=0x2012, seq=2/512, ttl=64 (request in 32)
```

Figure 8– Protocol ICMP with Ping 50 bytes

We can see that when we ping 50 bytes there is no fragmentation for an MTU=100. We can see that the packet size is 92, which means that there is no fragmentation because the size is less than the MTU.

On the other hand, if we ping 80 bytes, we see that it is truncated because the packet size is 102. Thanks to the IPv4 protocol, we can see that the packets are truncated into two, one of size 42 and the other of size 60. So 80-byte ICMP packets are truncated.

```
120 2024-09-25 15:08:10.126178353   192.168.28.8    192.168.28.7    IPv4    114 Fragmented IP protocol (proto=ICMP 1, off=0, ID=c9ee) [Reassembled in #121]
121 2024-09-25 15:08:10.126187299   192.168.28.8    192.168.28.7    ICMP     42 Echo (ping) request  id=0x2013, seq=1/256, ttl=64 (reply in 123)
122 2024-09-25 15:08:10.126427339   192.168.28.7    192.168.28.8    IPv4    114 Fragmented IP protocol (proto=ICMP 1, off=0, ID=7d25) [Reassembled in #123]
123 2024-09-25 15:08:10.126442344   192.168.28.7    192.168.28.8    ICMP     60 Echo (ping) reply    id=0x2013, seq=1/256, ttl=64 (request in 121)
```

Figure 9– Protocol ICMP with Ping 80 bytes

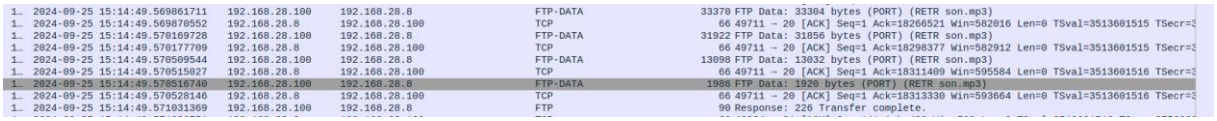


Figure 10*– Protocol TCP with MTU=1500*

With the TCP protocol, we can see that the protocol does not respect the MTU. In fact, TCP is an old protocol, so it does not respect the MTU, which was introduced later with IPv4. As a result, packet sizes are very large, up to 32,000 despite an MTU of 1,500.

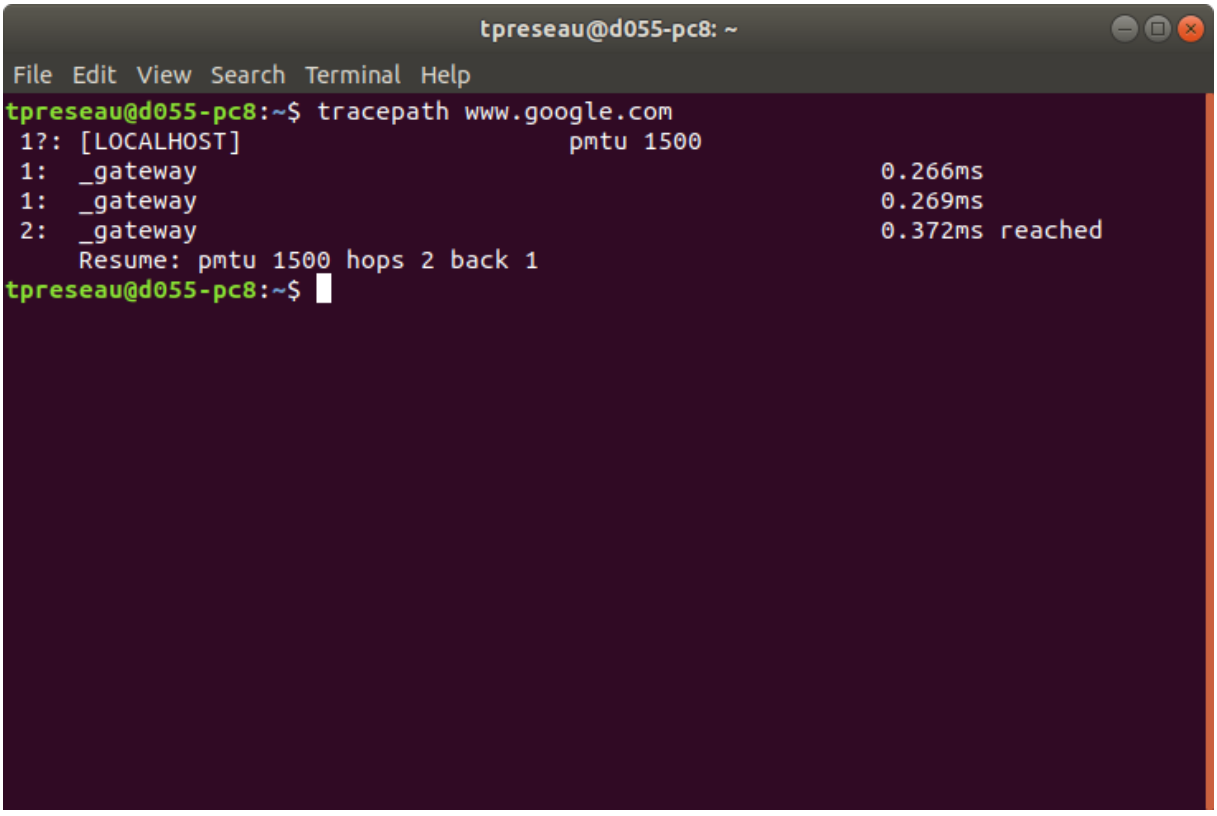


Figure 11*– TracePath*

All websites agreed that the best MTU is the 1500, it's the value by default.

III –TCP Window Size

The script is creating a server which is running and listening on 127.0.01 at port 9999. It is handling TCP request until we stop it.

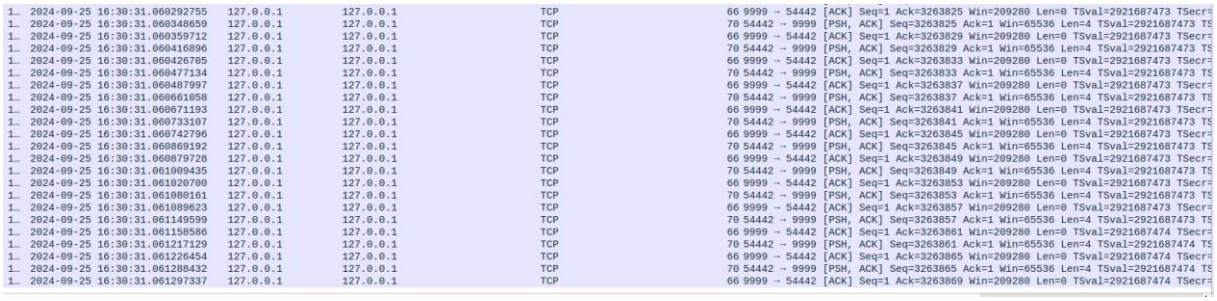


Figure 12 - Window Capture without sleep

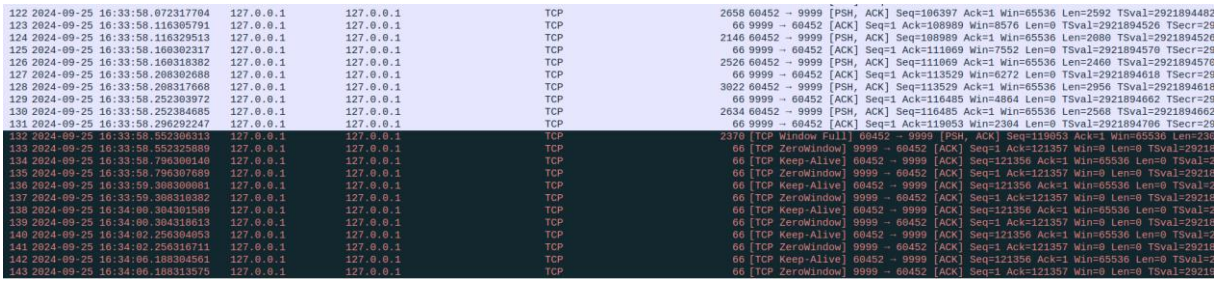


Figure 13 - Window capture with sleep

Without the line 10 command, the window will continue to fill without stopping. If, on the other hand, the line is uncommented, the window fills rapidly until it reaches saturation, i.e. it can no longer respond, and packets are put on hold. To obtain this capture, we had to observe in the Loopback.

IV – Capturing a web session with Telnet

```
tpreseau@d055-pc8:~$ telnet www.ensea.fr 80
Trying 10.10.17.5...
Connected to enseaweb.ensea.fr.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 301 Moved Permanently
Date: Wed, 25 Sep 2024 14:57:42 GMT
Server: Apache/2.4.41 (Ubuntu)
Location: https:///
Content-Length: 295
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https:///">here</a>.</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at www.ensea.fr Port 80</address>
</body></html>
Connection closed by foreign host.
tpreseau@d055-pc8:~$ telnet www.ensea.fr 80
Trying 10.10.17.5...
Connected to enseaweb.ensea.fr.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.ensea.fr

HTTP/1.1 301 Moved Permanently
Date: Wed, 25 Sep 2024 14:58:00 GMT
Server: Apache/2.4.41 (Ubuntu)
Location: https://www.ensea.fr/
Content-Length: 307
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://www.ensea.fr/">here</a>.</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at www.ensea.fr Port 80</address>
</body></html>
Connection closed by foreign host.
tpreseau@d055-pc8:~$
```

We use Telnet to connect to a site (here www.ensea.fr) and make an HTTP request using the following model GET / HTTP/1.1 then the following line: Host: www.ensea.fr.

The first '/' corresponds to retrieving the file or target from the root of the host. Note that the second line is not necessary if you are using HTTP version 1.0.

As a result, we will get a 301 error, which will return us to the host page, but in HTTPS, a more secure HTTP protocol.

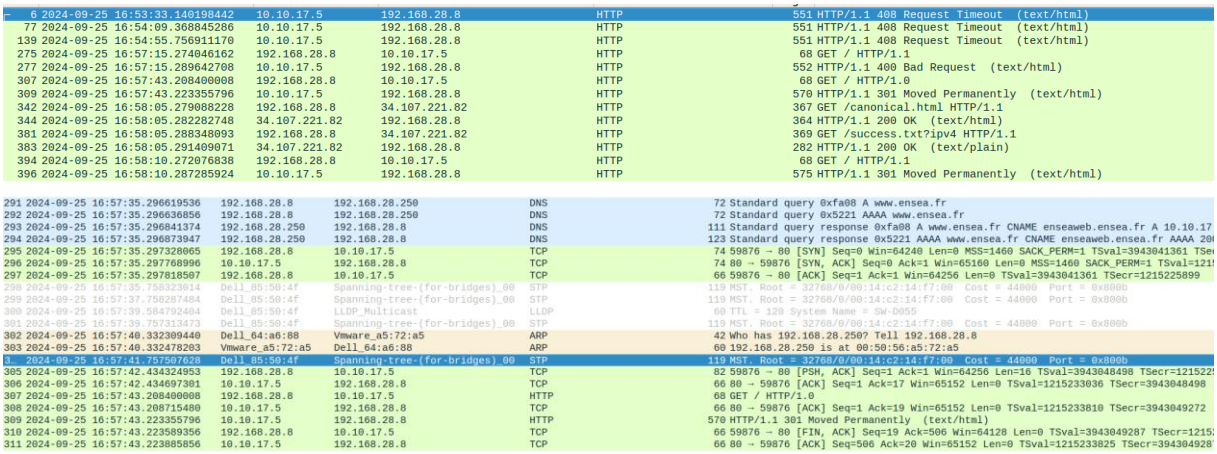


Figure 14 - Capture during HTTP

V – SSH Protocol

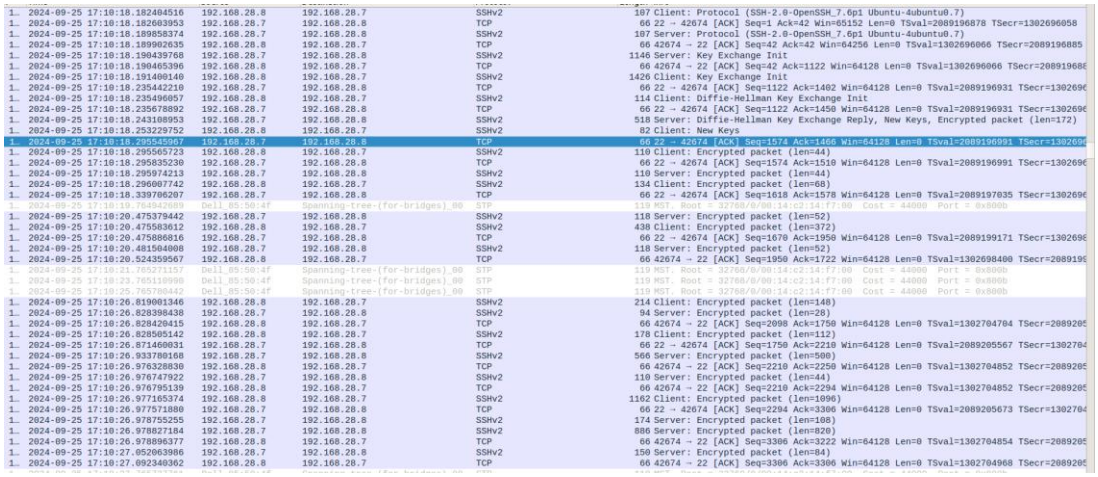


Figure 15 - SSH Capture

We can see an exchange of encrypted packets due to the wall command sending broadcast messages.