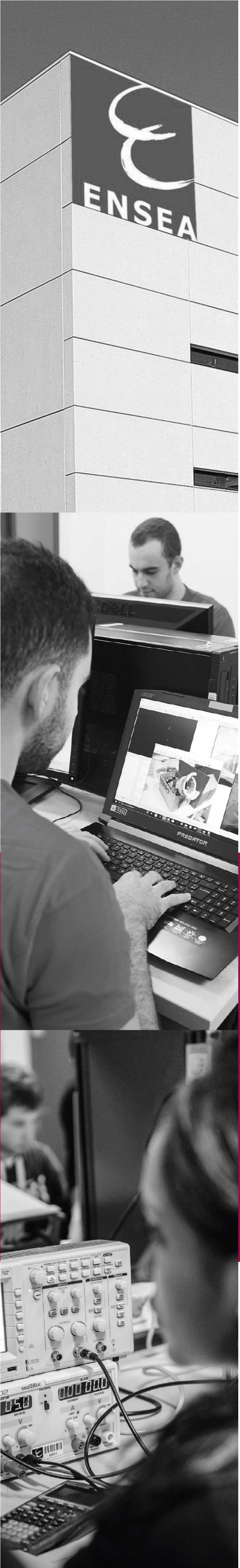


ENSEA

Beyond Engineering

TP2 ARCHITECTURE & PROTOCOLS

RTS TP1

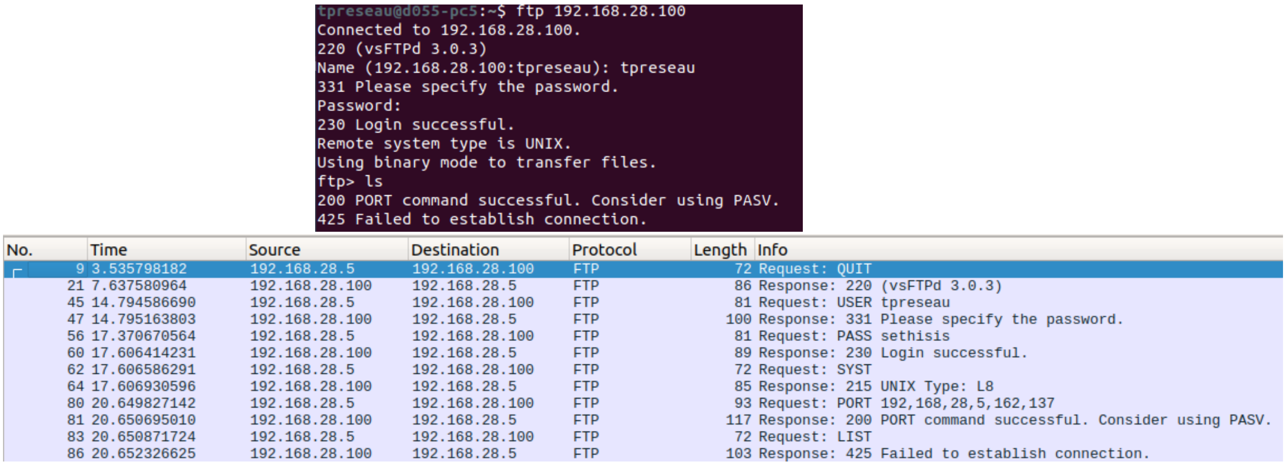
Use advanced network commands and study them with Wireshark.

PAN Jimmy-Antoine - LI Guillaume

PAN Jimmy-Antoine - LI Guillaume

11th October 2022

1. Capture FTP session



```
tpreseau@d055-pc5:~$ ftp 192.168.28.100
Connected to 192.168.28.100.
220 (vsFTPd 3.0.3)
Name (192.168.28.100:tpreseau): tpreseau
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
425 Failed to establish connection.
```

No.	Time	Source	Destination	Protocol	Length	Info
9	3.535798182	192.168.28.5	192.168.28.100	FTP	72	Request: QUIT
21	7.637580964	192.168.28.100	192.168.28.5	FTP	86	Response: 220 (vsFTPd 3.0.3)
45	14.794586690	192.168.28.5	192.168.28.100	FTP	81	Request: USER tpreseau
47	14.795163803	192.168.28.100	192.168.28.5	FTP	100	Response: 331 Please specify the password.
56	17.370670564	192.168.28.5	192.168.28.100	FTP	81	Request: PASS sethisis
60	17.606414231	192.168.28.100	192.168.28.5	FTP	89	Response: 230 Login successful.
62	17.606586291	192.168.28.5	192.168.28.100	FTP	72	Request: SYST
64	17.606930596	192.168.28.100	192.168.28.5	FTP	85	Response: 215 UNIX Type: L8
80	20.649827142	192.168.28.5	192.168.28.100	FTP	93	Request: PORT 192,168,28,5,162,137
81	20.650695010	192.168.28.100	192.168.28.5	FTP	117	Response: 200 PORT command successful. Consider using PASV.
83	20.650871724	192.168.28.5	192.168.28.100	FTP	72	Request: LIST
86	20.652326625	192.168.28.100	192.168.28.5	FTP	103	Response: 425 Failed to establish connection.

Figure 1 – Attempt of connection on the FTP session

We try to connect to the PC100 (192.168.28.100). As we can see, the connection is successful. However, we cannot make any command to manipulate the folders and the files inside : we have the error "Connection Failed". With the command "ls", "get" and "send", we were supposed to display all files, and move them on the server.

2. MTU (Maximum Transfer Unit)

- **Identify the default value assigned to the MTU parameter on your main interface**

To check the value of the MTU of our main interface we use the *-ifconfig* command which gives us the following picture :

```
enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.28.5  netmask 255.255.255.0  broadcast 192.168.28.255
        inet6 fe80::56bf:64ff:fe64:af8a  prefixlen 64  scopeid 0x20<link>
        ether 54:bf:64:64:af:8a  txqueuelen 1000  (Ethernet)
        RX packets 1927  bytes 309544 (309.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 748  bytes 206382 (206.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xef400000-ef420000
```

FIGURE 2 – Main interface : MTU = 1500

As we can see for enp0s31f6 (main interface) we have a size of 1500 for the MTU.

- **Explain the role of this parameter**

The MTU is the maximum size of the packet that can be sent in one time. We can note that the real maximum size of the packet sent is 1496 because packets are multiple of 8 bits ($1496 = 187 * 8$). If the packet's size exceeds the MTU size, the packet will be fragmented until all of them can be sent.

- **MTU and ICMP**

To change the MTU size to 100 we need to write the following line in the terminal :

ifconfig enp0s31f6 mtu 100 up

This give us the following plot :

```
enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 100
        inet 192.168.28.5  netmask 255.255.255.0  broadcast 192.168.28.255
        ether 54:bf:64:64:af:8a  txqueuelen 1000  (Ethernet)
        RX packets 12415  bytes 5979815 (5.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9695  bytes 1245968 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xef400000-ef420000
```

FIGURE 3 – Main interface : MTU = 100

Now, the maximum size of the packets is 96 ($= 12 * 8$).

No.	Time	Source	Destination	Protoco	Length	Info
1	0.000…	192.168.28.5	192.168.28.7	ICMP	92	Echo (ping) request id=0x1361, seq=39/9984, ttl=64 (reply in 2)
2	0.000…	192.168.28.7	192.168.28.5	ICMP	92	Echo (ping) reply id=0x1361, seq=39/9984, ttl=64 (request in 1)
3	1.023…	192.168.28.5	192.168.28.7	ICMP	92	Echo (ping) request id=0x1361, seq=40/10240, ttl=64 (reply in 4)
4	1.024…	192.168.28.7	192.168.28.5	ICMP	92	Echo (ping) reply id=0x1361, seq=40/10240, ttl=64 (request in 3)
5	2.047…	192.168.28.5	192.168.28.7	ICMP	92	Echo (ping) request id=0x1361, seq=41/10496, ttl=64 (reply in 6)
6	2.048…	192.168.28.7	192.168.28.5	ICMP	92	Echo (ping) reply id=0x1361, seq=41/10496, ttl=64 (request in 5)
7	3.072…	192.168.28.5	192.168.28.7	ICMP	92	Echo (ping) request id=0x1361, seq=42/10752, ttl=64 (reply in 8)
8	3.072…	192.168.28.7	192.168.28.5	ICMP	92	Echo (ping) reply id=0x1361, seq=42/10752, ttl=64 (request in 7)
9	4.095…	192.168.28.5	192.168.28.7	ICMP	92	Echo (ping) request id=0x1361, seq=43/11008, ttl=64 (reply in 10)
10	4.096…	192.168.28.7	192.168.28.5	ICMP	92	Echo (ping) reply id=0x1361, seq=43/11008, ttl=64 (request in 9)
11	5.120…	192.168.28.5	192.168.28.7	ICMP	92	Echo (ping) request id=0x1361, seq=44/11264, ttl=64 (reply in 12)
12	5.120…	192.168.28.7	192.168.28.5	ICMP	92	Echo (ping) reply id=0x1361, seq=44/11264, ttl=64 (request in 11)

No.	Time	Source	Destination	Protoco	Length	Info
248	86.40…	192.168.28.5	192.168.28.7	ICMP	42	Echo (ping) request id=0x17f7, seq=16/4096, ttl=64 (reply in 249)
249	86.40…	192.168.28.7	192.168.28.5	ICMP	122	Echo (ping) reply id=0x17f7, seq=16/4096, ttl=64 (request in 248)
250	87.42…	192.168.28.5	192.168.28.7	IPv4	114	Fragmented IP protocol (proto=ICMP 1, off=0, ID=dde9) [Reassembled in #251]

FIGURE 4 – Sending packets 50 and 80 bytes

We can observe that sending a packet of 50 bytes (respectively 80 bytes), only 92 bytes (122 bytes) are sent in reality. In fact, we must add the header which is composed of 42 bytes : 14 for the Ethernet Header, 20 for the IP Header and 8 for the ICMP Header (14+20+8=42). We saw that the MTU was 100, so for the first case, our packet of 50 bytes can pass without fragmentation ($92 < 100$). However, for the second case, our packets of 80 must be fragmented ($122 > 100$) and can't be send in one frame. The first packet will be 96 bytes ($12 * 8$) and the second one will be 26 bytes.

- **MTU and FTP**

As we cannot connect to the FTP server, we cannot answer this question...

- **Use the tracepath software to determine the best MTU value to access the Internet from the ENSEA network.**

```
tpreseau@d055-pc5:~$ tracepath google.com
 1?: [LOCALHOST]                      pmtu 1500
 1:  ucopia                                              0.333ms reached
 1:  ucopia                                              0.318ms reached
     Resume: pmtu 1500 hops 1 back 1
tpreseau@d055-pc5:~$ tracepath www.ensea.fr
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                            0.314ms
 1:  _gateway                                            0.388ms
 2:  _gateway                                            0.308ms reached
```

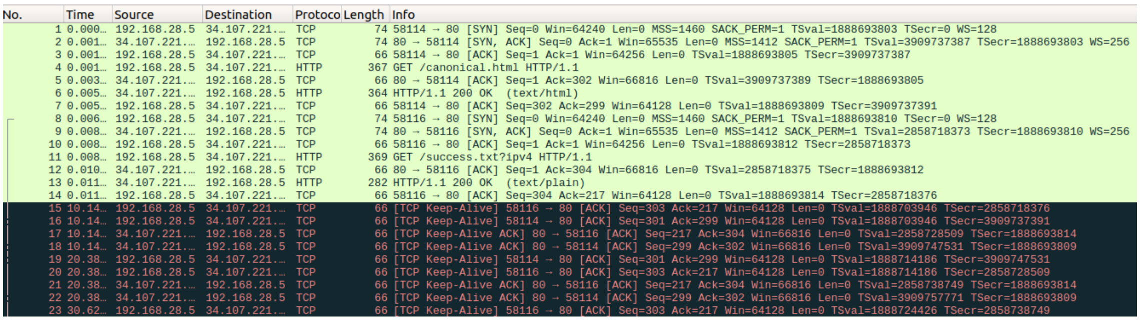
FIGURE 5 – Tracepath of Google and ENSEA website.

As we can observe, for both websites, the command tracepath in the terminal shows us that the optimum MTU is 1500, the default value.

3. TCP Window Size

- **What does this script do ?**

The script does the following commmands :
- *line 7* - Data of TCP requests are stored in the data variable as a file.
- *line 8* - Stop the programm if there are no data.
- *line 9* - Display the data of received requests in the shell.
- *line 10* - Delay of 1 second.
- *line 14* - Create a server on localhost :9999.
- *line 15* - Activate the serveur until receiving a shutdown command.



No.	Time	Source	Destination	Protoco	Length	Info
1	0.000…	192.168.28.5	34.107.221.…	TCP	74	58114 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1888693803 TSecr=0 WS=128
2	0.001…	34.107.221.…	192.168.28.5	TCP	74	80 → 58114 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 TSval=3909737387 TSecr=1888693803 WS=256
3	0.001…	192.168.28.5	34.107.221.…	TCP	66	58114 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1888693805 TSecr=3909737387
4	0.001…	192.168.28.5	34.107.221.…	HTTP	367	GET /canonical.html HTTP/1.1
5	0.003…	34.107.221.…	192.168.28.5	TCP	66	80 → 58114 [ACK] Seq=1 Ack=302 Win=66816 Len=0 TSval=3909737389 TSecr=1888693805
6	0.005…	34.107.221.…	192.168.28.5	HTTP	364	HTTP/1.1 200 OK (text/html)
7	0.005…	192.168.28.5	34.107.221.…	TCP	66	58114 → 80 [ACK] Seq=302 Ack=299 Win=64128 Len=0 TSval=1888693809 TSecr=3909737391
8	0.006…	192.168.28.5	34.107.221.…	TCP	74	58116 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1888693810 TSecr=0 WS=128
9	0.008…	34.107.221.…	192.168.28.5	TCP	74	80 → 58116 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 TSval=2858718373 TSecr=1888693810 WS=256
10	0.008…	192.168.28.5	34.107.221.…	TCP	66	58116 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1888693812 TSecr=2858718373
11	0.008…	192.168.28.5	34.107.221.…	HTTP	369	GET /success.txt?ipv4 HTTP/1.1
12	0.010…	34.107.221.…	192.168.28.5	TCP	66	80 → 58116 [ACK] Seq=1 Ack=304 Win=66816 Len=0 TSval=2858718375 TSecr=1888693812
13	0.011…	34.107.221.…	192.168.28.5	HTTP	282	HTTP/1.1 200 OK (text/plain)
14	0.011…	192.168.28.5	34.107.221.…	TCP	66	58116 → 80 [ACK] Seq=304 Ack=217 Win=64128 Len=0 TSval=1888693814 TSecr=2858718376
15	10.14…	192.168.28.5	34.107.221.…	TCP	66	[TCP Keep-Alive] 58116 → 80 [ACK] Seq=303 Ack=217 Win=64128 Len=0 TSval=1888703946 TSecr=2858718376
16	10.14…	192.168.28.5	34.107.221.…	TCP	66	[TCP Keep-Alive] 58114 → 80 [ACK] Seq=301 Ack=299 Win=64128 Len=0 TSval=1888703946 TSecr=3909737391
17	10.14…	34.107.221.…	192.168.28.5	TCP	66	[TCP Keep-Alive ACK] 80 → 58116 [ACK] Seq=217 Ack=304 Win=66816 Len=0 TSval=2858728509 TSecr=1888693814
18	10.14…	34.107.221.…	192.168.28.5	TCP	66	[TCP Keep-Alive ACK] 80 → 58114 [ACK] Seq=299 Ack=302 Win=66816 Len=0 TSval=3909747531 TSecr=1888693809
19	20.38…	192.168.28.5	34.107.221.…	TCP	66	[TCP Keep-Alive] 58114 → 80 [ACK] Seq=301 Ack=299 Win=64128 Len=0 TSval=1888714186 TSecr=3909747531
20	20.38…	192.168.28.5	34.107.221.…	TCP	66	[TCP Keep-Alive] 58116 → 80 [ACK] Seq=303 Ack=217 Win=64128 Len=0 TSval=1888714186 TSecr=2858728509
21	20.38…	34.107.221.…	192.168.28.5	TCP	66	[TCP Keep-Alive ACK] 80 → 58116 [ACK] Seq=217 Ack=304 Win=66816 Len=0 TSval=2858738749 TSecr=1888693814
22	20.38…	34.107.221.…	192.168.28.5	TCP	66	[TCP Keep-Alive ACK] 80 → 58114 [ACK] Seq=299 Ack=302 Win=66816 Len=0 TSval=3909757771 TSecr=1888693809
23	30.62…	192.168.28.5	34.107.221.…	TCP	66	[TCP Keep-Alive] 58116 → 80 [ACK] Seq=303 Ack=217 Win=64128 Len=0 TSval=1888724426 TSecr=2858738749

FIGURE 6 – Capture Wireshark while running the 3 terminals

After running the script we can find a new script with the word "foo" written.

- **What can you say about the Window fields ?**

4. Capturing a Web session with Telnet

Here we are using Telnet which is like a remote terminal. Thanks to this we can get the header and other information of the website we are inspecting. So we use the command : *telnet facebook.com 80*. We add the 80 for the HTTP port number.

Then we need to type the command *GET \ index.html* to get the website source code. Here is the source code of Facebook main page :

```
tpreseau@d055-pc5:~$ telnet www.facebook.com 80
Trying 2a03:2880:f130:83:face:b00c:0:25de...
Connected to star-mini.c10r.facebook.com.
Escape character is '^]'.
GET \index.html
HTTP/1.1 400 Bad Request
Content-Type: text/html; charset=utf-8
Date: Tue, 11 Oct 2022 15:08:12 GMT
Connection: close
Content-Length: 2959

<!DOCTYPE html>
<html lang="en" id="facebook">
  <head>
    <title>Facebook | Error</title>
    <meta charset="utf-8">
```

FIGURE 7 – Capture a Facebook session with telnet

As we can see in the picture there is the beginning of the source code of Facebook main page.

5. SSH Protocol

Now, we are going to execute a command from our computer 192.168.28.5) on an other one in the same local network (192.168.28.1). To make a connexion on this computer, we execute the command : ssh IPAddress.

Let's try to create a file on the desktop of our target. With the command mkdir, we can create a file on the computer whose IP adress is 192.168.28.1. We choose the path (cd/ls/etc.) to reach the desktop. We execute the command mkdir to add the file on the Desktop with ssh protocol as we can see on the follwoing picture :

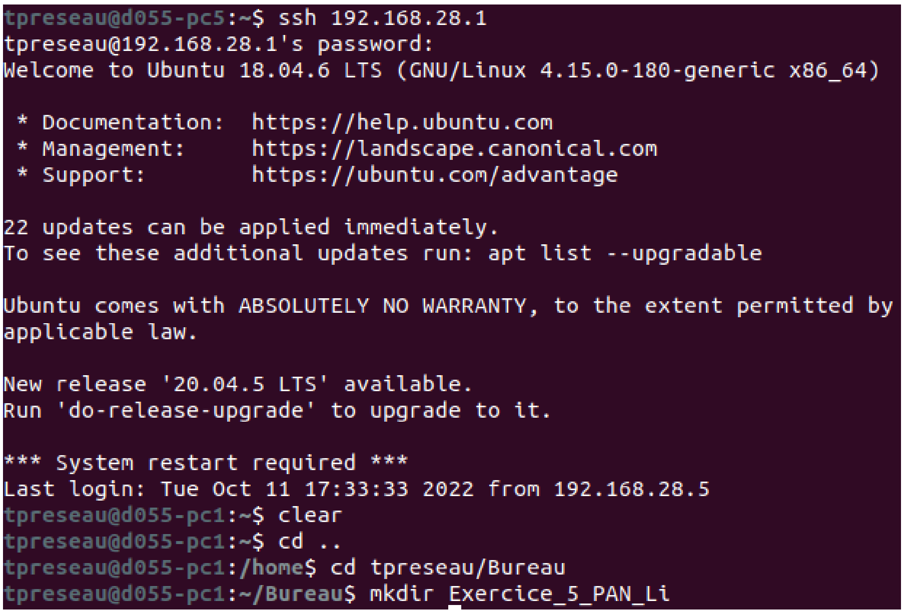


FIGURE 8 – Capture of the terminal creating a file "Exercice_5_PAN_LI"

We can observe the resultat directly on the computer of our classmate :

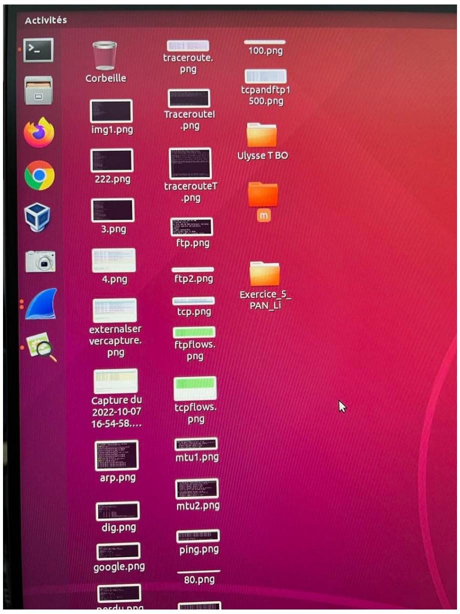


FIGURE 9 – Capture a Facebook session with telnet

The files has been created on the desktop with a SSH Protocol.