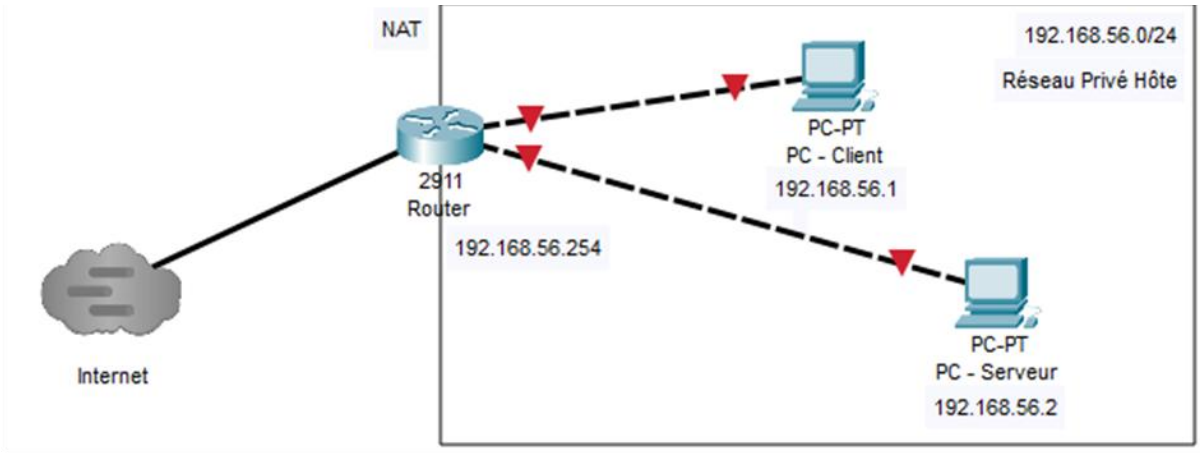
TP
Transport Control Protocol

CONSUEGRA Yanis L3 ESSIR

Table des matières

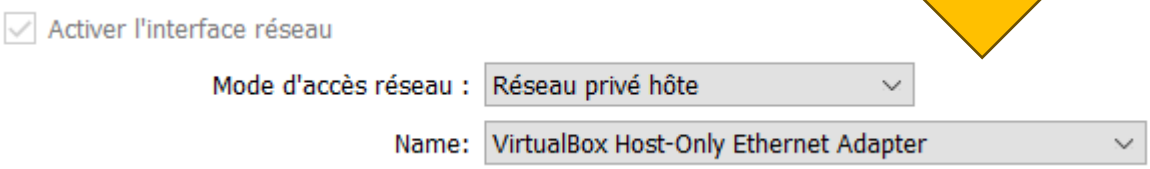
Schéma de l'infrastructure



Environnement et contexte.

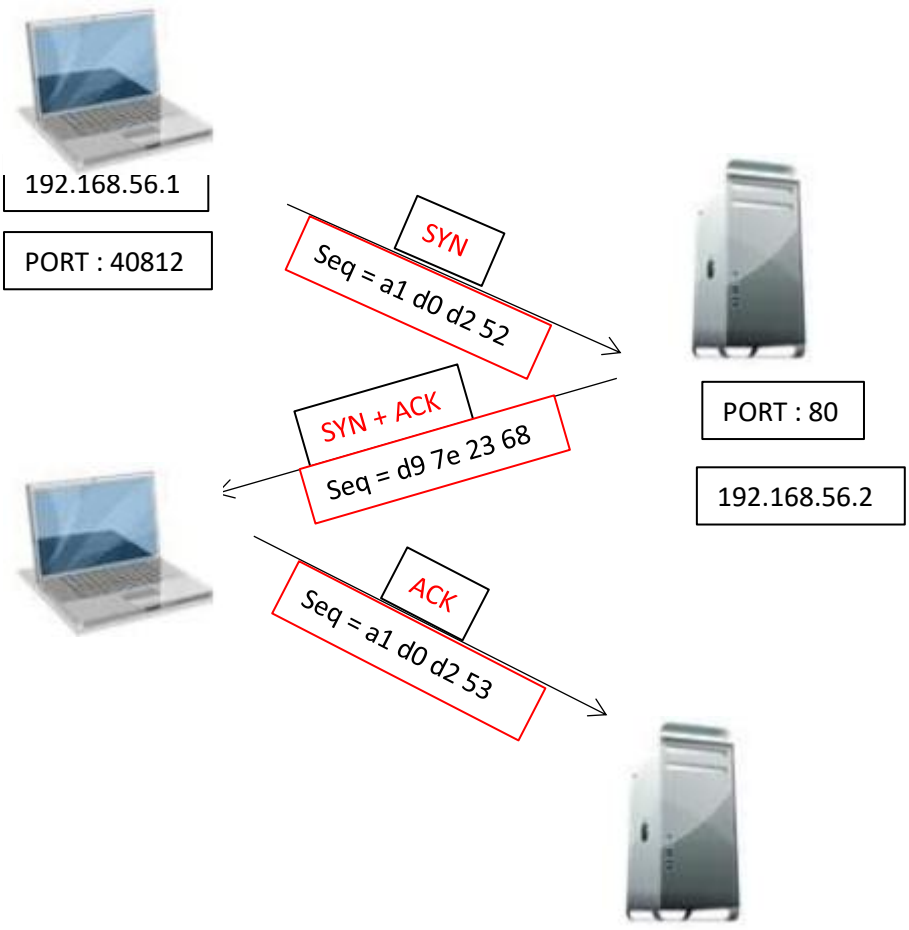
- Le TP a été réalisé en cours puis retravailler à partir d'un environnement virtuel sur Virtual Box avec 2 machines linux ; une machine cliente et une machine serveur, et un routeur virtuel pour adresse de passerelle 192.168.56.254. l'adresse réseau est donc 192.168.56.0/24.

On change bien l'accès réseau en réseau privé hôte afin que nos différentes machines communiques entres-elles sur le même réseau.



Exercice 1 : Analyse de trame

a. Segment ouverture de connexion



Numéros de séquence et d'acquittement, et flag

3	9.205013526	192.168.56.2	192.168.56.1	TCP	76	40812 → 8 [SYN] Seq=0 Win=64240 L
4	9.205318590	192.168.56.1	192.168.56.2	TCP	76	80 → 4081 [SYN, ACK] Seq=0 Ack=1
5	9.205355358	192.168.56.2	192.168.56.1	TCP	68	40812 → 8 [ACK] Seq=1 Ack=1 Win=6

➔ Pour le premier contact, le SYN flag n'est JAMAIS à 0, il est à 1 et le ACK flag est 0

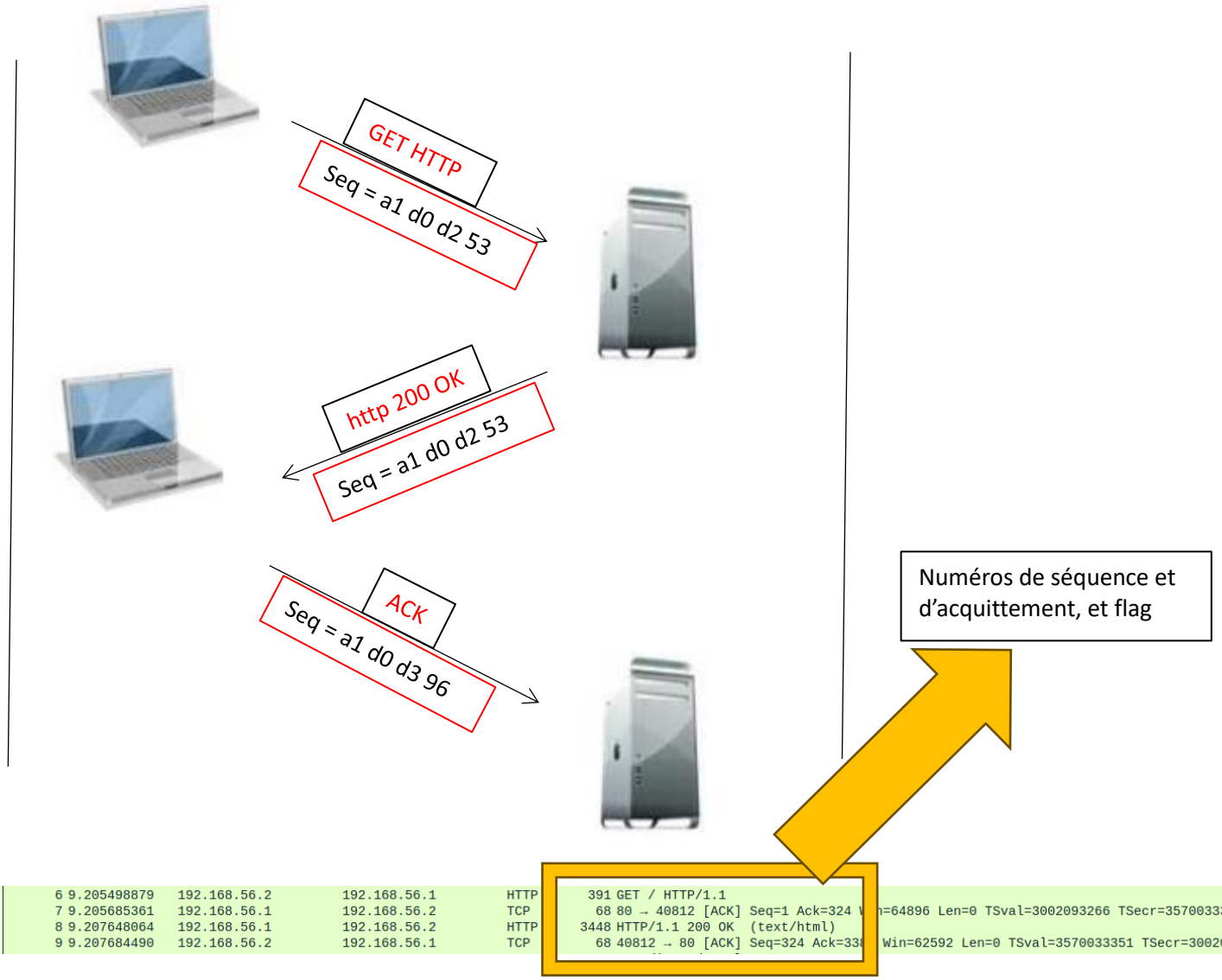
```
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 2714817106
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1010 .... = Header Length: 40 bytes (10)
  ▸ Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0xf182 [unverified]
0000  00 04 00 01 00 06 08 00  27 c3 49 a1 00 00 08 00   ........ '.I.....
0010  45 00 00 3c 59 dd 40 00  40 06 ef 8a c0 a8 38 02   E..<Y.@. @.....8.
0020  c0 a8 38 01 9f 6c 00 50  a1 d0 d2 52 00 00 00 00   ..8..l.P ...R....
0030  a0 02 fa f0 f1 82 00 00  02 04 05 b4 04 02 08 0a   ........ ........
0040  d4 ca 62 c4 00 00 00 00  01 03 03 07               ..b..... ....
```

b. Les options TCP

- **Le maximum size :** Indique la taille des paquets que peut recevoir le destinataire et que peut envoyer l'émetteur.
- **TCP SACK :** Prise en charge de la fonctionnalité « SACK » (Selective Acknowledgment)
- **TimeStamps :** Permet d'échanger des informations de temps dans les entêtes TCP.
- **No-Operation :** Cela fait référence à une option pour remplir de l'espace dans l'entête TCP.
- **Windows-Scale :** Etendre la taille de la fenêtre de réception.
- **Checksum :** Permet de vérifier l'intégrité des données.

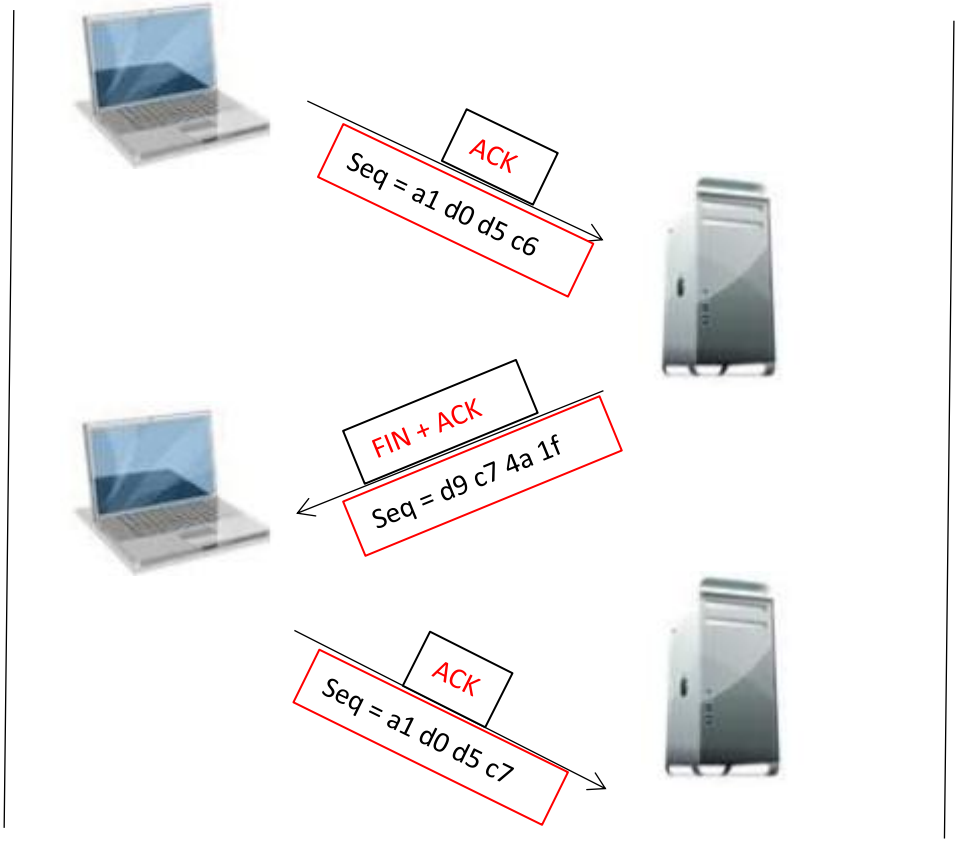
```
▾ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  ▸ TCP Option - Maximum segment size: 1460 bytes
  ▸ TCP Option - SACK permitted
  ▸ TCP Option - Timestamps: TSval 3570033348, TSecr 0
  ▸ TCP Option - No-Operation (NOP)
  ▸ TCP Option - Window scale: 7 (multiply by 128)
```

c. Acquittement des segments



- Le PC1 souhaite récupérer les données du serveur web en faisant une requête HTTP GET. Le serveur acquitte sa demande en émettant ACK et lui renvoie les données avec pour réponse HTTP 200. Le PC1 confirme ensuite réception des données du serveur web en émettant un segment TCP ACK.

d. Fermeture de la connexion



- Le serveur va émettre une demande de fermeture des connexions sur le poste client. Le paquet va contenir deux flags : FIN et ACK. Ensuite on aura une réponse du client qui accepte que la connexion entre les deux hôtes soit bien interrompue (il faut que les 2 hôtes soient d'accord afin qu'une connexion TCP soit fermée). Enfin, on peut voir la confirmation du serveur que la connexion TCP doit être fermée, on peut noter que le numéro de séquence à augmenté de 1 car il suit l'acquittement du client. Quant au numéro d'acquittement, il a lui aussi augmenté de 1 car le segment TCP envoyé par le poste client contient le flag FIN.

```
68 40812 → 80 [ACK] Seq=884 Ack=9911 Win=6
68 40812 → 80 [FIN, ACK] Seq=884 Ack=9911
68 80 → 40812 [FIN, ACK] Seq=9911 Ack=885
68 40812 → 80 [ACK] Seq=885 Ack=9912 Win=6
```

e. Connexion TCP : commande netstat (ou ss)

- Afin de repérer cette connexion parmi celle du system on peut utiliser les commande :
  - Netstat | grep tcp
  - Ss -o state ESTABLISHED | grep tcp

- On obtient ceci sur le terminal après avoir tester avec le serveur web apache2 (cette entrée reste environ 2 secondes après la fermeture de connexion) :

```
┌──(root㉿kali-Y)-[~]
└─# ss -o state ESTABLISHED | grep tcp
tcp   0      0                    127.0.0.1:33322         127.0.0.1:http  timer:(keepalive,7.144m
s,0)
```

Exercice 2 : identifier les problèmes de connexions

Déroulement des scénarios

Scénario 1

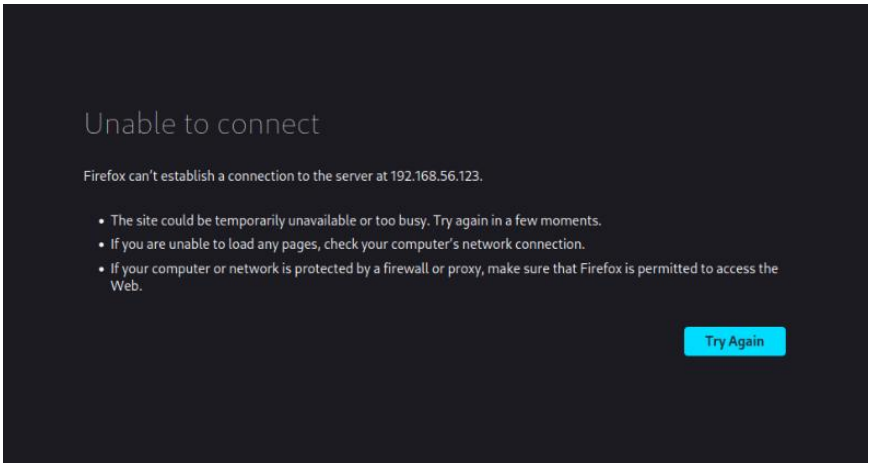
Sur le même sous-réseau :

- Sous le même sous-réseau si on tente de joindre un serveur web inexistant (ici 123.45.67.89), le client va émettre des requêtes ARP en diffusion afin de connaître l'adresse MAC, néanmoins l'opération va se répéter car le serveur web n'existe pas.

arp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_de:48:2e		ARP	62	Who has 192.168.56.1? Tell 192.168.56.254
2	0.000013693	PcsCompu_62:da:f9		ARP	44	192.168.56.1 is at 08:00:27:62:da:f9
7	8.787273406	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
8	9.808567099	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
9	10.832648341	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
11	11.856771691	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
13	12.880271442	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
14	13.904333108	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
18	14.928671390	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
22	15.952638652	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1
25	16.976467225	PcsCompu_62:da:f9		ARP	44	Who has 192.168.56.123? Tell 192.168.56.1

→ Voici le message d'erreur qui s'affiche sur le moteur de recherche :



Le serveur est incapable de se connecter à l'adresse indiquée

- Sur un sous réseau différents, le client émet de nombreuses requêtes SYN à différent intervalles sans jamais établir de connexion et sans obtenir de réponse :

15	6.198000762	123.45.67.89	192.168.56.1	TCP	62 80 → 46168 [RST, ACK] Seq=1 Ack=2 Win=65535 Len=0
16	10.101308965	192.168.56.1	123.45.67.89	TCP	56 [TCP Keep-Alive] 46152 → 80 [ACK] Seq=323 Ack=1 Win=64240 Len=0
17	10.101633639	123.45.67.89	192.168.56.1	TCP	62 [TCP Keep-Alive ACK] 80 → 46152 [ACK] Seq=1 Ack=324 Win=65535 Len=0
18	20.341574913	192.168.56.1	123.45.67.89	TCP	56 [TCP Keep-Alive] 46152 → 80 [ACK] Seq=323 Ack=1 Win=64240 Len=0
19	20.342098256	123.45.67.89	192.168.56.1	TCP	62 [TCP Keep-Alive ACK] 80 → 46152 [ACK] Seq=1 Ack=324 Win=65535 Len=0
20	30.581303250	192.168.56.1	123.45.67.89	TCP	56 [TCP Keep-Alive] 46152 → 80 [ACK] Seq=323 Ack=1 Win=64240 Len=0
21	30.581621305	123.45.67.89	192.168.56.1	TCP	62 [TCP Keep-Alive ACK] 80 → 46152 [ACK] Seq=1 Ack=324 Win=65535 Len=0
22	40.821302512	192.168.56.1	123.45.67.89	TCP	56 [TCP Keep-Alive] 46152 → 80 [ACK] Seq=323 Ack=1 Win=64240 Len=0
23	40.821813336	123.45.67.89	192.168.56.1	TCP	62 [TCP Keep-Alive ACK] 80 → 46152 [ACK] Seq=1 Ack=324 Win=65535 Len=0
26	51.061421750	192.168.56.1	123.45.67.89	TCP	56 [TCP Keep-Alive] 46152 → 80 [ACK] Seq=323 Ack=1 Win=64240 Len=0
27	51.061708490	123.45.67.89	192.168.56.1	TCP	62 [TCP Keep-Alive ACK] 80 → 46152 [ACK] Seq=1 Ack=324 Win=65535 Len=0
30	61.301309667	192.168.56.1	123.45.67.89	TCP	56 [TCP Keep-Alive] 46152 → 80 [ACK] Seq=323 Ack=1 Win=64240 Len=0
31	61.301710271	123.45.67.89	192.168.56.1	TCP	62 [TCP Keep-Alive ACK] 80 → 46152 [ACK] Seq=1 Ack=324 Win=65535 Len=0

Scénario 2

Fermer le port 80 et 443

- Afin de fermer les ports 80 et 443 sur notre serveur pour rendre inaccessible quiconque qui tente de se connecter à notre serveur web apache2 il faut utiliser la commande « ufw »

- Il faut d'abord mettre à jour la liste des dépôts :

```
└─# apt update
```

- Ensuite on va installer le paquet « ufw », c'est une interface de commande pour netfilter

```
-# apt install ufw
```

- On peut ensuite activer le firewall avec la commande « ufw enable » et ainsi bloquer les ports 80 et 443, voici ce qu'on est censé obtenir lorsque que l'on tape la commande « ufw status »

```
└─# ufw status
Status: active

To                         Action      From
--                         ------      ----
80                         DENY        Anywhere
443                        DENY        Anywhere
80 (v6)                    DENY        Anywhere (v6)
443 (v6)                   DENY        Anywhere (v6)
```

- Si on tente maintenant de se connecter au service apache2 mais que le port est fermé, Wireshark nous renvoi une trame avec un flag TCP RST. Il signifie qu'il souhaite généralement interrompre une connexion à la suite d'un problème de communication ou une réponse anormale.

4	6.719055167	192.168.56.2	192.168.56.1	TCP	76 49528 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=50799963 TSecr=0 WS=128
5	6.719066938	192.168.56.1	192.168.56.2	TCP	56 80 → 49528 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
6	6.733923631	192.168.56.2	192.168.56.1	TCP	76 44876 → 443 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=50799978 TSecr=0 WS=128
7	6.734025179	192.168.56.1	192.168.56.2	TCP	56 443 → 44876 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Exercice 3 : Nmap et SS

Qu’est-ce que Nmap

- Nmap est un puissant outil de scan réseau open-source. Il est utilisé pour découvrir et auditer des réseaux, en analysant les hôtes et les services actifs sur un réseau.

1. Scan de port avec NMAP :

- On peut voir après la commande « nmap -p 0-1024 localhost » que seul les ports SSH (22) et HTTP (80) sont ouverts.

2. Scan de port sur Wireshark :

- On peut voir que Wireshark va tester 1 par 1 tous les ports que l’on a entré dans notre plage de port sur la commande. On peut également voir que lorsqu’un port et fermé, un « [RST-ACK] sera renvoyé alors que quand un port est ouvert (comme le port 80) on peut voir que Wireshark nous retourne un [SYN-ACK].

5	0.000035922	127.0.0.1	127.0.0.1	TCP	58 43172 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6	0.000039880	127.0.0.1	127.0.0.1	TCP	54 25 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.000046871	127.0.0.1	127.0.0.1	TCP	58 43172 → 110 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8	0.000063624	127.0.0.1	127.0.0.1	TCP	54 110 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	0.000070633	127.0.0.1	127.0.0.1	TCP	58 43172 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10	0.000084416	127.0.0.1	127.0.0.1	TCP	58 80 → 43172 [SYN, ACK] Seq=0 Ack=1 Win=65495 Len=0 M
11	0.000087683	127.0.0.1	127.0.0.1	TCP	54 43172 → 80 [RST] Seq=1 Win=0 Len=0
12	0.000095940	127.0.0.1	127.0.0.1	TCP	58 43172 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
13	0.000102669	127.0.0.1	127.0.0.1	TCP	58 22 → 43172 [SYN, ACK] Seq=0 Ack=1 Win=65495 Len=0 M
14	0.000105531	127.0.0.1	127.0.0.1	TCP	54 43172 → 22 [RST] Seq=1 Win=0 Len=0
15	0.000112718	127.0.0.1	127.0.0.1	TCP	58 43172 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
16	0.000116562	127.0.0.1	127.0.0.1	TCP	54 23 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	0.000123278	127.0.0.1	127.0.0.1	TCP	58 43172 → 53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
18	0.000127094	127.0.0.1	127.0.0.1	TCP	54 53 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	0.000133803	127.0.0.1	127.0.0.1	TCP	58 43172 → 445 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
20	0.000137524	127.0.0.1	127.0.0.1	TCP	54 445 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21	0.000144162	127.0.0.1	127.0.0.1	TCP	58 43172 → 256 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
22	0.000147871	127.0.0.1	127.0.0.1	TCP	54 256 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	0.000221608	127.0.0.1	127.0.0.1	TCP	58 43172 → 113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
24	0.000225878	127.0.0.1	127.0.0.1	TCP	54 113 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	0.000234295	127.0.0.1	127.0.0.1	TCP	58 43172 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
26	0.000238059	127.0.0.1	127.0.0.1	TCP	54 111 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27	0.000244466	127.0.0.1	127.0.0.1	TCP	58 43172 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
28	0.000248167	127.0.0.1	127.0.0.1	TCP	54 135 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29	0.000255402	127.0.0.1	127.0.0.1	TCP	58 43172 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
30	0.000259082	127.0.0.1	127.0.0.1	TCP	54 143 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31	0.000265483	127.0.0.1	127.0.0.1	TCP	58 43172 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
32	0.000269158	127.0.0.1	127.0.0.1	TCP	54 139 → 43172 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

- En linux, la commande « ss » permet d'afficher les connexions TCP, les numéros de ports ainsi que les adresses associés, on peut voir après avoir exécuté la page web du serveur web apache 2, qu'une requête sur le port 80 a été faite :