

TP Wireshark

TP U6 SISR

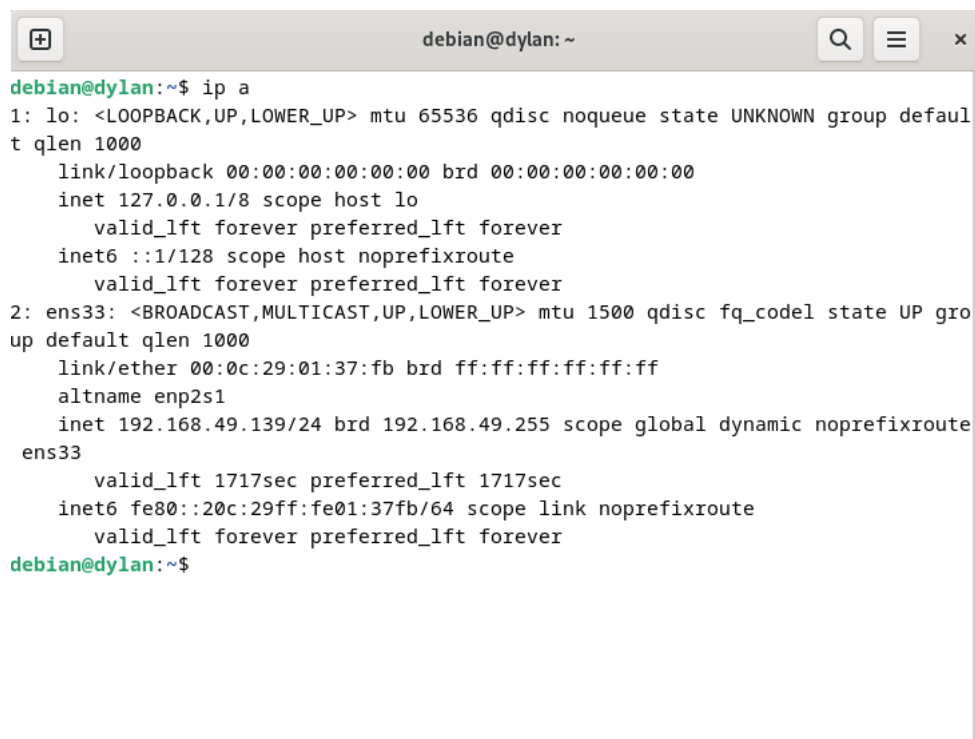
SPINELLI Dylan
BTS SIO SISR | PARIS YNOV CAMPUS

Table des matières

1-	Capture et analyse du trafic dans Wireshark	2
2-	Analyse de la connexion TCP	4
3-	Analyse du trafic HTTP	8
4-	Analyse de la fermeture TCP	13
5-	Cas des drapeaux RST	14
6-	Analyse des retransmissions TCP	15
7-	Analyse du trafic SSH	17

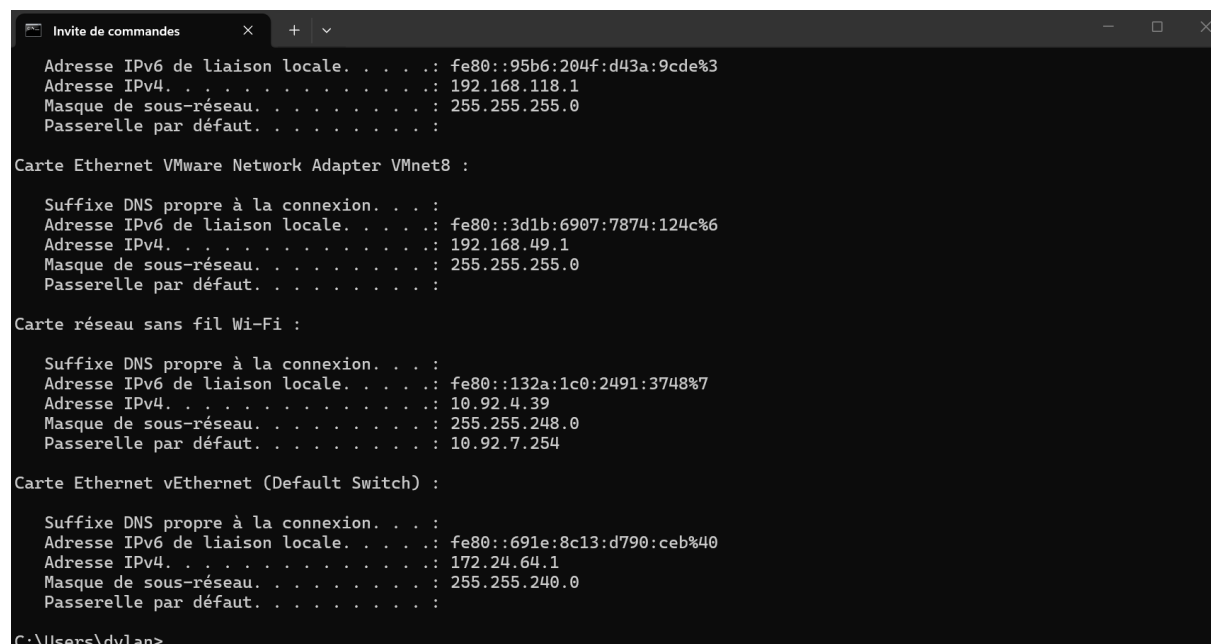
1-Capture et analyse du trafic dans Wireshark

Voici l'adresse IP de notre machine virtuelle :



```
debian@dylan: ~$ ip a
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
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:01:37:fb brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.49.139/24 brd 192.168.49.255 scope global dynamic noprefixroute ens33
        valid_lft 1717sec preferred_lft 1717sec
    inet6 fe80::20c:29ff:fe01:37fb/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
debian@dylan: ~$
```

Voici l'interface réseau de la machine physique :



```
Invite de commandes
Adresse IPv6 de liaison locale. . . . . : fe80::95b6:204f:d43a:9cde%3
Adresse IPv4. . . . . : 192.168.118.1
Masque de sous-réseau. . . . . : 255.255.255.0
Passerelle par défaut. . . . . :

Carte Ethernet VMware Network Adapter VMnet8 :

    Suffixe DNS propre à la connexion. . . . :
    Adresse IPv6 de liaison locale. . . . . : fe80::3d1b:6907:7874:124c%6
    Adresse IPv4. . . . . : 192.168.49.1
    Masque de sous-réseau. . . . . : 255.255.255.0
    Passerelle par défaut. . . . . :

Carte réseau sans fil Wi-Fi :

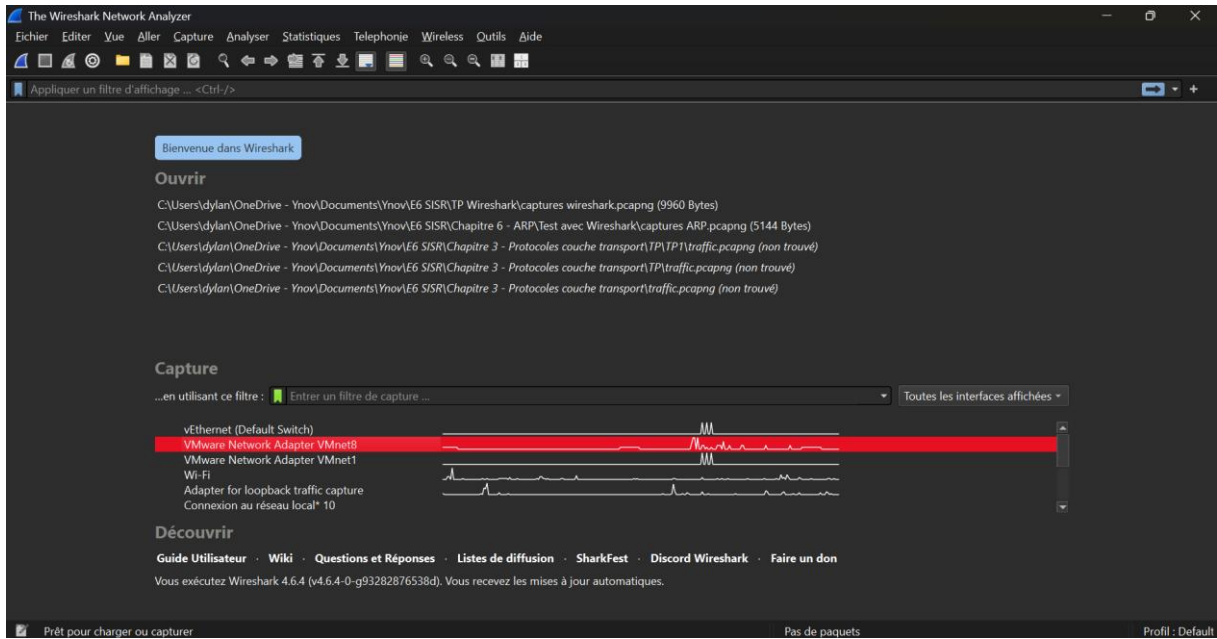
    Suffixe DNS propre à la connexion. . . . :
    Adresse IPv6 de liaison locale. . . . . : fe80::132a:1c0:2491:3748%7
    Adresse IPv4. . . . . : 10.92.4.39
    Masque de sous-réseau. . . . . : 255.255.248.0
    Passerelle par défaut. . . . . : 10.92.7.254

Carte Ethernet vEthernet (Default Switch) :

    Suffixe DNS propre à la connexion. . . . :
    Adresse IPv6 de liaison locale. . . . . : fe80::691e:8c13:d790:ceb%40
    Adresse IPv4. . . . . : 172.24.64.1
    Masque de sous-réseau. . . . . : 255.255.240.0
    Passerelle par défaut. . . . . :

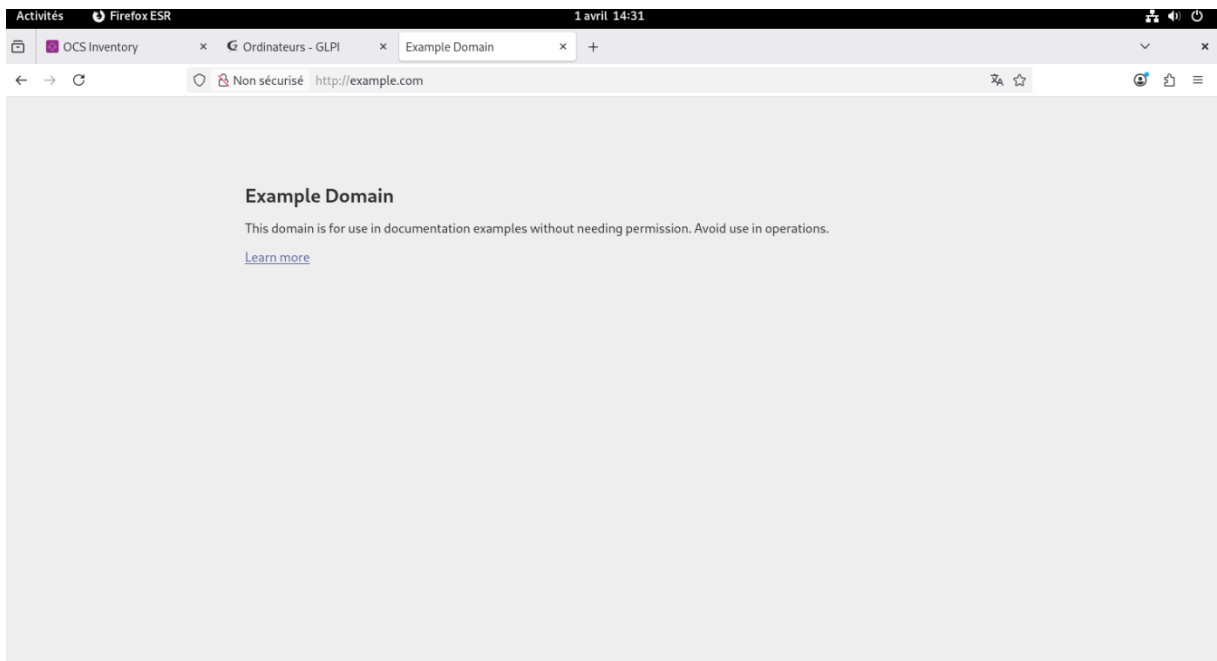
C:\Users\dylan>
```

On sélectionne l'interface VMnet8 dans Wireshark qui a pour adresse IP 192.168.49.1 :



On démarre ensuite la capture du trafic dans Wireshark

On ouvre la page <http://example.com> afin d'analyser le trafic :



On ferme ensuite le navigateur web afin de pouvoir analyser le trafic lors de la fermeture de la connexion.

2-Analyse de la connexion TCP

Voici les principaux champs à connaître pour le TCP :

Champ	Description
Source Port	Port de l'émetteur
Destination Port	Port du destinataire
Sequence Number	Numéro d'ordre du premier octet du segment
Acknowledgment Number	Prochain octet attendu par le récepteur
Flags (SYN, ACK, FIN, RST...)	Drapeaux de contrôle de la connexion
Window Size	Taille du tampon de réception (contrôle de flux)
Checksum	Vérification de l'intégrité du paquet
TCP Segment Length	Taille des données utiles du segment

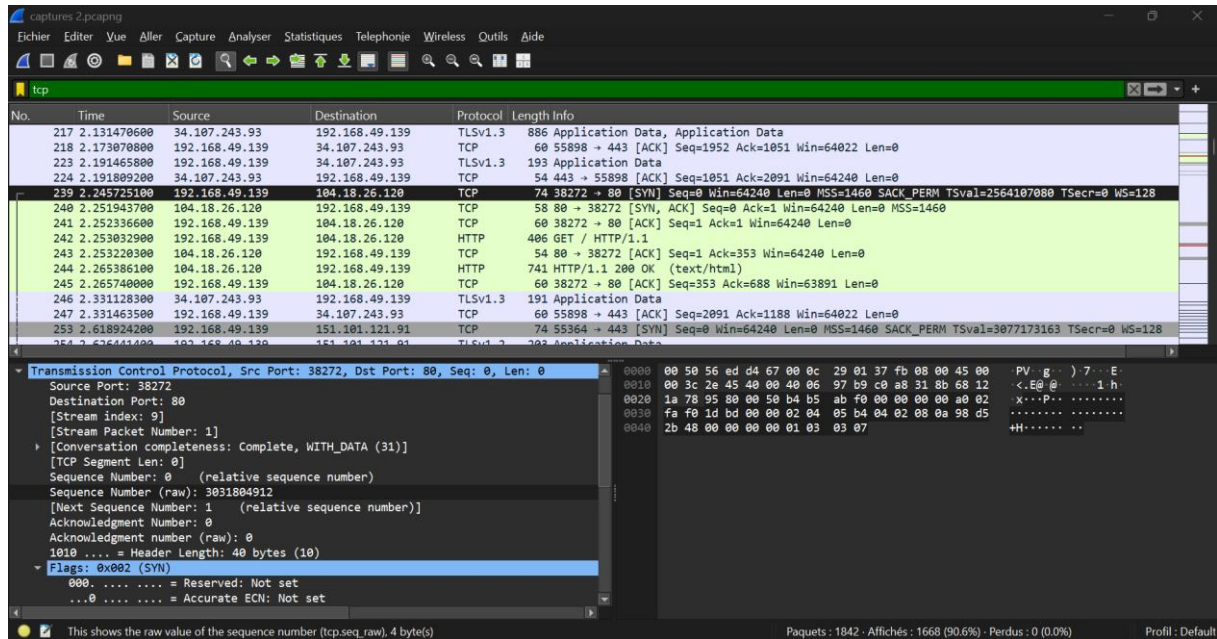
L'établissement d'une connexion TCP se fait en 3 étapes :

- Étape 1 — SYN : Le client envoie un paquet avec le drapeau SYN activé. Il annonce son numéro de séquence initial (ISN).
- Étape 2 — SYN-ACK : Le serveur répond avec SYN + ACK. Il accuse réception du SYN client et annonce son propre ISN.
- Étape 3 — ACK : Le client accuse réception du SYN serveur. La connexion est Etablie

La fermeture propre d'une connexion TCP utilise 4 étapes (Four-Way Teardown) :

- L'hôte qui ferme envoie FIN + ACK
- L'autre hôte accuse réception avec ACK
- L'autre hôte envoie à son tour FIN + ACK
- Le premier hôte répond ACK. La connexion est fermée.

On regarde le premier segment TCP « SYN » :



Le numéro de séquence du premier segment TCP « SYN » (Seq) = 3031804912

Le numéro relatif est = 0

Champ	Valeur observée	Signification
Source Port	38272	Port de l'émetteur
Destination Port	80	Port du destinataire
Sequence Number (raw)	3031804912	Numéro d'ordre du premier octet du segment
Header Length	40 bytes	Taille du header
Window Size	64240	Taille du tampon de réception (contrôle de flux)
Checksum	0x1dbd	Vérification de l'intégrité du paquet
TCP Segment Length	0	Taille des données utiles du segment

Analysons le segment suivant « SYN, ACK » :

The screenshot shows a Wireshark capture of a network packet. The packet list pane displays a table of captured packets. The selected packet is a TCP segment with the following details:

No.	Time	Source	Destination	Protocol	Length	Info
239	2.245725100	192.168.49.139	104.18.26.120	TCP	74	38272 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2564107080 TSecr=0 WS=128

The packet details pane shows the following information for the selected packet:

- Transmission Control Protocol, Src Port: 80, Dst Port: 38272, Seq: 0, Ack: 1, Len: 0
- Source Port: 80
- Destination Port: 38272
- [Stream Index: 9]
- [Stream Packet Number: 2]
- [Conversation completeness: Complete, WITH_DATA (31)]
- [TCP Segment Len: 0]
- Sequence Number: 0 (relative sequence number)
- Sequence Number (raw): 1588935850
- [Next Sequence Number: 1 (relative sequence number)]
- Acknowledgment Number: 1 (relative ack number)
- Acknowledgment number (raw): 3031804913
- 0110 ... = Header Length: 24 bytes (6)
- Flags: 0x012 (SYN, ACK)
- Window: 64240
- [Calculated window size: 64240]

La valeur du champ acknowledgment number = 3031804913

La valeur est égale a celle du du segment « SYN » + 1. Cela signifie que le serveur distant à bien reçu le segment « SYN » que nous avons envoyé précédemment.

Le numéro de séquence initial (Seq) en valeur brute = 1588935850

Wireshark affiche 0 en valeur relative car maintenant que la connexion TCP est établie avec le 3 way handshake, le prochain segment contiendra les données que l'on souhaite envoyer ou recevoir.

On trouve ensuite à la fin le segment TCP « ACK » :

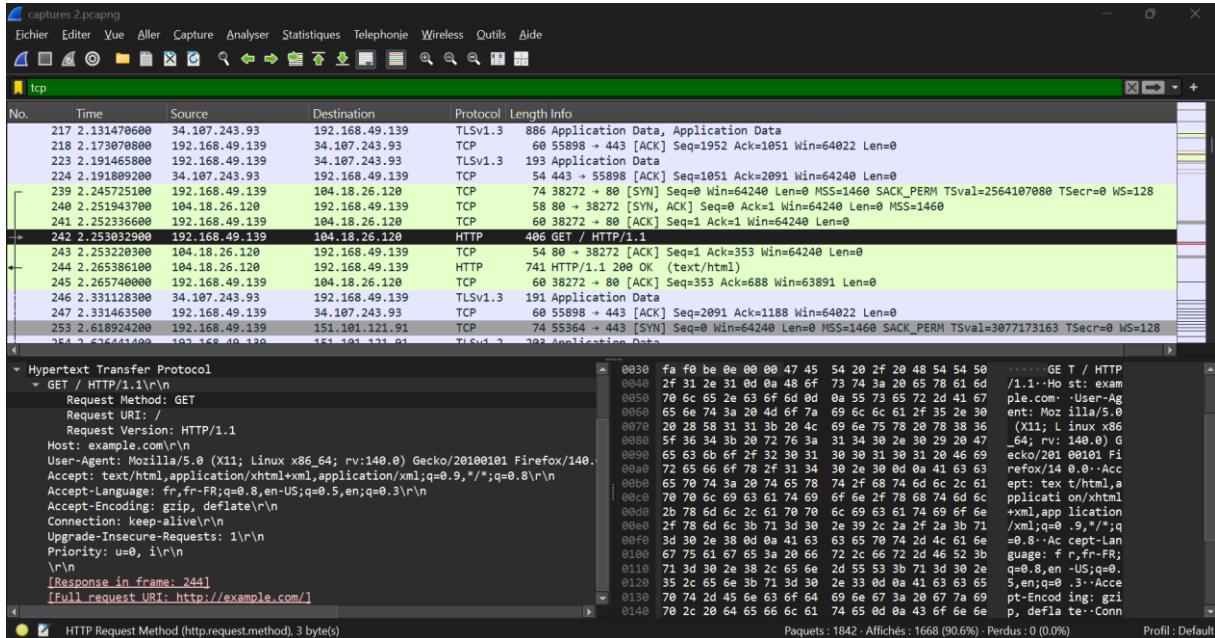
The screenshot shows a network traffic capture in Wireshark. The packet list pane displays several packets, with packet 9 (SYN), packet 10 (SYN-ACK), and packet 11 (ACK) highlighted. The packet details pane for packet 11 shows the following information:

- Transmission Control Protocol, Src Port: 38272, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
- Destination Port: 80
- [Stream index: 9]
- [Stream Packet Number: 3]
- [Conversation completeness: Complete, WITH_DATA (31)]
- [TCP Segment Len: 0]
- Sequence Number: 1 (relative sequence number)
- Sequence Number (raw): 3031804913
- [Next Sequence Number: 1 (relative sequence number)]
- Acknowledgment Number: 1 (relative ack number)
- Acknowledgment number (raw): 1588935851
- 0010 ... = Header Length: 20 bytes (5)
- Flags: 0x010 (ACK)
- Window: 64240
- [Calculated window size: 64240]

Paquet	Etape	Drapeaux	IP source	IP destination	N° Sequence
9	SYN	SYN	192.168.49.139	104.18.26.120	3031804912
10	SYN-ACK	SYN-ACK	104.18.26.120	192.168.49.139	1588935850
11	ACK	ACK	192.168.49.139	104.18.26.120	3031804913

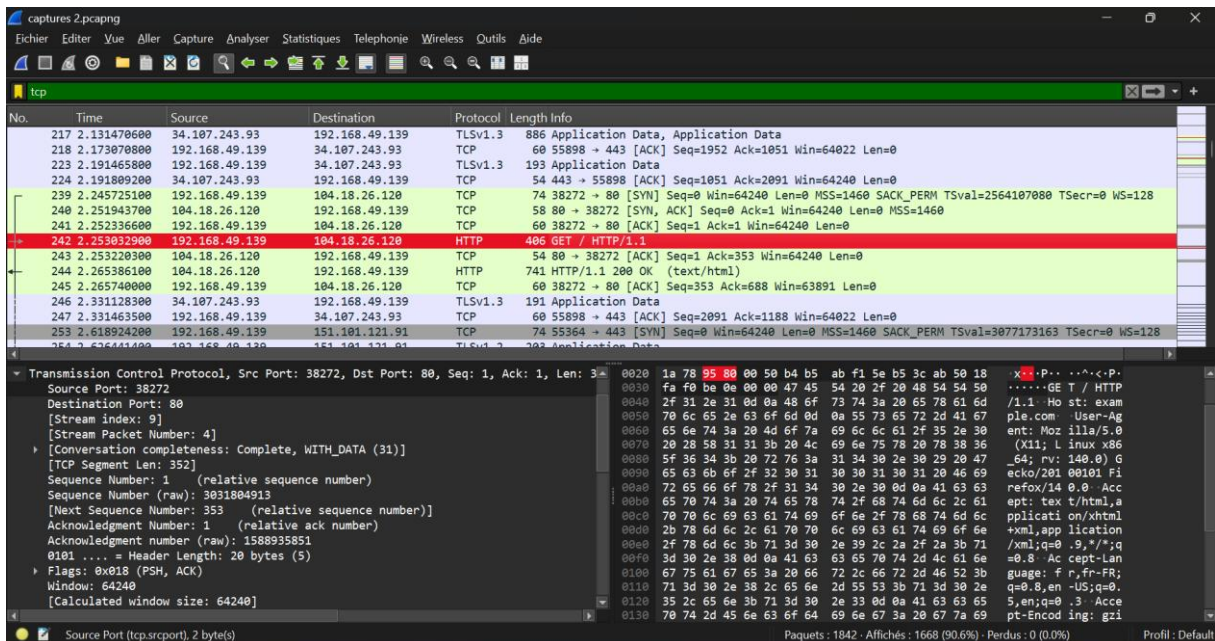
3-Analyse du trafic HTTP

Analysons la première requête HTTP « GET » :



La méthode HTTP utilisée est GET. La ressource demandée est « example.com »

La version utilisée est HTTP/1.1



Le numéro de séquence (Seq) = 1

La taille du segment (TCP Segment Len) = 352

Le serveur renvoie un segment « ACK » pour accuser réception de la requête « GET » :

The screenshot shows a Wireshark capture of network traffic. The main pane displays a list of packets. Packet 243 is highlighted in red and is a TCP segment with the following details:

- Source Port: 80
- Destination Port: 38272
- Seq: 1
- Ack: 353
- Len: 20

The packet details pane shows the following information:

- Transmission Control Protocol, Src Port: 80, Dst Port: 38272, Seq: 1, Ack: 353, Len: 20
- Destination Port: 38272
- [Stream index: 9]
- [Stream Packet Number: 5]
- [Conversation completeness: Complete, WITH_DATA (31)]
- [TCP Segment Len: 0]
- Sequence Number: 1 (relative sequence number)
- Sequence Number (raw): 1588935851
- [Next Sequence Number: 1 (relative sequence number)]
- Acknowledgment Number: 353 (relative ack number)
- Acknowledgment number (raw): 3031805265
- 0101 ... = Header Length: 20 bytes (5)
- Flags: 0x010 (ACK)
- Window: 64240
- [Calculated window size: 64240]

The packet bytes pane shows the raw data of the segment: 0000 00 0c 29 01 37 fb 00 50 56 ed d4 67 08 00 45 00 0010 00 28 1f 79 00 00 80 06 a6 99 68 12 1a 78 c0 a8 0020 31 80 60 50 95 80 5e b5 3c ab b4 b5 ad 51 50 10 0030 fa f0 ac ed 00 00

Dans l'ACK renvoyé par le serveur, le champ « Acknowledgment Number » à la valeur : 353

L'Acknowledgment Number (numéro d'acquiescement) sert à indiquer à l'émetteur le prochain octet que le récepteur s'attend à recevoir. Il se calcule en prenant le numéro de séquence du segment reçu, auquel on ajoute la taille de ses données utiles (le *payload*).

$$\text{Seq(reçu)} + \text{Len(reçu)} = \text{Ack(renvoyé)}$$

$$1 + 352 = 353$$

Le serveur envoie ensuite la réponse HTTP contenant le contenu de la page :

The screenshot shows a Wireshark capture of an HTTP response. The packet list pane shows packet 244 as an HTTP 200 OK response. The packet details pane shows the response structure, including the status code 200 and the 'Transfer-Encoding: chunked' header. The packet bytes pane shows the raw data of the response, including the status line and the start of the chunked body.

```
244 2.265386100 104.18.26.120 192.168.49.139 HTTP 741 HTTP/1.1 200 OK (text/html)
245 2.265740000 192.168.49.139 104.18.26.120 TCP 60 38272 -> 80 [ACK] Seq=353 Ack=688 Win=63891 Len=0
246 2.331128300 34.107.243.93 192.168.49.139 TLSv1.3 191 Application Data
247 2.331463500 192.168.49.139 34.107.243.93 TCP 60 55898 -> 443 [ACK] Seq=2091 Ack=1188 Win=64022 Len=0
253 2.618924200 192.168.49.139 151.101.121.91 TCP 74 55364 -> 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3077173163 TSecr=0 WS=128
```

```
Hypertext Transfer Protocol, has 2 chunks (including last chunk)
  HTTP/1.1 200 OK\r\n
  Response Version: HTTP/1.1
  Status Code: 200
  [Status Code Description: OK]
  Response Phrase: OK
  Date: Mon, 06 Apr 2026 10:47:46 GMT\r\n
  Content-Type: text/html\r\n
  Transfer-Encoding: chunked\r\n
  Connection: keep-alive\r\n
  Server: cloudflare\r\n
  Last-Modified: Tue, 24 Mar 2026 22:07:32 GMT\r\n
  Allow: GET, HEAD\r\n
  cf-cache-status: HIT\r\n
  Age: 2982\r\n
  Content-Encoding: gzip\r\n
```

Le code renvoyé est bien « 200 OK »

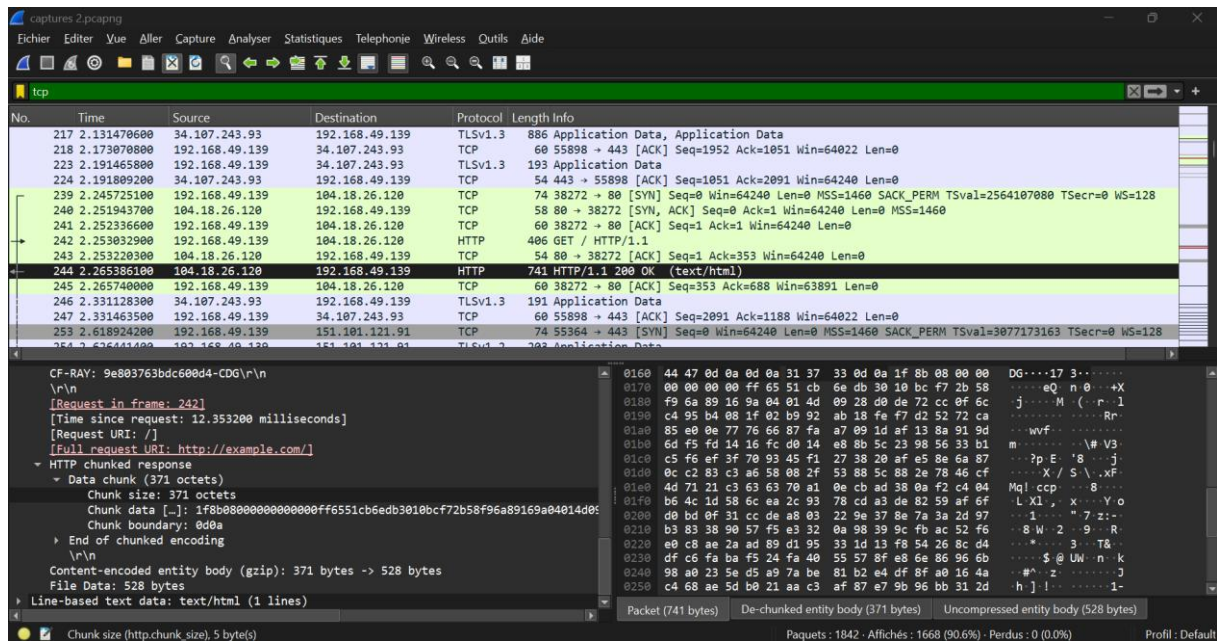
Le contenu renvoyé par le serveur est envoyé par morceaux. Si on regarde « Transfer-encoding: chunked » :

This screenshot is similar to the first one, but it highlights the 'Transfer-Encoding: chunked' header in the packet details pane. The packet bytes pane shows the raw data of the response, including the status line and the start of the chunked body.

```
244 2.265386100 104.18.26.120 192.168.49.139 HTTP 741 HTTP/1.1 200 OK (text/html)
245 2.265740000 192.168.49.139 104.18.26.120 TCP 60 38272 -> 80 [ACK] Seq=353 Ack=688 Win=63891 Len=0
246 2.331128300 34.107.243.93 192.168.49.139 TLSv1.3 191 Application Data
247 2.331463500 192.168.49.139 34.107.243.93 TCP 60 55898 -> 443 [ACK] Seq=2091 Ack=1188 Win=64022 Len=0
253 2.618924200 192.168.49.139 151.101.121.91 TCP 74 55364 -> 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3077173163 TSecr=0 WS=128
```

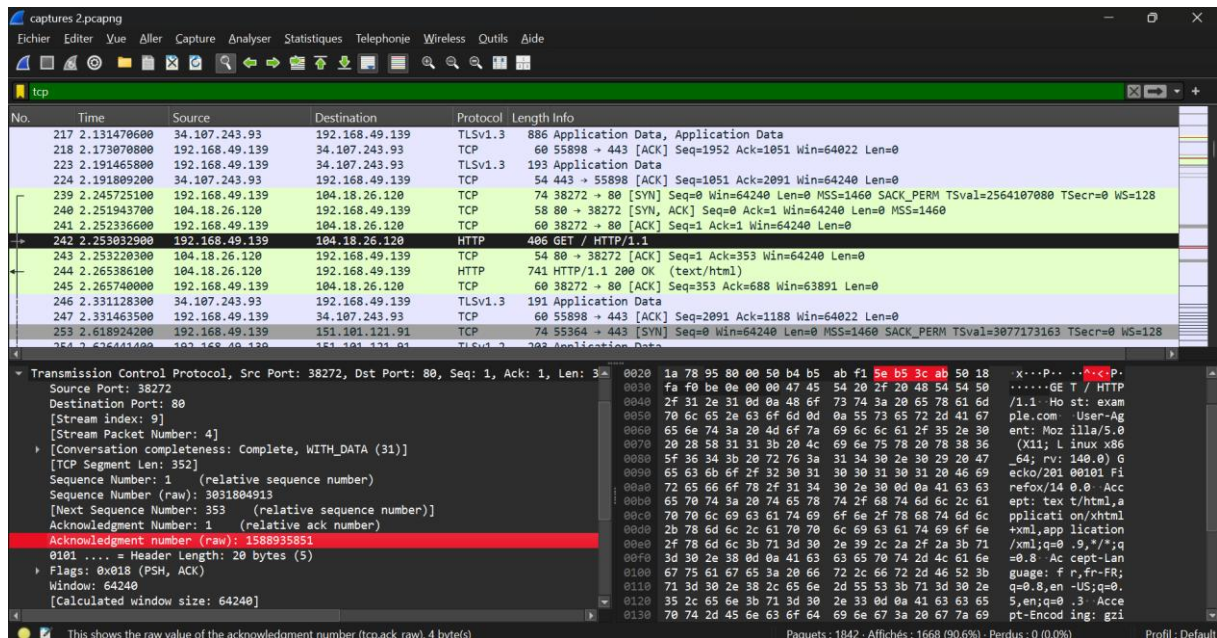
```
Hypertext Transfer Protocol, has 2 chunks (including last chunk)
  HTTP/1.1 200 OK\r\n
  Response Version: HTTP/1.1
  Status Code: 200
  [Status Code Description: OK]
  Response Phrase: OK
  Date: Mon, 06 Apr 2026 10:47:46 GMT\r\n
  Content-Type: text/html\r\n
  Transfer-Encoding: chunked\r\n
  Connection: keep-alive\r\n
  Server: cloudflare\r\n
  Last-Modified: Tue, 24 Mar 2026 22:07:32 GMT\r\n
  Allow: GET, HEAD\r\n
  cf-cache-status: HIT\r\n
  Age: 2982\r\n
  Content-Encoding: gzip\r\n
```

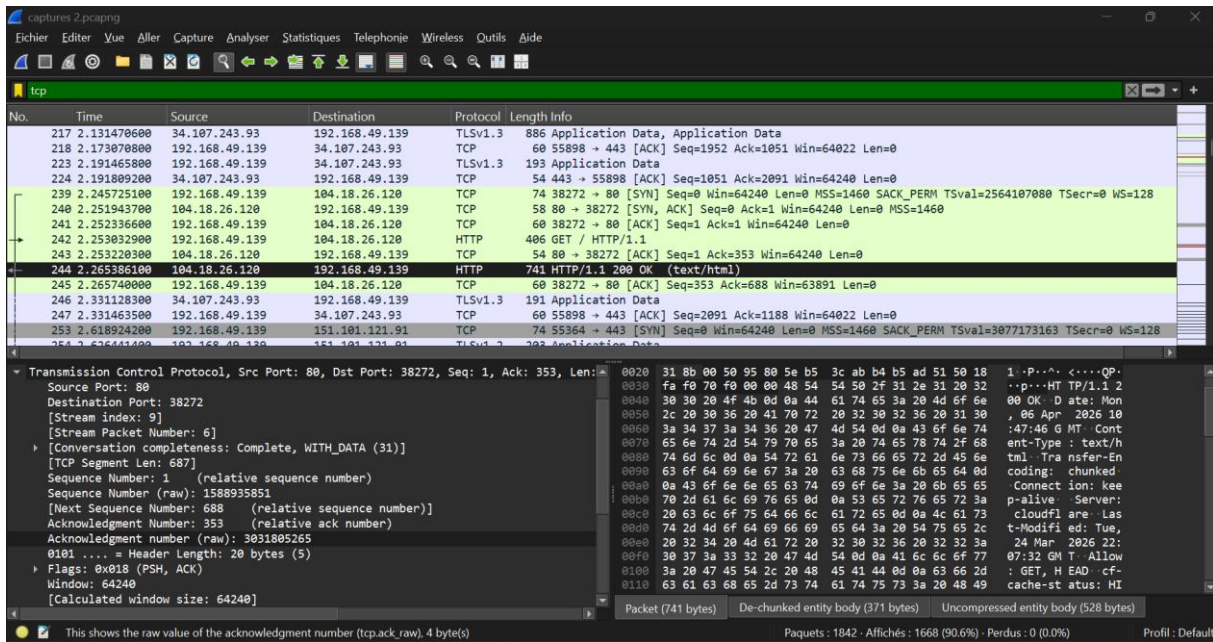
Si on regarde le menu « http chunked response », on voit de le serveur a envoyé un morceau de données de taille 371 octets :



En dessous, on voit le menu « File Data » qui nous indique que la taille totale des données envoyées est 528 bytes

Comparons les « Sequence number » et « acknowledgment number » entre la requête « GET » et la réponse du serveur :





Paquet HTTP	GET	200 OK
Sequence number relatif	1	1
Sequence number (raw)	3031804913	1588935851
Acknowledgment number relatif	1	353
Acknowledgment number (raw)	1588935851	3031805265

Schéma :

1. Envoi de la requête web

Requête HTTP GET

- Seq = 1 (Le client commence à parler)
- Ack = 1 (Le client confirme qu'il a reçu le SYN-ACK précédent)
- Len = 352 (La taille de la requête GET)

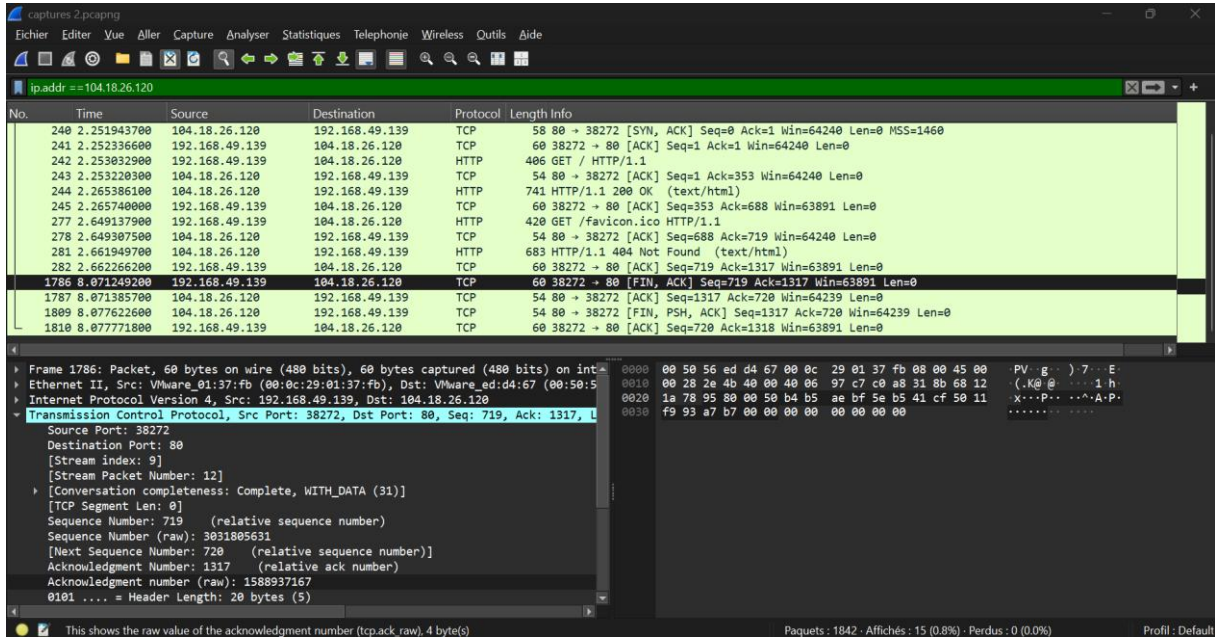
2. Traitement et réponse du serveur

Réponse HTTP 200 OK (Début des chunks)

- Seq = 1 (Le serveur commence à envoyer ses données web)
- Ack = 353 (Le serveur valide les 352 octets du GET : 1 + 352 = 353)

4- Analyse de la fermeture TCP

Analysons désormais la fermeture de la connexion TCP :

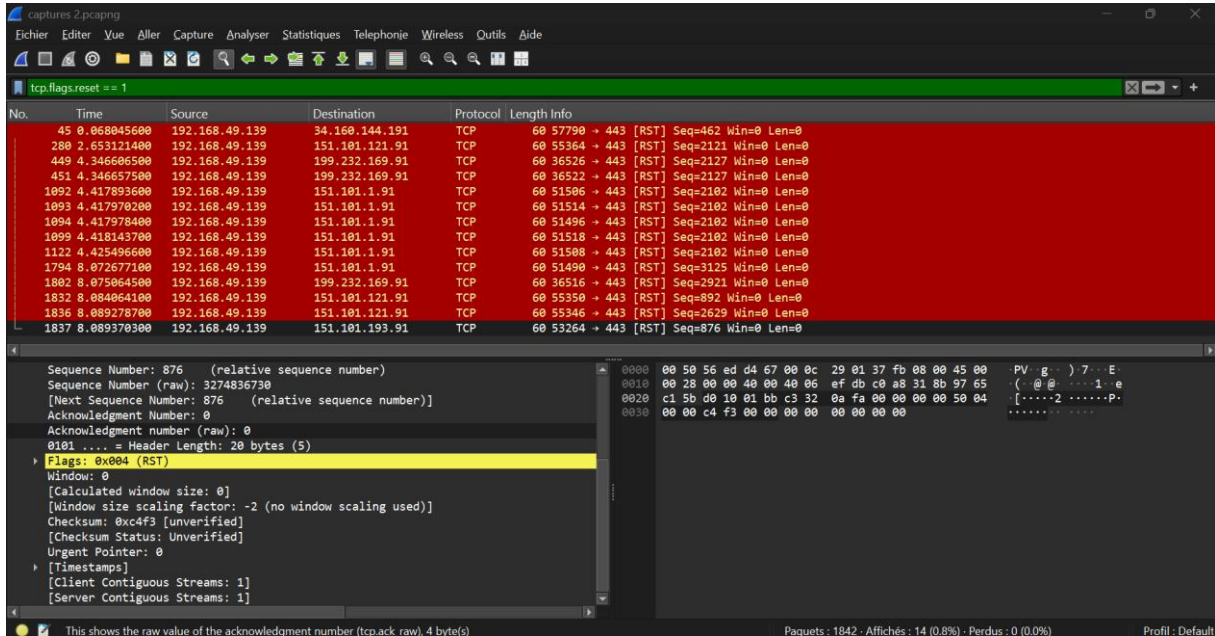


Paquet	Drapeau	Emetteur	Seq / Ack	Signification
#1786	FIN + ACK	Serveur	Seq relatif = 719 Seq (raw) = 3031805631 Ack relatif = 1317 Ack (raw) = 1588937167	Le serveur demande la fermeture
#1787	ACK	Client	Seq relatif = 1317 Seq (raw) = 1588937167 Ack relatif = 720 Ack (raw) = 3031805632	Le client confirme la fermeture du serveur
#1809	FIN + ACK	Client	Seq relatif = 1317 Seq (raw) = 1588937167 Ack relatif = 720 Ack (raw) = 3031805632	Le client ferme la connexion et la confirme
#1810	ACK	Serveur	Seq relatif = 720 Seq (raw) = 3031805632 Ack relatif = 1318 Ack (raw) = 1588937168	Le serveur confirme la fermeture du client

Le drapeau FIN incrémente le numéro d'acquittement de 1 comme le drapeau SYN car TCP doit garantir de manière absolue que les drapeaux SYN et FIN ont bien été reçus. C'est pour cela qu'ils doivent consommer un octet. Lors de l'acquittement, la présence de cet octet est nécessaire car elle prouve que l'autre machine a bien reçu l'information.

5-Cas des drapeaux RST

Le drapeau RST signifie une fermeture abrupte de connexion :

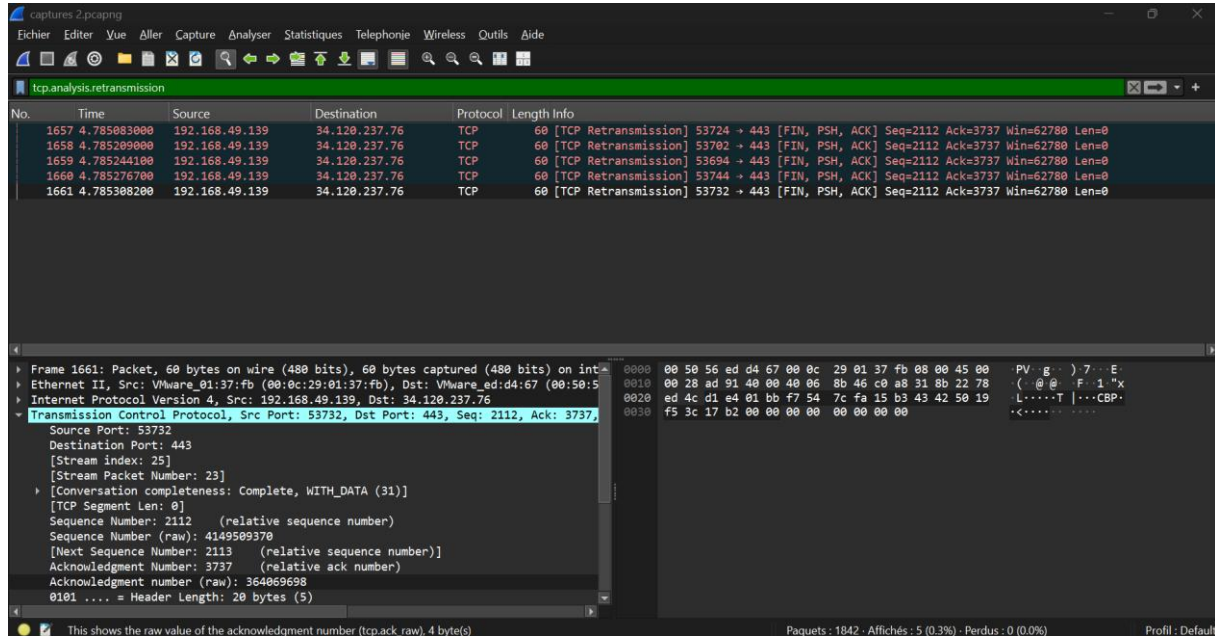


Nous avons plusieurs Drapeaux RST dans notre capture.

Dans un contexte normal, c'est le drapeau FIN qui met fin à une connexion. Le drapeau RST est plus brutal. Il peut survenir en cas de port fermé, de fermeture brutale de l'application, de blocage pare-feu, etc...

6-Analyse des retransmissions TCP

On utilise le filtre « tcp.analysis.retransmission » pour voir les retransmissions TCP :

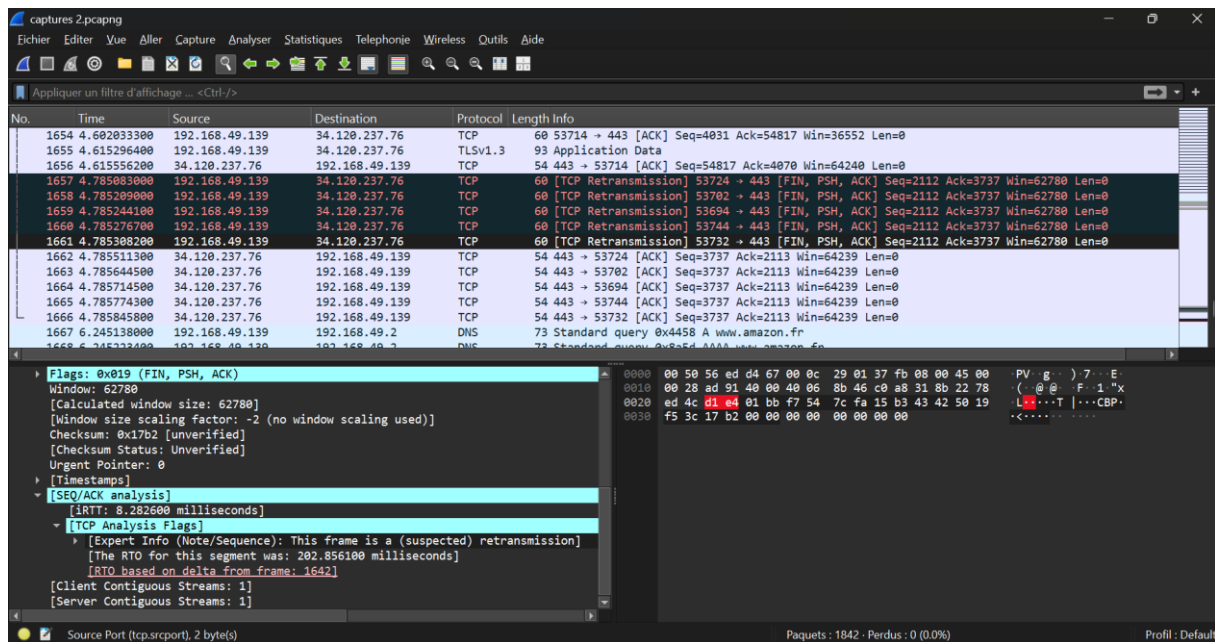


Wireshark les affiche en noir par défaut

Il existe deux types de retransmissions TCP :

Type	Mécanisme
Retransmission RTO	L'émetteur attend l'expiration d'un minuteur (RTO). Si aucun ACK n'est reçu avant, il retransmet le paquet.
Retransmission rapide	Si l'émetteur reçoit 3 ACK dupliqués consécutifs (dupthresh = 3), il retransmet immédiatement sans attendre le RTO.

On peut voir le RTO ainsi que les différents ACK envoyés :



Il s'agit retransmissions RTO car il n'y a pas de ACK dupliqués. Les retransmissions ont bien attendu la fin du RTO pour s'effectuer.

En temps normal, quand deux ordinateurs communiquent via TCP, le récepteur envoie un ACK (Acknowledgment / Accusé de réception) pour valider ce qu'il a reçu.

La subtilité de TCP, c'est que l'ACK indique le prochain paquet attendu, et non pas celui qui vient d'arriver.

- Si je reçois le paquet n°100, je renvoie un ACK 101

Le problème survient quand un paquet se perd, mais que les suivants arrivent. Imaginons qu'un serveur vous envoie les paquets 1, 2, 3, 4 et 5.

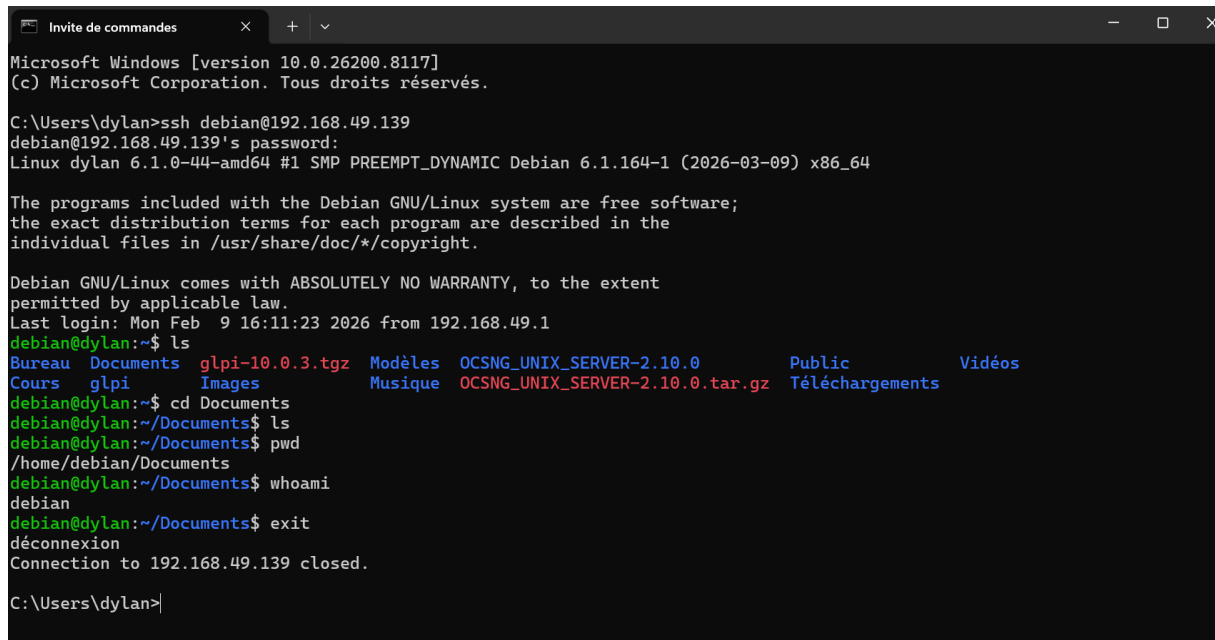
1. Vous recevez le paquet 1. Vous répondez : ACK 2 (J'attends le 2).
2. Le paquet 2 est perdu par le réseau.
3. Vous recevez le paquet 3. Vous ne pouvez pas accuser réception du 3, car il vous manque le 2 (TCP a besoin de tout dans l'ordre).
4. Votre ordinateur va donc répéter sa dernière demande valide : il renvoie un ACK 2. C'est cela, un ACK dupliqué (Dup ACK).

La retransmission rapide se déclenche à la réception de 3 ACK dupliqués consécutifs.

Au lieu d'attendre que le chronomètre (RTO) expire, le serveur renvoie immédiatement le paquet perdu au bout de 3 ACK

7-Analyse du trafic SSH

Connectons-nous en SSH à notre VM depuis notre machine physique et effectuons quelques commandes :



```
Microsoft Windows [version 10.0.26200.8117]
(c) Microsoft Corporation. Tous droits réservés.

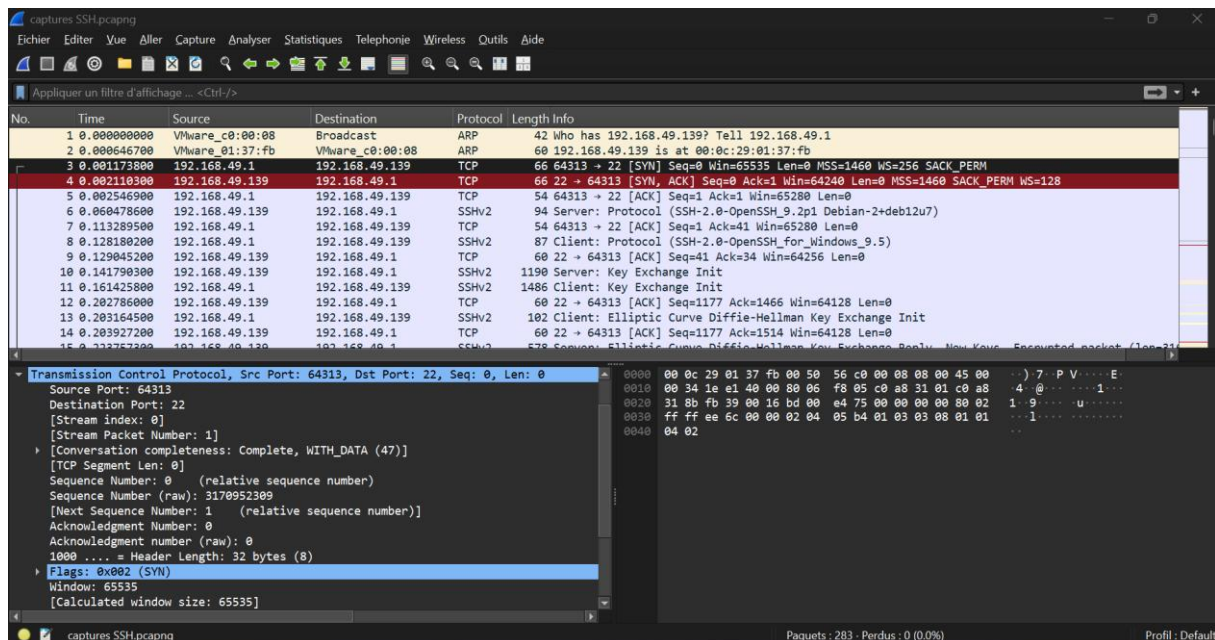
C:\Users\dylan>ssh debian@192.168.49.139
debian@192.168.49.139's password:
Linux dylan 6.1.0-44-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.164-1 (2026-03-09) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 9 16:11:23 2026 from 192.168.49.1
debian@dylan:~$ ls
Bureau Documents glpi-10.0.3.tgz Modèles OCSNG_UNIX_SERVER-2.10.0 Public Vidéos
Cours glpi Images Musique OCSNG_UNIX_SERVER-2.10.0.tar.gz Téléchargements
debian@dylan:~$ cd Documents
debian@dylan:~/Documents$ ls
debian@dylan:~/Documents$ pwd
/home/debian/Documents
debian@dylan:~/Documents$ whoami
debian
debian@dylan:~/Documents$ exit
déconnexion
Connection to 192.168.49.139 closed.

C:\Users\dylan>
```

On peut ensuite observer le three-way handshake lors de la connexion SSH :



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Vmware_c0:00:08	Broadcast	ARP	42	who has 192.168.49.139? Tell 192.168.49.1
2	0.000646700	Vmware_01:37:fb	Vmware_c0:00:08	ARP	60	192.168.49.139 is at 00:0c:29:01:37:fb
3	0.001173800	192.168.49.1	192.168.49.139	TCP	66	64313 → 22 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
4	0.002110300	192.168.49.139	192.168.49.1	TCP	66	22 → 64313 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM Win=128
5	0.002546900	192.168.49.1	192.168.49.139	TCP	54	64313 → 22 [ACK] Seq=1 Ack=1 Win=65280 Len=0
6	0.004786000	192.168.49.139	192.168.49.1	SSHv2	94	Server: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u7)
7	0.113289500	192.168.49.1	192.168.49.139	TCP	54	64313 → 22 [ACK] Seq=1 Ack=41 Win=65280 Len=0
8	0.128180200	192.168.49.1	192.168.49.139	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_for_Windows_9.5)
9	0.129045200	192.168.49.139	192.168.49.1	TCP	60	22 → 64313 [ACK] Seq=41 Ack=34 Win=64256 Len=0
10	0.141790300	192.168.49.139	192.168.49.1	SSHv2	1190	Server: Key Exchange Init
11	0.161425800	192.168.49.1	192.168.49.139	SSHv2	1486	Client: Key Exchange Init
12	0.202786000	192.168.49.139	192.168.49.1	TCP	60	22 → 64313 [ACK] Seq=1177 Ack=1466 Win=64128 Len=0
13	0.203164500	192.168.49.1	192.168.49.139	SSHv2	102	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
14	0.203927200	192.168.49.139	192.168.49.1	TCP	60	22 → 64313 [ACK] Seq=1177 Ack=1514 Win=64128 Len=0
15	0.212757200	192.168.49.139	192.168.49.1	SSHv2	578	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply: New Key: Encrypted packet (100...

Transmission Control Protocol, Src Port: 64313, Dst Port: 22, Seq: 0, Len: 0
Destination Port: 22
[Stream index: 0]
[Stream Packet Number: 1]
[Conversation completeness: Complete, WITH_DATA (47)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 3170952309
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1000 ... = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)
Window: 65535
[Calculated window size: 65535]

En effet, le protocole SSH utilise bien le TCP. Le port utilisé par le protocole SSH est 22

Regardons ensuite les 2 premier paquets SSH :

The screenshot shows a Wireshark capture of network traffic. The packet list pane highlights packet 6, which is an SSH2 Server: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u7) sent from 192.168.49.139 to 192.168.49.1. The packet details pane shows the following structure:

- Flags: 0x018 (PSH, ACK)
- Window: 502
- [Calculated window size: 64256]
- [Window size scaling factor: 128]
- Checksum: 0x1fae [Unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- [Timestamps]
- [SEQ/ACK analysis]
- [Client Contiguous Streams: 1]
- [Server Contiguous Streams: 1]
- TCP payload (40 bytes)
- SSH Protocol
 - Protocol: SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u7
 - [Direction: Server to Client]

The packet bytes pane shows the raw data, including the ASCII string "SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u7".

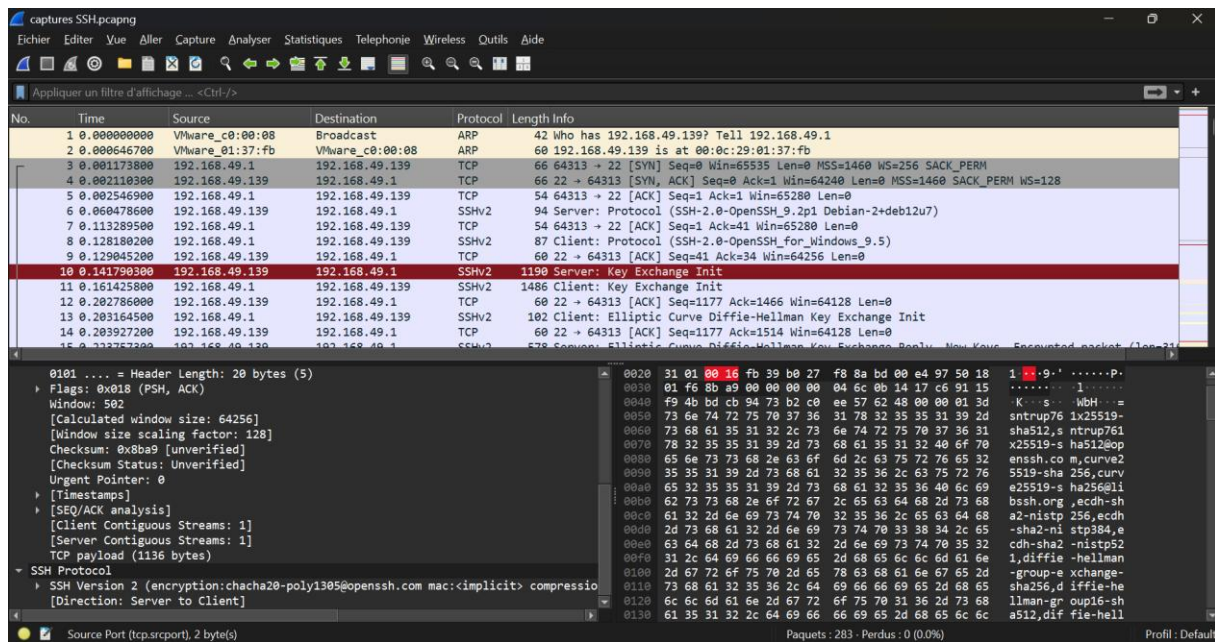
The screenshot shows a Wireshark capture of network traffic. The packet list pane highlights packet 8, which is an SSH2 Client: Protocol (SSH-2.0-OpenSSH_for_Windows_9.5) sent from 192.168.49.139 to 192.168.49.1. The packet details pane shows the following structure:

- Flags: 0x018 (PSH, ACK)
- Window: 256
- [Calculated window size: 65280]
- [Window size scaling factor: 256]
- Checksum: 0x3d39 [Unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- [Timestamps]
- [SEQ/ACK analysis]
- [Client Contiguous Streams: 1]
- [Server Contiguous Streams: 1]
- TCP payload (33 bytes)
- SSH Protocol
 - Protocol: SSH-2.0-OpenSSH_for_Windows_9.5
 - [Direction: Client to Server]

The packet bytes pane shows the raw data, including the ASCII string "SSH-2.0-OpenSSH_for_Windows_9.5".

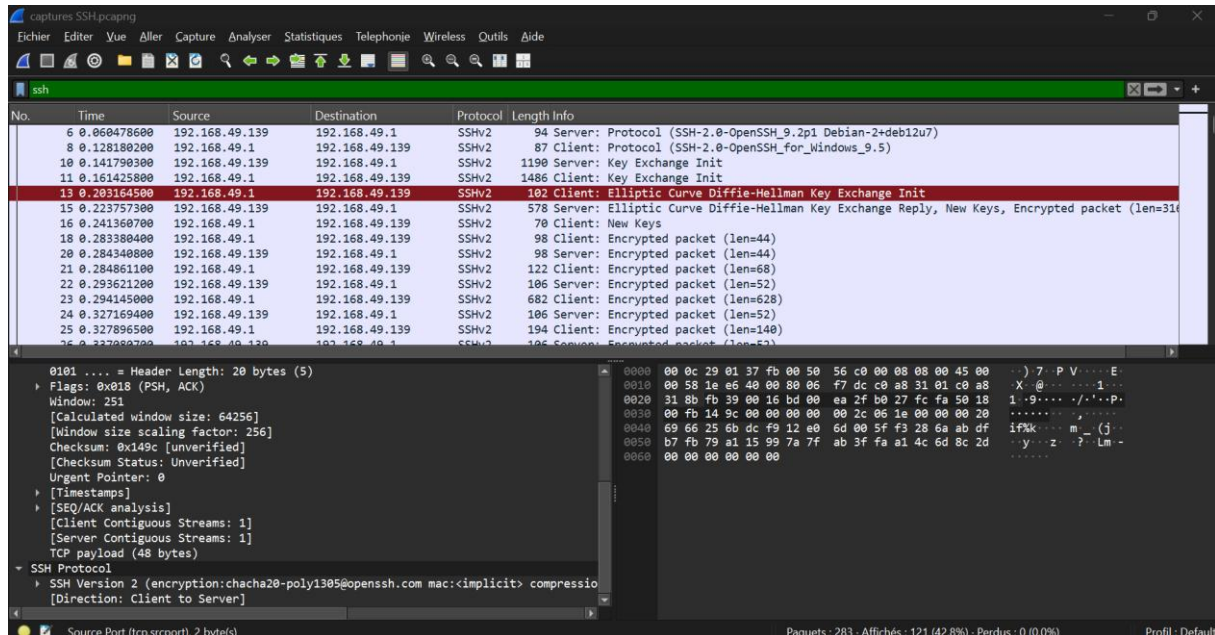
On voit les deux machines s'annoncer en clair. Les deux paquets annoncent la version utilisée par le client et le serveur : SSH 2.0 OpenSSH for Windows 9.5, SSH 2.0 OpenSSH for Debian

On peut ensuite observer les paquets « Key exchange unit » client / serveur :



Ces paquets correspondent à l'échange des clés de sécurité et indiquent l'algorithme de sécurité utilisé par les deux machines.

L'échange des clés Diffie-Hellman se produit ensuite :

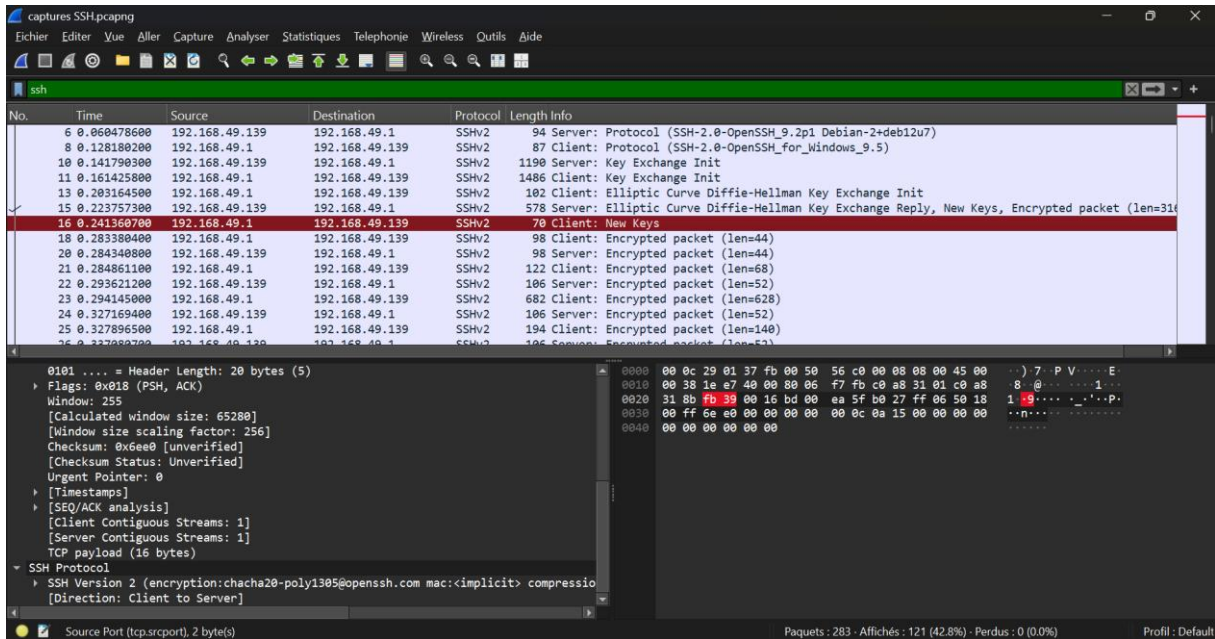


Nous observons deux paquets :

- Diffie-Hellman Key Exchange Init
- Diffie-Hellman Key Exchange Reply

C'est à ce moment que les machines vont créer ensemble un secret partagé sans jamais le faire transiter sur le réseau.

A partir du paquet suivant « New Keys », la connexion devient totalement chiffrée :



On peut observer tous les paquets suivants sont encryptés « Encrypted Packet »

Ces paquets encryptés correspondent aux commandes qui ont été transmises à la machine virtuelle en SSH : ls, cd, pwd, whoami

Suite à la commande « exit » entrée en SSH, on peut observer la fin de la communication TCP avec le drapeau « RST ACK » sur le port 22 (SSH) :

