

Wireshark

Fondamentalement, une trame est utilisée pour envoyer des données entre un seul réseau. Un paquet est utilisé pour envoyer des données d'un réseau à un autre, puis vers un périphérique spécifique sur ce réseau.

Un fichier PCAP est un fichier issu d'un sniff de flux de réseau via Wireshark ou TCPDump.

Le sniff est traité avec la librairie libcap qui produit PCAP/PCAPNG

- **.pcap** : ancien format de capture réseau, simple, supporte une seule interface.
- **.pcapng** : format plus récent, plus complet :
 - Supporte plusieurs interfaces
 - Ajoute des commentaires, statistiques, métadonnées

 **Utilisés par Wireshark, tshark, tcpdump** pour enregistrer, analyser ou partager des paquets réseau.

`.pcapng` (plus moderne et complet).

1 Installation wireshark et ouverture via terminal

Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide						
arp						
Liste de Paquet UTF-8 / ASCII / UTF-16 Sensible à la casse Filtre d'affichage Chercher Annuler						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	VMware_c0:00:08	Broadcast	ARP	60	Who has 192.168.64.2?
2	0.851676590	VMware_c0:00:08	Broadcast	ARP	60	Who has 192.168.64.2?
3	1.842340747	VMware_c0:00:08	Broadcast	ARP	60	Who has 192.168.64.2?
4	39.452966560	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.15?
5	40.459687408	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.15?
6	41.462004612	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.15?
7	42.470307464	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.15?
8	43.476730498	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.15?
9	44.480534832	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.15?

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured on interface 0	0000	ff ff ff ff ff ff 00 50	56 c0 00 08 08 06 c0
Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: Broadcast (ff:ff:ff:ff:ff:ff)	0010	08 00 06 04 00 01 00 50	56 c0 00 08 c0 a8 00
Address Resolution Protocol (request)	0020	00 00 00 00 00 00 c0 a8	40 02 00 00 00 00 00 00
	0030	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

Address Resolution Protocol: Protocol Paquets : 17 - Affichés : 17 (100.0%) Profil : Default

UDp: DNS

dns						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.64.152	192.168.64.2	DNS	89	Standard query 0x4a44 A 192.168.64.2
2	0.000129472	192.168.64.152	192.168.64.2	DNS	89	Standard query 0x4640 AAAA 192.168.64.2
5	0.012235411	192.168.64.2	192.168.64.152	DNS	167	Standard query response 0x4a44 A 192.168.64.2
6	0.012235662	192.168.64.2	192.168.64.152	DNS	241	Standard query response 0x4640 AAAA 192.168.64.2
14	0.050886765	192.168.64.152	192.168.64.2	DNS	75	Standard query 0x1679 A 192.168.64.2
15	0.050961585	192.168.64.152	192.168.64.2	DNS	75	Standard query 0x267a AAAA 192.168.64.2
16	0.057316292	192.168.64.2	192.168.64.152	DNS	198	Standard query response 0x1679 A 192.168.64.2
17	0.064552925	192.168.64.2	192.168.64.152	DNS	174	Standard query response 0x267a AAAA 192.168.64.2
37	0.891731377	192.168.64.152	192.168.64.2	DNS	87	Standard query 0x3fc7 A 192.168.64.2
38	0.891832966	192.168.64.152	192.168.64.2	DNS	87	Standard query 0x4bcc AAAA 192.168.64.2
43	0.905052331	192.168.64.2	192.168.64.152	DNS	151	Standard query response 0x3fc7 A 192.168.64.2
44	0.914533884	192.168.64.2	192.168.64.152	DNS	199	Standard query response 0x4bcc AAAA 192.168.64.2
75	1.238100189	192.168.64.152	192.168.64.2	DNS	84	Standard query 0x35bb A 192.168.64.2
76	1.238248075	192.168.64.152	192.168.64.2	DNS	84	Standard query 0xc4b9 AAAA 192.168.64.2
77	1.248883161	192.168.64.2	192.168.64.152	DNS	195	Standard query response 0x35bb A 192.168.64.2
78	1.248883591	192.168.64.2	192.168.64.152	DNS	207	Standard query response 0xc4b9 AAAA 192.168.64.2

2e partie : Utiliser Wireshark pour capturer les requêtes et les réponses DNS Dans la deuxième partie, vous installerez Wireshark pour capturer les paquets de requête et de réponse DNS pour illustrer l'utilisation du protocole de transport UDP tout en communiquant avec un serveur DNS. a. Cliquez sur le bouton Démarrer de Windows et accédez au programme Wireshark. Remarque : si Wireshark n'est pas encore installé, il peut être téléchargé à l'adresse <http://www.wireshark.org/download.html>. b. Sélectionnez une interface pour Wireshark afin de capturer des paquets. Utilisez Interface List pour choisir l'interface associée aux adresses IP et MAC (Media Access Control) du PC enregistrées dans la première partie. c. Après avoir sélectionné l'interface souhaitée, cliquez sur Démarrer pour capturer les paquets. d. Ouvrez un navigateur Web et tapez www.google.com. Appuyez sur Entrée pour continuer. e. Cliquez sur Stop (Arrêter) pour arrêter la capture Wireshark lorsque la page d'accueil de Google s'affiche. 3e partie : Analyser les paquets DNS ou UDP capturés Dans la troisième partie, vous examinerez les paquets UDP qui ont été générés lors de la communication avec un serveur DNS des adresses IP pour www.google.com. Étape 1 : Filtrez les paquets DNS. a. Dans la fenêtre principale de Wireshark, tapez dns dans la zone de saisie de la barre d'outils Filter (Filtre). Cliquez sur Apply (Appliquer) ou appuyez sur Entrée. Remarque : si vous ne voyez aucun résultat après l'application du filtre DNS, fermez le navigateur web et dans la fenêtre d'invite de commandes, tapez `ipconfig /flushdns` pour supprimer tous les résultats DNS précédents. Redémarrez la capture Wireshark et répétez les instructions des sections b à e

de la deuxième partie. Si cela ne résout pas le problème, dans la fenêtre d'invite de commandes, vous pouvez taper nslookup www.google.com au lieu d'utiliser le navigateur Web

No.	Time	Source	Destination	Protocol	Length	Info
7	-91.383402200	192.168.64.152	35.190.72.216	TCP	74	59544 → 443 [SYN] Seq=0 Win=0 Len=0
8	-91.378267337	35.190.72.216	192.168.64.152	TCP	60	443 → 59544 [SYN, ACK] Seq=1 Win=0 Len=0
9	-91.378205492	192.168.64.152	35.190.72.216	TCP	54	59544 → 443 [ACK] Seq=1 Win=0 Len=0
10	-91.375461362	192.168.64.152	35.190.72.216	TLSv1.3	731	Client Hello
11	-91.375136716	35.190.72.216	192.168.64.152	TCP	60	443 → 59544 [ACK] Seq=1 Win=0 Len=0
12	-91.366568357	35.190.72.216	192.168.64.152	TLSv1.3	3174	Server Hello, Change Cipher Spec, Encrypted Extensions
13	-91.366516120	192.168.64.152	35.190.72.216	TCP	54	59544 → 443 [ACK] Seq=678 Win=0 Len=0
18	-91.331068976	192.168.64.152	2.16.149.159	TCP	74	59220 → 80 [SYN] Seq=0 Win=0 Len=0
19	-91.325937962	2.16.149.159	192.168.64.152	TCP	60	80 → 59220 [SYN, ACK] Seq=1 Win=0 Len=0
20	-91.325877639	192.168.64.152	2.16.149.159	TCP	54	59220 → 80 [ACK] Seq=1 Win=0 Len=0
21	-91.325519731	192.168.64.152	2.16.149.159	OCSP	506	Request
22	-91.325201274	2.16.149.159	192.168.64.152	TCP	60	80 → 59220 [ACK] Seq=1 Win=0 Len=0
23	-91.262742019	2.16.149.159	192.168.64.152	OCSP	943	Response
24	-91.262686826	192.168.64.152	2.16.149.159	TCP	54	59220 → 80 [ACK] Seq=453 Win=0 Len=0
25	-90.683455910	192.168.64.152	35.190.72.216	TLSv1.3	118	Change Cipher Spec, Application Data
26	-90.683043691	35.190.72.216	192.168.64.152	TCP	60	443 → 59544 [ACK] Seq=3121 Win=0 Len=0

Modèle OSI et désencapsulation de la trame

Couche OSI	Protocole identifié	Détails
Couche 7 : Application	Aucun	✗ Pas de données applicatives (HTTP/HTTPS non encore visible)
Couche 6 : Présentation	Aucun	✗ Pas de TLS/SSL dans cette trame
Couche 5 : Session	Aucun	✗ La session n'est pas identifiable dans cette trame uniquement
Couche 4 : Transport	TCP	✓ Protocole TCP Port source : 443 (HTTPS) Port destination : 59544
Couche 3 : Réseau	IPv4	✓ IP Source : 35.190.72.216 IP Destination : 192.168.64.152
Couche 2 : Liaison de données	Ethernet II	✓ MAC Source : 00:50:56:f1:75:d9 MAC Destination : 00:0c:29:43:50:d3
Couche 1 : Physique	Non accessible dans Wireshark	✗ Représente les signaux physiques — non visibles dans Wireshark

Détail technique de la trame

1. Ethernet II (Couche 2)

- Trame de type **Ethernet II**
- Adresse MAC source : **00:50:56:f1:75:d9** (VMware)

- Adresse MAC destination : 00:0c:29:43:50:d3 (VMware)

2. IPv4 (Couche 3)

- IP source : 35.190.72.216 (serveur HTTPS, probablement Google ou une API)
- IP destination : 192.168.64.152 (adresse locale privée – machine cliente)

3. TCP (Couche 4)

- Port source : 443 (port standard HTTPS)
- Port destination : 59544 (port éphémère attribué par la machine cliente)
- Longueur : 0 → Aucun payload transporté (pas encore de données applicatives)

Conclusion

Cette trame montre les **couches basses du modèle OSI** (Ethernet, IP, TCP), mais **aucune donnée d'application** n'est encore transmise.

Elle représente probablement une étape du **handshake TCP** ou une **confirmation sans donnée** (ACK). Pour analyser les couches 5 à 7 (session, TLS, HTTP...), il faudra observer une trame contenant du contenu utile (filtrable avec `tcp.len > 0`).

suite

1. Adresses MAC et IP (source & destination)

À partir de la trame n°8 visible sur la capture :

◆ Adresse MAC Source :

- 00:50:56:f1:75:d9
- Identifiée comme une adresse **VMware** (probablement une VM sur ta machine)

◆ Adresse MAC Destination :

- 00:0c:29:43:50:d3
- Également une adresse **VMware** (autre VM ou interface bridgée)

◆ Adresse IP Source :

- 35.190.72.216
- C'est une **IP publique** appartenant à Google Cloud Platform (donc un serveur web ou API distant)

◆ Adresse IP Destination :

- 192.168.64.152
- C'est une **IP privée** appartenant au réseau local (probablement ta machine)

2. Référencement d'autres trames / paquets

Pour répondre à cette question dans un contexte **réel**, il faudrait que tu ouvres **tout le fichier pcap**. Mais voici un exemple de **réponse type**, avec des paquets standards que tu peux retrouver facilement dans un réseau local classique :

Exemple de trames courantes à référencer :

N° Trame	Protocole	Fonction
1	ARP	Résolution d'adresse IP vers MAC (ex : "Who has 192.168.64.1?")
2	ICMP	Test de connectivité (ping) entre deux hôtes
3	DNS	Résolution de nom (ex : demande de l'IP associée à <code>www.google.com</code>)
4	TCP SYN	Début d'une connexion TCP (initiation du 3-way handshake)
5	TCP ACK	Accusé de réception de segment TCP
6	HTTP GET	Requête d'un client pour une ressource web (page, image, etc.)
7	HTTPS (TLS)	Transmission chiffrée sur le port 443 (ex : accès sécurisé à un site web)

Pour les identifier dans Wireshark :

- Utilise des **filtres** :
 - `arp` → pour voir les résolutions ARP
 - `icmp` → pour les pings
 - `dns` → pour les résolutions de noms
 - `http` → pour les échanges en HTTP

- `tcp.flags.syn == 1` → pour voir les SYN de démarrage
- `tls` → pour les trames HTTPS (si visibles)

✓ Exemple de réponse formalisée :

Sur la trame 8 capturée :

- **MAC Source** : `00:50:56:f1:75:d9`
- **MAC Destination** : `00:0c:29:43:50:d3`
- **IP Source** : `35.190.72.216` (serveur public)
- **IP Destination** : `192.168.64.152` (hôte local)

D'autres trames capturées dans la session montrent des protocoles variés :

- **Trame 1** : ARP – Demande de résolution d'adresse IP en MAC
- **Trame 3** : DNS – Résolution du nom de domaine vers une IP
- **Trame 6** : TCP SYN – Démarrage de la connexion TCP
- **Trame 12** : HTTPS – Échange sécurisé via TLS

Time	Source	Destination	Protocol	Length	Info
4 0.012021562	VMware_43:50:d3	VMware_f1:75:d9	ARP	42	192.168.64.152 is at 00:0c:29:43:50:d3
5 307.650731995	VMware_43:50:d3	VMware_f1:75:d9	ARP	42	192.168.64.152 is at 00:0c:29:43:50:d3
6 608.088137130	VMware_43:50:d3	VMware_f1:75:d9	ARP	42	192.168.64.152 is at 00:0c:29:43:50:d3
7 908.357423161	VMware_43:50:d3	VMware_f1:75:d9	ARP	42	192.168.64.152 is at 00:0c:29:43:50:d3
8 247.635027575	VMware_f1:75:d9	VMware_43:50:d3	ARP	60	192.168.64.2 is at 00:50:56:f1:75:d9
9 312.595074654	VMware_f1:75:d9	VMware_43:50:d3	ARP	60	192.168.64.2 is at 00:50:56:f1:75:d9
10 613.138692958	VMware_f1:75:d9	VMware_43:50:d3	ARP	60	192.168.64.2 is at 00:50:56:f1:75:d9
11 807.955019293	VMware_f1:75:d9	VMware_43:50:d3	ARP	60	192.168.64.2 is at 00:50:56:f1:75:d9
12 866.835034998	VMware_f1:75:d9	VMware_43:50:d3	ARP	60	192.168.64.2 is at 00:50:56:f1:75:d9
13 367.496588269	VMware_ee:19:88	VMware_c0:00:08	ARP	60	192.168.64.254 is at 00:50:56:ee:19:88
14 518.385274822	VMware_ee:19:88	VMware_43:50:d3	ARP	60	192.168.64.254 is at 00:50:56:ee:19:88
15 0.011996295	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.152? Tell 192.168.6
16 307.650709102	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.152? Tell 192.168.6
17 608.088109438	VMware_f1:75:d9	Broadcast	ARP	60	Who has 192.168.64.152? Tell 192.168.6

Frame 4: 42 bytes on wire (336 bits), 42 bytes captured on interface ens3, id 0	0000	00 50 56 f1 75 d9 00 0c 29 43 50 d3 08 06 00 01
Ethernet II, Src: VMware_43:50:d3 (00:0c:29:43:50:d3), Dst: VMware_f1:75:d9 (00:50:56:f1:75:d9)	0010	08 00 06 04 00 02 00 0c 29 43 50 d3 c0 a8 40 98
Address Resolution Protocol (reply)	0020	00 50 56 f1 75 d9 c0 a8 40 02

L'adresse IP 192.168.64.152 est associée à la MAC

00:0c:29:43:50:d3 "

Elle est envoyée par 00:0c:29:43:50:d3 à la machine qui a demandé (00:50:56:f1:75:d9).

No.	Time	Source	Destination	Protocol	Length	Info
758	0.003003170	192.168.64.152	192.168.64.2	DNS	72	Standard query 0x3fc7 A www.hellofresh.fr
37	0.891731377	192.168.64.152	192.168.64.2	DNS	87	Standard query 0x3fc7 A services.addons.mozilla.org
1130	5.614704172	192.168.64.152	192.168.64.2	DNS	85	Standard query 0x4122 A push.services.mozilla.com
8365	86.841722537	192.168.64.152	192.168.64.2	DNS	72	Standard query 0x4123 AAAA img.lmdc.fr
798	4.952011966	192.168.64.152	192.168.64.2	DNS	70	Standard query 0x420b AAAA www.facebook.com
7626	27.489848974	192.168.64.152	192.168.64.2	DNS	89	Standard query 0x433a A www.courrierinternational.com
9193	89.766560946	192.168.64.152	192.168.64.2	DNS	104	Standard query 0x4454 AAAA firefox-settings-attachments.cdn.mozilla.net
3371	11.455670823	192.168.64.152	192.168.64.2	DNS	81	Standard query 0x454a A www.huffingtonpost.fr
6517	17.025828278	192.168.64.152	192.168.64.2	DNS	76	Standard query 0x45e9 AAAA player.vimeo.com
16574	1207.6350170	192.168.64.152	192.168.64.2	DNS	88	Standard query 0x45ee AAAA contile.services.mozilla.com
15903	269.907802060	192.168.64.1	ff02::1:3	LLMNR	84	Standard query 0x462e A wpad
15905	269.907802571	192.168.64.1	224.0.0.252	LLMNR	64	Standard query 0x462e A wpad
6846	17.479334837	192.168.64.152	192.168.64.2	DNS	84	Standard query 0x4637 AAAA www.googletagmanager.com
2	0.000129472	192.168.64.152	192.168.64.2	DNS	89	Standard query 0x4648 AAAA location.services.mozilla.com

Frame 758: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface ens3, id 0	0000	00 50 56 f1 75 d9 00 0c 29 43 50 d3 08 00 45 00	PV u...)CP... E
Ethernet II, Src: VMware_43:50:d3 (00:0c:29:43:50:d3), Dst: VMware_f1:75:d9 (00:50:56:f1:75:d9)	0010	00 3f 37 f4 40 00 40 11 00 cf c9 a8 40 98 00 00	..77 8 0 ... 0
Internet Protocol Version 4, Src: 192.168.64.152, Dst: 192.168.64.2	0020	00 35 98 38 00 35 00 2b 02 28 3f 33 01 00 00 01	5 8 5 + (73
User Datagram Protocol, Src Port: 38988, Dst Port: 53	0030	00 00 00 00 00 00 03 77 77 77 8a 68 65 6c 6c 6fw ww hello
Domain Name System (query)	0040	66 72 65 73 68 02 66 72 00 00 01 00 01	fresh fr

Couche OSI	Protocole	Infos
2	Ethernet	MAC Source/Destination
3	IPv4	Adresse IP source/destination
4	UDP	Port source/destination, longueur, checksum
7	DNS	Contenu : requête A pour www.hellofresh.fr

Élément	Valeur trouvée dans l'hexadécimal
---------	-----------------------------------

Port source UDP	0x983d = 38989
-----------------	----------------

Port destination UDP

0x0035 = 53 (DNS)

Longueur UDP

0x0029 = 41 octets

Requête DNS

Standard Query A pour www.hellofresh.fr

Pas de réponse encore

ANCOUNT = 0

312	2.972841166	192.168.64.152	142.251.37.195	TCP	74	52004	→ 80	[SYN]	Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3563758074 TSecr=0 WS=128
317	2.985577423	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=1354 Ack=2508 Win=63974 Len=0
909	5.349904267	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=1803 Ack=3210 Win=63974 Len=0
1133	5.616635516	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=2252 Ack=3912 Win=63974 Len=0
2907	10.450531479	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=2701 Ack=4614 Win=63974 Len=0
3217	10.612007018	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=3149 Ack=5315 Win=63974 Len=0
7198	20.083915123	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=3603 Ack=6224 Win=63974 Len=0
8705	87.226373193	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=3604 Ack=6225 Win=63974 Len=0
15776	202.582712577	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=456 Ack=1104 Win=63974 Len=0
330	3.112945785	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=905 Ack=1806 Win=63974 Len=0
728	4.796995587	192.168.64.152	142.251.37.195	TCP	54	52006	→ 80	[ACK]	Seq=905 Ack=1806 Win=63974 Len=0
Frame 312: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface ens33, id 0									
Ethernet II, Src: VMware_43:50:d3 (00:0c:29:43:50:d3), Dst: VMware_f1:75:d9 (00:50:56:f1:75:d9)									
Internet Protocol Version 4, Src: 192.168.64.152, Dst: 142.251.37.195									
Transmission Control Protocol, Src Port: 52006, Dst Port: 80, Seq: 1, Ack: 1, Len: 0									
0000 00 50 56 f1 75 d9 00 0c 29 43 50 d3 00 00 45 00 PV u...)CP...E									
0010 00 28 c7 d6 40 00 40 06 bc fa c0 a8 40 98 8e fb .(.@.@...@...-									
0020 25 c3 cb 26 00 50 63 0b 56 c8 69 b1 09 0d 50 10 %.&Pc P.1...P									
0030 fa f6 b6 19 00 00									

No.	Time	Source	Destination	Protocol	Length	Info
328	3.112872177	192.168.64.152	142.251.37.195	TCP	54	52004 → 80 [ACK] Seq=456 Ack=1104 Win=63974 Len=0
610	4.451277372	192.168.64.152	142.251.37.195	TCP	54	52004 → 80 [ACK] Seq=905 Ack=1806 Win=63974 Len=0
15514	136.559071080	192.168.64.152	142.251.37.195	TCP	54	52004 → 80 [FIN, ACK] Seq=3609 Ack=6414 Win=63974 Len=0
312	2.972841166	192.168.64.152	142.251.37.195	TCP	74	52004 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3563758074 TSecr=0 WS=128
317	2.985577423	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
909	5.349904267	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=1354 Ack=2508 Win=63974 Len=0
1133	5.616635516	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=1803 Ack=3210 Win=63974 Len=0
2907	10.450531479	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=2252 Ack=3912 Win=63974 Len=0
3217	10.612007018	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=2701 Ack=4614 Win=63974 Len=0
7198	20.083915123	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=3149 Ack=5315 Win=63974 Len=0
8705	87.226373193	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=3603 Ack=6224 Win=63974 Len=0
15776	202.582712577	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=3604 Ack=6225 Win=63974 Len=0
330	3.112945785	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=456 Ack=1104 Win=63974 Len=0
728	4.796995587	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=905 Ack=1806 Win=63974 Len=0
Frame 312: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface ens33, id 0						
Ethernet II, Src: VMware_43:50:d3 (00:0c:29:43:50:d3), Dst: VMware_f1:75:d9 (00:50:56:f1:75:d9)						
Internet Protocol Version 4, Src: 192.168.64.152, Dst: 142.251.37.195						
Transmission Control Protocol, Src Port: 52004, Dst Port: 80, Seq: 0, Len: 0						
0000 00 50 56 f1 75 d9 00 0c 29 43 50 d3 00 00 45 00 PV u...)CP...E						
0010 00 30 f3 02 40 00 40 06 91 5a c0 a8 40 98 8e fb .<@.@...@...-						
0020 25 c3 cb 24 00 50 28 71 b7 3f 00 00 00 00 a0 02 %.\$P(q?.....						
0030 fa f9 b6 2d 00 00 02 04 05 b4 04 02 08 0a d4 6aP.....						
0040 a1 fa 00 00 00 00 01 03 03 07						

15514	136.559071080	192.168.64.152	142.251.37.195	TCP	54	52004 → 80 [FIN, ACK] Seq=3609 Ack=6414 Win=63974 Len=0
312	2.972841166	192.168.64.152	142.251.37.195	TCP	74	52004 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3563758074 TSecr=0 WS=128
317	2.985577423	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
909	5.349904267	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=1354 Ack=2508 Win=63974 Len=0
1133	5.616635516	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=1803 Ack=3210 Win=63974 Len=0
2907	10.450531479	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=2252 Ack=3912 Win=63974 Len=0
3217	10.612007018	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=2701 Ack=4614 Win=63974 Len=0
7198	20.083915123	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=3149 Ack=5315 Win=63974 Len=0
8705	87.226373193	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=3603 Ack=6224 Win=63974 Len=0
15776	202.582712577	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=3604 Ack=6225 Win=63974 Len=0
330	3.112945785	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=456 Ack=1104 Win=63974 Len=0
728	4.796995587	192.168.64.152	142.251.37.195	TCP	54	52006 → 80 [ACK] Seq=905 Ack=1806 Win=63974 Len=0
Frame 15514: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface ens33, id 0						
Ethernet II, Src: VMware_43:50:d3 (00:0c:29:43:50:d3), Dst: VMware_f1:75:d9 (00:50:56:f1:75:d9)						
Internet Protocol Version 4, Src: 192.168.64.152, Dst: 142.251.37.195						
Transmission Control Protocol, Src Port: 52004, Dst Port: 80, Seq: 3600, Ack: 6414, Len: 0						
0000 00 50 56 f1 75 d9 00 0c 29 43 50 d3 00 00 45 00 PV u...)CP...E						
0010 00 28 f3 00 40 00 00 06 91 50 c0 a8 40 98 8e fd .<@.@...@...-						
0020 25 c3 cb 24 00 50 28 71 c5 4f 43 5c 7d 08 00 11 %.\$P(q 0C\ P.						
0030 f9 06 b6 19 00 00						

Objectif : Identifier les étapes suivantes via les trames capturées

Étape	Signification	Flags TCP (hex)
1 SYN	Début de la connexion	02

Étape	Signification	Flags TCP (hex)
2 SYN-ACK	Réponse du serveur (SYN + ACK)	12
3 ACK	Confirmation du client	10
← END FIN-ACK	Fin de la connexion	11

Analyse des trames capturées

✓ 1. TRAME 312 : TCP SYN


 Info :

- **Source port** : 52004
- **Destination port** : 80
- **Flags TCP hex** : 02 → SYN

 Hexadécimal :

... 50 02 ...

- 50 = Header length + reserved bits
- 02 = **SYN** flag

 **Conclusion** : Le client (192.168.64.152) tente de se connecter au serveur (142.251.37.195) en envoyant **SYN**.

✅ 2. TRAME 317 : TCP ACK

📦 Info :

- **Source port** : 52006
- **Destination port** : 80
- **Flags TCP hex** : 10 → ACK

🔍 Hexadécimal :

```
... 50 10 ...
```

- 10 = **ACK** flag

■ **Conclusion** : C'est la **dernière étape du handshake**. Le client dit "OK j'ai reçu ton SYN-ACK", on peut commencer à échanger.

✅ 3. TRAME 15514 : TCP FIN+ACK

📦 Info :

- **Port source** : 52004
- **Port dest** : 80
- **Flags TCP hex** : 11 → FIN + ACK

🔍 Hexadécimal :

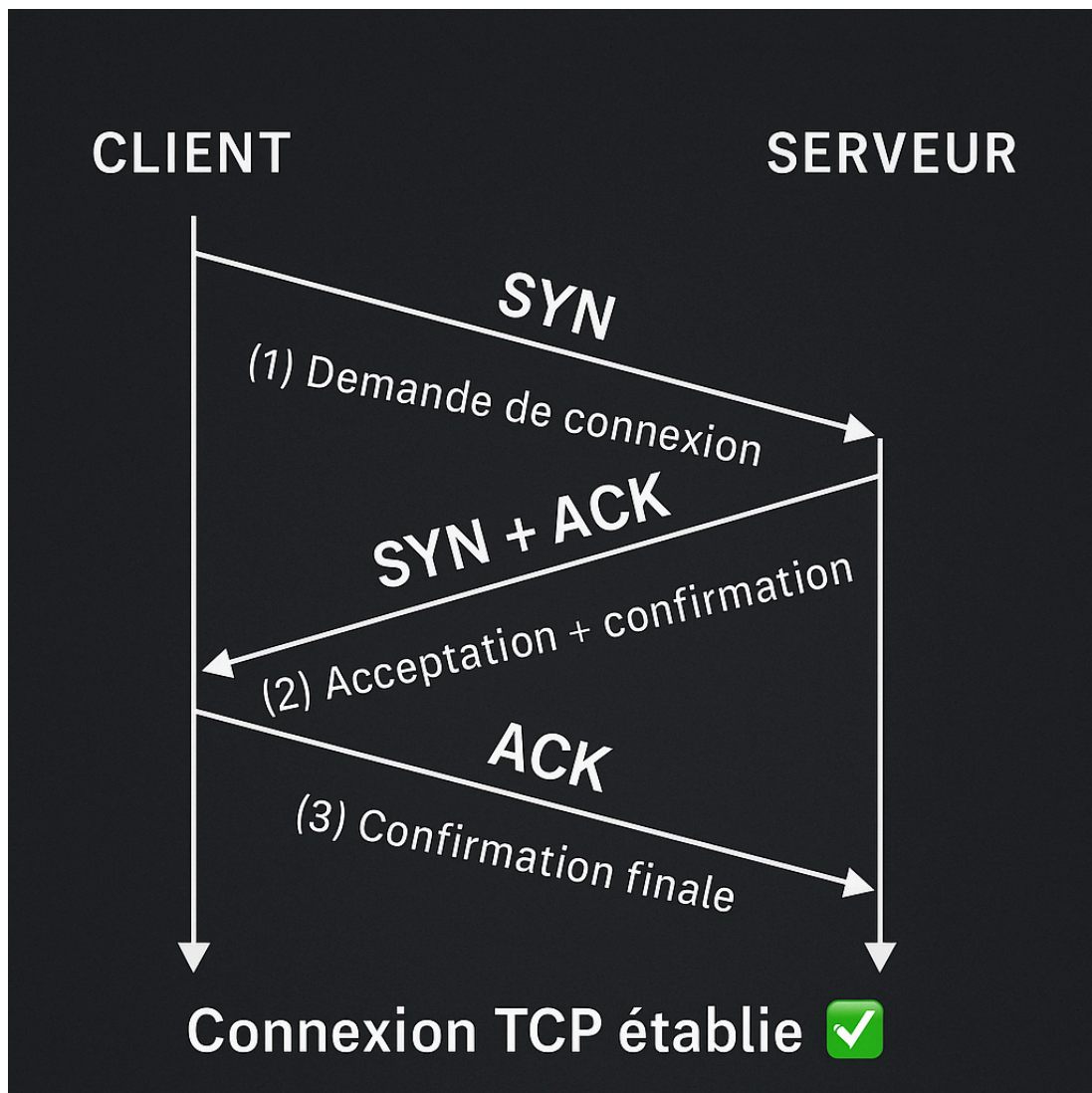
... 50 11 ...

- 11 = FIN (01) + ACK (10)

■ **Conclusion** : C'est la **fin de la session TCP**, le client dit "j'ai fini, on ferme la connexion".

✓ Résumé du cycle TCP dans tes trames :

Étape	Trame n°	Flags	Fonction
1	312	02	SYN (ouverture)
2	317	10	ACK (handshake terminé)
← END	15514	11	FIN + ACK (fermeture)



partie 2

⚙️ 1. PRÉPARATION ENVIRONNEMENT (VM en NAT)

Pré-requis :

- 1 VM serveur (Ubuntu ou Windows)
- 1 VM client
- Les deux en mode **NAT** (accès internet requis)

- Wireshark installé sur **une des deux machines**
(idéalement le client)

2. INSTALLATION / ACTIVATION DES SERVICES À CAPTURER

Protocole	Action à faire
DHCP	Redémarrer l'interface réseau (ça déclenche une requête DHCP automatiquement)
DNS	Faire <code>nslookup google.com</code> ou <code>dig google.com</code>
mDNS	Installer <code>avahi-daemon</code> (Linux) et ping un <code>.local</code> ex : <code>ping raspberrypi.local</code>
SSL/TLS/HTTPS	Aller sur <code>https://www.google.com</code> via un navigateur
FTP	Installer un serveur FTP (<code>vsftpd</code> sur Linux), et s'y connecter avec <code>ftp IP</code>
SMB	Installer <code>samba</code> et se connecter à un partage avec <code>smbclient</code>
TLSv1.2	Aller sur un site https compatible avec TLS 1.2
Messages	Peut faire référence à ICMP (<code>ping</code>), ou messagerie SMTP/POP – demande à préciser

3. FILTRES WIRESHARK À UTILISER POUR CHAQUE PROTOCOLE

Protocole	Filtre Wireshark
DHCP	<code>bootp (ou dhcp)</code>
DNS	<code>dns</code>
mDNS	<code>mdns</code> ou <code>udp.port == 5353</code>
SSL/TLS	<code>ssl</code> ou <code>tls</code>
HTTPS	<code>tcp.port == 443</code>

Protocole	Filtre Wireshark
FTP	<code>ftp ou tcp.port == 21</code>
SMB	<code>smb ou smb2</code>
TLSv1.2	<code>tls.version == 0x0303</code>
Message (ICMP)	<code>icmp</code>

4. CAPTURE ET SAUVEGARDE

Étapes :

1. Lance Wireshark
2. Applique un filtre selon le protocole que tu veux tester
3. Réalise l'action correspondante (ex: ping, visite d'un site, FTP...)
4. Stop la capture
5. **Fichier > Enregistrer sous > .pcapng ou .pcap**

💡 Tu peux aussi :

- **Marquer les paquets intéressants** (clic droit > "Marquer")
- Puis **Exporter uniquement les paquets marqués**

C'est quoi le mDNS ?

- mDNS = **Multicast DNS**

- Permet de résoudre des noms dans un réseau local **sans serveur DNS central**
- Utilisé par exemple par **AirPlay, Chromecast, Bonjour (Apple), etc.**
- Fonctionne sur le port **UDP 5353**
- Ex : `raspberrypi.local` → résolution via mDNS

50	102.496272918	VMware_c0:00:08	Broadcast	ARP	60	Who has 192.168.64.2? Tell 192.168.64.1
51	128.248144159	192.168.64.154	178.32.23.17	NTP	90	NTP Version 4, client
52	128.280155296	178.32.23.17	192.168.64.154	NTP	90	NTP Version 4, server
53	133.324426630	VMware_ab:25:a2	VMware_f1:75:d9	ARP	60	Who has 192.168.64.2? Tell 192.168.64.154
54	133.324427031	VMware_f1:75:d9	VMware_ab:25:a2	ARP	60	192.168.64.2 is at 00:50:56:f1:75:d9

Frame 53: 60 bytes on wire (480 bits), 60 by Ethernet II, Src: VMware_ab:25:a2 (00:0c:29:00:00:00), Dst: VMware_f1:75:d9 (00:0c:29:00:00:00), Protocol: ARP (0x0806), Length: 60

```

root@debian:/home/noa# nslookup google.com
Server: 192.168.64.2#53(192.168.64.2) (UDP)
When: Sun May 18 23:32:05 CEST 2025
MSG SIZE rcvd: 55

Non-authoritative answer:
Name:   google.com
Address: 142.251.37.206
Name:   google.com
Address: 2a00:1450:4006:812::200e

root@debian:/home/noa# dig google.com

;<<< Dig 9.18.33-1-deb12u2-Debian <<<> google.com
;; global options: <cmd>
;; got answer:
;; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 38543
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: 0, MBZ: 0x0000, udp: 4096
;; QUESTION SECTION:
;; google.com.
;; ANSWER SECTION:
google.com. 5 IN A 142.251.37.206

;; Query time: 51 msec
;; SERVER: 192.168.64.2#53(192.168.64.2) (UDP)
;; WHEN: Sun May 18 23:33:08 CEST 2025
;; MSG SIZE rcvd: 55

```

Echier Editur Vue Aller Capture Analyser Statistiques Telephone Wireless Outils Aide

No.	Time	Source	Destination	Protocol	Length	Info
1	15.12.331151841	192.168.64.152	192.168.64.2	DNS	70	Standard query 9490f6 A
2	15.12.331151841	192.168.64.2	192.168.64.152	DNS	86	Standard query response 1
3	17.12.331794280	192.168.64.152	192.168.64.2	DNS	70	Standard query 0xa5ac AA
4	15.12.338767146	192.168.64.2	192.168.64.152	DNS	90	Standard query response 1
5	33.14.961066576	192.168.64.152	192.168.64.2	DNS	93	Standard query 0x968f A
6	34.15.008892945	192.168.64.2	192.168.64.152	DNS	97	Standard query response 1

Frame 14: 70 bytes on wire (560 bits), 70 by Ethernet II, Src: VMware_43:50:d3 (00:0c:29:00:00:00), Dst: VMware_43:50:d3 (00:0c:29:00:00:00), Protocol: Internet Protocol Version 4, Src: 192.168.64.152, Dst: 192.168.64.2, Length: 60

Interprétation des paquets capturés - FTP avec et sans TLS

Échanges FTP sans TLS/SSL (FTP standard)

Lorsqu'on capture des paquets FTP standard (sans chiffrement), on observe les caractéristiques suivantes:

1. **Lisibilité complète des commandes et réponses:**

- Toutes les commandes FTP (USER, PASS, LIST, RETR, STOR, etc.) sont visibles en texte clair
- Toutes les réponses du serveur sont également visibles

2. **Informations d'authentification exposées:**

- La commande `USER` révèle le nom d'utilisateur
- La commande `PASS` expose le mot de passe en texte clair

3. **Visibilité du contenu transféré:**

- Le contenu des fichiers transférés est visible dans les paquets de données
- Les noms des fichiers et dossiers sont lisibles lors des commandes LIST, RETR, STOR

4. **Exemple de séquence typiquement visible:**

```
Client → Serveur: USER username
Serveur → Client: 331 Password required
Client → Serveur: PASS mypassword123
Serveur → Client: 230 Login successful
```

Récupération de données sensibles: Il est absolument possible (et trivial) de récupérer les informations d'authentification et tout le contenu transféré en écoutant simplement le trafic FTP non chiffré. Un attaquant effectuant une attaque de type "man-in-the-middle" peut facilement intercepter ces informations.

Échanges FTP avec TLS/SSL (FTPS)

Lorsqu'on capture des paquets FTPS (FTP sécurisé avec TLS/SSL), on constate:

1. Négociation TLS/SSL visible:

- Échange initial pour établir la connexion TLS (handshake)
- Échange de certificats
- Négociation des algorithmes de chiffrement

2. Données chiffrées:

- Après la négociation TLS, tous les échanges sont chiffrés
- Les paquets contiennent des données qui apparaissent comme aléatoires

3. Impossibilité de lire le contenu:

- Les informations d'authentification (USER/PASS) sont chiffrées
- Les commandes FTP sont chiffrées

- Le contenu des fichiers transférés est chiffré

4. **Structure typique des paquets:**

- Les paquets contiennent des en-têtes TCP/IP lisibles
- Le payload (contenu) est chiffré et apparaît comme des données aléatoires
- Les enregistrements TLS sont visibles mais leur contenu est illisible

Récupération de données sensibles: Avec FTP sur TLS (FTPS), il n'est pas possible de récupérer les données sensibles comme les mots de passe ou le contenu des fichiers par simple écoute du trafic. Le chiffrement TLS protège efficacement ces informations contre l'interception passive.

Différences majeures entre FTP et FTPS

1. **Sécurité:**

- FTP: Aucune protection, toutes les données sont exposées
- FTPS: Protection par chiffrement, données non intelligibles

2. **Métadonnées:**

- Dans les deux cas, certaines métadonnées comme les adresses IP, ports, timing et volume de données restent visibles

3. Impact pour un attaquant:

- FTP: Peut facilement obtenir des informations d'identification et intercepter/modifier des données
- FTPS: Limité à observer les métadonnées du trafic sans pouvoir accéder au contenu

En conclusion, FTP sans TLS/SSL représente un risque de sécurité majeur car toutes les données, y compris les informations d'authentification et le contenu des fichiers, circulent en clair sur le réseau. FTPS résout ce problème en chiffrant l'ensemble des échanges, rendant impossible la lecture des données sensibles par simple interception du trafic.

```
0 packets captured
root@debian:/home/noa# sudo tshark -i any -Y "ftp.request.command == \"USER\" || ftp.request.command =
= \"PASS\"" -T fields -e ftp.request.command -e ftp.request.arg
Running as user "root" and group "root". This could be dangerous.
Capturing on 'any'
** (tshark:5804) 00:29:52.406932 [Main MESSAGE] -- Capture started.
** (tshark:5804) 00:29:52.407017 [Main MESSAGE] -- File: "/tmp/wireshark_anyAHGU62.pcapng"
USER      r
PASS      r*
^Ctshark:
2 packets captured
root@debian:/home/noa#
```

```

29 6.960002141 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=3393 Win=255 Len=0
30 7.426080598 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
31 7.473187518 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=3629 Win=255 Len=0
32 7.938380008 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
33 7.986313105 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=3865 Win=254 Len=0
34 8.449915111 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
35 8.497694040 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=4101 Win=253 Len=0
36 8.962291303 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
37 9.010824728 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=4337 Win=252 Len=0
38 9.473897896 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
39 9.523488923 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=4573 Win=251 Len=0
40 9.986337855 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
41 10.035533014 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=4809 Win=250 Len=0
42 10.498695142 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
43 10.548548831 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=5045 Win=255 Len=0
44 11.010815016 192.168.64.152 → 192.168.64.1 SSH 186 Server: Encrypted packet (len=132)
45 11.011009369 192.168.64.152 → 192.168.64.1 SSH 202 Server: Encrypted packet (len=148)
46 11.011449482 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=5325 Win=254 Len=0
47 11.517407077 192.168.64.152 → 192.168.64.1 SSH 386 Server: Encrypted packet (len=332)
48 11.558110865 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=5657 Win=253 Len=0
49 12.034253932 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
50 12.086428111 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=5893 Win=252 Len=0
51 12.546041802 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
52 12.600002563 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=6129 Win=251 Len=0
53 13.058191914 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
54 13.112762604 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=6365 Win=250 Len=0
55 13.570447663 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
56 13.625727970 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=6601 Win=255 Len=0
57 14.082665889 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
58 14.137905832 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=6837 Win=255 Len=0
59 14.594217771 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
60 14.635166023 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=7073 Win=254 Len=0
61 15.106248166 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
62 15.147935362 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=7309 Win=253 Len=0
63 15.664303832 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
64 15.707796374 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=7545 Win=252 Len=0
65 16.414722327 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
66 16.471816266 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=7781 Win=251 Len=0
67 16.976979249 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
68 17.023743058 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=8017 Win=250 Len=0
69 17.725731283 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
70 17.768533514 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=8253 Win=255 Len=0
71 18.242984849 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
72 18.297020869 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=8489 Win=255 Len=0
^C 73 18.754101466 192.168.64.152 → 192.168.64.1 SSH 290 Server: Encrypted packet (len=236)
74 18.794624084 192.168.64.1 → 192.168.64.152 TCP 60 50068 → 22 [ACK] Seq=1 Ack=8725 Win=254 Len=0
tshark:
74 packets captured

```

Explication des options TShark et méthodes de filtrage

Options communes de TShark et leur signification

Options de base pour la capture

- `-i <interface>` : Spécifie l'interface réseau à utiliser pour la capture

```
sudo tshark -i eth0    # Capture sur l'interface  
eth0  
sudo tshark -i any     # Capture sur toutes les  
interfaces
```

- `-f "<filtre pcap>"` : Applique un filtre de capture Berkeley Packet Filter (BPF)

```
sudo tshark -i any -f "tcp port 80"    # Capture  
uniquement le trafic HTTP
```

- `-c <nombre>` : Limite le nombre de paquets capturés

```
sudo tshark -i any -c 100    # Capture seulement  
100 paquets puis s'arrête
```

- `-a duration:<secondes>` : Arrête la capture après un délai spécifié

```
sudo tshark -i any -a duration:30    # Capture  
pendant 30 secondes
```

- `-w <fichier>` : Écrit les paquets bruts dans un fichier pcap

```
sudo tshark -i any -w capture.pcap #  
Sauvegarde la capture dans un fichier
```

Options de filtrage et d'affichage

- `-Y "<filtre d'affichage>"` : Applique un filtre d'affichage Wireshark

```
sudo tshark -i any -Y "http" # N'affiche que  
les paquets HTTP  
sudo tshark -i any -Y "dns &&  
ip.src==192.168.1.1" # Filtre complexe
```

- `-T <format>` : Définit le format de sortie

```
sudo tshark -r capture.pcap -T fields #  
Format en colonnes  
sudo tshark -r capture.pcap -T json #  
Format JSON  
sudo tshark -r capture.pcap -T pdml #  
Format XML détaillé
```

- `-e <champ>` : Spécifie les champs à afficher (avec `-T fields`)


```
sudo tshark -r capture.pcap -T fields -e ip.src  
-e ip.dst -e http.request.method
```

- **-V** : Mode verbeux, affiche tous les détails des paquets

```
sudo tshark -i any -Y "http" -V      # Affiche les  
détails complets des paquets HTTP
```

- **-O <protocole>** : Affiche les détails d'un protocole spécifique

```
bash  
sudo tshark -i any -O http      # Affiche les  
détails du protocole HTTP
```



Méthodes de filtrage des captures

1. Filtrage direct avec les options de TShark

TShark propose deux principaux types de filtres:

1. **Filtres de capture** (avec l'option **-f**):
 - Appliqués au niveau du moteur de capture (libpcap)
 - Syntaxe Berkeley Packet Filter (BPF)

- Très efficaces car ils filtrent avant que les paquets ne soient traités

```
sudo tshark -i any -f "tcp port 80 or tcp port 443"
sudo tshark -i any -f "host 192.168.1.1"
```

2. Filtres d'affichage (avec l'option `-Y`):

- Utilisent la syntaxe des filtres Wireshark
- Plus puissants et flexibles, permettent de filtrer sur n'importe quel champ
- Appliqués après la capture, donc moins efficaces en termes de performances

```
sudo tshark -i any -Y "http.request.method == GET"
sudo tshark -i any -Y "dns.qry.name contains google"
```

2. Filtrage par redirection du résultat

Vous pouvez également capturer tous les paquets et filtrer la sortie en utilisant des outils Unix standards:

1. Pipeline avec grep:

```
sudo tshark -i any | grep "GET"    # Filtre les  
lignes contenant "GET"
```

2. Redirection vers un fichier puis traitement:

```
sudo tshark -i any -w capture.pcap    # Capture  
dans un fichier  
# Plus tard, analyser le fichier avec filtres  
tshark -r capture.pcap -Y "http" >  
http_traffic.txt
```

3. Utilisation de awk, sed ou autres outils Unix:

```
sudo tshark -i any -T fields -e ip.src -e ip.dst  
| awk '$1=="192.168.1.1"'
```

4. Combinaison de filtres TShark et d'outils Unix:

```
sudo tshark -i any -Y "dns" -T fields -e  
dns.qry.name | sort | uniq -c
```

Chaque méthode a ses avantages:

- Les filtres de capture (`-f`) sont les plus efficaces pour réduire la charge système

- Les filtres d'affichage (-Y) offrent plus de précision et de flexibilité
- Les redirections avec outils Unix permettent des analyses complexes et des post-traitements

Le choix de la méthode dépend de vos besoins spécifiques: performance, précision du filtrage, ou besoins d'analyse.