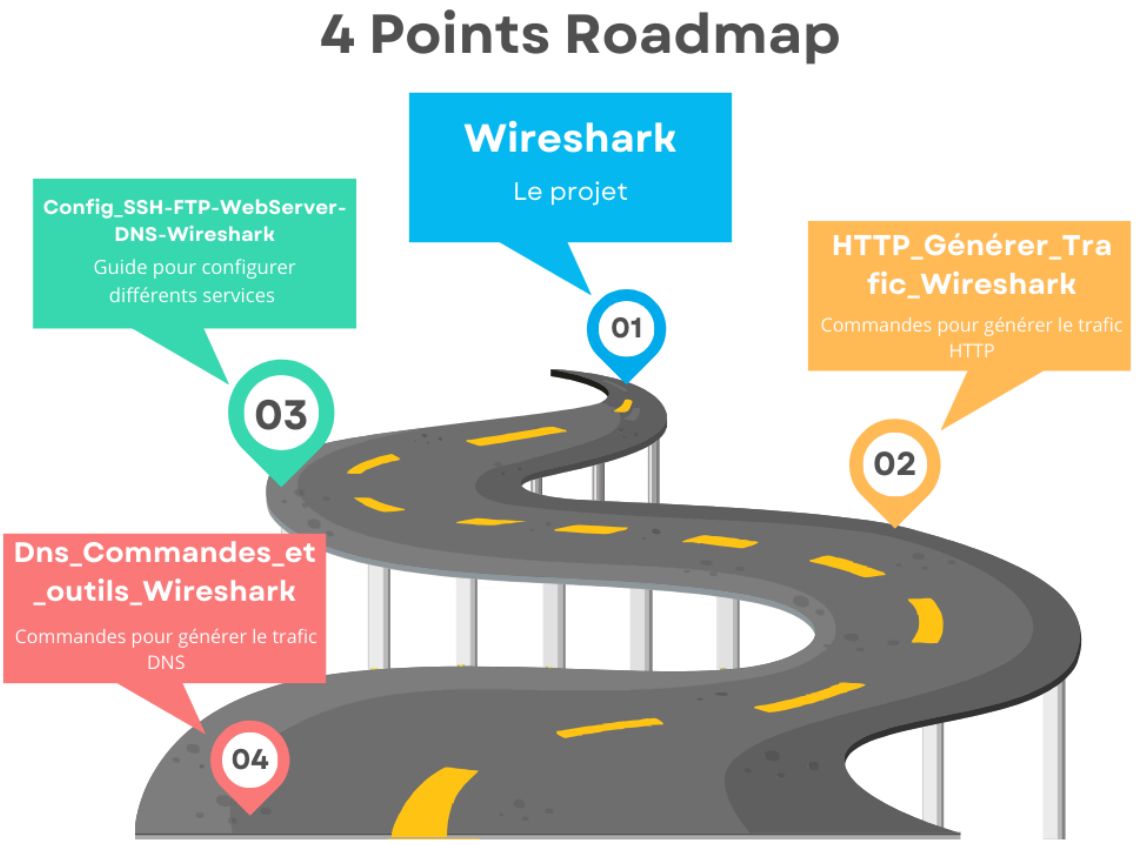
Wireshark



Partie 1

Wireshark est un puissant outil d'analyse de protocoles réseau qui permet de capturer et d'examiner le trafic réseau, permettant de visualiser les différentes couches du modèle OSI pour chaque trame capturée. Cela s'appelle la "désencapsulation" des trames.

Quelle est la différence entre une trame et un paquet ? Qu'est-ce que le format pcap/pcapng ?

La différence entre une trame et un paquet réside dans les couches du modèle OSI auxquelles ils appartiennent.

Quelle est la différence entre une trame et un paquet ?

Trame

Une trame fait référence aux données de la couche liaison de données (couche 2) du modèle OSI. C'est l'unité de données transférée sur un support physique comme Ethernet ou Wi-Fi. Une trame contient:

- L'en-tête de la couche liaison (adresses MAC source et destination)
- Les données de la couche réseau (paquet IP)
- La séquence de contrôle de trame

Paquet

Un paquet fait référence aux données de la couche réseau (couche 3) du modèle OSI, généralement IP. Un paquet IP contient:

- L'en-tête IP (adresses IP source et destination)
- Les données de la couche transport (segment TCP ou datagramme UDP)

Donc une trame encapsule un ou plusieurs paquets IP, avec des en-têtes supplémentaires pour le support physique.

Qu’est-ce que le format pcap/pcapng ?

PCAP et PCAPNG sont des formats de fichier utilisés pour stocker des captures de trafic réseau, contenant à la fois les trames et les paquets.

PCAP

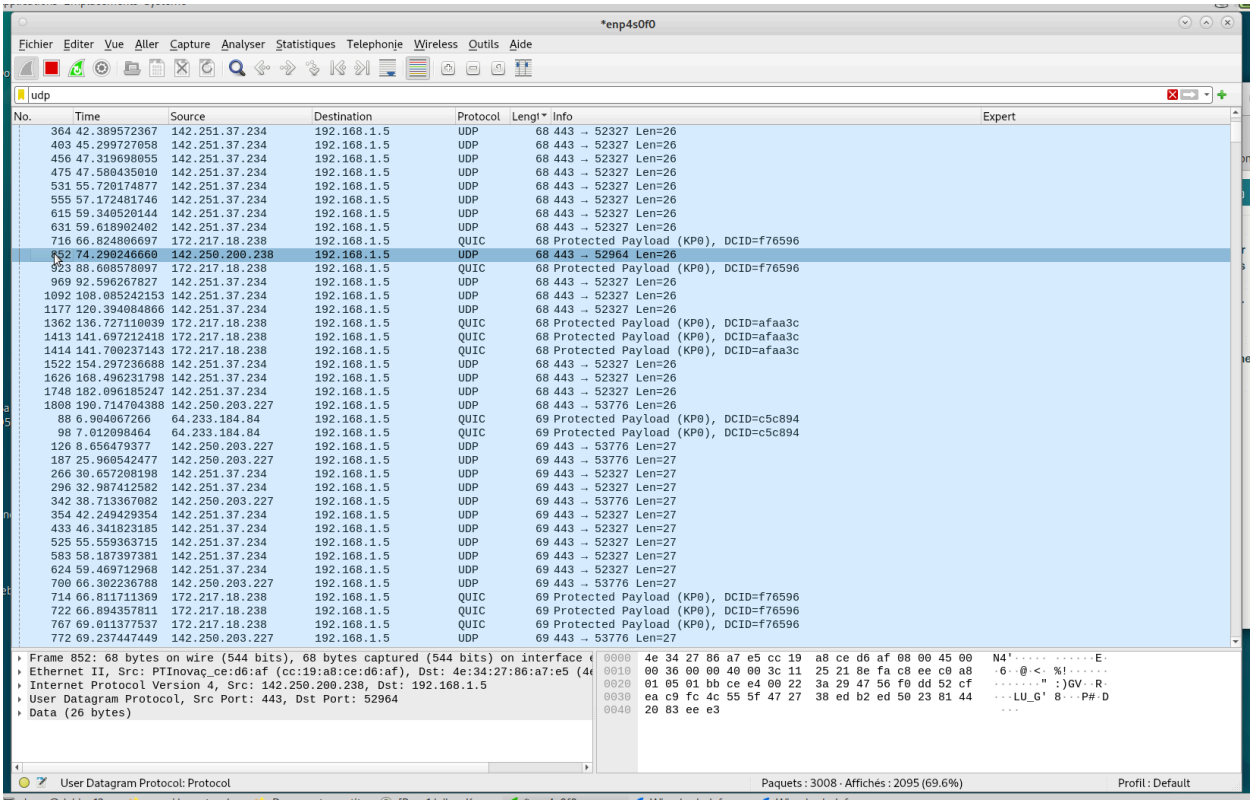
- Format historique créé en 1987 pour tcpdump
- Stocke les trames avec un horodatage et la longueur
- Limité à une résolution de l'horodatage en microsecondes
- Ne peut stocker qu'un seul type de trame par fichier

PCAPNG

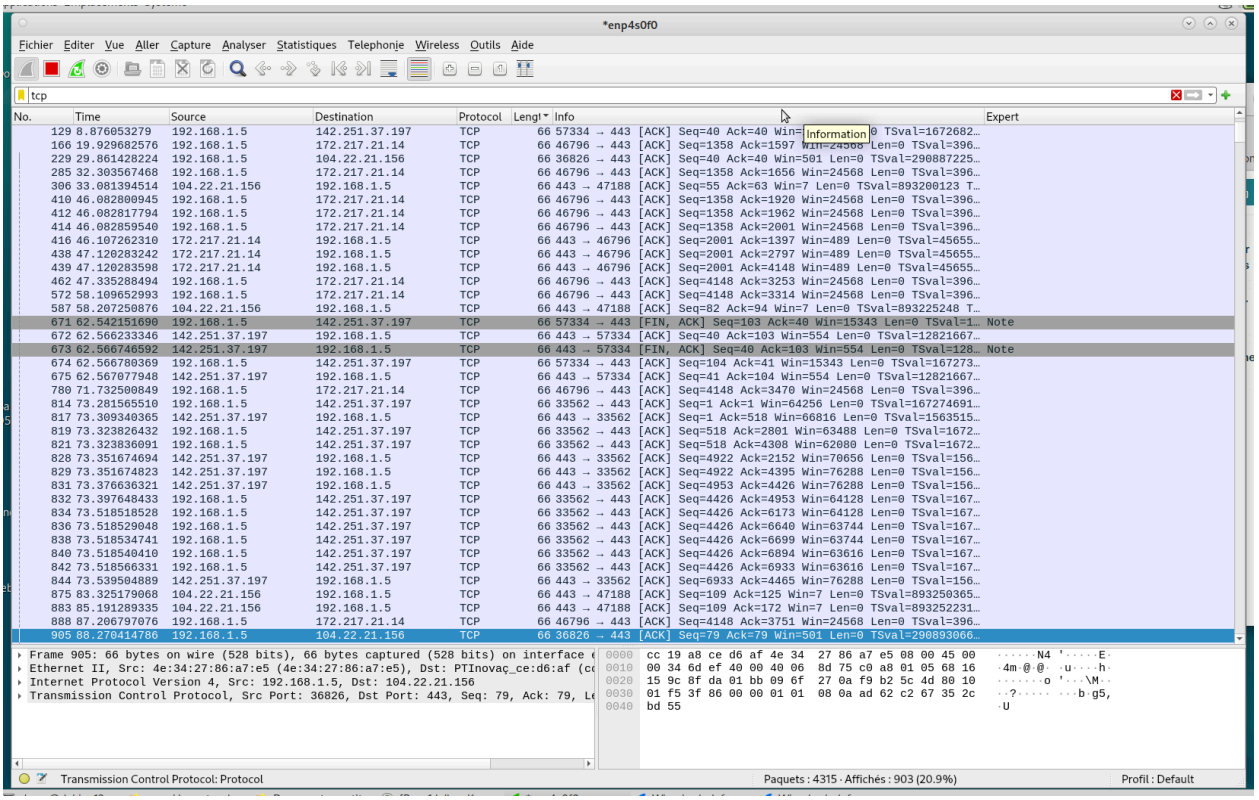
- Format "nouvelle génération" plus récent et extensible
- Supporte plusieurs types de trames dans un même fichier
- Résolution de l'horodatage configurable (nanosecondes possibles)
- Permet de stocker des métadonnées supplémentaires (commentaires, statistiques, etc.)
- Pris en charge par Wireshark, tcpdump, etc.

  En résumé, PCAPNG est un format plus moderne et flexible que PCAP pour stocker des captures réseau, tout en conservant la compatibilité avec les outils existants

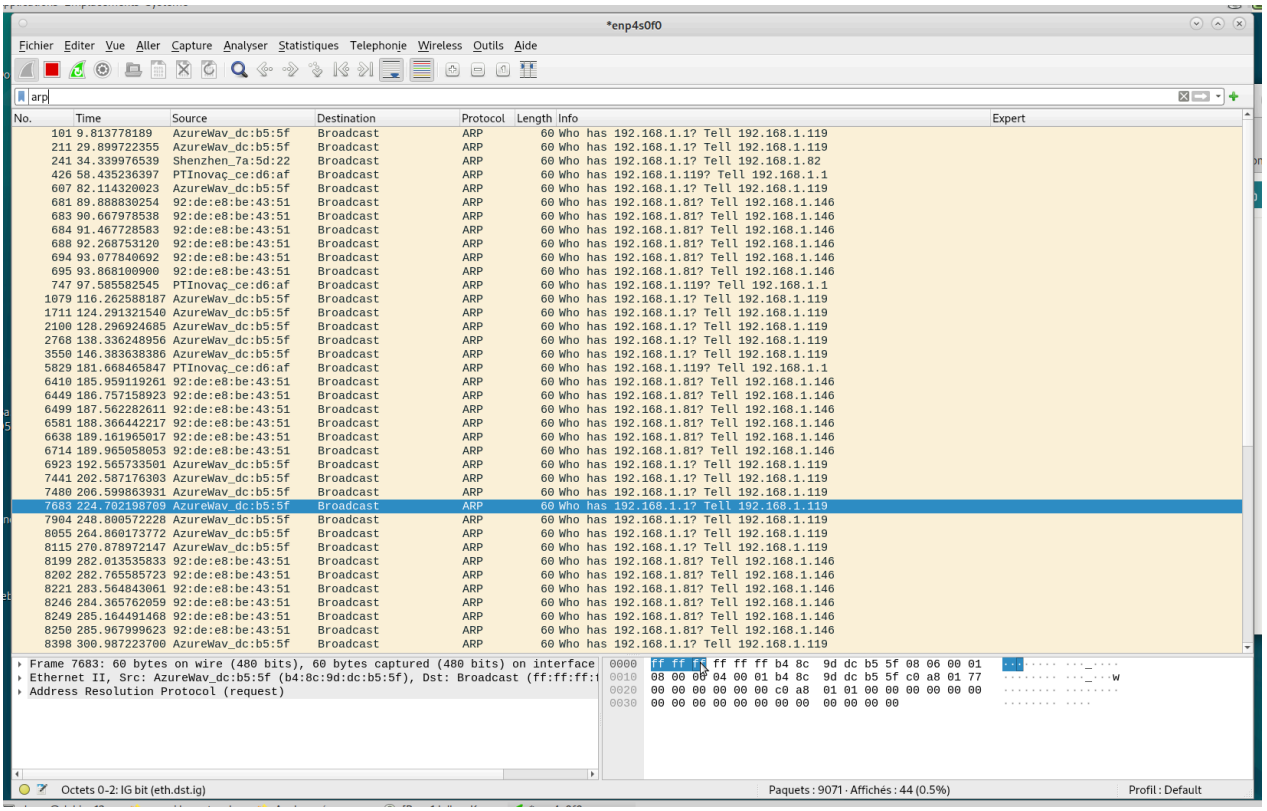
UDP


*enp4s0f0
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
udp
No. Time Source Destination Protocol Lengt Info Expert
364 42.389572367 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
403 45.299727058 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
456 47.319698055 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
475 47.580435010 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
531 55.720174877 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
555 57.172481746 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
615 59.340520144 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
631 59.618902402 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
716 66.824806697 172.217.18.238 192.168.1.5 QUIC 68 Protected Payload (KP0), DCID=f76596
852 74.290246660 142.250.200.238 192.168.1.5 UDP 68 443 → 52964 Len=26
923 88.608578097 172.217.18.238 192.168.1.5 QUIC 68 Protected Payload (KP0), DCID=f76596
969 92.596267827 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
1092 108.085242153 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
1177 120.394084866 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
1362 136.727110039 172.217.18.238 192.168.1.5 QUIC 68 Protected Payload (KP0), DCID=afaa3c
1413 141.697212418 172.217.18.238 192.168.1.5 QUIC 68 Protected Payload (KP0), DCID=afaa3c
1414 141.700237143 172.217.18.238 192.168.1.5 QUIC 68 Protected Payload (KP0), DCID=afaa3c
1522 154.297236688 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
1626 168.496231798 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
1748 182.096185247 142.251.37.234 192.168.1.5 UDP 68 443 → 52327 Len=26
1808 190.714704388 142.250.203.227 192.168.1.5 UDP 68 443 → 53776 Len=26
88 6.904067266 64.233.184.84 192.168.1.5 QUIC 69 Protected Payload (KP0), DCID=c5c894
98 7.012098464 64.233.184.84 192.168.1.5 QUIC 69 Protected Payload (KP0), DCID=c5c894
126 8.656479377 142.250.203.227 192.168.1.5 UDP 69 443 → 53776 Len=27
187 25.960542477 142.250.203.227 192.168.1.5 UDP 69 443 → 53776 Len=27
266 30.657208198 142.251.37.234 192.168.1.5 UDP 69 443 → 52327 Len=27
296 32.987412582 142.251.37.234 192.168.1.5 UDP 69 443 → 52327 Len=27
342 38.713367082 142.250.203.227 192.168.1.5 UDP 69 443 → 53776 Len=27
354 42.249429354 142.251.37.234 192.168.1.5 UDP 69 443 → 52327 Len=27
433 46.341823185 142.251.37.234 192.168.1.5 UDP 69 443 → 52327 Len=27
525 55.559363715 142.251.37.234 192.168.1.5 UDP 69 443 → 52327 Len=27
583 58.187397381 142.251.37.234 192.168.1.5 UDP 69 443 → 52327 Len=27
624 59.469712968 142.251.37.234 192.168.1.5 UDP 69 443 → 52327 Len=27
700 66.302236788 142.250.203.227 192.168.1.5 UDP 69 443 → 53776 Len=27
714 66.811711369 172.217.18.238 192.168.1.5 QUIC 69 Protected Payload (KP0), DCID=f76596
722 66.894357811 172.217.18.238 192.168.1.5 QUIC 69 Protected Payload (KP0), DCID=f76596
767 69.011377537 172.217.18.238 192.168.1.5 QUIC 69 Protected Payload (KP0), DCID=f76596
772 69.237447449 142.250.203.227 192.168.1.5 UDP 69 443 → 53776 Len=27
Frame 852: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface
Ethernet II, Src: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af), Dst: 4e:34:27:86:a7:e5 (4
Internet Protocol Version 4, Src: 142.250.200.238, Dst: 192.168.1.5
User Datagram Protocol, Src Port: 443, Dst Port: 52964
Data (26 bytes)
0000 4e 34 27 86 a7 e5 cc 19 a8 ce d6 af 08 00 45 00
0010 00 36 00 00 40 00 3c 11 25 21 8e fa c8 ee c0 a8
0020 01 05 01 bb ce e4 00 22 3a 29 47 56 f0 dd 52 cf
0030 ea c9 fc 4c 55 5f 47 27 38 ed b2 ed 50 23 81 44
0040 20 83 ee e3
User Datagram Protocol: Protocol
Paquets : 3008 · Affichés : 2095 (69.6%)
Profil : Default

TCP


*enp4s0f0
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
tcp
No. Time Source Destination Protocol Lengt Info Expert
129 8.876053279 192.168.1.5 142.251.37.197 TCP 66 57334 → 443 [ACK] Seq=40 Ack=40 Win= Information 0 TSval=1672682…
166 19.929682576 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=1358 Ack=1597 Win=24568 Len=0 TSval=396…
229 29.861428224 192.168.1.5 104.22.21.156 TCP 66 36826 → 443 [ACK] Seq=40 Ack=40 Win=501 Len=0 TSval=290887225…
285 32.303567468 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=1358 Ack=1656 Win=24568 Len=0 TSval=396…
306 33.081394514 104.22.21.156 192.168.1.5 TCP 66 443 → 47188 [ACK] Seq=55 Ack=63 Win=7 Len=0 TSval=893200123 T…
410 46.082800945 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=1358 Ack=1920 Win=24568 Len=0 TSval=396…
412 46.082817794 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=1358 Ack=1962 Win=24568 Len=0 TSval=396…
414 46.082859540 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=1358 Ack=2001 Win=24568 Len=0 TSval=396…
416 46.107262310 172.217.21.14 192.168.1.5 TCP 66 443 → 46796 [ACK] Seq=2001 Ack=1397 Win=489 Len=0 TSval=45655…
438 47.120283242 172.217.21.14 192.168.1.5 TCP 66 443 → 46796 [ACK] Seq=2001 Ack=2797 Win=489 Len=0 TSval=45655…
439 47.120283598 172.217.21.14 192.168.1.5 TCP 66 443 → 46796 [ACK] Seq=2001 Ack=4148 Win=489 Len=0 TSval=45655…
462 47.335288494 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=4148 Ack=3253 Win=24568 Len=0 TSval=396…
572 58.109652993 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=4148 Ack=3314 Win=24568 Len=0 TSval=396…
587 58.207250876 104.22.21.156 192.168.1.5 TCP 66 443 → 47188 [ACK] Seq=82 Ack=94 Win=7 Len=0 TSval=893225248 T…
671 62.542151690 192.168.1.5 142.251.37.197 TCP 66 57334 → 443 [FIN, ACK] Seq=103 Ack=40 Win=15343 Len=0 TSval=1… Note
672 62.566233346 142.251.37.197 192.168.1.5 TCP 66 443 → 57334 [ACK] Seq=40 Ack=103 Win=554 Len=0 TSval=12821667…
673 62.566746592 142.251.37.197 192.168.1.5 TCP 66 443 → 57334 [FIN, ACK] Seq=40 Ack=103 Win=554 Len=0 TSval=128… Note
674 62.566780369 192.168.1.5 142.251.37.197 TCP 66 57334 → 443 [ACK] Seq=104 Ack=41 Win=15343 Len=0 TSval=167273…
675 62.567077948 142.251.37.197 192.168.1.5 TCP 66 443 → 57334 [ACK] Seq=41 Ack=104 Win=554 Len=0 TSval=12821667…
780 71.732500849 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=4148 Ack=3470 Win=24568 Len=0 TSval=396…
814 73.281565510 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=167274691…
817 73.309340365 142.251.37.197 192.168.1.5 TCP 66 443 → 33562 [ACK] Seq=1 Ack=518 Win=66816 Len=0 TSval=1563515…
819 73.323826432 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=518 Ack=2801 Win=63488 Len=0 TSval=1672…
821 73.323836091 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=518 Ack=4308 Win=62080 Len=0 TSval=1672…
828 73.351674694 142.251.37.197 192.168.1.5 TCP 66 443 → 33562 [ACK] Seq=4922 Ack=2152 Win=70656 Len=0 TSval=156…
829 73.351674823 142.251.37.197 192.168.1.5 TCP 66 443 → 33562 [ACK] Seq=4922 Ack=4395 Win=76288 Len=0 TSval=156…
831 73.376636321 142.251.37.197 192.168.1.5 TCP 66 443 → 33562 [ACK] Seq=4953 Ack=4426 Win=76288 Len=0 TSval=156…
832 73.397648433 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=4426 Ack=4953 Win=64128 Len=0 TSval=167…
834 73.518518528 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=4426 Ack=6173 Win=64128 Len=0 TSval=167…
836 73.518529048 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=4426 Ack=6640 Win=63744 Len=0 TSval=167…
838 73.518534741 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=4426 Ack=6699 Win=63744 Len=0 TSval=167…
840 73.518540410 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=4426 Ack=6894 Win=63616 Len=0 TSval=167…
842 73.518566331 192.168.1.5 142.251.37.197 TCP 66 33562 → 443 [ACK] Seq=4426 Ack=6933 Win=63616 Len=0 TSval=167…
844 73.539504889 142.251.37.197 192.168.1.5 TCP 66 443 → 33562 [ACK] Seq=6933 Ack=4465 Win=76288 Len=0 TSval=156…
875 83.325179068 104.22.21.156 192.168.1.5 TCP 66 443 → 47188 [ACK] Seq=109 Ack=125 Win=7 Len=0 TSval=893250365…
883 85.191289335 104.22.21.156 192.168.1.5 TCP 66 443 → 47188 [ACK] Seq=109 Ack=172 Win=7 Len=0 TSval=893252231…
888 87.206797076 192.168.1.5 172.217.21.14 TCP 66 46796 → 443 [ACK] Seq=4148 Ack=3751 Win=24568 Len=0 TSval=396…
905 88.270414786 192.168.1.5 104.22.21.156 TCP 66 36826 → 443 [ACK] Seq=79 Ack=79 Win=501 Len=0 TSval=290893066…
Frame 905: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface
Ethernet II, Src: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5), Dst: PTInovaç_ce:d6:af (cc
Internet Protocol Version 4, Src: 192.168.1.5, Dst: 104.22.21.156
Transmission Control Protocol, Src Port: 36826, Dst Port: 443, Seq: 79, Ack: 79, Le
0000 cc 19 a8 ce d6 af 4e 34 27 86 a7 e5 08 00 45 00
0010 00 34 6d ef 40 00 40 06 8d 75 c0 a8 01 05 68 16
0020 15 9c 8f da 01 bb 09 6f 27 0a f9 b2 5c 4d 80 10
0030 01 f5 3f 86 00 00 01 01 08 0a ad 62 c2 67 35 2c
0040 bd 55
Transmission Control Protocol: Protocol
Paquets : 4315 · Affichés : 903 (20.9%)
Profil : Default

ARP



Quelles sont les adresses MAC sources, les IP sources et les adresses MAC sources, les IP destinations des données capturées ?

Généralement les adresses MAC source, IP source, MAC destination et IP destination que l'on peut trouver dans les données capturées avec Wireshark :

Adresse MAC source

- C'est l'adresse MAC de la carte réseau de l'appareil qui envoie les données

.

- Elle est présente dans l'en-tête de la trame Ethernet de niveau 2

Adresse IP source

- C'est l'adresse IP de l'appareil qui envoie les données
- Elle est présente dans l'en-tête IP de niveau 3, encapsulé dans la trame Ethernet

Adresse MAC destination

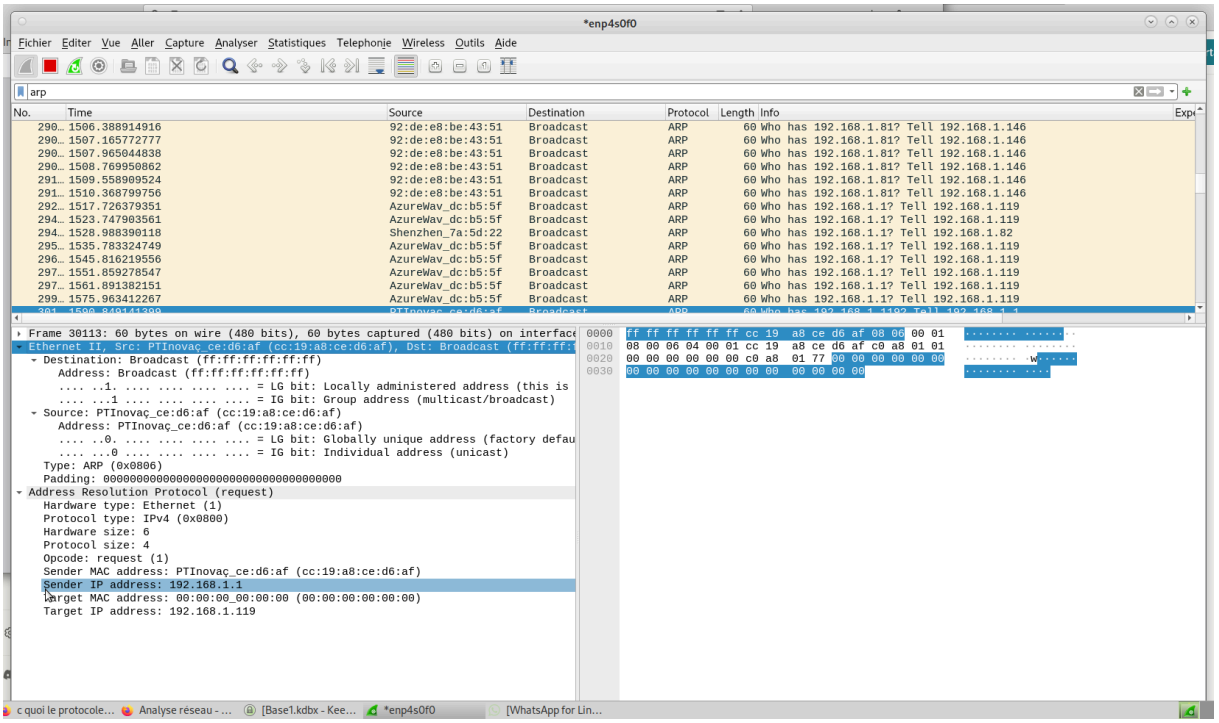
- C'est l'adresse MAC de la carte réseau de l'appareil destinataire des données
- Elle est présente dans l'en-tête de la trame Ethernet de niveau 2.
  Si l'adresse MAC de destination n'est pas connue, la valeur sera l'adresse de diffusion (broadcast) ff:ff:ff:ff:ff:ff

Adresse IP destination

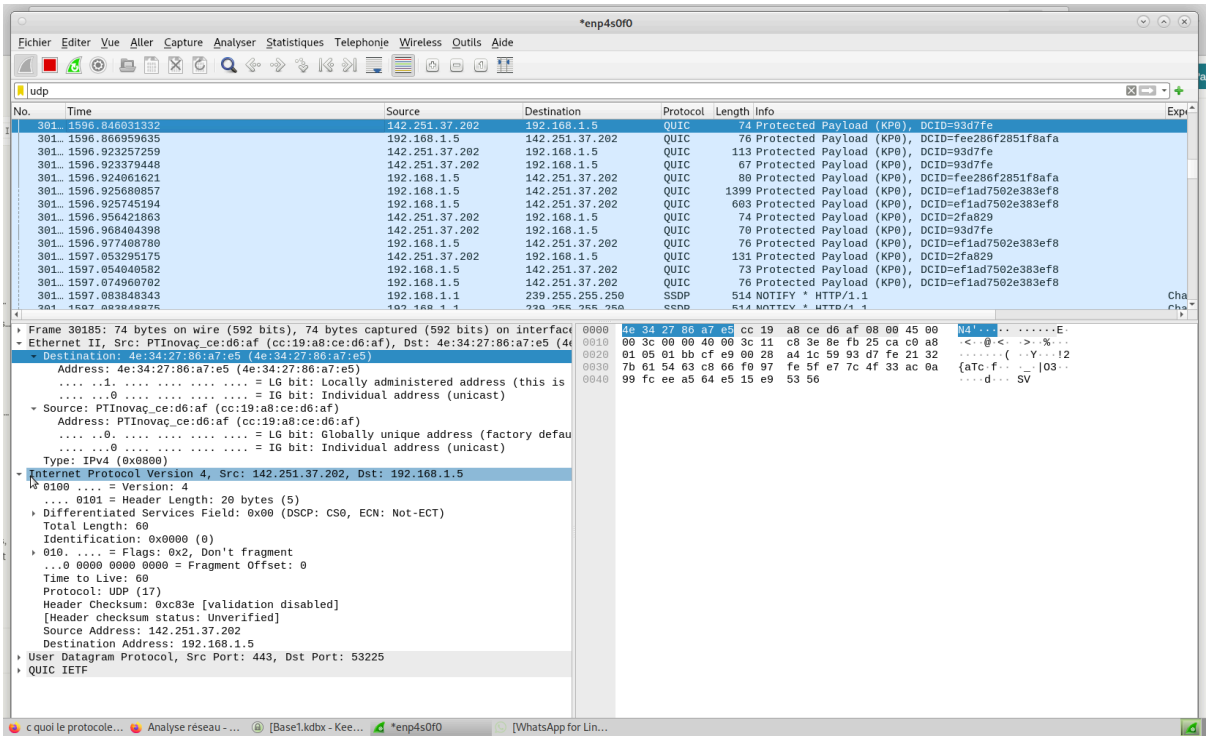
- C'est l'adresse IP de l'appareil destinataire des données
- Elle est présente dans l'en-tête IP de niveau 3, encapsulé dans la trame Ethernet

Donc en résumé, dans une capture Wireshark, on peut généralement voir l'adresse MAC source et IP source de l'appareil émetteur, ainsi que l'adresse MAC destination et IP destination de l'appareil destinataire pour chaque trame.

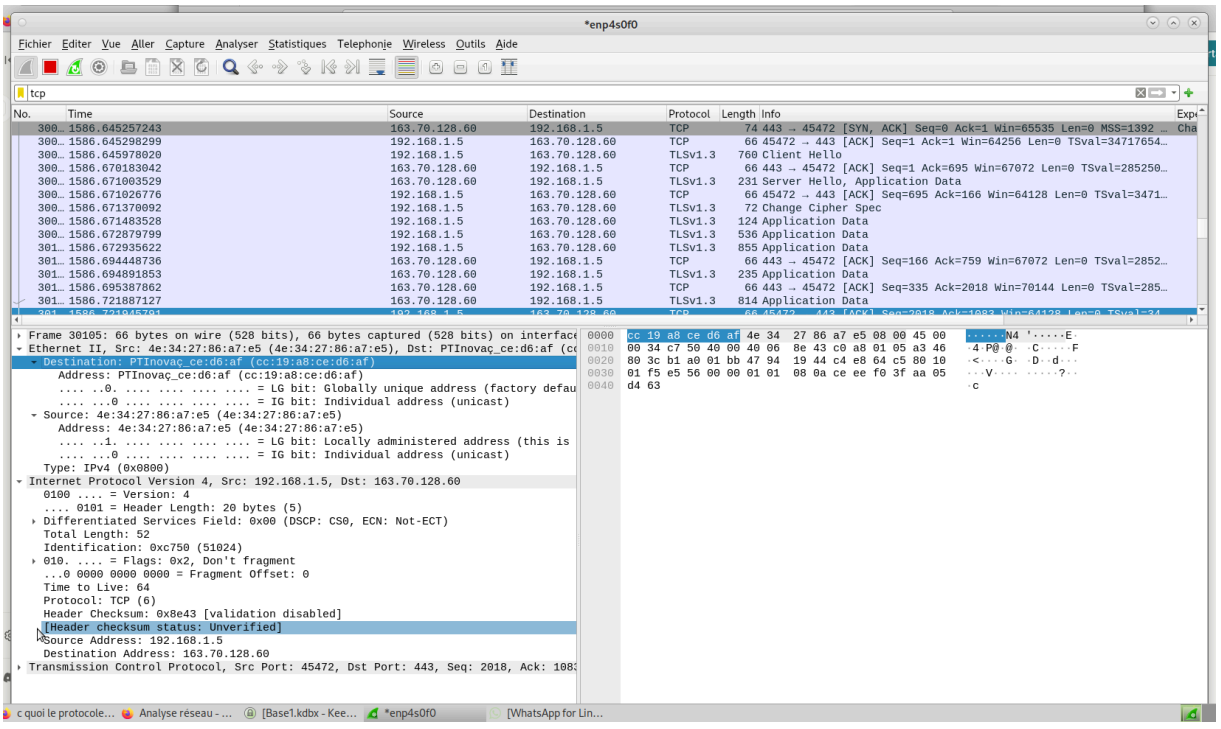
ARP



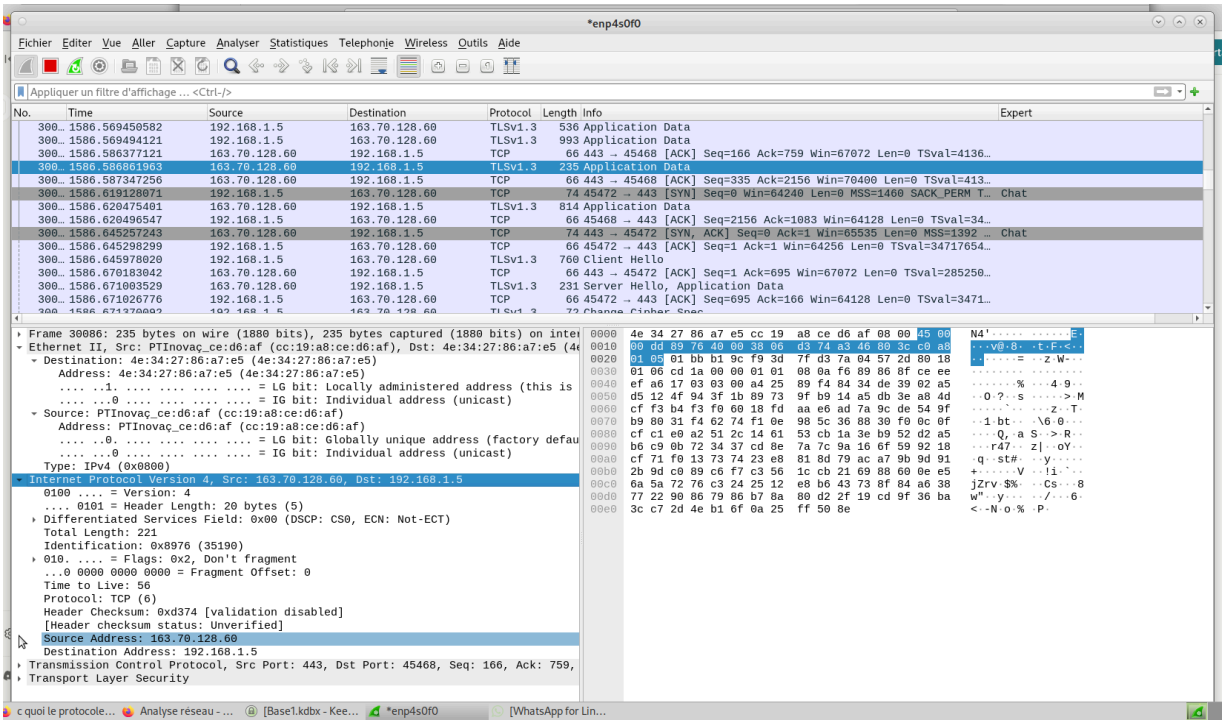
UDP


*enp4s0f0
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
udp
No. Time Source Destination Protocol Length Info
301… 1596.846031332 142.251.37.202 192.168.1.5 QUIC 74 Protected Payload (KP0), DCID=93d7fe
301… 1596.866959635 192.168.1.5 142.251.37.202 QUIC 76 Protected Payload (KP0), DCID=fee286f2851f8afa
301… 1596.923257259 142.251.37.202 192.168.1.5 QUIC 113 Protected Payload (KP0), DCID=93d7fe
301… 1596.923379448 142.251.37.202 192.168.1.5 QUIC 67 Protected Payload (KP0), DCID=93d7fe
301… 1596.924061621 192.168.1.5 142.251.37.202 QUIC 80 Protected Payload (KP0), DCID=fee286f2851f8afa
301… 1596.925680857 192.168.1.5 142.251.37.202 QUIC 1399 Protected Payload (KP0), DCID=ef1ad7502e383ef8
301… 1596.925745194 192.168.1.5 142.251.37.202 QUIC 603 Protected Payload (KP0), DCID=ef1ad7502e383ef8
301… 1596.956421863 142.251.37.202 192.168.1.5 QUIC 74 Protected Payload (KP0), DCID=2fa829
301… 1596.968404398 142.251.37.202 192.168.1.5 QUIC 70 Protected Payload (KP0), DCID=93d7fe
301… 1596.977408780 192.168.1.5 142.251.37.202 QUIC 76 Protected Payload (KP0), DCID=ef1ad7502e383ef8
301… 1597.053295175 142.251.37.202 192.168.1.5 QUIC 131 Protected Payload (KP0), DCID=2fa829
301… 1597.054040582 192.168.1.5 142.251.37.202 QUIC 73 Protected Payload (KP0), DCID=ef1ad7502e383ef8
301… 1597.074960702 192.168.1.5 142.251.37.202 QUIC 76 Protected Payload (KP0), DCID=ef1ad7502e383ef8
301… 1597.083848343 192.168.1.1 239.255.255.250 SSDP 514 NOTIFY * HTTP/1.1
Frame 30185: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface
Ethernet II, Src: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af), Dst: 4e:34:27:86:a7:e5 (4e
Destination: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
Address: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
.... ..1. .... = LG bit: Locally administered address (this is
....0 .... = IG bit: Individual address (unicast)
Source: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
Address: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
.... ..0. .... = LG bit: Globally unique address (factory defau
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 142.251.37.202, Dst: 192.168.1.5
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 60
Identification: 0x0000 (0)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 60
Protocol: UDP (17)
Header Checksum: 0xc83e [validation disabled]
[Header checksum status: Unverified]
Source Address: 142.251.37.202
Destination Address: 192.168.1.5
User Datagram Protocol, Src Port: 443, Dst Port: 53225
QUIC IETF
0000 4e 34 27 86 a7 e5 cc 19 a8 ce d6 af 08 00 45 00
0010 00 3c 00 00 40 00 3c 11 c8 3e 8e fb 25 ca c0 a8
0020 01 05 01 bb cf e9 00 28 a4 1c 59 93 d7 fe 21 32
0030 7b 61 54 63 c8 66 f0 97 fe 5f e7 7c 4f 33 ac 0a
0040 99 fc ee a5 64 e5 15 e9 53 56
c quoi le protocole… Analyse réseau - … [Base1.kdbx - Kee… *enp4s0f0 [WhatsApp for Lin…

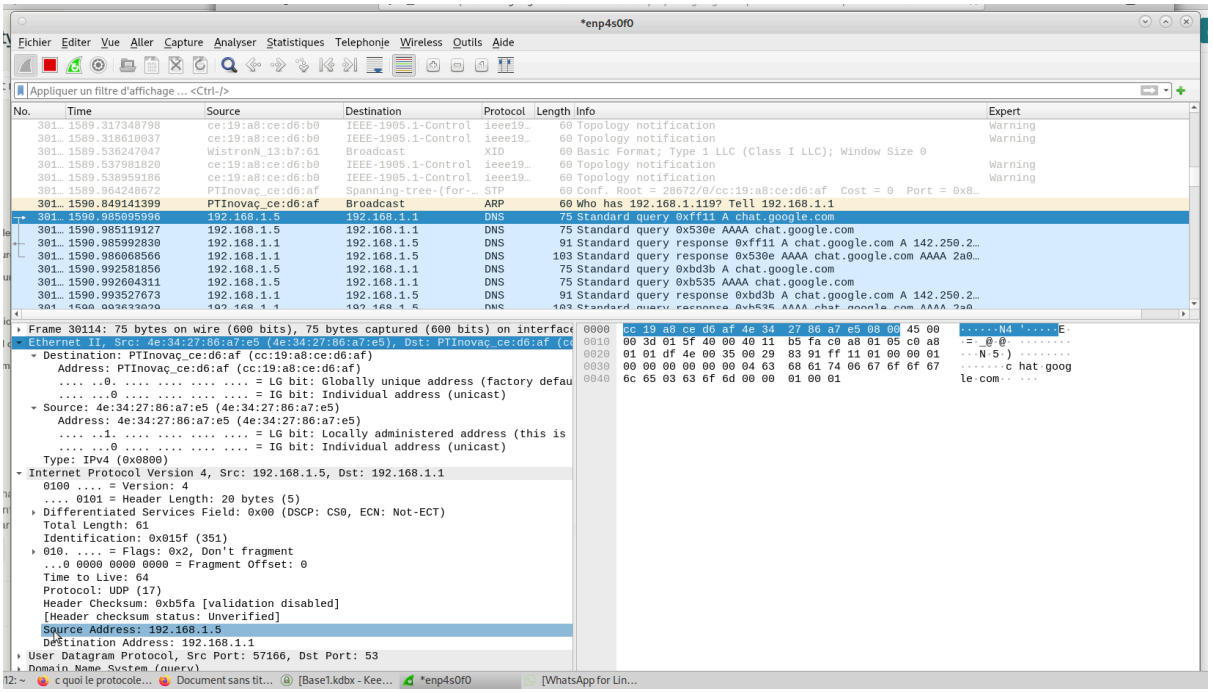
TCP


*enp4s0f0
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
tcp
No. Time Source Destination Protocol Length Info
300… 1586.645257243 163.70.128.60 192.168.1.5 TCP 74 443 → 45472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 …
300… 1586.645298299 192.168.1.5 163.70.128.60 TCP 66 45472 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=34717654…
300… 1586.645978020 192.168.1.5 163.70.128.60 TLSv1.3 760 Client Hello
300… 1586.670183042 163.70.128.60 192.168.1.5 TCP 66 443 → 45472 [ACK] Seq=1 Ack=695 Win=67072 Len=0 TSval=285250…
300… 1586.671003529 163.70.128.60 192.168.1.5 TLSv1.3 231 Server Hello, Application Data
300… 1586.671026776 192.168.1.5 163.70.128.60 TCP 66 45472 → 443 [ACK] Seq=695 Ack=166 Win=64128 Len=0 TSval=3471…
300… 1586.671370092 192.168.1.5 163.70.128.60 TLSv1.3 72 Change Cipher Spec
300… 1586.671483528 192.168.1.5 163.70.128.60 TLSv1.3 124 Application Data
300… 1586.672879799 192.168.1.5 163.70.128.60 TLSv1.3 536 Application Data
301… 1586.672935622 192.168.1.5 163.70.128.60 TLSv1.3 855 Application Data
301… 1586.694448736 163.70.128.60 192.168.1.5 TCP 66 443 → 45472 [ACK] Seq=166 Ack=759 Win=67072 Len=0 TSval=2852…
301… 1586.694891853 163.70.128.60 192.168.1.5 TLSv1.3 235 Application Data
301… 1586.695387862 192.168.1.5 163.70.128.60 TCP 66 443 → 45472 [ACK] Seq=335 Ack=2018 Win=70144 Len=0 TSval=285…
301… 1586.721887127 163.70.128.60 192.168.1.5 TLSv1.3 814 Application Data
Frame 30105: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface
Ethernet II, Src: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5), Dst: PTInovaç_ce:d6:af (cc
Destination: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
Address: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
.... ..0. .... = LG bit: Globally unique address (factory defau
....0 .... = IG bit: Individual address (unicast)
Source: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
Address: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
.... ..1. .... = LG bit: Locally administered address (this is
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.1.5, Dst: 163.70.128.60
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 52
Identification: 0xc750 (51024)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: TCP (6)
Header Checksum: 0x8e43 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.1.5
Destination Address: 163.70.128.60
Transmission Control Protocol, Src Port: 45472, Dst Port: 443, Seq: 2018, Ack: 1083
0000 cc 19 a8 ce d6 af 4e 34 27 86 a7 e5 08 00 45 00
0010 00 34 c7 50 40 00 40 06 8e 43 c0 a8 01 05 a3 46
0020 80 3c b1 a0 01 bb 47 94 19 44 c4 e8 64 c5 80 10
0030 01 f5 e5 56 00 00 01 01 08 0a ce ee f0 3f aa 05
0040 d4 63
c quoi le protocole… Analyse réseau - … [Base1.kdbx - Kee… *enp4s0f0 [WhatsApp for Lin…

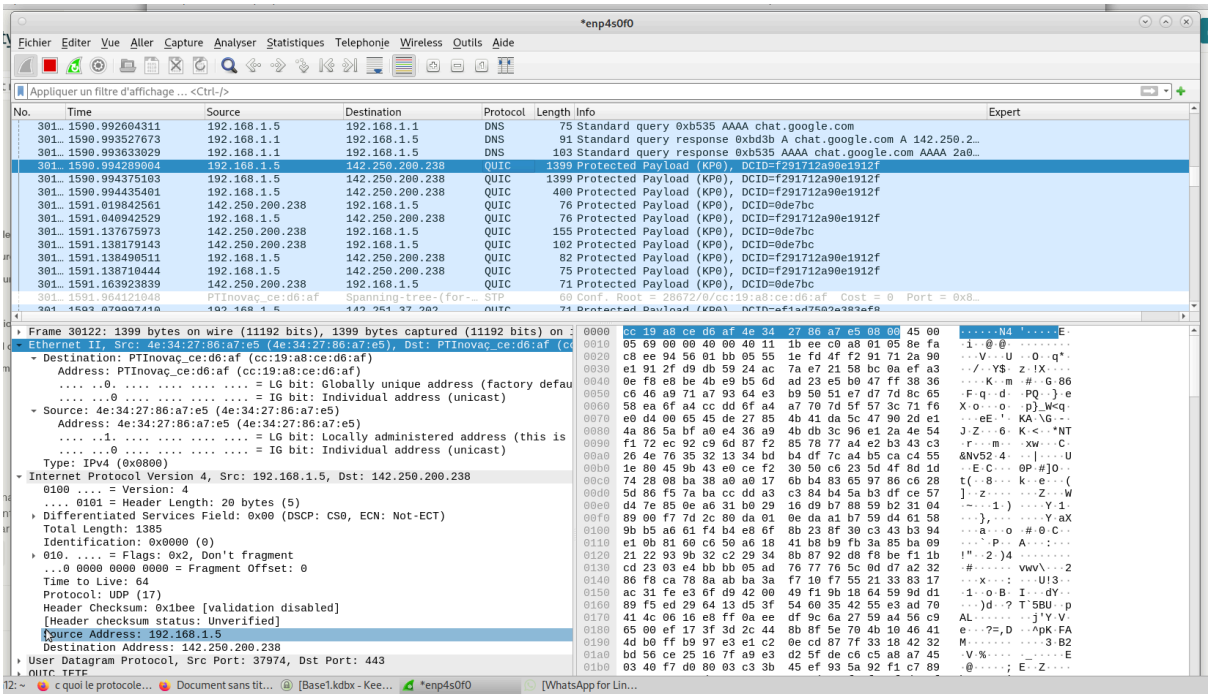
TLSv1.3


*enp4s0f0
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Appliquer un filtre d'affichage … <Ctrl-/>
No. Time Source Destination Protocol Length Info Expert
300… 1586.569450582 192.168.1.5 163.70.128.60 TLSv1.3 536 Application Data
300… 1586.569494121 192.168.1.5 163.70.128.60 TLSv1.3 993 Application Data
300… 1586.586377121 163.70.128.60 192.168.1.5 TCP 66 443 → 45468 [ACK] Seq=166 Ack=759 Win=67072 Len=0 TSval=4136…
300… 1586.586861963 163.70.128.60 192.168.1.5 TLSv1.3 235 Application Data
300… 1586.587347256 163.70.128.60 192.168.1.5 TCP 66 443 → 45468 [ACK] Seq=335 Ack=2156 Win=70400 Len=0 TSval=413…
300… 1586.619128071 192.168.1.5 163.70.128.60 TCP 74 45472 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM T… Chat
300… 1586.620475401 163.70.128.60 192.168.1.5 TLSv1.3 814 Application Data
300… 1586.620496547 192.168.1.5 163.70.128.60 TCP 66 45468 → 443 [ACK] Seq=2156 Ack=1083 Win=64128 Len=0 TSval=34…
300… 1586.645257243 163.70.128.60 192.168.1.5 TCP 74 443 → 45472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 … Chat
300… 1586.645298299 192.168.1.5 163.70.128.60 TCP 66 45472 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=34717654…
300… 1586.645978020 192.168.1.5 163.70.128.60 TLSv1.3 760 Client Hello
300… 1586.670183042 163.70.128.60 192.168.1.5 TCP 66 443 → 45472 [ACK] Seq=1 Ack=695 Win=67072 Len=0 TSval=285250…
300… 1586.671003529 163.70.128.60 192.168.1.5 TLSv1.3 231 Server Hello, Application Data
300… 1586.671026776 192.168.1.5 163.70.128.60 TCP 66 45472 → 443 [ACK] Seq=695 Ack=166 Win=64128 Len=0 TSval=3471…
Frame 30086: 235 bytes on wire (1880 bits), 235 bytes captured (1880 bits) on inter
Ethernet II, Src: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af), Dst: 4e:34:27:86:a7:e5 (4
Destination: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
Address: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
.... ..1. .... = LG bit: Locally administered address (this is
....0 .... = IG bit: Individual address (unicast)
Source: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
Address: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
.... ..0. .... = LG bit: Globally unique address (factory defau
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 163.70.128.60, Dst: 192.168.1.5
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 221
Identification: 0x8976 (35190)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 56
Protocol: TCP (6)
Header Checksum: 0xd374 [validation disabled]
[Header checksum status: Unverified]
Source Address: 163.70.128.60
Destination Address: 192.168.1.5
Transmission Control Protocol, Src Port: 443, Dst Port: 45468, Seq: 166, Ack: 759,
Transport Layer Security
0000 4e 34 27 86 a7 e5 cc 19 a8 ce d6 af 08 00 45 00
0010 00 dd 89 76 40 00 38 06 d3 74 a3 46 80 3c c0 a8
0020 01 05 01 bb b1 9c f9 3d 7f d3 7a 04 57 2d 80 18
0030 01 06 cd 1a 00 00 01 01 08 0a f6 89 86 8f ce ee
0040 ef a6 17 03 03 00 a4 25 89 f4 84 34 de 39 02 a5
0050 d5 12 4f 94 3f 1b 89 73 9f b9 14 a5 db 3e a8 4d
0060 cf f3 b4 f3 f0 60 18 fd aa e6 ad 7a 9c de 54 9f
0070 b9 80 31 f4 62 74 f1 0e 98 5c 36 88 30 f0 0c 0f
0080 cf c1 e0 a2 51 2c 14 61 53 cb 1a 3e b9 52 d2 a5
0090 b6 c9 0b 72 34 37 cd 8e 7a 7c 9a 16 6f 59 92 18
00a0 cf 71 f0 13 73 74 23 e8 81 8d 79 ac a7 9b 9d 91
00b0 2b 9d c0 89 c6 f7 c3 56 1c cb 21 69 88 60 0e e5
00c0 6a 5a 72 76 c3 24 25 12 e8 b6 43 73 8f 84 a6 38
00d0 77 22 90 86 79 86 b7 8a 80 d2 2f 19 cd 9f 36 ba
00e0 3c c7 2d 4e b1 6f 0a 25 ff 50 8e
c quoi le protocole… Analyse réseau - … [Base1.kdbx - Kee… *enp4s0f0 [WhatsApp for Lin…

DNS


*enp4s0f0
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Appliquer un filtre d'affichage … <Ctrl-/>
No. Time Source Destination Protocol Length Info Expert
301… 1589.317348798 ce:19:a8:ce:d6:b0 IEEE-1905.1-Control ieee19… 60 Topology notification Warning
301… 1589.318610037 ce:19:a8:ce:d6:b0 IEEE-1905.1-Control ieee19… 60 Topology notification Warning
301… 1589.536247047 WistronN_13:b7:61 Broadcast XID 60 Basic Format; Type 1 LLC (Class I LLC); Window Size 0
301… 1589.537981820 ce:19:a8:ce:d6:b0 IEEE-1905.1-Control ieee19… 60 Topology notification Warning
301… 1589.538959186 ce:19:a8:ce:d6:b0 IEEE-1905.1-Control ieee19… 60 Topology notification Warning
301… 1589.964248672 PTInovaç_ce:d6:af Spanning-tree-(for-… STP 60 Conf. Root = 28672/0/cc:19:a8:ce:d6:af Cost = 0 Port = 0x8…
301… 1590.849141399 PTInovaç_ce:d6:af Broadcast ARP 60 Who has 192.168.1.119? Tell 192.168.1.1
301… 1590.985095996 192.168.1.5 192.168.1.1 DNS 75 Standard query 0xff11 A chat.google.com
301… 1590.985119127 192.168.1.5 192.168.1.1 DNS 75 Standard query 0x530e AAAA chat.google.com
301… 1590.985992830 192.168.1.1 192.168.1.5 DNS 91 Standard query response 0xff11 A chat.google.com A 142.250.2…
301… 1590.986068566 192.168.1.1 192.168.1.5 DNS 103 Standard query response 0x530e AAAA chat.google.com AAAA 2a0…
301… 1590.992581856 192.168.1.5 192.168.1.1 DNS 75 Standard query 0xbd3b A chat.google.com
301… 1590.992604311 192.168.1.5 192.168.1.1 DNS 75 Standard query 0xb535 AAAA chat.google.com
301… 1590.993527673 192.168.1.1 192.168.1.5 DNS 91 Standard query response 0xbd3b A chat.google.com A 142.250.2…
Frame 30114: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface
Ethernet II, Src: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5), Dst: PTInovaç_ce:d6:af (cc
Destination: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
Address: PTInovaç_ce:d6:af (cc:19:a8:ce:d6:af)
.... ..0. .... = LG bit: Globally unique address (factory defau
....0 .... = IG bit: Individual address (unicast)
Source: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
Address: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
.... ..1. .... = LG bit: Locally administered address (this is
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.1
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 61
Identification: 0x015f (351)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0xb5fa [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.1.5
Destination Address: 192.168.1.1
User Datagram Protocol, Src Port: 57166, Dst Port: 53
Domain Name System (query)
0000 cc 19 a8 ce d6 af 4e 34 27 86 a7 e5 08 00 45 00
0010 00 3d 01 5f 40 00 40 11 b5 fa c0 a8 01 05 c0 a8
0020 01 01 df 4e 00 35 00 29 83 91 ff 11 01 00 00 01
0030 00 00 00 00 00 00 04 63 68 61 74 06 67 6f 6f 67
0040 6c 65 03 63 6f 6d 00 00 01 00 01
12: ~ c quoi le protocole… Document sans tit… [Base1.kdbx - Kee… *enp4s0f0 [WhatsApp for Lin…

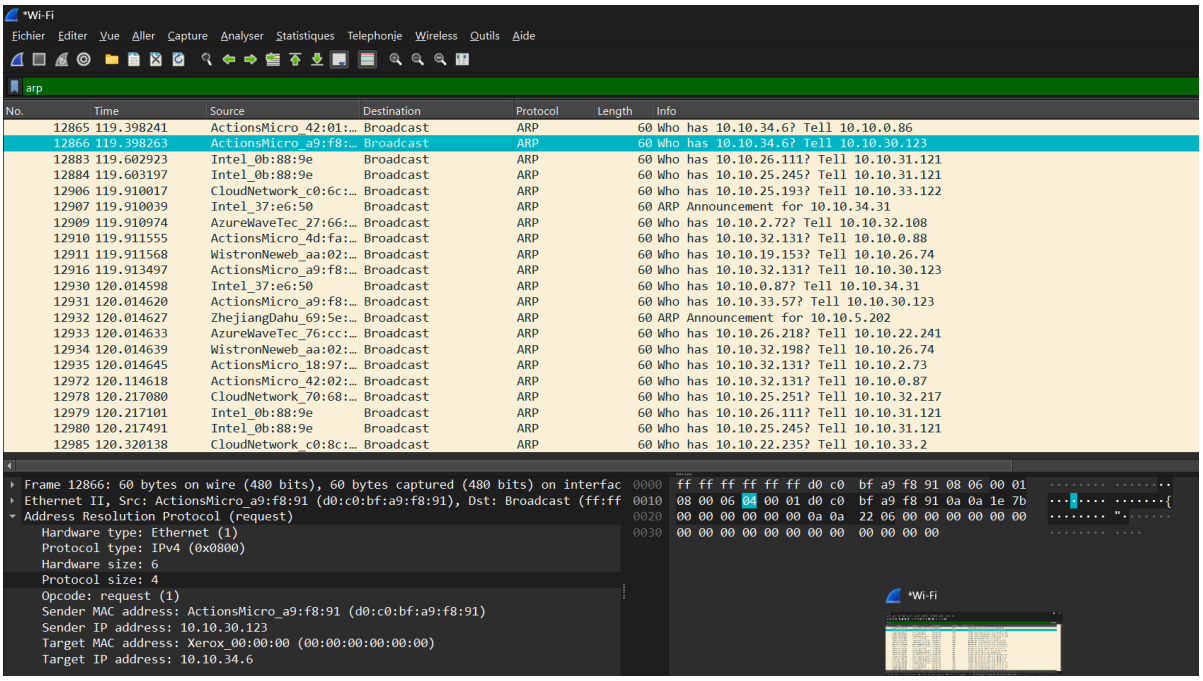
QUIC


*enp4s0f0
Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide
Appliquer un filtre d'affichage … <Ctrl-/>
No. Time Source Destination Protocol Length Info Expert
301… 1590.992604311 192.168.1.5 192.168.1.1 DNS 75 Standard query 0xb535 AAAA chat.google.com
301… 1590.993527673 192.168.1.1 192.168.1.5 DNS 91 Standard query response 0xbd3b A chat.google.com A 142.250.2…
301… 1590.993633029 192.168.1.1 192.168.1.5 DNS 103 Standard query response 0xb535 AAAA chat.google.com AAAA 2a0…
301… 1590.994289004 192.168.1.5 142.250.200.238 QUIC 1399 Protected Payload (KP0), DCID=f291712a90e1912f
301… 1590.994375103 192.168.1.5 142.250.200.238 QUIC 1399 Protected Payload (KP0), DCID=f291712a90e1912f
301… 1590.994435401 192.168.1.5 142.250.200.238 QUIC 400 Protected Payload (KP0), DCID=f291712a90e1912f
301… 1591.019842561 142.250.200.238 192.168.1.5 QUIC 76 Protected Payload (KP0), DCID=0de7bc
301… 1591.040942529 192.168.1.5 142.250.200.238 QUIC 76 Protected Payload (KP0), DCID=f291712a90e1912f
301… 1591.137675973 142.250.200.238 192.168.1.5 QUIC 155 Protected Payload (KP0), DCID=0de7bc
301… 1591.138179143 142.250.200.238 192.168.1.5 QUIC 102 Protected Payload (KP0), DCID=0de7bc
301… 1591.138490511 192.168.1.5 142.250.200.238 QUIC 82 Protected Payload (KP0), DCID=f291712a90e1912f
301… 1591.138710444 192.168.1.5 142.250.200.238 QUIC 75 Protected Payload (KP0), DCID=f291712a90e1912f
301… 1591.163923839 142.250.200.238 192.168.1.5 QUIC 71 Protected Payload (KP0), DCID=0de7bc
301… 1591.964121048 PTInovac_ce:d6:af Spanning-tree-(for-… STP 60 Conf. Root = 28672/0/cc:19:a8:ce:d6:af Cost = 0 Port = 0x8…
Frame 30122: 1399 bytes on wire (11192 bits), 1399 bytes captured (11192 bits) on i
Ethernet II, Src: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5), Dst: PTInovac_ce:d6:af (cc
Destination: PTInovac_ce:d6:af (cc:19:a8:ce:d6:af)
Address: PTInovac_ce:d6:af (cc:19:a8:ce:d6:af)
.... ..0. .... = LG bit: Globally unique address (factory defau
....0 .... = IG bit: Individual address (unicast)
Source: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
Address: 4e:34:27:86:a7:e5 (4e:34:27:86:a7:e5)
.... ..1. .... = LG bit: Locally administered address (this is
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.1.5, Dst: 142.250.200.238
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1385
Identification: 0x0000 (0)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x1bee [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.1.5
Destination Address: 142.250.200.238
User Datagram Protocol, Src Port: 37974, Dst Port: 443
QUIC IETF
0000 cc 19 a8 ce d6 af 4e 34 27 86 a7 e5 08 00 45 00
0010 05 69 00 00 40 00 40 11 1b ee c0 a8 01 05 8e fa
0020 c8 ee 94 56 01 bb 05 55 1e fd 4f f2 91 71 2a 90
0030 e1 91 2f d9 db 59 24 ac 7a e7 21 58 bc 0a ef a3
0040 0e f8 e8 be 4b e9 b5 6d ad 23 e5 b0 47 ff 38 36
0050 c6 46 a9 71 a7 93 64 e3 b9 50 51 e7 d7 7d 8c 65
0060 58 ea 6f a4 cc dd 6f a4 a7 70 7d 5f 57 3c 71 f6
0070 e0 d4 00 65 45 de 27 85 4b 41 da 5c 47 90 2d e1
0080 4a 86 5a bf a0 e4 36 a9 4b db 3c 96 e1 2a 4e 54
0090 f1 72 ec 92 c9 6d 87 f2 85 78 77 a4 e2 b3 43 c3
00a0 26 4e 76 35 32 13 34 bd b4 df 7c a4 b5 ca c4 55
00b0 1e 80 45 9b 43 e0 ce f2 30 50 c6 23 5d 4f 8d 1d
00c0 74 28 08 ba 38 a0 a0 17 6b b4 83 65 97 86 c6 28
00d0 5d 86 f5 7a ba cc dd a3 c3 84 b4 5a b3 df ce 57
00e0 d4 7e 85 0e a6 31 b0 29 16 d9 b7 88 59 b2 31 04
00f0 89 00 f7 7d 2c 80 da 01 0e da a1 b7 59 d4 61 58
0100 9b b5 a6 61 f4 b4 e8 6f 8b 23 8f 30 c3 43 b3 94
0110 e1 0b 81 60 c6 50 a6 18 41 b8 b9 fb 3a 85 ba 09
0120 21 22 93 9b 32 c2 29 34 8b 87 92 d8 f8 be f1 1b
0130 cd 23 03 e4 bb bb 05 ad 76 77 76 5c 0d d7 a2 32
0140 86 f8 ca 78 8a ab ba 3a f7 10 f7 55 21 33 83 17
0150 ac 31 fe e3 6f d9 42 00 49 f1 9b 18 64 59 9d d1
0160 89 f5 ed 29 64 13 d5 3f 54 60 35 42 55 e3 ad 70
0170 41 4c 06 16 e8 ff 0a ee df 9c 6a 27 59 a4 56 c9
0180 65 00 ef 17 3f 3d 2c 44 8b 8f 5e 70 4b 10 46 41
0190 4d b0 ff b9 97 e3 e1 c2 0e cd 87 7f 33 18 42 32
01a0 bd 56 ce 25 16 7f a9 e3 d2 5f de c6 c5 a8 a7 45
01b0 03 40 f7 d0 80 03 c3 3b 45 ef 93 5a 92 f1 c7 89
c quoi le protocole…
Document sans tit…
[Base1.kdbx - Kee…
*enp4s0f0
[WhatsApp for Lin…

Pour chercher les spécifications du format des messages ARP, UDP et TCP afin de faire correspondre les captures en hexadécimal, voici les étapes à suivre :

Format des messages ARP

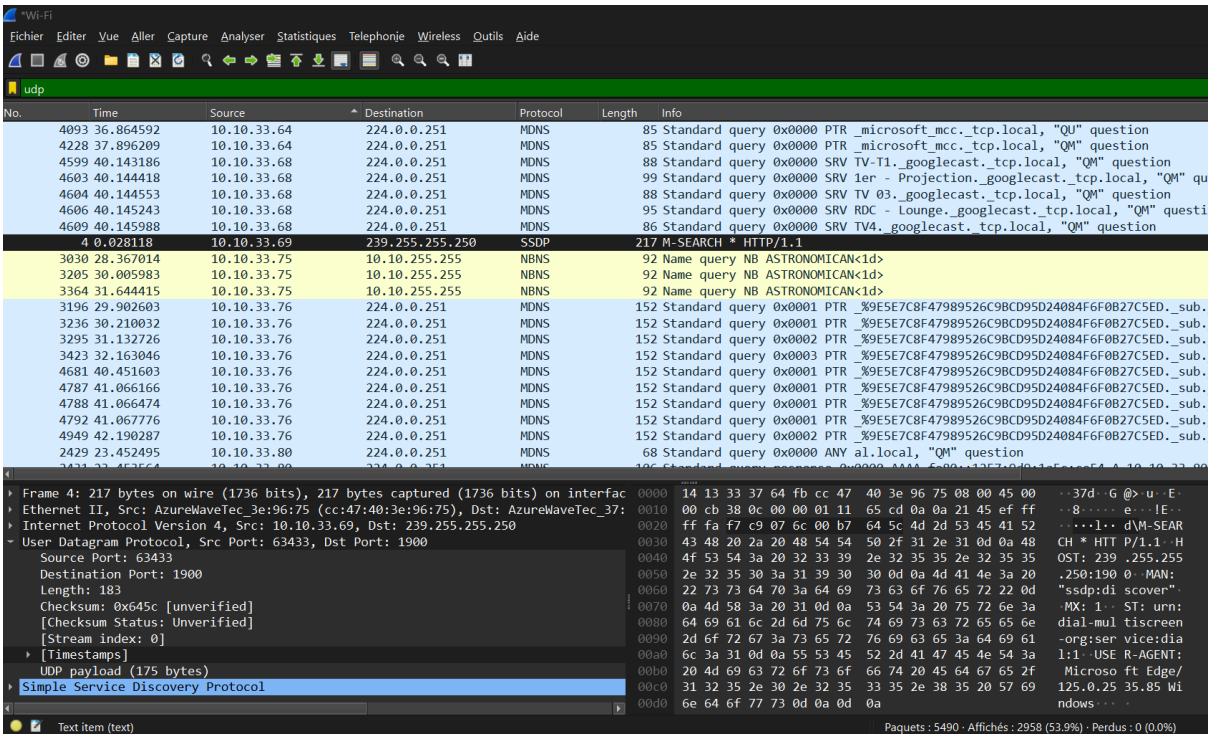
1. Le format des messages ARP est défini dans la RFC 826
1. On y trouve la structure détaillée des différents champs.
2. Dans Wireshark, sélectionnez une trame ARP capturée et développez la section "Address Resolution Protocol (request/reply)".
3. Vous pouvez alors faire correspondre chaque champ hexadécimal affiché avec sa spécification dans la RFC, comme :
    - Hardware type (0x0001 pour Ethernet)
    - Protocol type (0x0800 pour IPv4)
    - Opération (0x0001 pour requête, 0x0002 pour réponse)
    - Adresses MAC et IP source/destination



Format des segments UDP

Le format des segments UDP est défini dans la RFC 768

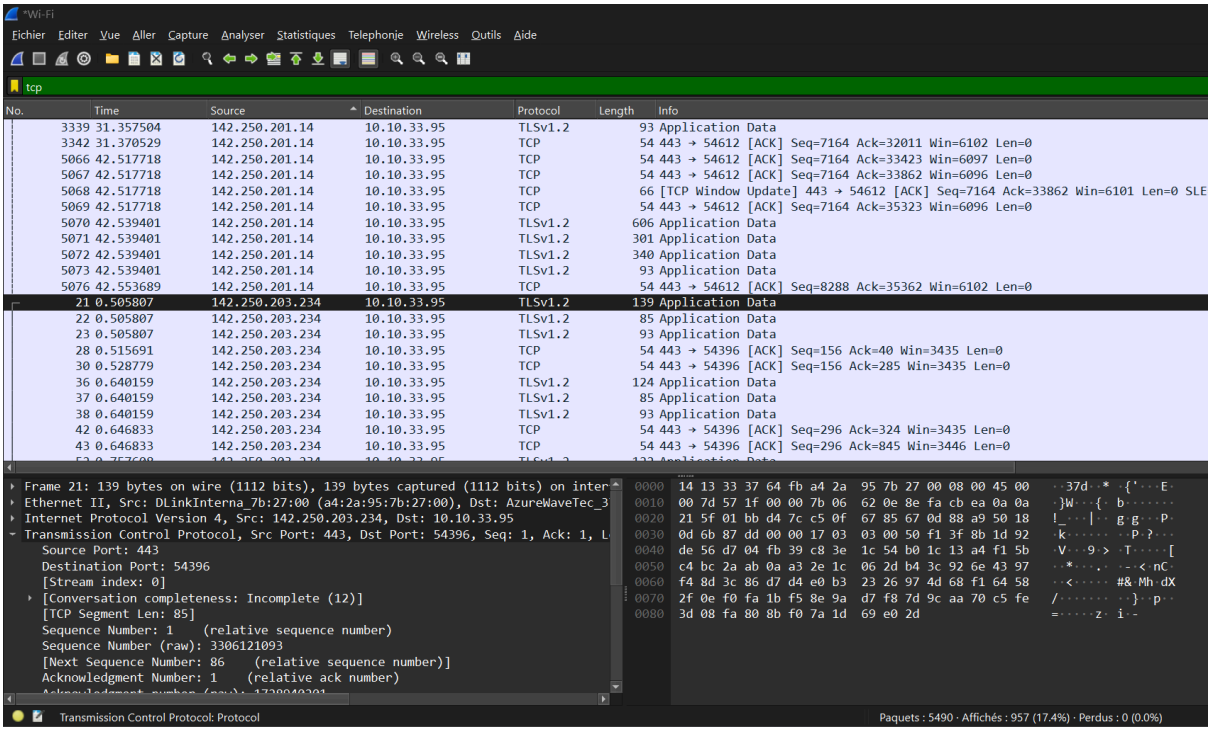
1. Dans Wireshark, sélectionnez un paquet UDP et développez la section "User Datagram Protocol".
2. Faites correspondre les champs hexadécimaux avec la RFC :
    - Source/Destination Ports
    - Longueur du segment
    - Checksum



Format des segments TCP

Le format des segments TCP est défini dans la RFC 793

1. Dans Wireshark, sélectionnez un paquet TCP et développez la section "Transmission Control Protocol".
2. Faites correspondre les champs avec la RFC :
    - Ports source/destination
    - Numéro de séquence/acquittement
    - Longueur de l'en-tête
    - Drapeaux (SYN, ACK, etc.)
    - Fenêtre de réception
    - Somme de contrôle
    - Pointeur d'urgence



En vous référant aux RFCs définissant les formats de ces protocoles, vous pouvez aisément faire le lien entre les spécifications et les données hexadécimales capturées par Wireshark pour chaque trame/segment.

Décrivez le mécanisme de connexion avec un diagramme.

Pour trouver les paquets TCP correspondant aux différentes étapes de la connexion entre votre hôte et un serveur (SYN, ACK, FIN, etc.), voici la démarche à suivre avec Wireshark :

1. Lancez une capture de trafic sur l'interface réseau connectée à Internet.
2. Établissez une connexion TCP vers un serveur (par exemple en ouvrant un navigateur web).
3. Dans Wireshark, appliquez un filtre d'affichage sur le protocole TCP avec `tcp` pour n'afficher que les paquets TCP.
4. Examinez les drapeaux TCP (SYN, ACK, FIN, etc.) dans les paquets capturés :

* Paquet avec le drapeau SYN=1 : C'est la requête d'ouverture de connexion envoyée par le client

* Paquet avec les drapeaux SYN=1 et ACK=1 : C'est la réponse du serveur acceptant la connexion

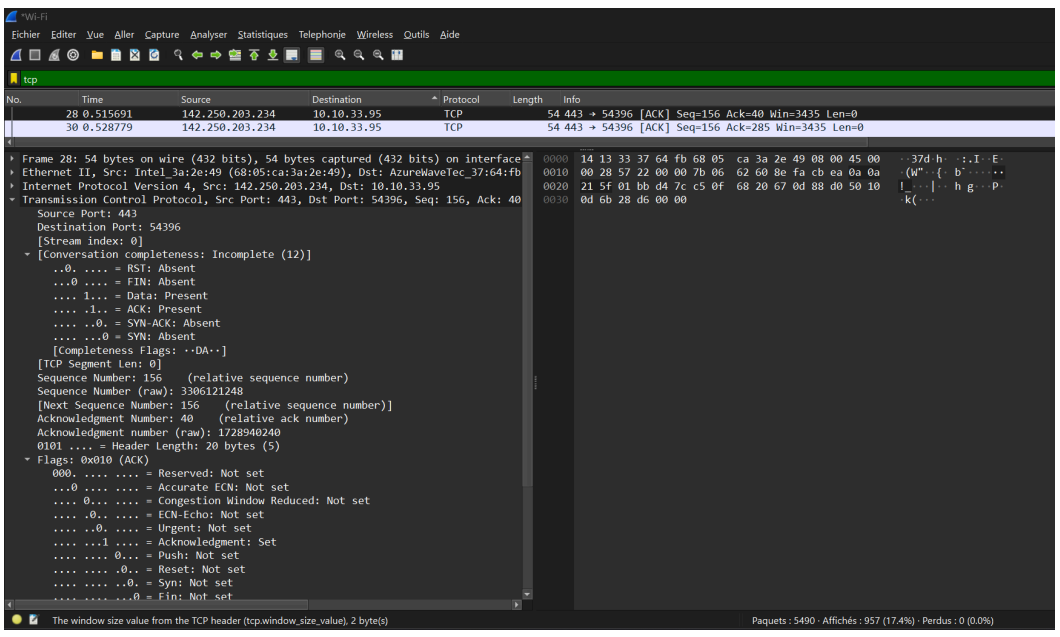
* Paquets avec seulement ACK=1 : Ce sont les acquittements échangés pendant le transfert de données

.

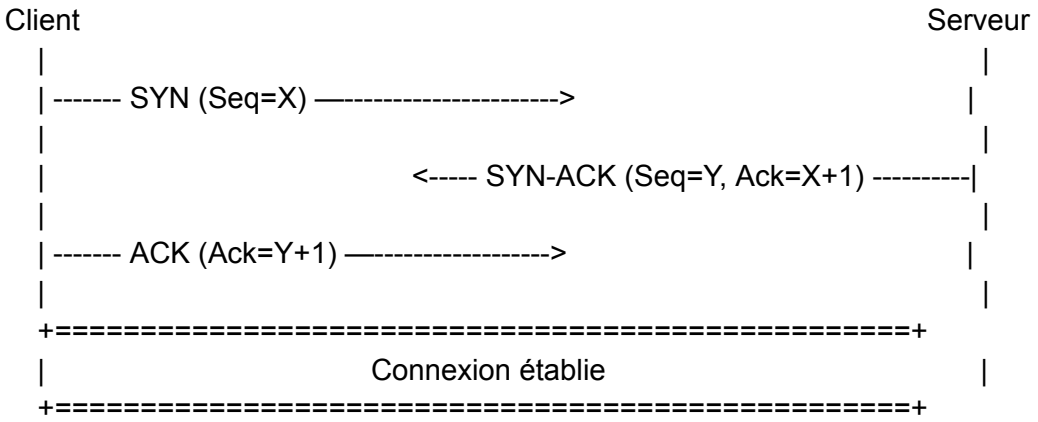
* Paquet avec le drapeau FIN=1 : C'est la demande de fin de connexion envoyée par l'un des hôtes

* Paquet avec les drapeaux FIN=1 et ACK=1 : C'est l'acquittement de la fin de connexion

Vous pouvez ainsi identifier facilement les paquets SYN, SYN-ACK, ACK, FIN, etc. qui correspondent aux différentes étapes de l'établissement, du transfert de données et de la fermeture de la connexion TCP. Les champs TCP flags dans Wireshark affichent les drapeaux, et les numéros de séquence/acquittement permettent de suivre l'état de la connexion.



Voici une description détaillée du mécanisme de connexion TCP avec un diagramme, basée sur les sources fournies : La connexion TCP suit un processus en 3 étapes appelé "three-way handshake" pour établir une session fiable entre un client et un serveur.



SYN : Le client envoie un segment avec le drapeau SYN=1 et un numéro de séquence initial aléatoire X

.

SYN-ACK : Le serveur répond avec un segment ayant les drapeaux SYN=1 et ACK=1. Il contient son propre numéro de séquence initial Y et acquitte le numéro de séquence du client avec Ack=X+1

.

ACK : Le client acquitte le numéro de séquence du serveur avec un segment ayant le drapeau ACK=1 et le champ Ack=Y+1

Après cette négociation en 3 étapes, la connexion TCP est établie et le transfert bidirectionnel de données peut commencer, avec une numérotation des séquences basée sur les numéros initiaux X et Y. Quelques points clés :

- Les numéros de séquence initiaux sont générés aléatoirement pour chaque nouvelle connexion afin d'éviter les collisions

Le champ Ack contient toujours le prochain numéro de séquence attendu par l'expéditeur.

Wireshark permet de visualiser les drapeaux TCP (SYN, ACK, etc.) et les numéros de séquence/acquittement pour suivre l'état de la connexion.

Ce mécanisme d'établissement de connexion TCP garantit la fiabilité et l'intégrité des données transférées, en permettant la synchronisation en cas de perte de paquets.

Filtres d'affichage

Voici quelques informations sur l'utilisation des filtres dans Wireshark, accompagnées d'exemples pratiques.

Les filtres d'affichage permettent de n'afficher que les paquets correspondant à certains critères, sans modifier la capture elle-même. On les applique dans la barre de filtre en haut de Wireshark. Quelques exemples de filtres d'affichage courants :

- `ip.addr == 192.168.1.1` : Affiche les paquets impliquant l'adresse IP 192.168.1.1
- `tcp.port == 80` : Affiche les paquets TCP sur le port 80 (HTTP)
- `http.request.method == "GET"` : Affiche les requêtes HTTP GET
-

Filtres de capture

Les filtres de capture permettent de capturer uniquement les paquets correspondant à certains critères, directement lors de la capture réseau. Pour définir un filtre de capture, cliquez sur "Capture Options" dans Wireshark, puis entrez le filtre dans le champ "Capture Filter". Exemple : `host 8.8.8.8` ne capturera que le trafic impliquant l'adresse IP 8.8.8.8.

Tests de filtres

J'ai effectué quelques tests de filtres sur une capture de trafic web :

1. `http` : Affiche uniquement les paquets HTTP
2. `http.host contains "google"` : Affiche les requêtes HTTP vers des hôtes contenant "google"
3. `tcp.flags.reset == 1` : Affiche les paquets TCP avec le drapeau RST (réinitialisation de connexion)

Vous pouvez combiner plusieurs critères avec **`and`** `“&&”`, **`or`** `“||”` et `“not”`our créer des filtres plus complexes. L'utilisation judicieuse des filtres dans Wireshark est cruciale pour analyser efficacement de grandes quantités de trafic réseau. N'hésitez pas à vous entraîner et à consulter la documentation de Wireshark pour maîtriser cette fonctionnalité essentielle.

Partie 2

Concernant la partie installation les services qui permettent d'écouter
les protocoles demandés, capturer les paquets en utilisant des filtres Wireshark. Voir la documentation Config_SSH-FTP-WebServer-DNS-Wireshark

Analyser le Trafic

HTTP : `http` Exemples pratiques pour générer et capturer du trafic HTTP

Avec Wireshark, tu peux :
- Visualiser les requêtes et réponses HTTP : Utiliser les filtres pour trier par méthode (GET, POST, etc.).
- Examiner les en-têtes HTTP : Vérifie les en-têtes envoyés et reçus.
- Analyser le contenu : Visualiser les données transportées, comme les pages HTML, les fichiers, etc.

Voici un exemple complet de génération de trafic HTTP et de capture avec Wireshark :

1. Configurer le Serveur HTTP (`Apache` ou autre) pour servir des pages simples.
2. Configurer le client pour accéder au serveur HTTP (`192.168.233.135`).
3. Lancer Wireshark sur le client ou le serveur.
4. Utiliser des commandes `curl` et `wget` pour effectuer des requêtes HTTP:

curl http://192.168.233.135
wget http://192.168.233.135/file.txt

5. Naviguer vers le site web dans un navigateur pour générer du trafic supplémentaire.
6. Examiner les paquets HTTP capturés dans Wireshark pour voir les requêtes et réponses.

En utilisant ces commandes et techniques, tu pourras générer et capturer du trafic HTTP varié pour ton analyse avec Wireshark.

Pour plus d' exemples pratiques et requêtes HTTP basiques, avancées, et des interactions entre un client et un serveur HTTP j'ai rédigé HTTP_Générer_Trafic_Wireshark

FTP : `ftp` Exemples pratiques pour générer et capturer du trafic

Pour te donner une idée, voici un exemple de session FTP où tu télécharges et téléverser des fichiers :

Se connecter au serveur FTP

ftp 192.168.233.135

Nom d'utilisateur et mot de passe (utiliser un utilisateur valide)

Name: user

Password:* ***

Lister les fichiers dans le répertoire courant

ftp> ls

Changer de répertoire

ftp> cd /some/dir

Télécharger un fichier

ftp> get example.txt

Télécharger plusieurs fichiers

ftp> mget *.txt

Téléverser un fichier

ftp> put localfile.txt

Supprimer un fichier sur le serveur

ftp> delete serverfile.txt

Quitter la session FTP

ftp> bye

DNS : `dns` Exemple Pratique pour Générer et Capturer du Trafic

Voici un exemple complet de génération de trafic DNS et de capture avec Wireshark :

1. Configurer le serveur DNS (`Bind9` ou autre) pour répondre aux requêtes de `example.com`.

2. Configurer le client pour utiliser le serveur DNS (`192.168.233.135`).

3. Lancer Wireshark sur le client ou le serveur.

4. Utiliser des commandes `dig` et `nslookup` pour effectuer des requêtes DNS** :

dig example.com @192.168.233.135

dig example.com MX @192.168.233.135

nslookup example.com 192.168.233.135

5. Examiner les paquets DNS capturés dans Wireshark pour voir les requêtes et réponses.

En utilisant ces commandes et techniques, tu pourras générer et capturer du trafic DNS varié pour ton analyse avec Wireshark.

Cliquer sur les paquets pour voir les détails des protocoles et les données échangées.

Avec Wireshark, on peut :

- Visualiser les requêtes et réponses DNS : Utilise les filtres pour trier par type de requête (A, AAAA, MX, etc.).

- Examiner les enregistrements DNSSEC: Vérifier la validité des signatures.

- Analyser les temps de réponse : Identifier les latences dans la résolution des noms.

J'ai créé un fichier pour détailler d'autres méthodes, commandes et outils sur une machine Debian pour générer du trafic DNS à analyser avec Wireshark:
Dns_Commandes_et_outils_Wireshark

Étape 6: Sauvegarder et Partager les Captures

Pour sauvegarder les captures pour une analyse ultérieure ou pour les partager :

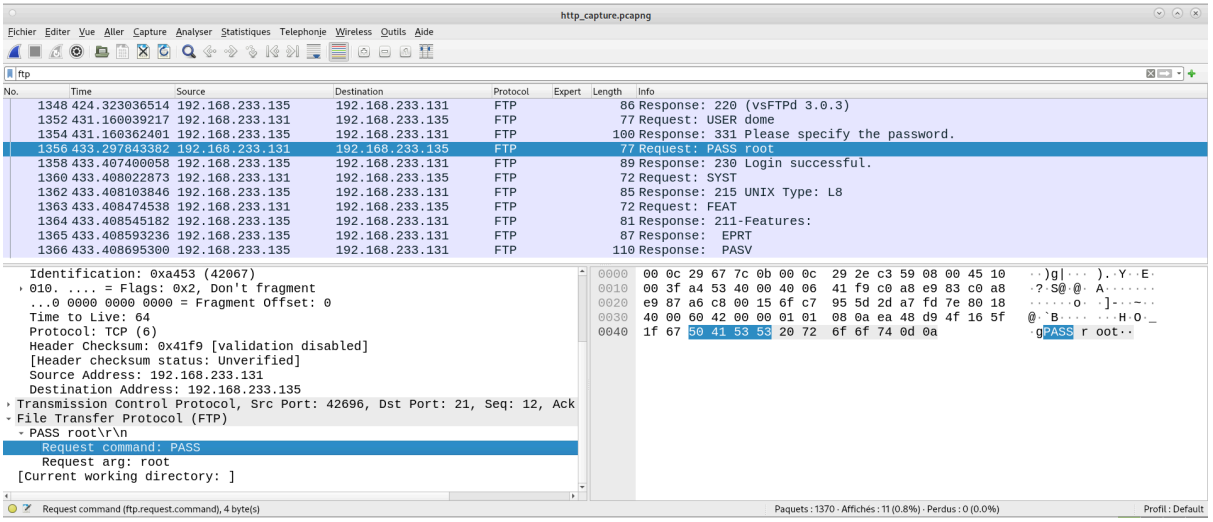
1. Dans Wireshark, va dans **File > Save As**.

2. Choisis le format `.pcap` et un emplacement pour sauvegarder le fichier.

Tu peux ensuite charger ces fichiers dans Wireshark pour une analyse ultérieure

En écoutant des échanges FTP sans TLS, que remarquez-vous dans les paquets ? Est-il possible de récupérer des données sensibles de connexion ? En est-il de même avec les échanges SSL ?

Dans les paquet Ftp on peut observer en clair le nom utilisateur et mot de passe



FTP sans TLS (FTP non sécurisé) :

- Données en clair : Les données, y compris les noms d'utilisateur, les mots de passe et les commandes FTP, sont transmises en texte brut sur le réseau.
- Vulnérabilité : Toute personne capable d'intercepter le trafic réseau peut facilement lire et extraire ces informations sensibles en analysant les paquets FTP capturés.
- Exemples de paquets : Vous verrez les commandes FTP telles que USER, PASS pour l'authentification, et les données de fichier transférées comme le contenu brut dans les paquets capturés.

En résumé, oui, il est possible de récupérer des données sensibles de connexion telles que les noms d'utilisateur et les mots de passe lors de l'utilisation de FTP non sécurisé.

FTP avec SSL/TLS (FTP sécurisé) :

- Chiffrement : Lorsque FTP est utilisé avec SSL/TLS (FTPS), toutes les données, y compris les commandes et les données de fichier, sont chiffrées avant d'être transmises sur le réseau.
- Protection : Le chiffrement SSL/TLS empêche les tiers d'intercepter et de lire le contenu des commandes FTP, des identifiants et des données de fichier.
- Complexité de l'analyse : Bien que les données soient chiffrées, il est possible d'identifier des sessions FTPS et de savoir qu'une connexion a lieu, mais l'accès aux données elles-mêmes est rendu extrêmement difficile sans la clé de déchiffrement appropriée.

En résumé, avec FTPS (FTP sécurisé via SSL/TLS), bien que l'interception de données soit encore possible, la lecture des données sensibles est largement empêchée par le chiffrement SSL/TLS.

Il est crucial de toujours utiliser des méthodes sécurisées comme FTPS (FTP avec SSL/TLS) pour protéger les informations sensibles transmises sur un réseau. Le FTP non sécurisé expose les données à une interception facile et à une utilisation malveillante. Assurez-vous de choisir les méthodes appropriées en fonction des exigences de sécurité de votre environnement réseau.

Partie 3

Utilisation de tshark pour Capturer et Analyser des Paquets

Installation de tshark

Tshark est une commande en ligne de commande qui permet de capturer et d'analyser des paquets réseau. Pour l'installer sous Linux :

sudo apt install tshark

Capture de Paquets avec tshark

Pour capturer des paquets de différents protocoles avec tshark, vous pouvez utiliser des filtres directement dans la commande ou rediriger la sortie vers un fichier pour une analyse ultérieure.

Capture de Paquets UDP :

Pour capturer les paquets UDP avec tshark et les afficher en temps réel dans le terminal :

sudo tshark -i <interface> -f "udp"

-i <interface> : Spécifie l'interface réseau à écouter.
-f "udp" : Filtre les paquets pour ne capturer que ceux utilisant le protocole UDP.

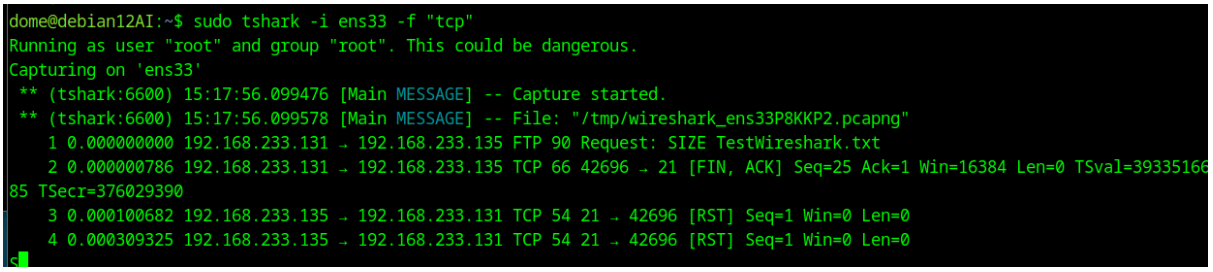
```
dome@debian12AI:~$ sudo tshark -i ens33 -f "udp"
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ens33'
 ** (tshark:6485) 15:05:05.771464 [Main MESSAGE] -- Capture started.
 ** (tshark:6485) 15:05:05.771546 [Main MESSAGE] -- File: "/tmp/wireshark_ens331KYYP2.pcapng"
    1 0.000000000 fe80::df92:ba94:985f:4be3 → ff02::c      UDP 718 56994 → 3702 Len=656
    2 0.001066770 192.168.233.1 → 239.255.255.250 UDP 698 56993 → 3702 Len=656
    3 0.232928682 fe80::df92:ba94:985f:4be3 → ff02::c      UDP 718 56994 → 3702 Len=656
    4 0.263537096 192.168.233.1 → 239.255.255.250 UDP 698 56993 → 3702 Len=656
    5 0.685727989 fe80::df92:ba94:985f:4be3 → ff02::c      UDP 718 56994 → 3702 Len=656
    6 0.762935499 192.168.233.1 → 239.255.255.250 UDP 698 56993 → 3702 Len=656
    7 1.588735790 fe80::df92:ba94:985f:4be3 → ff02::c      UDP 718 56994 → 3702 Len=656
    8 1.761499969 192.168.233.1 → 239.255.255.250 UDP 698 56993 → 3702 Len=656
    9 3.381164127 fe80::df92:ba94:985f:4be3 → ff02::c      UDP 718 56994 → 3702 Len=656
   10 3.768633299 192.168.233.1 → 239.255.255.250 UDP 698 56993 → 3702 Len=656
^C   11 5.392971694 fe80::df92:ba94:985f:4be3 → ff02::c      UDP 718 56994 → 3702 Len=656
^X   12 5.784023205 192.168.233.1 → 239.255.255.250 UDP 698 56993 → 3702 Len=656
   13 7.398319890 fe80::df92:ba94:985f:4be3 → ff02::c      UDP 718 56994 → 3702 Len=656
   14 7.787415362 192.168.233.1 → 239.255.255.250 UDP 698 56993 → 3702 Len=656
```

Capture de Paquets TCP :

Pour capturer les paquets TCP avec tshark :

sudo tshark -i <interface> -f "tcp"

-f "tcp" : Filtre les paquets pour ne capturer que ceux utilisant le protocole TCP.

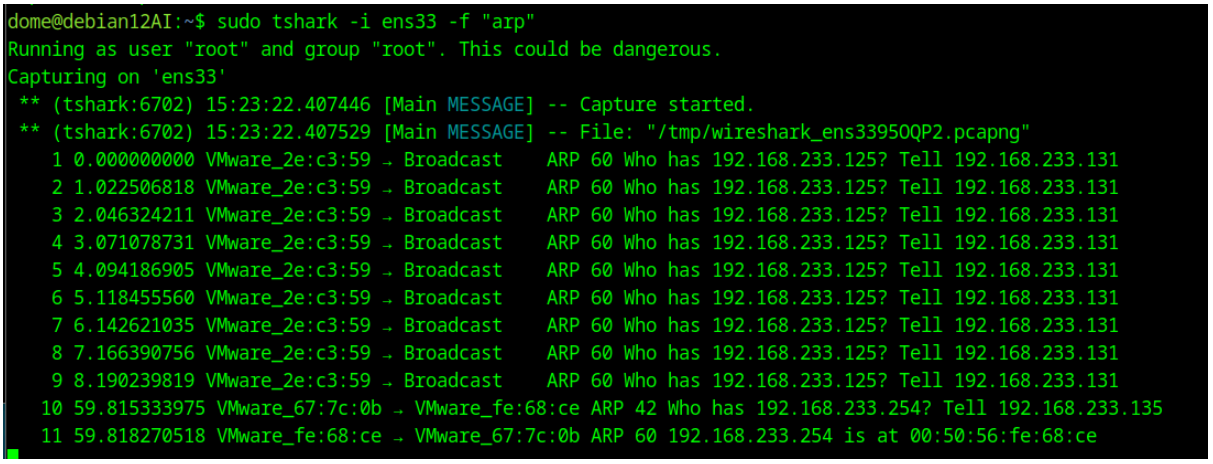


Capture de Paquets ARP :

Pour capturer les paquets ARP avec tshark :

sudo tshark -i <interface> -f "arp"

-f "arp" : Filtre les paquets pour ne capturer que ceux utilisant le protocole ARP.



Explication des Options de tshark

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

-f "<filter>" : Permet de définir un filtre pour sélectionner les paquets à capturer. Le filtre peut être basé sur des protocoles (tcp, udp, arp), des adresses IP, des ports, etc.

-w <output_file> : Option pour rediriger la sortie de tshark vers un fichier pcap pour une analyse ultérieure. Par exemple :

Capture de Paquets et rediriger la sortie de tshark vers un fichier pcap

sudo tshark -i eth0 -f "tcp" -w capture_tcp.pcap

Cela capturera tous les paquets TCP sur l'interface eth0 et les enregistrera dans le fichier capture_tcp.pcap.

-Y "<display_filter>" : Utilisé pour filtrer les paquets lors de l'affichage en temps réel dans le terminal. Par exemple :

sudo tshark -i eth0 -Y "ip.addr == 192.168.1.1"

Cela afficherait seulement les paquets sur eth0 qui sont destinés à l'adresse IP 192.168.1.1.

Conclusion

Tshark est un outil puissant pour capturer et analyser les paquets réseau via le terminal. En utilisant des options de filtre comme -f, -Y, et -w, vous pouvez sélectionner et enregistrer spécifiquement les paquets qui vous intéressent pour un diagnostic détaillé ou une analyse ultérieure. Assurez-vous d'avoir les permissions appropriées (généralement en exécutant sudo) pour capturer sur les interfaces réseau.