

# Network sniffing

Documentation référence

Ce projet est une initiation concrète au modèle OSI et aux principaux protocoles réseau.

---

## Wireshark

Présentation rapide wireshark

C'est un analyseur réseau qui permet de choisir une interface réseau et d'écouter les trames qui circulent sur le réseau. Il est possible de filtrer ces trames avec différentes options et il est aussi possible d'intégrer les captures dans des scripts bash avec la commande tshark. Wireshark est un outil puissant et polyvalent pour l'analyse réseau, apprécié pour sa capacité à fournir des informations détaillées et précises sur les communications réseau, ce qui le rend essentiel dans de nombreux domaines de l'informatique et des télécommunications.

Quelle est la différence entre une trame et un paquet ?

Dans le contexte des réseaux informatiques, les termes "trame" et "paquet" désignent des unités de données utilisées dans la transmission de l'information, mais ils se situent à des niveaux différents du modèle OSI (Open Systems Interconnection).

Voici les différences principales entre ces deux termes :

### Trame

1. **Niveau du Modèle OSI** : Une trame (ou frame en anglais) se situe au niveau de la couche de liaison de données (Layer 2) du modèle OSI.
2. **Encapsulation** : Une trame encapsule des paquets et des segments provenant des couches supérieures. Elle inclut des en-têtes et des bandes de queue spécifiques à la couche de liaison de données.
3. **Structure** : La structure typique d'une trame inclut un en-tête (header) contenant des informations telles que les adresses MAC source et destination, des données de contrôle, et une bande de queue (trailer) pour la vérification d'erreurs (comme une FCS - Frame Check Sequence).
4. **Utilisation** : Les trames sont utilisées pour le transfert de données sur un segment de réseau local (LAN). Elles sont pertinentes pour les technologies comme Ethernet, Wi-Fi, et autres protocoles de liaison de données.

### Paquet

1. **Niveau du Modèle OSI** : Un paquet (ou packet en anglais) se situe au niveau de la couche réseau (Layer 3) du modèle OSI.
2. **Encapsulation** : Un paquet encapsule des segments provenant de la couche transport. Il inclut un en-tête spécifique à la couche réseau.
3. **Structure** : La structure typique d'un paquet inclut un en-tête (header) contenant des informations telles que les adresses IP source et destination, le type de protocole, et des informations de fragmentation et de routage.
4. **Utilisation** : Les paquets sont utilisés pour le transfert de données sur des réseaux interconnectés, notamment les réseaux longue distance (WAN) et Internet. Les

protocoles comme IP (Internet Protocol) utilisent des paquets pour router les données d'un réseau à un autre.

## Comparaison et Exemple

- **Trame Ethernet** : Dans un réseau Ethernet, une trame peut transporter un paquet IP. L'en-tête de la trame Ethernet contient les adresses MAC des appareils source et destination, tandis que le paquet IP encapsulé contient les adresses IP des appareils source et destination.
- **Routage et Commutation** : Lorsqu'une trame arrive à un routeur, ce dernier extrait le paquet IP pour déterminer l'itinéraire vers le prochain routeur ou l'appareil de destination finale. Les commutateurs (switches) travaillent principalement avec des trames, tandis que les routeurs travaillent principalement avec des paquets.

En résumé, une trame est une unité de données utilisée au niveau de la couche de liaison de données et est pertinente pour la communication locale sur un réseau. Un paquet, quant à lui, est une unité de données utilisée au niveau de la couche réseau, essentielle pour la communication inter-réseaux et le routage.

☐ Qu'est-ce que le format pcap/pcapng ?

Le format pcap (Packet Capture) et son successeur pcapng (Packet Capture Next Generation) sont des formats de fichier utilisés pour stocker les données capturées sur un réseau. Ces formats sont couramment utilisés par des outils d'analyse réseau tels que Wireshark. Voici une description plus détaillée de chacun :

### Format pcap

1. **Origine et Utilisation** : Le format pcap est l'un des formats les plus anciens et les plus couramment utilisés pour la capture de paquets réseau. Il est pris en charge par des outils comme tcpdump, Wireshark, et d'autres logiciels d'analyse réseau.
2. **Structure** : Un fichier pcap contient une séquence de paquets, chaque paquet étant précédé par un en-tête qui contient des informations telles que le timestamp (l'heure de capture), la taille du paquet capturé, et la taille originale du paquet.
3. **Limites** : Le format pcap a certaines limitations, comme un support limité pour les métadonnées et des options d'extension limitées, ce qui peut poser des problèmes pour les captures complexes ou les environnements de capture avancés.

### Format pcapng

1. **Origine et Avantages** : Le format pcapng (Packet Capture Next Generation) a été introduit pour surmonter les limitations du format pcap. Il offre plus de flexibilité et de fonctionnalités.
2. **Structure** : Un fichier pcapng est structuré en blocs, chaque bloc pouvant contenir différents types d'informations, ce qui permet d'inclure des métadonnées supplémentaires. Les principaux types de blocs sont :
  - **Section Header Block** : Contient des informations globales sur la capture.
  - **Interface Description Block** : Décrit les interfaces réseau utilisées pour la capture.
  - **Enhanced Packet Block** : Contient les paquets capturés, avec des informations détaillées comme les options d'interface et les métadonnées.
  - **Simple Packet Block** : Contient uniquement les données de paquets sans métadonnées supplémentaires.
  - **Name Resolution Block** : Permet de stocker des informations de résolution de noms.

### 3. Avantages :

- **Support des Métadonnées** : pcapng permet de stocker des informations supplémentaires comme les noms d'interface, les résolutions de noms, et les commentaires utilisateurs.
- **Extensibilité** : La structure en blocs permet d'ajouter de nouveaux types de blocs à l'avenir, facilitant l'évolution du format sans casser la compatibilité.
- **Support Multiplateforme** : pcapng prend en charge des captures provenant de multiples interfaces et plateformes, ce qui est utile dans des environnements complexes.

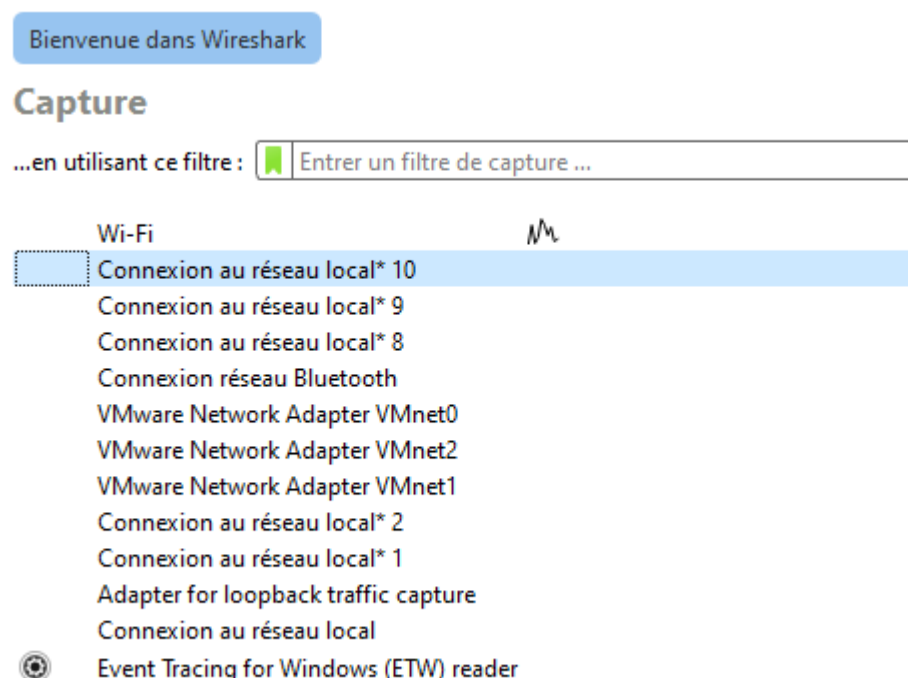
## Utilisation dans Wireshark

- **Compatibilité** : Wireshark supporte les deux formats, pcap et pcapng. Par défaut, Wireshark enregistre les captures en pcapng en raison de ses capacités avancées.
- **Conversion** : Il est possible de convertir des fichiers pcap en pcapng et vice versa à l'aide d'outils de conversion comme editcap, qui fait partie du suite d'outils de Wireshark.

## Conclusion

En résumé, le format pcap est un format traditionnel et largement utilisé pour la capture de paquets réseau, mais il présente des limitations en termes de flexibilité et de métadonnées. Le format pcapng, en revanche, offre une structure plus avancée et extensible, permettant d'inclure des informations supplémentaires et de mieux s'adapter aux besoins des captures réseau modernes et complexes.

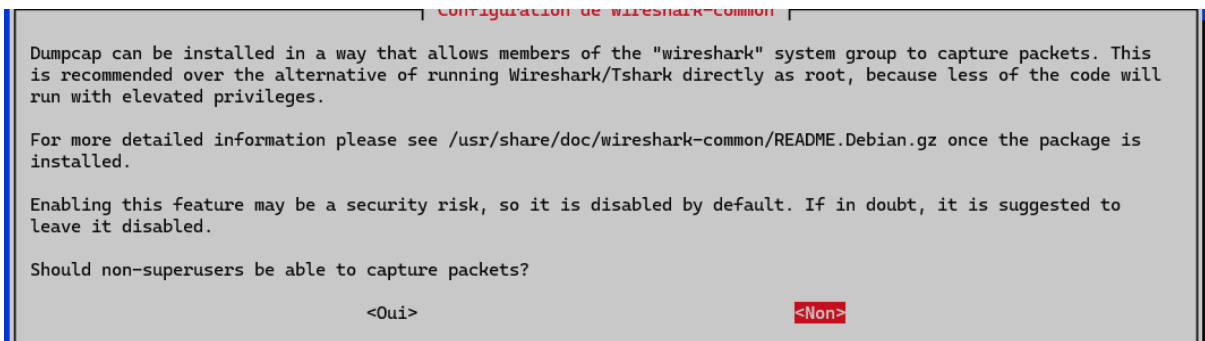
### Wireshark:



### Installation Wireshark sur la VM.

Wireshark et tshark ont besoin de privilèges élevés (root) pour capturer des paquets réseau sur la plupart des interfaces.

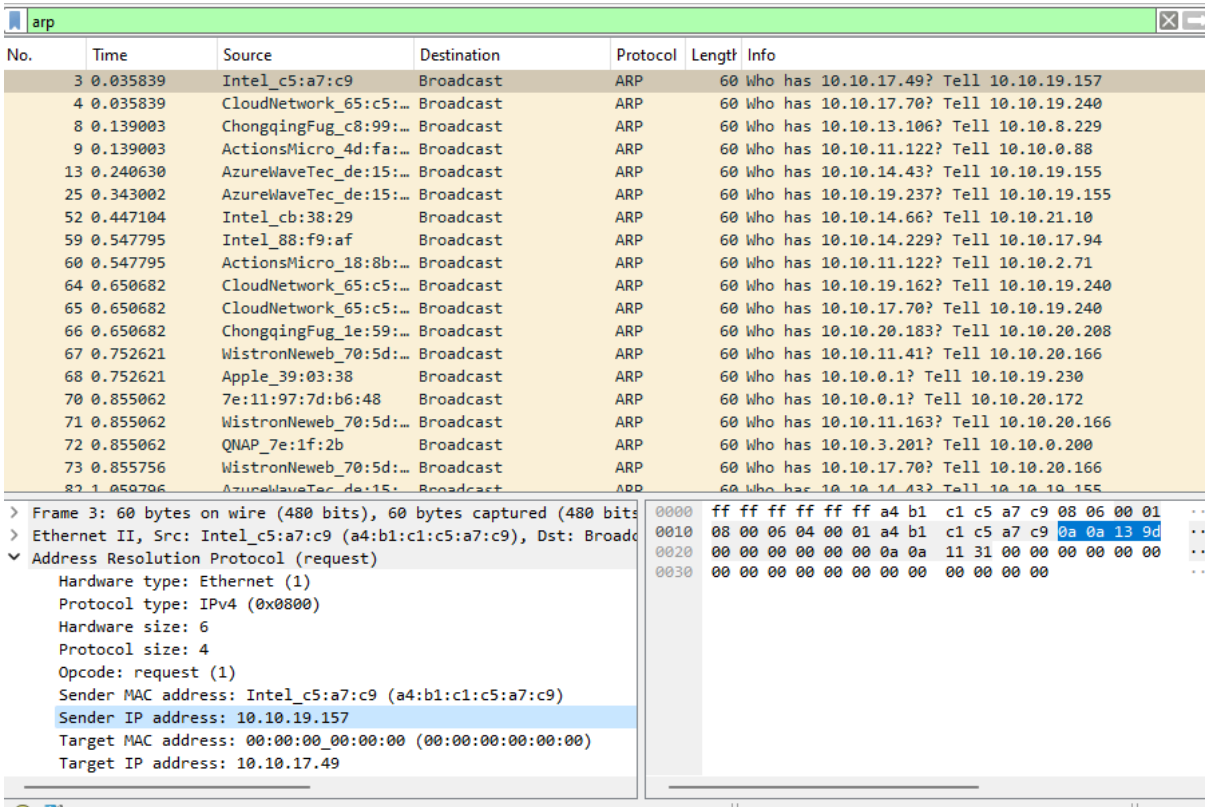
L'option "Oui" permet à des utilisateurs spécifiques d'utiliser Wireshark/tshark sans avoir des droits root, par compte ces utilisateurs doivent être ajoutés au groupe wireshark. Si l'utilisateur est préoccupé par les questions de sécurité et qu'il préfère limiter cette capacité, c'est mieux de choisir "Non".



```
dione@mdl:~$ sudo usermod -aG wireshark dione
```

Là, je dois me déconnecter et me reconnecter pour que les changements prennent effet.

### ARP (Address Resolution Protocol):



### UDP (User Datagram Protocol):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	142.251.37.206	10.10.20.200	UDP	64	443 → 59905 Len=22
2	0.000409	10.10.20.200	142.251.37.206	UDP	75	59905 → 443 Len=33
7	0.130424	10.10.20.29	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
10	0.172731	10.10.20.200	142.251.37.206	UDP	71	59905 → 443 Len=29
11	0.189833	10.10.20.29	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
12	0.200917	142.251.37.206	10.10.20.200	UDP	66	443 → 59905 Len=24
15	0.267924	142.251.37.206	10.10.20.200	UDP	1006	443 → 59905 Len=964
16	0.267924	142.251.37.206	10.10.20.200	UDP	108	443 → 59905 Len=66
17	0.267924	142.251.37.206	10.10.20.200	UDP	76	443 → 59905 Len=34
18	0.267924	142.251.37.206	10.10.20.200	UDP	225	443 → 59905 Len=183
19	0.268371	10.10.20.200	142.251.37.206	UDP	77	59905 → 443 Len=35
20	0.268509	10.10.20.200	142.251.37.206	UDP	73	59905 → 443 Len=31
21	0.296757	142.251.37.206	10.10.20.200	UDP	66	443 → 59905 Len=24
22	0.296993	10.10.20.200	142.251.37.206	UDP	74	59905 → 443 Len=22
23	0.302519	142.251.37.206	10.10.20.200	UDP	64	443 → 59905 Len=22
24	0.303176	10.10.20.200	142.251.37.206	UDP	75	59905 → 443 Len=33
27	0.353748	10.10.20.200	10.10.0.1	DNS	89	Standard query 0x9de9 A v20.events.data.microsoft.com
28	0.361880	10.10.0.1	10.10.20.200	DNS	238	Standard query response 0x9de9 A v20.events.data.microsoft.com CNAME win...
43	0.445743	fa80::d5ah:40bd:317	ff02::1:3	LLMNR	65	Standard query 0x600b ANY DESKTOP-MMMH133U

> Frame 2: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF...  
 ✓ Ethernet II, Src: AzureWaveTec\_a7:80:77 (cc:47:40:a7:80:77), Dst: Intel\_3a:2e:49 (68:05:ca:3a:2e:49)  
 > Destination: Intel\_3a:2e:49 (68:05:ca:3a:2e:49)  
 > Source: AzureWaveTec\_a7:80:77 (cc:47:40:a7:80:77)  
 Type: IPv4 (0x0800)  
 > Internet Protocol Version 4, Src: 10.10.20.200, Dst: 142.251.37.206  
 > User Datagram Protocol, Src Port: 59905, Dst Port: 443  
 > Data (33 bytes)

0000 68 05 ca 3a 2e 49 cc 47 40 a7 80 77 08 00 45 00 h...I.G@...w...E...  
 0010 00 3d f7 74 40 00 00 11 2f a0 0a 0a 14 c8 8e fb .-t@.../.....  
 0020 25 c6 ea 01 01 bb 00 29 b3 17 49 e8 1c aa 54 ac %.....-I...T...  
 0030 c5 50 87 3a 88 4d f3 56 7b 62 52 cd 55 25 3f fe .P...M.V {bR-U%?...  
 0040 4f 2c 18 40 fb db 82 69 10 b8 b0 O, @...i ...

## TCP (Transmission Control Protocol)

No.	Time	Source	Destination	Protocol	Length	Info
29	0.362044	10.10.20.200	51.116.246.104	TCP	66	60489 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
30	0.385223	51.116.246.104	10.10.20.200	TCP	66	443 → 60489 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
31	0.385375	10.10.20.200	51.116.246.104	TCP	54	60489 → 443 [ACK] Seq=1 Ack=1 Win=132352 Len=0
32	0.386549	10.10.20.200	51.116.246.104	TLSv1.2	266	Client Hello (SNI=v20.events.data.microsoft.com)
33	0.410908	51.116.246.104	10.10.20.200	TCP	1514	443 → 60489 [ACK] Seq=1 Ack=213 Win=4194304 Len=1460 [TCP segment of a reassembled PDU]
34	0.410908	51.116.246.104	10.10.20.200	TCP	1514	443 → 60489 [ACK] Seq=1461 Ack=213 Win=4194304 Len=1460 [TCP segment of a reassembled PDU]
35	0.410908	51.116.246.104	10.10.20.200	TCP	1514	443 → 60489 [ACK] Seq=2921 Ack=213 Win=4194304 Len=1460 [TCP segment of a reassembled PDU]
36	0.410908	51.116.246.104	10.10.20.200	TLSv1.2	167	Server Hello, Certificate, Server Key Exchange, Server Hello Done
37	0.411096	10.10.20.200	51.116.246.104	TCP	54	60489 → 443 [ACK] Seq=213 Ack=4494 Win=132352 Len=0
38	0.415611	10.10.20.200	51.116.246.104	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
39	0.445560	51.116.246.104	10.10.20.200	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
40	0.445560	51.116.246.104	10.10.20.200	TLSv1.2	123	Application Data
42	0.445644	10.10.20.200	51.116.246.104	TCP	54	60489 → 443 [ACK] Seq=371 Ack=4614 Win=132352 Len=0
48	0.446986	10.10.20.200	51.116.246.104	TLSv1.2	141	Application Data
49	0.446972	10.10.20.200	51.116.246.104	TLSv1.2	701	Application Data
50	0.447045	10.10.20.200	51.116.246.104	TLSv1.2	92	Application Data
51	0.447084	10.10.20.200	51.116.246.104	TLSv1.2	1477	Application Data
53	0.447627	142.250.200.234	10.10.20.200	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
54	0.468501	51.116.246.104	10.10.20.200	TCP	54	443 → 60489 [ACK] Seq=4614 Ack=3566 Win=4104560 Len=0

> Frame 29: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF...  
 ✓ Ethernet II, Src: AzureWaveTec\_a7:80:77 (cc:47:40:a7:80:77), Dst: Intel\_3a:2e:49 (68:05:ca:3a:2e:49)  
 > Destination: Intel\_3a:2e:49 (68:05:ca:3a:2e:49)  
 > Source: AzureWaveTec\_a7:80:77 (cc:47:40:a7:80:77)  
 Type: IPv4 (0x0800)  
 > Internet Protocol Version 4, Src: 10.10.20.200, Dst: 51.116.246.104  
 > Transmission Control Protocol, Src Port: 60489, Dst Port: 443, Seq: 0, Len: 0

0000 68 05 ca 3a 2e 49 cc 47 40 a7 80 77 08 00 45 00 h...I.G@...w...E...  
 0010 00 34 97 ad 40 00 00 06 1a 68 0a 0a 14 c8 33 74 .4..@...h...3t...  
 0020 f6 68 8c 49 01 bb bf 75 6d 4d 00 00 00 00 02 .h...u mM.....  
 0030 fa f0 10 a9 00 00 02 04 05 b4 01 03 03 08 01 01 .....  
 0040 04 02 ..

Référez d'autres trames ou paquets circulants sur le réseau. Identifiez leurs protocoles et leur fonction.

TLSv1.2

<ul style="list-style-type: none"> <li>Frame 32: 266 bytes on wire (2128 bits), 266 bytes captured (2128 bits) on interface 10</li> <li>Ethernet II, Src: AzureWaveTec-a7:80:77 (cc:47:40:a7:80:77), Dst: Intel_3a:2e:49 (68:05:90:00:00:00) <ul style="list-style-type: none"> <li>Destination: Intel_3a:2e:49 (68:05:90:00:00:00)</li> <li>Source: AzureWaveTec-a7:80:77 (cc:47:40:a7:80:77)</li> <li>Type: IPv4 (0x0800)</li> </ul> </li> <li>Internet Protocol Version 4, Src: 10.10.20.200, Dst: 51.116.246.104</li> <li>Transmission Control Protocol, Src Port: 60489, Dst Port: 443, Seq: 1, Ack: 1, Len: 212</li> <li>Transport Layer Security <ul style="list-style-type: none"> <li>TLSv1.2 Record Layer: Handshake Protocol: Client Hello</li> </ul> </li> </ul>		<pre> 0030 02 05 a8 d2 00 00 16 03 03 00 cf 01 00 00 cb 03 ..... 0040 03 66 59 af 20 9e 51 f8 f4 a9 38 cd da 0f a1 9e ...f4:~Q:~0:~ 0050 f1 50 55 87 16 ff a0 f6 32 f4 18 cc da a1 68 c1 ...0:~0:~0:~Ah~ 0060 20 00 00 20 00 20 c0 2b c0 14 00 00 20 00 00 ...0:~0:~0:~0:~ 0070 c0 28 c0 27 c0 0a c0 99 c0 c1 14 c0 13 9d 90 9c ...0:~0:~0:~0:~ 0080 00 3d 00 3c 00 35 00 2f 01 00 00 7e 00 00 00 22 ...0:~0:~0:~0:~ 0090 00 20 00 00 1d 76 32 00 2e 65 76 65 6e 74 73 2e ...0:~0:~0:~0:~ 00a0 64 61 74 61 2e 6d 69 63 72 6f 73 6f 66 74 2e 63 ...data.mic rosoft.c 00b0 6f 6d 00 05 00 05 01 00 00 00 00 00 0a 00 08 00 ...om:mic rosoft.c 00c0 06 00 1d 00 17 00 18 00 0b 00 02 01 00 00 00 00 ... 00d0 1a 00 18 00 04 08 05 08 06 04 a1 05 01 02 01 04 ... 00e0 03 05 03 02 03 02 02 06 01 06 03 00 23 00 00 ...~0:~0:~0:~#~ 00f0 10 00 0e 00 0c 02 68 32 08 68 74 74 70 2f 31 2e ...~h2-http/1. 0100 31 00 17 00 00 ff 01 00 01 00 00 00 00 00 00 00 ...1..... </pre>
--	--	---

Id	Time	Source	Destination	Protocol	Length	Info
27	0.353748	10.10.20.200	10.10.0.1	DNS	89	Standard query 0x9de9 A v20.events.data.microsoft.com
28	0.361080	10.10.0.1	10.10.20.200	DNS	238	Standard query response 0x9de9 A v20.events.data.microsoft.com CNAME win-global-asimov-leafs-events-d
103	1.398160	10.10.20.200	10.10.0.1	DNS	89	Standard query 0x9e78 A addons-pa.clients6.google.com
104	1.398496	10.10.20.200	10.10.0.1	DNS	89	Standard query 0xfce2 HTTPS addons-pa.clients6.google.com
105	1.402556	10.10.0.1	10.10.20.200	DNS	139	Standard query response 0xfce2 HTTPS addons-pa.clients6.google.com SOA ns1.google.com
106	1.402556	10.10.0.1	10.10.20.200	DNS	105	Standard query response 0x9e78 A addons-pa.clients6.google.com A.142.251.37.234

```

> Frame 28: 238 bytes on wire (1904 bits), 238 bytes captured (1904 bits) on interface 10
Ethernet II, Src: Intel_3a:2e:49 (68:05:ca:3a:2e:49), Dst: AzureWaveTec_a7:80:77 (cc:47:
  > Destination: AzureWaveTec_a7:80:77 (cc:47:40:a7:80:77)
    > Source: Intel_3a:2e:49 (68:05:ca:3a:2e:49)
      Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.10.0.1, Dst: 10.10.20.200
User Datagram Protocol, Src Port: 53, Dst Port: 50634
Domain Name System (response)
0010  00 e0 43 b3 00 00 40 11 0d 7e 0a 0a 00 01 0a 0a  ...5...
0020  14 c8 00 35 c5 ca 00 c6 3d 30 9d e9 81 80 01 01  ...S...
0030  00 03 00 00 00 00 03 76 62 08 06 65 76 65 6e 74  ...v 20:event
0040  75 04 01 61 74 61 09 6d 03 73 72 6f 73 6f 66 74  s:data-microsoft
0050  03 63 6f 6d 00 00 01 00 01 c0 06 00 05 00 01 00  .com...
0060  00 00 6d 00 38 23 77 09 6e 2d 67 6c 6f 62 61 61  -m-BW n-global
0070  2d 61 73 69 6d 6f 76 2b 6c 65 61 66 73 2d 65 76  -aslmov-leafs-ev
0080  61 0e 74 73 2d 64 61 74 61 0e 74 72 61 65 66 68  ents-dat-e-traffic
0090  63 6d 61 6e 61 67 65 72 03 6e 65 74 80 c0 30 00  -manager-net-1
00a0  05 00 01 00 00 00 16 00 35 10 0f 6e 65 64 73 63  -s:onedisc
00b0  6f 60 70 72 64 67 77 63 30 30 12 67 65 72 6d 61  olprdwg 00:germa
00c0  6e 79 77 65 73 74 63 65 6e 74 72 61 6c 08 63 6d  nywestce ntral-cl
00d0  6f 75 64 61 70 70 85 61 7a 75 72 65 c0 26 c0 7f  oudapp-a-zure-8
00e0  00 01 00 01 00 00 00 02 00 04 33 74 f6 68  ...3t-h

```

No.	Time	Source	Destination	Protocol	Length	Info
65	0.650682	CloudNetwork_65:c5:...	Broadcast	ARP	60	who has 10.10.17.70? Tell 10.10.19.240
4	0.035839	CloudNetwork_65:c5:...	Broadcast	ARP	60	who has 10.10.17.70? Tell 10.10.19.240
256	4.029547	Intel_c5:a7:c9	Broadcast	ARP	60	who has 10.10.17.49? Tell 10.10.19.157
208	3.005544	Intel_c5:a7:c9	Broadcast	ARP	60	who has 10.10.17.49? Tell 10.10.19.157
154	2.136259	Intel_c5:a7:c9	Broadcast	ARP	60	who has 10.10.17.49? Tell 10.10.19.157
3	0.035839	Intel_c5:a7:c9	Broadcast	ARP	60	who has 10.10.17.49? Tell 10.10.19.157
158	2.083967	CloudNetwork_de:0f:...	Broadcast	ARP	60	who has 10.10.16.100? Tell 10.10.20.209
176	2.391711	Intel_cb:38:29	Broadcast	ARP	60	who has 10.10.14.66? Tell 10.10.21.10
112	1.469570	Intel_cb:38:29	Broadcast	ARP	60	who has 10.10.14.66? Tell 10.10.21.10
52	0.447104	Intel_cb:38:29	Broadcast	ARP	60	who has 10.10.14.66? Tell 10.10.21.10
147	1.981694	AzureWaveTec_de:15:...	Broadcast	ARP	60	who has 10.10.14.43? Tell 10.10.19.155
82	1.059796	AzureWaveTec_de:15:...	Broadcast	ARP	60	who has 10.10.14.43? Tell 10.10.19.155
13	0.240630	AzureWaveTec_de:15:...	Broadcast	ARP	60	who has 10.10.14.43? Tell 10.10.19.155
229	3.313141	Intel_40:00:dd	Broadcast	ARP	60	who has 10.10.14.229? Tell 10.10.20.223
164	2.288570	Intel_40:00:dd	Broadcast	ARP	60	who has 10.10.14.229? Tell 10.10.20.223
99	1.367043	Intel_40:00:dd	Broadcast	ARP	60	who has 10.10.14.229? Tell 10.10.20.223
259	4.030821	Intel_88:f9:af	Broadcast	ARP	60	who has 10.10.14.229? Tell 10.10.17.94
213	3.108630	Intel_88:f9:af	Broadcast	ARP	60	who has 10.10.14.229? Tell 10.10.17.94
50	0.547705	Intel_88:f9:af	Broadcast	ARP	60	who has 10.10.14.229? Tell 10.10.17.94

Frame 147: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF{...}

Ethernet II, Src: AzureWaveTec\_de:15:07 (10:68:38:de:15:07), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

- Destination: Broadcast (ff:ff:ff:ff:ff:ff)
- Source: AzureWaveTec\_de:15:07 (10:68:38:de:15:07)
  - Type: ARP (0x0806)
  - Padding: 00000000000000000000000000000000
- Address Resolution Protocol (request)

0000	ff ff ff ff ff ff	10 68 38 de 15 07	08 06 00 01	.....h 8.....
0010	08 00 06 04 00 01	10 68 38 de 15 07	0a 0a 13 9b	.....h 8.....
0020	00 00 00 00 00 0a	0e 2b 00 00 00 00 00	00 00 00 00	.....+
0030	00 00 00 00 00 00	00 00 00 00	00 00 00 00	.....+

## UDP hexadécimal:

Wireshark packet capture for UDP. The packet list shows a series of DNS queries and responses. The selected packet (No. 18) is a Standard query response from 10.10.26.187 to 224.0.0.251. The packet details pane shows the Internet Protocol Version 4 header and the UDP header. The packet bytes pane shows the raw data in hexadecimal and ASCII.

## TCP hexadécimal:

Wireshark packet capture for TCP. The packet list shows a series of TCP segments. The selected packet (No. 188) is a TCP segment from 10.10.27.76 to 10.10.0.87. The packet details pane shows the Internet Protocol Version 4 header and the TCP header. The packet bytes pane shows the raw data in hexadecimal and ASCII.

Pour le TCP, essayez de trouver les paquets correspondants aux étapes de connexion entre votre hôte et un serveur. (SYN ACK FIN ...).

## tcp.flags.syn

Wireshark packet capture for TCP. The packet list shows a series of TCP segments. The selected packet (No. 188) is a TCP segment from 10.10.27.76 to 10.10.0.87. The packet details pane shows the Internet Protocol Version 4 header and the TCP header. The packet bytes pane shows the raw data in hexadecimal and ASCII.



## tcp.flags.ack

No.	Time	Source	Destination	Protocol	Length	Info
188	1.836368	10.10.27.76	10.10.0.87	TCP	171	50240 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=509 Len=117 [TCP segment of a reassembled PDU]
189	1.836368	10.10.27.76	10.10.0.87	TCP	171	50245 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=511 Len=117 [TCP segment of a reassembled PDU]
198	1.852809	10.10.0.87	10.10.27.76	TCP	171	8009 → 50240 [PSH, ACK] Seq=1 Ack=118 Win=502 Len=117 [TCP segment of a reassembled PDU]
199	1.852809	10.10.0.87	10.10.27.76	TCP	171	8009 → 50245 [PSH, ACK] Seq=1 Ack=118 Win=330 Len=117 [TCP segment of a reassembled PDU]
200	1.898421	10.10.27.76	10.10.0.87	TCP	54	50240 → 8009 [ACK] Seq=118 Ack=118 Win=508 Len=0
201	1.898806	10.10.27.76	10.10.0.87	TCP	54	50245 → 8009 [ACK] Seq=118 Ack=118 Win=511 Len=0

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
Total Length: 40  
Identification: 0xa918 (43288)  
> 010. .... = Flags: 0x2, Don't fragment  
...0 0000 0000 0000 = Fragment Offset: 0  
Time to Live: 128  
Protocol: TCP (6)  
Header Checksum: 0x2201 [validation disabled]  
[Header checksum status: Unverified]  
Source Address: 10.10.27.76  
Destination Address: 10.10.0.87  
> Transmission Control Protocol, Src Port: 50245, Dst Port: 8009, Seq: 118, Ack: 118, L

0000 d0 c0 bf 42 02 fa cc 47 40 a7 80 77 08 00 45 00 ...B...G @...w...E...  
0010 00 28 a9 18 40 00 80 06 22 01 0a 0a 1b 4d 0a 0a ...@... [....L...  
0020 00 57 c4 45 1f 49 b9 f9 e6 96 bc 79 6a b5 50 10 ...W.E.I...ut...P...  
0030 01 ff d2 d0 00 00 .....N.....p.....  
0040 00 01 82 76 2e d5 62 17 31 f1 33 76 9a 4a cd 67 ...v...b...1 3v...g  
0050 9c 8a 5b cd 9e 79 cf e6 47 3c 69 32 06 ce 12 29 ...[-y...6c12...  
0060 8c c1 05 90 41 bf e2 78 fe 67 d3 28 81 a7 a2 4a ...A...x g...[...  
0070 f9 ca 0c 69 21 1e 1c e0 45 8a 68 bd ae bb 05 31 ...il...E h...1  
0080 8a 24 2a 2b 53 bf e8 32 47 09 45 a6 ea 5b 54 f2 ...\$\*+S...2 G.E...[T...  
0090 e0 0d b3 c2 94 b6 f6 c9 00 10 e3 50 c1 89 f9 33 .....P...3  
00a0 a0 82 3e 52 fb af ad 46 8b 62 3a ...>R...F -b:

Source Address (in use): 4 bytes(s) | Destination: 222 - Affichés : 6 (2.7%) - Perdus : 0 (0.0%) | Profil : Default

## tcp.flags.fin

No.	Time	Source	Destination	Protocol	Length	Info
201	1.898806	10.10.27.76	10.10.0.87	TCP	54	50245 → 8009 [ACK] Seq=118 Ack=118 Win=511 Len=0
200	1.898421	10.10.27.76	10.10.0.87	TCP	54	50240 → 8009 [ACK] Seq=118 Ack=118 Win=508 Len=0
199	1.852809	10.10.0.87	10.10.27.76	TCP	171	8009 → 50245 [PSH, ACK] Seq=1 Ack=118 Win=330 Len=117 [TCP segment of a reassembled PDU]
198	1.852809	10.10.0.87	10.10.27.76	TCP	171	8009 → 50240 [PSH, ACK] Seq=1 Ack=118 Win=502 Len=117 [TCP segment of a reassembled PDU]
189	1.836368	10.10.27.76	10.10.0.87	TCP	171	50245 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=511 Len=117 [TCP segment of a reassembled PDU]
188	1.836368	10.10.27.76	10.10.0.87	TCP	171	50240 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=509 Len=117 [TCP segment of a reassembled PDU]

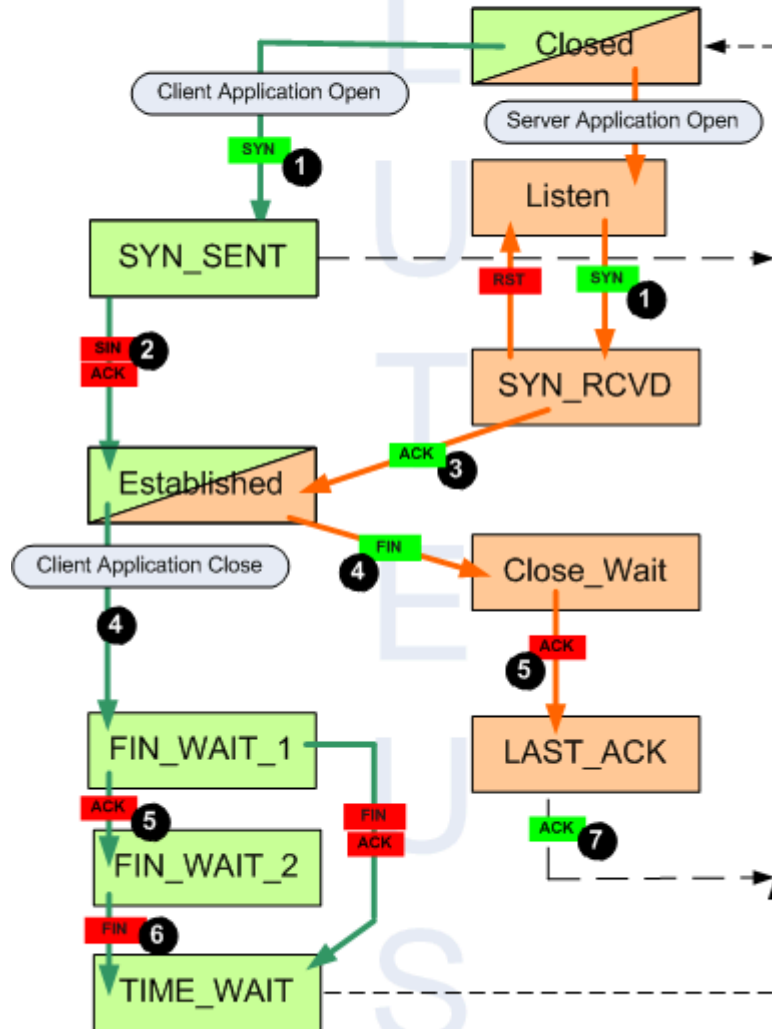
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
Total Length: 157  
Identification: 0xa916 (43286)  
> 010. .... = Flags: 0x2, Don't fragment  
...0 0000 0000 0000 = Fragment Offset: 0  
Time to Live: 128  
Protocol: TCP (6)  
Header Checksum: 0x218e [validation disabled]  
[Header checksum status: Unverified]  
Source Address: 10.10.27.76  
Destination Address: 10.10.0.87  
> Transmission Control Protocol, Src Port: 50240, Dst Port: 8009, Seq: 1, Ack: 1, Len:

0000 d0 c0 bf 42 02 fa cc 47 40 a7 80 77 08 00 45 00 ...B...G @...w...E...  
0010 00 9d a9 16 40 00 80 06 21 8e 0a 0a 1b 4c 0a 0a ...@... [....L...  
0020 00 57 c4 40 1f 49 18 3a fe 75 74 81 0d 0e 50 18 ...W@I...ut...P...  
0030 01 fd 4e 05 00 00 17 03 03 00 70 00 00 00 00 00 ...N.....p.....  
0040 00 01 82 76 2e d5 62 17 31 f1 33 76 9a 4a cd 67 ...v...b...1 3v...g  
0050 9c 8a 5b cd 9e 79 cf e6 47 3c 69 32 06 ce 12 29 ...[-y...6c12...  
0060 8c c1 05 90 41 bf e2 78 fe 67 d3 28 81 a7 a2 4a ...A...x g...[...  
0070 f9 ca 0c 69 21 1e 1c e0 45 8a 68 bd ae bb 05 31 ...il...E h...1  
0080 8a 24 2a 2b 53 bf e8 32 47 09 45 a6 ea 5b 54 f2 ...\$\*+S...2 G.E...[T...  
0090 e0 0d b3 c2 94 b6 f6 c9 00 10 e3 50 c1 89 f9 33 .....P...3  
00a0 a0 82 3e 52 fb af ad 46 8b 62 3a ...>R...F -b:

"tcp.flags.f" is neither a field nor a protocol name. | Paquets : 222 - Affichés : 6 (2.7%) - Perdus : 0 (0.0%) | Profil : Default

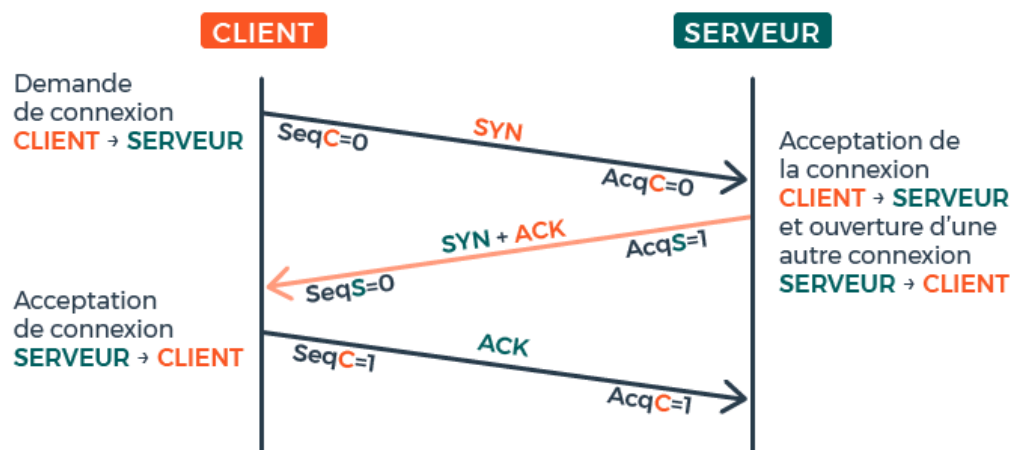


# TCP STATE TRANSITION DIAGRAM



LUTEUS

TCP\_Diagram\_Etat



# Les filtres de capture Wireshark

L'objectif est de filtrer les flux de capture en entrée afin de voir seulement le trafic qui nous intéresse, pour ensuite l'analyser plus facilement, car la capture sera épurée.

Wireshark s'appuie sur la librairie libpcap pour réaliser les filtres de capture comme tcpdump.

## Où faire un filtre de capture ?

Ouvrir Wireshark

Ajouter un filtre (depuis la page d'accueil) en sélectionnant votre interface de capture.

ou

Ajouter un filtre de capture en allant dans le menu « Capture » et ensuite « Options... »

Une nouvelle boîte de dialogue apparaît et vous pouvez ajouter votre filtre, sans omettre de sélectionner votre interface de capture.

## Premier filtre de capture

Utiliser un filtre se basant sur une adresse IP afin de récupérer dans la capture uniquement les paquets correspondants à une adresse IP spécifique.

Utiliser le filtre « host » et d'ajouter l'adresse IP que nous souhaitons voir apparaître dans notre capture.

Pour l'adresse IP « 192.168.1.4 », on obtient le filtre

Si votre filtre est bon, il apparaîtra sur un fond vert tandis qu'en cas d'erreur un fond rouge apparaîtra avec un message.

Lancer votre capture, et vous allez voir dans l'angle en haut à gauche de Wireshark que votre filtre sera affiché.

Dans votre capture, vous voyez seulement les flux en provenance ou bien en réception de cette adresse IP (Les paquets où cette adresse IP est la source ou la destination).

## Gestion des filtres de capture

Pour sauvegarder un filtre, un petit signet, vous indique si le filtre est enregistré ou non suivant la couleur.

En cliquant sur le signet de la gestion des filtres, vous remarquerez que des filtres de capture sont déjà préenregistrés sur Wireshark, comme pour le protocole HTTP.

Pour sauvegarder notre filtre d'affichage, il suffit de cliquer sur le signet puis "Sauvegarder ce filtre".

Une nouvelle ligne apparaît pour donner un nom à notre filtre, afin de l'identifier.

Double cliquer sur « nouveau filtre de capture » pour changer le nom et valider en cliquant sur « OK ». Pour ma part, je nomme ce nouveau filtre "host nas". Le signet passe alors en couleur jaune pour indiquer que la sauvegarde du filtre est effective, et celui-ci se retrouve dans la liste des filtres proposés.

Pour supprimer un filtre de capture, il suffit de cliquer sur le signet et supprimer ce filtre, immédiatement la couleur du signet repasse au vert.

On peut dupliquer un filtre de capture (pratique lorsque vous avez plusieurs filtres similaires à mémoriser).

Pour dupliquer un filtre, il faut cliquer sur le signet comme précédemment, et ensuite "Gérer les filtres de capture".

Cliquez sur l'icône, c'est une copie conforme du filtre existant qui sera créée. Renommer le filtre puis cliquer sur « OK » (même principe que pour la sauvegarde) pour valider. Désormais, j'ai deux fois le filtre "host nas".

## Où sont sauvegardés nos filtres et sous quel format ?

Retourner dans la fenêtre « Gestion des filtres de capture », puis en bas à droite, vous avez la localisation du fichier

Par défaut, il se trouve dans le profil de l'utilisateur au sein du répertoire

Le fichier se nomme « cfilters » qui est un simple fichier texte, donc vous pouvez aussi passer par ici pour gérer vos filtres de capture. On voit bien la correspondance entre le contenu de ce fichier et l'interface de Wireshark.

## Les opérateurs des filtres

Ajouter des opérateurs de plusieurs filtres pour affiner notre analyse sur un protocole, par exemple. Autrement dit, cela va permettre de créer des conditions basées sur des opérateurs tels que "ET" ou "OU".

## Exemples de filtres de capture Wireshark

Filtres de captures	Exemples
Sur une adresse MAC	ether host 00:00:5e:00:53:00
Sur un réseau	net 192.168.1.0/24
Sur le protocole QUIC	udp port 443
Exclure le trafic de type broadcast	not broadcast
Le protocole ICMP	icmp
Le protocole OSPF	ip proto ospf
Sur un numéro de VLAN	vlan 10
Sur le trafic HTTPS	tcp port 443

## Partie 2

```

GNU nano 7.2 /etc/hosts
127.0.0.1    localhost
127.0.1.1    mdl
172.16.50.133 nas
# The following lines are desirable for IPv6 capable hosts
::1    localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

```

```

GNU nano 7.2 /etc/resolv.conf
domain localdomain
search localdomain
nameserver 172.16.50.2

```

VM server:

DHCP (Dynamic Host Configuration Protocol)

DHCP Settings

Network: vmnet2

Subnet IP: 172.16.50.0

Subnet mask: 255.255.255.0

Starting IP address: 172.16.50.128

Ending IP address: 172.16.50.254

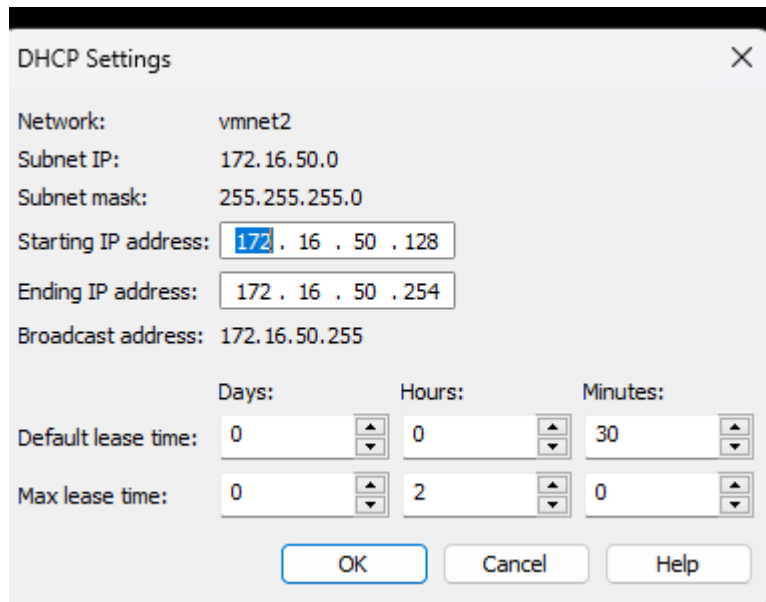
Broadcast address: 172.16.50.255

Default lease time: Days: 0 Hours: 0 Minutes: 30

Max lease time: Days: 0 Hours: 2 Minutes: 0

OK Cancel Help

## VM client:



DHCP Settings

Network: vmnet2

Subnet IP: 172.16.50.0

Subnet mask: 255.255.255.0

Starting IP address: 172.16.50.128

Ending IP address: 172.16.50.254

Broadcast address: 172.16.50.255

Days: Hours: Minutes:

Default lease time: 0 0 30

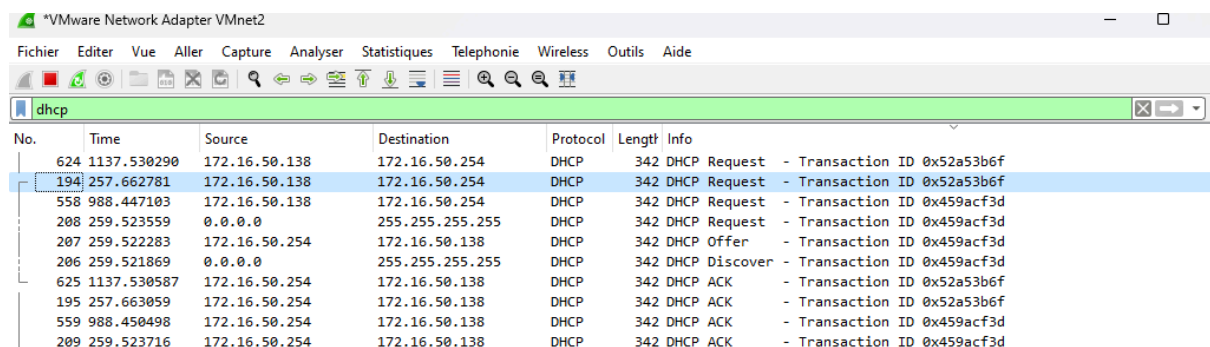
Max lease time: 0 2 0

OK Cancel Help

**sudo dhclient -r libère le DHCP l'adresse IP actuellement attribuée par le serveur DHCP.**

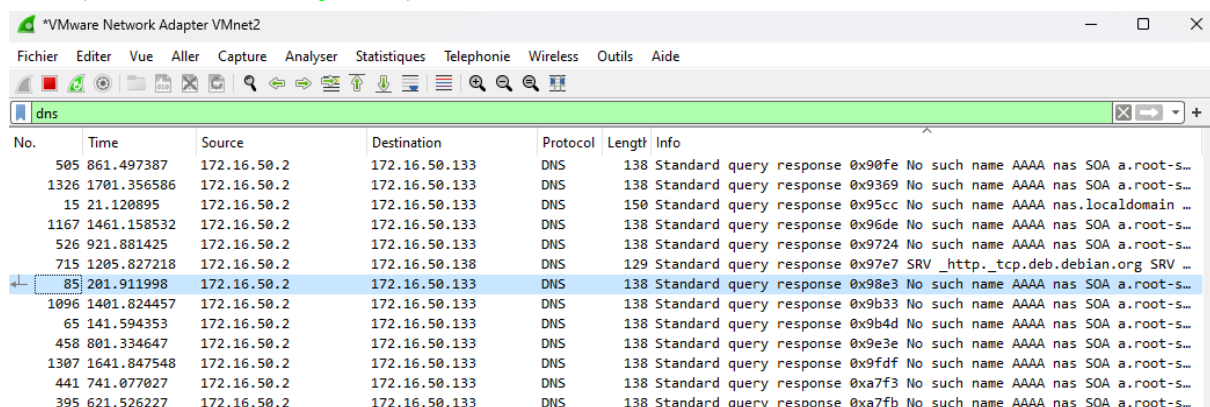
**sudo dhclient = envoie une nouvelle demande de DHCP pour obtenir une nouvelle adresse IP.**

## DHCP Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
624	1137.530290	172.16.50.138	172.16.50.254	DHCP	342	DHCP Request - Transaction ID 0x52a53b6f
194	257.662781	172.16.50.138	172.16.50.254	DHCP	342	DHCP Request - Transaction ID 0x52a53b6f
558	988.447103	172.16.50.138	172.16.50.254	DHCP	342	DHCP Request - Transaction ID 0x459acf3d
208	259.523559	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x459acf3d
207	259.522283	172.16.50.138	172.16.50.138	DHCP	342	DHCP Offer - Transaction ID 0x459acf3d
206	259.521869	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x459acf3d
625	1137.530587	172.16.50.138	172.16.50.138	DHCP	342	DHCP ACK - Transaction ID 0x52a53b6f
195	257.663059	172.16.50.254	172.16.50.138	DHCP	342	DHCP ACK - Transaction ID 0x52a53b6f
559	988.450498	172.16.50.254	172.16.50.138	DHCP	342	DHCP ACK - Transaction ID 0x459acf3d
209	259.523716	172.16.50.254	172.16.50.138	DHCP	342	DHCP ACK - Transaction ID 0x459acf3d

## DNS (Domain Name System)



No.	Time	Source	Destination	Protocol	Length	Info
505	861.497387	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x90fe No such name AAAA nas SOA a.root-s...
1326	1701.356586	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x9369 No such name AAAA nas SOA a.root-s...
15	21.120895	172.16.50.2	172.16.50.133	DNS	150	Standard query response 0x95cc No such name AAAA nas.localdomain ...
1167	1461.158532	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x96de No such name AAAA nas SOA a.root-s...
526	921.881425	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x9724 No such name AAAA nas SOA a.root-s...
715	1205.827218	172.16.50.2	172.16.50.138	DNS	129	Standard query response 0x97e7 SRV _http._tcp.deb.debian.org SRV ...
85	201.911998	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x98e3 No such name AAAA nas SOA a.root-s...
1096	1401.824457	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x9b33 No such name AAAA nas SOA a.root-s...
65	141.594353	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x9b4d No such name AAAA nas SOA a.root-s...
458	801.334647	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x9e3e No such name AAAA nas SOA a.root-s...
1307	1641.847548	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x9fdf No such name AAAA nas SOA a.root-s...
441	741.077027	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0xa7f3 No such name AAAA nas SOA a.root-s...
395	621.526227	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0xa7fb No such name AAAA nas SOA a.root-s...

## mDNS:

**sudo apt-get install avahi-daemon**

**sudo systemctl start avahi-daemon**

No.	Time	Source	Destination	Protocol	Length	Info
159	292.845984	172.16.50.138	224.0.0.251	MDNS	247	Standard query 0x0000 ANY 4.e.e.2.d.9.e.f.f.9.2.c.0.2.0.0.0.0...
161	293.097550	172.16.50.138	224.0.0.251	MDNS	247	Standard query 0x0000 ANY 4.e.e.2.d.9.e.f.f.9.2.c.0.2.0.0.0.0...
163	293.348023	172.16.50.138	224.0.0.251	MDNS	247	Standard query 0x0000 ANY 4.e.e.2.d.9.e.f.f.9.2.c.0.2.0.0.0.0...
160	292.846570	172.16.50.1	224.0.0.251	MDNS	247	Standard query 0x0000 ANY 4.e.e.2.d.9.e.f.f.9.2.c.0.2.0.0.0.0...
162	293.097901	172.16.50.1	224.0.0.251	MDNS	247	Standard query 0x0000 ANY 4.e.e.2.d.9.e.f.f.9.2.c.0.2.0.0.0.0...
164	293.348330	172.16.50.1	224.0.0.251	MDNS	247	Standard query 0x0000 ANY 4.e.e.2.d.9.e.f.f.9.2.c.0.2.0.0.0.0...
26	33.596048	172.16.50.1	224.0.0.251	MDNS	71	Standard query 0x0000 ANY dione.local, "QM" question
27	33.597091	fe80::2328:bc33:393...	ff02::fb	MDNS	91	Standard query 0x0000 ANY dione.local, "QM" question
125	242.687217	172.16.50.1	224.0.0.251	MDNS	71	Standard query 0x0000 ANY dione.local, "QM" question
126	242.688362	fe80::2328:bc33:393...	ff02::fb	MDNS	91	Standard query 0x0000 ANY dione.local, "QM" question
64	80.288298	172.16.50.1	224.0.0.251	MDNS	85	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question
65	80.289724	fe80::2328:bc33:393...	ff02::fb	MDNS	105	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question
62	79.273836	172.16.50.1	224.0.0.251	MDNS	85	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QU" question

**mDNS**, ou multicast Domain Name System est un protocole qui nous permet de bénéficier des fonctionnalités de DNS, sans avoir un serveur DNS sur le réseau. Il est particulièrement utilisé dans les petits réseaux, comme ceux des domiciles ou des petits bureaux, pour permettre aux appareils de se découvrir et de communiquer entre eux de manière automatique.

**Message FTP:**

**Serveur:**

`sudo apt update`

`sudo apt install vsftpd`

`sudo systemctl start vsftpd`

**Client:**

`sudo apt install ftp`

`ftp + ip de mon server`

No.	Time	Source	Destination	Protocol	Length	Info
272	568.287305	172.16.50.133	172.16.50.138	FTP	78	Request: USER dione
238	544.427249	172.16.50.133	172.16.50.138	FTP	76	Request: USER mdp
283	570.346723	172.16.50.138	172.16.50.133	FTP	87	Response: EPRT
284	570.346834	172.16.50.138	172.16.50.133	FTP	110	Response: PASV
282	570.346668	172.16.50.138	172.16.50.133	FTP	81	Response: 211-Features:
280	570.345823	172.16.50.138	172.16.50.133	FTP	85	Response: 215 UNIX Type: L8
236	541.757462	172.16.50.138	172.16.50.133	FTP	86	Response: 220 (vsFTPd 3.0.3)
270	561.123172	172.16.50.138	172.16.50.133	FTP	86	Response: 220 (vsFTPd 3.0.3)
262	558.796264	172.16.50.138	172.16.50.133	FTP	80	Response: 221 Goodbye.
277	570.344844	172.16.50.138	172.16.50.133	FTP	89	Response: 230 Login successful.
240	544.428052	172.16.50.138	172.16.50.133	FTP	100	Response: 331 Please specify the password.
274	568.287787	172.16.50.138	172.16.50.133	FTP	100	Response: 331 Please specify the password.
258	553.547532	172.16.50.138	172.16.50.133	FTP	88	Response: 530 login incorrect.

**Message SMB:**

`sudo apt update`

`sudo apt install samba`

```
[share]
  path = /srv/samba/share
  available = yes
  valid users = dione
  read only = no
  browsable = yes
  public = yes
  writable = yes
```

## Les droits:

```
sudo mkdir -p /srv/samba/share
```

```
sudo chown -R nobody:nogroup /srv/samba/share
```

```
sudo chmod -R 0775 /srv/samba/share
```

## Ajout l'utilisateur (dione)

```
sudo smbpasswd -a dione
```

```
sudo systemctl restart smbd
```

## Installer smbclient sur le client (VM2)

```
sudo apt update
```

```
sudo apt install smbclient
```

J'ai utiliser **smbclient** pour accéder au partage SMB sur le serveur :

```
smbclient //172.16.50.138/share -U dione
```

```
laplateforme@nas:~$ smbclient //172.16.50.138/share -U dione
Password for [WORKGROUP\dione]:
Try "help" to get a list of possible commands.
smb: \> |
```

## Message SMB wireshark:

smb2						
No.	Time	Source	Destination	Protocol	Length	Info
92	85.000929	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
97	90.041302	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
100	95.079640	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
107	100.018874	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
114	105.058136	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
118	109.994772	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
121	115.049538	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
125	119.989673	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
140	125.029617	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
147	130.074417	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
150	135.010366	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
153	140.055728	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
158	145.010754	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response

## Messages HTTPS:

http						
No.	Time	Source	Destination	Protocol	Length	Info
292	363.360247	172.16.50.138	146.75.54.132	HTTP	281	GET /debian-security/dists/bookworm-security/InRelease HTTP/1.1
299	363.389970	172.16.50.138	146.75.54.132	HTTP	266	GET /debian/dists/bookworm-updates/InRelease HTTP/1.1
291	363.360174	172.16.50.138	146.75.54.132	HTTP	258	GET /debian/dists/bookworm/InRelease HTTP/1.1
295	363.388450	146.75.54.132	172.16.50.138	HTTP	378	HTTP/1.1 304 Not Modified
296	363.388486	146.75.54.132	172.16.50.138	HTTP	339	HTTP/1.1 304 Not Modified
304	363.440177	146.75.54.132	172.16.50.138	HTTP	338	HTTP/1.1 304 Not Modified

## Message TLSv1.2:

```
sudo apt update
```



**sudo apt install openssl**

**openssl s\_client -connect 172.16.50.138:443 -tls1\_2** = permet d'établir une connexion avec le serveur à l'adresse spécifiée sur le port 443 (le port HTTPS par défaut) en utilisant TLSv1.2.

```
SSL handshake has read 5 bytes and written 188 bytes
Verification: OK
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : 0000
    Session-ID:
    Session-ID-ctx:
    Master-Key:
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1717764850
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
```

No.	Time	Source	Destination	Protocol	Length	Info
1382	1896.852336	194.177.211.216	172.16.50.138	TLSv1.3	117	Application Data, Application Data
729	876.684089	194.177.211.216	172.16.50.138	TLSv1.3	1369	Application Data, Application Data, Application Data
741	876.962340	194.177.211.216	172.16.50.138	TLSv1.3	262	Application Data, Application Data, Application Data
1365	1891.443383	194.177.211.216	172.16.50.138	TLSv1.3	1369	Application Data, Application Data, Application Data
1377	1891.647102	194.177.211.216	172.16.50.138	TLSv1.3	791	Application Data, Application Data, Application Data, Application...
731	876.685261	172.16.50.138	194.177.211.216	TLSv1.3	60	Change Cipher Spec
1367	1891.446495	172.16.50.138	194.177.211.216	TLSv1.3	60	Change Cipher Spec
1738	2446.401649	172.16.50.133	172.16.50.138	TLSv1	254	Client Hello
725	876.533166	172.16.50.138	194.177.211.216	TLSv1.3	571	Client Hello (SNI=appstream.debian.org)
1361	1891.354405	172.16.50.138	194.177.211.216	TLSv1.3	571	Client Hello (SNI=appstream.debian.org)
727	876.684054	194.177.211.216	172.16.50.138	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
1363	1891.443349	194.177.211.216	172.16.50.138	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data

**SSL (Secure Sockets Layer) et TLS (Transport Layer Security)** sont tous deux des protocoles de sécurité utilisés pour sécuriser les communications sur un réseau, généralement sur Internet. TLS est essentiellement une évolution de SSL.

Interprétez les paquets capturés avec les spécifications des protocoles.

**DHCP (Dynamic Host Configuration Protocol)**

**Fonction** : Assigne dynamiquement des adresses IP et d'autres paramètres de configuration réseau aux appareils sur un réseau.

**Port** : Utilise les ports UDP 67 (serveur) et 68 (client).

## DNS (Domain Name System)

**Fonction** : Résout les noms de domaine en adresses IP, permettant aux utilisateurs d'accéder aux ressources Internet via des noms de domaine compréhensibles.

**Port** : Utilise le port UDP 53 (et parfois TCP 53 pour des requêtes plus volumineuses).

## mDNS (Multicast DNS)

**Fonction** : Permet la résolution de noms de domaine en adresses IP sur les petits réseaux sans configuration spéciale, souvent utilisé dans des réseaux locaux.

**Port** : Utilise le port UDP 5353.

## SSL (Secure Sockets Layer)

**Fonction** : Ancien protocole pour sécuriser les communications sur un réseau. Il chiffre les données pour assurer la confidentialité et l'intégrité.

**Ports** : Utilise généralement le port TCP 443 (HTTPS).

## FTP (File Transfer Protocol)

**Fonction** : Transfert de fichiers entre un client et un serveur sur un réseau.

**Ports** : Utilise les ports TCP 20 (données) et 21 (commandes).

## SMB (Server Message Block)

**Fonction** : Protocole de partage de fichiers qui permet le partage de fichiers, d'imprimantes et de communications entre nœuds d'un réseau.

**Port** : Utilise le port TCP 445.

## HTTPS (Hypertext Transfer Protocol Secure)

**Fonction** : Version sécurisée de HTTP utilisant TLS pour chiffrer les communications entre le navigateur et le serveur Web.

**Port** : Utilise le port TCP 443.

### TLShv1.2 (Transport Layer Security version 1.2)

**Fonction** : Protocole de sécurité pour chiffrer les communications sur un réseau, successeur de SSL. TLSv1.2 est une version spécifique avec des améliorations de sécurité par rapport aux versions précédentes.

**Ports** : Utilise les ports TCP 443 (HTTPS) et d'autres selon les applications spécifiques.

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 ?

#### FTP sans TLS

Toutes les données échangées, y compris les noms d'utilisateur, les mots de passe, et les fichiers transférés, sont envoyées en clair. Les informations sensibles sont exposées et facilement récupérables.

#### SSL/TLS (FTPS/HTTPS)

Toutes les données échangées sont chiffrées, rendant difficile pour un attaquant de lire ou de modifier les informations interceptées. Pour sécuriser les échanges FTP, il est recommandé d'utiliser FTPS (FTP sécurisé par SSL/TLS) ou SFTP (FTP sur SSH), qui chiffrent les données échangées.

## Partie 3

---

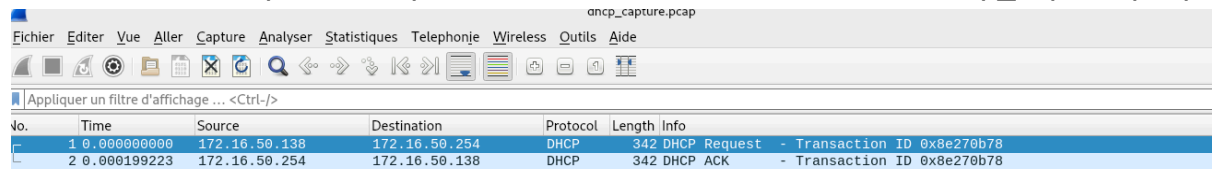
```
sudo apt-get update
sudo apt-get install tshark
mkdir /home/dione/Documents
sudo chmod 755 /home/dione/Documents
sudo usermod -aG wireshark dione
su - dione
```

#### #Installation d'interface graphique

```
sudo apt update && sudo apt install -y task-gnome-desktop &&
sudo systemctl set-default graphical.target && sudo reboot
```

## tshark DHCP

tshark -i ens33 -f "port 67 or port 68" -w /home/dione/Documents/dhcp\_capture.pcap

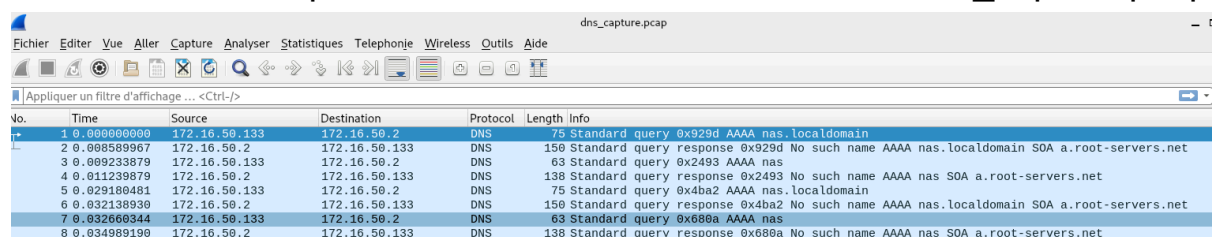


The screenshot shows the Wireshark interface with a DHCP capture. The packet list shows two packets: a DHCP Request (port 68) and a DHCP ACK (port 67). The packet details pane shows the DHCP message structure.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.50.138	172.16.50.254	DHCP	342	DHCP Request - Transaction ID 0x8e270b78
2	0.000199223	172.16.50.254	172.16.50.138	DHCP	342	DHCP ACK - Transaction ID 0x8e270b78

## tshark DNS

tshark -i ens33 -f "port 53" -w /home/dione/Documents/dns\_capture.pcap

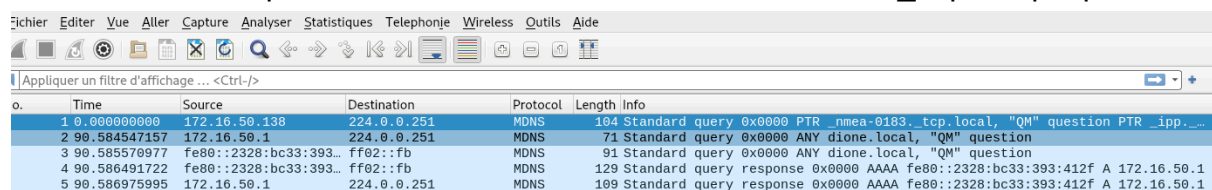


The screenshot shows the Wireshark interface with a DNS capture. The packet list shows several DNS queries and responses. The packet details pane shows the DNS message structure.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.50.133	172.16.50.2	DNS	75	Standard query 0x929d AAAA nas.localdomain
2	0.000589967	172.16.50.2	172.16.50.133	DNS	150	Standard query response 0x929d No such name AAAA nas.localdomain SOA a.root-servers.net
3	0.000923879	172.16.50.133	172.16.50.2	DNS	63	Standard query 0x2493 AAAA nas
4	0.011239079	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x2493 No such name AAAA nas SOA a.root-servers.net
5	0.029180481	172.16.50.133	172.16.50.2	DNS	75	Standard query 0x4ba2 AAAA nas.localdomain
6	0.032138939	172.16.50.2	172.16.50.133	DNS	150	Standard query response 0x4ba2 No such name AAAA nas.localdomain SOA a.root-servers.net
7	0.032660344	172.16.50.133	172.16.50.2	DNS	63	Standard query 0x680a AAAA nas
8	0.034989190	172.16.50.2	172.16.50.133	DNS	138	Standard query response 0x680a No such name AAAA nas SOA a.root-servers.net

## tshark mDNS

tshark -i ens33 -f "port 5353" -w /home/dione/Documents/mdns\_capture.pcap

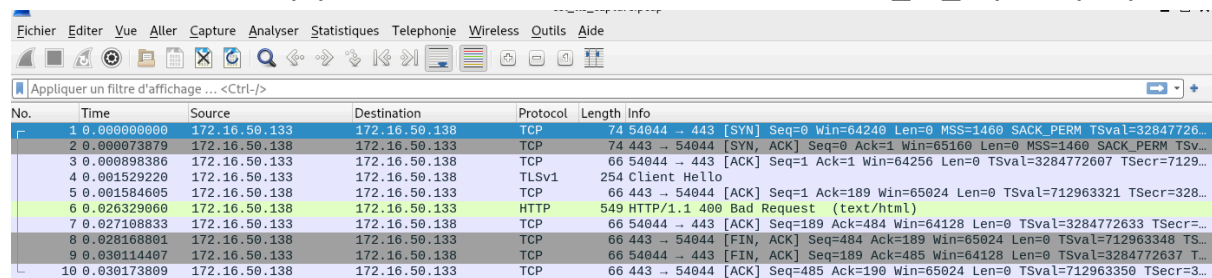


The screenshot shows the Wireshark interface with an mDNS capture. The packet list shows several mDNS queries and responses. The packet details pane shows the mDNS message structure.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.50.138	224.0.0.251	MDNS	104	Standard query 0x0000 PTR _nmea-0183._tcp.local, "QM" question PTR _ipp._...
2	0.000073879	172.16.50.1	224.0.0.251	MDNS	71	Standard query 0x0000 ANY dione.local, "QM" question
3	0.000585709	fe80::2328:bc33:393... ff02::fb		MDNS	91	Standard query 0x0000 ANY dione.local, "QM" question
4	0.000585709	fe80::2328:bc33:393... ff02::fb		MDNS	129	Standard query response 0x0000 AAAA fe80::2328:bc33:393:412f A 172.16.50.1
5	0.000585709	172.16.50.1	224.0.0.251	MDNS	109	Standard query response 0x0000 AAAA fe80::2328:bc33:393:412f A 172.16.50.1

## tshark SSL/TLSv1.2

tshark -i ens33 -f "tcp port 443" -w /home/dione/Documents/ssl\_tls\_capture.pcap

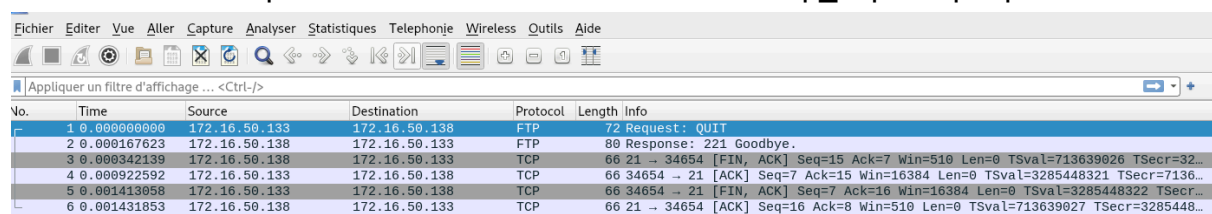


The screenshot shows the Wireshark interface with an SSL/TLS capture. The packet list shows several TCP and TLSv1.2 packets. The packet details pane shows the TLS message structure.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.50.133	172.16.50.138	TCP	74	54044 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=32847726...
2	0.000073879	172.16.50.138	172.16.50.133	TCP	74	443 → 54044 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSv...
3	0.000098386	172.16.50.133	172.16.50.138	TCP	66	54044 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3284772607 TSecr=7129...
4	0.000152922	172.16.50.133	172.16.50.138	TLSv1	254	Client Hello
5	0.000158460	172.16.50.138	172.16.50.133	TCP	66	443 → 54044 [ACK] Seq=1 Ack=189 Win=65024 Len=0 TSval=712963321 TSecr=328...
6	0.026329969	172.16.50.138	172.16.50.133	HTTP	549	HTTP/1.1 400 Bad Request (text/html)
7	0.027188833	172.16.50.133	172.16.50.138	TCP	66	54044 → 443 [ACK] Seq=189 Ack=484 Win=64128 Len=0 TSval=3284772633 TSecr=...
8	0.028168801	172.16.50.138	172.16.50.133	TCP	66	443 → 54044 [FIN, ACK] Seq=464 Ack=189 Win=65024 Len=0 TSval=712963348 TS...
9	0.030114407	172.16.50.133	172.16.50.138	TCP	66	54044 → 443 [FIN, ACK] Seq=189 Ack=485 Win=64128 Len=0 TSval=3284772637 T...
10	0.030173809	172.16.50.138	172.16.50.133	TCP	66	443 → 54044 [ACK] Seq=485 Ack=190 Win=65024 Len=0 TSval=712963350 TSecr=3...

## tshark FTP

tshark -i ens33 -f "port 21" -w /home/dione/Documents/ftp\_capture.pcap



The screenshot shows the Wireshark interface with an FTP capture. The packet list shows several FTP and TCP packets. The packet details pane shows the FTP message structure.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.50.133	172.16.50.138	FTP	72	Request: QUIT
2	0.000167623	172.16.50.138	172.16.50.133	FTP	80	Response: 221 Goodbye.
3	0.000342139	172.16.50.138	172.16.50.133	TCP	66	21 → 34654 [FIN, ACK] Seq=15 Ack=7 Win=510 Len=0 TSval=713639026 TSecr=32...
4	0.000922592	172.16.50.133	172.16.50.138	TCP	66	34654 → 21 [ACK] Seq=7 Ack=15 Win=16384 Len=0 TSval=3285448321 TSecr=7136...
5	0.001413058	172.16.50.133	172.16.50.138	TCP	66	34654 → 21 [FIN, ACK] Seq=7 Ack=16 Win=16384 Len=0 TSval=3285448322 TSecr...
6	0.001431853	172.16.50.138	172.16.50.133	TCP	66	21 → 34654 [ACK] Seq=16 Ack=8 Win=510 Len=0 TSval=713639027 TSecr=3285448...

## tshark SMB

tshark -i ens33 -f "port 445" -w /home/dione/Documents/smb\_capture.pcap

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.50.133	172.16.50.138	SMB2	138	KeepAlive Request
2	0.000241001	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
3	0.001094028	172.16.50.133	172.16.50.138	TCP	66	39968 → 445 [ACK] Seq=73 Ack=73 Win=501 Len=0 TSval=3285650113 TSecr=713840819
4	4.940095133	172.16.50.133	172.16.50.138	SMB2	138	KeepAlive Request
5	4.940405444	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
6	4.941346255	172.16.50.133	172.16.50.138	TCP	66	39968 → 445 [ACK] Seq=145 Ack=145 Win=501 Len=0 TSval=3285655053 TSecr=713845759
7	9.983473235	172.16.50.133	172.16.50.138	SMB2	138	KeepAlive Request
8	9.983998929	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
9	9.984523561	172.16.50.133	172.16.50.138	TCP	66	39968 → 445 [ACK] Seq=217 Ack=217 Win=501 Len=0 TSval=3285660097 TSecr=713850803
10	14.922565759	172.16.50.133	172.16.50.138	SMB2	138	KeepAlive Request
11	14.922832358	172.16.50.138	172.16.50.133	SMB2	138	KeepAlive Response
12	14.923343565	172.16.50.133	172.16.50.138	TCP	66	39968 → 445 [ACK] Seq=289 Ack=289 Win=501 Len=0 TSval=3285665035 TSecr=713855742
13	19.967472983	172.16.50.133	172.16.50.138	SMB2	138	KeepAlive Request

## Les différentes options que vous utiliser:

pour le DHCP:

**-i ens33**: Spécifie l'interface réseau à écouter.

**-f "port 67 or port 68"**: Utilise un filtre BPF (Berkeley Packet Filter) pour capturer uniquement le trafic DHCP (port 67 pour les serveurs DHCP et port 68 pour les clients DHCP).

**-w /home/dione/Documents/dhcp\_capture.pcap**: Indique le fichier de sortie où les paquets capturés seront enregistrés.