

# Network Sniffing : WireShark

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

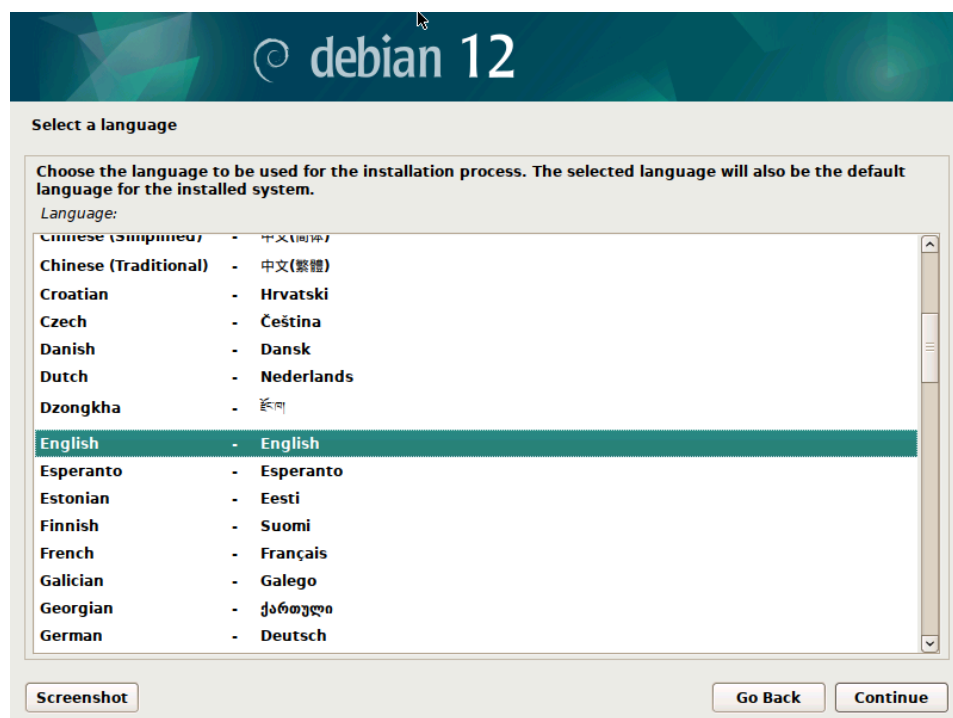
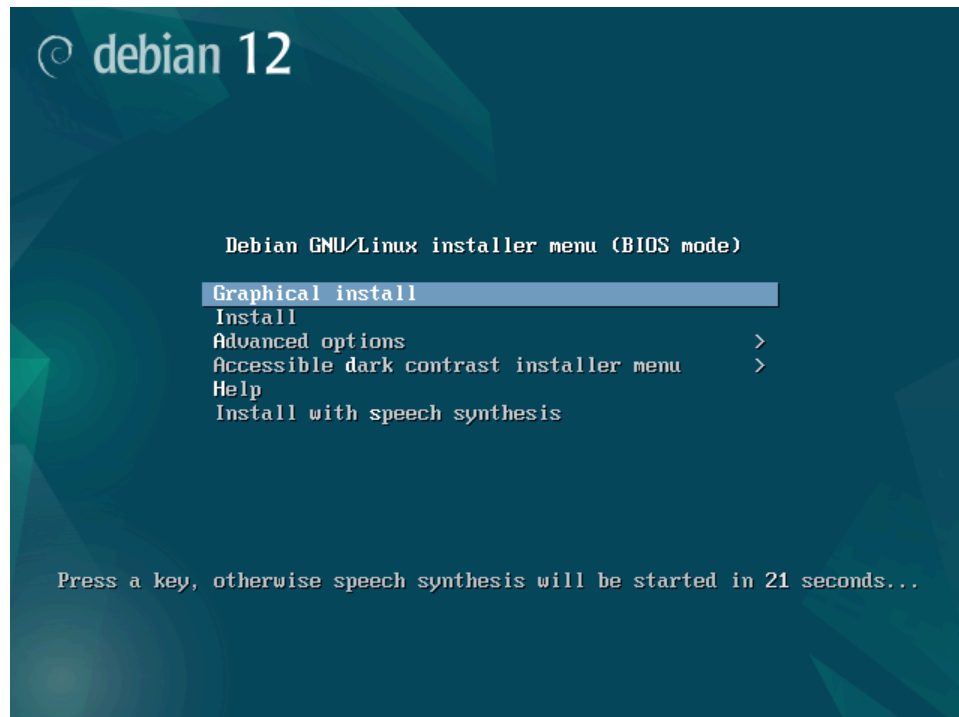
- **Une trame** est utilisée pour envoyer des données entre un seul réseau.
- **Un paquet** est utilisé pour envoyer des paquets d'un réseau à un autre puis vers un périphérique spécifique sur ce réseau.
- Le format **PCAP** (Packet Capture) est un format de fichier standard utilisé pour enregistrer des paquets de données capturés sur un réseau.
  - **Utilisation** : Utilisé par des outils comme **Wireshark**, **tcpdump**, et autres analyseurs de paquets pour stocker et analyser les données réseau.
- Le format **PCAPNG** (Packet Capture Next Generation) est une version plus récente et plus flexible du format PCAP. Il offre des améliorations significatives en termes de fonctionnalités et de flexibilité.
  - **Utilisation** : Le format PCAPNG (Packet Capture Next Generation) est conçu pour fournir une flexibilité et une extensibilité supérieures par rapport au format PCAP classique.

## En résumé :

- **PCAP** : Utilisé pour des captures simples, des analyses de base, et des scénarios où la compatibilité avec des outils plus anciens est nécessaire.
- **PCAPNG** : Préféré pour des analyses réseau avancées, des diagnostics détaillés, des environnements multi-interface, et des applications nécessitant des métadonnées enrichies et des horodatages précis.

En fonction de la complexité du réseau et des besoins d'analyse, on choisira l'un ou l'autre format pour capturer et analyser le trafic réseau efficacement.

## I – MISE EN PLACE D'UNE VM DEBIAN (AVEC GUI)





### Configure the network

Please enter the hostname for this system.

The hostname is a single word that identifies your system to the network. If you don't know what your hostname should be, consult your network administrator. If you are setting up your own home network, you can make something up here.

Hostname:



### Set up users and passwords

You need to set a password for 'root', the system administrative account. A malicious or unqualified user with root access can have disastrous results, so you should take care to choose a root password that is not easy to guess. It should not be a word found in dictionaries, or a word that could be easily associated with you.

A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.

The root user should not have an empty password. If you leave this empty, the root account will be disabled and the system's initial user account will be given the power to become root using the "sudo" command.

Note that you will not be able to see the password as you type it.

Root password:

☐ Show Password in Clear



### Set up users and passwords

A user account will be created for you to use instead of the root account for non-administrative activities.

Please enter the real name of this user. This information will be used for instance as default origin for emails sent by this user as well as any program which displays or uses the user's real name. Your full name is a reasonable choice.

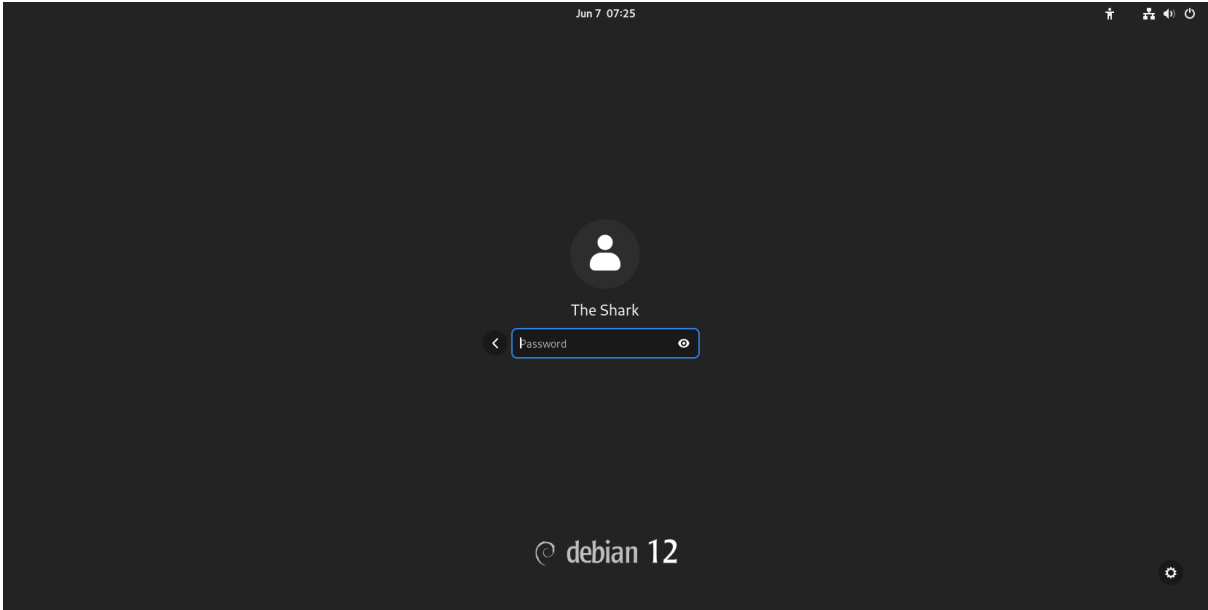
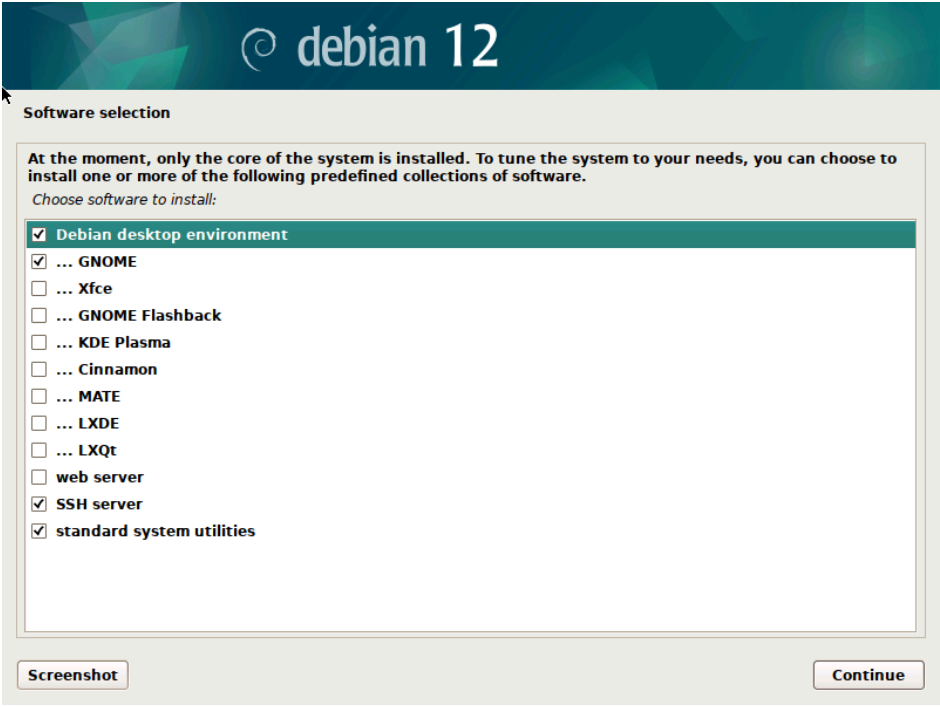
Full name for the new user:



### Set up users and passwords

Select a username for the new account. Your first name is a reasonable choice. The username should start with a lower-case letter, which can be followed by any combination of numbers and more lower-case letters.

Username for your account:



## II – DÉFINITION ET INSTALLATION DE WIRESHARK

### Définition et ce qu'il y à savoir sur wireshark

**Wireshark** est un logiciel open source d'analyse des protocoles réseau créé par **Gerald Combs** en **1998**. Un groupe international d'experts réseau et de développeurs gère aujourd'hui cet outil et le met à jour pour assurer sa compatibilité avec les nouvelles technologies réseau et méthodes de chiffrement. Wireshark ne pose absolument aucun risque de sécurité. Il est notamment utilisé par des agences gouvernementales, de grandes entreprises, des organisations à but **non lucratif** et des établissements pédagogiques pour résoudre des problèmes réseau et assurer des formations. Il n'y a pas de meilleur moyen pour apprendre le fonctionnement des réseaux que d'analyser du trafic sous le microscope de Wireshark. La question de **la légalité** de Wireshark est souvent posée, car il s'agit d'un puissant outil de capture de paquets. Pour rester du côté lumineux de la Force, vous ne devez utiliser Wireshark que sur les réseaux dont vous avez l'autorisation d'inspecter les paquets. Utiliser Wireshark pour observer des paquets **sans autorisation** vous ferait basculer du **côté obscur** de la Force.

### Le Fonctionnement de wireshark

Wireshark est un outil de **capture et d'analyse de paquets**. Il capture le trafic du réseau local et stocke les données ainsi obtenues pour permettre leur analyse hors ligne. Wireshark est capable de capturer le trafic **Ethernet**, **Bluetooth**, **sans fil** (IEEE.802.11), **Token Ring**, **Frame Relay** et plus encore.

*Remarque : un « **paquet** » est un message d'un protocole réseau (par ex., TCP, DNS, etc.). Le trafic du réseau local est basé sur le concept de diffusion, cela signifie qu'un seul ordinateur disposant de Wireshark peut visualiser le trafic reliant deux autres ordinateurs. Pour visualiser le trafic émis vers un site externe, vous devez capturer les paquets sur l'ordinateur local.*

Wireshark vous permet de filtrer le journal avant le début de la capture ou

pendant l'analyse. Il vous est ainsi possible d'éliminer le bruit pour trouver exactement ce que vous recherchez dans la trace réseau. Par exemple, vous pouvez définir un filtre qui n'affiche que le trafic **TCP** entre deux **adresses IP**. Vous pouvez également choisir de n'afficher que les paquets envoyés depuis un ordinateur précis. Si Wireshark est devenu une référence de l'analyse de paquets, c'est en grande partie grâce à ses filtres.

## Installation de Wireshark sur Linux (Debian)

L'un des avantages de Debian est que Wireshark réside par défaut dans son référentiel de logiciels. L'implication de ceci est double : premièrement, cela accélère le processus d'installation car il n'est pas nécessaire de télécharger manuellement ou de compiler à partir du code source ; Deuxièmement, Wireshark reste mis à jour parallèlement aux mises à jour de votre système, vous fournissant la version la plus récente et la plus sécurisée.

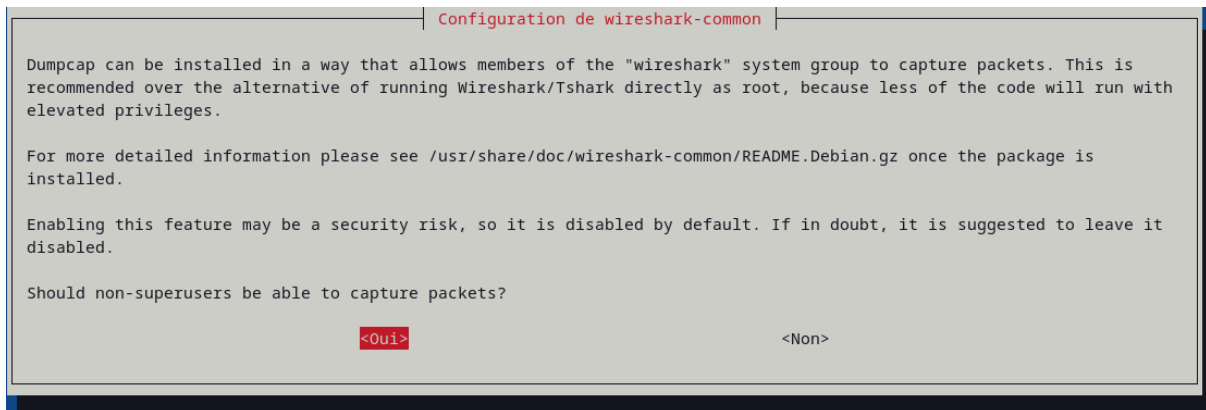
Pour procéder à l'installation à partir du référentiel Debian, utilisez la commande ci-dessous :

**sudo apt install wireshark**

```
laplateforme@wireshark:~$ sudo apt install wireshark
```

Pendant l'installation, une invite peut apparaître demandant si les non-superutilisateurs doivent être autorisés à exécuter Wireshark. Cette décision dépend des autorisations système nécessaires au fonctionnement de l'application et doit être évaluée en tenant compte de vos exigences en matière de sécurité.

**Remarque :** Si vous décidez de ne pas accorder cet accès, chaque utilisateur doit être ajouté individuellement au groupe d'utilisateurs « wireshark ».



Une fois la configuration terminée, vous pouvez passer en **root** procéder à l'inclusion de votre utilisateur dans le groupe «wireshark»:

**usermod -a -G wireshark laplateforme**

**laplateforme@wireshark:~\$** usermod -a -G wireshark laplateforme

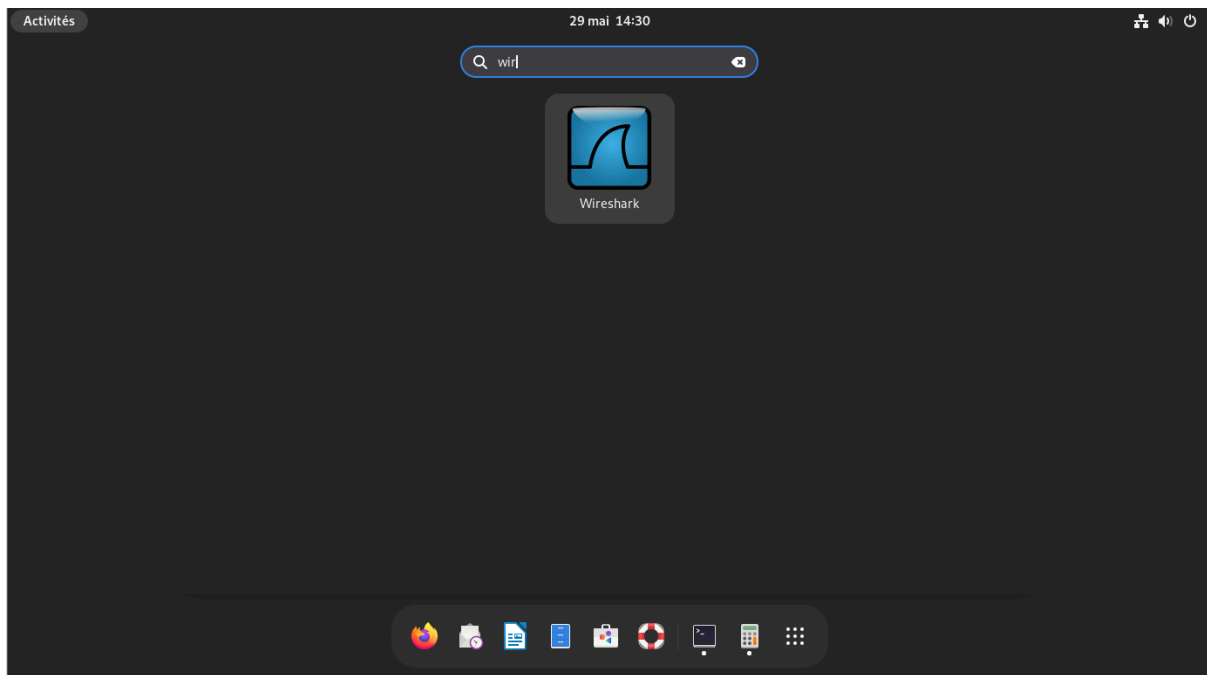
Après avoir terminé l'ajout de l'utilisateur au groupe « wireshark », revenez à votre compte utilisateur habituel. Cependant je vous conseille d'exécuter wireshark avec l'utilisateur **root**.

Vous avez deux possibilités, vous exécutez directement dans le terminal la commande suivante :

**wireshark &**

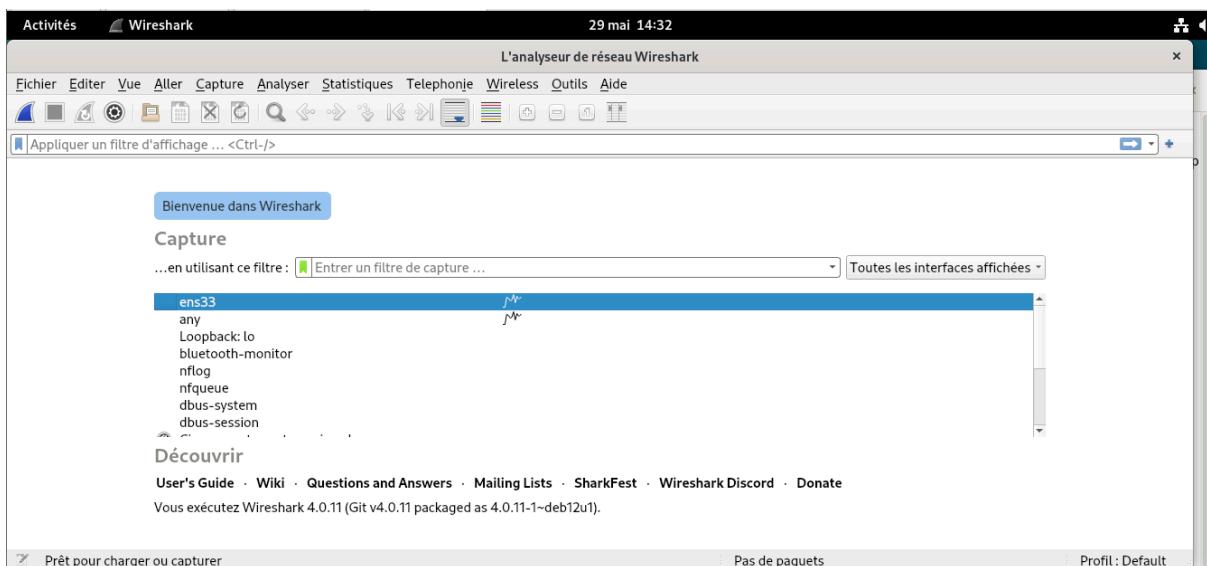
**root@wireshark:~#** wireshark &

Pour les utilisateurs de bureau qui privilégient les interfaces graphiques, Debian rend Wireshark facilement accessible via son menu Applications via le chemin suivant :



## Résultat :

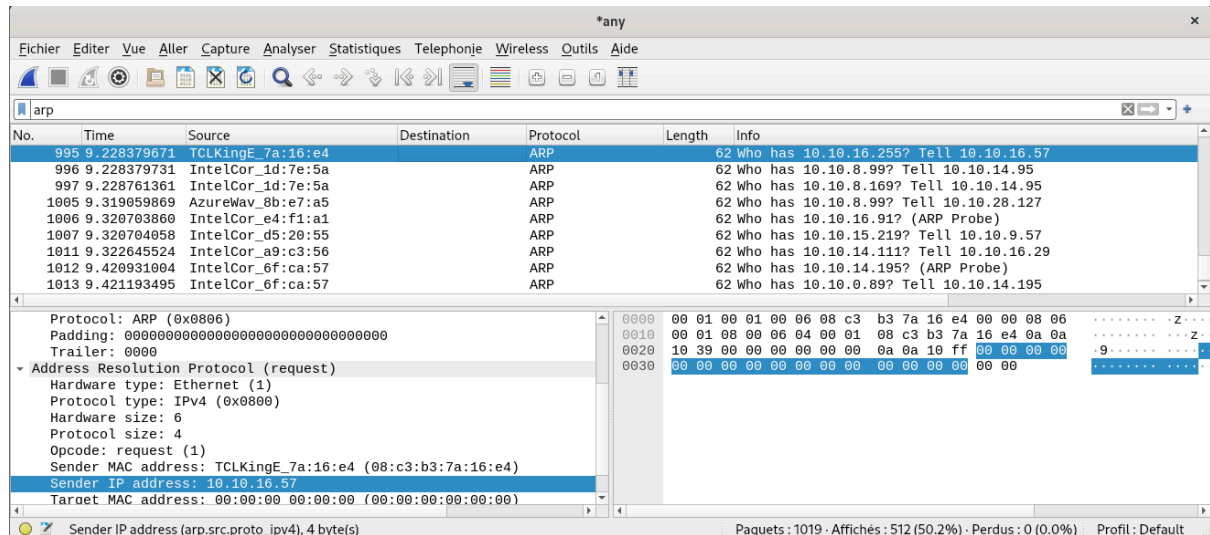
Le lancement de wireshark va faire apparaître l'image ci-dessous, et dans cette image vous remarquez plusieurs interfaces réseaux. Vous pouvez maintenant choisir une interface et capturer les paquets selon vos besoin :





# III – CAPTURE DE PAQUETS SUR WIRESHARK SUR LE RÉSEAU DE L'ALCAZAR

## Capture des paquets ARP : (Address Resolution Protocol)



The screenshot shows a Wireshark capture of ARP packets. The packet list on the left shows several ARP requests and probes. The selected packet (No. 995) is an ARP request from 10.10.16.57 to 10.10.16.255. The packet details pane on the right shows the structure of the ARP request, including the hardware type (Ethernet), protocol type (IPv4), and the sender and target MAC and IP addresses. The packet bytes pane on the right shows the raw data of the packet.

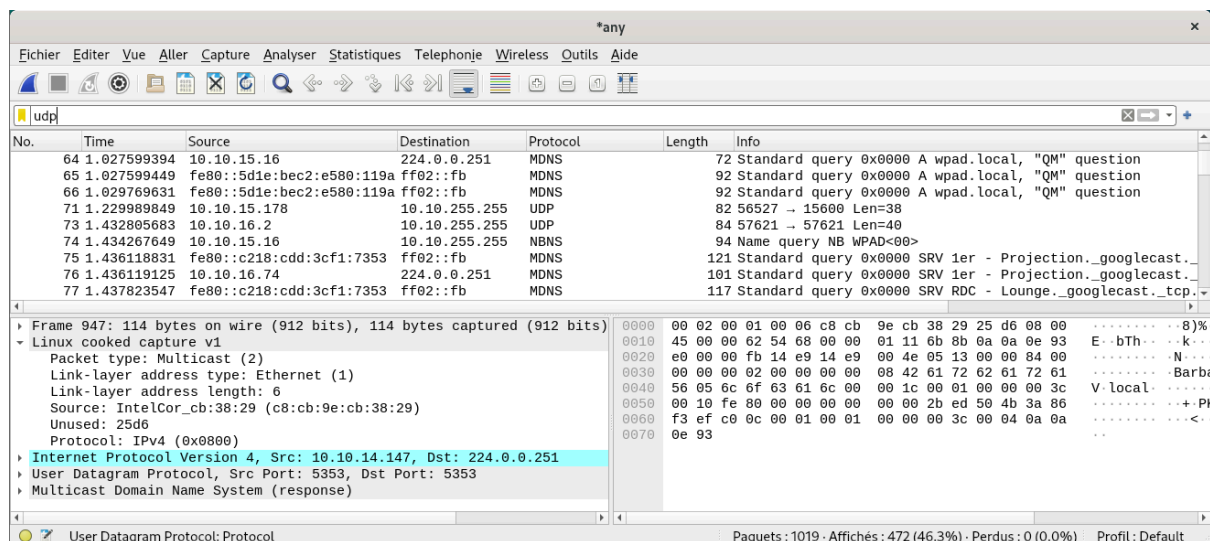
No.	Time	Source	Destination	Protocol	Length	Info
995	9.228379671	TCLKingE_7a:16:e4		ARP	62	Who has 10.10.16.255? Tell 10.10.16.57
996	9.228379731	IntelCor_1d:7e:5a		ARP	62	Who has 10.10.8.99? Tell 10.10.14.95
997	9.228761361	IntelCor_1d:7e:5a		ARP	62	Who has 10.10.8.169? Tell 10.10.14.95
1005	9.319059869	AzureWav_8b:e7:a5		ARP	62	Who has 10.10.8.99? Tell 10.10.28.127
1006	9.320703860	IntelCor_e4:f1:a1		ARP	62	Who has 10.10.16.91? (ARP Probe)
1007	9.320704058	IntelCor_d5:20:55		ARP	62	Who has 10.10.15.219? Tell 10.10.9.57
1011	9.322645524	IntelCor_a9:c3:56		ARP	62	Who has 10.10.14.111? Tell 10.10.16.29
1012	9.420931004	IntelCor_6f:ca:57		ARP	62	Who has 10.10.14.195? (ARP Probe)
1013	9.421193495	IntelCor_6f:ca:57		ARP	62	Who has 10.10.0.89? Tell 10.10.14.195

Protocol: ARP (0x0806)  
Padding: 00000000000000000000000000000000  
Trailer: 0000

Address Resolution Protocol (request)  
Hardware type: Ethernet (1)  
Protocol type: IPv4 (0x0800)  
Hardware size: 6  
Protocol size: 4  
Opcode: request (1)  
Sender MAC address: TCLKingE\_7a:16:e4 (08:c3:b3:7a:16:e4)  
Sender IP address: 10.10.16.57  
Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)

Paquets: 1019 · Affichés: 512 (50.2%) · Perdus: 0 (0.0%) · Profil: Default

## Capture des paquets UDP : (User Datagram Protocol)



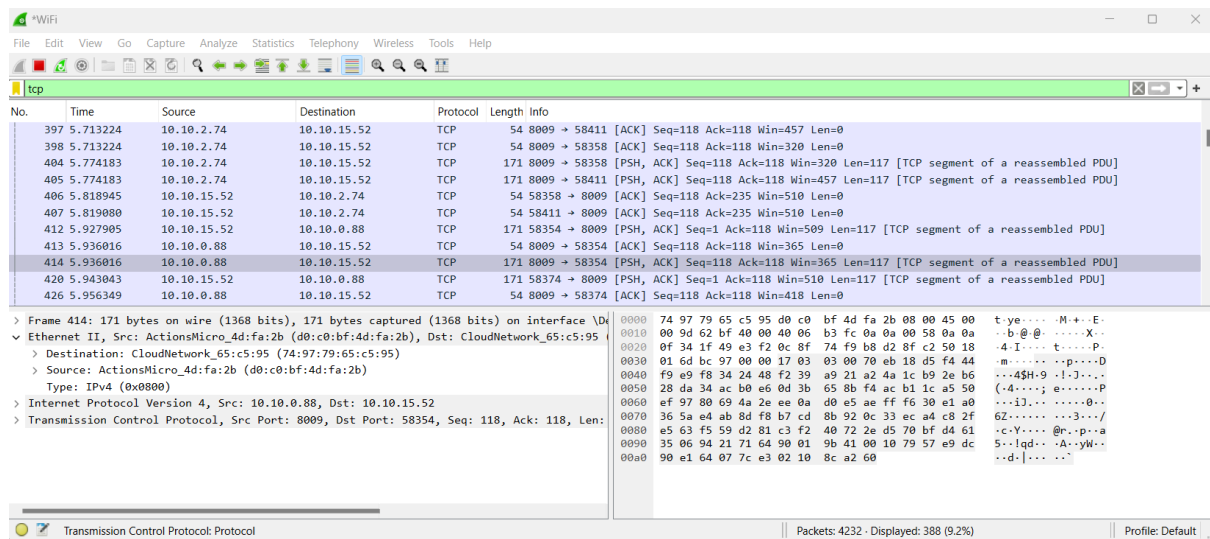
The screenshot shows a Wireshark capture of UDP packets. The packet list on the left shows several UDP packets. The selected packet (No. 947) is a UDP packet from 10.10.14.147 to 224.0.0.251. The packet details pane on the right shows the structure of the UDP packet, including the source and destination ports (5353). The packet bytes pane on the right shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
64	1.027599394	10.10.15.16	224.0.0.251	MDNS	72	Standard query 0x0000 A wpad.local, "QM" question
65	1.027599449	fe80::5d1e:bec2:e580:119a	ff02::fb	MDNS	92	Standard query 0x0000 A wpad.local, "QM" question
66	1.029769631	fe80::5d1e:bec2:e580:119a	ff02::fb	MDNS	92	Standard query 0x0000 A wpad.local, "QM" question
71	1.229989849	10.10.15.178	10.10.255.255	UDP	82	56527 → 15600 Len=38
73	1.432805683	10.10.16.2	10.10.255.255	UDP	84	57621 → 57621 Len=40
74	1.434267649	10.10.15.16	10.10.255.255	NBNS	94	Name query NB WPAD<00>
75	1.436118831	fe80::c218:cdd:3cf1:7353	ff02::fb	MDNS	121	Standard query 0x0000 SRV 1er - Projection._googlecast._
76	1.436119125	10.10.16.74	224.0.0.251	MDNS	101	Standard query 0x0000 SRV 1er - Projection._googlecast._
77	1.437823547	fe80::c218:cdd:3cf1:7353	ff02::fb	MDNS	117	Standard query 0x0000 SRV RDC - Lounge._googlecast._tcp._

Frame 947: 114 bytes on wire (912 bits), 114 bytes captured (912 bits)  
Linux cooked capture v1  
Packet type: Multicast (2)  
Link-layer address type: Ethernet (1)  
Link-layer address length: 6  
Source: IntelCor\_cb:38:29 (c8:cb:9e:cb:38:29)  
Unused: 25d6  
Protocol: IPv4 (0x0800)  
Internet Protocol Version 4, Src: 10.10.14.147, Dst: 224.0.0.251  
User Datagram Protocol, Src Port: 5353, Dst Port: 5353  
Multicast Domain Name System (response)

Paquets: 1019 · Affichés: 472 (46.3%) · Perdus: 0 (0.0%) · Profil: Default

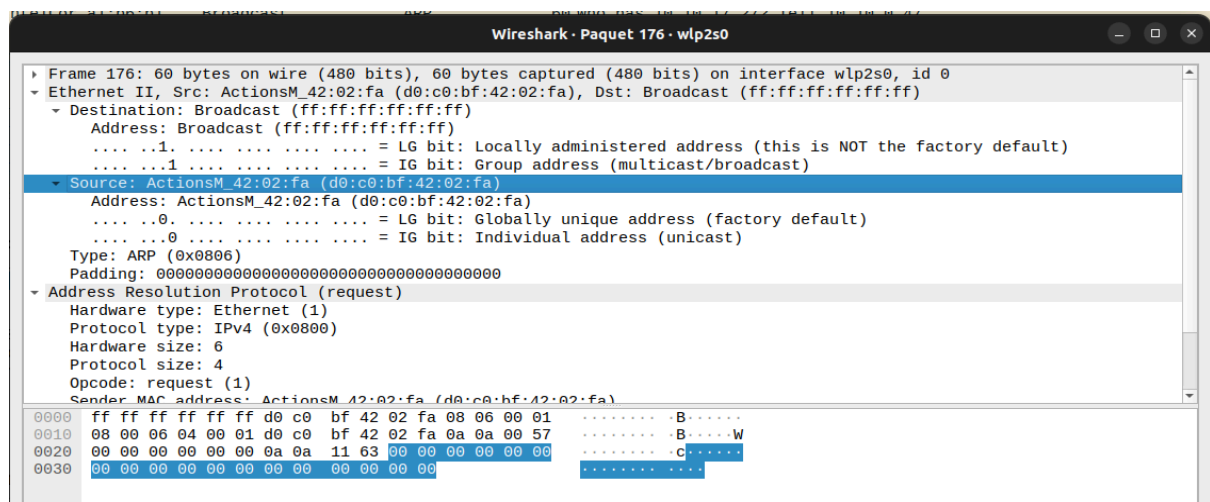
## Capture des paquets TCP : (Transmission Control Protocol)



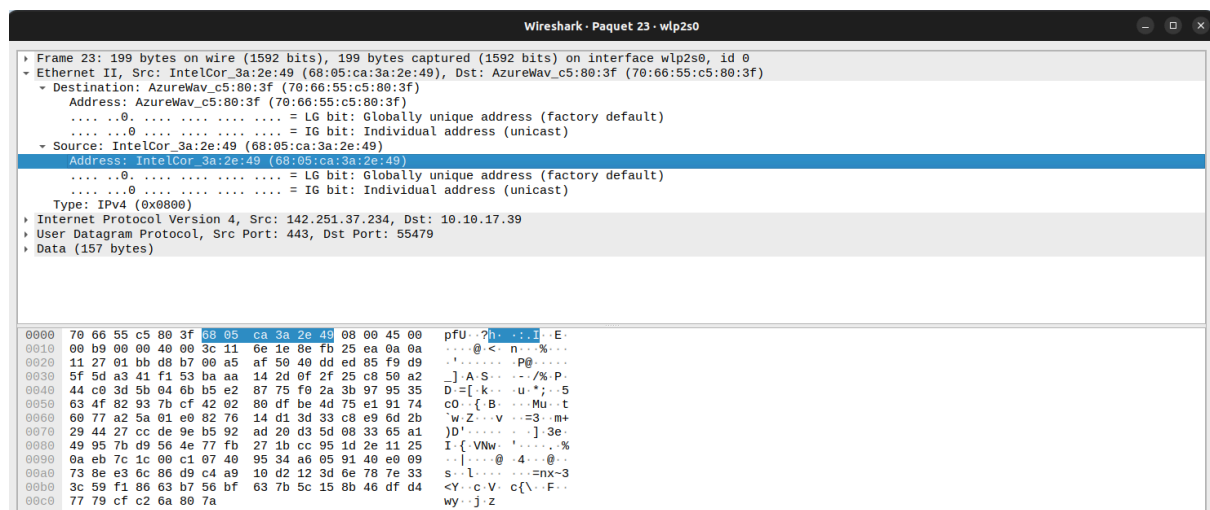
Avec Wireshark, désencapsulons les trames pour retrouver les différentes couches du modèle OSI.

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

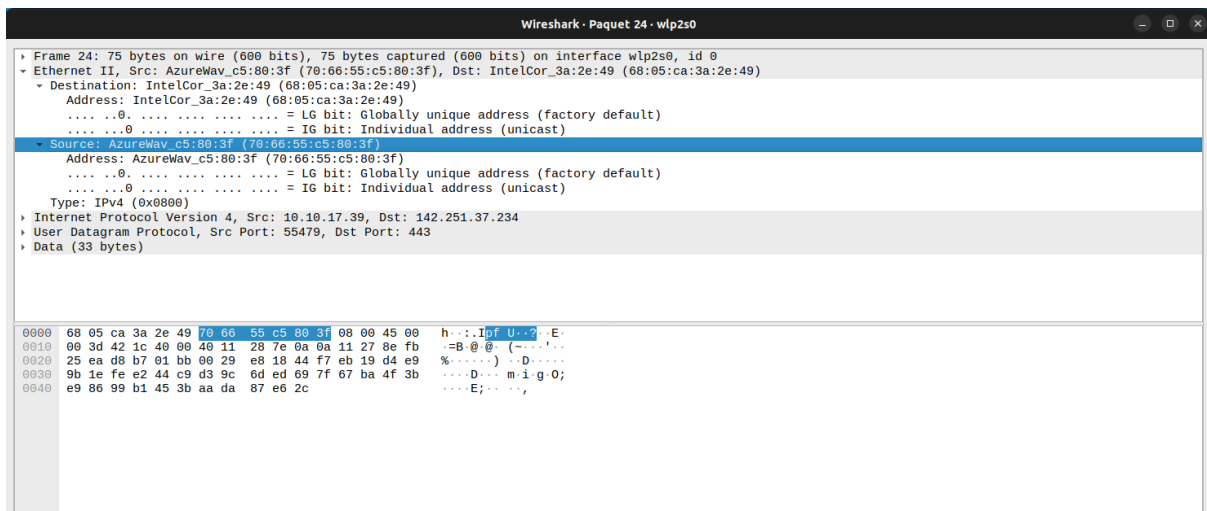
- **Adresse MAC** : Une adresse **MAC** (Media Access Control) est un identifiant **unique** attribué à chaque interface réseau pour la communication au sein d'un segment de réseau local. Les adresses **MAC** sont utilisées pour assurer la transmission correcte des données au niveau de la couche liaison de données dans le modèle **OSI**.



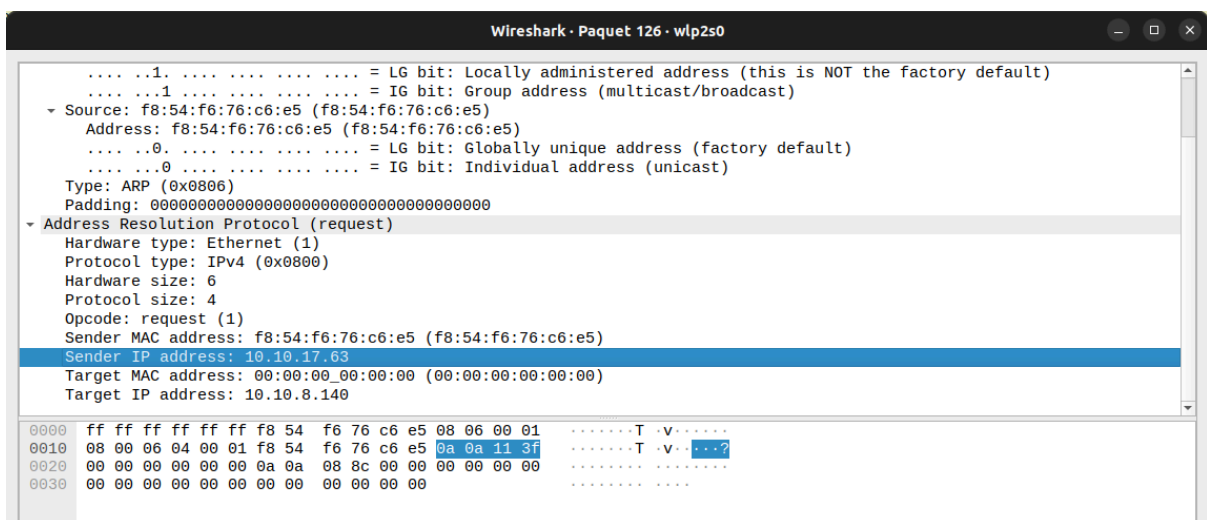
Pour le protocole **ARP** que nous venons de capturer, nous voyons que **l'adresse Mac source** est (**d0:c0:bf:42:02:fa**) et que **l'adresse Mac de destination** est (**ff:ff:ff:ff:ff:ff**).



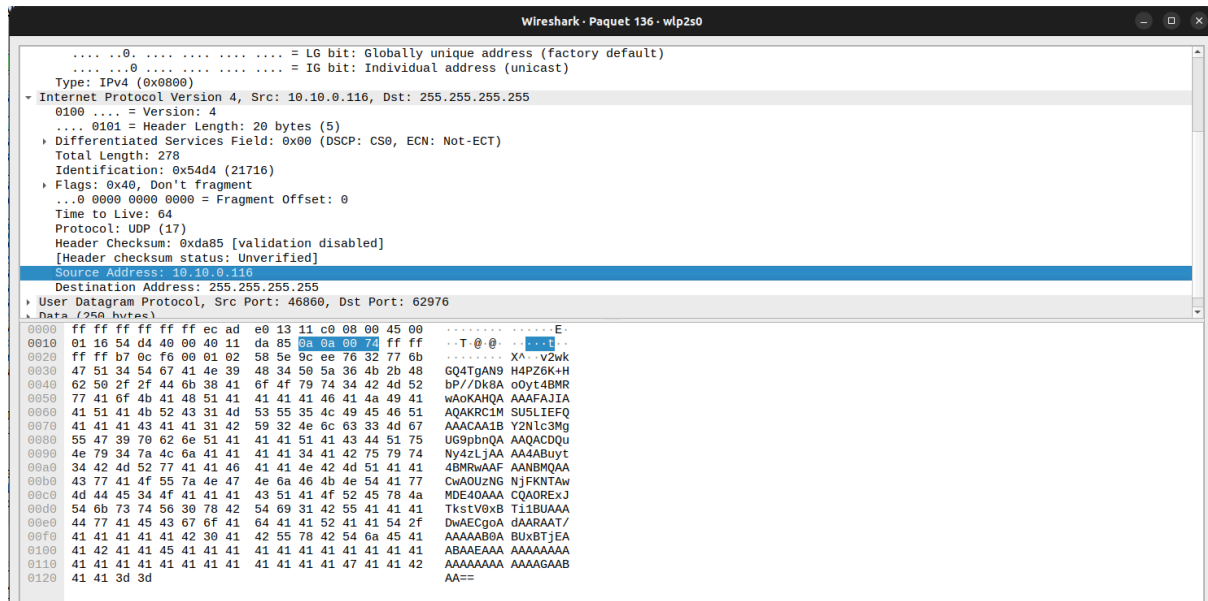
Pour le protocole **UDP** que nous venons de capturer, nous voyons que **l'adresse Mac source** est (**68:05:ca:3a:2e:49**) et que **l'adresse Mac de destination** est (**70:66:55:c5:80:3f**).



Pour le protocole **TCP** que nous venons de capturer, nous voyons que **l'adresse Mac source** est (**70:66:55:c5:80:3f**) et que **l'adresse Mac de destination** est (**68:05:ca:3a:2e:49**).



Pour le protocole **ARP** que nous venons de capturer, nous voyons que **l'IP Source** est **10.10.17.63** et que **l'IP de Destination** est **10.10.8.140**.

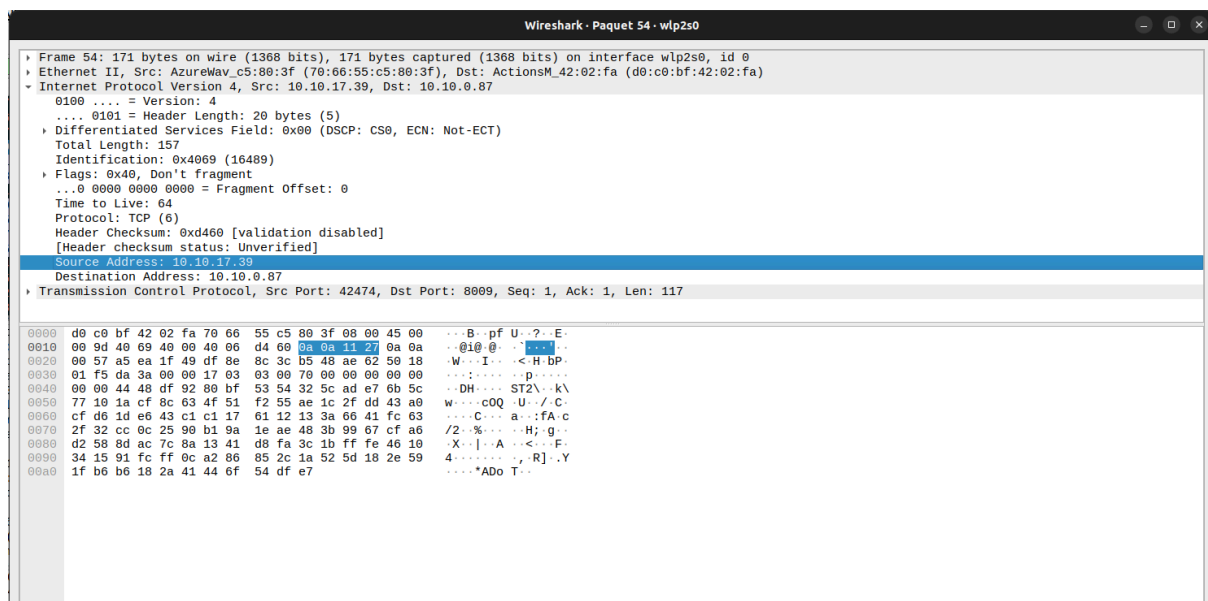


The image shows a Wireshark packet capture window titled "Wireshark - Paquet 136 - wlp2s0". The packet list on the left shows a packet of type "Internet Protocol Version 4" with source address 10.10.0.116 and destination address 255.255.255.255. The packet details pane shows the following information:

- Type: IPv4 (0x0800)
- Internet Protocol Version 4, Src: 10.10.0.116, Dst: 255.255.255.255
- 0100 .... = Version: 4
- .... 0101 = Header Length: 20 bytes (5)
- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 278
- Identification: 0x54d4 (21716)
- Flags: 0x40, Don't fragment
- ...0 0000 0000 0000 = Fragment Offset: 0
- Time to Live: 64
- Protocol: UDP (17)
- Header Checksum: 0xda85 [validation disabled]
- [Header checksum status: Unverified]
- Source Address: 10.10.0.116
- Destination Address: 255.255.255.255
- User Datagram Protocol, Src Port: 46860, Dst Port: 62976
- Data (258 bytes)

The packet bytes pane shows the raw data of the packet, including the IPv4 header and the UDP payload.

Pour le protocole **UDP** que nous venons de capturer, nous voyons que **l'IP source** est **10.10.0.116** et que **l'IP de destination** est **255.255.255.255**.



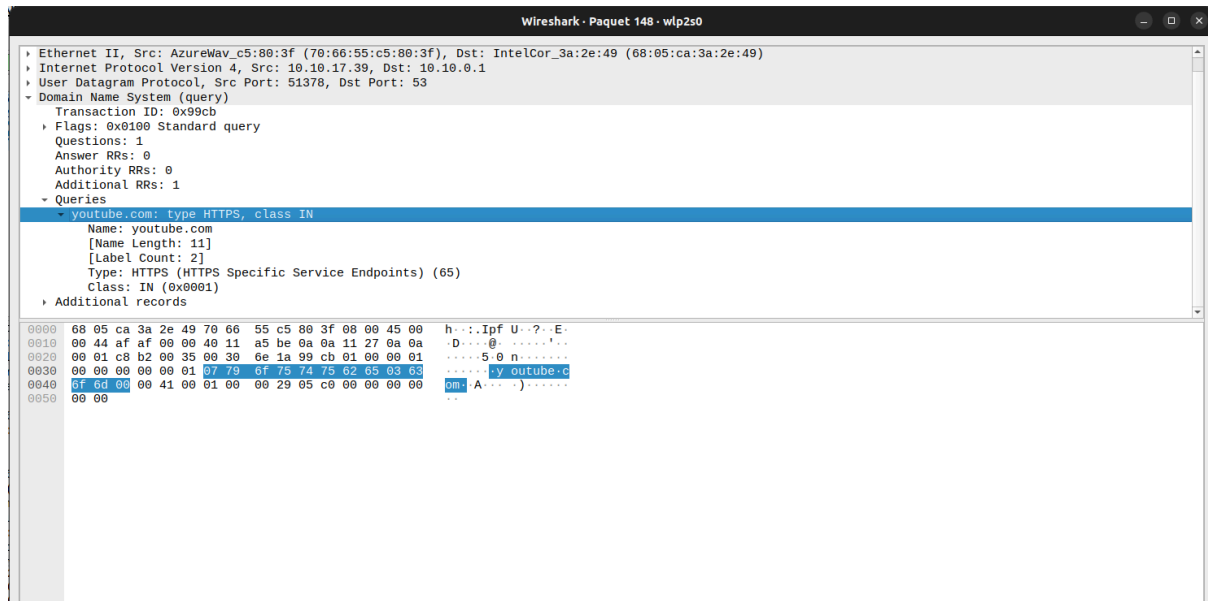
The image shows a Wireshark packet capture window titled "Wireshark - Paquet 54 - wlp2s0". The packet list on the left shows a packet of type "Transmission Control Protocol" with source address 10.10.17.39 and destination address 10.10.0.87. The packet details pane shows the following information:

- Frame 54: 171 bytes on wire (1368 bits), 171 bytes captured (1368 bits) on interface wlp2s0, id 0
- Ethernet II, Src: AzureWav\_c5:80:3f (70:66:55:c5:80:3f), Dst: ActionsM\_42:02:fa (d0:c0:bf:42:02:fa)
- Internet Protocol Version 4, Src: 10.10.17.39, Dst: 10.10.0.87
- 0100 .... = Version: 4
- .... 0101 = Header Length: 20 bytes (5)
- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 157
- Identification: 0x4069 (16489)
- Flags: 0x40, Don't fragment
- ...0 0000 0000 0000 = Fragment Offset: 0
- Time to Live: 64
- Protocol: TCP (6)
- Header Checksum: 0xd460 [validation disabled]
- [Header checksum status: Unverified]
- Source Address: 10.10.17.39
- Destination Address: 10.10.0.87
- Transmission Control Protocol, Src Port: 42474, Dst Port: 8009, Seq: 1, Ack: 1, Len: 117

The packet bytes pane shows the raw data of the packet, including the IPv4 header, the TCP header, and the application data.

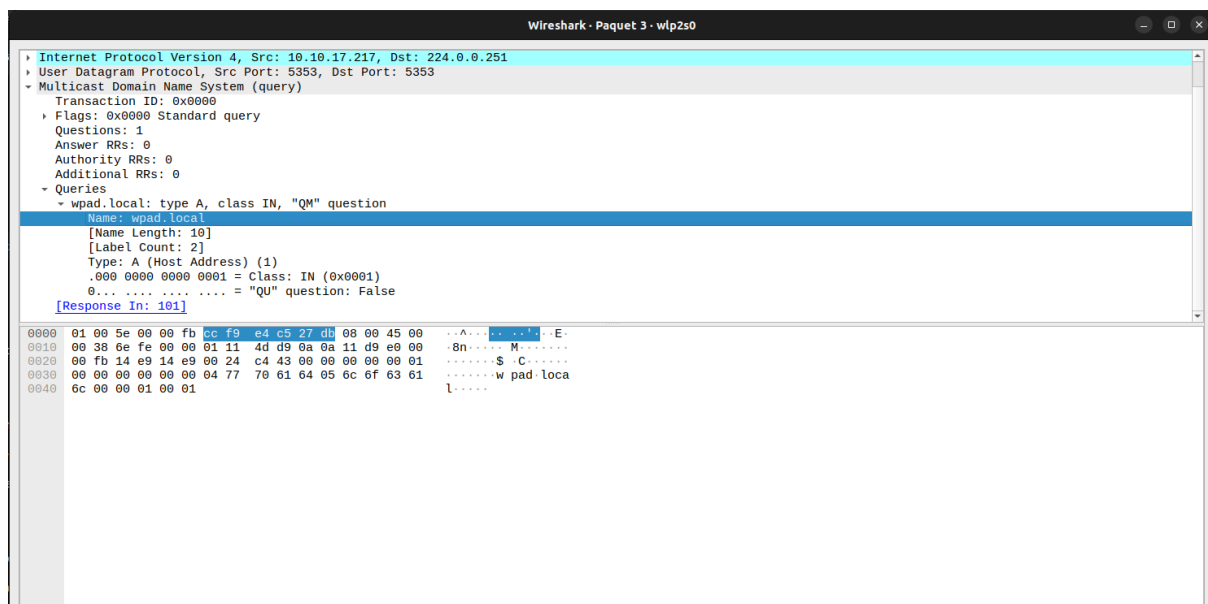
Pour le protocole **TCP** que nous venons de capturer, nous voyons que **L'IP source** est **10.10.17.39** et que **l'IP de destination** est **10.10.0.87**.

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



## DNS (Domain Name System)

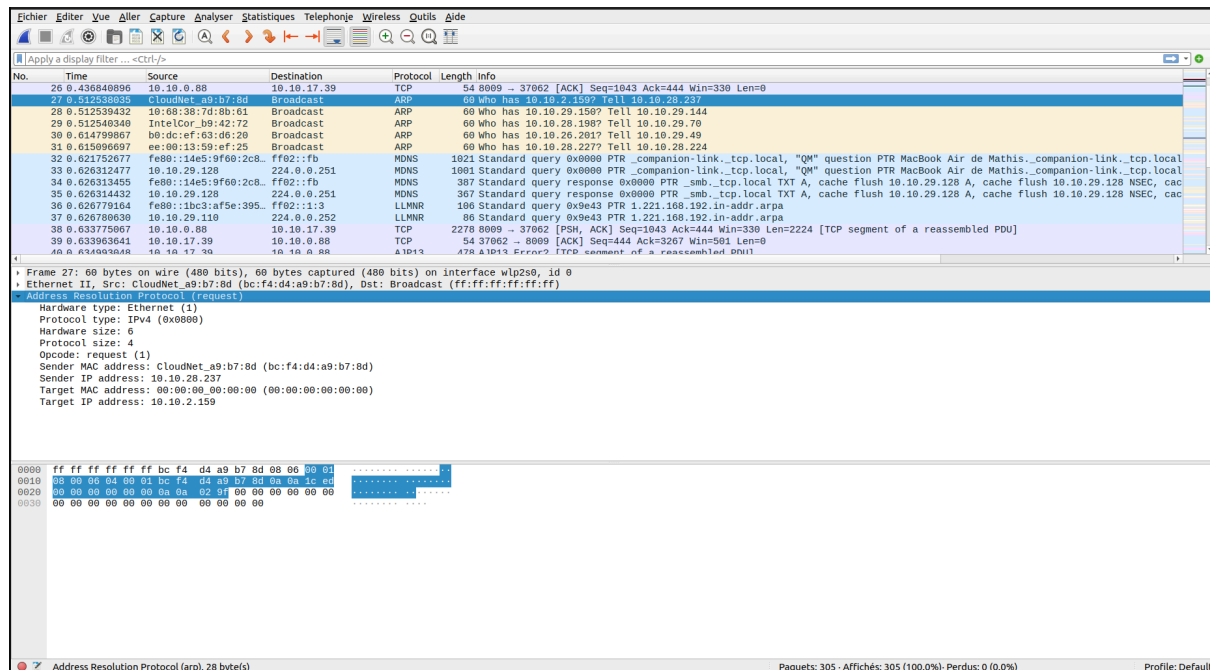
- **Protocole** : **DNS** (Domain Name System)
- **Fonction** : Utilisé pour traduire les noms de domaine en adresses IP. Les clients DNS envoient des requêtes aux serveurs DNS pour obtenir l'adresse IP associée à un nom de domaine spécifique.
- **Exemple** :
  - **Source IP** : 192.168.1.10
  - **Destination IP** : 8.8.8.8 (serveur DNS de Google)
  - **Fonction** : "Quelle est l'adresse IP de www.youtube.com ?"



## mDNS (Multicast DNS)

- **Protocole** : **mDNS** (Multicast Domain Name System)
- **Fonction** : Utilisé pour la résolution de noms de domaine sur les réseaux locaux sans configuration spécifique de serveur DNS. Les appareils utilisent mDNS pour annoncer et résoudre les noms d'hôtes à travers les paquets multicast.
- **Exemple** :
  - **Source IP** : 192.168.1.5
  - **Destination IP** : 224.0.0.251 (adresse multicast mDNS)
  - **Fonction** : "Quel appareil répond au nom de host.local ?"

**Cherchez les spécifications du format des messages ARP/UDP/TCP et faites correspondre les captures en hexadécimal.**



## ARP (Address Resolution Protocol)

L'en-tête ARP a une longueur fixe de 28 octets et contient les champs suivants :

1. **Hardware Type** (2 octets) : Typiquement **0001** pour Ethernet.
2. **Protocol Type** (2 octets) : Typiquement **0800** pour IPv4.
3. **Hardware Address Length** (1 octet) : Typiquement **06** pour une adresse MAC.
4. **Protocol Address Length** (1 octet) : Typiquement **04** pour une adresse IPv4.
5. **Operation** (2 octets) : **0001** pour la requête, **0002** pour la réponse.
6. **Sender Hardware Address** (6 octets)
7. **Sender Protocol Address** (4 octets)
8. **Target Hardware Address** (6 octets)



## 9. Target Protocol Address (4 octets)

The image shows a Wireshark network traffic capture. The top pane displays a list of captured packets. The bottom pane shows the details of the selected packet (No. 230), which is a User Datagram Protocol (UDP) packet. The packet details include the source and destination ports, length, checksum, and the payload data.

No.	Time	Source	Destination	Protocol	Length	Info
221	3.998802220	10.10.28.211	224.0.0.2	IGMPv2	46	Leave Group 224.0.0.252
222	3.998803128	fe80::529e:be45:7318::18ae	ff02::16	IGMPv6	96	Multicast Listener Report Message v2
223	3.998804105	10.10.28.211	224.0.0.252	IGMPv2	46	Membership Report group 224.0.0.252
224	3.998805013	10.10.28.211	224.0.0.251	MDNS	81	Standard query 0x0000 ANY LAPTOP-INT10H6UK.local, "QM" question
225	3.998844322	10.10.28.211	224.0.0.251	MDNS	119	Standard query response 0x0000 AAAA fe80::529e:be45:7318::18ae A 10.10.28.211
226	4.001910326	fe80::529e:be45:7318::18ae	ff02::fb	MDNS	101	Standard query 0x0000 ANY LAPTOP-INT10H6UK.local, "QM" question
227	4.001911298	fe80::529e:be45:7318::18ae	ff02::fb	MDNS	139	Standard query response 0x0000 AAAA fe80::529e:be45:7318::18ae A 10.10.28.211
228	4.096728350	CloudNet_5f:d1:e9	Broadcast	ARP	60	Who has 10.10.29.159? Tell 10.10.29.177
229	4.096729250	e4:fa:c4:08:3a:f9	Broadcast	ARP	60	Who has 10.10.26.243? Tell 10.10.29.40
230	4.097048717	10.10.28.221	10.10.255.255	UDP	82	57621 -> 57621 Len=40
231	4.198443386	AzureWav_5e:56:c3	Broadcast	ARP	60	Who has 10.10.25.198? Tell 10.10.25.83
232	4.198608061	10.10.8.54	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
233	4.198601059	fe80::7b4f:db79:a71::	ff02::fb	MDNS	107	Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
234	4.199117085	0-Linkin_89:e6:b8	Broadcast	LOOP	60	Unknown Function (256)
235	4.301254807	CloudNet_70:01:29	Broadcast	ARP	60	Who has 10.10.26.212? Tell 10.10.29.149
236	4.301481934	IntelCor_8f:ee:c2	Broadcast	ARP	60	Who has 10.10.23.172? Tell 10.10.29.19
237	4.306132879	ActionsM_18:97:56	Broadcast	ARP	60	Who has 10.10.29.137? Tell 10.10.2.73
238	4.306433303	CloudNet_5f:d1:e9	Broadcast	ARP	60	Who has 10.10.26.243? Tell 10.10.29.40

Frame 230: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface wlp2s0, id 0  
Ethernet II, Src: 1e:89:01:0c:3d:87 (1e:89:01:0c:3d:87), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
Internet Protocol Version 4, Src: 10.10.28.221, Dst: 10.10.255.255  
User Datagram Protocol, Src Port: 57621, Dst Port: 57621  
Source Port: 57621  
Destination Port: 57621  
Length: 40  
Checksum: 0x8b86 [unverified]  
[Checksum Status: Unverified]  
[Stream index: 47]  
[Timestamps]  
UDP payload (40 bytes)  
Data (40 bytes)  
ff ff ff ff ff ff 1e 89 01 0c 3d 87 08 00 45 00 .....E  
00 44 01 d8 40 00 40 11 07 e1 0a 0a 1c dd 0a 0a .....D.@  
ff ff 15 15 15 00 00 00 53 70 6f 74 55 64 .....Spotud  
70 30 d2 3b 4e d0 09 38 50 c8 00 01 00 00 1c 33 .....p0|N-8P  
36 08 7d aa 9b ed 77 de 9d f6 fa e5 96 f2 d4 70 .....6}~w  
15 7a .....z

## UDP (User Datagram Protocol)

L'en-tête UDP a une longueur fixe de 8 octets et contient les champs suivants :

1. **Source Port** (2 octets)
2. **Destination Port** (2 octets)
3. **Length** (2 octets) : Longueur totale du datagramme UDP incluant l'en-tête.
4. **Checksum** (2 octets) : Optionnel en IPv4, obligatoire en IPv6.

No.	Time	Source	Destination	Protocol	Length	Info
23	0.427863315	10.10.0.88	10.10.17.39	TCP	54	8009 → 37662 [ACK] Seq=817 Ack=319 Win=330 Len=0
24	0.427864296	10.10.0.88	10.10.17.39	TCP	288	8009 → 37662 [PSH, ACK] Seq=817 Ack=319 Win=330 Len=226 [TCP segment of a reassembled PDU]
25	0.428372596	10.10.17.39	10.10.0.88	TCP	179	37662 → 8009 [PSH, ACK] Seq=319 Ack=1043 Win=501 Len=125 [TCP segment of a reassembled PDU]
26	0.436840896	10.10.0.88	10.10.17.39	TCP	54	8009 → 37662 [ACK] Seq=1043 Ack=444 Win=330 Len=0
38	0.633775067	10.10.0.88	10.10.17.39	TCP	2278	8009 → 37662 [PSH, ACK] Seq=1043 Ack=444 Win=330 Len=2224 [TCP segment of a reassembled PDU]
39	0.633963641	10.10.17.39	10.10.0.88	TCP	54	37662 → 8009 [ACK] Seq=444 Ack=3267 Win=501 Len=0
40	0.634993848	10.10.17.39	10.10.0.88	AJP13	478	AJP13 Error? [TCP segment of a reassembled PDU]
41	0.637881838	10.10.17.39	10.10.0.88	TCP	193	37662 → 8009 [PSH, ACK] Seq=868 Ack=3267 Win=501 Len=139 [TCP segment of a reassembled PDU]
42	0.652861628	10.10.0.88	10.10.17.39	TCP	54	8009 → 37662 [ACK] Seq=3267 Ack=868 Win=375 Len=0
43	0.656815566	10.10.0.88	10.10.17.39	TCP	54	8009 → 37662 [ACK] Seq=3267 Ack=1007 Win=421 Len=0
46	0.784898774	10.10.17.39	10.10.0.87	TCP	171	42988 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=501 Len=117 [TCP segment of a reassembled PDU]
47	0.790388215	10.10.0.88	10.10.17.39	TCP	332	8009 → 37662 [PSH, ACK] Seq=3267 Ack=1007 Win=421 Len=278 [TCP segment of a reassembled PDU]
48	0.809580858	10.10.0.87	10.10.17.39	TCP	171	8009 → 42988 [PSH, ACK] Seq=1 Ack=118 Win=419 Len=117 [TCP segment of a reassembled PDU]
49	0.809825744	10.10.17.39	10.10.0.87	TCP	54	42988 → 8009 [ACK] Seq=118 Ack=118 Win=501 Len=0
54	0.831795984	10.10.17.39	10.10.0.88	TCP	54	37662 → 8009 [ACK] Seq=1007 Ack=3545 Win=501 Len=0
104	1.896120856	10.10.17.39	10.10.0.86	TCP	171	59216 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=501 Len=117 [TCP segment of a reassembled PDU]
109	1.905711092	10.10.0.86	10.10.17.39	TCP	171	8009 → 59216 [PSH, ACK] Seq=1 Ack=118 Win=457 Len=117 [TCP segment of a reassembled PDU]
404	4.065733746	10.10.17.39	10.10.0.88	TCP	54	8009 → 37662 [ACK] Seq=1043 Ack=444 Win=330 Len=0

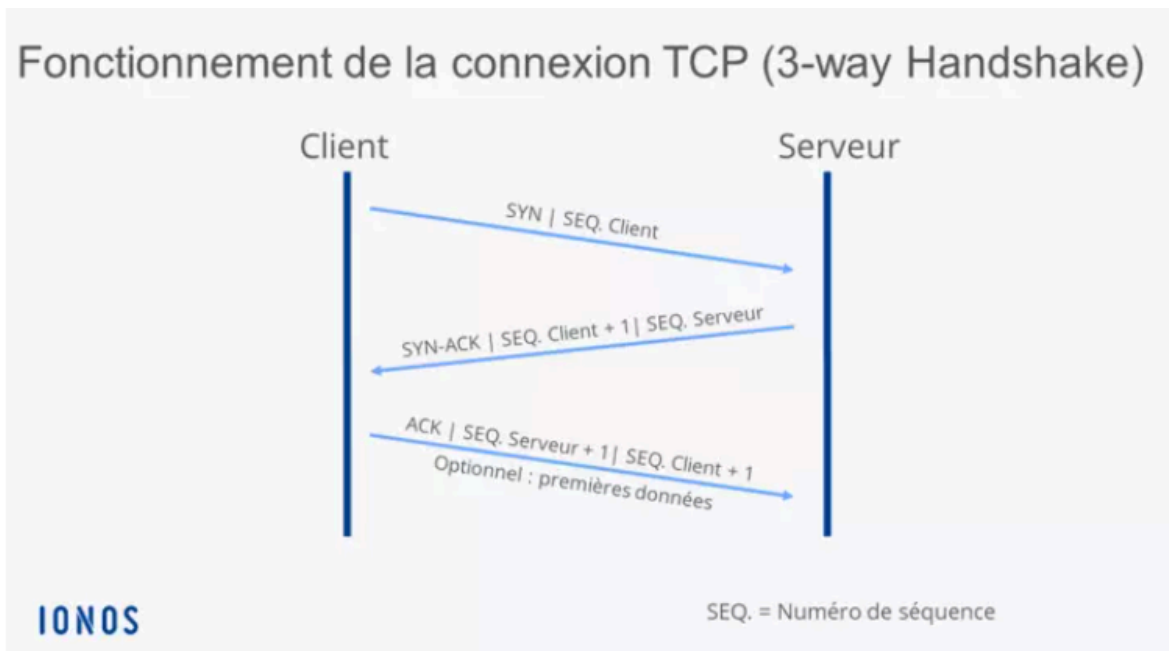
Transmission Control Protocol, Src Port: 8009, Dst Port: 37662, Seq: 817, Ack: 319, Len: 0	
Source Port: 8009	Destination Port: 37662
[Stream index: 0]	
[Conversation completeness: Incomplete (12)]	
[TCP Segment Len: 0]	
Sequence Number: 817 (relative sequence number)	Sequence Number (raw): 2484947050
[Next Sequence Number: 817 (relative sequence number)]	
Acknowledgment Number: 319 (relative ack number)	Acknowledgment number (raw): 3846330729
0101... = Header Length: 20 bytes (5)	
Flags: 0x010 (ACK)	
Window: 330	
0000 70 66 55 c5 80 3f 80 c0 b7 4d fa 2b 08 00 45 00	pFu...?M+...E
0010 00 28 ba f9 40 00 06 5a 44 0a 0a 00 58 0a 0a	(...@...ZD...X...
0020 11 27 1f 49 90 c6 94 1d 48 6a b5 93 51 00 50 10	...I...HJ...Q'P...
0030 01 4a f5 0c 00 00	...J...L...

## TCP (Transmission Control Protocol)

L'en-tête TCP a une longueur minimale de 20 octets, mais peut être plus longue en fonction des options, et contient les champs suivants :

1. **Source Port** (2 octets)
2. **Destination Port** (2 octets)
3. **Sequence Number** (4 octets)
4. **Acknowledgment Number** (4 octets)
5. **Data Offset** (4 bits) : Taille de l'en-tête TCP.
6. **Reserved** (3 bits) : Réserve pour le futur.
7. **Flags** (9 bits) : Contrôle divers (e.g., SYN, ACK, FIN).
8. **Window Size** (2 octets)
9. **Checksum** (2 octets)
10. **Urgent Pointer** (2 octets) : Utilisé si le flag URG est défini.
11. **Options** (variable) : Présent si Data Offset >

## Décrivez le mécanisme de connexion avec un diagramme

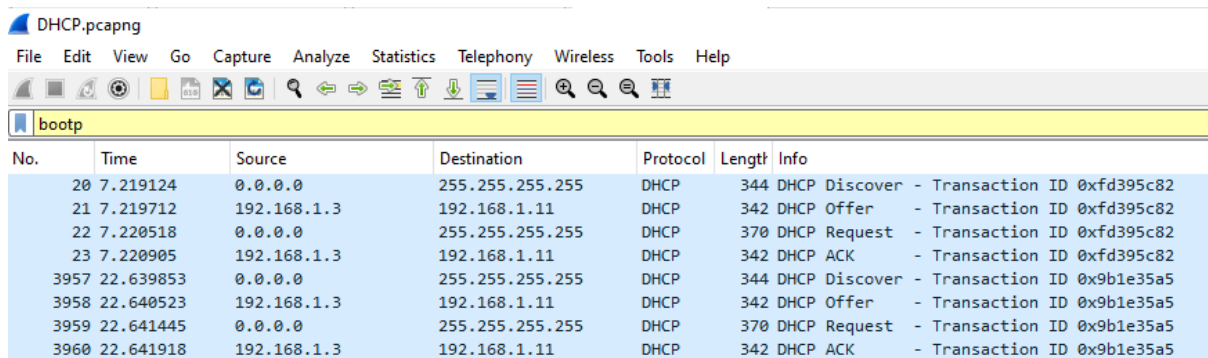


Connexion TCP (« 3-Way Handshake »)

## III – Partie 2

Capture des paquet :

**DHCP** (utiliser **bootp** pour filtrer et afficher uniquement les échanges DHCP) :  
téléchargez le paquet [ici](#)



No.	Time	Source	Destination	Protocol	Length	Info
20	7.219124	0.0.0.0	255.255.255.255	DHCP	344	DHCP Discover - Transaction ID 0xfd395c82
21	7.219712	192.168.1.3	192.168.1.11	DHCP	342	DHCP Offer - Transaction ID 0xfd395c82
22	7.220518	0.0.0.0	255.255.255.255	DHCP	370	DHCP Request - Transaction ID 0xfd395c82
23	7.220905	192.168.1.3	192.168.1.11	DHCP	342	DHCP ACK - Transaction ID 0xfd395c82
3957	22.639853	0.0.0.0	255.255.255.255	DHCP	344	DHCP Discover - Transaction ID 0x9b1e35a5
3958	22.640523	192.168.1.3	192.168.1.11	DHCP	342	DHCP Offer - Transaction ID 0x9b1e35a5
3959	22.641445	0.0.0.0	255.255.255.255	DHCP	370	DHCP Request - Transaction ID 0x9b1e35a5
3960	22.641918	192.168.1.3	192.168.1.11	DHCP	342	DHCP ACK - Transaction ID 0x9b1e35a5

**DNS** => le paquet [ici](#)

**mDNS ? (qu'est-ce que c'est ?)** => le paquet [ici](#)

**mDNS**, qui signifie **Multicast DNS**, est un protocole qui permet de bénéficier des fonctionnalités du DNS sans avoir un serveur DNS sur le réseau.

Le mDNS utilise une méthode différente de celle du DNS traditionnel : au lieu de solliciter un serveur de noms, tous les participants du réseau sont directement adressés. Le client envoie un multicast (une diffusion) dans le réseau et demande à quel participant du réseau le nom d'hôte correspond.

**SSL/TLS** => Le paquet [ici](#)

**FTP** => Le paquet [ici](#)

**SMB** => Le paquet [ici](#)

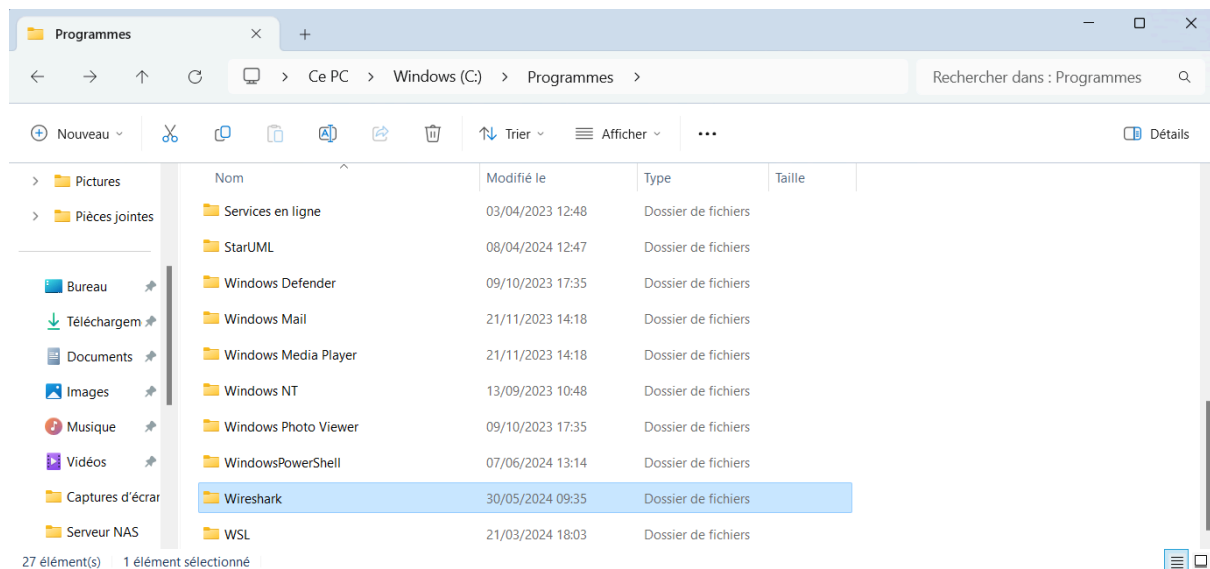
## IV – Partie 3

A présent, nous allons utiliser notre terminal et des scripts pour écouter le réseau et extraire les données :

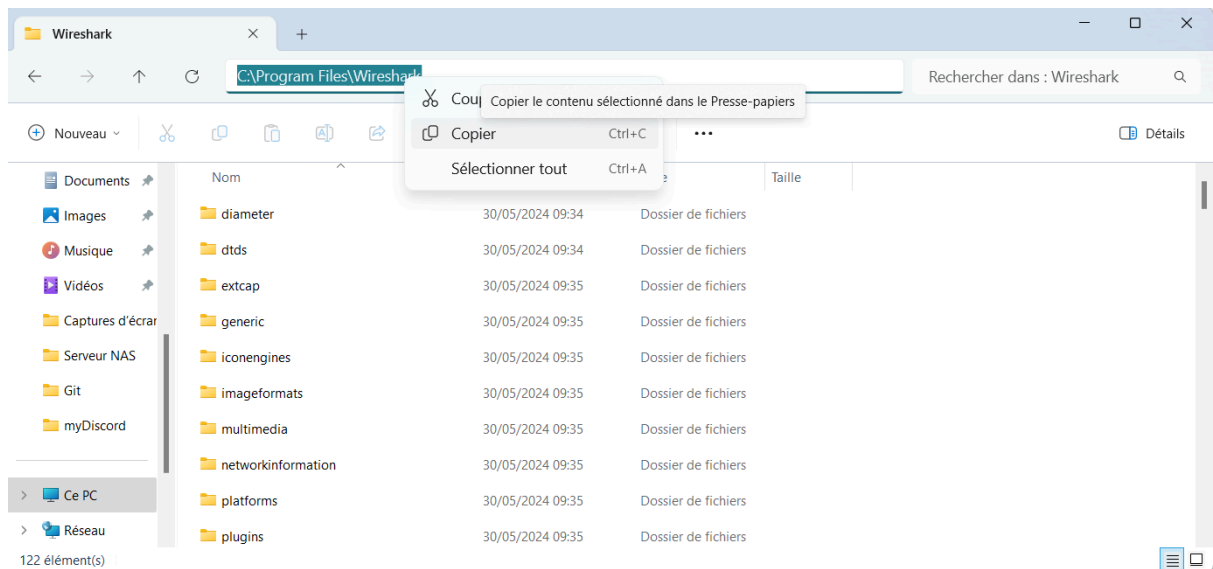
Pour utiliser la commande **tshark** sur Window, il va falloir suivre les étapes suivantes :

Premièrement, nous allons chercher dans notre répertoire système (plus précisément dans le répertoire **“Programme”** où se trouve le dossier et les modules **tshark** qui se trouvent dans le répertoire de **Wireshark** qu’on a précédemment installer :

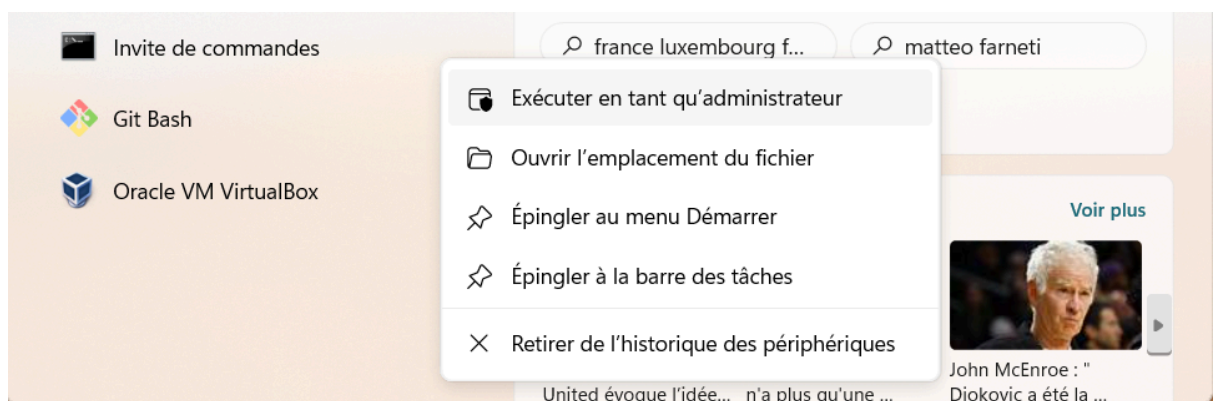
**pour pouvoir utiliser le tshark je consulte le répertoire wireshark**



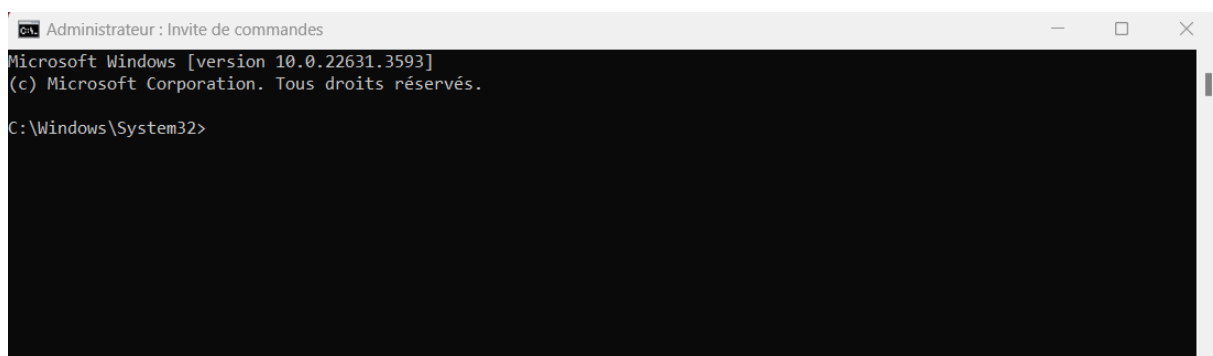
On ouvre le répertoire **Wireshark** et on copie le chemin du répertoire.



Ensuite, nous venons dans la barre de recherche du PC, pour ouvrir **une invite de commande** en tant qu'**administrateur**.



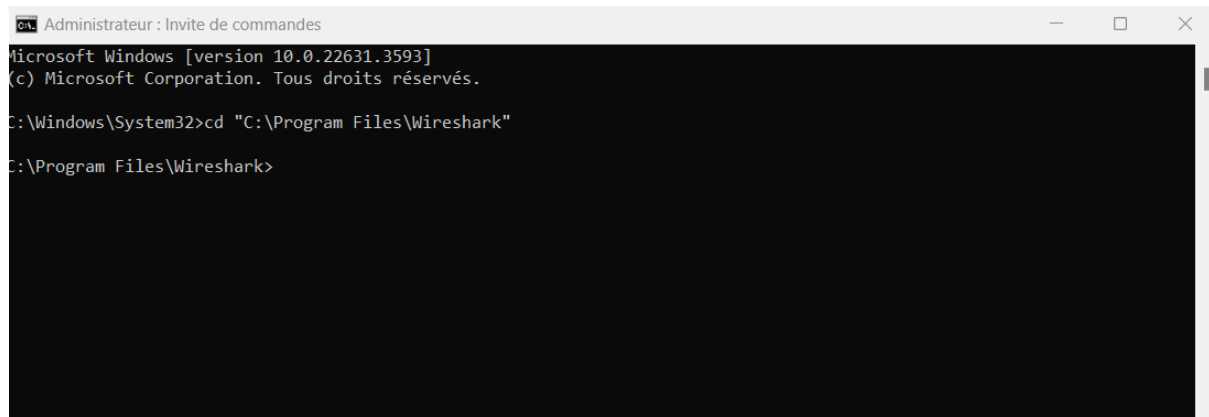
Normalement, nous aurons un environnement comme le suivant :



Là, nous allons venir coller le chemin de répertoire vers **Wireshark** que nous avons copié, précédé de la commande **cd** et mettre le chemin en guillemets :

**cd "C:\Program Files\Wireshark"**

**Résultat :**

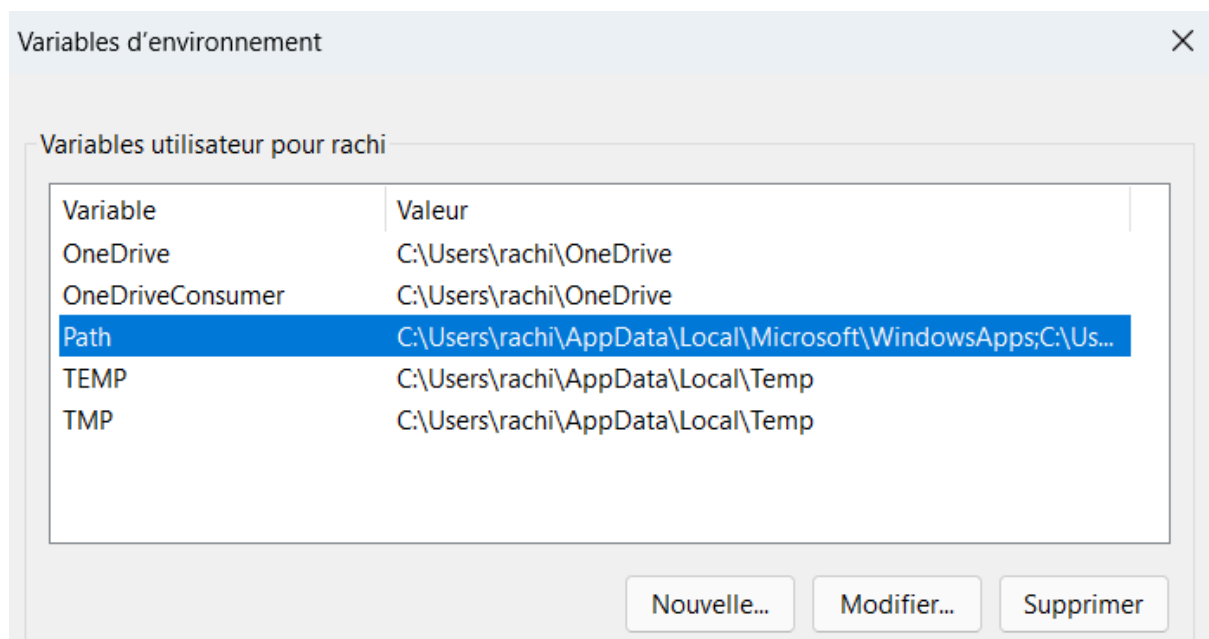


```
Administrateur : Invite de commandes
Microsoft Windows [version 10.0.22631.3593]
(c) Microsoft Corporation. Tous droits réservés.

C:\Windows\System32>cd "C:\Program Files\Wireshark"

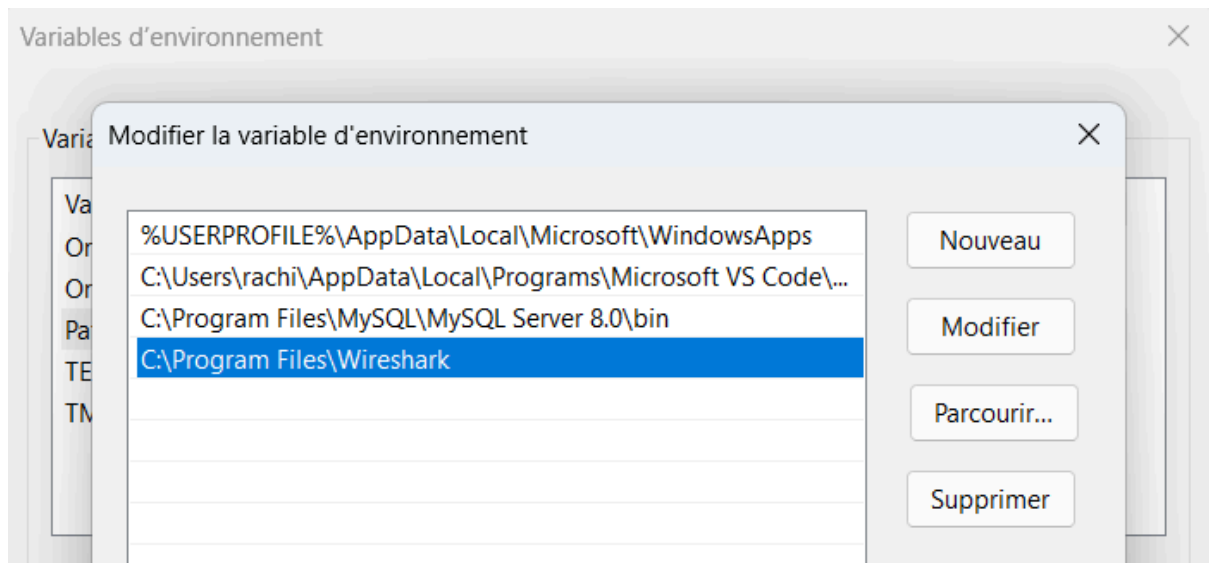
C:\Program Files\Wireshark>
```

Après nous allons modifier l'option **"Path"** de **"Modifier les variables de l'environnement pour votre compte"** de l'ordinateur. On clique sur **Modifier** :



Et on ajoute le chemin de notre répertoire suivant:

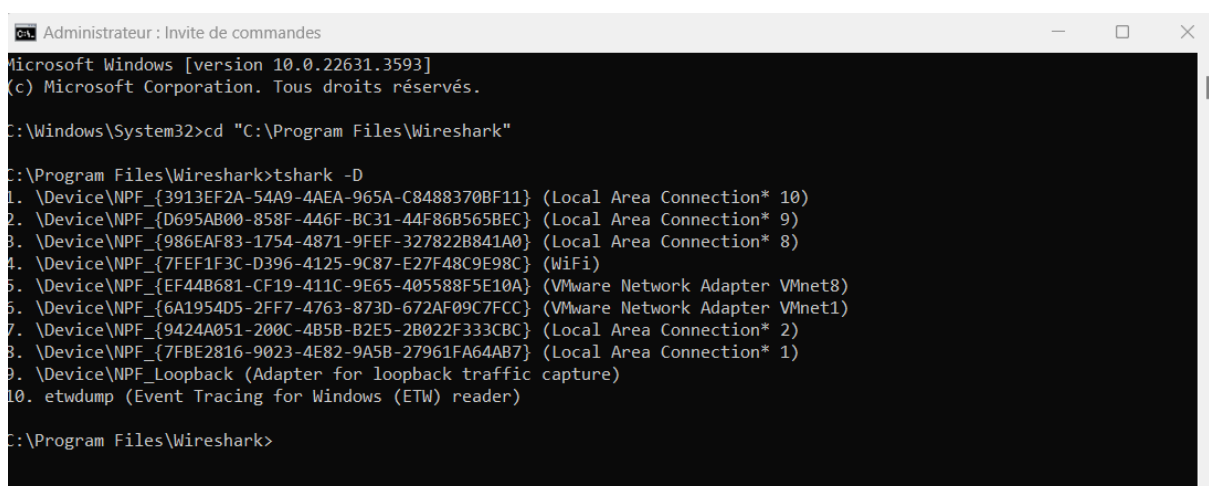
**C:\Program Files\Wireshark**



## Afficher les interfaces réseau

Maintenant, tout est bon pour l'utilisation de la commande `tshark` dans le terminal de l'ordinateur. Pour faire un test, on peut exécuter la commande suivante pour afficher les interfaces réseau disponibles sur l'ordinateur :

## tshark -D





## Sélectionner le réseau

Nous pouvons sélectionner le réseau en utilisant le numéro ou le nom du réseau. Pour ce faire, il faut exécuter la commande suivante pour capturer les paquets du réseau choisi :

### tshark -i 2

### tshark -i WiFi

On peut constater les packet capturés sur **WiFi**

```
Administrateur : Invite de commandes
C:\Program Files\Wireshark>tshark -i WiFi
Capturing on 'WiFi'
1  0.000000 CloudNetwork_6f:e5:8d → Broadcast    ARP 60 Who has 10.10.26.145? Tell 10.10.36.47 CloudNetwork_6f:e5:8d Broadcast
2  0.000000 Intel_d5:20:55 → Broadcast    ARP 60 Who has 10.10.32.193? Tell 10.10.34.159 Intel_d5:20:55 Broadcast
3  0.000084 Intel_d5:20:55 → Broadcast    ARP 60 Who has 10.10.36.144? Tell 10.10.34.159 Intel_d5:20:55 Broadcast
4  0.000084 Intel_d5:20:55 → Broadcast    ARP 60 Who has 10.10.32.175? Tell 10.10.34.159 Intel_d5:20:55 Broadcast
5  0.000084 Intel_d5:20:55 → Broadcast    ARP 60 Who has 10.10.36.126? Tell 10.10.34.159 Intel_d5:20:55 Broadcast
6  0.000084 10.10.21.62 → 10.10.255.255 NBNS 92 Name query NB C824-029813<00> AzureWaveTec_77:63:b1 Broadcast
7  0.001634 10.10.36.54 → 224.0.0.251 MDNS 183 Standard query response 0x0000 PTR VCCApi-3793d86a.vccapi._tcp.local TXT SRV 0 0 5477 VCCApi-3793d86a.vccapi.local A 127.0.0.1 AzureWaveTec_c1:ea:2a IPv4mcast_fb
8  0.002331 fe80::214f:b506:6096:5763 → ff02::fb MDNS 107 Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question Intel_cf:26:eb IPv6mcast_fb
9  0.002331 Intel_d5:20:55 → Broadcast    ARP 60 Who has 10.10.32.184? Tell 10.10.34.159 Intel_d5:20:55 Broadcast
10 0.101820 Intel_d5:20:55 → Broadcast    ARP 60 Who has 10.10.32.174? Tell 10.10.34.159 Intel_d5:20:55 Broadcast
11 0.115351 10.10.35.146 → 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1 CloudNetwork_10:25:d1 CloudNetwork_65:c5:95
12 0.125157 10.10.36.153 → 10.10.0.87 TCP 171 53976 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=117 [TCP segment of a reassembled PDU] CloudNetwork_65:c5:95 ActionsMicro_42:02:fa
13 0.132166 10.10.0.87 → 10.10.36.153 TCP 171 8009 → 53976 [PSH, ACK] Seq=1 Ack=118 Win=660 Len=117 [TCP segment of a reassembled PDU] ActionsMicro_42:02:fa CloudNetwork_65:c5:95
14 0.167602 10.10.36.154 → 239.255.255.250 SSDP 336 NOTIFY * HTTP/1.1 ActionsMicro_a9:f8:91 CloudNetwork_65:c5:95
15 0.167602 10.10.36.154 → 239.255.255.250 SSDP 390 NOTIFY * HTTP/1.1 ActionsMicro_a9:f8:91 CloudNetwork_65:c5:95
16 0.167602 10.10.36.154 → 239.255.255.250 SSDP 402 NOTIFY * HTTP/1.1 ActionsMicro_a9:f8:91 CloudNetwork_65:c5:95
17 0.167602 10.10.36.154 → 239.255.255.250 SSDP 400 NOTIFY * HTTP/1.1 ActionsMicro_a9:f8:91 CloudNetwork_65:c5:95
18 0.167602 10.10.36.154 → 239.255.255.250 SSDP 392 NOTIFY * HTTP/1.1 ActionsMicro_a9:f8:91 CloudNetwork_65:c5:95
19 0.167602 10.10.36.154 → 239.255.255.250 SSDP 345 NOTIFY * HTTP/1.1 ActionsMicro_a9:f8:91 CloudNetwork_65:c5:95
20 0.183863 10.10.35.231 → 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1 Intel_57:cd:1c CloudNetwork_65:c5:95
21 0.187934 10.10.36.153 → 10.10.0.87 TCP 54 53976 → 8009 [ACK] Seq=118 Ack=118 Win=511 Len=0 CloudNetwork_65:c5:95
```