**Network sniffing**

Analyse réseaux avec **wireshark**



projet réalisé par Mansour, Rija, Rheda et Manon

Avant-projet Network sniffing :

1. Qu'est ce que le wireshark ?
2. Quelle est la différence entre une trame et un paquet ?
3. Qu’est-ce que le format pcap/pcapng ?

* Wireshark en root sur Linux.
* Interface (navigateur ?) qui connecte à alcasar.
* Écouter et capturer des paquets ( ARP, UDP , TCP)
* Désencapsuler les trames pour retrouver les différentes couches du modèle OSI.

1. Quelles sont les adresses MAC sources, les IP sources et les adresses MAC sources, les IP destinations des données capturées ?
2. Référencez d’autres trames ou paquets circulants sur le réseau. Identifiez leurs protocoles et leur fonction.

* cherchez les spécifications du format des messages ARP/UDP/TCP
* Faites correspondre les captures en hexadécimal
* Essayez de trouver les paquets correspondants aux étapes de  
  connexion entre votre hôte et un serveur. (SYN ACK FIN ...).

1. Décrivez le mécanisme de connexion avec un diagramme manon

* utiliser les filtres de Wireshark.

1. Documentez vous sur le sujet et faites quelques tests pour n’afficher que les trames qui vous intéressent

PARTIE 2 :

1. Installer les services qui vous permettront d’écouter quelques-uns des protocoles suivants :

* Message
* DHCP
* Message DNS
* Message mDNS ? (qu’est-ce que c’est ?)
* Message SSL
* Message FTP
* Message SMB
* Messages HTTPS
* Message TLSv1.2
* Capturez à présent les paquets en utilisant des filtres Wireshark.

1. Interpréter les paquets capturés avec les spécifications des protocoles .
2. 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 ?

PARTIE 3

1. Utiliser des scripts pour écouter le réseau et extraire les données.

Installez le paquet nécessaire : ( La commande s’appelle tshark)

Trouvez la commande qui permets d’écouter et capturer les paquets de quelques- uns de protocoles (en lien avec la partie 2)

1. Quelles sont les différentes options que nous utilisons ? Mansour

* Filtrer les captures :   
  ➔ Directement avec des options de tshark

➔ En redirigeant son résultat

**PROJET**

1. **Qu'est ce que le wireshark ?**

Wireshark est un puissant logiciel analyseur de paquets réseau open-source, largement utilisé dans le domaine de la cybersécurité et par les administrateurs réseau. Un analyseur de paquets, aussi appelé sniffer de réseau, permet de capturer et d'examiner les données circulant sur un réseau informatique en temps réel. En d'autres termes, Wireshark vous permet de voir ce qui se passe sur votre réseau à un niveau très détaillé.

***Pourquoi utiliser Wireshark ?***

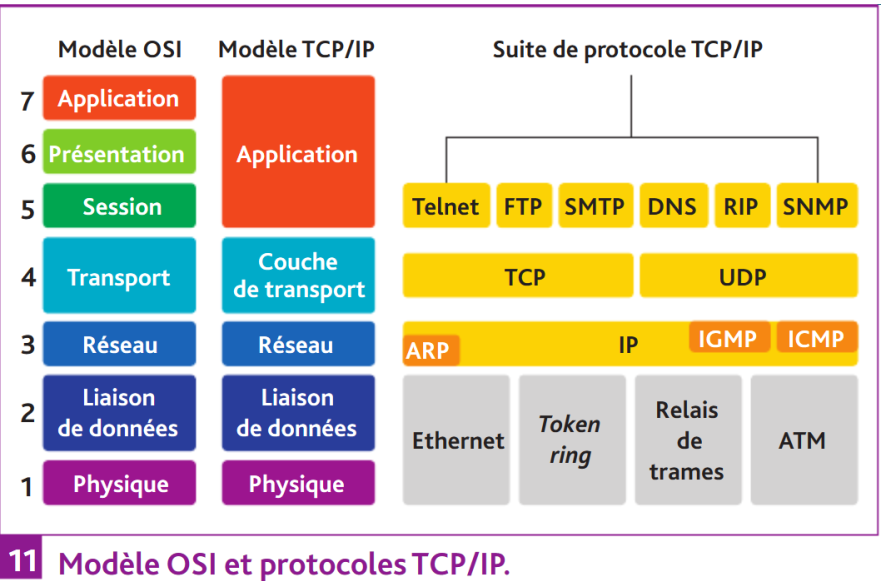
* Dépannage réseau
* Analyse de sécurité
* Développement et test de protocoles
* Rétro-ingénierie des protocoles

***Qui utilise Wireshark ?***

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.

1. **Quelle est la différence entre une trame et un paquet ?**

*Une trame est utilisée pour la transmission de données sur un lien physique local (couche 2), tandis qu'un paquet est utilisé pour l'acheminement des données à travers différents réseaux (couche 3).*

*RAPPEL:  
*

**ARP** : Trouve les adresses MAC à partir des adresses IP dans un réseau local.

**UDP** : Protocole rapide et léger pour envoyer des messages courts sans garantie de livraison.

**TCP** : Protocole fiable et orienté connexion pour des communications nécessitant une haute fiabilité.

**Trame (Frame) :**

Couche : Utilisée à la couche de liaison de données (couche 2) du modèle OSI.

Fonction : Encapsule les données pour la transmission sur un lien physique.

Structure : Contient des informations spécifiques à la liaison, comme les adresses MAC et le contrôle d'erreurs (par exemple, CRC).

Exemple : Ethernet, Wi-Fi.

**Paquet (Packet) :**

Couche : Utilisé à la couche réseau (couche 3) du modèle OSI.

Fonction : Transporte les données à travers différents réseaux et segments de réseau.

Structure : Inclut des adresses IP source et destination, ainsi que des informations de routage et de protocole.

Exemple : IP (Internet Protocol).

Relation

Les données sont encapsulées dans un paquet au niveau de la couche réseau.

Ce paquet est ensuite encapsulé dans une trame pour la transmission au niveau de la couche de liaison de données.

**En résumé, la trame est utilisée pour la transmission locale sur un lien physique, tandis que le paquet est utilisé pour le routage à travers plusieurs réseau**x.

1. **Qu’est-ce que le format pcap/pcapng ?**

**Pcap (packet capture)** est un format de fichier standard utilisé pour stocker les paquets capturés à partir d'un réseau. Il est couramment utilisé dans le domaine de l'analyse du réseau et de la sécurité informatique pour enregistrer le trafic réseau à des fins d'inspection et de débogage.

Le format Pcap original a été introduit avec la bibliothèque libpcap et présente certaines limitations, notamment :

* Incapacité à stocker des paquets avec des types de couche de liaison différents
* Limites de taille de fichier
* Manque de prise en charge des métadonnées étendues

**Pcapng (pcap next generation)** est un format de fichier plus récent conçu pour surmonter ces limitations. Il offre plusieurs avantages par rapport au format Pcap original, notamment :

* Prise en charge de divers types de couche de liaison
* Prise en charge de fichiers plus volumineux
* Champs de métadonnées étendues pour des informations supplémentaires sur les paquets capturés
* Amélioration de la scalabilité et des performances

Voici un tableau récapitulatif des principales différences entre les formats Pcap et Pcapng:  

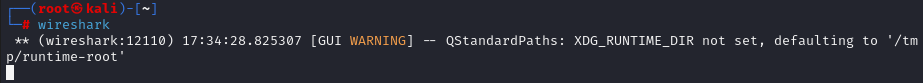

**En général, Pcapng est le format de fichier préféré pour la capture de paquets** en raison de ses avantages en termes de flexibilité, de scalabilité et de performances. Cependant, le format Pcap est toujours largement utilisé et pris en charge par de nombreux outils d'analyse réseau.

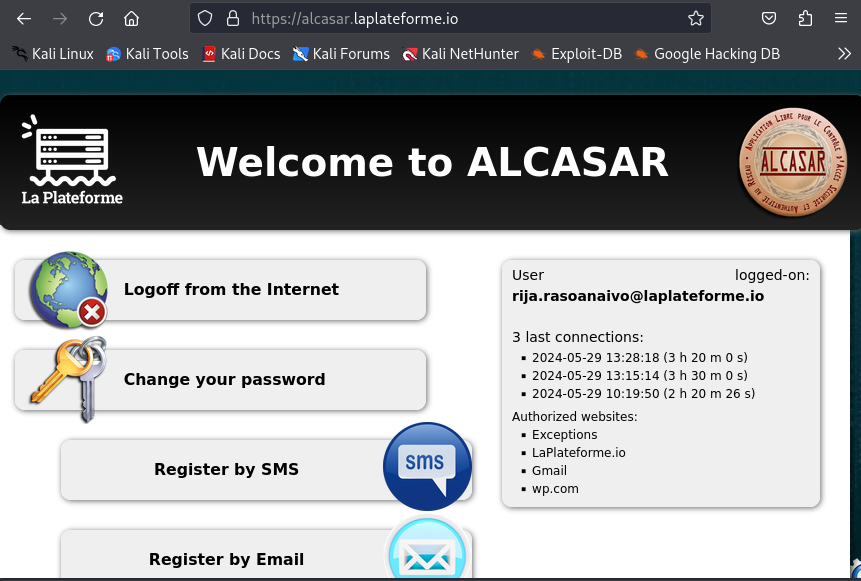
Voici quelques ressources supplémentaires pour en savoir plus sur les formats Pcap et Pcapng :

* <https://en.wikipedia.org/wiki/Pcap>
* <https://pcapng.com/>
* <https://www.solarwinds.com/resources/it-glossary/pcap>
* <https://www.it-connect.fr/manipulation-avancee-des-fichiers-pcap-avec-editcap/>

Pour ce projet, nous avons décidé d’utiliser l’**OS KaliLinux.**

**- Lancer wireshark en root:**Nous nous mettons en utilisateur root et on tape la commande suivante : 



**- Interface connectée à alcasar :  
**

**- Écouter et capturer des paquets ( ARP, UDP , TCP) :**

**ARP**:  


| 96 | Il s'agit du numéro de la trame dans la capture Wireshark. C'est la 96ème trame capturée. |
| --- | --- |
| 14.700033660 | Ce champ indique le temps écoulé depuis le début de la capture jusqu'à la capture de cette trame, en secondes. Ici, cela signifie que la trame a été capturée 14.700033660 secondes après le début de la capture. |
| VMWare\_c0:00:08 | Il s'agit de l'adresse MAC de l'appareil qui a envoyé cette trame. "VMware\_c0:00:08" est une adresse MAC qui indique que l'appareil est probablement une machine virtuelle VMware. |
| VMWare\_f9:15:e2 | Il s'agit de l'adresse MAC de l'appareil à qui la trame est destinée. "VMware\_f9:15  " est aussi une adresse MAC, probablement d'une autre machine virtuelle VMware. |
| ARP | Cela indique que cette trame utilise le protocole ARP (Address Resolution Protocol). |
| 60 | La longueur de la trame, en octets. |
| Who has 192.168.197.254? Tell 192.168.197.1 | Cela indique le contenu de la requête ARP. En termes simples, l'appareil avec l'adresse MAC "VMware\_c0:00:08" (192.168.197.1) demande à tous les appareils sur le réseau "Qui a l'adresse IP 192.168.197.254 ?" et demande à cet appareil de répondre à "192.168.197.1". |

**Explication de la requête ARP**

Lorsqu'un appareil veut communiquer avec un autre appareil sur le même réseau local et ne connaît que son adresse IP, il utilise ARP pour découvrir l'adresse MAC correspondante.

Dans ce cas :

* L'appareil à l'adresse MAC "VMware\_c0:00:08" (probablement avec l'adresse IP 192.168.197.1) essaie de découvrir l'adresse MAC associée à l'adresse IP 192.168.197.254.
* Il envoie une requête ARP sur le réseau demandant "Who has 192.168.197.254? Tell 192.168.197.1".
* Si un appareil sur le réseau a l'adresse IP 192.168.197.254, il répondra avec son adresse MAC, permettant à l'initiateur de la requête de mettre à jour sa table ARP et de communiquer directement avec l'appareil cible.

**UDP**:  


| 47 | Il s'agit du numéro de la trame dans la capture Wireshark. C'est la 47ème trame capturée. |
| --- | --- |
| 28.435283479 | Ce champ indique le temps écoulé depuis le début de la capture jusqu'à la capture de cette trame, en secondes. Ici, cela signifie que la trame a été capturée 28.435283479 secondes après le début de la capture. |
| 192.168.197.1 | Il s'agit de l'adresse IP de l'appareil qui a envoyé ce paquet UDP. Dans ce cas, l'appareil avec l'adresse IP 192.168.197.1 est l'expéditeur. |
| 192.168.197.140 | Il s'agit de l'adresse IP de l'appareil à qui le paquet UDP est destiné. Dans ce cas, l'appareil avec l'adresse IP 192.168.197.140 est le destinataire. |
| UDP | Cela indique que cette trame utilise le protocole UDP (User Datagram Protocol). |
| 214 | La longueur totale du paquet en octets, y compris l'en-tête UDP et les données. Ici, le paquet fait 214 octets de long. |
| 4002 → 17732 | **Ports source et destination** :   * **4002** est le port source, utilisé par l'expéditeur. * **17732** est le port de destination, utilisé par le récepteur.   Ces ports sont utilisés pour identifier les applications spécifiques qui envoient et reçoivent les données. |
| Len=172 | La longueur des données utiles (payload) transportées par le paquet UDP. Ici, les données utiles font 172 octets. |

### **Explication détaillée**

En résumé, cette ligne décrit un paquet UDP envoyé de l'adresse IP 192.168.197.1 à l'adresse IP 192.168.197.140. Le paquet utilise le port source 4002 et le port de destination 17732. La longueur totale du paquet est de 214 octets, dont 172 octets sont des données utiles (payload).

Le protocole UDP est un protocole de transport sans connexion, ce qui signifie que ce paquet a été envoyé sans établissement préalable de connexion et sans garantie de livraison, de commande, ou de détection de duplicata.

**TCP**:  


| 7 | Il s'agit du numéro de la trame dans la capture Wireshark. C'est la 7ème trame capturée. |
| --- | --- |
| 0.517079763 | Ce champ indique le temps écoulé depuis le début de la capture jusqu'à la capture de cette trame, en secondes. Ici, cela signifie que la trame a été capturée 0.517079763 secondes après le début de la capture. |
| 192.168.197.137 | Il s'agit de l'adresse IP de l'appareil qui a envoyé ce segment TCP. Dans ce cas, l'appareil avec l'adresse IP 192.168.197.137 est l'expéditeur. |
| 62.257.227.224 | Il s'agit de l'adresse IP de l'appareil à qui le segment TCP est destiné. Dans ce cas, l'appareil avec l'adresse IP 65.254.227.224 est le destinataire. |
| TCP | Cela indique que ce segment utilise le protocole TCP (Transmission Control Protocol). |
| 54 | La longueur totale du segment en octets, y compris l'en-tête TCP et les données. Ici, le segment fait 54 octets de long. |
| 40006 → 80 [ACK] | **Ports source et destination :** **- 40006** est le port source, utilisé par l'expéditeur. **- 80** est le port de destination, généralement utilisé pour les requêtes HTTP.  [ACK] indique que ce segment est un accusé de réception (Acknowledgment). |
| Seq=1 Ack=1 | Numéros de séquence et d'accusé de réception :   * Seq=1 signifie que le numéro de séquence du premier octet de données dans ce segment est 1. * Ack=1 signifie que l'expéditeur de ce segment accuse réception de toutes les données jusqu'au numéro de séquence 1 du destinataire. |
| Win=32120 | Taille de la fenêtre :  Cela indique la taille de la fenêtre de réception, en octets. La fenêtre de réception est un mécanisme de contrôle de flux qui informe l'émetteur du nombre d'octets qu'il peut encore envoyer sans recevoir d'accusé de réception supplémentaire. Ici, la taille de la fenêtre est de 32120 octets. |
| Len=0 | La longueur des données utiles (payload) transportées par ce segment TCP. Ici, il n'y a pas de données utiles, donc la longueur est de 0. |

### **Explication détaillée**

En résumé, cette ligne décrit un segment TCP envoyé de l'adresse IP 192.168.197.137 à l'adresse IP 65.254.227.224. Le segment utilise le port source 40006 et le port de destination 80 (port standard pour les requêtes HTTP). Ce segment est un accusé de réception (ACK) qui indique que l'expéditeur a reçu des données jusqu'au numéro de séquence 1 du destinataire. Le numéro de séquence du segment lui-même est 1. La taille de la fenêtre de réception est de 32120 octets et il n'y a pas de données utiles dans ce segment (Len=0).

### **Contexte de l'accusé de réception (ACK)**

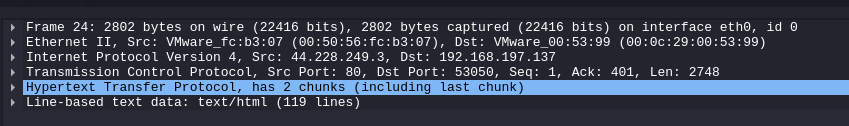
Dans une connexion TCP, l'établissement et la gestion de la connexion se font par le biais de segments avec différents indicateurs (flags) comme SYN (synchronize), ACK (acknowledgment), FIN (finish), etc. Voici quelques détails supplémentaires :

* **Établissement de la connexion :** Une connexion TCP commence par un échange de segments SYN pour synchroniser les numéros de séquence, suivi par un segment SYN-ACK, et enfin par un segment ACK.
* **Transfert de données :** Une fois la connexion établie, les données sont envoyées et accusées de réception avec des segments ACK pour confirmer la réception des données.
* **Contrôle de flux :** La taille de la fenêtre (Win) aide à contrôler le flux de données en informant l'émetteur de la quantité de données qu'il peut envoyer avant de devoir attendre un accusé de réception.

Dans ce cas précis, le segment avec l'indicateur [ACK] montre que l'expéditeur 192.168.197.137 a reçu les données jusqu'au numéro de séquence 1 et envoie cette information au destinataire 65.254.227.224. Ce type de segment est typique dans la gestion continue de la connexion TCP pour assurer la fiabilité et le contrôle de flux.

**- Désencapsuler les trames pour retrouver les différentes couches du modèle OSI :**Pour Désencapsuler une trame; il faut cliquer deux fois dessus ****

Une fenêtre apparaît, Wireshark peut vous aider à **désencapsuler les trames** et à **visualiser les données** qui traversent chaque couche.



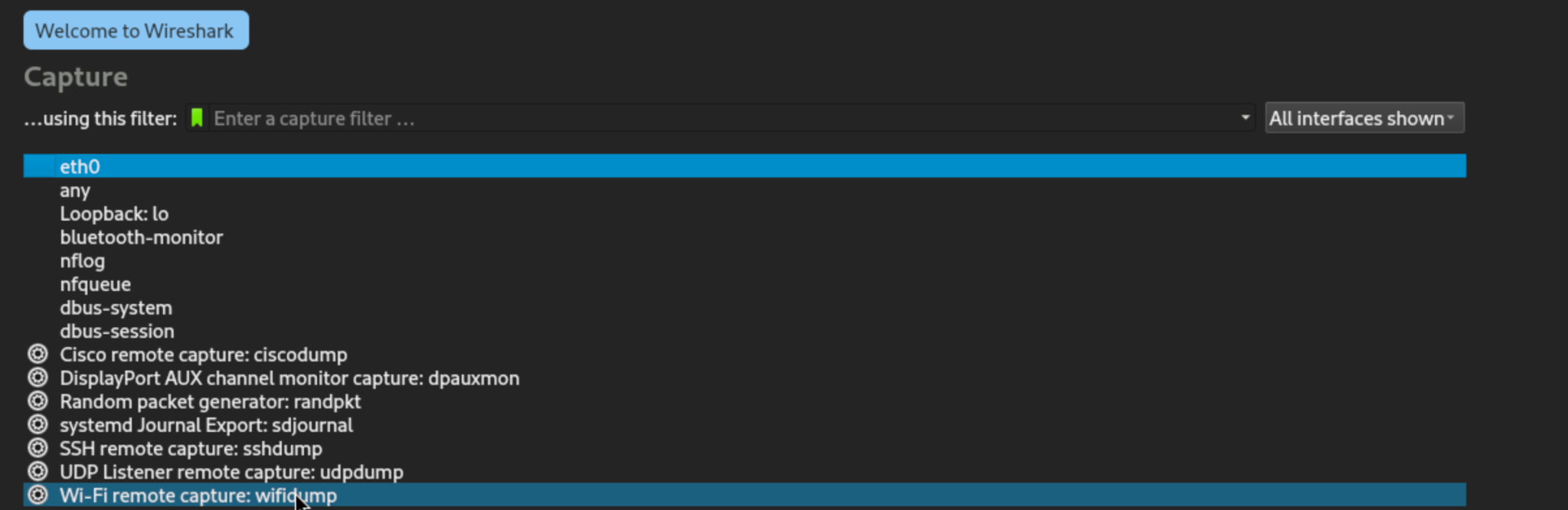
**4 - Quelles sont les adresses MAC sources, les IP sources et les adresses MAC sources, les IP destinations des données capturées ?**

1. **via le terminal lancer le wireshark/lancer le serveur**

* sudo wireshark

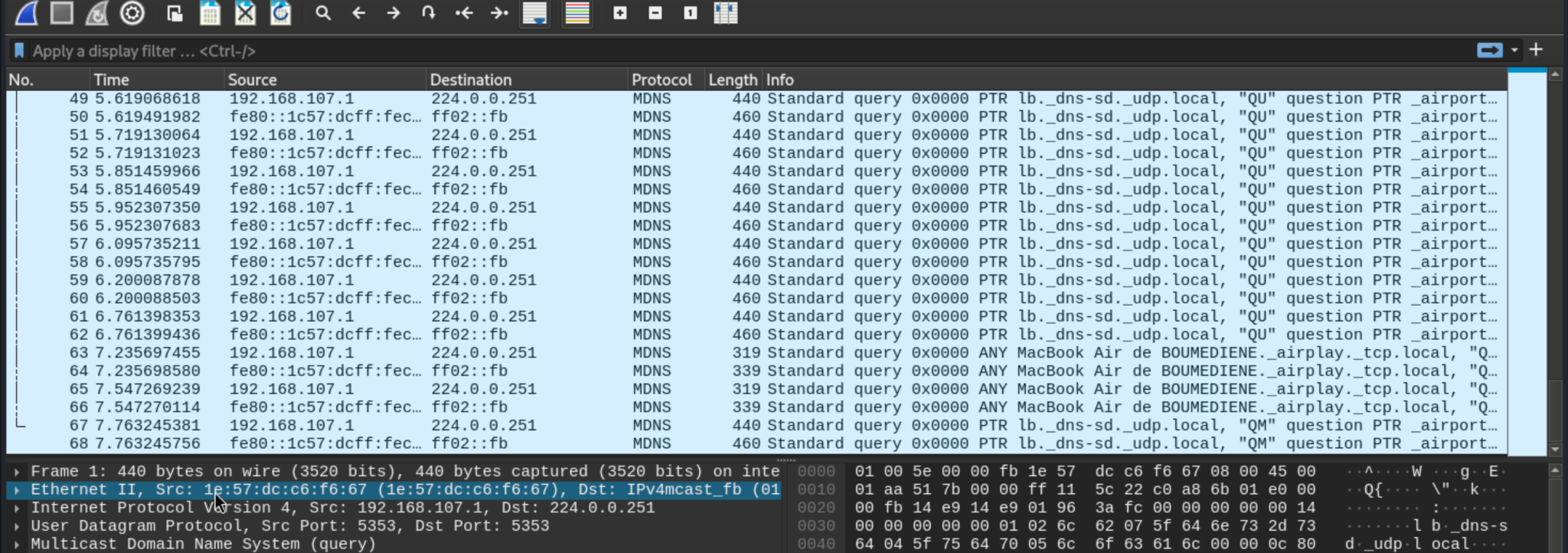
1. **sélection de l’interface réseau**

* double clique



1. **pour faire une capture d’écran**

* bouton démarrer/arrêter pour générer des paquets



**chaque ligne représente :**

* le numéro unique de séquence du paquets
* le moment ou le paquet a était capturé
* la source ( adresse ip ou MAC de l'émetteur du paquet )
* la destination ( adresse ip ou MAC du récepteur du paquet )
* le protocol utilisée ( permet de voir la nature du trafic réseau)
* affichage de la longueur du paquet
* des informations supplémentaires comme les méthodes de requêtes (GET, POST, autres …) + un résumé du paquet.

5 - Référencez d’autres trames ou paquets circulants sur le réseau. Identifiez leurs protocoles et leur fonction.

* cherchez les spécifications du format des messages ARP/UDP/TCP

Voici un résumé des spécifications des formats des messages pour les protocoles ARP, UDP et TCP.

**1. ARP (Address Resolution Protocol)**

Source : [RFC 826](https://tools.ietf.org/html/rfc826)\*\*

L'en-tête ARP inclut :

- **Hardware Type (2 octets) :** Type de réseau (1 pour Ethernet).

- **Protocol Type (2 octets)** : Protocole (0x0800 pour IPv4).

**- Hardware Address Length (1 octet)** : Longueur de l'adresse MAC (6 pour Ethernet).

**- Protocol Address Length (1 octet)** : Longueur de l'adresse IP (4 pour IPv4).

**- Operation\*\* (2 octets) :** Type d'opération (1 pour requête, 2 pour réponse).

**- Sender Hardware Address (6 octets)** : Adresse MAC de l'expéditeur.

**- Sender Protocol Address (4 octets) :** Adresse IP de l'expéditeur.

**- Target Hardware Address (6 octets)** : Adresse MAC du destinataire (0 dans une requête).

**- Target Protocol Address (4 octets) :** Adresse IP du destinataire.

**2. UDP (User Datagram Protocol)**

Source : [RFC 768](https://tools.ietf.org/html/rfc768)

L'en-tête UDP inclut :

- Source Port\*\* (2 octets) : Port source.

-Destination Port\*\* (2 octets) : Port de destination.

-Length\*\* (2 octets) : Longueur totale du datagramme, incluant l'en-tête et les données.

- Checksum\*\* (2 octets) : Vérifie l'intégrité des données.

3. TCP (Transmission Control Protocol)

Source : [RFC 793](https://tools.ietf.org/html/rfc793)\*\*

L'en-tête TCP inclut :

- Source Port (2 octets) : Port source.

- Destination Port (2 octets) : Port de destination.

- Sequence Number (4 octets) : Numéro de séquence.

- Acknowledgment Number (4 octets) : Numéro d'accusé de réception (si ACK activé).

- Data Offset (4 bits) : Longueur de l'en-tête TCP.

- Flags (9 bits) : Indicateurs de contrôle (URG, ACK, PSH, RST, SYN, FIN).

- Window Size (2 octets) : Taille de la fenêtre de réception.

- Checksum(2 octets) : Vérifie l'intégrité des données.

- Urgent Pointer (2 octets) : Position des données urgentes.

- Options (variable) : Options TCP.

- Data (variable) : Les données de l'application.

Références Complètes :

- ARP : [RFC 826](https://tools.ietf.org/html/rfc826)

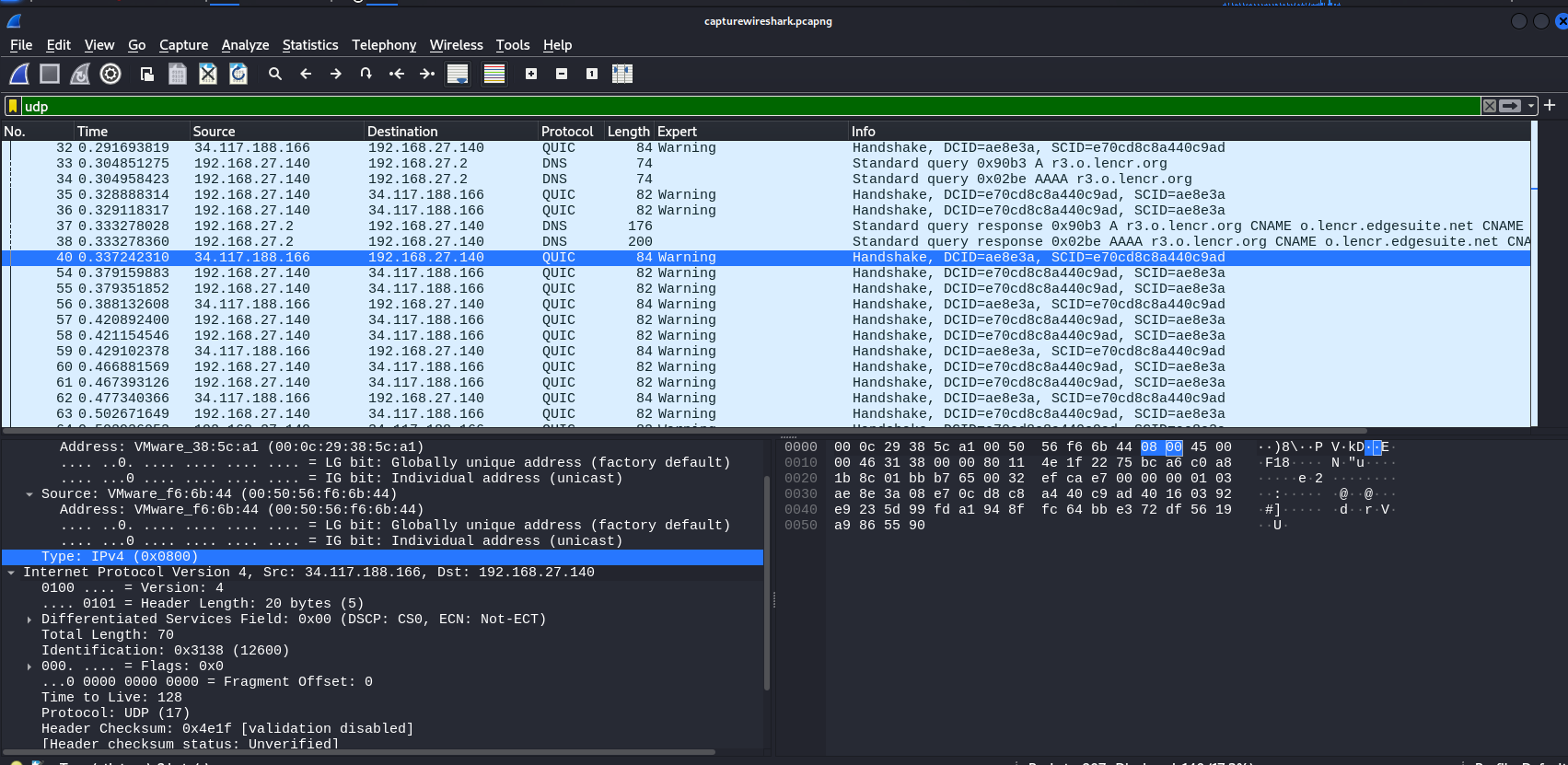
- UDP : [RFC 768](https://tools.ietf.org/html/rfc768)

- TCP : [RFC 793](https://tools.ietf.org/html/rfc793)

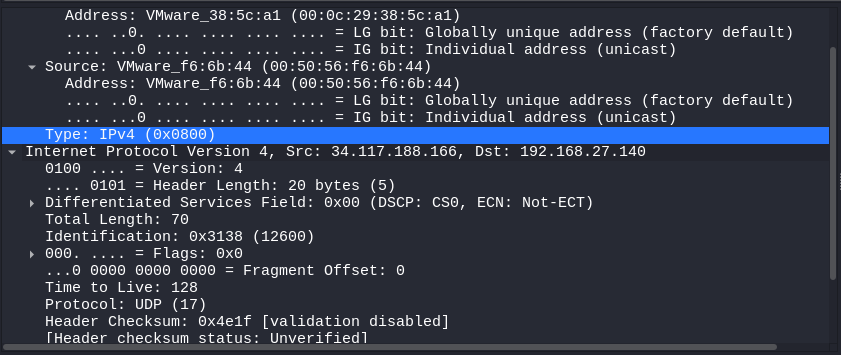
Ces RFC fournissent des spécifications détaillées et sont des ressources essentielles pour comprendre les formats et les fonctions de ces protocoles.

* Faites correspondre les captures en hexadécimal

Pour se faire on fait une capture, on choisi un filtre et on sélectionne le paquet qui nous intéresse pour l’analyser



**Fenêtre de Gauche : Détails des Paquets**

****

Cette fenêtre affiche une vue détaillée et structurée des informations du paquet sélectionné, décomposée en différentes couches du modèle OSI. Voici les éléments clés :

1. En-tête Ethernet :

- Source MAC: `VMware\_38:5c:a1 (00:0c:29:38:5c:a1)`

- Type: IPv4 (`0x0800`)

2. **En-tête IPv4 :**

- Source IP: `192.168.27.140`

- Destination IP: `192.168.27.2`

- Version:4

- Taille Totale: 63 octets

- Identification: `0xf4e4` (62692)

- Flags: `0x2` (Don't Fragment)

- TTL (Time To Live): 64

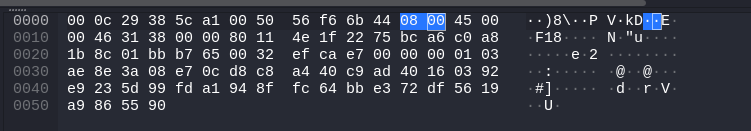
- Protocole: UDP (17)

- Checksum de l'en-tête: `0x8ea0` (validation désactivée)

3. Détails UDP :

- Inclut les ports source et destination, la longueur du paquet UDP, et le checksum UDP (non visible en entier dans l'image).

**Fenêtre de Droite : Vue Hexadécimale et ASCII**

****

Cette fenêtre montre une représentation brute des données du paquet, en hexadécimal et en ASCII :

1. Colonne de Gauche : Offsets

- Numéros indiquant les positions des octets dans le paquet (0000, 0010, 0020, etc.).

2. Colonne du Milieu : Données Hexadécimales

- Représentation des données brutes en hexadécimal. Par exemple, `00 0c 29 38 5c a1` pour l'adresse MAC source.

3. Colonne de Droite : Interprétation ASCII

- Caractères ASCII correspondants aux données hexadécimales. Les caractères non imprimables sont remplacés par des points (`.`).

- Exemple : `63 65 72 74 2d 63 6f 6d` se traduit en `cert-com`.

**Correspondance entre les Fenêtres**

- Adresse MAC Source :

- Visible en détails à gauche et en hexadécimal à droite (`00:0c:29:38:5c:a1` correspond à `00 0c 29 38 5c a1`).

- Type IPv4 :

- Indiqué à gauche (`0x0800`) et visible à droite (`08 00`).

- Données UDP et Application :

- La chaîne ASCII `cert-com` apparaît dans les données hexadécimales à droite et est également identifiable dans les détails à gauche.

**Conclusion**

Les deux fenêtres offrent des vues complémentaires des données du paquet :

- Fenêtre de gauche : Vue détaillée et structurée des informations de chaque couche.

- Fenêtre de droite : Vue brute des données en hexadécimal et ASCII, utile pour une comparaison directe avec des données externes.

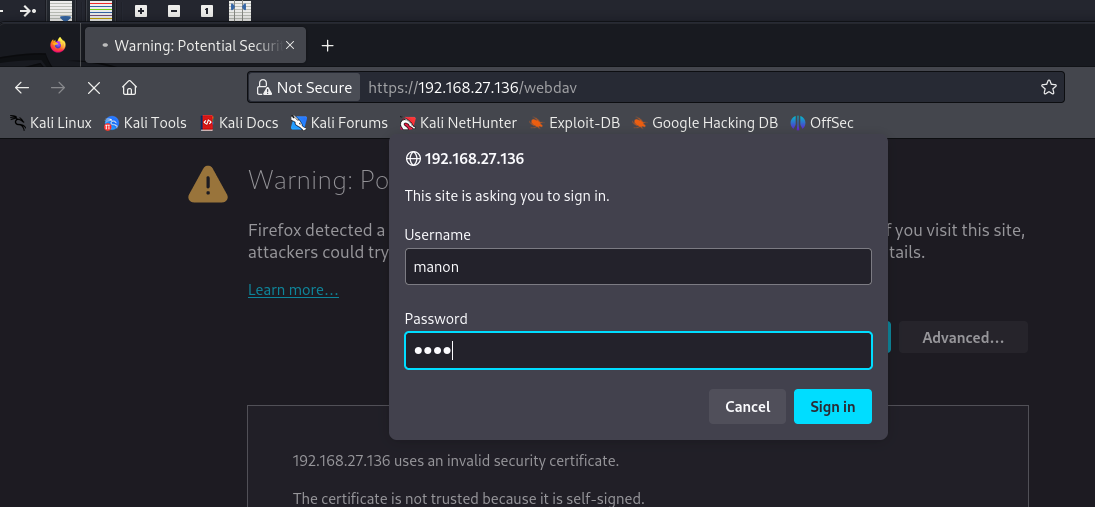
Utilisez ces deux vues ensemble pour analyser et vérifier les paquets réseau en détail.

* Essayez de trouver les paquets correspondants aux étapes de  
  connexion entre votre hôte et un serveur. (SYN ACK FIN ...).

Pour trouver les paquets correspondants aux étapes de connexion entre votre hote et un serveur, il suffit de rentrer la commande suivante dans wireshark pour filtre la capture sur l’adresse ip du serveur “ 192.168.27.136” pour nous correspondant a notre serveur NAS.

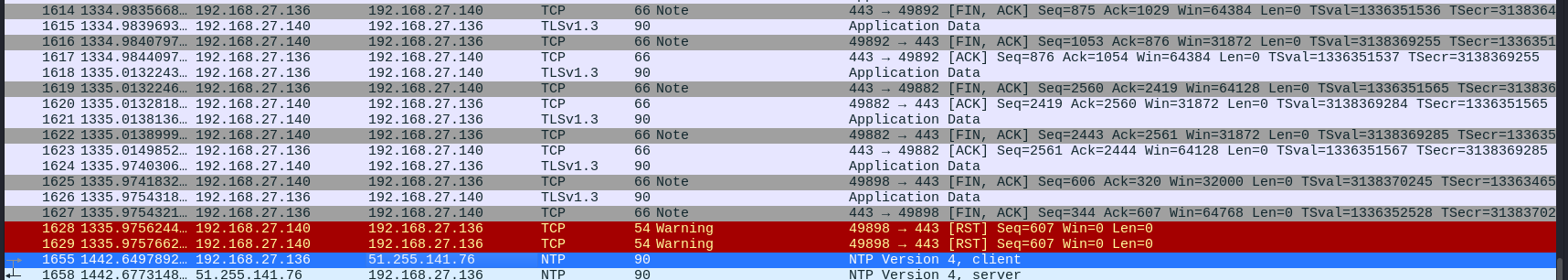
Voici la commande: et lancé la capture

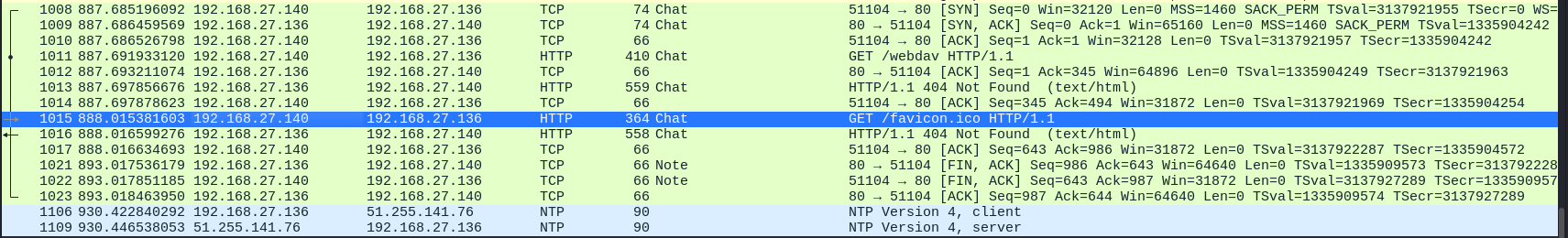
Maintenant allé sur le navigateur de notre vm ou se trouve wireshark et essayé de se connecté au serveur nas

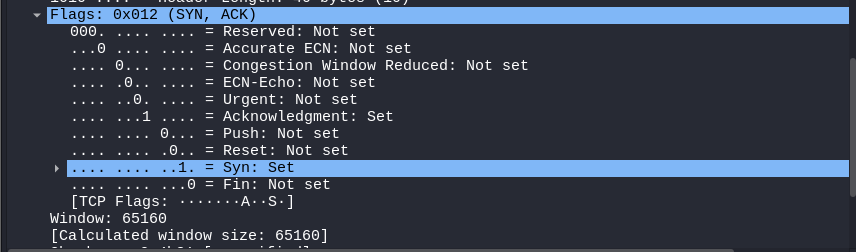


retourné sur wireshark pour voir ce qu’il en est









Ce paquet fait partie de l'établissement d'une connexion TCP, également connu sous le nom de "Three-way Handshake" (Poignée de main à trois étapes) :

SYN (Synchronize)

Envoyé par le client pour initier la connexion.

SYN-ACK (Synchronize-Acknowledgment)

Envoyé par le serveur en réponse au SYN du client, confirmant la réception du SYN et synchronisant également les numéros de séquence.

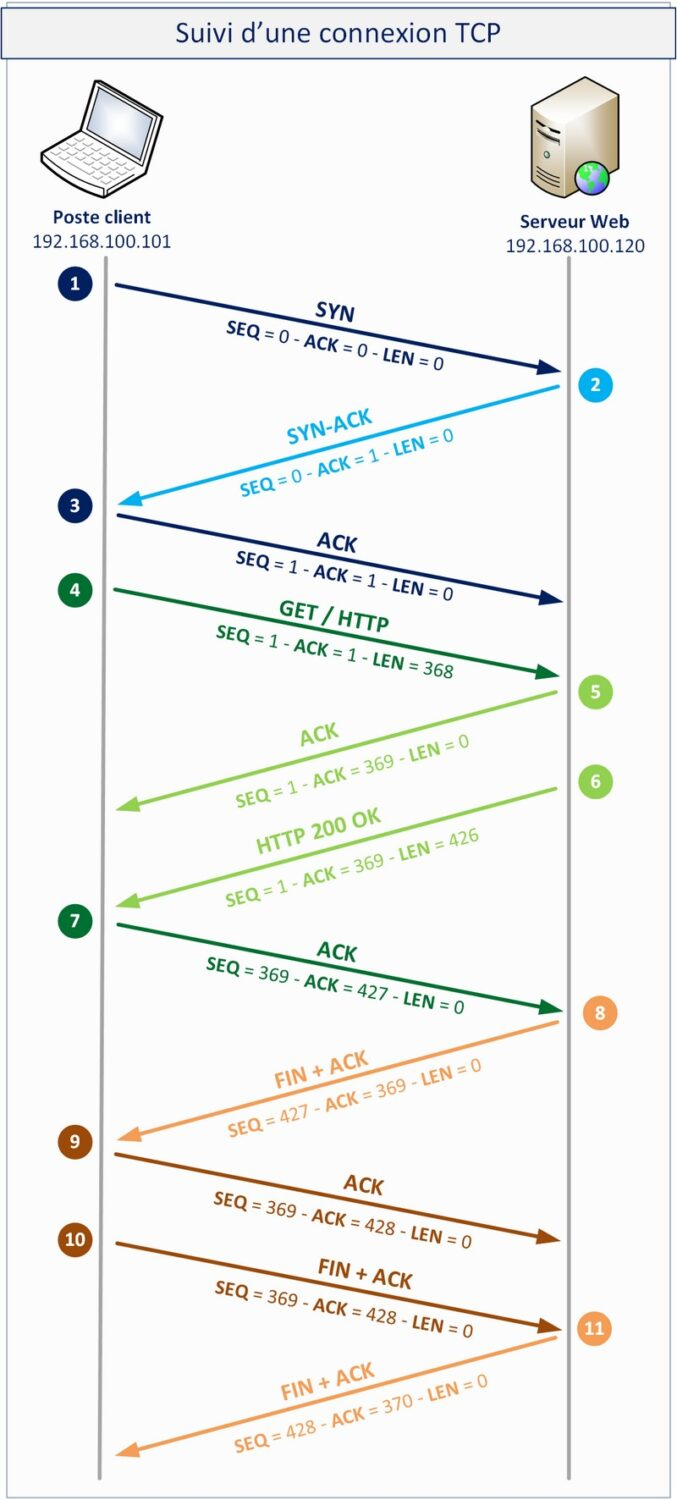
C'est ce que montre votre capture d'écran.

ACK (Acknowledgment)

Envoyé par le client pour accuser réception du SYN-ACK du serveur, finalisant ainsi l'établissement de la connexion.

En résumé, l'image montre un paquet TCP avec les drapeaux SYN et ACK définis, indiquant que le serveur répond au SYN du client et accuser réception, ce qui est la deuxième étape du processus de connexion TCP.

6 - Décrivez le mécanisme de connexion avec un diagramme



EXPLICATION:

**Établissement de la Connexion TCP (Three-Way Handshake)**

1. Paquet n°1 : Demande d'initialisation du client (SYN)  
Le client envoie un paquet SYN pour demander l'établissement d'une connexion TCP avec le serveur.

2. Paquet n°2 : Réponse du serveur (SYN, ACK)  
Le serveur répond au client avec un paquet contenant les flags SYN et ACK pour indiquer qu'il a reçu la demande de connexion et souhaite établir la connexion. Le numéro d'acquittement est 1 pour confirmer la réception du SYN du client.

3. Paquet n°3 : Acquittement du client (ACK)  
Le client envoie un paquet ACK pour confirmer qu'il a reçu le SYN-ACK du serveur. Le numéro de séquence est 1 pour suivre l'acquittement du serveur.

**La connexion TCP est établie entre le client et le serveur.**

**Transfert de Données via TCP**

1. Paquet n°4 : Requête HTTP du client au serveur  
Le client envoie une requête HTTP au serveur pour demander le contenu de la page Web. La taille du segment est de 368 octets, correspondant aux données de la requête.

2. Paquet n°5 : Réponse du serveur (ACK)  
Le serveur envoie un paquet ACK pour confirmer qu'il a reçu les 368 octets de la requête HTTP du client. Le numéro d'acquittement est 369, indiquant la réception complète des données du client.

3. Paquet n°6 : Envoi des données du serveur au client  
Le serveur envoie le contenu de la page Web au client. La taille du segment est de 426 octets, correspondant aux données de la réponse.

4. Paquet n°7 : Acquittement du client (ACK)  
Le client envoie un paquet ACK pour confirmer qu'il a reçu les 426 octets de la réponse HTTP du serveur. Le numéro d'acquittement est 427.

**Les données sont échangées entre le client et le serveur, et les numéros de séquence et d'acquittement évoluent en fonction des octets transférés.**

**Fermeture de la Connexion TCP**

1. Paquet n°8 : Demande de fermeture du serveur (FIN, ACK)  
Le serveur envoie un paquet contenant les flags FIN et ACK pour indiquer qu'il souhaite fermer la connexion TCP.

2. Paquet n°9 : Acquittement du client (ACK)  
Le client envoie un paquet ACK pour confirmer qu'il a reçu la demande de fermeture du serveur. Le numéro d'acquittement est incrémenté de 1 à 428 pour indiquer la réception du flag FIN.

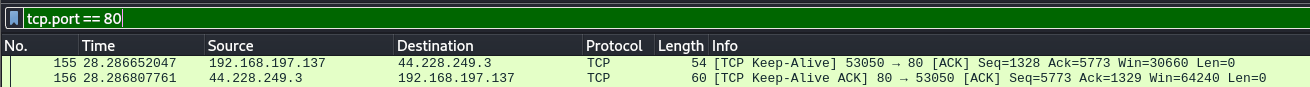
3. Paquet n°10 : Demande de fermeture du client (FIN, ACK)  
Le client envoie un paquet contenant les flags FIN et ACK pour indiquer qu'il est d'accord pour fermer la connexion TCP

4. Paquet n°11 : Acquittement final du serveur (ACK)  
Le serveur envoie un paquet ACK pour confirmer la fermeture de la connexion TCP.  
Le numéro de séquence est 428 et le numéro d'acquittement est 370.

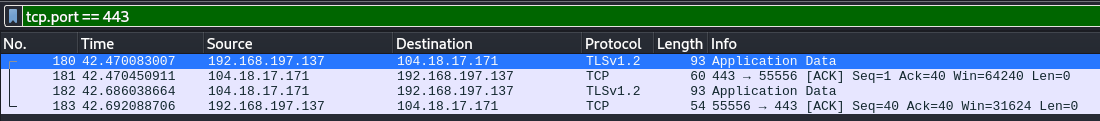
**La connexion TCP est correctement fermée après l'échange des paquets de fermeture entre le client et le serveur.**

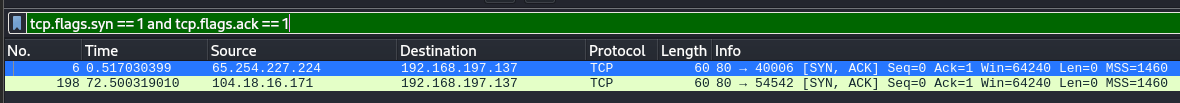
7 - Documentez vous sur le sujet et faites quelques tests pour n’afficher que les trames qui vous intéressent

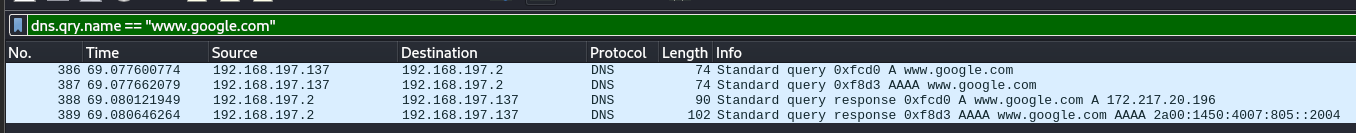
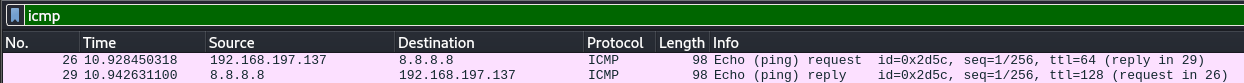
- **Capturer uniquement les connexions TCP vers le port 80 (HTTP):**tcp.port == 80



- **Capturer uniquement les connexions TCP vers le port 443 (HTTPS):**  
tcp.port == 443



**- Afficher les paquets SYN-ACK:**  
tcp.flags.syn == 1 and tcp.flags.ack == 1  


**- Capturer les trames DNS pour le nom de domaine "www.google.com":**  
dns.qry.name == "www.google.com"   
  
  
**- Afficher les paquets ICMP (ping) provenant ou allant vers l'adresse 8.8.8.8:**  
imcp  


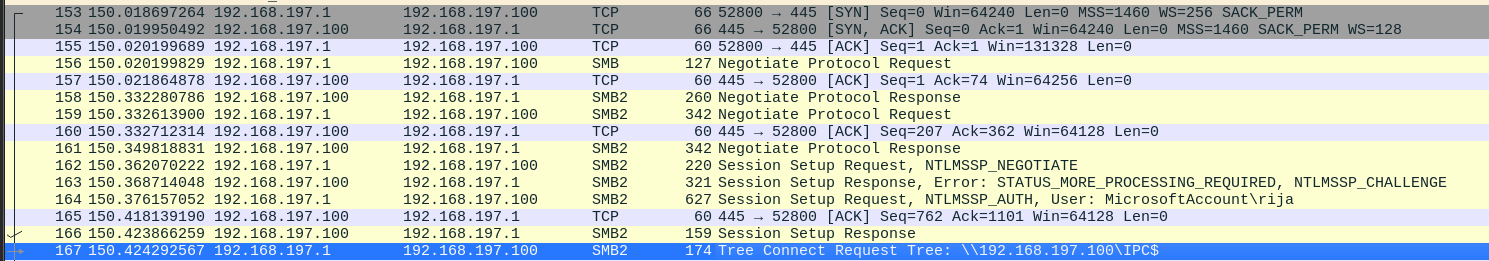
**PARTIE 2**

1 - Installer les services qui vous permettront d’écouter quelques-uns des protocoles suivants :

* Message
* DHCP
* Message DNS
* Message mDNS
* Message SSL
* Message FTP
* Message SMB
* Messages HTTPS
* Message TLSv1.2

Capturez à présent les paquets en utilisant des filtres Wireshark.

**MESSAGE SMB:**

Pour écouter le protocole SMB nous avons créé un réseau local composé d’un serveur NAS et d’un client Windows.  
  
Se connecter au serveur NAS via le client windows:  
Sur Windows, connecter un lecteur réseau et observer ce qui ce passe sur wireshark:  
   
Ci-dessus, de la ligne 153 à 155, le mécanisme de connexion entre le PC Windows et le serveur NAS.  
De la ligne 156 à 161:  
**Ligne 156 :**

* Protocole : SMB2
* Taille : 127 octets
* Description : Cette ligne représente une requête de négociation de protocole SMB2 envoyée par le client au serveur. Elle indique que le client souhaite établir une session SMB2 avec le serveur et fournit des informations sur les capacités du client.

**Ligne 157 :**

* Protocole : TCP
* Taille : 268 octets
* Description : Cette ligne représente une réponse TCP du serveur au client. Elle indique que le serveur a reçu la requête de négociation de protocole SMB2 et est prêt à établir une session.

**Ligne 158 :**

* Protocole : SMB2
* Taille : 342 octets
* Description : Cette ligne représente une réponse de négociation de protocole SMB2 envoyée par le serveur au client. Elle fournit des informations sur les capacités du serveur et confirme que les deux parties sont d'accord sur les paramètres de la session.

**Ligne 159 :**

* Protocole : SMB2
* Taille : 60 octets
* Description : Cette ligne représente une requête de configuration de session SMB2 envoyée par le client au serveur. Elle indique que le client souhaite établir une session SMB2 avec le nom de partage spécifié ("IPC$") et fournit des informations d'authentification NTLMSSP.

**Ligne 160 :**

* Protocole : TCP
* Taille : 342 octets
* Description : Cette ligne représente une réponse TCP du serveur au client. Elle indique que le serveur a reçu la requête de configuration de session SMB2 et est prêt à établir la session.

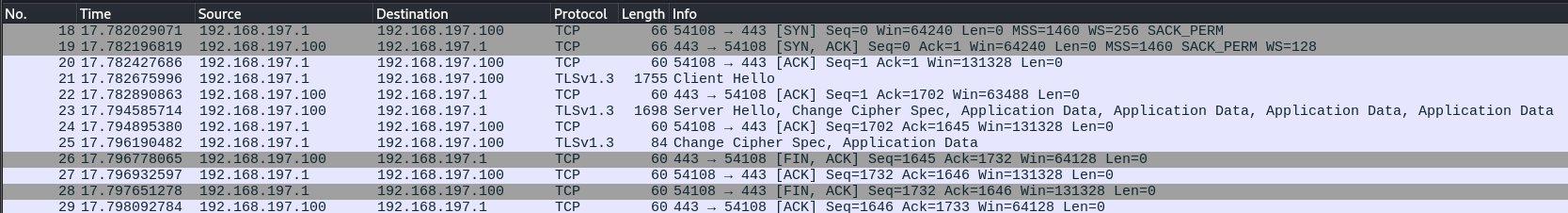
**Ligne 161 :**

* Protocole : SMB2
* Taille : 220 octets
* Description : Cette ligne représente une réponse de configuration de session SMB2 envoyée par le serveur au client. Elle indique que la session SMB2 a été établie avec succès et fournit des informations sur les paramètres de la session.

En résumé, **les lignes 156 à 161 illustrent les étapes finales de l'établissement d'une session SMB2 sécurisée entre un client et un serveur.** Ces étapes impliquent la négociation des capacités des deux parties, l'authentification NTLMSSP et la configuration de la session.

**Message TLS V1.3:**

Pour écouter le protocole TLS V1.3 je me connecte au WebDAV en HTTPS:



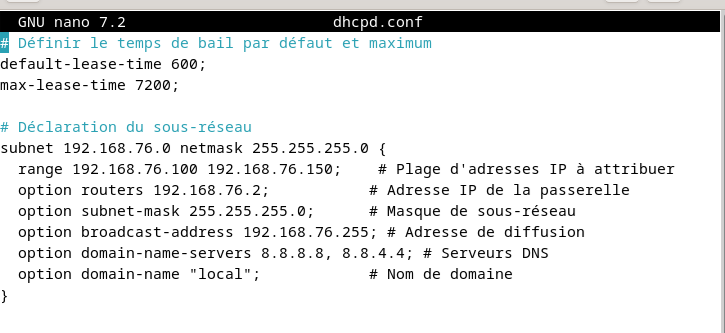
**Ligne 21** : Le client envoie un paquet "Client Hello" pour initier une connexion sécurisée (TLS).

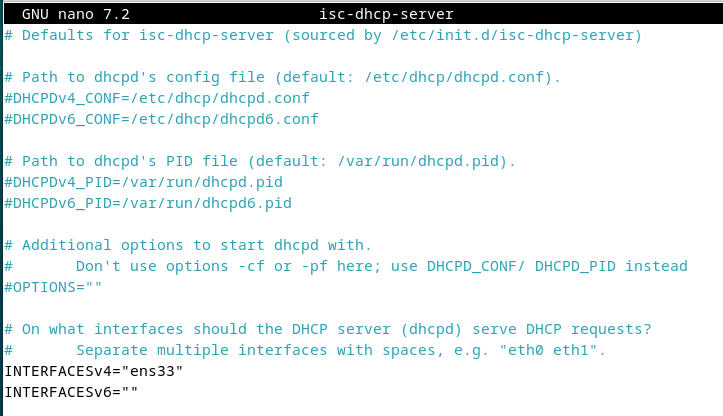
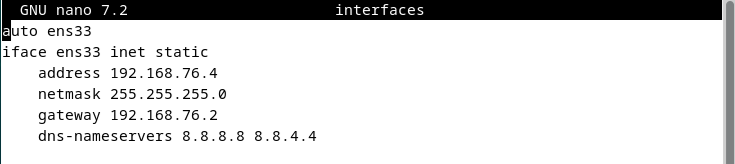
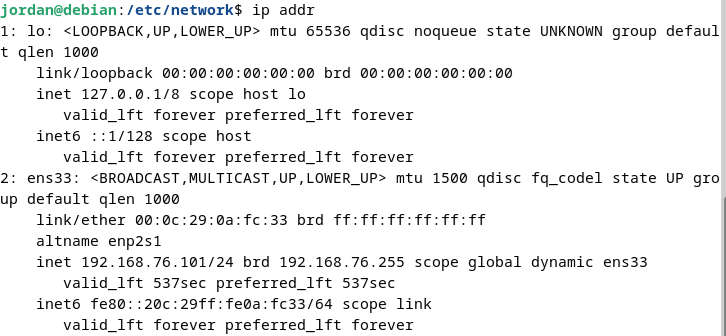
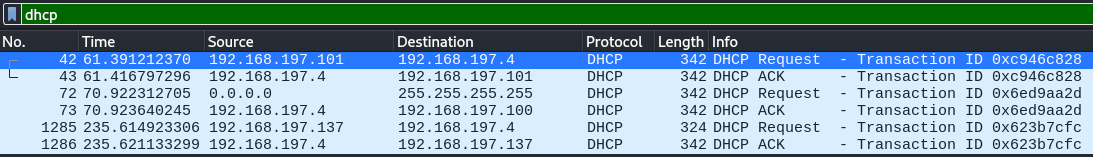
**Ligne 22** : Le serveur confirme la réception du paquet "Client Hello" avec un paquet ACK.

**Ligne 23 :** Le serveur répond avec un paquet "Server Hello", "Change Cipher Spec", et des données d'application pour établir la connexion sécurisée.

**Ligne 24 :** Le client confirme la réception du paquet "Server Hello" avec un paquet ACK.

**Ligne 25 :** Le client envoie un paquet "Change Cipher Spec" et des données d'application pour continuer la configuration sécurisée.

**MESSAGE DHCP:**Pour écouter le protocole DHCP j’installe un serveur DHCP et un client **VM Serveur :** Installation du serveur DHCP  **/etc/dhcp/dhcpd.conf  
**

**/etc/default/isc-dhcp-server  
  
/etc/network/interfaces**  
  
**VM Client :** Configurez la VM client pour obtenir une adresse IP via DHCP.  
  
**/etc/network/interfaces**  
  
**Capture** : Lancez Wireshark sur la VM client pour capturer les paquets DHCP lors de l'initialisation de la connexion réseau.  
  
  
Ces lignes montrent une séquence de communication DHCP (Dynamic Host Configuration Protocol) utilisée pour l'attribution d'adresses IP sur un réseau. Voici une explication simplifiée de chaque ligne :  
  
**Ligne 42 :** Le client (192.168.197.101, le pc Hôte Windows) envoie une requête DHCP (Request) au serveur DHCP (192.168.197.4) pour demander une adresse IP. Transaction ID : 0xc946c828.  
Ci-dessous la source et la destination basée sur l’adresse MAC.  


**Ligne 43 :** Le serveur DHCP (192.168.197.4) envoie un accusé de réception (ACK) au client (192.168.197.101) pour confirmer l'attribution de l'adresse IP. Transaction ID : 0xc946c828.

**Ligne 72 :** Un autre client DHCP (la VM Debian qui vient de s’allumer) envoie une requête DHCP (Request) à **l'adresse de diffusion (broadcast)** pour demander une adresse IP. Transaction ID : 0x6ed90aa2d.  
Ci-dessous la source et la destination basée sur l’adresse MAC.  
  
  
**Voici une explication détaillée de pourquoi le client envoie initialement la requête à l'adresse de diffusion (broadcast) au lieu de l'envoyer directement au serveur DHCP :**  
  
**Phase d'Initialisation du DHCP**

1. **Découverte (DHCP Discover) :**
   * Quand un client DHCP se connecte au réseau pour la première fois et n'a pas encore d'adresse IP, il ne connaît pas l'adresse IP du serveur DHCP.
   * Il envoie une requête de découverte (DHCP Discover) à l'adresse de diffusion (broadcast) 255.255.255.255 pour tenter de localiser un serveur DHCP sur le réseau.
2. **Offre (DHCP Offer) :**
   * Tous les serveurs DHCP qui reçoivent cette requête répondent avec une offre (DHCP Offer) contenant une adresse IP proposée pour le client.
3. **Demande (DHCP Request) :**
   * Le client sélectionne une des offres reçues (s'il en reçoit plusieurs) et envoie une requête (DHCP Request) pour demander l'adresse IP proposée par un des serveurs DHCP. Cette requête est aussi envoyée en diffusion (broadcast) pour informer tous les serveurs DHCP de son choix.
4. **Accusé de Réception (DHCP ACK) :**
   * Le serveur DHCP sélectionné envoie une confirmation (DHCP ACK) pour attribuer l'adresse IP au client.

### **Raisons pour l'Utilisation de l'Adresse de Diffusion**

* **Absence d'Adresse IP Initiale :**
  + Au début, le client n'a pas d'adresse IP valide et ne peut donc pas envoyer de messages à une adresse IP spécifique.
  + En envoyant la requête en diffusion, le client s'assure que tous les serveurs DHCP potentiels sur le réseau local recevront sa requête.
* **Notification Globale :**
  + En envoyant la requête de demande (DHCP Request) en diffusion, le client informe tous les serveurs DHCP potentiels qu'il accepte une offre particulière. Cela permet aux autres serveurs DHCP de savoir qu'ils n'ont pas été sélectionnés.

**Ligne 73 :** Le serveur DHCP (192.168.197.4) envoie un accusé de réception (ACK) au client (192.168.197.100 pour la VM Debian) pour confirmer l'attribution de l'adresse IP. Transaction ID : 0x6ed90aa2d.

**Message mDNS ? (qu’est-ce que c’est ?)**

mDNS, ou multicast Domain Name System, est un protocole réseau qui permet de bénéficier des fonctionnalités du DNS sans nécessiter un serveur DNS centralisé sur le réseau. Il facilite la résolution de noms de domaine pour les appareils sur un réseau local, simplifiant ainsi la découverte et l'accès aux services et aux périphériques.

À travers mDNS, les appareils peuvent annoncer leurs propres noms de domaine et résoudre les noms des autres appareils sur le réseau local sans dépendre d'un serveur DNS central. Cela permet une découverte automatique des services et des périphériques sur le réseau, sans nécessiter de configuration manuelle des adresses IP ou des noms de domaine.

Des exemples concrets d'utilisation de mDNS incluent :

1. La découverte automatique des services tels que les imprimantes, les systèmes de stockage réseau (NAS) ou les appareils IoT sur un réseau domestique.

2. L'accès aux périphériques par leur nom convivial plutôt que par leur adresse IP, facilitant ainsi l'utilisation de ressources réseau.

3. L'intégration et l'interopérabilité entre les appareils IoT de différents fabricants, rendues plus simples grâce à l'annonce des services via mDNS.

4. L'utilisation par les développeurs de logiciels pour le développement et les tests d'applications, permettant de simuler des environnements réseau réalistes.

5. La configuration de réseaux temporaires ou ad-hoc lors d'événements ou de réunions, où mDNS simplifie la communication entre les appareils sans nécessiter de configuration préalable.

En résumé, mDNS simplifie la configuration et l'accès aux appareils sur un réseau local en permettant la résolution de noms de domaine sans avoir besoin d'un serveur DNS centralisé, ce qui le rend particulièrement utile dans divers scénarios d'utilisation, tant domestiques que professionnels.

**Pratique :**

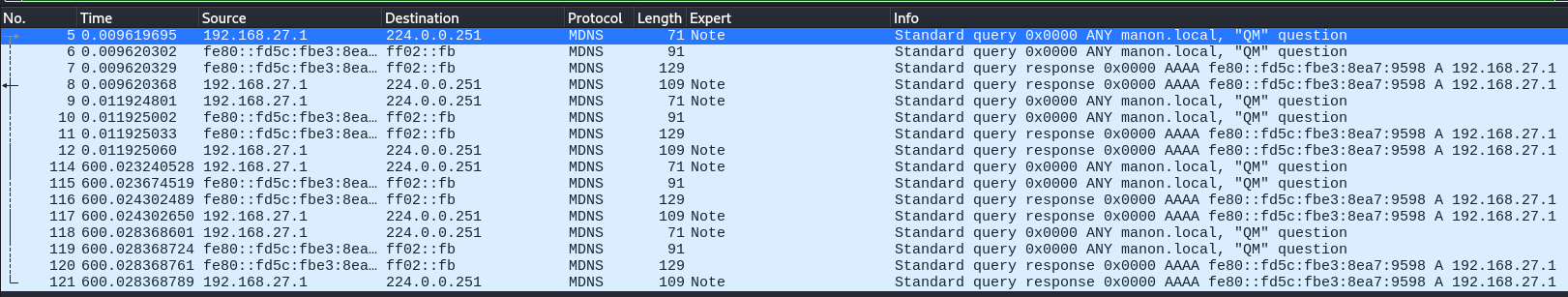
Pour utiliser mDNS, il faut d’abord l'installer via notre terminal   
voici les commandes à faire:

Tout d’abord on commence par mettre à jour les bibliothèques avec update puis apt install pour installer mDNS.  
 

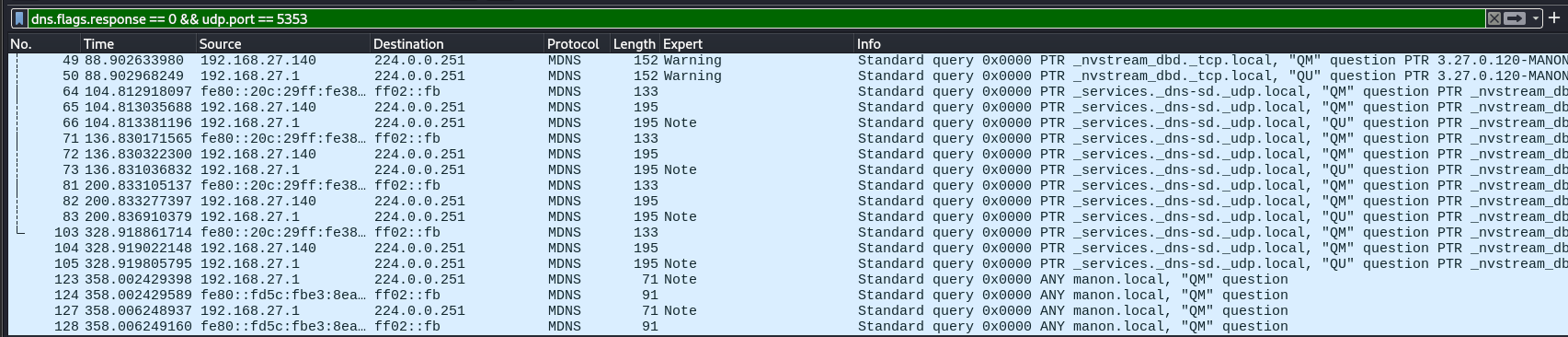
En suite on active le service 

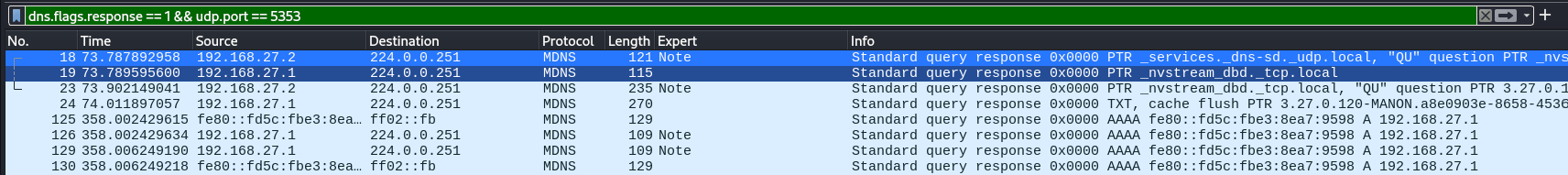
puis on vérifie que le service fonctionne avec cette commande 

on tape cette commande qui permet d’envoyer une requête dns  


Une fois le service installé et actif et la requête envoyée, on se rend sur wireshark pour faire une capture avec le filtre suivant:  
  


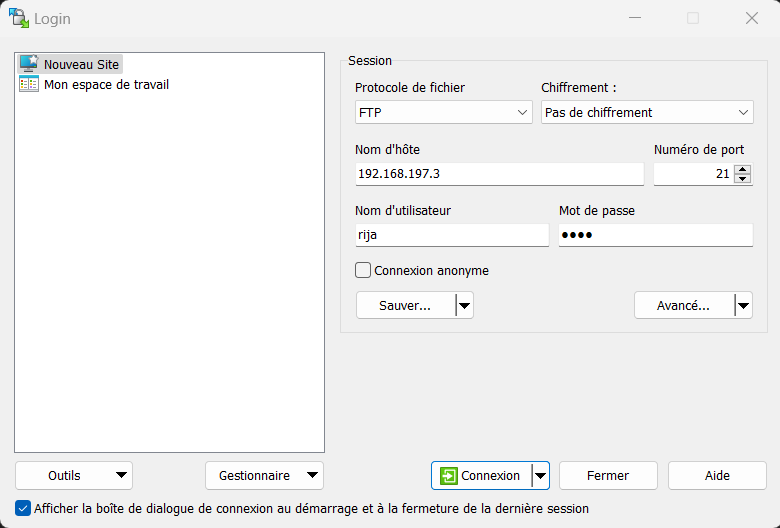
Filtre les paquets capturés pour afficher uniquement les requêtes mDNS  

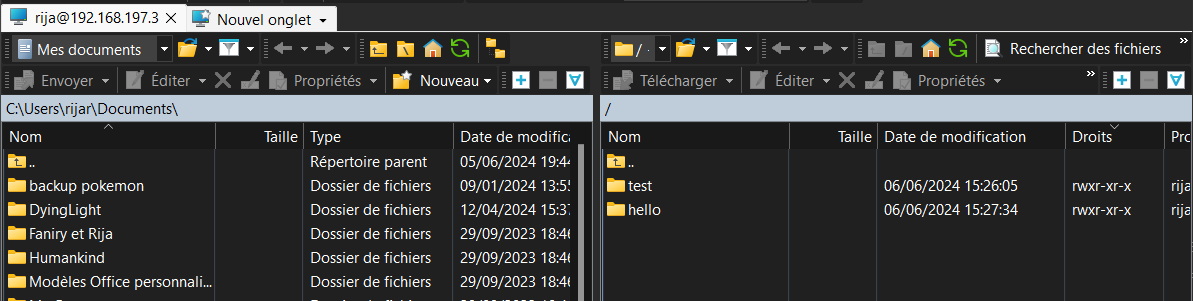
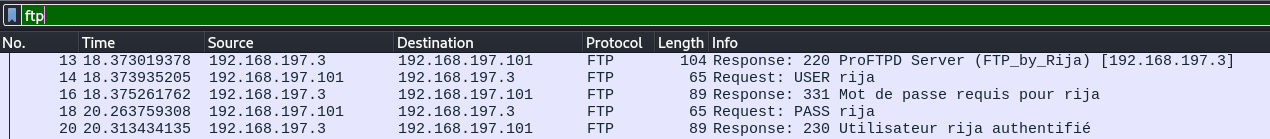



Filtrez les paquets capturés pour afficher uniquement les réponses mDNS  
  


**Message FTP:**

Pour écouter le protocole FTP nous avons créé un serveur FTP avec un utilisateur “rija”. Nous nous somme ensuite connecté au serveur ftp via WinSCP:

**Voici ce que je tape sur WinSCP:  
**

**Je peux maintenant voir ce qu’il y a sur le serveur FTP:  
  
  
Voici ce qui se passe sur WireShark quand je me connecte:  
**

**Ces lignes montrent une séquence d'authentification FTP (File Transfer Protocol) entre un client et un serveur. Voici une explication de chaque ligne :  
  
Ligne 13 :**

* Source : 192.168.197.3 (Serveur FTP).
* Destination : 192.168.197.101 (Client FTP).
* Protocole : FTP.
* Détails : Le serveur FTP envoie un message de bienvenue indiquant qu'il est prêt à accepter des connexions. La réponse contient le message "220 ProFTPD Server (FTP\_by\_Rija)".

**Ligne 14 :**

* Source : 192.168.197.101 (Client FTP).
* Destination : 192.168.197.3 (Serveur FTP).
* Protocole : FTP.
* Détails : Le client FTP envoie une commande "USER" avec le nom d'utilisateur "rija". C'est une demande pour commencer la procédure de connexion avec ce nom d'utilisateur.

**Ligne 16 :**

* Source : 192.168.197.3 (Serveur FTP).
* Destination : 192.168.197.101 (Client FTP).
* Protocole : FTP.
* Détails : Le serveur FTP répond avec un code "331" demandant le mot de passe pour l'utilisateur "rija". Le message est "331 Mot de passe requis pour rija".

**Ligne 18 :**

* Source : 192.168.197.101 (Client FTP).
* Destination : 192.168.197.3 (Serveur FTP).
* Protocole : FTP.
* Détails : Le client FTP envoie la commande "PASS" avec le mot de passe correspondant à l'utilisateur "rija". (Le mot de passe réel n'est pas visible ici pour des raisons de sécurité).

**Ligne 20 :**

* Source : 192.168.197.3 (Serveur FTP).
* Destination : 192.168.197.101 (Client FTP).
* Protocole : FTP.
* Détails : Le serveur FTP répond avec un code "230" indiquant que l'utilisateur "rija" a été authentifié avec succès. Le message est "230 Utilisateur rija authentifié".

**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 ?**

**Observation des Échanges FTP sans TLS:**

1. **Transfert en Clair :**
   * Les échanges FTP sans TLS (FTP non sécurisé) transmettent toutes les données, y compris les informations de connexion (nom d'utilisateur et mot de passe), en texte clair. Cela signifie que toute personne capable d'intercepter le trafic réseau peut lire ces informations sans effort particulier.
2. **Exemples de Paquets Sensibles :**

* Les paquets contenant les commandes "USER" et "PASS" sont particulièrement sensibles car ils incluent le nom d'utilisateur et le mot de passe en clair.
* Par exemple :
  + USER rija : Cette commande montre le nom d'utilisateur utilisé pour la connexion.
  + PASS rija : Cette commande montre le mot de passe utilisé, s'il était visible dans le paquet.

### **Sécurité et Risques**

* **Risque d'Interception** :
  + Un attaquant ayant accès au réseau (par exemple, via une attaque de type Man-in-the-Middle) peut facilement capturer ces paquets et récupérer les noms d'utilisateur et mots de passe.
  + Cela peut être fait en utilisant des outils de capture de paquets comme Wireshark

**Comparaison avec les Échanges SSL/TLS**

1. **Chiffrement des Données :**

* Contrairement aux échanges FTP non sécurisés, les échanges via FTP sécurisé (FTPS) ou toute autre forme de SSL/TLS chiffrent les données transmises entre le client et le serveur.
* Cela signifie que même si un attaquant intercepte les paquets, il ne pourra pas lire les informations contenues dans les paquets sans déchiffrer le contenu.

1. **Paquets Sécurisés :**

* Lors de l'utilisation de FTPS ou HTTPS, les informations de connexion, les commandes et les données sont toutes chiffrées.
* Les paquets capturés avec Wireshark ou un autre outil de capture de paquets apparaîtront comme du texte chiffré et non lisible directement.

1. **Sécurité des Informations Sensibles :**

* Avec SSL/TLS, les informations de connexion (nom d'utilisateur et mot de passe) sont protégées par un chiffrement fort, rendant leur interception et utilisation par un attaquant beaucoup plus difficile sans accès aux clés de déchiffrement.

### **Conclusion**

* **FTP sans TLS** : Les échanges FTP sans TLS ne sont pas sécurisés. Il est possible de récupérer des données sensibles de connexion telles que les noms d'utilisateur et mots de passe simplement en interceptant les paquets réseau.
* **SSL/TLS** : Les échanges utilisant SSL/TLS sont sécurisés. Les données sont chiffrées, ce qui empêche l'interception et la lecture directe des informations de connexion et des données échangées.

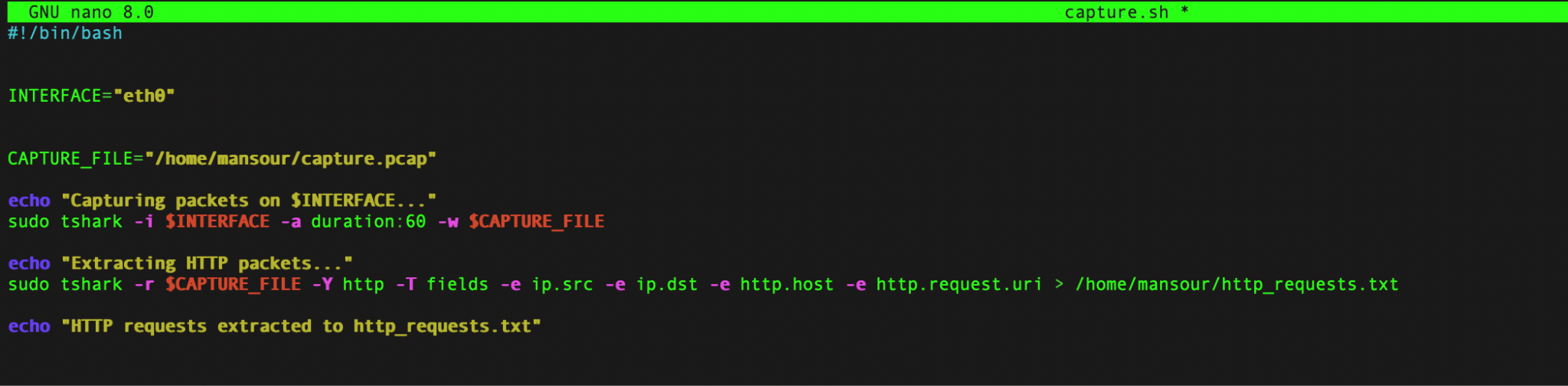
**Partie 3**

Il existe deux façons de filtrer

1 - le filtre de capture : Les filtres de capture sont utilisés pour déterminer quels paquets seront réellement capturés par l'outil de capture comme tshark ou tcpdump. Ils sont appliqués au niveau du noyau, ce qui signifie que seuls les paquets qui passent ce filtre sont enregistrés dans le fichier de capture ou affichés à l'écran.

2 - le filtre d’affichage : les filtres d’affichage sont utilisés après la capture des paquets pour afficher uniquement les paquets qui corresponde aux critères spécifiés. Appliqués au niveau de l’application ce qui signifie donc que tous les paquets sont capturés, mais seuls ceux qui corresponde au critères du filtre d’affichage sont montrées à l'utilisateur.

1. **Utiliser des scripts pour écouter le réseau et extraire les données.**



**Explication du script:**

INTERFACE contient mon interface réseau eth0

CAPTURE\_FILE est le fichier ou dans lequel les paquets capturés seront enregistrés -> capture.pcap

sudo tshark -i $INTERFACE -a duration:60 -w $CAPTURE\_FILE c’est cette commande qui va nous permettre de capturer les paquets sur l’interface réseau eth0 pendant 60 secondes et les enregistré dans le fichier capture\_file

sudo tshark -r $CAPTURE\_FILE -Y http -T fields -e ip.src -e ip.dst -e http.host -e http.request.uri > /home/mansour/http\_request.txt

extraction des paquets http, adresse ip sources/destination, l'hôte http et l’uri de la requête http

et redirection des résultats dans le fichier de http\_request.txt

Installez le paquet nécessaire : **sudo apt install tshark**

sudo tshark -i eth0 -Y http

sudo tshark -i eth0 -Y dns

sudo tshark -i eth0 -Y ssh

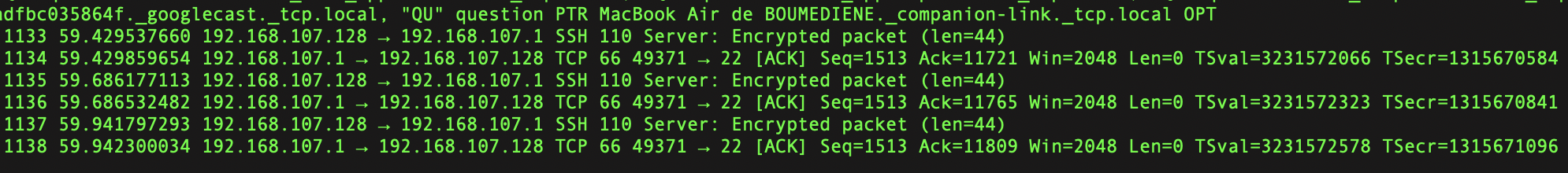
sudo tshark -i eth0 -Y ftp

sudo tshark -i eth0 -Y smtp

sudo tshark -i eth0 -Y icmp

La commande qui permets d’écouter et capturer les paquets de quelques-uns des protocoles

**sudo tshark -r /home/mansour/capture.pcap**



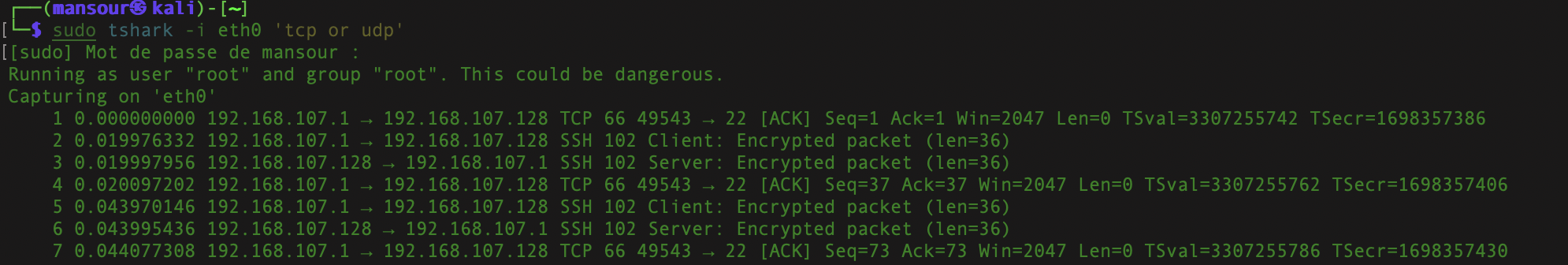
***Filtrage avec des options de tshark***

1. filtrage avec des Berkeley Packet Filter, sont des expressions utilisées pour filtrer les paquets réseau au niveau du noyau avant qu’ils ne soient capturé par l’outils tshark (par exemple)

**Pourquoi l'utiliser ?**

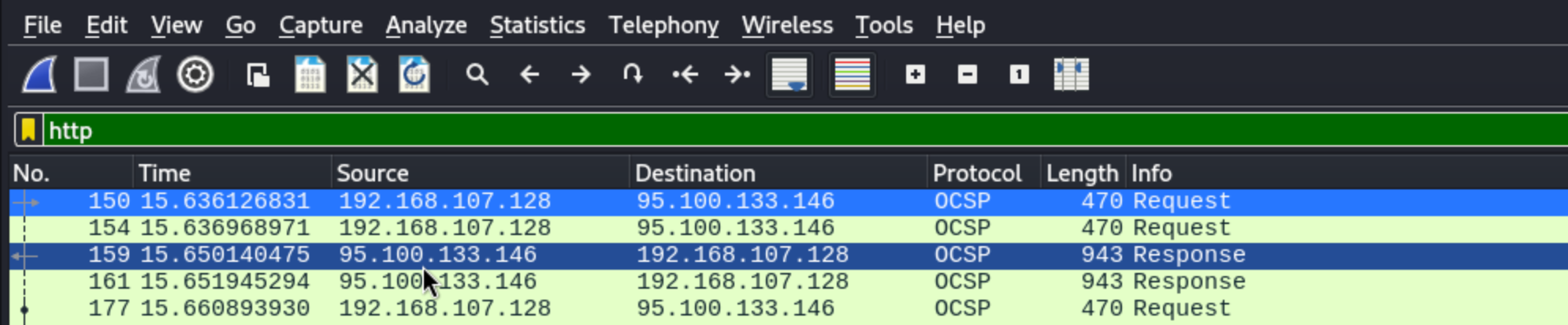
En filtrant les paquets dès le niveau du noyau, nous réduisons la charge sur le système et nous limitons la quantité de données traitées par tshark.

Nous obtenons uniquement les paquets qui nous intéressent .



**Filtrage avec des Display Filters**

Le “Display Filters” sélectionne et affiche uniquement les paquets qui répondent aux critères spécifiés après la capture, il permet de mettre en place des filtres complexes en combinant plusieurs critères (adresse ip, port, protocole) mais aussi il simplifie l’analyse des captures en réduisant la quantité de données à examiner .



#### **Filtrer en Redirigeant le Résultat**

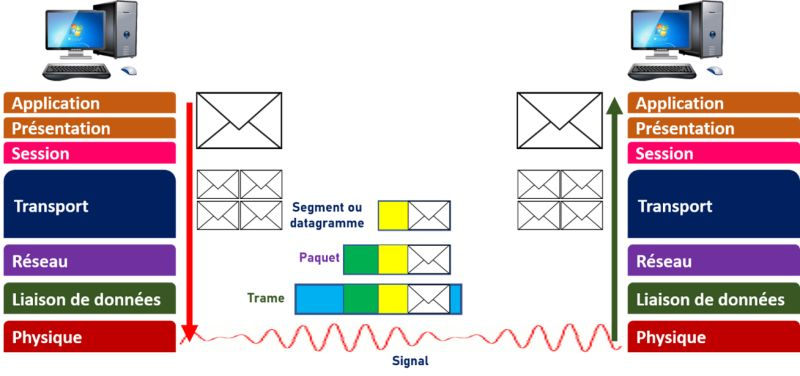
#### Vous pouvez également filtrer les captures en redirigeant le résultat vers un fichier ou un autre processus. Par exemple pour extraire les adresses IP source et destination des paquets HTTP de la capture et les enregistrer dans un fichier texte, vous pouvez utiliser la commande suivante :

sudo tshark -r capture.pcap -Y "http" -T fields -e ip.src -e ip.dst > http\_addresses.txt

Cela extrait les adresses IP source et destination des paquets HTTP de la capture capture.pcap et les enregistre dans le fichier http\_addresses.txt.

**LEXIQUE**

***MODÈLE OSI et TCP/IP*:**

****

**Rôles des couches du modèle OSI** :

➞Couche 7 : Application

Interface entre le réseau humain et le réseau de données

----------------------------------

➞Couche 6 : Présentation

Représentation commune des données :

• Codage / Décodage

• Chiffrement / Déchiffrement

• Compression / Décompression

----------------------------------

➞Couche 5 : Session

Synchronisation de dialogue et gestion de sessions :

• Ouverture d’une session

• Maintien d’une session

• Fermeture d’une session

----------------------------------

➞Couche 4 : Transport

• Segmentation / Reconstitution des données

• Gestion des communications de bout en bout entre les différents services et applications

----------------------------------

➞Couche 3 : Réseau

Gestion des communications entre les périphériques réseau :

• Adressage

• Routage

----------------------------------

➞Couche 2 : Liaison de données

• Méthodes d’accès aux supports

• Communication dans un réseau local via un commutateur

----------------------------------

➞Couche 1 : Physique

Supports, équipements réseau, connecteurs, prises, types de signaux, etc.

**L’encapsulation** est effectuée au niveau de l’émetteur à partir de la couche transport : Segment/Datagramme ➞ Paquet ➞ Trame ➞ Bits

**Le décapsulage** est effectué au niveau du récepteur : Bits ➞ Trame ➞Paquet ➞ Segment/Datagramme

**Couches du modèle TCP/IP :**

Dans le modèle TCP/IP, il existe 4 couches :

• Application

• Transport

• Internet

• Accès réseau

***DATAGRAMME:***

Un datagramme est un [paquet de données](https://fr.wikipedia.org/wiki/Paquet_(r%C3%A9seau)) transmis avec ses adresses de source et de destination par un [réseau de télécommunications](https://fr.wikipedia.org/wiki/R%C3%A9seau_de_t%C3%A9l%C3%A9communications) ([WAN](https://fr.wikipedia.org/wiki/R%C3%A9seau_%C3%A9tendu)) ou un réseau local ([LAN](https://fr.wikipedia.org/wiki/R%C3%A9seau_local)). Un service de datagramme est la variante [sans connexion](https://fr.wikipedia.org/wiki/Transmission_en_mode_sans_connexion) des services de transmission utilisant la [commutation de paquets](https://fr.wikipedia.org/wiki/Commutation_de_paquets) (l'autre variante, [avec connexion](https://fr.wikipedia.org/wiki/Mode_connect%C3%A9), étant un service de [circuit virtuel](https://fr.wikipedia.org/wiki/Circuit_virtuel)). Exemples de services datagramme : [Ethernet](https://fr.wikipedia.org/wiki/Ethernet), [Internet Protocol](https://fr.wikipedia.org/wiki/Internet_Protocol) (IP).

***ENCAPSULATION ET DECAPSULATION***

Encapsulation et décapsulage sont des processus clés dans les communications réseau, assurant le transfert de données entre appareils en passant par les différentes couches du modèle OSI.

**Encapsulation**

L'encapsulation consiste à ajouter des informations de contrôle aux données à chaque couche du modèle OSI, de la couche application à la couche physique :

1. Application : Les données (ex. email) sont générées.

2. Transport : Les données sont divisées en segments ou datagrammes et les numéros de port sont ajoutés.

3. Réseau : Les segments sont encapsulés dans des paquets avec des adresses IP source et destination.

4. Liaison de Données : Les paquets sont encapsulés dans des trames avec des adresses MAC.

5. Physique : Les trames sont converties en bits pour transmission sur le support physique (câble, fibre optique, ondes radio).

**Décapsulage**

Le décapsulage est le processus inverse où chaque couche du récepteur retire les informations de contrôle pour reconstituer les données originales :

1. Physique: Les bits sont convertis en trames.

2. Liaison de Donnée : Les trames sont transformées en paquets en retirant les en-têtes/pieds.

3. Réseau : Les paquets sont transformés en segments en retirant les en-têtes.

4. Transport : Les segments sont reconstitués en données de l'application.

5. Application: Les données originales sont remises à l'application destinataire (ex. affichage d'un email).

Exemple

- Encapsulation: Un email envoyé est encapsulé en segments, paquets, trames, puis bits pour transmission.

- Décapsulage : L'email reçu est décapsulé en bits, trames, paquets, segments, puis remis à l'application de messagerie.

En résumé, l'encapsulation et le décapsulage structurent et organisent les données pour un transfert fiable et efficace à travers un réseau, chaque couche ajoutant ou retirant des informations spécifiques à ses fonctions.

***PAQUETS ARP***

Paquets ARP (Address Resolution Protocol)

- Fonction : Trouver l'adresse MAC d'une machine à partir de son adresse IP dans un réseau local.

- Processus :

- ARP Request : Demande l'adresse MAC associée à une adresse IP.

- ARP Reply : Répond avec l'adresse MAC correspondante.

- Utilité : Essentiel pour la communication dans un réseau local.

***PAQUETS UDP***

Paquets UDP (User Datagram Protocol)

- Fonction : Protocole de transport rapide pour envoyer des messages courts (datagrammes).

- Caractéristiques :

- Non-fiable : Pas de garantie de livraison, d'ordre ou d'absence de duplicata.

- Sans connexion : Pas de session établie entre l'expéditeur et le récepteur.

- Léger : Moins de surcharge de traitement.

- Utilité : Idéal pour le streaming vidéo, les jeux en ligne, et d'autres applications où la vitesse est plus importante que la fiabilité.

***PAQUETS TCP***

Paquets TCP (Transmission Control Protocol)

- Fonction : Protocole de transport fiable qui garantit la livraison, l'ordre et l'absence de duplicata des paquets.

- Caractéristiques :

- Fiable : Assure la réception correcte et dans l'ordre des paquets.

- \*Orienté connexion : Établit une connexion avant de transférer des données.

- Contrôle de flux et de congestion : Gère le débit pour éviter la congestion du réseau.

- Utilité : Crucial pour les applications nécessitant une haute fiabilité comme les e-mails, la navigation web, et les transferts de fichiers.

***mDNS***

Le **Multicast DNS** est un protocole réseau qui permet l**a résolution de noms de domaine dans des réseaux locaux sans nécessiter de serveur DNS spécifique.**

Il fonctionne en utilisant des **requêtes multicast pour diffuser les demandes de résolution de noms de domaine sur le réseau local.** Les appareils du réseau répondent directement à ces requêtes avec les informations nécessaires. mDNS est largement utilisé pour la **découverte automatique** des services sur un réseau local, simplifiant ainsi la **configuration réseau** et facilitant l'accès aux appareils comme les imprimantes, les **serveurs multimédia**, etc. Des technologies telles **qu'Apple Bonjour** reposent sur mDNS pour la découverte automatique des appareils et des services. En résumé, mDNS simplifie la **connectivité** et l'accès aux services dans les **réseaux locaux.**