# Network Sniffing

Wireshark est un analyseur de protocoles réseau largement utilisé. Il permet de capturer et d'afficher le trafic réseau en temps réel, ce qui facilite le dépannage, l'analyse, le développement de logiciels et l'enseignement des protocoles réseau. Wireshark est un logiciel open-source et il est disponible pour de nombreuses plateformes, dont Windows, macOS, Linux et BSD.

## Différence entre une trame et un paquet :

Dans le contexte de la communication réseau, les données sont souvent encapsulées dans des structures de données spéciales pour la transmission. Une trame et un paquet sont deux de ces structures de données, mais elles sont utilisées à des niveaux différents de la pile de protocoles réseau.

Une trame est la structure de données utilisée au niveau de la couche liaison de données (couche 2 du modèle OSI). Elle contient non seulement les données à transmettre, mais aussi des informations de contrôle et d'adressage nécessaires à la transmission au niveau de la couche liaison de données. Les trames sont utilisées pour la transmission de données sur un réseau local (LAN), où les périphériques réseau sont directement connectés les uns aux autres.

Un paquet, en revanche, est la structure de données utilisée au niveau de la couche réseau (couche 3 du modèle OSI). Comme une trame, un paquet contient les données à transmettre ainsi que des informations de contrôle et d'adressage.

Cependant, les informations d'adressage dans un paquet sont conçues pour la transmission de données sur de longues distances, éventuellement via plusieurs réseaux, plutôt que sur un simple réseau local. Les paquets sont utilisés pour la transmission de données sur des réseaux étendus (WAN), y compris Internet.

En résumé, les trames et les paquets sont des structures de données utilisées pour la transmission de données sur un réseau, mais les trames sont utilisées au niveau de la couche liaison de données pour la transmission sur un réseau local, tandis que les paquets sont utilisés au niveau de la couche réseau pour la transmission sur de longues distances.

# Format pcap/pcapng :

Le format pcap (Packet Capture) est un format de fichier standard pour la sauvegarde des données de capture de paquets réseau. Il a été introduit pour la première fois dans le programme de capture de paquets UNIX tcpdump, et est maintenant utilisé par de nombreux autres programmes de capture de paquets, y compris Wireshark.

Le format pcap est conçu pour être une représentation fidèle des paquets réseau tels qu'ils ont été capturés, avec un minimum d'interprétation ou de modification.

| | MAC sources | IP sources | IP destinations |
|---|---|---|---|
| TCP | d0:c0:bf:42:01:e6<br>d0:c0:bf:4d:fa:2b | 10.10.0.86<br>10.10.0.88 | 10.10.5.129 -14:13:33:39:5a:c3-<br>-d0:c0:bf:42:01:e6- |
| UDP | 14:13:33:39:5a:c3<br>a0:a3:f0:b3:bb:80 | 10.10.5.129<br>142.251.37.234 | 142.251.37.234<br>-68:05:ca:3a:2e:49-<br><br>14:13:33:39:5a:c3 |
| ARP | 24:5e:be:7e:1f:2b<br>10:68:38:09:16:62 | QNAP_7e:1f:2b<br>AzureWaveTec_09:16:62 | 10.10.3.203 - 10.10.0.200 -<br>-00:00:00_00:00:00-<br>-24:5e:be:7e:1f:2b-<br><br>AzureWaveTec_06:5a:c3<br><br>-14:13:33:39:5a:c3- |

Sur notre capture réseau nous retrouvons comme protocole le SSDP, le MDNS, le QUIC...

Simple Service Discovery Protocol (SSDP) est un protocole réseau basé sur la suite de protocoles Internet pour la diffusion et la découverte de services de réseau et d'informations de présence. Il accomplit cette tâche sans l'aide de mécanismes de configuration basés sur le serveur, tels que le protocole DHCP (Dynamic Host Configuration Protocol) ou le DNS (Domain Name System), et sans configuration statique particulière d'un hôte de réseau.

Le protocole mDNS, ou multicast Domain Name System est un protocole qui nous permet de bénéficier des fonctionnalités de DNS, sans avoir un serveur DNS sur le réseau.

QUIC est un protocole de transport fiable et sécurisé, en mode connecté, mis au point par Jim Roskind chez Google. Il est repris par l'IETF en 2015 dans le but de le normaliser par les RFC 8999, 9000, 9001 et 9002 et le rendre ainsi utilisable par n'importe quel protocole de la couche application.

En examinant les champs spécifiques de chaque protocole, nous avons pu décoder et interpréter les informations contenues dans les trames réseau.

- **ARP** : Utilisé pour lier les adresses IP aux adresses MAC, le format ARP comprend des champs pour les types de matériel et de protocole, les adresses MAC et IP de l'expéditeur et du destinataire.

- **UDP** : Un protocole de transport simple et rapide, l'en-tête UDP contient les ports source et destination, la longueur totale et une somme de contrôle facultative.

- **TCP** : Un protocole de transport fiable et complexe, l'en-tête TCP inclut des informations détaillées telles que les numéros de séquence et d'acquittement, des indicateurs de contrôle, la taille de la fenêtre et des options.

La correspondance des captures en hexadécimal avec ces formats nous permet de comprendre la structure et le contenu des communications réseau. Cette analyse est essentielle pour le diagnostic réseau, la sécurité informatique et l'optimisation des performances réseau.

| UDP | TCP | ARP |
|---|---|---|
| 0000 14 13 33 39 5a c3 a0 a3 f0 b3 bb 80 08 00 45 00 ..39Z.....E.<br>0010 00 3c 00 00 40 00 3c 11 7a 41 8e fb 25 ea 0a 0a .<..@.<.zA..%...<br>0020 05 81 01 bb ed d9 00 28 45 75 49 0f b3 a7 e6 9f .....(EuI.....<br>0030 1e 9c 4f f1 2b c6 8b 4e 69 37 01 1a 39 a7 f7 38 ..O.+..Ni7..9..8<br>0040 0c db 02 32 60 30 91 c4 60 f7 ...2`0..`. | 0000 d0 c0 bf 42 02 fa 14 13 33 39 5a c3 08 00 45 00 ...B....39Z...E.<br>0010 00 9d 4e 6c 40 00 80 06 92 03 0a 0a 05 81 0a 0a ..Nl@.....<br>0020 00 57 d2 74 1f 49 98 8f da f3 db af 61 da 50 18 .W.t.I.....a.P.<br>0030 02 00 8a fa 00 00 17 03 03 00 70 00 00 00 00 00 .....p.....<br>0040 00 00 6a 92 e3 06 c9 22 30 b8 63 4a c6 22 b1 9c ..j....."0.cJ."..<br>0050 b1 0d 07 f3 1c 5f d0 d2 44 27 b5 ef 1f 9d 23 1d ....._..D'....#.<br>0060 38 97 66 1b ef 7b 46 a5 9f f2 d7 00 5f 58 ff a8 8.f..{F....._X..<br>0070 f9 61 fe 67 e6 89 ed d9 00 ee 69 eb c3 32 91 04 .a.g.....i..2..<br>0080 ef 34 cf 4c b3 79 23 94 26 51 11 ff 6c a1 cf 7a .4.L.y#.&Q..l..z<br>0090 9b 10 a0 10 d5 c4 1b ae 52 7b ab 95 85 60 78 7e ........R{...`x~<br>00a0 e6 54 7d ad 36 a8 76 e8 e2 05 7a .T}.6.v...z | 0000 10 68 38 09 16 62 14 13 33 39 5a c3 08 06 00 01 .h8..b..39Z.....<br>0010 08 00 06 04 00 01 14 13 33 39 5a c3 0a 0a 05 81 ........39Z.....<br>0020 10 68 38 09 16 62 0a 0a 16 c6 .h8..b.... |

# Analyse des Paquets

Pour chaque paquet (UDP, TCP, ARP), nous allons décomposer les données hexadécimales pour identifier et analyser les différents champs conformément aux spécifications des formats de messages.

*Paquet UDP*

```
0000   14 13 33 39 5a c3 a0 a3 f0 b3 bb 80 08 00 45 00   ..39Z.........E.
0010   00 3c 00 00 40 00 3c 11 7a 41 8e fb 25 ea 0a 0a   .<..@.<.zA..%...
0020   05 81 01 bb ed d9 00 28 45 75 49 0f b3 a7 e6 9f   .......(EuI.....
0030   1e 9c 4f f1 2b c6 8b 4e 69 37 01 1a 39 a7 f7 38   ..O.+..Ni7..9..8
0040   0c db 02 32 60 30 91 c4 60 f7                     ...2`0..`.
```

### 1. *En-tête Ethernet*

Destination MAC: `14 13 33 39 5a c3`

Source MAC: `a0 a3 f0 b3 bb 80`

Type: `08 00` (IPv4)

### 2. *En-tête IP*

Version & Header Length: `45`

Type of Service: `00`

Total Length: `00 3c` (60 octets)

Identification: `00 00`

Flags & Fragment Offset: `40 00`

Time to Live: `3c`

Protocol: `11` (UDP)

Header Checksum: `7a 41`

Source IP: `8e fb 25 ea` (142.251.37.234)

Destination IP: `0a 0a 05 81` (10.10.5.129)

### 3. *En-tête UDP*

Source Port: `01 bb` (443)

Destination Port: `ed d9` (60889)

Length: `00 28` (40 octets)

Checksum: `45 75`

### 4. *Données UDP*

```
49 0f b3 a7 e6 9f 1e 9c 4f f1 2b c6 8b 4e 69 37 01 1a 39 a7 f7 38 0c db 02 32 60
30 91 c4 60 f7
```

*Paquet TCP*

```
0000   d0 c0 bf 42 02 fa 14 13 33 39 5a c3 08 00 45 00   ...B....39Z...E.
0010   00 9d 4e 6c 40 00 80 06 92 03 0a 0a 05 81 0a 0a   ..Nl@...........
0020   00 57 d2 74 1f 49 98 8f da f3 db af 61 da 50 18   .W.t.I......a.P.
0030   02 00 8a fa 00 00 17 03 03 00 70 00 00 00 00 00   ..........p.....
0040   00 00 6a 92 e3 06 c9 22 30 b8 63 4a c6 22 b1 9c   ..j...."0.cJ."..
0050   b1 0d 07 f3 1c 5f d0 d2 44 27 b5 ef 1f 9d 23 1d   ....._..D'....#.
0060   38 97 66 1b ef 7b 46 a5 9f f2 d7 00 5f 58 ff a8   8.f..{F....._X..
0070   f9 61 fe 67 e6 89 ed d9 00 ee 69 eb c3 32 91 04   .a.g......i..2..
0080   ef 34 cf 4c b3 79 23 94 26 51 11 ff 6c a1 cf 7a   .4.L.y#.&Q..l..z
0090   9b 10 a0 10 d5 c4 1b ae 52 7b ab 95 85 60 78 7e   ........R{...`x~
00a0   e6 54 7d ad 36 a8 76 e8 e2 05 7a                  .T}.6.v...z
```

### 1. *En-tête Ethernet*

Destination MAC: `d0 c0 bf 42 02 fa`

Source MAC: `14 13 33 39 5a c3`

Type: `08 00` (IPv4)

### 2. *En-tête IP*

Version & Header Length: `45`

Type of Service: `00`

Total Length: `00 9d` (157 octets)

Identification: `4e 6c`

Flags & Fragment Offset: `40 00`

Time to Live: `80`

Protocol: `06` (TCP)

Header Checksum: `92 03`

Source IP: `0a 0a 05 81` (10.10.5.129)

Destination IP: `0a 0a 00 57` (10.10.0.87)

### 3. *En-tête TCP*

Source Port: d2 74 (53876)

Destination Port: 1f 49 (8009)

Sequence Number: 98 8f da f3

Acknowledgment Number: db af 61 da

Data Offset & Reserved: 50 (Header length = 20 octets)

Flags: 18 (ACK, PSH)

Window Size: 02 00 (512)

Checksum: 8a fa

Urgent Pointer: 00 00

### *4. Données TCP*

```
17 03 03 00 70 00 00 00 00 00 00 00 6a 92 e3 06 c9 22 30 b8 63 4a c6 22 b1 9c b1
0d 07 f3 1c 5f d0 d2 44 27 b5 ef 1f 9d 23 1d 38 97 66 1b ef 7b 46 a5 9f f2 d7 00
5f 58 ff a8 f9 61 fe 67 e6 89 ed d9 00 ee 69 eb c3 32 91 04 ef 34 cf 4c b3 79 23
94 26 51 11 ff 6c a1 cf 7a 9b 10 a0 10 d5 c4 1b ae 52 7b ab 95 85 60 78 7e e6 54
7d ad 36 a8 76 e8 e2 05 7a
```

*Paquet ARP*

```
0000   10 68 38 09 16 62 14 13 33 39 5a c3 08 06 00 01   .h8..b..39Z.....
0010   08 00 06 04 00 01 14 13 33 39 5a c3 0a 0a 05 81   ........39Z.....
0020   10 68 38 09 16 62 0a 0a 16 c6                     .h8..b....
```

**1. *En-tête Ethernet***

Destination MAC: `10 68 38 09 16 62`

Source MAC: `14 13 33 39 5a c3`

Type: `08 06` (ARP)

**2. *En-tête ARP***

Hardware Type: `00 01` (Ethernet)

Protocol Type: `08 00` (IPv4)

Hardware Size: `06`

Protocol Size: `04`

Opcode: `00 01` (Request)

Sender MAC Address: `14 13 33 39 5a c3`

Sender IP Address: `0a 0a 05 81` (10.10.5.129)

Target MAC Address: `10 68 38 09 16 62`

Target IP Address: `0a 0a 16 c6` (10.10.22.198)

---

Pour identifier les paquets TCP correspondant aux différentes étapes de connexion entre un hôte et un serveur, nous devons rechercher les paquets avec des drapeaux spécifiques définissant l'établissement et la terminaison de la connexion TCP. Les étapes clés sont les suivantes :

1. **SYN** : Le client envoie un paquet SYN pour initier une connexion.

2. **SYN-ACK** : Le serveur répond avec un paquet SYN-ACK pour accepter la connexion.

3. **ACK** : Le client envoie un paquet ACK pour finaliser l'établissement de la connexion.

4. **FIN** : Une partie (client ou serveur) envoie un paquet FIN pour terminer la connexion.

5. **FIN-ACK** : L'autre partie répond avec un paquet FIN-ACK pour accepter la terminaison de la connexion.

6. **ACK** : La partie initiale répond avec un paquet ACK pour finaliser la terminaison de la connexion.

Analysons les paquets TCP fournis pour identifier ces étapes.

## *Paquet TCP donné*

```
0000   d0 c0 bf 42 02 fa 14 13 33 39 5a c3 08 00 45 00   ...B....39Z...E.
0010   00 9d 4e 6c 40 00 80 06 92 03 0a 0a 05 81 0a 0a   ..Nl@...........
0020   00 57 d2 74 1f 49 98 8f da f3 db af 61 da 50 18   .W.t.I......a.P.
0030   02 00 8a fa 00 00 17 03 03 00 70 00 00 00 00 00   ..........p.....
0040   00 00 6a 92 e3 06 c9 22 30 b8 63 4a c6 22 b1 9c   ..j...."0.cJ."..
0050   b1 0d 07 f3 1c 5f d0 d2 44 27 b5 ef 1f 9d 23 1d   ....._..D'....#.
0060   38 97 66 1b ef 7b 46 a5 9f f2 d7 00 5f 58 ff a8   8.f..{F....._X..
0070   f9 61 fe 67 e6 89 ed d9 00 ee 69 eb c3 32 91 04   .a.g......i..2..
0080   ef 34 cf 4c b3 79 23 94 26 51 11 ff 6c a1 cf 7a   .4.L.y#.&Q..l..z
0090   9b 10 a0 10 d5 c4 1b ae 52 7b ab 95 85 60 78 7e   ........R{...`x~
00a0   e6 54 7d ad 36 a8 76 e8 e2 05 7a                  .T}.6.v...z
```

## *Décomposition du paquet TCP*

### 1. *En-tête Ethernet*

- Destination MAC: `d0 c0 bf 42 02 fa`
- Source MAC: `14 13 33 39 5a c3`
- Type: `08 00` (IPv4)

### 2. *En-tête IP*

- Version & Header Length: `45`
- Type of Service: `00`
- Total Length: `00 9d` (157 octets)
- Identification: `4e 6c`
- Flags & Fragment Offset: `40 00`
- Time to Live: `80`
- Protocol: `06` (TCP)
- Header Checksum: `92 03`
- Source IP: `0a 0a 05 81` (10.10.5.129)
- Destination IP: `0a 0a 00 57` (10.10.0.87)

### 3. *En-tête TCP*

- Source Port: d2 74 (53876)
- Destination Port: 1f 49 (8009)
- Sequence Number: 98 8f da f3
- Acknowledgment Number: db af 61 da
- Data Offset & Reserved: 50 (Header length = 20 octets)
- Flags: 18 (ACK, PSH)
- Window Size: 02 00 (512)
- Checksum: 8a fa
- Urgent Pointer: 00 00

## *Identification des Étapes*

Pour identifier les étapes de la connexion TCP, nous devons examiner les valeurs des drapeaux TCP :

- *SYN* : Flag 02

- *SYN-ACK* : Flags 12

- *ACK* : Flag 10

- *FIN* : Flag 01

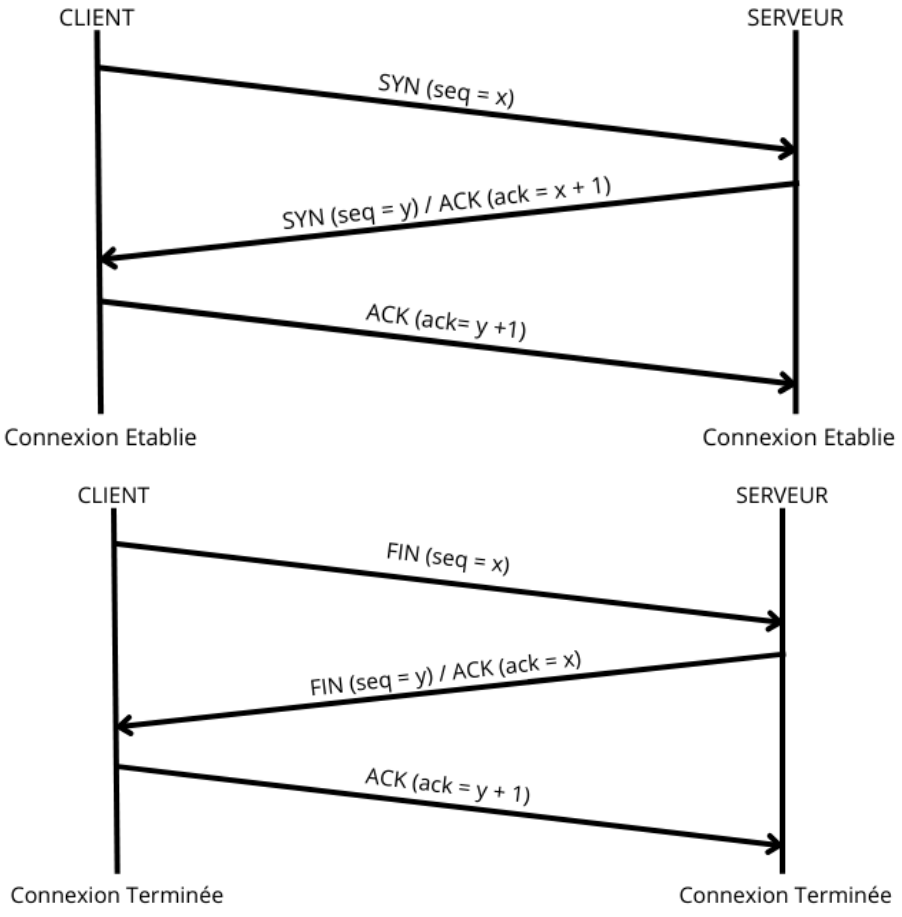
- *FIN-ACK* : Flags 11

Dans le paquet donné, les drapeaux 18 (ACK, PSH) indiquent que ce paquet n'est ni un SYN, ni un FIN, mais un paquet de données intermédiaire avec une demande de poussée des données.

La connexion TCP est établie en utilisant un processus en trois étapes connu sous le nom de "handshake" à trois voies (three-way handshake).

D

22



Pour utiliser les filtres Wireshark, il faut d'abord comprendre la syntaxe de base des filtres. Les filtres Wireshark utilisent des opérateurs logiques et de comparaison pour définir les critères de filtrage.

Voici quelques exemples de filtres Wireshark :

ip.src == 192.168.1.100 : affiche uniquement les trames dont l'adresse IP source est 192.168.1.100.

ip.dst == 192.168.1.200 : affiche uniquement les trames dont l'adresse IP de destination est 192.168.1.200.

tcp.port == 80 : affiche uniquement les trames TCP qui utilisent le port 80 (généralement le trafic HTTP).

udp.port == 53 : affiche uniquement les trames UDP qui utilisent le port 53 (généralement le trafic DNS).

http.request.method == "GET" : affiche uniquement les requêtes HTTP qui utilisent la méthode GET.

Pour appliquer un filtre dans Wireshark, il faut saisir le filtre dans la barre de filtre en haut de l'écran et appuyer sur la touche Entrée. Il est également possible d'utiliser les boutons de filtre prédéfinis pour afficher rapidement les trames d'un type spécifique (par exemple, les trames DNS ou HTTP).

Il est important de noter que les filtres Wireshark sont appliqués après la capture des trames, ce qui signifie que toutes les trames sont capturées et stockées dans la mémoire tampon de Wireshark avant d'être filtrées. Cela peut entraîner une utilisation importante de la mémoire et du processeur si un grand nombre de trames est capturé.

Pour éviter ce problème, il existe les filtres de capture Wireshark, qui sont appliqués avant la capture des trames. Les filtres de capture Wireshark utilisent une syntaxe légèrement différente de celle des filtres de display, mais ils fonctionnent de la même manière.

Pour utiliser un filtre de capture dans Wireshark, cliquer sur le bouton "Capture Options" dans la barre d'outils, puis saisir le filtre de capture dans la zone "Capture Filter". Il est également possible d'utiliser les boutons de filtre de capture prédéfinis pour capturer rapidement les trames d'un type spécifique.

# Partie 2 : Analyse du réseau local

En écoutant le trafic entre les échanges FTP sans TLS, nous remarquons que les données sensibles ainsi que les intitulés de fichiers sont visibles sur le réseau.

Lors de la connexion de l'utilisateur, son nom d'utilisateur et son mot de passe sont clairement identifiables dans les trames de données capturées par Wireshark.

Pendant un transfert de fichier du client vers le serveur (commande PUT en FTP), nous observons que l'intitulé du nom de fichier ainsi que son extension sont également visibles dans les trames de données.

Cependant, lorsque le protocole TLS (SSL) est utilisé, toutes les données transitant sur le réseau sont cryptées, assurant ainsi une protection adéquate de la confidentialité des informations échangées.

# Partie 3 : TShark et Captures de Paquets Réseau

## *Installation de TShark*

Pour capturer des paquets réseau avec TShark, on commence par installer le paquet approprié.

```
sudo apt-get update
sudo apt-get install tshark
```

## *Capture de Paquets pour Différents Protocoles*

TShark est la version en ligne de commande de Wireshark. Il permet de capturer et d'analyser les paquets réseau directement depuis le terminal. Voici comment utiliser TShark pour capturer des paquets pour différents protocoles :

*Message (HTTP, FTP, SMB) :*

```
sudo tshark -i eth0 -f "tcp port 80 or tcp port 21 or tcp port 445" -w
capture_http_ftp_smb.pcap
```

*DHCP :*

```
sudo tshark -i eth0 -f "udp port 67 or udp port 68" -w capture_dhcp.pcap
```

*DNS :*

```
sudo tshark -i eth0 -f "udp port 53" -w capture_dns.pcap
```

*mDNS :*

```
sudo tshark -i eth0 -f "udp port 5353" -w capture_mdns.pcap
```

*SSL/TLS/HTTPS :*

```
sudo tshark -i eth0 -f "tcp port 443" -w capture_https.pcap
```

**Options Utilisées :**

- *sudo*: Exécuter la commande avec des privilèges super-utilisateur, nécessaires pour capturer les paquets réseau.

- *tshark*: Commande pour lancer TShark.

- -i eth0: Spécifie l'interface réseau eth0 à surveiller. Remplacez eth0 par l'interface réseau appropriée pour votre configuration.

- *-f* "filter": Applique un filtre BPF (Berkeley Packet Filter) pour capturer uniquement les paquets correspondant aux critères spécifiés.

  - *tcp port 80*: Capture les paquets HTTP.

  - *tcp port 24*: Capture les paquets FTP.

  - *tcp port 445*: Capture les paquets SMB.

  - *udp port 67 ou udp port 68*: Capture les paquets DHCP.

  - *udp port 53*: Capture les paquets DNS.

  - *udp port 5353*: Capture les paquets mDNS.

  - *tcp port 443*: Capture les paquets HTTPS/SSL/TLS.

- `*-w capture.pcap*`: Sauvegarde les paquets capturés dans un fichier au format `.pcap`.

### Capture et Analyse des Paquets

1. *On lance la capture* :

   - On exécute la commande TShark appropriée pour le protocole souhaité.

2. On *interagit avec les services* :

   - On effectue des actions réseau correspondantes aux services configurés (par exemple, accéder à une page web HTTPS, transférer un fichier via FTP, etc.).

3. *On arrête la capture* :

   - On appuie sur Ctrl+C dans le terminal pour arrêter la capture.

**Redirection des Résultats et Filtrage Post-Capture**

En plus des filtres directs, TShark permet de rediriger les résultats pour une analyse ou un filtrage ultérieur.

*Capture brute sans filtre et redirection pour filtrage ultérieur* :

```
sudo tshark -i eth0 -w capture_all.pcap
```

*Filtrage post-capture avec TShark* :

On utilise TShark pour lire le fichier de capture et appliquer un filtre :

```
tshark -r capture_all.pcap -Y "http" -w filtered_http.pcap
```

- Options :

  - *-r capture_all.pcap*: Lit le fichier de capture capture_all.pcap.

  - *-Y "http"*: Applique un filtre d'affichage pour ne capturer que les paquets HTTP.

- *-w filtered_http.pcap*: Sauvegarde les paquets filtrés dans un nouveau fichier.

TShark est un outil puissant pour la capture et l'analyse de paquets réseau en ligne de commande. En utilisant des options de filtrage appropriées, vous pouvez capturer spécifiquement les paquets des protocoles souhaités. La redirection et le filtrage post-capture offrent une flexibilité supplémentaire pour une analyse approfondie des données réseau.