

La différence entre une trame et un paquet réside principalement dans les couches du modèle OSI auxquelles ils appartiennent et dans les composants qu'ils contiennent :

Trame

- **Niveau OSI** : La trame opère au niveau de la couche liaison de données (couche 2).
- **Contenu** : Une trame comprend une en-tête de couche de liaison de données, les données utiles (payload), et une séquence de contrôle de trame (FCS - Frame Check Sequence) pour la détection d'erreurs. L'en-tête contient des informations telles que les adresses MAC source et destination, le type de protocole de la couche supérieure (par exemple, IP), etc.
- **Exemple** : Trames Ethernet, trames Wi-Fi.

Paquet

- **Niveau OSI** : Le paquet opère au niveau de la couche réseau (couche 3).
- **Contenu** : Un paquet comprend une en-tête de couche réseau, les données utiles (payload) qui peuvent inclure une trame de couche de liaison, et éventuellement une queue de paquet. L'en-tête contient des informations telles que les adresses IP source et destination, le protocole de la couche supérieure (par exemple, TCP ou UDP), le numéro de séquence, etc.
- **Exemple** : Paquets IP.

En résumé, une trame est un conteneur de données utilisé au niveau de la couche liaison de données, alors qu'un paquet est un conteneur de données utilisé au niveau de la couche réseau. Une trame peut encapsuler un paquet lorsque celui-ci est transmis sur un réseau.

Format pcap/pcapng

- **pcap (Packet Capture)** : C'est un format de fichier utilisé pour enregistrer les paquets de données capturés sur un réseau. Le format pcap est très utilisé par des outils comme Wireshark, tcpdump, et autres. Chaque enregistrement dans un fichier pcap contient une en-tête de paquet (timestamp, longueur du paquet, etc.) suivie par les données de paquet elles-mêmes (en-tête de la couche réseau et payload).

- **pcapng (pcap Next Generation)** : C'est une version améliorée du format pcap qui offre plus de flexibilité et de fonctionnalités. Il prend en charge plusieurs interfaces de capture dans un même fichier, des blocs d'options pour stocker des informations supplémentaires (comme des commentaires, des informations sur l'interface, des statistiques d'interface, etc.), et des types de blocs supplémentaires pour une extensibilité future.

En résumé, pcap et pcapng sont des formats de fichiers utilisés pour stocker des paquets de données capturés sur un réseau, avec pcapng offrant une structure plus avancée et flexible par rapport à pcap.

-partie 1

Adresses MAC et IP des paquets capturés

1. **Paquet ARP (Frame 2):**
 - **MAC source:** 74:97:79:65:c5:95 (CloudNetwork_65:c5:95)
 - **IP source:** 10.10.29.219
 - **MAC destination:** ff:ff:ff:ff:ff:ff (Broadcast)
 - **IP destination:** 10.10.14.33 (ARP cible)
2. **Paquet UDP (Frame 327):**
 - **MAC source:** 68:05:ca:3a:2e:49 (Intel_3a:2e:49)
 - **IP source:** 142.250.200.206
 - **MAC destination:** 20:0b:74:2c:9f (AzureWaveTec_2c:9f)
 - **IP destination:** 10.10.30.255

Spécifications des formats de message ARP/UDP

Format des messages ARP

- **Hardware type:** 2 bytes
- **Protocol type:** 2 bytes
- **Hardware size:** 1 byte
- **Protocol size:** 1 byte

- **Opcode:** 2 bytes
- **Sender MAC address:** 6 bytes
- **Sender IP address:** 4 bytes
- **Target MAC address:** 6 bytes
- **Target IP address:** 4 bytes

Format des messages UDP

- **Source port:** 2 bytes
- **Destination port:** 2 bytes
- **Length:** 2 bytes
- **Checksum:** 2 bytes
- **Data:** variable length

Pour analyser les étapes de connexion TCP entre notre hôte et un serveur, nous devons identifier les paquets SYN, SYN-ACK et ACK qui forment le processus de poignée de main à trois voies (three-way handshake). Voici les détails des trois paquets que vous avez fournis :

1. **Frame 46: TCP SYN Packet**
 - **Source Address:** 10.10.30.255
 - **Destination Address:** 20.189.173.7
 - **Source Port:** 58300
 - **Destination Port:** 443
 - **Flags:** SYN (0x002)
2. **Frame 47: TCP SYN-ACK Packet**
 - **Source Address:** 20.189.173.7
 - **Destination Address:** 10.10.30.255
 - **Source Port:** 443
 - **Destination Port:** 58300
 - **Flags:** SYN, ACK (0x012)
3. **Frame 48: TCP ACK Packet**
 - **Source Address:** 10.10.30.255
 - **Destination Address:** 20.189.173.7
 - **Source Port:** 58300
 - **Destination Port:** 443
 - **Flags:** ACK (0x010)

Mécanisme de Connexion TCP (Three-Way Handshake)

Le processus de connexion TCP se compose de trois étapes principales :

1. **Client envoie un paquet SYN :**
 - Le client (10.10.30.255) envoie un paquet avec le drapeau SYN au serveur (20.189.173.7) pour initier une connexion.
 - **Frame 46 :** Ce paquet est identifié avec le drapeau SYN.
2. **Serveur répond avec un paquet SYN-ACK :**

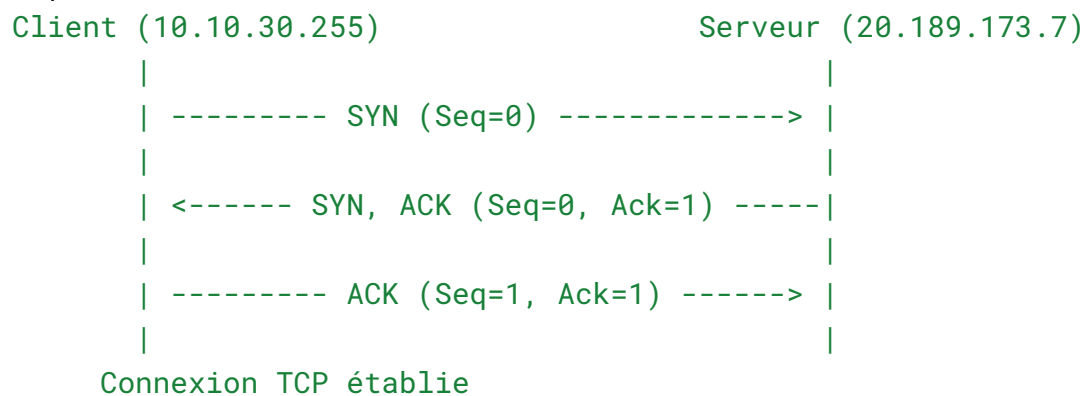
- Le serveur (20.189.173.7) répond au client avec un paquet ayant les drapeaux SYN et ACK pour reconnaître la réception de la demande de connexion et indiquer qu'il est prêt à établir la connexion.
 - **Frame 47** : Ce paquet est identifié avec les drapeaux SYN et ACK.
3. **Client envoie un paquet ACK** :
- Le client (10.10.30.255) envoie un paquet avec le drapeau ACK au serveur pour accuser réception de la réponse du serveur et terminer la poignée de main.
 - **Frame 48** : Ce paquet est identifié avec le drapeau ACK.

Diagramme de la Connexion

Voici un diagramme simple représentant les étapes de la connexion TCP :

scss

Copier le code



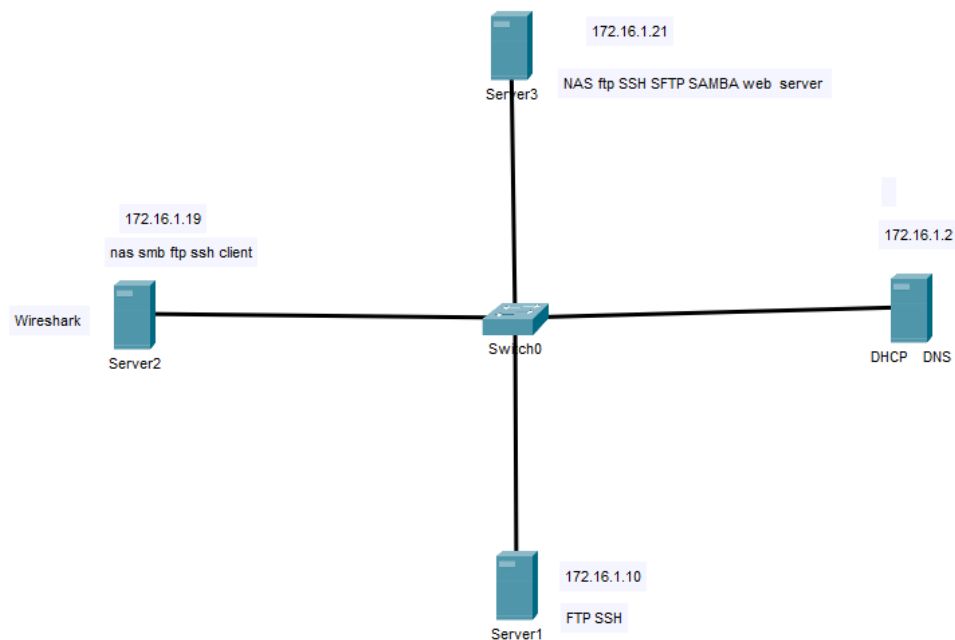
Description des Paquets

- Frame 46: SYN**
 - **Src Port**: 58300
 - **Dst Port**: 443
 - **Flags**: SYN (0x002)
 - **Seq**: 0
- Frame 47: SYN-ACK**
 - **Src Port**: 443
 - **Dst Port**: 58300
 - **Flags**: SYN, ACK (0x012)
 - **Seq**: 0
 - **Ack**: 1
- Frame 48: ACK**
 - **Src Port**: 58300
 - **Dst Port**: 443
 - **Flags**: ACK (0x010)
 - **Seq**: 1
 - **Ack**: 1

Ces trois paquets montrent le processus de handshake TCP utilisé pour établir une connexion fiable entre un client et un serveur.

Partie 2

-notre lab LAN



-filtre appliqué pour certains protocols

icmp or (udp port 67 or udp port 68) or port 53 or udp port 5353 or port 6500 or port 443 or port 21 or port 20 or tcp port 445 or tcp port 139 or tcp port 22 or http or port 10000

icmp : Capture tous les paquets ICMP.

udp port 67 or udp port 68 : Capture les paquets UDP sur les ports 67 et 68 (DHCP).

port 53 : Capture les paquets TCP ou UDP sur le port 53 (DNS).

udp port 5353 : Capture les paquets UDP sur le port 5353 (mDNS).

port 6500 : Capture les paquets TCP ou UDP sur le port 6500 (port personnalisé).

port 443 : Capture les paquets TCP ou UDP sur le port 443 (HTTPS et TLS).

port 21 or port 20 : Capture les paquets TCP sur les ports 21 et 20 (FTP).

tcp port 445 or tcp port 139 : Capture les paquets TCP sur les ports 445 et 139 (SMB).

tcp port 22 : Capture les paquets TCP sur le port 22 (SFTP).

http : Capture les paquets HTTP sur n'importe quel port.

ssl : Capture les paquets SSL/TLS (ce qui inclut HTTPS) sur n'importe quel port.

-manipulation sur nos serveurs

-ping 172.16.1.10

-ouvrir un page web

sftp alaa@172.16.1.21:/home/alaalraid5

smbclient //172.16.1.21/alaalraid5 -U alaa

ssh alaa@172.16.1.21

https://172.16.1.21:10000

ftp 172.16.1.10

ssh -p 6500 laplateforme@172.16.1.10

partie 2

-ICMP

Paquet 106

- **Détails de la trame :**
 - **Taille :** 98 octets (784 bits)
 - **Arrivée :** 9 juin 2024, 22:48:27.170299550 CEST
 - **Delta depuis la trame précédente capturée :** 0,142865329 secondes
 - **Delta depuis la trame précédente affichée :** 1,280370547 secondes
 - **Temps depuis la référence ou la première trame :** 28,472826970 secondes
- **Ethernet II :**
 - **Source :** VMware_68:ef
(00:0c:29:68:ef
)
 - **Destination :** VMware_3a:2a
(00:0c:29:3a:2a
)
 - **Type :** IPv4 (0x0800)
- **Protocole IPv4 :**
 - **Source :** 172.16.1.19
 - **Destination :** 172.16.1.10
 - **TTL :** 64
 - **Protocole :** ICMP (1)
 - **Checksum en-tête :** 0xf7f2 (non vérifié)
 - **Flags :** Ne pas fragmenter
- **Protocole ICMP :**
 - **Type :** 8 (Echo (ping) request)
 - **Code :** 0
 - **Checksum :** 0x4a01 (correct)
 - **Identifiant :** 18969 (0x4a19)
 - **Numéro de séquence :** 1 (0x0001)

- **Horodatage des données ICMP** : 9 juin 2024, 22:48:27.000000000 CEST
- **Données** : 48 octets
(2196020000000000101112131415161718191a1b1c1d1e1f202122232425262728292a2b...)

Paquet 107

- **Détails de la trame :**
 - **Taille** : 98 octets (784 bits)
 - **Arrivée** : 9 juin 2024, 22:48:27.170981489 CEST
 - **Delta depuis la trame précédente capturée** : 0,000681939 secondes
 - **Delta depuis la trame précédente affichée** : 0,000681939 secondes
 - **Temps depuis la référence ou la première trame** : 28,473508909 secondes
- **Ethernet II :**
 - **Source** : VMware_3a:2a
(00:0c:29:3a:2a)
 - **Destination** : VMware_68:ef
(00:0c:29:68:ef)
 - **Type** : IPv4 (0x0800)
- **Protocole IPv4 :**
 - **Source** : 172.16.1.10
 - **Destination** : 172.16.1.19
 - **TTL** : 64
 - **Protocole** : ICMP (1)
 - **Checksum en-tête** : 0xc2ac (non vérifié)
 - **Flags** : Aucun (fragmentation non activée)
- **Protocole ICMP :**
 - **Type** : 0 (Echo (ping) reply)
 - **Code** : 0
 - **Checksum** : 0x5201 (correct)
 - **Identifiant** : 18969 (0x4a19)
 - **Numéro de séquence** : 1 (0x0001)
 - **Temps de réponse** : 0,682 ms
 - **Horodatage des données ICMP** : 9 juin 2024, 22:48:27.000000000 CEST
 - **Données** : 48 octets
(2196020000000000101112131415161718191a1b1c1d1e1f202122232425262728292a2b...)

Résumé

- **Paquet 106** : C'est une requête ping (ICMP Echo Request) de l'adresse IP 172.16.1.19 vers 172.16.1.10.
- **Paquet 107** : C'est une réponse ping (ICMP Echo Reply) de l'adresse IP 172.16.1.10 vers 172.16.1.19, répondant à la requête du paquet 106.

Les informations importantes incluent les adresses MAC et IP, les types de paquets ICMP, les numéros de séquence et les identifiants, ainsi que les timestamps et la latence mesurée.

40

-DHCP

Trame 91: DHCP Request

Frame Details:

- **Length:** 342 bytes
- **Protocols:** Ethernet, IPv4, UDP, DHCP

Ethernet II:

- **Src:** VMware_3a:2a (00:0c:29:3a:2a)
- **Dst:** VMware_29:59:71 (00:0c:29:29:59:71)
- **Type:** IPv4 (0x0800)

IPv4:

- **Src IP:** 172.16.1.10
- **Dst IP:** 172.16.1.2
- **Total Length:** 328
- **Protocol:** UDP (17)
- **Identification:** 0x012a (298)
- **Flags:** Don't fragment

UDP:

- **Src Port:** 68
- **Dst Port:** 67

- **Length:** 308

DHCP:

- **Message Type:** Boot Request (1)
- **Client IP Address:** 172.16.1.10
- **Client MAC Address:** VMware_3a:2a
(00:0c:29:3a:2a
)
- **Options:**
 - **(53) DHCP Message Type:** Request (3)
 - **(12) Host Name:** debian
 - **(55) Parameter Request List:** Subnet Mask, Broadcast Address, Time Offset, Router, Domain Name, Domain Name Server, Domain Search, Host Name, NetBIOS over TCP/IP Name Server, NetBIOS over TCP/IP Scope, Interface MTU, Classless Static Route, Network Time Protocol Servers
 - **(61) Client identifier:** Link-layer address plus time

Résumé: Cette trame est une requête DHCP envoyée par le client (IP 172.16.1.10) à un serveur DHCP pour demander une configuration réseau. Le client utilise l'adresse MAC VMware_3a:2a

et demande divers paramètres réseau (masque de sous-réseau, passerelle, etc.).

Trame 92: DHCP ACK

Frame Details:

- **Length:** 342 bytes
- **Protocols:** Ethernet, IPv4, UDP, DHCP

Ethernet II:

- **Src:** VMware_29:59:71 (00:0c:29:29:59:71)
- **Dst:** VMware_3a:2a
(00:0c:29:3a:2a
)
- **Type:** IPv4 (0x0800)

IPv4:

- **Src IP:** 172.16.1.2
- **Dst IP:** 172.16.1.10
- **Total Length:** 328
- **Protocol:** UDP (17)

- **Identification:** 0x4b76 (19318)
- **Flags:** Don't fragment

UDP:

- **Src Port:** 67
- **Dst Port:** 68
- **Length:** 308

DHCP:

- **Message Type:** Boot Reply (2)
- **Client IP Address:** 172.16.1.10
- **Your (client) IP Address:** 172.16.1.10
- **Client MAC Address:** VMware_3a:2a
(00:0c:29:3a:2a
)
- **Options:**
 - **(53) DHCP Message Type:** ACK (5)
 - **(54) DHCP Server Identifier:** 172.16.1.2
 - **(51) IP Address Lease Time:** 600 seconds (10 minutes)
 - **(1) Subnet Mask:** 255.255.0.0
 - **(3) Router:** 172.16.1.1
 - **(15) Domain Name:** example.org
 - **(6) Domain Name Server:** 172.16.1.2

Résumé: Cette trame est une réponse DHCP ACK envoyée par le serveur DHCP (IP 172.16.1.2) au client (IP 172.16.1.10). Le serveur confirme que le client peut utiliser l'adresse IP 172.16.1.10 et fournit d'autres configurations réseau, telles que le masque de sous-réseau, la passerelle par défaut, et le serveur DNS.

Conclusion

Ces deux trames montrent un échange DHCP où un client (172.16.1.10) demande une configuration réseau et le serveur (172.16.1.2) répond avec les informations nécessaires. La première trame est la requête DHCP, et la deuxième est l'accusé de réception confirmant l'attribution des paramètres demandés.

-DNS

Analyse des Trames 77 et 97 : Communication DNS

Trame 77 : Requête DNS

1. Informations Générales :
 - Longueur de la Trame : 81 octets (648 bits)
 - Temps d'Arrivée : 9 juin 2024, 22:48:18.890453410 CEST
 - Delta de Temps depuis la Trame Précédente : 0.000000601 secondes
 - Temps depuis la Première Trame : 20.192980830 secondes
 - Numéro de la Trame : 77
2. En-tête Ethernet :
 - Adresse MAC Source : VMware_0b:4c
(00:0c:29:0b:4c
)
 - Adresse MAC Destination : VMware_29:59:71 (00:0c:29:29:59:71)
 - Type : IPv4 (0x0800)
3. Protocole IP :
 - Version : IPv4
 - Adresse IP Source : 172.16.1.21
 - Adresse IP Destination : 172.16.1.2
 - Taille de l'En-tête : 20 octets
 - Longueur Totale : 67 octets
 - Identification : 0xfd50 (64848)
 - Flags : Don't fragment
 - TTL (Time to Live) : 64
 - Protocole : UDP (17)
 - Checksum de l'En-tête : 0xe321 (non vérifié)
4. Protocole UDP :
 - Port Source : 41135
 - Port Destination : 53
 - Longueur : 47 octets
 - Checksum : 0x133d (non vérifié)
5. Protocole DNS :
 - ID de Transaction : 0xc146
 - Flags : 0x0100 (Requête standard, Recursion desired)
 - Questions : 1
 - Réponses : 0
 - Autorité : 0
 - Additionnelles : 0
 - Requête : 2.debian.pool.ntp.org, type AAAA (IPv6 Address), classe IN

Trame 97 : Réponse DNS

1. Informations Générales :
 - Longueur de la Trame : 81 octets (648 bits)
 - Temps d'Arrivée : 9 juin 2024, 22:48:25.889361663 CEST
 - Delta de Temps depuis la Trame Précédente : 0.055711689 secondes
 - Temps depuis la Première Trame : 27.191889083 secondes
 - Numéro de la Trame : 97
2. En-tête Ethernet :

- **Adresse MAC Source** : VMware_29:59:71 (00:0c:29:29:59:71)
- **Adresse MAC Destination** : VMware_0b:4c (00:0c:29:0b:4c)
- **Type** : IPv4 (0x0800)
- 3. **Protocole IP** :
 - **Version** : IPv4
 - **Adresse IP Source** : 172.16.1.2
 - **Adresse IP Destination** : 172.16.1.21
 - **Taille de l'En-tête** : 20 octets
 - **Longueur Totale** : 67 octets
 - **Identification** : 0xba5a (47706)
 - **Flags** : 0x0 (Pas de fragmentation)
 - **TTL (Time to Live)** : 64
 - **Protocole** : UDP (17)
 - **Checksum de l'En-tête** : 0x6618 (non vérifié)
- 4. **Protocole UDP** :
 - **Port Source** : 53
 - **Port Destination** : 41135
 - **Longueur** : 47 octets
 - **Checksum** : 0x92ba (non vérifié)
- 5. **Protocole DNS** :
 - **ID de Transaction** : 0xc146
 - **Flags** : 0x8182 (Réponse standard, Échec du serveur)
 - **Code de Réponse** : 2 (Échec du serveur)
 - **Questions** : 1
 - **Réponses** : 0
 - **Autorité** : 0
 - **Additionnelles** : 0
 - **Requête** : 2.debian.pool.ntp.org, type AAAA (IPv6 Address), classe IN

Analyse Complémentaire :

- **Temps entre les Trames** : Il y a un intervalle de 6.998908253 secondes entre la requête DNS (trame 77) et la réponse DNS (trame 97). Ce délai peut indiquer que le serveur DNS a pris du temps à traiter la requête avant de répondre avec un échec.
- **Adresse IP et Ports** : La trame 77 montre que l'ordinateur avec l'adresse IP 172.16.1.21 demande la résolution du nom 2.debian.pool.ntp.org via le port UDP 53 (port DNS standard). La trame 97 montre que le serveur DNS à l'adresse IP 172.16.1.2 répond à cette demande.
- **Échec du Serveur DNS** : La réponse DNS (trame 97) indique un échec du serveur (code de réponse 2). Cela signifie que le serveur DNS n'a pas pu traiter la demande de résolution du nom pour 2.debian.pool.ntp.org.

Ces informations montrent une communication standard entre un client et un serveur DNS, où la requête de résolution échoue, peut-être en raison de problèmes de configuration du serveur DNS ou d'autres problèmes internes au serveur.

-mDNS

Message mDNS ? (qu'est-ce que c'est ?)

mDNS, ou Multicast DNS, est un protocole de réseau utilisé pour la découverte et la résolution de noms de domaine au sein de petits réseaux locaux sans serveur DNS centralisé. Il permet aux appareils sur le réseau de s'identifier et de communiquer entre eux en utilisant des noms d'hôte simples tels que "mon-ordinateur.local" plutôt que des adresses IP numériques. En utilisant des diffusions multicast, les appareils peuvent annoncer leurs noms de domaine et résoudre les noms des autres appareils sans nécessiter d'infrastructure de serveur DNS. Cela rend la configuration et l'utilisation des appareils sur le réseau local plus simples et plus conviviales pour les utilisateurs.

Analyse de la Trame 14 : Requête Multicast DNS (mDNS)

Informations Générales

1. **Longueur de la Trame** : 87 octets (696 bits)
2. **Temps d'Arrivée** : 9 juin 2024, 22:47:59.930327426 CEST
3. **Delta de Temps depuis la Trame Précédente** : 0.192119258 secondes
4. **Temps depuis la Première Trame** : 1.232854846 secondes
5. **Numéro de la Trame** : 14

En-tête Ethernet

1. **Adresse MAC Source** : VMware_68:ef
(00:0c:29:68:ef
)
2. **Adresse MAC Destination** : IPv4mcast_fb (01:00:5e:00:00
)
 - **Adresse Globale** : Oui (LG bit: 0)
 - **Adresse de Groupe (Multicast)** : Oui (IG bit: 1)

3. **Type** : IPv4 (0x0800)

Protocole IP

1. **Version** : IPv4
2. **Adresse IP Source** : 172.16.1.19
3. **Adresse IP Destination** : 224.0.0.251 (Multicast DNS)
4. **Taille de l'En-tête** : 20 octets
5. **Longueur Totale** : 73 octets
6. **Identification** : 0xca0b (51723)
7. **Flags** : Don't fragment
8. **TTL (Time to Live)** : 255
9. **Protocole** : UDP (17)
10. **Checksum de l'En-tête** : 0x2379 (non vérifié)

Protocole UDP

1. **Port Source** : 5353 (mDNS)
2. **Port Destination** : 5353 (mDNS)
3. **Longueur** : 53 octets
4. **Checksum** : 0x8e65 (non vérifié)

Protocole mDNS

1. **ID de Transaction** : 0x0000
2. **Flags** : 0x0000 (Requête standard)
3. **Questions** : 2
4. **Réponses** : 0
5. **Autorité** : 0
6. **Additionnelles** : 0

Détails des Requêtes mDNS

1. **Requête 1** :
 - **Nom** : `_ipps._tcp.local`
 - **Type** : PTR (domain name PoinTeR) (12)
 - **Classe** : IN (Internet) (0x0001)
 - **Question** : Non (QU bit: 0)
2. **Requête 2** :
 - **Nom** : `_ipp._tcp.local`
 - **Type** : PTR (domain name PoinTeR) (12)
 - **Classe** : IN (Internet) (0x0001)
 - **Question** : Non (QU bit: 0)

Analyse Complémentaire

- **Requête mDNS** : Cette trame représente une requête mDNS envoyée par un appareil pour découvrir des services réseau via Multicast DNS. Les requêtes mDNS sont couramment utilisées pour la découverte de services sur les réseaux locaux, tels que les imprimantes (services **_ipp** et **_ipps** pour le protocole d'impression sur Internet).
- **Adresse IP Destination** : 224.0.0.251 est l'adresse multicast réservée pour mDNS. Les appareils sur le réseau local peuvent écouter cette adresse pour répondre aux requêtes de découverte de services.
- **TTL de 255** : Indique que cette requête mDNS est destinée à rester sur le réseau local et ne doit pas être routée à travers des routeurs.
- **Adresses MAC** :
 - **Source (VMware_68:ef)** : Indique l'appareil qui a émis la requête.
 - **Destination (IPv4mcast_fb)** : Adresse multicast pour mDNS.

Cette trame illustre une opération typique de découverte de services via mDNS, où un appareil demande des informations sur les services d'impression disponibles sur le réseau local. Les autres appareils sur le réseau qui fournissent ces services devraient répondre à cette requête, permettant ainsi à l'émetteur de découvrir et d'interagir avec ces services.

-SMB

1. Trame 1002 :

- **Ethernet II** : Cette trame Ethernet indique :
 - Adresse MAC source : 00:0c:29:68:ef (VMware_68:ef)
 - Adresse MAC destination : 00:0c:29:0b:4c (VMware_0b:4c)
- **IPv4** : Cela concerne le protocole IPv4 et inclut :
 - Adresse IP source : 172.16.1.19
 - Adresse IP destination : 172.16.1.21
 - Protocole TCP utilisé.
- **TCP** : Les détails TCP incluent :
 - Port source : 54306
 - Port destination : 445
 - Flags TCP : PSH, ACK

- **Service de session NetBIOS** : Un message de session NetBIOS est présent.
 - **SMB2** : Il y a une négociation du protocole SMB2.
2. **Trame 1004** :
- **Ethernet II** : Cette trame Ethernet indique :
 - Adresse MAC source : 00:0c:29:0b:4c (VMware_0b:4c)
 - Adresse MAC destination : 00:0c:29:68:ef (VMware_68:ef)
 - **IPv4** : Les détails IPv4 sont :
 - Adresse IP source : 172.16.1.21
 - Adresse IP destination : 172.16.1.19
 - Protocole TCP utilisé.
 - **TCP** : Les détails TCP incluent :
 - Port source : 445
 - Port destination : 54306
 - Flags TCP : PSH, ACK
 - **Service de session NetBIOS** : Un message de session NetBIOS est présent.
 - **SMB2** : Il s'agit d'une réponse à la négociation SMB2.

Ces trames représentent une communication réseau entre deux machines. La première trame semble être une demande de négociation du protocole SMB2 de la machine ayant l'adresse IP 172.16.1.19 vers celle avec l'adresse IP 172.16.1.21. La trame 1004 représente la réponse à cette demande.

partie2_2

FTP

Ces paquets représentent une session FTP entre deux machines, où des commandes et des réponses sont échangées pour authentifier un utilisateur et établir une connexion de transfert de fichiers.

1. Paquet 1 (Frame 184538) :

- **Ethernet II** : Adresse MAC source : 00:0c:29:29:59:71, Adresse MAC destination : 00:0c:29:3a:2a:ff.
- **IPv4** : Adresse IP source : 172.16.1.2, Adresse IP destination : 172.16.1.10.
- **TCP** : Port source : 57450, Port destination : 21.
- **FTP** : Commande USER laplateforme\r\n envoyée par l'utilisateur.

2. Paquet 2 (Frame 184540) :

- **Ethernet II** : Adresse MAC source : 00:0c:29:3a:2a:ff, Adresse MAC destination : 00:0c:29:29:59:71.
- **IPv4** : Adresse IP source : 172.16.1.10, Adresse IP destination : 172.16.1.2.
- **TCP** : Port source : 21, Port destination : 57450.
- **FTP** : Réponse 331 Mot de passe requis pour laplateforme\r\n, demandant le mot de passe.

3. Paquet 3 (Frame 187754) :

- **Ethernet II** : Adresse MAC source : 00:0c:29:29:59:71, Adresse MAC destination : 00:0c:29:3a:2a:ff.
- **IPv4** : Adresse IP source : 172.16.1.2, Adresse IP destination : 172.16.1.10.
- **TCP** : Port source : 57450, Port destination : 21.
- **FTP** : Commande PASS alaa\r\n envoyée par l'utilisateur, avec le mot de passe.

4. Paquet 4 (Frame 187874) :

- **Ethernet II** : Adresse MAC source : 00:0c:29:3a:2a:ff, Adresse MAC destination : 00:0c:29:29:59:71.
- **IPv4** : Adresse IP source : 172.16.1.10, Adresse IP destination : 172.16.1.2.
- **TCP** : Port source : 21, Port destination : 57450.
- **FTP** : Réponse 230 Utilisateur laplateforme authentifié\r\n, indiquant que l'utilisateur est authentifié avec succès.

-HTTP et https

Paquet 1 (Frame 75840):

- **Ethernet II:**
 - Adresse source: 00:0c:29:68:ef
 - Adresse de destination: 00:0c:29:0b:4c
- **IPv4:**
 - Adresse source: 172.16.1.19
 - Adresse de destination: 172.16.1.21
 - Protocole: TCP (6)
- **TCP:**
 - Port source: 50342
 - Port de destination: 10000
 - Drapeaux: PSH, ACK
 - Données TCP: Requête HTTP GET

Paquet 2 (Frame 75850):

- **Ethernet II:**
 - Adresse source: 00:0c:29:0b:4c
 - Adresse de destination: 00:0c:29:68:ef
- **IPv4:**
 - Adresse source: 172.16.1.21
 - Adresse de destination: 172.16.1.19
 - Protocole: TCP (6)
- **TCP:**
 - Port source: 10000
 - Port de destination: 50342
 - Drapeaux: FIN, ACK
 - Données TCP: Réponse HTTP 200 OK

Analyse détaillée :

- Le premier paquet est une requête HTTP GET envoyée depuis l'adresse IP 172.16.1.19 au port 10000 de l'adresse IP 172.16.1.21. Le serveur web distant répond avec un paquet HTTP 200 OK.
- La taille des données dans le premier paquet est de 362 octets, tandis que la taille dans le second paquet est de 0 octet, indiquant la fin de la réponse.
- La connexion TCP est établie avec succès et se termine de manière ordonnée avec le drapeau FIN.
- Les données de la réponse HTTP incluent des métadonnées telles que le serveur (MiniServ), la date, le type de contenu (text/html), et la connexion est fermée après la réponse.

3.5

est ce que ca utilise le ssl ?

ChatGPT

Le premier paquet ne semble pas utiliser SSL (Secure Sockets Layer) ou son successeur, TLS (Transport Layer Security), car le port utilisé est le port 10000, qui est généralement associé à d'autres protocoles comme le Webmin ou d'autres applications personnalisées. De plus, les données de la requête HTTP GET sont envoyées en texte clair, ce qui est typique des communications HTTP non sécurisées.

Cependant, la réponse du serveur dans le deuxième paquet indique que le serveur web fonctionne en mode SSL (HTTPS). Voici pourquoi :

- Le message de réponse contient une redirection vers une URL HTTPS : "This web server is running in SSL mode. Trying to redirect to <https://172.16.1.21:10000/>".
- Bien que le premier paquet ne montre pas de connexion SSL, il est possible que la redirection vers HTTPS soit effectuée par le serveur après réception de la requête HTTP non sécurisée.

En résumé, bien que la communication initiale ne semble pas utiliser SSL/TLS, le serveur web semble capable de fonctionner en mode SSL/TLS et tente de rediriger les clients HTTP vers une connexion sécurisée.

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

Dans les échanges FTP sans TLS, nous pouvons observer les commandes envoyées et les réponses reçues. Par exemple, dans le premier paquet, nous voyons une commande USER envoyée pour l'utilisateur "laplateforme". Dans la réponse du serveur, nous avons le code 331, indiquant que le mot de passe est requis pour l'utilisateur spécifié.

Il est tout à fait possible de récupérer des données sensibles de connexion dans ce contexte, car les échanges FTP sans TLS ne chiffrent pas le trafic. Ainsi, les noms d'utilisateur et les mots de passe sont envoyés en texte clair et peuvent être facilement interceptés par des tiers.

En ce qui concerne les échanges SSL (Secure Sockets Layer) ou TLS (Transport Layer Security), ils fournissent un niveau de sécurité supplémentaire en chiffrant les données transitant entre le client et le serveur. Cela signifie que même si vous interceptez les paquets, les données sensibles, telles que les noms d'utilisateur et les mots de passe, seront illisibles car elles sont chiffrées.

partie 3

1- les paquet capturé sont séparés selon le protocol dans des fichiers séparés

```
#!/bin/bash
```

```
INTERFACE="ens33" # Votre interface réseau
```

```
# Fonction pour nettoyer les processus en arrière-plan
```

```
cleanup() {  
    echo "Stopping captures..."  
    for pid in "${PIDS[@]"; do  
        kill $pid  
    done  
    exit 0  
}
```

```
# Trap pour capturer Ctrl+C (SIGINT) et exécuter la fonction de nettoyage  
trap cleanup SIGINT
```

```
# Démarrer les captures et enregistrer les PID  
PIDS=()
```

```
echo "Capturing DHCP packets..."  
tshark -i $INTERFACE -f "udp port 67 or udp port 68" -w dhcp_packets.pcap &  
PIDS+=($!)
```

```
echo "Capturing DNS packets..."  
tshark -i $INTERFACE -f "udp port 53 or tcp port 53" -w dns_packets.pcap &  
PIDS+=($!)
```

```
echo "Capturing mDNS packets..."
tshark -i $INTERFACE -f "udp port 5353" -w mdns_packets.pcap &
PIDS+=($!)

echo "Capturing SSL packets..."
tshark -i $INTERFACE -f "tcp port 443" -w ssl_packets.pcap &
PIDS+=($!)

echo "Capturing FTP packets..."
tshark -i $INTERFACE -f "tcp port 21 or tcp port 20" -w ftp_packets.pcap &
PIDS+=($!)

echo "Capturing SMB packets..."
tshark -i $INTERFACE -f "tcp port 139 or tcp port 445" -w smb_packets.pcap &
PIDS+=($!)

echo "Capturing HTTPS packets..."
tshark -i $INTERFACE -f "tcp port 443" -w https_packets.pcap &
PIDS+=($!)

echo "Capturing TLSv1.2 packets..."
tshark -i $INTERFACE -f "tcp port 443" -w tls12_packets.pcap &
PIDS+=($!)

echo "All captures started. Press Ctrl+C to stop."

# Attendre indéfiniment pour garder le script en cours d'exécution
wait
```

2-tout les protocole sont dans fichier pcap

```
#!/bin/bash
```

```
INTERFACE="ens33" # Votre interface réseau
```

```
echo "Capturing all specified packets..."
```

```
# Capturer tous les types de paquets spécifiés dans un seul fichier de capture
tshark -i $INTERFACE -f "udp port 67 or udp port 68 or udp port 53 or tcp port
53 or udp port 5353 or tcp port 443 or tcp port 21 or tcp port 20 or tcp port 139
or tcp port 445" -w all_packets.pcap &
```

```
# Enregistrer le PID du processus tshark pour pouvoir l'arrêter plus tard
TSPID=$!
```

```
echo "Capture started. Press Ctrl+C to stop."
```

```
# Fonction pour nettoyer les processus tshark
```

```
cleanup() {
    echo "Stopping capture..."
    kill $TSPID
    exit 0
}
```

```
# Trap pour capturer Ctrl+C (SIGINT) et exécuter la fonction de nettoyage
trap cleanup SIGINT
```

```
# Attendre indéfiniment pour garder le script en cours d'exécution
wait
```