Documentation Wireshark - Partie 1

Introduction à Wireshark

Wireshark est un analyseur de protocole réseau puissant et polyvalent qui permet de capturer et d'examiner en détail le trafic circulant sur un réseau informatique. Développé initialement sous le nom d'Ethereal, cet outil open-source est devenu une référence incontournable pour les administrateurs réseau, les experts en sécurité informatique et les développeurs. Sa capacité à décortiquer les paquets réseau en fait un allié précieux pour comprendre, diagnostiquer et résoudre les problèmes de communication entre les systèmes informatiques.

L'interface graphique de Wireshark offre une visualisation claire et organisée des données capturées, permettant d'analyser chaque couche du modèle OSI (Open Systems Interconnection). Cette fonctionnalité est particulièrement utile pour comprendre comment les différents protocoles interagissent entre eux et comment les données sont encapsulées à travers les différentes couches du réseau.

Différence entre trame et paquet

Dans le domaine des réseaux informatiques, les termes "trame" et "paquet" sont souvent utilisés, parfois de manière interchangeable, mais ils désignent en réalité des concepts distincts qui correspondent à différentes couches du modèle OSI.

Une trame (frame) est une unité de données au niveau de la couche 2 du modèle OSI, la couche liaison de données. Elle contient généralement une en-tête avec les adresses MAC source et destination, les données encapsulées (qui peuvent être un paquet de couche supérieure), et une séquence de contrôle permettant de vérifier l'intégrité des données. Les trames sont spécifiques au type de réseau utilisé, comme Ethernet, Wi-Fi ou Token Ring.

Un paquet (packet), quant à lui, est une unité de données au niveau de la couche 3 du modèle OSI, la couche réseau. Il contient les adresses IP source et destination, ainsi que d'autres informations de contrôle nécessaires au routage à travers différents réseaux. Le paquet est encapsulé dans une trame pour être transmis sur le réseau physique.

Pour illustrer cette différence, on peut observer dans les captures d'écran fournies que Wireshark affiche à la fois les informations de la trame Ethernet (couche 2) avec les adresses MAC, et les informations du paquet IP (couche 3) avec les adresses IP. Cette

hiérarchie d'encapsulation est fondamentale pour comprendre comment les données transitent à travers un réseau.

Le format PCAP/PCAPNG

Le format PCAP (Packet Capture) est un format de fichier standard utilisé pour stocker les données de capture réseau. Il a été développé pour l'outil tcpdump, mais est devenu un standard de facto pour de nombreux outils d'analyse réseau, dont Wireshark. PCAPNG (Packet Capture Next Generation) est une évolution plus récente et plus flexible de ce format.

Ces formats permettent de sauvegarder l'intégralité des paquets capturés, y compris les en-têtes et les données, avec des informations temporelles précises. Cela permet non seulement d'analyser les captures en temps réel, mais aussi de les stocker pour une analyse ultérieure ou de les partager avec d'autres analystes.

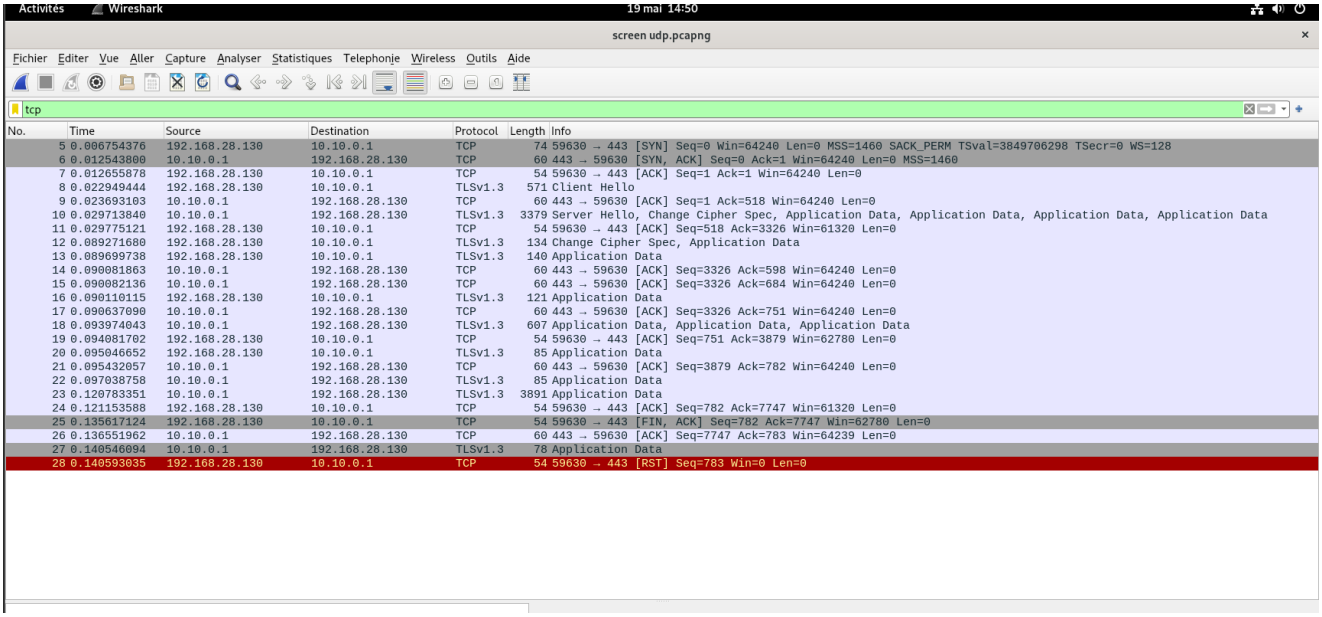
Les avantages du format PCAPNG par rapport au format PCAP original incluent : - La possibilité de stocker plusieurs interfaces dans un même fichier - L'ajout de métadonnées et de commentaires - Une meilleure prise en charge des différents types de liaisons - La possibilité d'inclure des statistiques et des informations sur les paquets perdus

Dans Wireshark, les captures sont généralement enregistrées au format PCAPNG par défaut, ce qui offre une plus grande flexibilité pour l'analyse et le partage des données.

Analyse des captures avec Wireshark

Capture de paquets TCP

Dans les captures d'écran fournies, nous pouvons observer une série de paquets TCP échangés entre deux hôtes. La première capture montre clairement l'établissement d'une connexion TCP suivant le processus de "three-way handshake" (poignée de main à trois temps).



Dans cette capture, on peut identifier : 1. Un paquet SYN (Synchronize) envoyé de 192.168.28.130 vers 10.10.0.1, marquant le début de la tentative de connexion 2. Un paquet SYN-ACK (Synchronize-Acknowledge) envoyé en réponse par 10.10.0.1 vers 192.168.28.130 3. Un paquet ACK (Acknowledge) final envoyé par 192.168.28.130 vers 10.10.0.1, confirmant l'établissement de la connexion

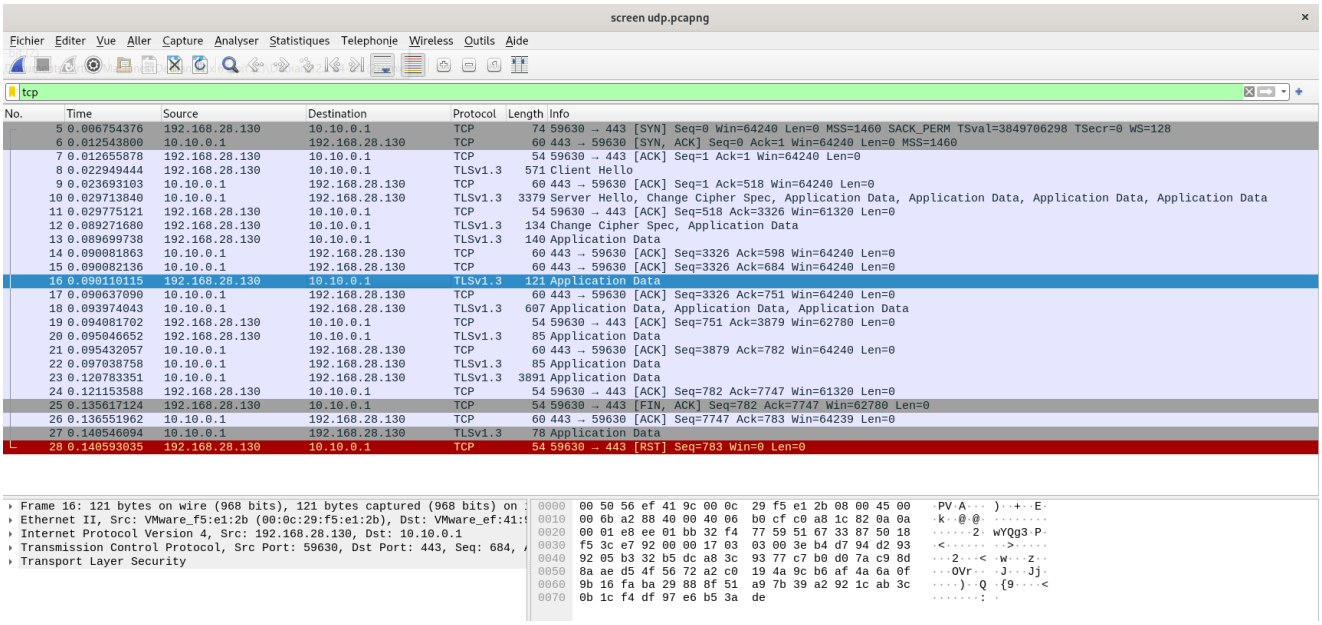
Cette séquence est fondamentale dans le protocole TCP et garantit que les deux parties sont prêtes à communiquer. On peut également observer dans la capture les numéros de séquence (Seq) et d'acquittement (Ack) qui permettent de suivre l'ordre des paquets et de détecter d'éventuelles pertes.

Après l'établissement de la connexion, on peut voir l'échange de données entre les deux hôtes, avec des paquets contenant des informations comme "Client Hello" et "Server Hello, Change Cipher Spec, Application Data", indiquant qu'une négociation TLS (Transport Layer Security) est en cours pour sécuriser la communication.

La fin de la capture montre également la terminaison de la connexion avec un paquet RST (Reset), indiquant une fermeture abrupte plutôt qu'une fermeture gracieuse qui utiliserait normalement des paquets FIN (Finish).

Désencapsulation des trames

Wireshark permet de visualiser la désencapsulation des trames pour identifier les différentes couches du modèle OSI. Dans la capture suivante, on peut observer en détail la structure d'un paquet TCP :

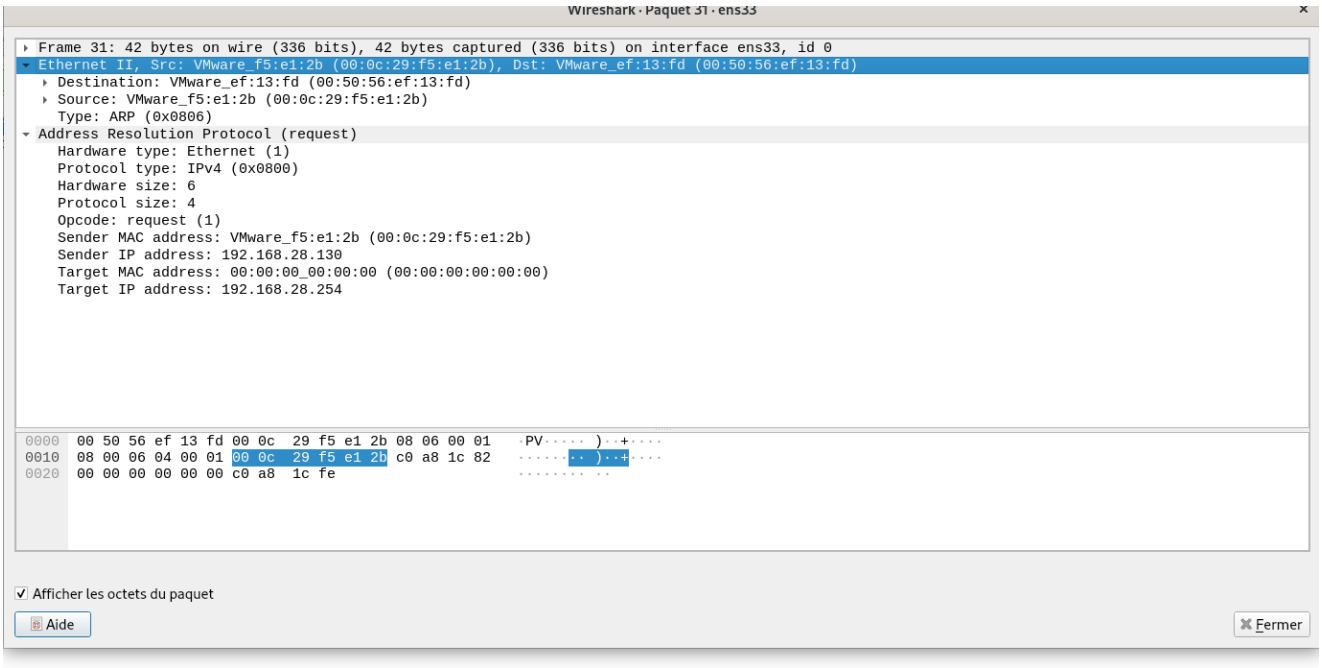


Cette capture montre : - La couche Ethernet (couche 2) avec les adresses MAC source et destination - La couche IP (couche 3) avec les adresses IP source et destination - La couche TCP (couche 4) avec les ports source et destination, les numéros de séquence et d'acquittement - La couche Application (couches 5-7) avec les données TLS

En bas de la fenêtre, on peut également voir la représentation hexadécimale du paquet, permettant d'analyser son contenu brut. Cette visualisation est particulièrement utile pour comprendre comment les données sont structurées et encapsulées à travers les différentes couches du réseau.

Capture de paquets ARP

Le protocole ARP (Address Resolution Protocol) est utilisé pour traduire les adresses IP en adresses MAC sur un réseau local. Dans la capture suivante, on peut observer une requête ARP :

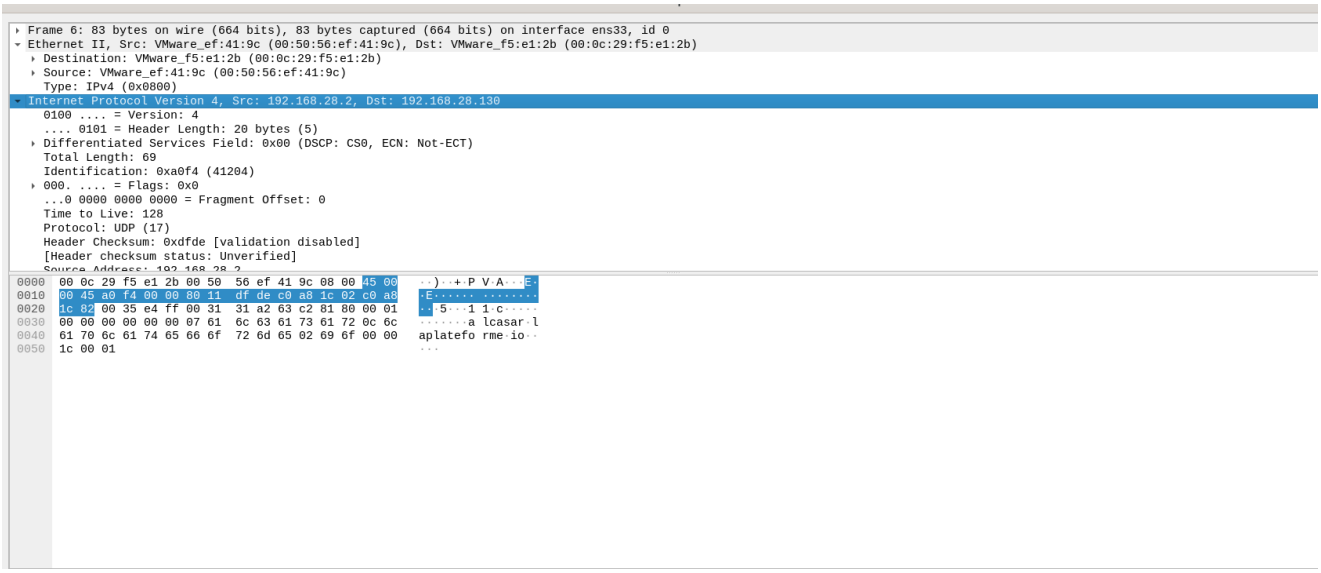


Cette capture montre une requête ARP envoyée par l'hôte 192.168.28.130 pour découvrir l'adresse MAC correspondant à l'adresse IP 192.168.28.254. On peut observer les détails suivants : - Type de matériel : Ethernet (1) - Type de protocole : IPv4 (0x0800) - Taille du matériel : 6 (octets pour l'adresse MAC) - Taille du protocole : 4 (octets pour l'adresse IPv4) - Opcode : requête (1) - Adresse MAC de l'expéditeur : VMware_f5:e1:2b (00:0c:29:f5:e1:2b) - Adresse IP de l'expéditeur : 192.168.28.130 - Adresse MAC cible : 00:00:00:00:00:00 (inconnue, c'est ce que la requête cherche à découvrir) - Adresse IP cible : 192.168.28.254

Cette requête ARP est un exemple parfait de la façon dont les protocoles de couche 2 et 3 interagissent pour permettre la communication sur un réseau local.

Capture de paquets UDP

Le protocole UDP (User Datagram Protocol) est un protocole de transport sans connexion, contrairement à TCP. Dans la capture suivante, on peut observer un paquet UDP :

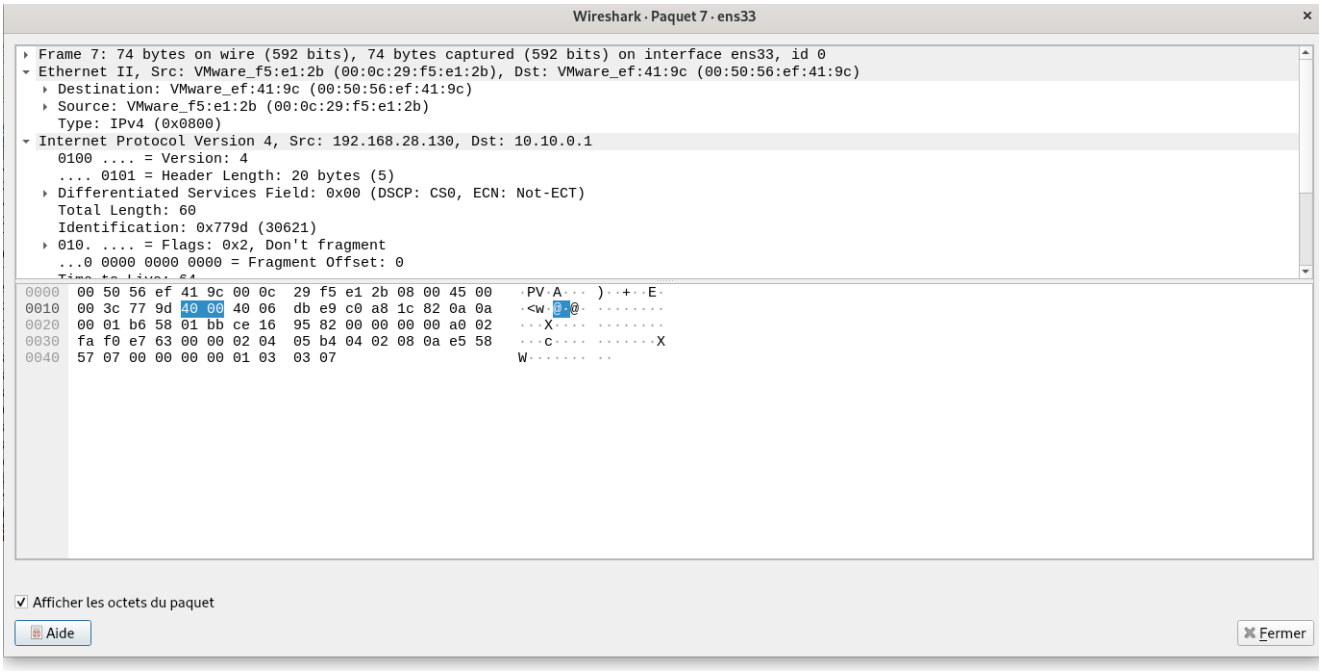


Cette capture montre un paquet UDP envoyé de 192.168.28.2 à 192.168.28.130. On peut observer les détails suivants : - Version IP : 4 - Longueur d'en-tête : 20 octets (5 mots de 32 bits) - Champ de services différenciés : 0x00 (DSCP: CS0, ECN: Not-ECT) - Longueur totale : 69 octets - Identification : 0xa0f4 (41204) - Drapeaux : 0x0 - Décalage de fragment : 0 - Durée de vie (TTL) : 128 - Protocole : UDP (17) - Somme de contrôle d'en-tête : 0xdfde [validation désactivée] - Adresse source : 192.168.28.2 - Adresse de destination : 192.168.28.130

La partie inférieure de la capture montre également la représentation hexadécimale du paquet, où l'on peut voir le contenu des données UDP. Contrairement à TCP, UDP n'établit pas de connexion et ne garantit pas la livraison des paquets, ce qui le rend plus léger et plus rapide, mais moins fiable.

Analyse détaillée d'un paquet TCP

Pour comprendre plus en profondeur le fonctionnement du protocole TCP, examinons en détail un paquet SYN capturé :



Cette capture montre un paquet SYN (le premier paquet du three-way handshake TCP) avec les détails suivants : - Trame 7 : 74 octets sur le fil (592 bits), 74 octets capturés (592 bits) - Ethernet II : adresses MAC source et destination - Internet Protocol Version 4 : adresses IP source (192.168.28.130) et destination (10.10.0.1) - Version : 4 - Longueur d'en-tête : 20 octets (5 mots de 32 bits) - Champ de services différenciés : 0x00 (DSCP: CS0, ECN: Not-ECT) - Longueur totale : 60 octets - Identification : 0x779d (30621) - Drapeaux : 0x2, Don't fragment - Décalage de fragment : 0 - Durée de vie : 64

La partie inférieure de la capture montre la représentation hexadécimale du paquet, où l'on peut identifier les différents champs de l'en-tête TCP. Par exemple, les octets "40 00" (surlignés en bleu) dans la ligne 0010 correspondent au champ de contrôle TCP indiquant qu'il s'agit d'un paquet SYN.

Mécanisme de connexion TCP

Le protocole TCP établit une connexion fiable entre deux hôtes à travers un processus appelé "three-way handshake" (poignée de main à trois temps). Ce processus se déroule en trois étapes :

1. **SYN** : L'hôte initiateur envoie un paquet avec le drapeau SYN activé et un numéro de séquence initial (ISN) aléatoire.
2. **SYN-ACK** : L'hôte récepteur répond avec un paquet ayant les drapeaux SYN et ACK activés, son propre ISN, et un numéro d'acquittement égal à l'ISN de l'initiateur + 1.
3. **ACK** : L'hôte initiateur envoie un paquet avec le drapeau ACK activé et un numéro d'acquittement égal à l'ISN du récepteur + 1.

Une fois ces trois étapes accomplies, la connexion est établie et les deux hôtes peuvent commencer à échanger des données.

La terminaison d'une connexion TCP se fait normalement par un processus similaire appelé "four-way handshake" (poignée de main à quatre temps), utilisant des paquets avec le drapeau FIN. Cependant, dans certains cas, comme on peut le voir dans la capture, une connexion peut être terminée abruptement avec un paquet RST.

Utilisation des filtres dans Wireshark

Wireshark offre des capacités de filtrage puissantes qui permettent d'isoler les paquets pertinents dans une capture volumineuse. Ces filtres peuvent être appliqués soit pendant la capture (filtres de capture), soit après la capture (filtres d'affichage).

Dans les captures fournies, on peut voir un filtre d'affichage "tcp" appliqué, qui limite l'affichage aux seuls paquets TCP. Ce type de filtre est particulièrement utile lorsqu'on cherche à analyser un protocole spécifique dans un trafic réseau dense.

Voici quelques exemples de filtres d'affichage courants : - `ip.addr == 192.168.1.1` : Affiche uniquement les paquets ayant 192.168.1.1 comme adresse IP source ou destination - `tcp.port == 80` : Affiche uniquement les paquets TCP utilisant le port 80 (HTTP) - `http` : Affiche uniquement les paquets HTTP - `arp` : Affiche uniquement les paquets ARP - `udp.port == 53` : Affiche uniquement les paquets UDP utilisant le port 53 (DNS)

Ces filtres peuvent être combinés avec des opérateurs logiques comme `and`, `or` et `not` pour créer des expressions de filtrage complexes. Par exemple, `ip.src == 192.168.1.1 and tcp.port == 80` afficherait uniquement les paquets HTTP envoyés depuis l'adresse IP 192.168.1.1.

L'utilisation efficace des filtres est essentielle pour naviguer dans des captures volumineuses et identifier rapidement les paquets d'intérêt.

Conclusion

Wireshark est un outil puissant pour l'analyse du trafic réseau, offrant une visibilité détaillée sur les communications entre les systèmes informatiques. En comprenant les différences entre les trames et les paquets, en maîtrisant les formats de capture comme PCAP/PCAPNG, et en sachant comment analyser les protocoles comme TCP, UDP et ARP, on peut diagnostiquer efficacement les problèmes réseau et approfondir sa compréhension des mécanismes de communication sous-jacents.

L'utilisation des filtres de Wireshark permet de se concentrer sur les aspects spécifiques du trafic réseau qui sont pertinents pour une analyse donnée, transformant ce qui pourrait être une tâche écrasante en un processus gérable et ciblé.

Cette première partie de la documentation a couvert les bases de l'utilisation de Wireshark et de l'analyse des protocoles réseau fondamentaux. Les parties suivantes exploreront des protocoles plus spécifiques et des techniques d'analyse plus avancées, notamment l'utilisation de tshark pour l'automatisation des captures et des analyses.