



Projet reseaux IP

spécialite: Reseaux infomatique et telecommunication

Fragmentation des packets IPV4 et IPV6

présenté par :

Yasmine Ferjani

Emna Drihem

Enseignat : Mr. Kamel Karoui

Année universitaire: 2024-2025

Résumé

Ce projet porte sur la technique de fragmentation des paquets IP dans les réseaux, en mettant l'accent sur l'analyse des comportements des protocoles IPv4 et IPv6, et en soulignant les différences entre ces deux protocoles. Des expérimentations ont été menées à l'aide d'outils de simulation et d'émulation tels que GNS3, complétées par des outils d'analyse comme Wireshark, afin d'étudier le comportement des paquets fragmentés et d'évaluer l'efficacité de chaque protocole dans des scénarios variés. Ces observations permettent de mieux comprendre les avantages et les limites de la fragmentation dans IPv4 et IPv6, tout en fournissant des pistes pour optimiser les performances des réseaux en fonction du protocole utilisé.

Contents

| | |
|---------------------------------------------------------------|-----------|
| Résume | 2 |
| 0.1 Introduction | 1 |
| Chapter 1: Les logiciels utilisés | 2 |
| 1.1 GNS3 | 2 |
| 1.1.1 Introduction au UI de GNS3 | 3 |
| 1.1.2 Fonctionnalités clés de GNS3 | 3 |
| 1.2 Wireshark | 5 |
| 1.2.1 Introduction au UI de Wireshark | 5 |
| 1.2.2 Fonctionnalités clés | 6 |
| 1.3 FTP | 7 |
| 1.3.1 Introduction à l'interface utilisateur de FTP | 7 |
| 1.3.2 Fonctionnalités clés de FTP | 8 |
| Chapter 2: Etude théorique de la fragmentation | 9 |
| 2.1 La fragmentation IPV4 | 10 |
| 2.1.1 Le processus de réassemblage | 11 |
| 2.1.2 Limites et défis de la fragmentation IPv4 | 11 |
| 2.2 La fragmentation IPV6 | 12 |
| 2.2.1 Le processus de réassemblage | 14 |
| 2.2.2 Limites et défis de la fragmentation IPv6 | 14 |
| 2.2.3 Découverte de la MTU | 14 |
| Chapter 3: Partie Pratique | 15 |
| 3.1 Configuration des équipements dans GNS3 | 15 |
| 3.1.1 Accéder aux paramètres de GNS3 | 15 |
| 3.1.2 Ajout des machines virtuelles VMware | 15 |
| 3.1.3 Ajout d'un routeur Cisco IOS | 16 |
| 3.1.4 Vérification des équipements ajoutés | 17 |
| 3.1.5 Conclusion | 17 |
| 3.2 Topologie | 18 |
| 3.2.1 Routeurs Cisco | 18 |
| 3.2.2 Switchs réseau | 18 |
| 3.2.3 Machines virtuelles | 18 |
| 3.2.4 Interconnexions | 19 |

| | | |
|-------|----------------------------------------------------------------|----|
| 3.2.5 | Sous-réseaux configurés | 19 |
| 3.3 | Configuration de la Topologie | 20 |
| 3.3.1 | Configuration du routeur R1 | 20 |
| 3.3.2 | Sauvegarde de la configuration sur le routeur R1 | 22 |
| 3.3.3 | Configuration de la machine virtuelle KALI1 | 22 |
| 3.3.4 | Configuration de R2 et KALI2 | 24 |
| 3.4 | Verification de la configuration | 25 |
| 3.5 | Fragmentation | 26 |
| 3.5.1 | Fragmentation IPv4 | 26 |
| 3.5.2 | Sauvegarde de la configuration sur le routeur R1 | 27 |
| 3.5.3 | Test de ping pour IPv6 | 29 |
| 3.5.4 | Comparaison des résultats IPv4 et IPv6 | 30 |
| 3.6 | Configuration du serveur FTP et transfert de fichier | 31 |
| 3.6.1 | Configuration du serveur FTP | 31 |
| 3.6.2 | Transfert du fichier de KALI1 vers KALI2 | 33 |
| 3.6.3 | Analyse de la capture Wireshark | 34 |
| 3.6.4 | Analyse de la capture Wireshark | 35 |
| 3.7 | Bibliographie | 37 |

0.1 Introduction

Depuis la conception des réseaux informatiques, le protocole IP (Internet Protocol) a été essentiel pour le routage et l'adressage des paquets de données, permettant leur acheminement à travers divers réseaux. Cependant, la conception initiale d'IPv4 n'a pas anticipé l'expansion rapide des réseaux et de leurs besoins, révélant ainsi des limitations, notamment au niveau du nombre d'adresses disponibles. Pour répondre à ces défis, le protocole IPv6 a été développé. Bien que les deux protocoles partagent des principes fondamentaux similaires, ils présentent des différences notables, notamment dans leur gestion de la fragmentation des paquets.

Les premiers réseaux se heurtaient à des limitations de capacité, chaque technologie ayant une taille maximale de paquet définie, connue sous le nom de **MTU** (Maximum Transmission Unit). La fragmentation a donc été introduite pour diviser un datagramme IP en morceaux plus petits, facilitant leur transmission à travers des réseaux dont les tailles de trames sont limitées.

Ce projet a pour objectif d'analyser les différences dans la gestion de la fragmentation entre IPv4 et IPv6 en utilisant des outils de capture et d'analyse de réseau, tels que Wireshark et GNS3, et de comparer l'efficacité et les impacts de la fragmentation dans chaque protocole. En menant plusieurs expérimentations, nous visons à mettre en évidence les implications pratiques de la fragmentation et à tirer des conclusions sur les avantages et les inconvénients des deux approches.

Chapter 1

Les logiciels utilisés

Pour mener à bien cette étude sur la fragmentation des protocoles IPv4 et IPv6, plusieurs logiciels spécialisés seront utilisés pour la configuration, l'émulation et l'analyse des réseaux.

1.1 GNS3

GNS3 (Graphical Network Simulator 3) est un logiciel puissant pour l'émulation, la configuration, le test et le dépannage de réseaux, qu'ils soient réels ou virtuels. Il permet de créer et d'exécuter des topologies de différentes tailles, allant de petites configurations locales à des infrastructures plus complexes hébergées dans le cloud. En outre, GNS3 prend en charge un large éventail d'appareils provenant de divers fournisseurs, offrant ainsi une flexibilité maximale dans la création de réseaux.

Avec GNS3, chaque périphérique de votre topologie peut être hébergé soit localement sur votre ordinateur (serveur local GNS3), dans une machine virtuelle (VM locale GNS3) ou encore dans le cloud (serveur distant GNS3 cloud). Le logiciel prend en charge à la fois l'émulation et la simulation d'appareils, chacune ayant son propre usage :

- **Emulation:** GNS3 reproduit le matériel d'un appareil, permettant ainsi l'exécution de vraies images système sur l'appareil virtuel.
- **Simulation:** GNS3 reproduit les fonctionnalités d'un appareil, sans exécuter de système d'exploitation réel. Il utilise plutôt un appareil simulé développé par GNS3 pour offrir les principales fonctionnalités du matériel.

En somme, GNS3 est un outil polyvalent qui permet aux ingénieurs réseau de concevoir, tester et dépanner des topologies de réseau complexes de manière flexible et évolutive.

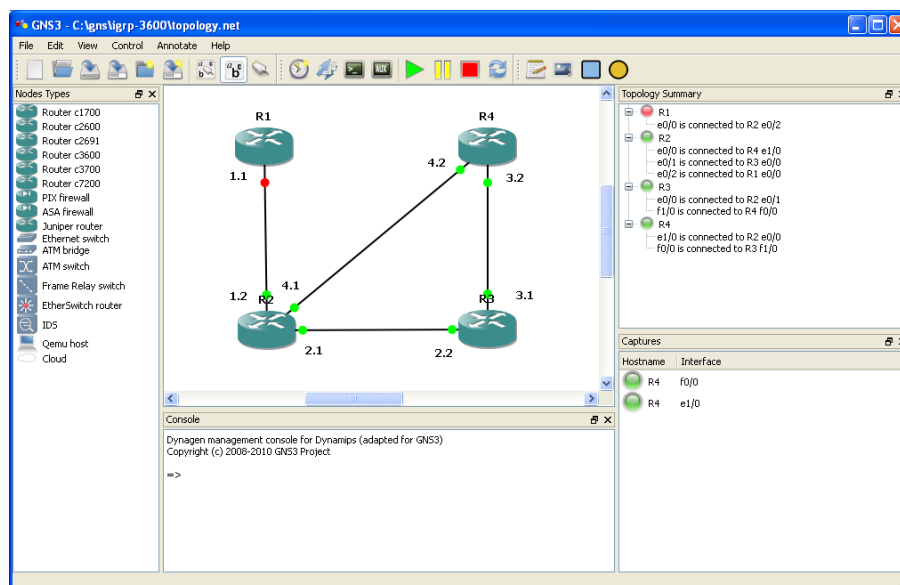


Figure 1.1: Fenêtre principale de GNS3

1.1.1 Introduction au UI de GNS3

La fenêtre principale de GNS3 est divisée en plusieurs volets, chacun ayant une fonction spécifique :

- **Barre de menu** : fournit un accès aux options principales telles que l'importation/exportation de projets, la configuration des préférences de l'interface et les outils de gestion des appareils.
- **Barre d'outils** : donne un accès rapide aux actions courantes comme ajouter de nouveaux appareils, démarrer/arrêter les simulations et gérer les instantanés de projet.
- **Volet de projet** : affiche la liste des appareils ajoutés au projet actuel, ainsi que leur état et leur configuration. Vous pouvez voir les connexions entre les appareils ici.
- **Volet de console** : vous permet d'accéder à la console des appareils, où vous pouvez entrer des commandes et interagir avec le réseau simulé en temps réel.
- **Zone de travail** : espace central où les appareils sont mis en place et connectés. Vous construisez et visualisez le réseau simulé ici.

1.1.2 Fonctionnalités clés de GNS3

- **Simulation de réseau**: GNS3 peut être utilisé pour créer des réseaux et les simuler virtuellement. Cela implique des appareils virtuels, y compris des routeurs, des commutateurs, des pare-feu et des appareils physiques. Les amateurs peuvent utiliser GNS3 pour tester et déployer des configurations réseau dans un environnement virtuel avant d'expérimenter dans un environnement en direct.

- **Intégration multi-fournisseurs:** GNS3 prend en charge plus d'images d'appareils que n'importe quel autre émulateur ou logiciel de routeur. En tant que tel, GNS3 prend en charge les images cisco, junper, mikrotik et bien d'autres. Le logiciel prend également en charge des entreprises informatiques réputées et respectées pour leurs appareils.
- **Topologie dynamique:** GNS3 fournit aux utilisateurs un espace de travail visuel dans lequel les utilisateurs peuvent déposer des appareils, les configurer et les connecter à l'aide de pointillés. Les utilisateurs peuvent supprimer ou ajouter des appareils à mesure que la topologie de leur réseau évolue.
- **Support pour les scripts et les API :** GNS3 permet les scripts et APIs pour automatiser la configuration et la gestion programme des devices, en autorisant l'automatisation et le déploiement de tests à grande échelle.
- **Collaboration et partage de projets :** Les projets GNS3 peuvent être sauvegardés, exportés, partagés, favorisant la collaboration et la réplication des configurations réseau.

1.2 Wireshark

L'application open source Wireshark, le plus utilisé des analyseurs de paquets au monde, permet d'analyser les paquets transitant sur un réseau donné, en fournissant le maximum de détails. Elle a trois fonctionnalités de premier plan :

- **Capture de paquets:** Wireshark capture en temps réel des flux intégralement sur une connexion réseau particulière écoute une connexion réseau en temps réel et capture des flux entiers de trafic.
- **Filtrage:** Wireshark permet de trier – et d'analyser les données capturées- ces données en temps réel grâce à des filtres. En appliquant un filtre, vous pouvez extraire uniquement les informations pertinentes que vous souhaitez examiner.
- **Visualisation:** Wireshark, eEn tant qu'"analyseur de paquets efficace,performant, Wire-shark vous permet d'"explorer en détailprofondeur le contenu de chaque paquet réseau, mais per. Il offre également aussila possibilité de visualiser l'"intégralité des conversations et des flux réseau.

Cette application est utilisée à diverses fins, notamment pour le dépannage de réseaux àaux performances défaillantes ou pendant des activités de cybersécurité, pour retrouv, ainsi que dans le domaine de la cybersécurité, où elle est employée pour tracer des connexions, examiner le contenu des transactions de réseaux jugées suspectes, relev et identifier des pics de trafic, etc. Wireshark estconstitue ainsi un outil essentielincontournable pour les ses professionnels de l'"informatique et de la sécurité.

1.2.1 Introduction au UI de Wireshark

La fenêtre principale de Wireshark est divisée en plusieurs volets, chacun ayant une fonction spécifique :

- **Barre de menu:** Offre diverses options pour les opérations de fichiers, les paramètres de capture, les outils d'analyse, et plus encore.
- **Barre d'outils:** Fournit un accès rapide aux actions courantes comme démarrer/arrêter la capture, filtrer les paquets, et zoomer avant/arrière.
- **Volet de la liste des paquets:** Affiche une liste des paquets capturés, incluant leurs horodatages, les adresses source et destination, le protocole, et un résumé de leur contenu.
- **Volet des détails des paquets:** Fournit des informations détaillées sur le paquet sélectionné, y compris ses couches de protocoles, les champs d'en-tête, et les données de charge utile.
- **Volet des octets de paquets:** Affiche les octets bruts du paquet sélectionné, permettant une analyse à bas niveau.

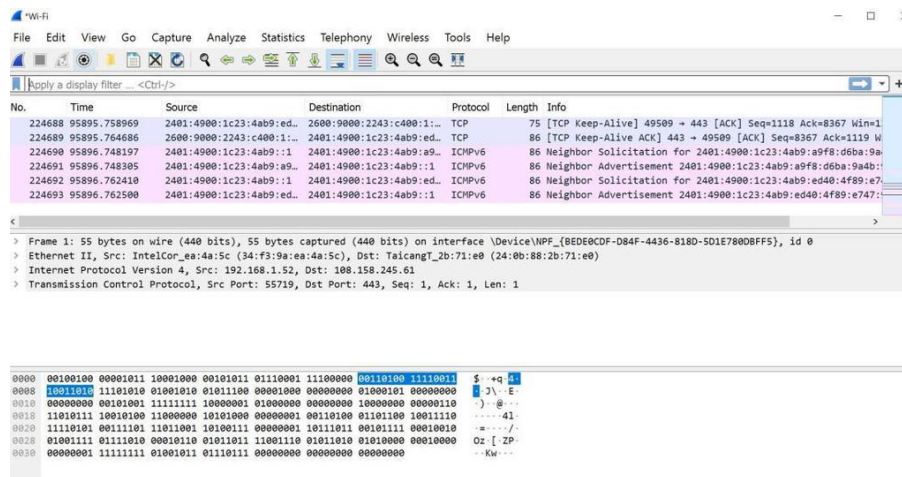


Figure 1.2: Fenêtre principale de Wireshark

1.2.2 Fonctionnalités clés

- **Filtrage de paquets:** Wireshark permet de filtrer les paquets en fonction de divers critères, tels que les adresses IP source/destination, les protocoles, les numéros de port, et plus encore. Cela vous aide à vous concentrer sur le trafic spécifique qui vous intéresse.
- **Analyse de protocoles :** Wireshark prend en charge une large gamme de protocoles, y compris TCP, UDP, HTTP, FTP, et bien d'autres. Il peut disséquer ces protocoles pour révéler leur structure sous-jacente et leurs données.
- **Analyse temporelle:** Wireshark fournit des outils pour analyser le trafic réseau au fil du temps, tels que des affichages de chronologie et des graphiques statistiques.
- **Exportation et importation:** Vous pouvez exporter les paquets capturés vers divers formats (par exemple, PCAP, CSV) et les importer pour une analyse ultérieure.

1.3 FTP

Le protocole FTP (File Transfer Protocol) est un protocole standard utilisé pour transférer des fichiers entre un client et un serveur sur un réseau. Il est largement utilisé pour le partage de fichiers dans différents environnements réseau, notamment pour le développement web et la gestion de serveurs. FTP se distingue par trois fonctionnalités principales :

- **Transfert de fichiers:** FTP permet le transfert de fichiers entre un client et un serveur, en mode binaire ou en mode texte. Cela facilite la transmission de fichiers de manière fiable et rapide.
- **Contrôle d'accès:** FTP offre plusieurs niveaux de contrôle d'accès, permettant aux administrateurs de configurer les permissions pour chaque utilisateur. Cela assure la sécurité des données et limite les accès aux fichiers sensibles.
- **Reprise de transfert:** Si une connexion est interrompue lors d'un transfert de fichier, FTP permet de reprendre le transfert à partir de l'endroit où il s'était arrêté, réduisant ainsi le risque de perte de données.

FTP est couramment utilisé pour la gestion de sites web, où les fichiers peuvent être téléchargés ou modifiés directement sur le serveur, et pour les échanges de fichiers dans des entreprises. Il reste un outil essentiel pour les administrateurs de réseau, bien que d'autres protocoles sécurisés, comme SFTP et FTPS, soient de plus en plus adoptés pour renforcer la sécurité des transferts de fichiers.

1.3.1 Introduction à l'interface utilisateur de FTP

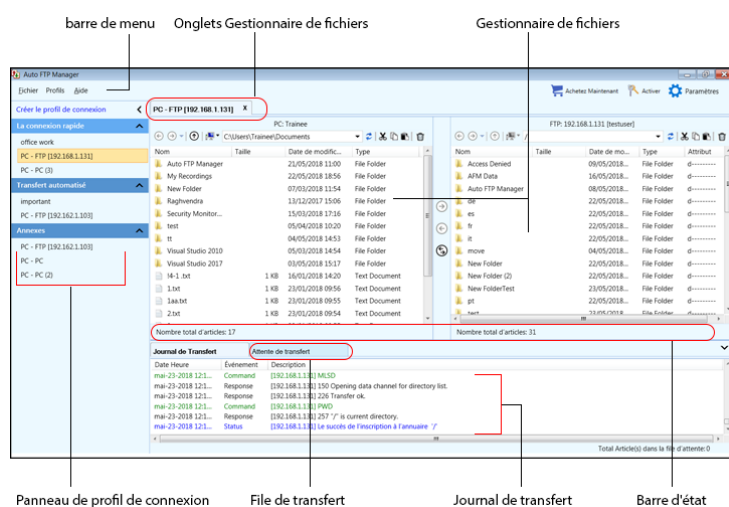


Figure 1.3: Fenêtre principale de FTP

L'interface utilisateur d'un client FTP est souvent divisée en plusieurs sections, facilitant l'organisation des transferts de fichiers :

- **Barre de menu:** Contient les options principales pour la configuration du client, la connexion au serveur et la gestion des transferts.
- **Barre d'outils:** Fournit un accès rapide aux actions telles que la connexion/déconnexion, la navigation dans les dossiers, et la gestion des transferts de fichiers.
- **Volet de l'explorateur de fichiers locaux:** Permet de parcourir et de sélectionner les fichiers locaux à transférer vers le serveur.
- **Volet de l'explorateur de fichiers distant:** Affiche la structure des fichiers et des dossiers sur le serveur distant, permettant de naviguer et de gérer les fichiers.
- **Volet de statut de transfert:** Affiche les informations sur les transferts en cours et terminés, avec des indicateurs de progression pour chaque fichier.

1.3.2 Fonctionnalités clés de FTP

- **Gestion des permissions:** FTP permet de définir des droits d'accès spécifiques pour chaque utilisateur, en contrôlant les actions autorisées sur le serveur (lecture, écriture, suppression).
- **Modes actif et passif:** FTP supporte les modes actif et passif, qui déterminent comment les connexions sont établies entre le client et le serveur, en fonction de la configuration des pare-feu et des réseaux.
- **Connexion sécurisée avec FTPS/SFTP:** Pour renforcer la sécurité, FTP peut être utilisé avec SSL (FTPS) ou en mode sécurisé avec SSH (SFTP), assurant la confidentialité et l'intégrité des données transférées.
- **Transferts automatisés:** De nombreux clients FTP permettent de planifier des transferts ou de créer des scripts automatisés, ce qui est utile pour les sauvegardes régulières ou les synchronisations de fichiers.

Chapter 2

Etude théorique de la fragmentation

Le paquet IP est conçu pour fonctionner sur différents types de réseaux physiques, chacun implémentant des protocoles de niveau 2 distincts. Cette diversité entraîne des contraintes notables, notamment en matière de MTU (Maximum Transmission Unit), qui représente la taille maximale de paquet de données acceptée par un appareil réseau. On peut comparer la MTU à une limite de hauteur sur une route : les véhicules surpassant cette limite ne peuvent pas traverser cette route. De même, les paquets plus grands que la MTU ne peuvent pas passer directement à travers le réseau et doivent donc être fragmentés (si cela est possible).

La fragmentation consiste à diviser un paquet IP en segments plus petits pour qu'ils puissent être transmis sur le réseau et réassemblés à leur destination. Ce processus est intégré dans les deux protocoles IPV4 et IPV6 chacun ayant montrant des spécificités différentes.

2.1 La fragmentation IPV4

Pour mieux comprendre la fragmentation avec le protocole IPV4, il est nécessaire d'examiner la structure du datagramme IPV4, dont le header est composé de plusieurs champs, comme illustré dans la figure ci-dessous.

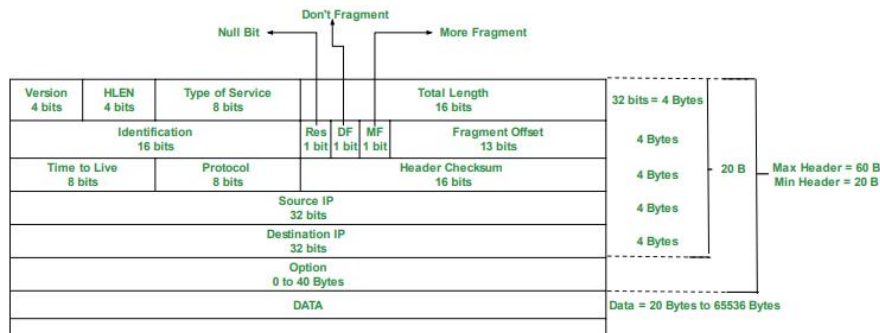


Figure 2.1: La structure du Header de l'IPV4

Pour la bonne compréhension du concept on doit comprendre les champs de base de l'IPV4.

- **Version:** Encodé sur 4 bits, ce champ indique si le header est de type IPv4 (0100) ou IPv6 (0110).
- **HLEN:** Champ de 4 bits spécifiant la taille du header. Sa valeur peut varier entre 5 et 15.
- **Longueur totale:** Champ de 16 bits qui indique la taille totale du paquet (header + données), allant de 20 à 65 535 octets.

Mais pour notre étude de la fragmentation, nous nous concentrerons particulièrement sur les champs situés entre le 5e et le 8e octet, spécifiques à la gestion de la fragmentation :

- **Flags:** Trois bits utilisés pour le contrôle de la fragmentation :
 - **R:** Bit réservé, toujours à 0.
 - **DF (Don't Fragment):** Si mis à 1, la fragmentation du paquet n'est pas autorisée.
 - **MF (More Fragments):** Indiqué à 1 pour tous les fragments sauf le dernier, signalant ainsi que d'autres fragments suivent.
- **Identification:** il s'agit de 16bits contenant l'identifiant unique de packet pour identifier le groupe de fragments d'un seul datagramme IP.
- **Fragment offset:** Il s'agit d'un champ de 13 bits et il représente le nombre d'octets de données précédant un fragment particulier dans le datagramme en question. Il est spécifié en termes de blocs de 8 octets, avec une valeur maximale de 65 528 octets.

Tous ces champs fonctionnent en harmonie pour que la fragmentation soit réussie. En effet, dans un réseaux, chaque routeur vérifie la taille de chaque paquet IP qu'il reçoit et la compare avec le MTU du prochain routeur auquel le paquet sera transmis. Si le paquet dépasse la MTU du prochain routeur et si la fragmentation est permise (càd que le bit DF est réglé sur 0), le premier routeur divise les données en des packets de plus petites tailles chacun avec ses propres en-têtes.

Les en-têtes des nouveaux paquets conservent les informations essentielles du paquet d'origine (telles que les adresses IP source et destination), mais avec des modifications spécifiques aux champs de fragmentation(indiqués précédemment). Ces ajustements indiquent que les paquets sont fragmentés et nécessitent un réassemblage, tout en précisant le nombre total de fragments ainsi que leur ordre de transmission.

2.1.1 Le processus de réassemblage

Le réassemblage des fragments se fait uniquement au niveau du destinataire final et non dans les routeurs intermédiaires. Le champ d'Identification permet au destinataire d'assembler les fragments appartenant au même datagramme, tandis que le champ *Fragment Offset* indique l'ordre de chaque fragment pour une reconstruction complète et correcte du paquet d'origine.

2.1.2 Limites et défis de la fragmentation IPv4

La fragmentation introduit certains défis, notamment une augmentation du délai de transmission et un risque accru de perte de paquets. En effet, si un seul fragment est perdu, l'ensemble du datagramme doit être renvoyé, ce qui peut réduire l'efficacité dans les réseaux peu fiables. De plus, la fragmentation et le réassemblage nécessitent une gestion supplémentaire, ce qui peut entraîner une surcharge et une latence accrue.

2.2 La fragmentation IPV6

Pour mieux comprendre la fragmentation avec le protocole IPv6, il est nécessaire d'examiner la structure du datagramme IPv6, dont le header est composé de plusieurs champs, comme illustré dans la figure ci-dessous.

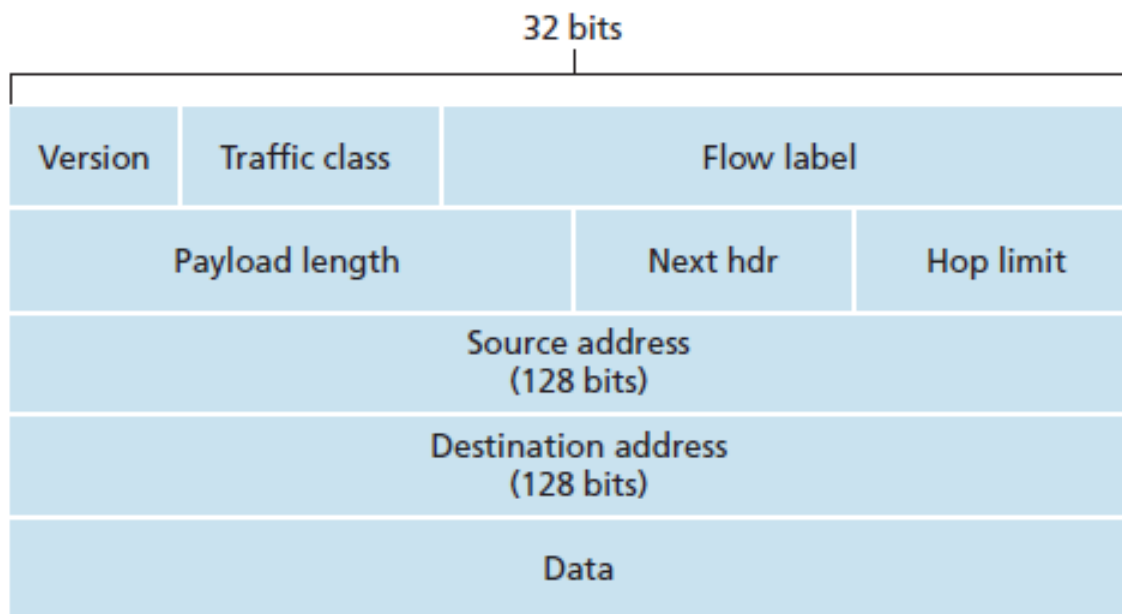


Figure 2.2: La structure du Header de l'IPv4

Pour la bonne compréhension du concept on doit comprendre les champs de base de l'IPv6.

- **Version:** Encodé sur 4 bits, ce champ indique la version du protocole IP, qui est fixée à 6 pour l'IPv6.
- **Classe de trafic:** Champ de 8 bits utilisé pour la classification et la priorité des paquets, afin de gérer la qualité de service (QoS).
- **Label de flux:** Champ de 20 bits utilisé pour identifier les paquets appartenant au même flux, permettant un traitement cohérent des paquets d'un même flux.
- **Longueur de la charge utile:** Champ de 16 bits indiquant la taille de la charge utile en octets, incluant les en-têtes d'extension éventuels et les données de l'application.
- **En-tête suivant:** Champ de 8 bits qui identifie le type de l'en-tête suivant. Il peut s'agir d'un protocole de couche supérieure (par exemple TCP ou UDP) ou d'une en-tête d'extension IPv6.

- **Limite de sauts (Hop Limit):** Champ de 8 bits spécifiant le nombre maximal de sauts que le paquet peut traverser. Ce champ est décrémenté à chaque routeur traversé et, s'il atteint zéro, le paquet est supprimé.
- **Adresse source:** Champ de 128 bits contenant l'adresse IPv6 de l'expéditeur du paquet.
- **Adresse de destination:** Champ de 128 bits contenant l'adresse IPv6 du destinataire du paquet.

Les en-têtes d'extension IPv6 fournissent des fonctionnalités supplémentaires. Voici quelques exemples :

- **En-tête optionnel de saut en saut:** Contient les informations à traiter par chaque routeur le long du chemin du paquet.
- **En-tête de routage:** Permet de spécifier une ou plusieurs adresses de nœuds intermédiaires que le paquet doit traverser avant d'atteindre sa destination finale.
- **En-tête de fragmentation:** Utilisé pour gérer la fragmentation du paquet. Dans IPv6, la fragmentation est réalisée uniquement par l'hôte source et non par les routeurs.
- **En-tête d'options de destination:** Contient des options qui ne doivent être traitées que par le ou les destinataires finaux du paquet.
- **En-têtes d'authentification et ESP (Encapsulating Security Payload):** Utilisés pour la sécurité IPsec, offrant l'intégrité, l'authenticité et la confidentialité des paquets IPv6.

La charge utile contient les données de la couche supérieure (par exemple, TCP, UDP ou les données de l'application) que le paquet IPv6 est censé transporter.

Tous ces champs fonctionnent en harmonie pour garantir un transfert efficace des données. Contrairement à IPv4, la fragmentation dans IPv6 est effectuée uniquement par l'hôte source et non par les routeurs intermédiaires. Cela signifie que chaque hôte source doit vérifier la taille de chaque paquet IPv6 par rapport au MTU (Maximum Transmission Unit) du réseau de destination avant l'envoi. Si un paquet dépasse la MTU du réseau de destination, l'hôte source doit diviser le paquet en fragments de plus petite taille, chacun avec sa propre en-tête IPv6.

Les en-têtes des fragments conservent les informations essentielles du paquet d'origine (telles que les adresses IP source et destination), tout en incluant des modifications spécifiques dans l'en-tête de fragmentation. Ces ajustements indiquent que les paquets sont fragmentés et nécessitent un réassemblage au niveau du destinataire final, en précisant le nombre total de fragments ainsi que leur ordre de transmission pour une reconstitution correcte du paquet d'origine.

2.2.1 Le processus de réassemblage

Le réassemblage des fragments dans IPv6 se fait uniquement au niveau du destinataire final, et non dans les routeurs intermédiaires. L'en-tête de fragmentation, ajouté par l'hôte source, contient un champ d'Identification permettant au destinataire d'identifier les fragments appartenant au même paquet. Le champ *Fragment Offset* précise l'ordre de chaque fragment, ce qui permet au destinataire de reconstruire le paquet d'origine de manière complète et correcte.

2.2.2 Limites et défis de la fragmentation IPv6

La fragmentation dans IPv6 présente certains défis, notamment une augmentation potentielle du délai de transmission et un risque accru de perte de fragments. En effet, si un seul fragment est perdu, l'ensemble du paquet doit être renvoyé, ce qui peut réduire l'efficacité dans les réseaux moins fiables. De plus, la fragmentation et le réassemblage nécessitent une gestion supplémentaire au niveau du destinataire, ce qui peut entraîner une surcharge et une latence accrue dans le traitement des paquets.

2.2.3 Découverte de la MTU

En IPv6, la découverte de la MTU de chemin (Path Maximum Transmission Unit, ou PMTU) est un processus crucial qui permet de déterminer la taille maximale de l'unité de transmission sur le chemin entre une source et une destination. Cela garantit que les paquets IPv6 peuvent être transmis sans nécessiter de fragmentation le long du chemin.

Contrairement à IPv4, IPv6 n'autorise pas les routeurs à fragmenter les paquets. Au lieu de cela, l'hôte émetteur est responsable de l'ajustement de la taille du paquet en fonction de la MTU du chemin réseau, qu'il détermine grâce à la découverte de la PMTU. Ce processus consiste à envoyer des paquets de test de différentes tailles et à recevoir un retour d'information jusqu'à ce que la plus grande taille possible pouvant être transmise sans fragmentation soit trouvée.

L'objectif de la découverte de la PMTU en IPv6 est d'optimiser l'efficacité de la transmission et de réduire la latence en évitant les fragmentations inutiles, rendant ainsi le réseau plus efficace et résilient.

Chapter 3

Partie Pratique

3.1 Configuration des équipements dans GNS3

Dans cette section, nous détaillons le processus de configuration des équipements réseau dans **GNS3**. Cela inclut l'ajout de machines virtuelles **Kali Linux** et d'un routeur **Cisco IOS** pour construire notre topologie.

3.1.1 Accéder aux paramètres de GNS3

Pour commencer, nous accédons aux préférences de GNS3 afin d'ajouter les équipements nécessaires. Comme indiqué dans la **Figure 3.1**, naviguez dans le menu **Edit > Preferences** ou utilisez le raccourci **Ctrl+Shift+P**.

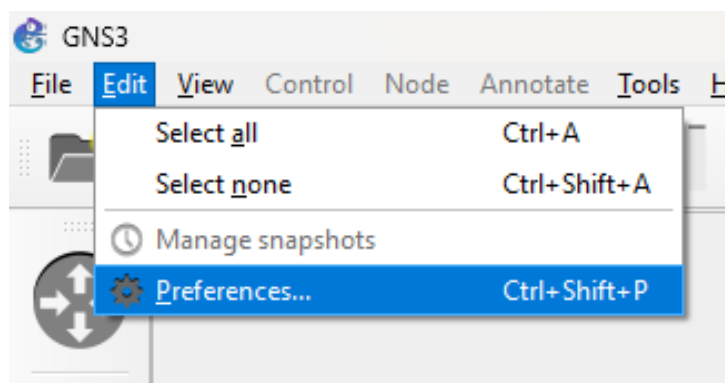


Figure 3.1: Accès aux paramètres de GNS3

3.1.2 Ajout des machines virtuelles VMware

Dans les paramètres de GNS3, nous configurons les machines virtuelles VMware. Les machines Kali Linux apparaissent dans la section **VMware VMs** (Figure 3.2).

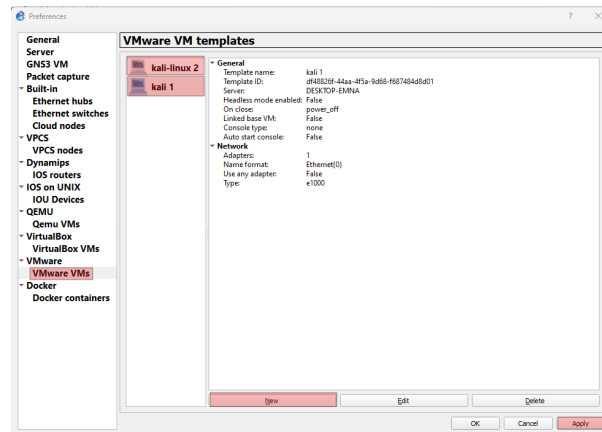


Figure 3.2: Ajout des machines virtuelles VMware (Kali Linux)

Voici les principales configurations observées :

- Nom des machines : kali 1 et kali-linux 2.
- Type d'adaptateur réseau : e1000.
- Configuration générale et réseau indiquée à droite.

3.1.3 Ajout d'un routeur Cisco IOS

Ensuite, nous ajoutons un routeur Cisco dans GNS3. La **Figure 3.3** montre la section IOS routers, où un routeur c3725 est configuré.

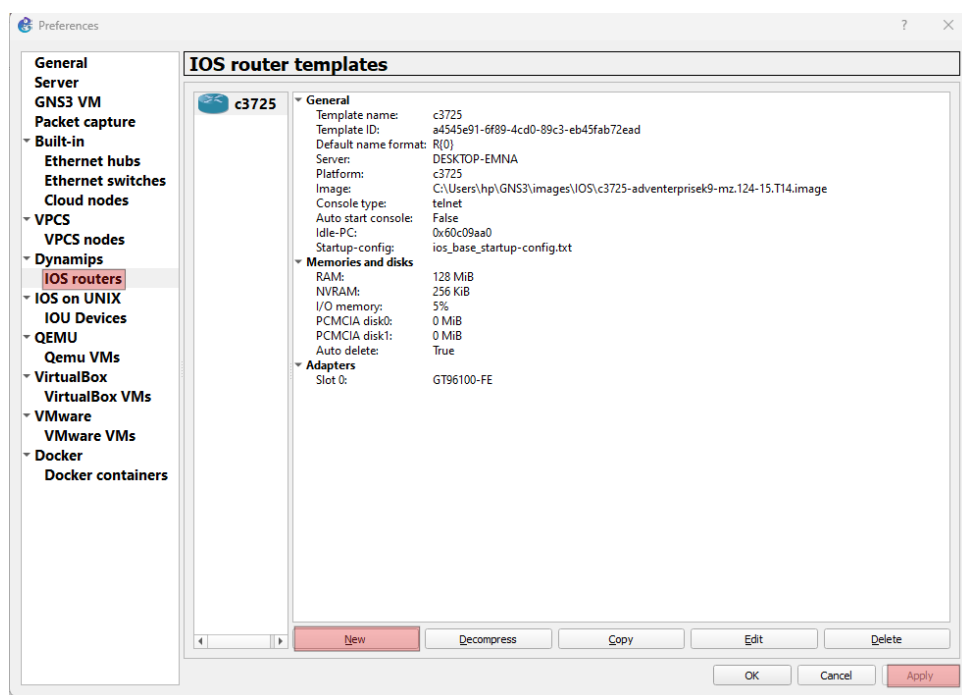


Figure 3.3: Ajout d'un routeur Cisco IOS

Les principales configurations du routeur incluent :

- Nom du routeur : c3725.
- Image IOS utilisée : c3725-adventerprisek9-mz.124-15.T14.
- Mémoire RAM : 128 MiB.
- Console type : Telnet.

3.1.4 Vérification des équipements ajoutés

Une fois les machines virtuelles et les routeurs ajoutés, ils apparaissent dans la barre d'outils des équipements disponibles (Figure 3.4). Cela confirme que les configurations sont correctement effectuées.

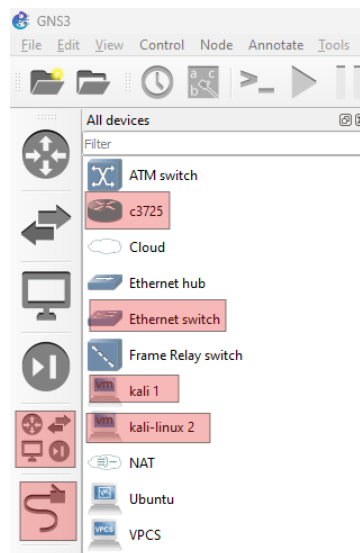


Figure 3.4: Vérification des équipements ajoutés dans GNS3

Les équipements ajoutés comprennent :

- Machines virtuelles : kali 1, kali-linux 2.
- Routeur : c3725.
- Commutateur Ethernet et autres périphériques réseau.

À cette étape, les équipements sont prêts à être utilisés dans la topologie réseau pour les expérimentations.

3.1.5 Conclusion

Cette configuration initiale de GNS3 permet l'intégration des machines Kali Linux et des routeurs Cisco nécessaires pour la mise en place de notre topologie réseau. Les étapes suivantes consisteront à configurer les interfaces réseau et tester la connectivité entre les équipements.

3.2 Topologie

La topologie réseau ci-dessous est conçue pour tester et analyser la fragmentation des paquets IPv4 et IPv6 dans un environnement simulé. Elle est composée des éléments suivants :

3.2.1 Routeurs Cisco

Deux routeurs Cisco **R1** et **R2** (modèle c3725).

Chaque routeur est configuré avec des interfaces IPv4 et IPv6 pour permettre la communication entre les machines virtuelles.

L'interface f0/1 de chaque routeur est connectée à un **Switch3**, permettant une liaison réseau avec :

- Sous-réseau IPv4 : 10.0.1.x/24.
- Sous-réseau IPv6 : 2001:db8:1:1::/64.

3.2.2 Switchs réseau

Trois switchs sont utilisés pour interconnecter les périphériques de la topologie :

- **Switch1** : situé à gauche, il connecte le routeur **R2** à la machine virtuelle **KALI2**.
- **Switch2** : situé à droite, il connecte le routeur **R1** à la machine virtuelle **KALI1**.
- **Switch3** : situé au centre, il relie les routeurs **R1** et **R2**.

3.2.3 Machines virtuelles

Deux machines virtuelles sont connectées aux extrémités de la topologie :

- **VM KALI2** :
 - Connectée au **Switch1** et au routeur **R2**.
 - Adresses IP :
 - * IPv4 : 192.168.0.2.
 - * IPv6 : 2001:db8:1:2::2.
- **VM KALI1** :
 - Connectée au **Switch2** et au routeur **R1**.
 - Adresses IP :
 - * IPv4 : 172.168.0.2.
 - * IPv6 : 2001:db8:1:3::2.

3.2.4 Interconnexions

Les interconnexions sont établies de la manière suivante :

- Les routeurs **R1** et **R2** sont connectés via le **Switch3**.
- Chaque routeur utilise son interface f0/0 pour la connexion aux machines virtuelles respectives à travers les switchs **Switch1** et **Switch2**.

3.2.5 Sous-réseaux configurés

- Sous-réseau IPv4 :
 - 10.0.1.x/24 : pour la liaison inter-routeurs via le **Switch3**.
 - 192.168.0.x/24 : pour la connexion entre **R2** et **VM KALI2**.
 - 172.168.0.x/24 : pour la connexion entre **R1** et **VM KALI1**.
- Sous-réseau IPv6 :
 - 2001:db8:1:1::/64 : pour la liaison inter-routeurs via le **Switch3**.
 - 2001:db8:1:2::/64 : pour la connexion entre **R2** et **VM KALI2**.
 - 2001:db8:1:3::/64 : pour la connexion entre **R1** et **VM KALI1**.

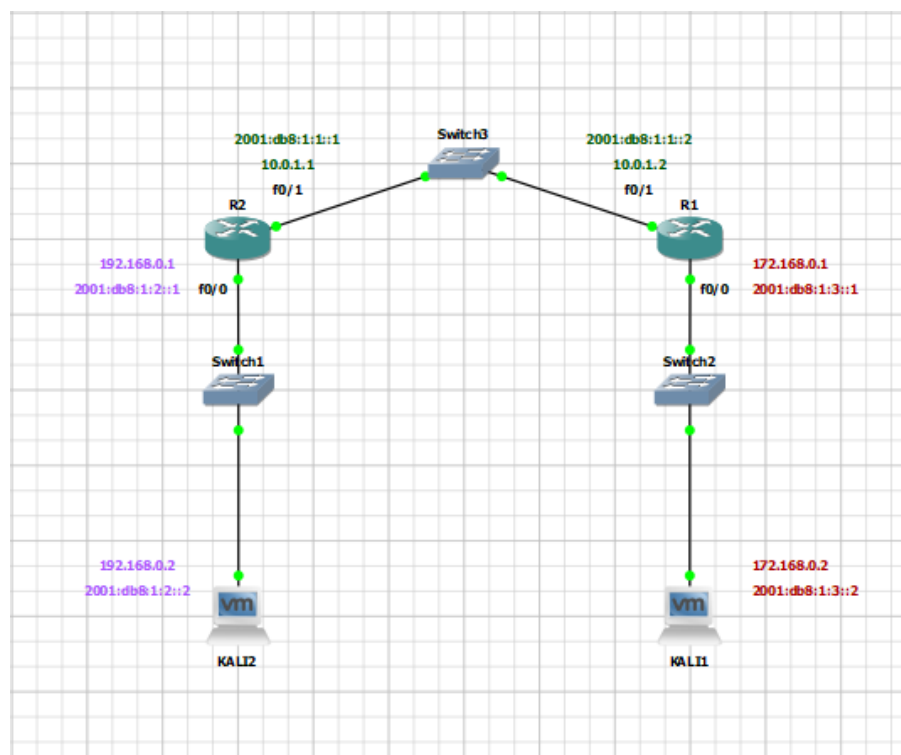


Figure 3.5: Schéma de la topologie pour l'analyse de la fragmentation

3.3 Configuration de la Topologie

Dans cette section, nous allons présenter la configuration des périphériques **R1** et **VM KALI1** pour les protocoles IPv4 et IPv6. La même configuration sera appliquée à **R2** et **VM KALI1** avec leurs adresses IP respectives.

3.3.1 Configuration du routeur R1

Le routeur **R1** est configuré avec une adresse IPv4 et une adresse IPv6 sur ses interfaces. Voici les commandes utilisées :

Listing 3.1: Configuration du routeur R1

```
R1> enable
R1# configure terminal
R1(config)# int f0/0
R1(config-if)# ip address 172.168.0.1 255.255.255.0
R1(config-if)# ipv6 address 2001:db8:1:3::1/64
R1(config-if)# no shutdown
R1(config-if)# exit
R1(config)# int f0/1
R1(config-if)# ip address 10.0.1.1 255.255.255.0
R1(config-if)# ipv6 address 2001:db8:1:1::1/64
R1(config-if)# no shutdown
R1(config-if)# exit
R1(config)# ipv6 unicast-routing
R1(config)# exit
R1# copy running-config startup-config
```

Vérification de la configuration Les commandes suivantes permettent de vérifier l'état des interfaces et les routes configurées sur le routeur R1.

- **Affichage des interfaces** : `show ip interface brief`
- **Table de routage IPv4** : `show ip route`
- **Table de routage IPv6** : `show ipv6 interface brief`
- **Table de routage IPv6** : `show ipv6 route`


```

R1#show ip int br
Interface                               IP-Address      OK? Method Status      Protocol
FastEthernet0/0                         172.168.0.1     YES NVRAM   up          up
FastEthernet0/1                         10.0.1.2        YES NVRAM   up          up

```

Figure 3.6: show ip interface brief

```

      172.168.0.0/24 is subnetted, 1 subnets
C       172.168.0.0 is directly connected, FastEthernet0/0
      10.0.0.0/24 is subnetted, 1 subnets
C       10.0.1.0 is directly connected, FastEthernet0/1
S       192.168.0.0/24 [1/0] via 10.0.1.1

```

Figure 3.7: show ip route

```

R1#show ipv6 int br
FastEthernet0/0                        [up/up]
    FE80::C001:21FF:FEF0:0
    2001:DB8:1:3::1
FastEthernet0/1                        [up/up]
    FE80::C001:21FF:FEF0:1
    2001:DB8:1:1::2

```

Figure 3.8: show ipv6 interface brief

```

C    2001:DB8:1:1::/64 [0/0]
    via ::, FastEthernet0/1
L    2001:DB8:1:1::2/128 [0/0]
    via ::, FastEthernet0/1
S    2001:DB8:1:2::/64 [1/0]
    via 2001:DB8:1:1::1
C    2001:DB8:1:3::/64 [0/0]
    via ::, FastEthernet0/0
L    2001:DB8:1:3::1/128 [0/0]
    via ::, FastEthernet0/0
L    FF00::/8 [0/0]
    via ::, Null0

```

Figure 3.9: show ipv6 route

3.3.2 Sauvegarde de la configuration sur le routeur R1

```
R1#copy run star
Destination filename [startup-config]?
Building configuration...
[OK]
```

Figure 3.10: Sauvegarde de la configuration sur le routeur R1

La capture ci-dessus montre la commande utilisée pour sauvegarder la configuration courante (running-config) dans la configuration de démarrage (startup-config) sur un routeur Cisco.

- **copy run start** : Cette commande copie la configuration active (en mémoire vive) vers la mémoire NVRAM pour assurer sa persistance après un redémarrage.
- **Destination filename** : Par défaut, le fichier est sauvegardé sous le nom `startup-config`.
- **Confirmation de sauvegarde** : L'indication `Building configuration... [OK]` confirme que la configuration a été sauvegardée avec succès.

Cette étape est essentielle pour éviter la perte des modifications de configuration en cas de redémarrage du routeur.

3.3.3 Configuration de la machine virtuelle KALI1

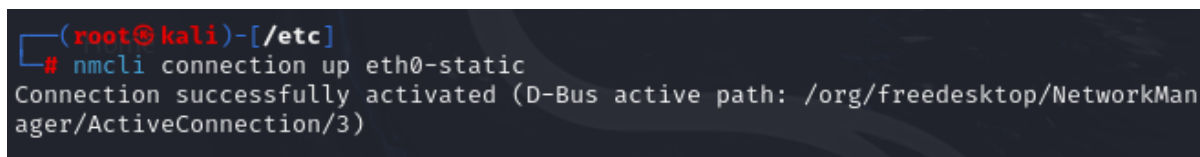
La machine virtuelle **KALI1** est configurée avec des adresses IPv4 et IPv6 statiques en utilisant l'outil `nmcli` sous Linux. Voici les commandes utilisées pour cette configuration :

```
# Configuration de l'adresse IPv4 sur l'interface eth0
nmcli connection add type ethernet con-name eth0-static \
    ifname eth0 ipv4.addresses 172.168.0.2/24 \
    ipv4.gateway 172.168.0.1 ipv4.method manual
```

```
(kali@kali)-[~]
$ nmcli connection add type ethernet con-name eth1-static ifname eth1 ipv6.
addresses 2001:db8:1:2::2/64 ipv6.gateway 2001:db8:1:2::1 ipv6.method manual
Connection 'eth1-static' (431129f4-9b3f-4cae-aaca-5e96322542ed) successfully
added.
```

Figure 3.11: Configuration réseau statique pour KALI1 avec nmcli

```
# Configuration de l'adresse IPv6 sur l'interface eth1
nmcli connection add type ethernet con-name eth1-static \
    ifname eth1 ipv6.addresses 2001:db8:1:3::2/64 \
    ipv6.gateway 2001:db8:1:3::1 ipv6.method manual
# Activation des connexions réseau
nmcli connection up eth0-static
nmcli connection up eth1-static
```



```
(root@kali)-[/etc]
# nmcli connection up eth0-static
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
```

Figure 3.12: activer l'interface

Explication des commandes :

- `nmcli connection add` : Crée une nouvelle connexion réseau avec des paramètres statiques.
- `type ethernet` : Spécifie que la connexion est de type Ethernet.
- `ifname` : Définit l'interface réseau à configurer (`eth0` pour IPv4 et `eth1` pour IPv6).
- `ipv4.addresses` : Configure l'adresse IP IPv4 et son masque de sous-réseau.
- `ipv4.gateway` : Spécifie la passerelle par défaut pour IPv4.
- `ipv6.addresses` : Configure l'adresse IP IPv6 et son préfixe.
- `ipv6.gateway` : Spécifie la passerelle par défaut pour IPv6.
- `connection up` : Active la connexion réseau configurée.

Ces commandes assurent une configuration statique des adresses IPv4 et IPv6 sur les interfaces **eth0** et **eth1** respectivement. La machine est ainsi prête pour les tests de connectivité et de fragmentation.

Vérification de la configuration sur KALI1 Pour vérifier les adresses IP configurées et les routes sur KALI1, nous utilisons les commandes suivantes :

- **Affichage des interfaces** : `ip addr show`
- **Table de routage IPv4** : `ip route`

- Table de routage IPv6 : `ip -6 route`

```
(kali@kali)-[~]
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:16:2c:9e brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.2/24 brd 192.168.0.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1:3:aecd:422c:b4b:5fff/64 scope global dynamic noprefixroute
        valid_lft 2591943sec preferred_lft 604743sec
    inet6 fe80::e390:57a9:265c:81a2/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Figure 3.13: `ip addr show`

```
(kali@kali)-[~]
$ ip -6 addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:db8:1:3:aecd:422c:b4b:5fff/64 scope global dynamic noprefixroute
        valid_lft 2591870sec preferred_lft 604670sec
    inet6 fe80::e390:57a9:265c:81a2/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 state DOWN qlen 1000
    inet6 2001:db8:1:3::2/64 scope global tentative noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::d48f:bb88:f7ee:c87a/64 scope link tentative noprefixroute
        valid_lft forever preferred_lft forever
```

Figure 3.14: `ipv6 addr show`

```
(root@kali)-[/etc]
$ ip route
default via 192.168.254.2 dev eth0 proto dhcp src 192.168.254.132 metric 100
192.168.254.0/24 dev eth0 proto kernel scope link src 192.168.254.132 metric 100
```

Figure 3.15: `ip route`

3.3.4 Configuration de R2 et KALI2

La configuration pour le routeur **R2** et la machine virtuelle **KALI2** est similaire à celle décrite pour R1 et KALI1. Les adresses IP utilisées pour R2 et KALI2 sont les suivantes :

- **R2** :
 - IPv4 : 192.168.0.1/24 (f0/0) et 10.0.1.2/24 (f0/1).
 - IPv6 : 2001:db8:1:2::1/64 (f0/0) et 2001:db8:1:1::1/64 (f0/1).
- **KALI2** :
 - IPv4 : 192.168.0.2/24.
 - IPv6 : 2001:db8:1:2::2/64.

La même logique de configuration et de vérification est appliquée à ces périphériques.

3.4 Verification de la configuration

Pour s'assurer que la configuration a été correctement réalisée et que la topologie est pleinement fonctionnelle, nous effectuons des tests de connectivité à l'aide de la commande ping entre les différents hôtes. Ces tests permettent de vérifier que les adresses IP ont été correctement configurées, que les routes sont bien établies, et que la communication entre les périphériques est opérationnelle.

- **Ping entre PC1 et PC2**

```
(kali㉿kali)-[~]  
$ ping 172.168.0.2  
PING 172.168.0.2 (172.168.0.2) 56(84) bytes of data.  
64 bytes from 172.168.0.2: icmp_seq=1 ttl=64 time=0.032 ms  
64 bytes from 172.168.0.2: icmp_seq=2 ttl=64 time=0.084 ms  
64 bytes from 172.168.0.2: icmp_seq=3 ttl=64 time=0.059 ms  
64 bytes from 172.168.0.2: icmp_seq=4 ttl=64 time=0.070 ms
```

Figure 3.16: Ping entre KALI1 et KALI2

```
(kali㉿kali)-[~]  
$ ping6 2001:db8:1:3::2  
PING 2001:db8:1:3::2 (2001:db8:1:3::2) 56 data b  
From 2001:db8:1:3:aecd:422c:b4b:5fff icmp_seq=1
```

Figure 3.17: Ping6 entre KALI1 et KALI2

- **Ping entre R1 et R2**

```
R1#ping 10.0.1.1  
  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 10.0.1.1, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/77/84 ms
```

Figure 3.18: Ping entre R1 et R2

```
R1#ping 2001:DB8:1:1::1  
  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 2001:DB8:1:1::1, timeout is 2 seconds:  
!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 36/52/96 ms
```

Figure 3.19: Ping6 entre R1 et R2

Tous les tests de ping ont été réalisés avec succès, ce qui confirme que la configuration a été correctement effectuée et que la topologie est pleinement fonctionnelle.

3.5 Fragmentation

Dans cette section, nous allons réaliser des tests de ping avec différentes tailles de paquets sur les protocoles **IPv4** et **IPv6**. Les captures seront effectuées à l'aide de l'outil **Wireshark** pour observer les comportements de fragmentation.

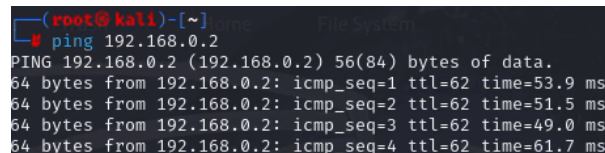
3.5.1 Fragmentation IPv4

Pour tester la fragmentation en IPv4, nous allons utiliser la commande ping avec l'option `-s` pour spécifier la taille du paquet. Voici les étapes du test :

1. Exécution des commandes de ping :

- **Ping avec une taille de paquet petite (par exemple 1000 octets) :**

```
ping -s 1000 192.168.0.2
```

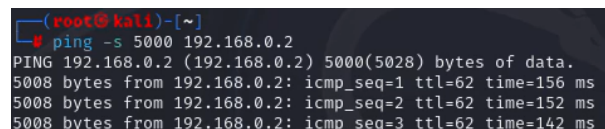


```
(root@kali)~[~]
$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=62 time=53.9 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=62 time=51.5 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=62 time=49.0 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=62 time=61.7 ms
```

Figure 3.20: ping IPv4

- **Ping avec une taille de paquet plus grande (par exemple 5000 octets) :**

```
ping -s 5000 192.168.0.2
```



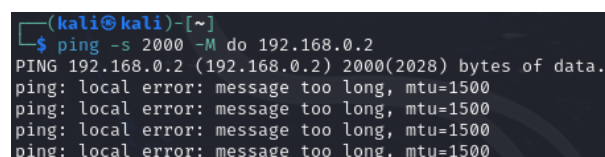
```
(root@kali)~[~]
$ ping -s 5000 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 5000(5028) bytes of data.
5008 bytes from 192.168.0.2: icmp_seq=1 ttl=62 time=156 ms
5008 bytes from 192.168.0.2: icmp_seq=2 ttl=62 time=152 ms
5008 bytes from 192.168.0.2: icmp_seq=3 ttl=62 time=142 ms
```

Figure 3.21: ping IPv4 de taille supérieur à MTU

- **Ping avec une taille de paquet plus grande (par exemple 2000 octets) :**

L'objectif est de tester la capacité du réseau à gérer des paquets de grande taille et d'observer si une fragmentation se produit ou non. La commande suivante est utilisée pour envoyer un ping avec un paquet de 2000 octets :

```
ping -s 2000 -M do 192.168.0.2
```



```
(kali@kali)~[~]
$ ping -s 2000 -M do 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 2000(2028) bytes of data.
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
```

Figure 3.22: ping avec un paquet de 2000 octets

Description des options :

- `-s 2000` : Spécifie la taille de la charge utile (payload) ICMP, ici 2000 octets.
- `-M do` : L'option `-M do` (Don't Fragment) interdit la fragmentation des paquets. Si le paquet dépasse la MTU (Maximum Transmission Unit) autorisée sur le chemin réseau, le ping échouera, permettant ainsi d'identifier les limitations.
- `192.168.0.2` : Adresse IP de la destination cible.

En cas de succès, cela signifie que le réseau accepte des paquets de grande taille sans nécessiter de fragmentation. Si la MTU est dépassée, la commande retournera une erreur indiquant que le paquet ne peut pas être acheminé sans fragmentation.

3.5.2 Sauvegarde de la configuration sur le routeur R1

La capture ci-dessus montre la commande utilisée pour sauvegarder la configuration courante (`running-config`) dans la configuration de démarrage (`startup-config`) sur un routeur Cisco.

- `copy run start` : Cette commande copie la configuration active (en mémoire vive) vers la mémoire NVRAM pour assurer sa persistance après un redémarrage.
- Destination filename : Par défaut, le fichier est sauvegardé sous le nom `startup-config`.
- Confirmation de sauvegarde : L'indication `Building configuration... [OK]` confirme que la configuration a été sauvegardée avec succès.

Cette étape est essentielle pour éviter la perte des modifications de configuration en cas de redémarrage du routeur.

```
R1#copy run star
Destination filename [startup-config]?
Building configuration...
[OK]
```

Figure 3.23: Sauvegarde de la configuration sur le routeur R1

2. Capture avec Wireshark :

- Démarrez la capture Wireshark sur l'interface connectée.
- Observez les trames ICMP capturées.
- Recherchez le champ **Flags** pour identifier si la fragmentation a eu lieu (`More Fragments` activé).
- Utilisez le filtre suivant pour afficher uniquement les trames ICMP : `icmp`

```

010. .... = Flags: 0x2, Don't fragment
0... .... = Reserved bit: Not set
.1.. .... = Don't fragment: Set
..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 63
Protocol: ICMP (1)

```

Figure 3.24: ping flags

| | | | | | | |
|----|------------|-------------|-------------|------|------|----------------|
| 28 | 142.309519 | 172.168.0.2 | 192.168.0.2 | IPv4 | 1514 | Fragmented ... |
| 29 | 142.325810 | 172.168.0.2 | 192.168.0.2 | IPv4 | 1514 | Fragmented ... |
| 30 | 142.341855 | 172.168.0.2 | 192.168.0.2 | IPv4 | 1514 | Fragmented ... |
| 31 | 142.357640 | 172.168.0.2 | 192.168.0.2 | ICMP | 602 | Echo (ping)... |

Figure 3.25: fragmentation IPv4

Dans les fragments capturés, le **deuxième fragment** présente un **saut d'offset de 0 à 1480**, indiquant que la charge utile du premier fragment occupe 1480 octets. Cet offset séquentiel permet d'identifier la position de chaque fragment dans le datagramme original. Dans le **quatrième fragment**, le champ More Fragments (MF) est réglé sur **0**, signalant qu'il s'agit du dernier fragment.

```

001. .... = Flags: 0x1, More fragments
0... .... = Reserved bit: Not set
.0.. .... = Don't fragment: Not set
..1. .... = More fragments: Set
...0 0000 0000 0000 = Fragment Offset: 0

```

Figure 3.26: fragment 1

```

001. .... = Flags: 0x1, More fragments
0... .... = Reserved bit: Not set
.0.. .... = Don't fragment: Not set
..1. .... = More fragments: Set
...0 0000 1011 1001 = Fragment Offset: 1480

```

Figure 3.27: fragment 2

```

000. .... = Flags: 0x0
0... .... = Reserved bit: Not set
.0.. .... = Don't fragment: Not set
..0. .... = More fragments: Not set
...0 0010 0010 1011 = Fragment Offset: 4440

```

Figure 3.28: fragment 4

3. **Analyse des résultats** : Vérifiez si les paquets sont fragmentés (champ Fragment Offset et More Fragments).

3.5.3 Test de ping pour IPv6

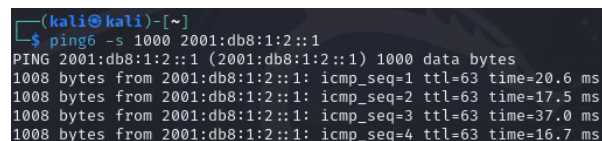
Pour IPv6, le processus est similaire, mais nous utilisons la commande ping adaptée à IPv6 et observons les extensions de fragmentation.

1. Exécution des commandes de ping :

- Ping avec une taille de paquet petite (par exemple 1000 octets) :

Listing 3.2: Commande Ping IPv6 avec taille 1000

```
ping -6 1000 2001:db8:1:3::2
```



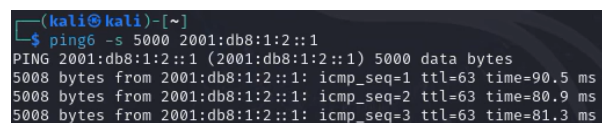
```
(kali@kali)-[~]
$ ping6 -s 1000 2001:db8:1:2::1
PING 2001:db8:1:2::1 (2001:db8:1:2::1) 1000 data bytes
1008 bytes from 2001:db8:1:2::1: icmp_seq=1 ttl=63 time=20.6 ms
1008 bytes from 2001:db8:1:2::1: icmp_seq=2 ttl=63 time=17.5 ms
1008 bytes from 2001:db8:1:2::1: icmp_seq=3 ttl=63 time=37.0 ms
1008 bytes from 2001:db8:1:2::1: icmp_seq=4 ttl=63 time=16.7 ms
```

Figure 3.29: ping6 1000

- Ping avec une taille de paquet plus grande (par exemple 5000 octets) :

Listing 3.3: Commande Ping IPv6 avec taille 2000

```
ping -6 5000 2001:db8:1:3::2
```

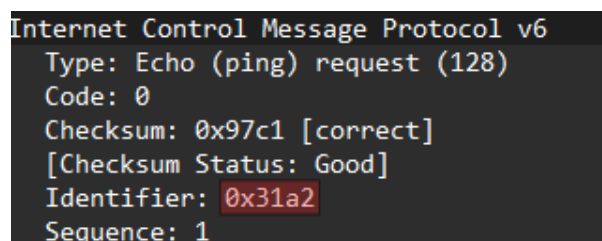


```
(kali@kali)-[~]
$ ping6 -s 5000 2001:db8:1:2::1
PING 2001:db8:1:2::1 (2001:db8:1:2::1) 5000 data bytes
5008 bytes from 2001:db8:1:2::1: icmp_seq=1 ttl=63 time=90.5 ms
5008 bytes from 2001:db8:1:2::1: icmp_seq=2 ttl=63 time=80.9 ms
5008 bytes from 2001:db8:1:2::1: icmp_seq=3 ttl=63 time=81.3 ms
```

Figure 3.30: ping6 5000

2. Capture avec Wireshark :

- Lancez la capture Wireshark sur l'interface IPv6.
- Utilisez le filtre suivant pour afficher uniquement les trames ICMPv6 :icmpv6
- Observez les détails de la fragmentation dans l'**extension de fragmentation** :
 - Champ Fragment Offset.
 - Champ More Fragment (MF).
 - Identification du paquet.



```
Internet Control Message Protocol v6
Type: Echo (ping) request (128)
Code: 0
Checksum: 0x97c1 [correct]
[Checksum Status: Good]
Identifier: 0x31a2
Sequence: 1
```

Figure 3.31: ping flags

| | | | | |
|--------------|-----------------|---------------------|--------|---------------------|
| 30 83.835208 | 2001:db8:1:2::1 | 2001:db8:1:3:aec... | IPv6 | 1510 IPv6 fragme... |
| 31 83.845898 | 2001:db8:1:2::1 | 2001:db8:1:3:aec... | IPv6 | 1510 IPv6 fragme... |
| 32 83.857789 | 2001:db8:1:2::1 | 2001:db8:1:3:aec... | IPv6 | 1510 IPv6 fragme... |
| 33 83.866972 | 2001:db8:1:2::1 | 2001:db8:1:3:aec... | ICMPv6 | 726 Echo (ping)... |

Figure 3.32: fragmentation IPv6

Dans les fragments capturés en **IPv6**, le deuxième fragment présente un **saut d'offset de 0 à 1480**, indiquant que la charge utile du premier fragment occupe 1480 octets. Cet offset séquentiel est spécifié dans l'**en-tête de fragmentation** et permet d'identifier la position de chaque fragment dans le paquet IPv6 original. Dans le quatrième fragment, le champ M (More Fragment) est réglé sur **0**, signalant qu'il s'agit du dernier fragment.

```
Fragment Header for IPv6
Next header: ICMPv6 (58)
Reserved octet: 0x00
0000 0000 0000 0... = Offset: 0 (0 bytes)
.... .... .... .00. = Reserved bits: 0
.... .... .... ...1 = More Fragments: Yes
Identification: 0xd9c4024b
```

Figure 3.33: fragment 1

```
Fragment Header for IPv6
Next header: ICMPv6 (58)
Reserved octet: 0x00
0000 0101 1010 1... = Offset: 181 (1448 bytes)
.... .... .... .00. = Reserved bits: 0
.... .... .... ...1 = More Fragments: Yes
Identification: 0xd9c4024b
```

Figure 3.34: fragment 2

```
Fragment Header for IPv6
Next header: ICMPv6 (58)
Reserved octet: 0x00
0001 0000 1111 1... = Offset: 543 (4344 bytes)
.... .... .... .00. = Reserved bits: 0
.... .... .... ...0 = More Fragments: No
Identification: 0xd9c4024b
```

Figure 3.35: fragment 4

3. **Analyse des résultats** : Vérifiez la présence de fragments et comparez les offsets.

3.5.4 Comparaison des résultats IPv4 et IPv6

Une fois les tests et les captures effectués, nous analysons les différences suivantes entre IPv4 et IPv6 :

- La gestion de la fragmentation (réalisée par les routeurs en IPv4, uniquement à la source en IPv6).

- Les champs d'en-tête utilisés pour la fragmentation (Flags et Fragment Offset pour IPv4, extension de fragmentation pour IPv6).
- La taille des paquets et le nombre de fragments générés.

Ces observations permettront de conclure sur l'efficacité et les différences de la gestion de la fragmentation entre IPv4 et IPv6.

3.6 Configuration du serveur FTP et transfert de fichier

Dans cette section, nous détaillons la configuration du serveur FTP sur la machine et les étapes nécessaires pour envoyer un fichier de **KALI1** vers **KALI2** en utilisant la commande `get`.

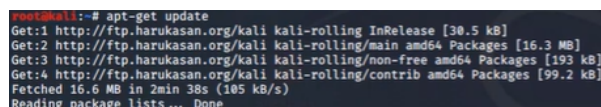
3.6.1 Configuration du serveur FTP

Pour configurer le serveur FTP sur **KALI1**, les étapes suivantes sont réalisées :

1. Mise à jour des paquets et installation de `vsftpd` :

La mise à jour des paquets est effectuée avec la commande suivante :

```
apt-get update
```

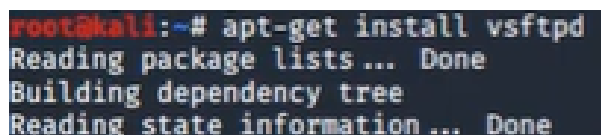


```
root@kali:~# apt-get update
Get:1 http://ftp.harukasan.org/kali kali-rolling InRelease [30.5 kB]
Get:2 http://ftp.harukasan.org/kali kali-rolling/main amd64 Packages [16.3 MB]
Get:3 http://ftp.harukasan.org/kali kali-rolling/non-free amd64 Packages [193 kB]
Get:4 http://ftp.harukasan.org/kali kali-rolling/contrib amd64 Packages [99.2 kB]
Fetched 16.6 MB in 2min 38s (105 kB/s)
Reading package lists... Done
```

Figure 3.36: Mise à jour des paquets

Ensuite, le serveur FTP `vsftpd` est installé :

```
apt-get install vsftpd
```



```
root@kali:~# apt-get install vsftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Figure 3.37: Installation de `vsftpd`

2. Configuration des fichiers nécessaires :

Création et modification des fichiers de configuration :

```
cd /etc/
vim vsftpd.conf
touch vsftpd.chroot_list
```

```

root@kali:~# cd /etc/
root@kali:/etc# vim vsftpd.conf
root@kali:/etc# touch vsftpd.chroot_list

```

Figure 3.38: Création des fichiers de configuration

- Le fichier `vsftpd.conf` est configuré pour permettre les connexions FTP. - Le fichier `vsftpd.chroot_list` liste les utilisateurs autorisés à accéder au serveur FTP.

3. Création d'un utilisateur pour le FTP :

Un nouvel utilisateur est créé pour l'accès FTP :

```
adduser dreams
```

```

(root@kali)-[/etc]
# adduser dreams123
info: Adding user 'dreams123' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group 'dreams123' (1001) ...
info: Adding new user 'dreams123' (1001) with group 'dreams123 (1001)' ...
info: Creating home directory '/home/dreams123' ...
info: Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for dreams123
Enter the new value, or press ENTER for the default
  Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
info: Adding new user 'dreams123' to supplemental / extra groups 'users' ...
info: Adding user 'dreams123' to group 'users' ...

```

Figure 3.39: Création de l'utilisateur FTP adduser 'dreams123'(exemple)

Après la création, le mot de passe est défini, et l'utilisateur est ajouté aux groupes nécessaires.

4. Démarrage et vérification du service vsftpd :

Le service FTP est démarré et son statut est vérifié :

```

service vsftpd start
systemctl status vsftpd

```

```

root@kali:/etc# vim vsftpd.chroot_list
root@kali:/etc# service vsftpd start

```

Figure 3.40: Démarrage et vérification de vsftpd

```

File Actions Edit View Help
(root@kali)~# systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled; preset>
   Active: active (running) since Fri 2024-12-13 04:05:03 EST; 4min 3s ago
   Invocation: 8c1aa8650fb64796abb66ff911396131
   Process: 861 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exit>
   Main PID: 875 (vsftpd)
   Tasks: 1 (limit: 4596)
   Memory: 1008K (peak: 1.2M)
   CPU: 31ms
   CGroup: /system.slice/vsftpd.service
           └─875 /usr/sbin/vsftpd /etc/vsftpd.conf

Dec 13 04:05:03 kali systemd[1]: Starting vsftpd.service - vsftpd FTP server>
Dec 13 04:05:03 kali systemd[1]: Started vsftpd.service - vsftpd FTP server.
lines 1-14/14 (END)
zsh: suspended systemctl status vsftpd

```

Figure 3.41: status du serveur FTP sur KALI1

3.6.2 Transfert du fichier de KALI1 vers KALI2

Une fois le serveur FTP configuré, le transfert du fichier `myfile.txt` de **KALI1** vers **KALI2** est effectué via la commande `ftp`. Voici les étapes réalisées :

1. Connexion au serveur FTP :

Depuis **KALI2**, connexion au serveur FTP en utilisant l'adresse IP de **KALI1** :

```
ftp 172.168.0.2
```

Une fois connecté, l'utilisateur `dreams` saisit son mot de passe pour accéder au serveur.

2. Navigation dans les répertoires et téléchargement du fichier :

- Listage des fichiers disponibles :

```
ls
```

- Navigation dans le répertoire contenant le fichier :

```
cd upload
```

```

(root@kali)~# ftp 172.168.0.2
Connected to 172.168.0.2.
220 (vsFTPd 3.0.3)
Name (172.168.0.2:kali): dreams
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||56356|)
150 Here comes the directory listing.
drwxr-xr-x  2 1002  1002    4096 Dec 08 10:29 upload
226 Directory send OK.
ftp> ls upload
229 Entering Extended Passive Mode (|||16783|)
150 Here comes the directory listing.
-rwxrwxrwx  1 1002  1002    22 Dec 08 10:28 myfile.txt
226 Directory send OK.
ftp> get upload/myfile.txt
local: upload/myfile.txt remote: upload/myfile.txt
ftp: Can't access 'upload/myfile.txt': No such file or directory
ftp> cd upload
250 Directory successfully changed.
ftp> get myfile.txt
local: myfile.txt remote: myfile.txt

```

Figure 3.42: Connexion au serveur + Navigation dans le répertoire upload

- Téléchargement du fichier `myfile.txt` :

```
get myfile.txt
```

```
ftp> get upload/myfile.txt
local: upload/myfile.txt remote: upload/myfile.txt
ftp: Can't access 'upload/myfile.txt': No such file or directory
ftp> cd upload
250 Directory successfully changed.
ftp> get myfile.txt
local: myfile.txt remote: myfile.txt
229 Entering Extended Passive Mode (|||9878|)
150 Opening BINARY mode data connection for myfile.txt (22 bytes).
100% |*****| 22 0.69 KiB/s 00:00 ETA
226 Transfer complete.
22 bytes received in 00:00 (0.19 KiB/s)
ftp>
```

Figure 3.43: Téléchargement du fichier avec get

3. Vérification du fichier transféré :

Après le téléchargement, nous vérifions la présence du fichier dans le répertoire local de la machine virtuelle KALI2 :

```
cd /root
ls
cat myfile.txt
```

```
(root@kali)-[//]
# cd /root

(root@kali)-[~]
# ls
myfile.txt

(root@kali)-[~]
# cat myfile.txt
Network Project 24/25
```

Figure 3.44: Vérification du fichier sur KALI2

Cette commande permet de visualiser le contenu du fichier pour confirmer le transfert.

Conclusion : Le fichier `myfile.txt` a été transféré avec succès de **KALI1** vers **KALI2**, comme vérifié par son contenu affiché dans le terminal.

3.6.3 Analyse de la capture Wireshark

La capture Wireshark ci-dessous présente un échange FTP entre deux hôtes (192.168.0.2 et 172.168.0.2) ainsi que les paquets TCP associés à l'établissement de la connexion et aux erreurs de connexion. Voici les détails importants observés dans la capture :

- **Établissement de la connexion TCP (Three-Way Handshake) :**

- Paquet **50** : L'hôte 192.168.0.2 envoie un paquet SYN pour établir une connexion TCP avec 172.168.0.2.
- Paquet **51** : L'hôte 172.168.0.2 répond avec SYN, ACK, acceptant la demande de connexion.
- Paquet **52** : L'hôte 192.168.0.2 envoie un ACK, complétant ainsi le processus de connexion.

- **Échange FTP :**

- Paquet **54** : Le serveur FTP (172.168.0.2) répond avec le message d'accueil 220 indiquant que le service FTP est prêt.
- Paquet **55** : L'utilisateur dreams est transmis avec la commande FTP USER dreams.
- Paquet **57** : Le serveur répond avec 331, demandant un mot de passe pour l'utilisateur.
- Paquet **59** : Une tentative de mot de passe incorrect est effectuée avec la commande PASS (mot de passe non affiché).
- Paquet **60** : Le serveur renvoie 530 Login incorrect, indiquant un échec d'authentification.

- **Paquets TCP liés à l'échange :**

- Les paquets **47, 48, et 58** montrent des ACK entre les deux hôtes confirmant la réception des segments.
- Les numéros de séquence (Seq) et d'acquittement (Ack) sont incrémentés de manière cohérente tout au long de la capture.

3.6.4 Analyse de la capture Wireshark

La capture Wireshark ci-dessous présente un échange FTP entre deux hôtes (192.168.0.2 et 172.168.0.2) ainsi que les paquets TCP associés à l'établissement de la connexion et aux erreurs de connexion. Voici les détails importants observés dans la capture :

- **Établissement de la connexion TCP (Three-Way Handshake) :**

- Paquet **50** : L'hôte 192.168.0.2 envoie un paquet SYN pour établir une connexion TCP avec 172.168.0.2.
- Paquet **51** : L'hôte 172.168.0.2 répond avec SYN, ACK, acceptant la demande de connexion.
- Paquet **52** : L'hôte 192.168.0.2 envoie un ACK, complétant ainsi le processus de connexion.

- **Échange FTP :**

- Paquet **54** : Le serveur FTP (172.168.0.2) répond avec le message d'accueil 220 indiquant que le service FTP est prêt.
- Paquet **55** : L'utilisateur dreams est transmis avec la commande FTP USER dreams.
- Paquet **57** : Le serveur répond avec 331, demandant un mot de passe pour l'utilisateur.
- Paquet **59** : Une tentative de mot de passe incorrect est effectuée avec la commande PASS (mot de passe non affiché).
- Paquet **60** : Le serveur renvoie 530 Login incorrect, indiquant un échec d'authentification.

• **Paquets TCP liés à l'échange :**

- Les paquets **47, 48, et 58** montrent des ACK entre les deux hôtes confirmant la réception des segments.
- Les numéros de séquence (Seq) et d'acquittement (Ack) sont incrémentés de manière cohérente tout au long de la capture.

| | | | | | |
|----|------------|-------------|-------------|-----|-----------------------------------------------------------------------------------------------------------|
| 46 | 546.748523 | 192.168.0.2 | 172.168.0.2 | FTP | 98 Request: PASS insatmpirt\033[8\033[2~\033[2~\033[6~ |
| 47 | 546.826429 | 172.168.0.2 | 192.168.0.2 | TCP | 66 21 → 58126 [ACK] Seq=55 Ack=46 Win=31872 Len=0 TSval=2798140692 TSecr=2816096913 |
| 48 | 550.103653 | 172.168.0.2 | 192.168.0.2 | FTP | 88 Response: 530 Login incorrect. |
| 49 | 550.134548 | 192.168.0.2 | 172.168.0.2 | TCP | 66 58126 → 21 [ACK] Seq=46 Ack=77 Win=65460 Len=0 TSval=2816100297 TSecr=2798143966 |
| 50 | 554.237773 | 192.168.0.2 | 172.168.0.2 | TCP | 74 37938 → 21 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM TSval=2816104402 TSecr=0 WS=2 |
| 51 | 554.268166 | 172.168.0.2 | 192.168.0.2 | TCP | 74 21 → 37938 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=2798148133 TSecr=2816104402 |
| 52 | 554.298954 | 192.168.0.2 | 172.168.0.2 | TCP | 66 37938 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2816104461 TSecr=2798148133 |
| 53 | 554.329028 | 172.168.0.2 | 192.168.0.2 | FTP | 86 Response: 220 (vsFTPd 3.0.3) |
| 54 | 554.360578 | 192.168.0.2 | 172.168.0.2 | TCP | 66 37938 → 21 [ACK] Seq=1 Ack=21 Win=65516 Len=0 TSval=2816104522 TSecr=2798148198 |
| 55 | 556.751389 | 192.168.0.2 | 172.168.0.2 | FTP | 79 Request: USER dreams |
| 56 | 556.782746 | 172.168.0.2 | 192.168.0.2 | TCP | 66 21 → 37938 [ACK] Seq=21 Ack=14 Win=31872 Len=0 TSval=2798150647 TSecr=2816106925 |
| 57 | 556.798935 | 172.168.0.2 | 192.168.0.2 | FTP | 100 Response: 331 Please specify the password. |
| 58 | 556.830181 | 192.168.0.2 | 172.168.0.2 | TCP | 66 37938 → 21 [ACK] Seq=14 Ack=55 Win=65482 Len=0 TSval=2816106993 TSecr=2798150647 |
| 59 | 560.695235 | 192.168.0.2 | 172.168.0.2 | FTP | 98 Request: PASS insatmpirt\033[8\033[2~\033[2~\033[6~ |
| 60 | 560.757525 | 172.168.0.2 | 192.168.0.2 | TCP | 66 21 → 37938 [ACK] Seq=55 Ack=46 Win=31872 Len=0 TSval=2798154632 TSecr=2816110857 |
| 61 | 563.843877 | 172.168.0.2 | 192.168.0.2 | FTP | 88 Response: 530 Login incorrect. |
| 62 | 563.874601 | 192.168.0.2 | 172.168.0.2 | TCP | 66 37938 → 21 [ACK] Seq=46 Ack=77 Win=65460 Len=0 TSval=2816114037 TSecr=2798157707 |

Figure 3.45: Capture Wireshark d'un échange FTP avec erreurs de connexion

Conclusion : La capture montre clairement l'établissement de la connexion TCP via un handshake en trois étapes ainsi que l'échec d'authentification FTP. Les messages 530 Login incorrect indiquent que les tentatives de connexion ont échoué en raison d'un mot de passe incorrect.

3.7 Bibliographie

- Documentation officielle GNS3: GNS3 Official Documentation
- Documentation officielle Wireshark: Wireshark Official Documentation
- *Wireshark for Security Professionals: Using Wireshark and the Metasploit Framework* by Jessey Bullock and Jeff T. Parker.
- Documentation officielle FTP: RFC 959 Official FTP Documentation
- Fragmentation IPV4: GeeksforGeeks article on IPv4 Datagram Header
- Fragmentation IPV6: RFC 8200: Internet Protocol, Version 6 (IPv6) Specification