République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option : Réseaux et Systèmes Distribués (RSD)

Thème

Conception et implémentation d'une plateforme de sécurité programmable basée sur SDN pour la détection et la mitigation des attaques DDoS

Présenté par :
- Khaldi Douaa
- Bali Meriem Ibtihel

Encadré par :
- Mr. Bambrik Ilyas MCB

Présenté le 21 juin 2026 devant le jury composé de :
- Nabila Labraoui Prof Président
- Abdeldjelil Hanane MCB Examinateur

Année universitaire : 2025-2026


بسم الله الرحمن الرحيم

Résumé

Les réseaux traditionnels présentent plusieurs limites en matière de gestion, de flexibilité et de sécurité face à l'évolution des cyberattaques. La technologie Software Defined Networking (SDN) apporte une nouvelle approche basée sur la séparation du plan de contrôle et du plan de données, permettant une gestion centralisée, dynamique et programmable du réseau.

Dans ce contexte, une plateforme de sécurité programmable multi-services basée sur SDN a été conçue et implémentée afin d'améliorer la supervision et la sécurité de réseau. Cette solution est conçue pour l'analyse du trafic, la détection des comportements anormaux ainsi que l'application automatique de mécanisme de sécurité contre certaines attaques réseau, telles que les attaques DDoS. L'architecture repose sur l'utilisation de contrôleur POX, des commutateurs OpenFlow et de l'environnement Mininet pour la simulation et les tests.

Mots-clés : SDN, cybersécurité, OpenFlow, POX, Mininet, DDoS, sécurité réseau.

Abstract

Traditional networks present several limitations in terms of management, flexibility and security in the face of evolving cyberattacks. Software Defined networking (SDN) introduces a new approach based on the separation of the control plane and the data plane, enabling centralized, dynamic, and programmable network management.

In this context, a programmable multi-service security platform based on SDN was designed and implemented to improve network monitoring and security. This solution is intended for traffic analysis, anomaly detection, and the automatic application of security mechanisms against certain network attacks, such as DDoS attacks. The architecture relies on the use of the POX controller, OpenFlow switches, and The Mininet environment for simulation and testing .

Keywords: SDN, cybersecurity, OpenFlow, POX, Mininet, DDoS, network security.

الملخص

تعاني الشبكات التقليدية من عدة قيود في مجال الإدارة والمرونة والأمن في ظل تطور الهجمات السيبرانية. وتقدم تقنية الشبكات المعرفة برمجيًا (SDN) مقاربة جديدة تعتمد على فصل مستوى التحكم عن مستوى نقل البيانات، مما يسمح بإدارة مركزية وديناميكية وقابلة للبرمجة للشبكة.

في هذا السياق، تم تصميم وتنفيذ منصة أمنية متعددة الخدمات وقابلة للبرمجة اعتمادًا على تقنية SDN بهدف تحسين مراقبة الشبكة وأمنها. صُممت هذه المنصة لتحليل حركة المرور، واكتشاف السلوكيات غير الطبيعية، بالإضافة إلى التطبيق التلقائي لآليات الحماية ضد بعض الهجمات الشبكية مثل هجمات حجب الخدمة الموزعة (DDoS). وتعتمد البنية المقترحة على استخدام متحكم POX ومبدلات OpenFlow وبيئة Mininet من أجل المحاكاة والاختبار.

الكلمات المفتاحية:الشبكات المعرفة برمجياً، الأمن السيبراني، بروتوكول أوبن فلو، متحكم بوكس، منصة مينينيت، كشف الهجمات، هجمات حجب الخدمة الموزعة، إدارة الشبكات القابلة للبرمجة.

Dédicace

Après des années d'efforts, de persévérance, d'épreuves et de nuits de travail, je tiens à me remercier moi-même pour n'avoir jamais abandonné et pour avoir continué malgré les difficultés rencontrées.

Je tiens à exprimer toute ma gratitude à mes chers parents, mon père « Mourad Bali » et ma mère « Chahrazed Mami », pour leur soutien inconditionnel, leurs sacrifices, leurs encouragements et leur amour tout au long de mon parcours.

Je remercie également mes chers frères « Samir, Nadir et Souhaib) pour leur présence et leur soutien, ainsi que ma chère sœur « Ismahen » , qui a toujours été pour moi comme une deuxième mère, par sa tendresse, ses conseils et son affection.

Je tiens à remercier mon binôme « Douaa Khaldi » pour sa collaboration, son aide précieuse et son esprit d'équipe tout au long de ce projet.

Je vous aime.

IBTIHEL MERIEM

Dédicace

Je tiens tout d'abord à remercier Allah, le Tout-Puissant, de m'avoir accordé la force, la patience et le courage nécessaires pour mener ce travail à son terme.

Je souhaite exprimer ma profonde gratitude à mes chers parents, mon père Khaldi Abdelkader et ma mère Achour Nadia, pour leur soutien inconditionnel, leurs sacrifices, leur amour et tous les efforts qu'ils ont consentis pour moi tout au long de mon parcours. Merci pour votre patience, votre confiance, vos encouragements permanents et pour avoir toujours cru en moi. Je vous aime de tout mon cœur, Maman et Papa.

Mes remerciements vont aussi à mon frère Anes, à mes belles cousines ainsi qu'à mes meilleurs amis, pour leur présence, leur écoute. Merci d'avoir toujours été là pour moi.

Enfin, je tiens à remercier particulièrement ma binôme Bali Meriem Ibtihel pour sa patience, son soutien, sa compréhension tout au long de ce projet. Merci d'avoir toujours été présente à mes côtés, d'avoir partagé avec moi les défis de ce travail et d'avoir contribué à sa réussite. Ce fut un réel plaisir de travailler avec toi.

KHALDI DOUAA

Remerciements

Avant tout, nous remercions Allah, le Tout-Puissant, de nous avoir accordé la santé, la force, le courage et la patience nécessaires pour mener à bien ce travail et atteindre cette étape importante de notre parcours universitaire.

Nous adressons notre profonde gratitude à notre encadreur, Dr Ilyas Bambrik, pour son accompagnement, sa disponibilité, ses précieux conseils, son soutien constant et ses orientations pertinentes tout au long de la réalisation de ce mémoire. Son expertise et son engagement ont été d'une grande valeur pour l'aboutissement de ce travail.

Nous exprimons également nos sincères remerciements aux honorables membres du jury : Mme Nabila Labraoui, Mme Abdeldjelil Hanane, qui ont aimablement accepté d'évaluer notre travail. Nous leur sommes reconnaissants pour le temps qu'elles consacrent à l'examen de ce mémoire ainsi que pour leurs remarques, suggestions et recommandations qui contribueront à son amélioration.

Nos remerciements s'adressent aussi à l'ensemble des enseignants et du personnel du Département d'Informatique de la Faculté des Sciences, pour la qualité de leur enseignement et les connaissances qu'ils nous ont transmises tout au long de notre formation.

Nous tenons à exprimer notre reconnaissance à nos chers parents et à nos familles pour leur amour, leur soutien indéfectible, leurs encouragements et les sacrifices consentis afin de nous permettre de poursuivre nos études dans les meilleures conditions.

Enfin, nous remercions tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce mémoire et nous ont soutenus durant cette aventure académique.

À toutes ces personnes, nous adressons nos plus sincères remerciements.

Sommaire

Liste des figures

Liste des tableaux

Abréviations

SDN : Software Defined Networking

TCP : Transmission Control Protocol

IP : Internet Protocol

TCP/IP : Transmission Control Protocol / Internet Protocol

GUI : Graphical User Interface

API : Application Programming Interface

DDoS : Distributed Denial of Service

DNS : Domain Name System

SQL : Structured Query Language

XSS : Cross-Site Scripting

VM : Virtual Machine

CLI : Command Line Interface

SVM : Support Vector Machine

Introduction générale

Avec l'évolution rapide des technologies de l'information et de la communication, les réseaux informatiques sont devenus un élément central du fonctionnement des entreprises, des administrations et des services informatiques. Aujourd'hui, une grande partie des activités critiques repose sur des infrastructures réseau complexes, interconnectées et à grande échelle, ce qui les rend particulièrement exposées aux menaces de sécurité.

Les attaques réseau modernes ne se limitent plus à de simples tentatives d'intrusion, mais prennent des formes variées et souvent massives, comme les attaques par déni de service distribué (Distributed Denial of Service – DDoS), l'usurpation d'identité (IP Spoofing) ou encore certaines techniques de tunneling malveillant. Ces attaques peuvent entraîner des interruptions de service, des pertes financières, une dégradation des performances du réseau ainsi que des atteintes à la confidentialité et à l'intégrité des données.

Les architectures réseau traditionnelles reposent sur une approche distribuée où chaque équipement (routeur, commutateur, pare-feu) intègre ses propres mécanismes de contrôle et de sécurité. Bien que ce modèle ait longtemps été efficace, il présente aujourd'hui plusieurs limites en matière de gestion, de flexibilité et de réaction face aux menaces.

Le paradigme du Software Defined Networking (SDN) propose une nouvelle approche pour la conception et la gestion des réseaux. Celui-ci repose sur la séparation entre le plan de contrôle et le plan de données. Le plan de contrôle est centralisé au niveau d'un contrôleur SDN, tandis que les équipements réseau se limitent principalement à l'acheminement des paquets.

Cette architecture offre plusieurs avantages majeurs en matière de sécurité. La centralisation du contrôle permet au contrôleur SDN de disposer d'une vision globale et en temps réel de l'état du réseau et du trafic. Cela facilite la détection des comportements anormaux, la prévention des attaques et la prise de décisions cohérentes. De plus, grâce au protocole OpenFlow, le contrôleur peut installer dynamiquement des règles de traitement du trafic sur les commutateurs. Cette programmabilité permet d'adapter rapidement les politiques de sécurité en fonction des menaces détectées.

C'est dans cette perspective que s'inscrit ce projet de fin d'études, qui vise la conception et l'implémentation d'une plateforme programmable basée sur SDN pour la détection et la mitigation des attaques DDoS. L'objectif est de mettre en

œuvre un mécanisme capable d'identifier les comportements caractéristiques d'une attaque DDoS et de réagir automatiquement afin de protéger les ressources du réseau et d'assurer la continuité du service.

Ce mémoire est organisé en trois chapitres :

- Le premier chapitre présente les principales attaques réseau, leurs caractéristiques, leurs impacts sur les infrastructures informatiques ainsi que les mécanismes de défense existants, avec un intérêt particulier pour les attaques DDoS.
- Le deuxième chapitre est consacré au paradigme Software Defined Networking (SDN). Il décrit ses concepts fondamentaux, son architecture, ses composants principaux ainsi que les avantages qu'il offre en matière de gestion et de sécurité des réseaux.
- Le troisième chapitre présente la Conception et l'implémentation d'une plateforme programmable basée sur SDN pour la détection et la mitigation des attaques DDoS. Il détaille l'architecture de la plateforme, les outils utilisés tels que Mininet, OpenFlow et le contrôleur POX, ainsi que les mécanismes de détection et de mitigation des attaques DDoS. Enfin, les résultats expérimentaux obtenus sont analysés afin d'évaluer l'efficacité de la solution développée.

Problématique

Avec le développement rapide des data centers modernes, les réseaux informatiques sont devenus plus complexes, plus dynamiques et plus exposés aux menaces de sécurité. En même temps, les attaques réseau sont de plus en plus nombreuses et variées.
Dans les réseaux traditionnels, la sécurité est généralement basée sur les équipements configurés manuellement et sur des règles fixes, telles que les pare-feu traditionnels et les systèmes de détection d'intrusion, qui sont généralement déployés de manière isolés et périphériques. Cette stratégie ne permet pas toujours une détection rapide et globale des attaques. En outre, celle-ci ne permet pas de s'adapter aux nouveaux scénarios de menace, qui nécessitent souvent des interventions manuelles complexes, augmentant ainsi le temps de réaction et l'impact des attaques sur les performances réseau.
Un autre problème important est que les réseaux traditionnels ne donnent pas une vision globale et claire de ce qui se passe dans tout le réseau. Chaque équipement voit seulement une partie du trafic. Cela rend la détection des attaques plus difficile et moins précise.

Face à ces difficultés, il est devenu nécessaire de trouver une nouvelle façon de gérer et de sécuriser les réseaux. La question principale est donc la suivante : comment mettre en place une solution de sécurité qui peut surveiller tout le réseau, détecter rapidement les attaques et réagir automatiquement pour bloquer celle-ci, sans avoir besoin d'interventions manuelles complexes ? Plus précisément, comment utiliser la technologie SDN pour construire une plateforme de sécurité simple, flexible, efficace et capable de protéger le réseau contre plusieurs types d'attaques ?

Chapitre I: Les attaques réseau

I.1. Introduction

Au fil des années, les réseaux informatiques sont devenus plus vulnérables. Ce chapitre présente les principales attaques réseau, à savoir les attaques DDos, les scans de ports et l'usurpation d'adresse IP (IP spoofing). Ensuite, des solutions de sécurité traditionnelles existantes sont discutées et à la fin de ce chapitre, une comparaison entre les mécanismes de sécurité classiques est présentée.

I.2. Modèle TCP/IP

Le modèle TCP/IP est la méthode standard de communication des données sur internet. Il a été développé pour assurer que les données soient transmises correctement entre les appareils. Il divise les messages en paquets pour éviter de renvoyer l'intégralité du message en cas de problème lors de la transmission. Les pâques sont automatiquement réassemblés une fois qu'ils atteignent leur destination. Chaque paquet peut prendre un chemin différent de l'ordinateur expéditeur à l'ordinateur destinataire, surtout si le chemin habituel est trop encombré ou indisponible.

Les quatre couches du modèle TCP/IP:

- Couche de liaison de données
- Couche internet
- Couche de transport
- Couche applicative [1]

La majorité des attaques réseau exploitent le fonctionnement de ces protocoles, en particulier ceux des couches Internet et Transport, comme le protocole IP ou les ports TCP.

I.3. Les Types d'attaques réseau

I.3.1 DDoS (Distributed Denial of Service)

DDoS signifie déni de service distribué, l'une des attaques réseau les plus dangereuses qui touchent les réseaux informatiques. Conçues pour semer le chaos en saturant les systèmes, les attaques DDoS envoient un flot de requêtes qui saturent les ressources du système ou des applications et les rendent inutilisables pour les utilisateurs légitimes.

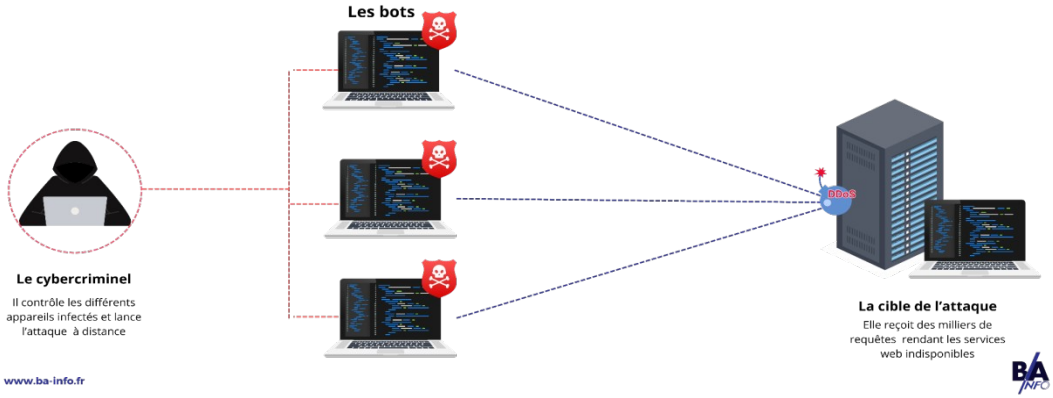


Figure 1: Attaque DDoS [3]

L'une des raisons pour lesquelles les attaques DDoS fonctionnent si bien, c'est sa facilité de mise en place. Les attaquants utilisent souvent des réseaux de zombies (botnets) composés de milliers, voire de millions, d'appareils piratés. Ces appareils peuvent être aussi bien des ordinateurs compromis que des objets connectés non sécurisés, comme des thermostats intelligents ou des caméras.
« Le plus sournois, c'est que ces attaques sont souvent distribuées, c'est-a-dire qu'elles proviennent de sources multiples, ce qui les rend incroyablement difficiles à bloquer sans affecter également le trafic légitime ».[2]

I.3.2 IP Spoofing

L'usurpation d'adresse IP est une technique qui consiste a modifier l'adresse IP source des paquets afin de cacher leur véritable origine. Grâce à cette manipulation, un attaquant peut se faire passer pour un appareil de confiance pour contourner les mécanismes de sécurité du réseau. Par la suite, celui-ci peut mener des actions malveillantes comme l'intrusion dans un réseau, le vol de données ou des attaques par déni de service (DoS).
Dans les communications sur Internet basées sur le modèle TCP/IP, chaque paquet contient une adresse IP source et une adresse IP de destination, ce qui permet aux applications d'identifier l'origine des requêtes et répondre correctement. Lors d'une attaque par usurpation d'adresse IP, l'adresse source est falsifiée, ce qui fait croire au système cible que les requêtes proviennent d'une source légitime.
Fonctionnement de l'usurpation de l'adresse IP (IP Spoofing) :

- Création des paquets : L'attaquant fabrique des paquets IP à l'aide des outils spécialisés. Il peut aussi intercepter des paquets qui circulent à l'aide d'outils d'analyse.

- Modification de l'en-tête IP : L'attaquant modifie l'adresse IP source du paquet et la remplace par une autre adresse, souvent celle d'une machine de confiance.
- Envoi du paquet : Le paquet falsifié est envoyé sur le réseau vers la machine cible.
- Réception par la cible : La machine ciblée reçoit le paquet et pense qu'il provient d'une source légitime, donc elle l'accepte.
- Lancement de l'attaque : L'attaquant profite de cette fausse identité pour contourner la sécurité, accéder à certaines ressources ou lancer d'autres attaques (DDoS, intrusion, etc).

Il est difficile de trouver la véritable localisation ou l'identité de l'attaquant car l'adresse IP source est falsifiée. [6]

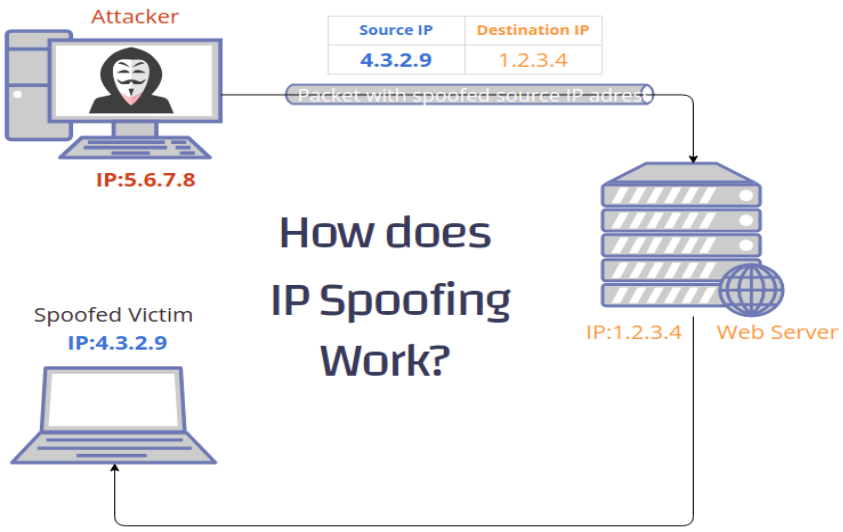


Figure 2: IP Spoofing [7]

I.3.3 Protocol Tunneling

Les attaques par tunneling de protocoles sont un type d'attaque réseau où un pirate déguise un trafic dangereux en le cachant dans un autre protocole autorisé par les contre-mesures de sécurité.

Cette méthode est souvent employée pour contourner les systèmes de sécurité habituels comme les pare-feu.[4] Particulièrement, le DNS Tunneling est une technique très appliquée. C'est une menace pour la sécurité qui cache des données et commandes C2 dans les requêtes DNS. Cette attaque est très difficile à détecter et peut conduire au vol d'informations en grandes volumes sans être détectée.

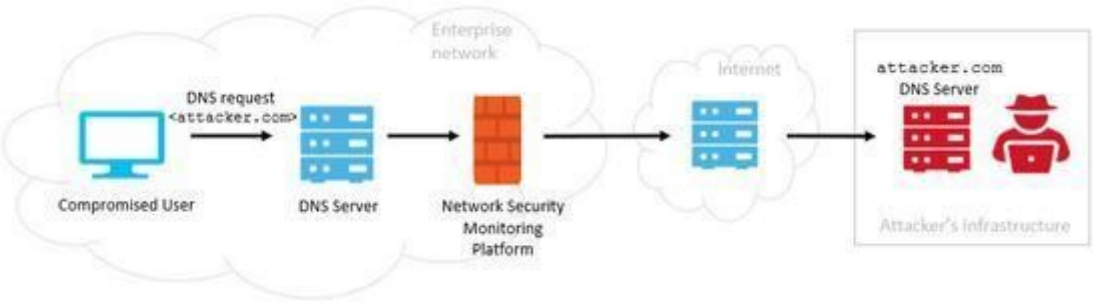


Figure 3: Exemple de DNS Tunneling [4]

Dans l'exemple présenté (Figure 3) :

1. Premièrement, l'attaquant enregistre un nom de domaine malveillant (par exemple attacker.com) hébergé sur un serveur de commande et de contrôle (C&C) géré par lui-même.
2. Une fois que l'attaquant prend le contrôle d'une machine au sein de réseau cible, l'équipement infecté envoie des requêtes DNS vers le domaine malveillant. Les requêtes DNS comportent les données volées.
3. Puisque les requêtes DNS sont souvent autorisées de sortir du réseau, la requête passe par le résolveur DNS et atteint le serveur C&C de l'attaquant, ou se trouve le programme de tunneling.

I.3.4 Cross-site Scripting (XSS)

Les attaques Cross-Site Scripting (XSS) sont un type d'injection où des scripts malveillants, généralement en JavaScript, sont injectés dans des sites web légitimes. Ces attaques se produisent lorsqu'un attaquant utilise une faille dans une application web pour placer son code malveillant, sous la forme d'un script que les visiteurs de l'application vont exécuter lors de chaque accès. Ces failles sont très fréquentes et apparaissent chaque fois qu'un site web affiche les informations fournies par un utilisateur sans les valider ni les encoder.

Par exemple, le navigateur de l'utilisateur, pensant que le script vient d'un site fiable, l'exécute. Le script peut alors accéder aux cookies, aux sessions et à d'autres informations sensibles, et même modifier le contenu HTML de la page web [8].

Les attaques XSS ont deux types principaux :

- **XSS persistant (attaque stockée) :**

  Une attaque côté serveur se produit lorsque le script injecté est stocké dans une zone accessible publiquement d'un site web. Lorsqu'un utilisateur visite cette zone, le navigateur récupère et

affiche les données, ce qui déclenche l'attaque XSS stockée dans le contexte du navigateur [9].

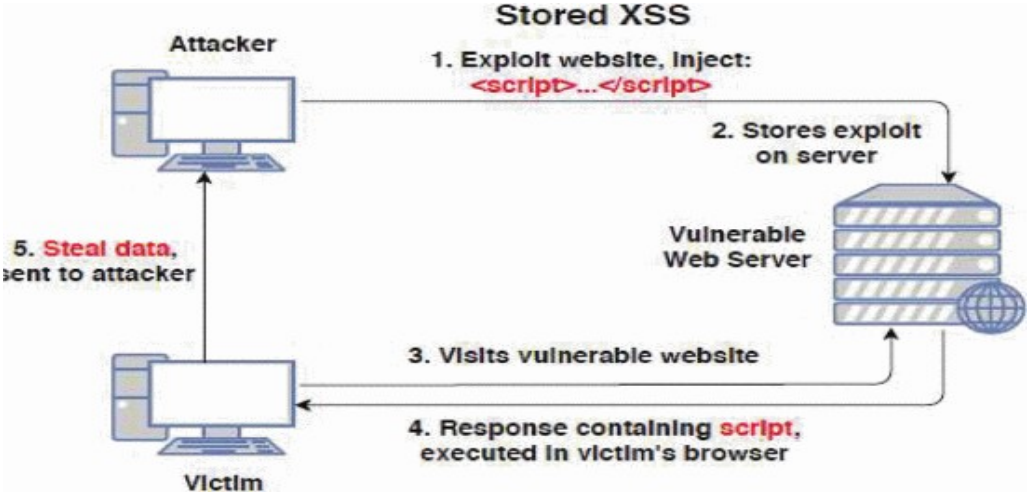


Figure 4: Déroulement typique d'une attaque XSS stockée [9]

- **Non persistante (attaque réfléchie)**

  Ce type d'attaque se produit lorsqu'un fichier saisi par l'utilisateur est envoyé à un site web lors d'une requête, et que ce dernier envoie instantanément des données au navigateur, sans vérification préalable. Pour qu'une attaque XSS inversée réussisse, l'attaquant doit amener la victime à affecter une requête spécifique contenant le script malveillant. Si l'utilisateur visite cette URL, le code malveillant s'exécutera dans son navigateur [9].

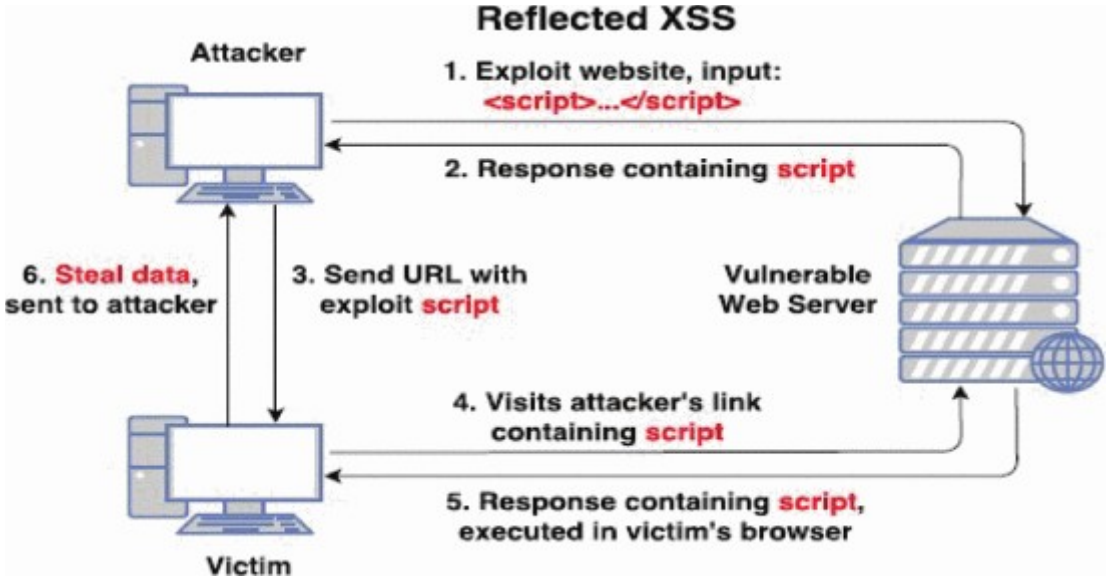


Figure 5: Déroulement typique d'une attaque XSS [9]

I.3.5 Session Hijacking (détournement de session)

L'attaque session hijacking est une méthode cyber-attaque ou un adversaire vole ou intercepte un jeton de session valide (comme cookie ou un identifiant d'authentification), pour prendre le contrôle d'une session déjà authentifié d'un utilisateur déjà légitime. Ces jetons servent à reconnaître un utilisateur sans demander l'authentification à nouveau [11].

Parfois ce type d'attaque est connu comme l'attaque Man in the Middle (MIMA). Après qu'un utilisateur s'est connecté avec succès à un service (par exemple application web), le serveur génère un identifiant de session.

L'attaquant vole cet identifiant en utilisant diverse méthodes [11]:

- Écoute de session : interception du trafic réseau non chiffré (par exemple, avec des outils comme Wireshark) sur un réseau Wi-Fi public.
- Cross-site scripting : insertion de code malveillant dans des sites web pour voler les cookies.
- Session fixation : vous inciter à adopter un identifiant de session spécifique via des liens de phishing.
- Malware de type infostealer : extraction des cookies du navigateur et des données d'empreinte numérique à partir de machines infectées.

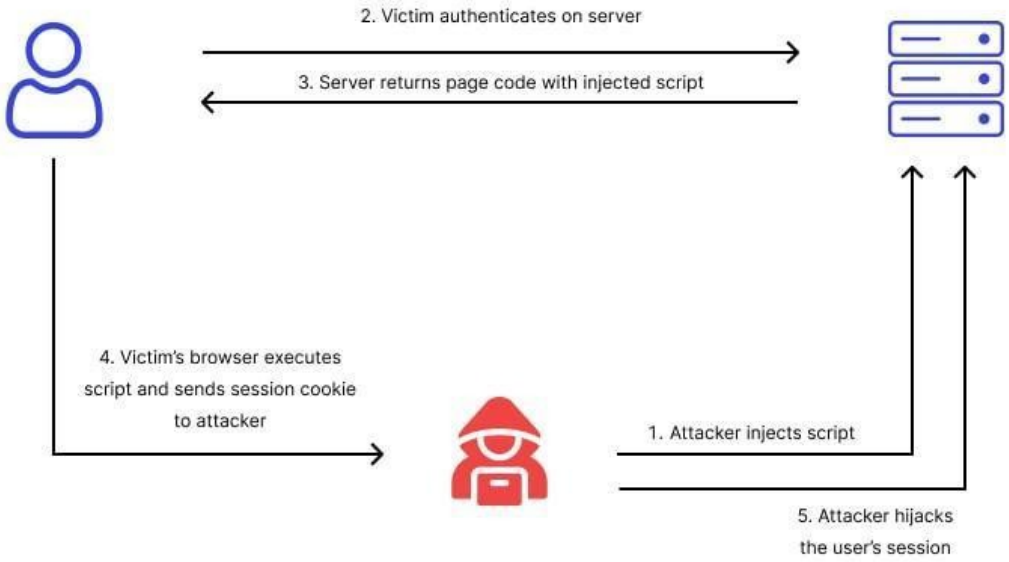


Figure 6: Session hijacking in action [12]

Les types de session hijacking sont les suivants [14] :

- Session Hijacking actif : l'objectif est de prendre directement le contrôle de la session et agir comme l'utilisateur légitime.

- Session Hijacking passif : Espionner la communication et récupérer des informations exploitables.
- Session Hijacking hybride : c'est le fait de combiner entre les approches active et passive.

Le détournement de session permet à l'attaquant d'agir avec les mêmes privilèges que l'utilisateur compromis. Les conséquences comprennent l'accès aux données sensibles, action non autorisées comme des transferts d'argent ou des modifications de contenu [11].

I.3.6 DNS Spoofing (Usurpation DNS)

L'usurpation de noms de domaine (DNS) est une technique utilisée par des pirates pour modifier des enregistrements DNS afin de rediriger les utilisateurs vers un faux site web. Une fois sur ce site frauduleux, la victime peut entrer des informations sensibles (comme des mots de passe ou des données bancaires) que le pirate pourra ensuite exploiter ou revendre. Le pirate peut aussi créer un site de mauvaise qualité ou avec un contenu choquant afin de nuire à l'image d'une entreprise concurrente.

Lors d'une attaque par usurpation DNS, l'attaquant profite du fait que l'utilisateur pense visiter un site légitime. Cette confiance lui permet de commettre des actions frauduleuses en se faisant passer pour une entreprise innocente, du point de vue de la victime [13].

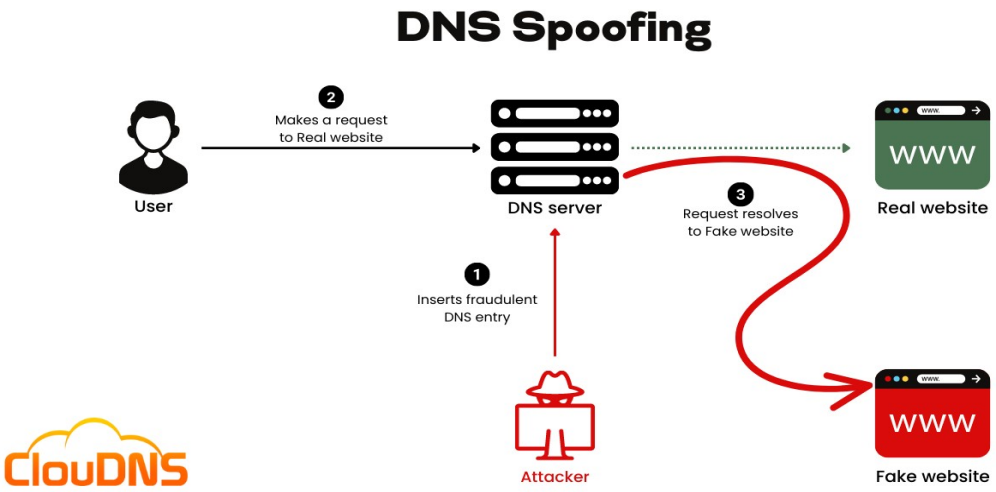


Figure 7: Fonctionnement d'une attaque par DNS Spoofing [10]

Pour se protéger contre ce type d'attaque, il est important de maintenir les serveurs DNS à jour. Les pirates exploitent souvent des failles de sécurité connues dans les anciennes versions des logiciels DNS, alors que les mises à jour corrigent généralement ces vulnérabilités.[13]

I.4. Solutions contre les attaques réseau

I.4.1. IDS

Un système de détection d'intrusion (IDS) est un outil de sécurité qui surveille continuellement le réseau ou les machines afin d'identifier des comportements anormaux ou des tentatives d'attaque. Il permet de repérer rapidement des activités suspectes, d'alerter des administrateurs et de faciliter l'analyse des incidents. Les IDS reposent principalement sur deux approches complémentaires [16] :

- La détection par signature, qui reconnaît les attaques déjà connues et identifiés dans une base de données de signatures.
- La détection par anomalie, qui identifie des écarts par rapport au comportement normal du système.

I.4.2. IPS

Les systèmes de prévention des intrusions (IPS) sont devenus largement reconnus comme des outils puissants et des éléments importants des mécanismes de sécurité informatique. Un IPS est un dispositif capable non seulement de détecter des attaques, qu'elles soient connues et inconnues, mais aussi d'agir pour empêcher leur réussite. Contrairement aux systèmes de détection d'intrusion (IDS), qui se limitent à l'identification des menaces, les IPS disposent de capacités de réponse active permettant de bloquer ou de neutraliser les attaques en temps réel [15].

Les méthodes de détection sont les mêmes mécanismes présentés dans les IDS sauf que les IPS doivent agir en temps réel [15].

I.4.3. Limitation de débit (Rate limiting)

Le rate limiting est un mécanisme fondamental dans la gestion et la sécurisation des API et services web modernes. Cette téchnique permet de contrôler le nombre de requêtes qu'un client peut envoyer pendant une période déterminée afin de garantir la disponibilité et la performance du système. Cette approche contribue à éviter la surcharge des ressources, à assurer une répartition équitable du service entre les utilisateurs et à protéger contre certains abus [17].

La limitation de débit est généralement mise en œuvre au niveau d'une application séparée plutôt qu'au niveau de serveur web. Elle consiste à surveiller les requêtes entrantes, principalement en identifiant leur origine à travers l'adresse IP ou un token / cookie et en analysant la fréquence à laquelle elles sont envoyées. Le système évalue ainsi le nombre de requêtes effectuées durant une période donnée ainsi que les intervalles de temps entre elles. Lorsqu'un seuil prédéfini est dépassé, le mécanisme limite ou bloque

temporairement les nouvelles requêtes provenant de la même source. Parmi les attaques qui peuvent être bloquées par la limitation de débit est le déni de service distribué (DDoS) [18].

I.4.4 Le pare-feu d'application web (WAF)

Le Web Application Firewall (WAF) est un pare-feu applicatif conçu pour protéger les applications web contre les attaques ciblant la couche applicative. Le WAF peut filtrer les paquets, afin de détecter et bloquer les comportements malveillants et effectuer la journalisation des activités (logging).

Un WAF repose sur ModSecurity et la méthode du Reverse Proxy. Le module ModSecurity est placé sur le serveur Reverse Proxy qui permet de cacher le serveur web et d'installer le WAF dessus, et des règles supplémentaires appelées OWASP Core Rule Set qui sont ajoutées pour renforcer la sécurité [19].

I.4.5 VPN (Virtual Private Network)

Un VPN (Virtual Private Network) est une technologie qui permet de créer un canal de communication sécurisé au-dessus d'un réseau public comme Internet. Il repose sur un mécanisme de tunneling qui encapsule les données dans un paquet chiffré afin d'assurer la confidentialité, l'intégrité et l'authentification des échanges entre deux entités distantes.

Les VPN traditionnels utilisent des algorithmes de chiffrement tels que DES (Data Encryption Standard), AES (Advanced Encryption Standard) ou Blowfish pour protéger les données transportées dans le tunnel. Cependant, avec l'évolution des menaces informatiques et l'augmentation de la puissance de calcul, il devient nécessaire de renforcer les mécanismes de protection. Une approche basée sur le chiffrement multi-phase consiste à appliquer plusieurs niveaux de chiffrement successifs sur le contenu transmis du paquet VPN, ce qui augmente considérablement la complexité du chiffrement et rend les tentatives de décryptage beaucoup plus difficiles à accomplir [20].

I.5. comparative des mécanismes de protection

Table 1: Analyse comparative des mécanismes de protection contre les attaques réseau.

Solution	Rôle principal	Couche OSI	Avantage	Limites
IDS	Détection des activités suspectes	Réseau/ Application	Détecte les attaques connues et inconnues	Ne bloque pas les attaques
IPS	Détection + blocage des attaques	Réseau/ Application	Réagit en temps réel	Peut générer des faux positifs
Rate Limiting	Limitation du nombre de requêtes par intervalle de temps	Application	Protège contre DDoS et abus	Basé souvent sur IP (facile à contourner)
WAF	Protection des applications Web	Application	Bloque les attaques XSS, SQL injection	Nécessite la configuratio n des règles
VPN	Sécurisation des communications	Réseau	Assure la confidentialité et l'intégrité	Ne protège pas contre toutes les attaques
SDN	Détection et mitigation dynamiques des attaques grâce au contrôle centralisé	Réseau	Vision globale du réseau, programmabilité , réaction rapide et automatisée	Dépendance au contrôleur SDN, risque de point unique de défaillance

Cette comparaison montre que chaque solution de sécurité à sa propre utilité. Les systèmes IDS et IPS sont essentiels pour la détection et la prévention des intrusions, tandis que la limitation de débit aide à diminuer les attaques de type déni de service. Le WAF protège spécifiquement les applications web et le VPN assure la sécurité des communications. Donc, aucune solution n’est parfaite seule, il conseillé de les utiliser ensemble pour assurer une protection complète.

I.6. Conclusion

Les réseaux informatiques sont aujourd'hui confrontés à des menaces de plus en plus variées et sophistiquées, pouvant impacter leur sécurité, leur performance et leur disponibilité. Pour y répondre, plusieurs mécanismes de protection sont mis en place, allant des solutions classiques comme les pare-feu et les IDS/IPS jusqu'aux approches plus avancées basées sur le SDN. L'étude comparative montre que les solutions traditionnelles restent limitées face à l'évolution rapide des attaques. Ainsi, l'amélioration continue des mécanismes de protection reste indispensable pour garantir la fiabilité des infrastructures modernes. Dans le chapitre suivant, l'adaptabilité et efficacité de la gestion de sécurité réseau avec le paradigme SDN sont illustrées.

Chapitre II: Software Defined Networking (SDN)

II.1. Introduction

Les réseaux traditionnels présentent des limites liées à leur fonctionnement statique, dans lesquelles toute modification doit être effectuée manuellement, rendant la gestion complexe et peu flexible. Face à ces limitations, le Software-Defined-Networking (SDN) a été introduit comme une solution permettant de rendre les réseaux plus dynamiques et facilement administrables [21].

II.2. Définition de SDN

Le SDN (Software-Defined-Networking), qui signifie « réseau défini par logiciel », constitue un nouveau paradigme dans le domaine des réseaux informatiques. Il permet de simplifier et optimiser la gestion des infrastructures réseau en séparant le plan de données, qui est responsable du transfert des paquets, et le plan de contrôle, qui est chargé de prise de décision et de l'administration du réseau. Cette séparation permet un contrôle centralisé et rend le réseau programmable [22].

La figure suivante illustre la différence entre un réseau traditionnel et un réseau SDN :

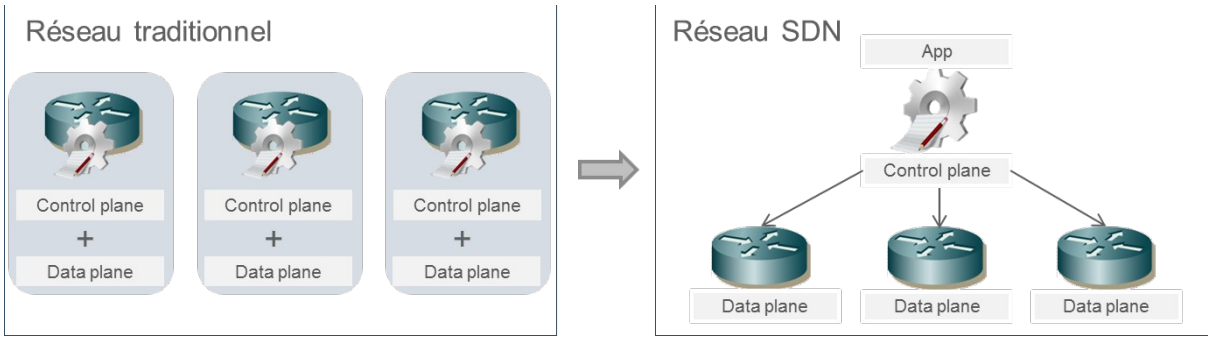


Figure 8: Comparaison entre réseau traditionnel et réseau SDN [23]

Dans le modèle traditionnel, chaque équipement intègre à la fois de plan de contrôle et de plan de données, ce qui rend la gestion plus complexe avec la mise à l'échelle. Tandis que le SDN sépare le plan de contrôle du plan de données, ce qui permet un contrôle centralisé, programmable et plus flexible du réseau.

Un réseau SDN basée sur :

- **La séparation du plan de données et du plan de control:**

  Le plan de données représente la partie du réseau responsable de la transmission des paquets entre les équipements au sein de l'architecture SDN. Dans ce paradigme, les Switchs SDN appliquent les instructions établies par le contrôleur pour diriger les paquets vers leur destination [24].

Le plan de contrôle : assume de définir comment les flux de données doivent être dirigés à travers le réseau. Son rôle principal est de déterminer et modifier les règles présentes dans les tables de flux pour assurer que le plan de données puisse transmettre correctement les paquets. Ce plan est un composant essentiel dans la dynamique de cette architecture [25].

- **Base sur flux :**

Un flux est un ensemble de paquets ayant des caractéristiques et des politiques de service communes, qui se dirige d'une source à une destination. Ainsi, le contrôleur a la capacité de gérer le trafic de manière plus flexible et performante [25].

II.3. Les couches de l'architecture SDN

Dans un réseau traditionnel, l'infrastructure est composée d'équipements d'interconnexion comme les commutateurs (switches) et les routeurs. Ces équipements assurent à la fois la transmission des données et le contrôle du réseau. Dans ce type d'architecture, il est difficile d'ajouter ou de développer de nouveaux services, car le plan de contrôle et le plan de données sont liés.

Pour rendre la gestion des réseaux plus facile et permettre le développement des nouveaux services, l'architecture SDN a été proposée. Cette architecture consiste à séparer le plan de contrôle du plan de transmission dans les équipements réseau. Le SDN est généralement composé de trois couches, ainsi que d'interfaces de communications entre ces couches. Les différentes couches de l'architecture SDN sont :

- **La couche de transmission :** appelée aussi plan de données, comprend les équipements d'acheminements, parmi ceux les Switch SDN. Le rôle principal de cette couche est de transmettre les données et de collecter des statistiques.
- **La couche de contrôle:** elle est composée d'un ou plusieurs contrôleurs SDN. Son rôle est de gérer et de contrôler les équipements du réseau via une interface appelée south-bound API.
- **La couche application :** elle regroupe les applications qui permettent de déployer de nouvelles fonctionnalités réseau, comme la gestion de trafic, la QoS, la sécurité, etc. Ces applications sont développées à l'aide d'une interface de programmation appelée north-bound API.[26]

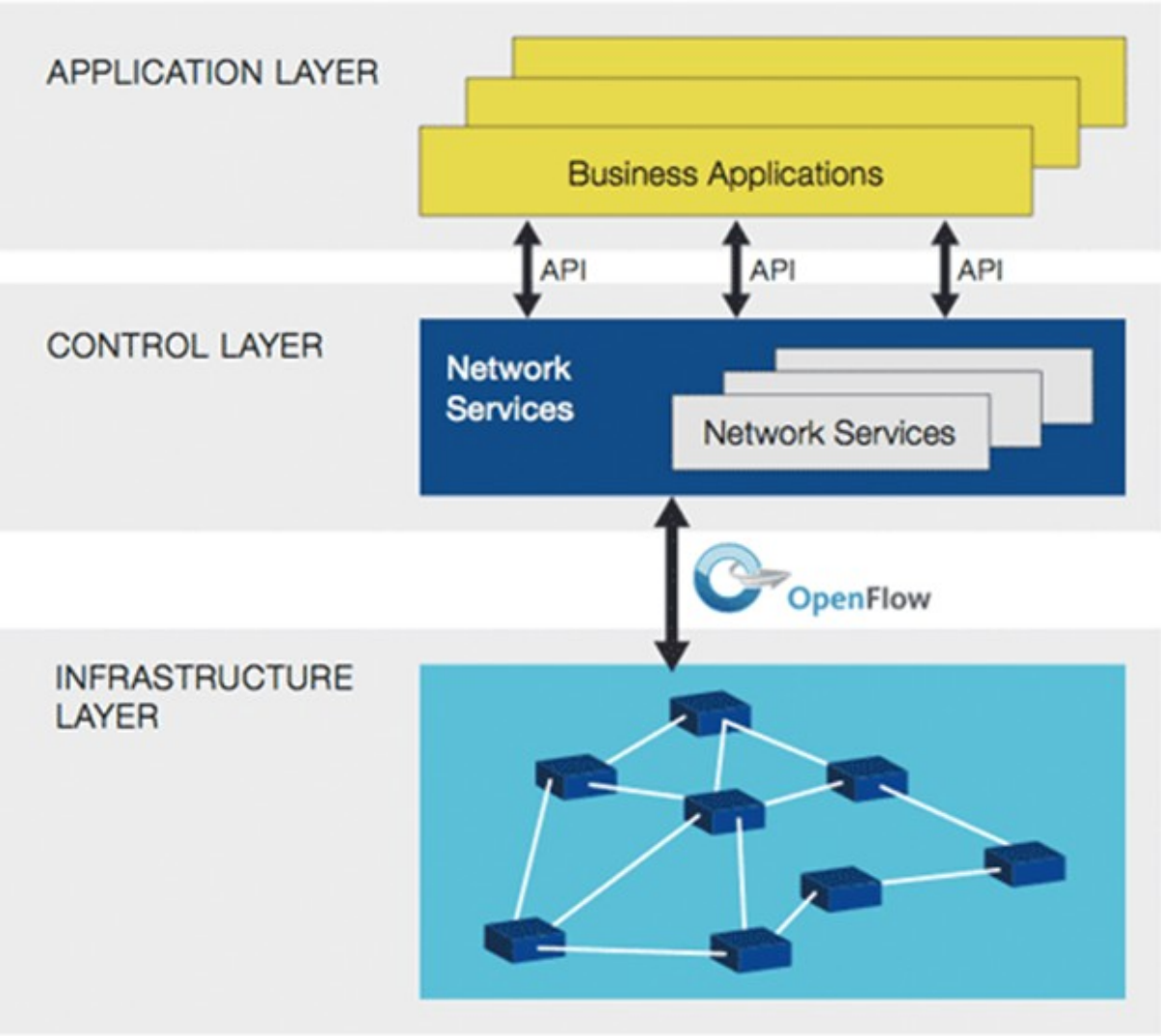


Figure 9: Les couches de l'architecture SDN [27]

II.4. OpenFlow

OpenFlow est un protocole standard qui permet au contrôleur d'un réseau SDN de communiquer avec les switchs afin de configurer leur fonctionnement. Grâce à ce protocole, le contrôleur peut définir ou installer des règles de traitement et d'acheminement des paquets dans les tables de flux des switchs.

Cette approche permet une gestion centralisée du réseau ainsi qu'une meilleure flexibilité dans le contrôle et l'optimisation du trafic [28].

Ce protocole joue un rôle essentiel dans le développement de l'architecture SDN et dans la virtualisation des fonctions réseau. Ce protocole a été initialement développé à Stanford University, ou des chercheurs envisageaient une solution permettant de tester de nouveaux protocoles réseau sans perturber le trafic du réseau de production.

Pour résoudre ce problème, ils ont conçu un mécanisme permettant de séparer le trafic de recherche du trafic de production dans la même infrastructure réseau IP.

II.4.1. Les spécifications OpenFlow

Au fils des années, le protocole Openflow a évolué pour affiner le paradigme SDN. Le tableau suivant liste les changements introduits à ce protocole.

Table 2: Comparaison des différentes versions du protocole Openflow [30]

Version OpenFlow	Additions apportées par la spécification
v1.0	- Les paquets Ethernet et IP sont identifiés selon l'adresse source et de destination. Port source ou destination UDP et TCP. - Champs Ethernet-type et Vlan pour la couche 2. - Champs protocole, DS et ECN pour la couche 3.
v1.1	- Ajout du pipeline, de la table de groupe et de métadonnée - Ajout des champs d'identification MPLS
v1.2	- Model OXM (openflow extensible match) qui apporte une support IPv6 et une structure de correspondance plus flexible - Un switch peut à présent être connecté à plusieurs contrôleurs en mode Master/Slave
v1.3	- Ajout de la table de mesure qui mesure le taux de paquet assignés à une entrée. - Amélioration du support des contrôleurs multiples.
v1.4	- Amélioration d'OXM et de la structure TLV ce qui apporte un gain de synchronisation.
v1.5	- Notion des tables de sortie qui permet de traiter les paquets sur le port de sortie en même temps que les ports d'entrée.

II.4.2 Structure d'un commutateur OpenFlow

Les commutateurs Ethernet et les routeurs modernes utilisent des tables de flux pour décider comment transmettre les paquets, en se basant sur les informations des couches 2, 3 et 4 (comme l'adresse MAC, les adresses IP ou les ports).

Même si chaque constructeur a sa propre manière de gérer ces tables, il existe des fonctions communes. Le protocole OpenFlow exploite ces fonctions pour permettre à un contrôleur central de programmer le comportement du commutateur [29].

Les fonctions principales du commutateur sont [29] :

- Fonction de contrôle :
  - Le commutateur communique avec le contrôleur SDN.
  - Le contrôleur peut ainsi gérer et programmer le commutateur à distance grâce à OpenFlow.
- Fonction d'acheminement des données: Le commutateur reçoit des paquets et les achemine vers la bonne destination en suivant des règles définies par le contrôleur.

Un switch SDN est composé d'une API pour communiquer avec le contrôleur, une couche d'abstraction qui consiste en un pipeline des tables des flux et une fonction de traitement de paquets [30].

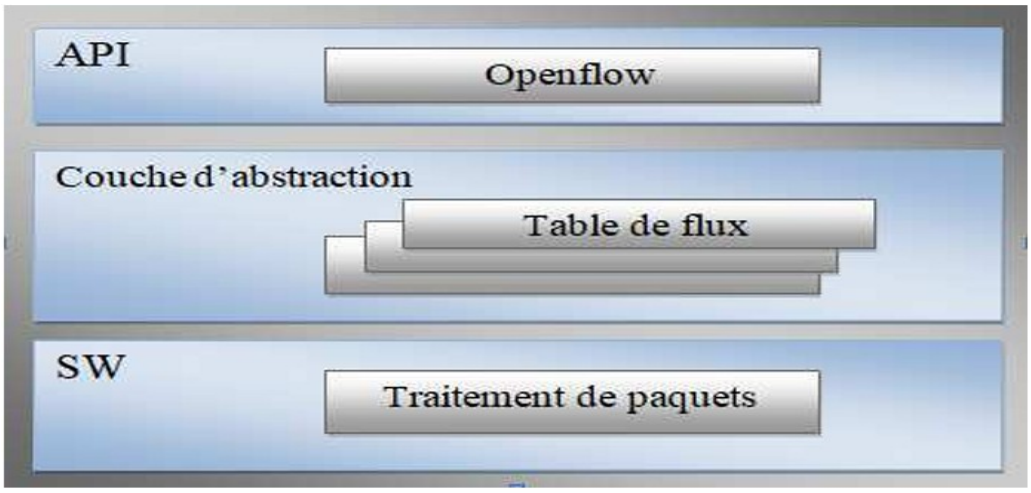


Figure 10: Anatomie logique d'un commutateur SDN [30]

Il existe deux types principaux de commutateurs OpenFlow [31]:

- Commutateur pure : Ce type de commutateur fonctionne exclusivement avec le protocole OpenFlow. Tous les paquets reçus sont obligatoirement traités via le pipeline OpenFlow, sans la possibilité d'utiliser les mécanismes de commutation réseau traditionnels.
- Commutateur hybride : Ce type combine le fonctionnement OpenFlow avec les mécanismes classiques des réseaux (comme la commutation ethernet, le routage IP, VLAN).

II.4.3 Table de flux

À partir de la version 1.1, les commutateurs SDN reposent sur un pipeline composé de plusieurs tables de flux. Ce mécanisme permet de gérer efficacement le cheminement des paquets dans le réseau. Il consiste à cumuler les actions de chaque table de flux pour les exécuter à la sortie du pipeline [30].

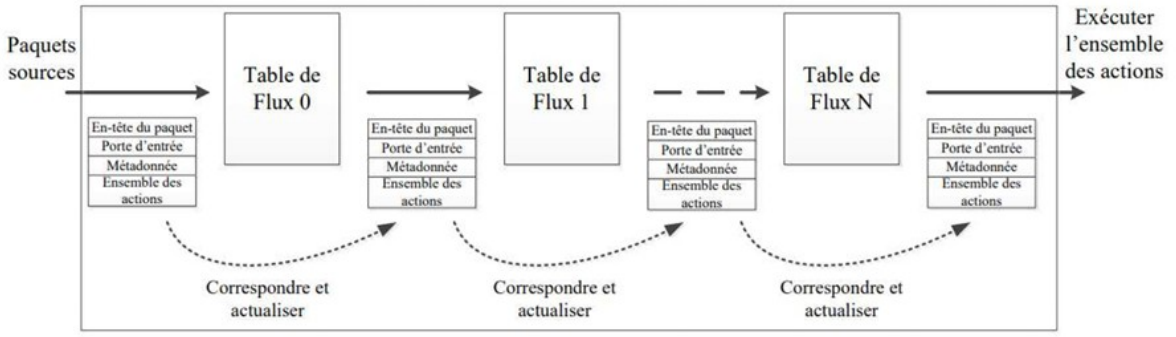


Figure 11: Pipeline OpenFlow v1.1 [32]

Nous allons maintenant présenter les différents champs qui constituent les entrées des tables de flux.

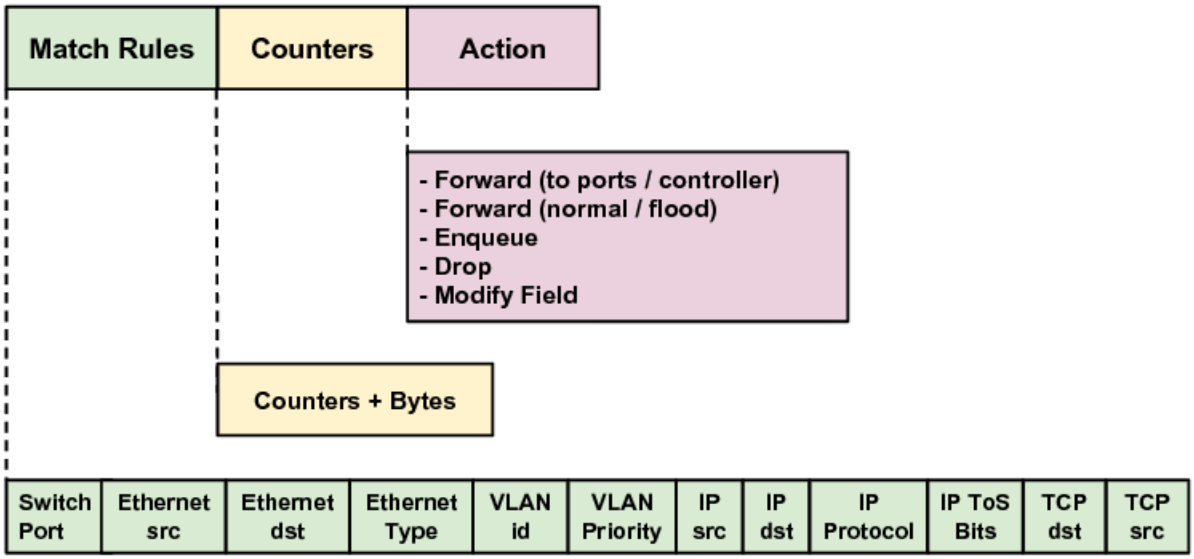


Figure 12: Architecture de la table de flux OpenFlow v1.0.0 [33]

II.4.3.1 Champs de correspondance

Ces champs sont utilisés pour rechercher le flux correspondant au paquet. Cette identification repose sur plusieurs informations présentes dans les en-têtes du paquet, allant de la couche 1 à la couche 4 de modèle OSI. La figure ci-dessus illustre quelques champs utilisés par le protocole. Dans les versions récentes, de nouveaux éléments d'identification ont été ajoutés, notamment les champs IPV6 et les étiquettes MPLS.[30]

II.4.3.2 Compteurs

Les compteurs servent à collecter les statistiques relatives aux flux. Tels que le nombre de paquets appartenant au flux, le volume de données transporté dans le flux. Ces derniers contribuent également à la gestion des entrées des tables de flux, notamment en permettant de déterminer si une entrée est toujours active ou

non. Des compteurs de statistiques sont maintenus pour chaque table, chaque flux et chaque port.[30]

II.4.3.3 Actions

Chaque entrée d'une table de flux est associée à un ensemble d'actions appliquées aux paquets avant leur transmission vers un port de sortie. Ces actions sont exécutées dans l'ordre défini dans la table. Lorsqu'aucune action n'est spécifiée, le paquet correspondant est automatiquement supprimé.

Un switch SDN peut ne pas prendre en charge toutes les actions possibles. Mais il doit supporter les actions requises pour le fonctionnement minimal d'OpenFlow :

- La retransmission des paquets sur un ou plusieurs ports de sortie afin d'assurer leur acheminement dans le réseau.
- L'encapsulation et l'envoi des paquets vers le contrôleur via un canal sécurisé, notamment lors de la détection d'un nouveau flux.
- La suppression des paquets appartenant à un flux est aussi une action qui peut être utilisée pour une raison de sécurité afin de mettre fin à une attaque par exemple.

Les actions optionnelles consistent à modifier certains champs du paquet, tels que l'identifiant VLAN, l'adresse MAC ou encore le TTL, entre autres.

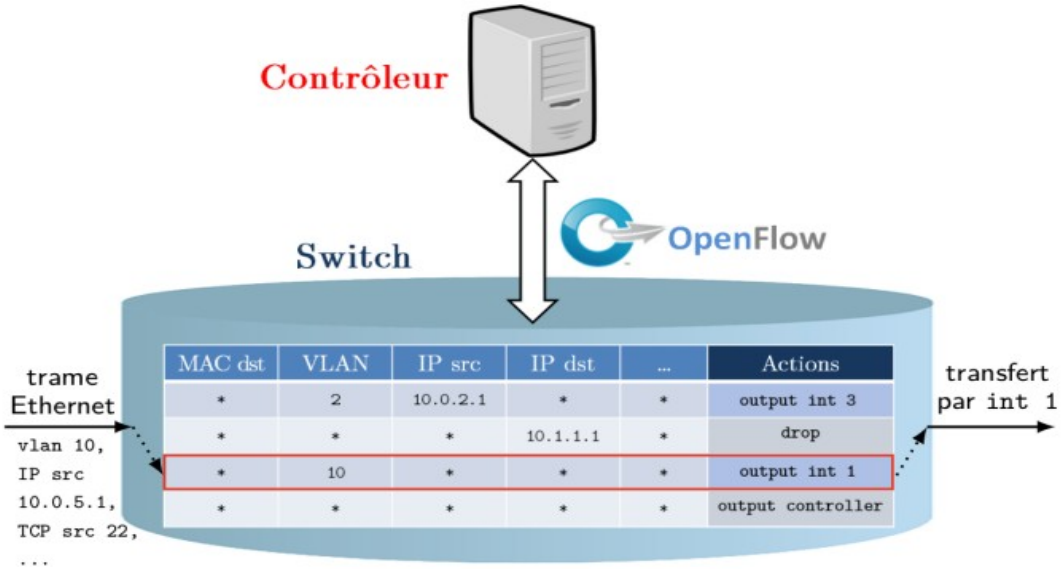


Figure 13: Schématisation de la table de flux [30]

II.4.4 Les messages OpenFlow

Chaque message commence par une entête OpenFlow spécifiant la longueur, l'identificateur de transaction et le type de message. Lorsqu'un commutateur dispose de l'adresse IP de contrôleur, il initie une connexion TLS sécurisée via TCP afin de permettre l'échange des messages [30].

Le protocole OpenFlow distingue trois types de messages :

- Contrôleur-Commutateur
- Symétrique
- Asynchrone

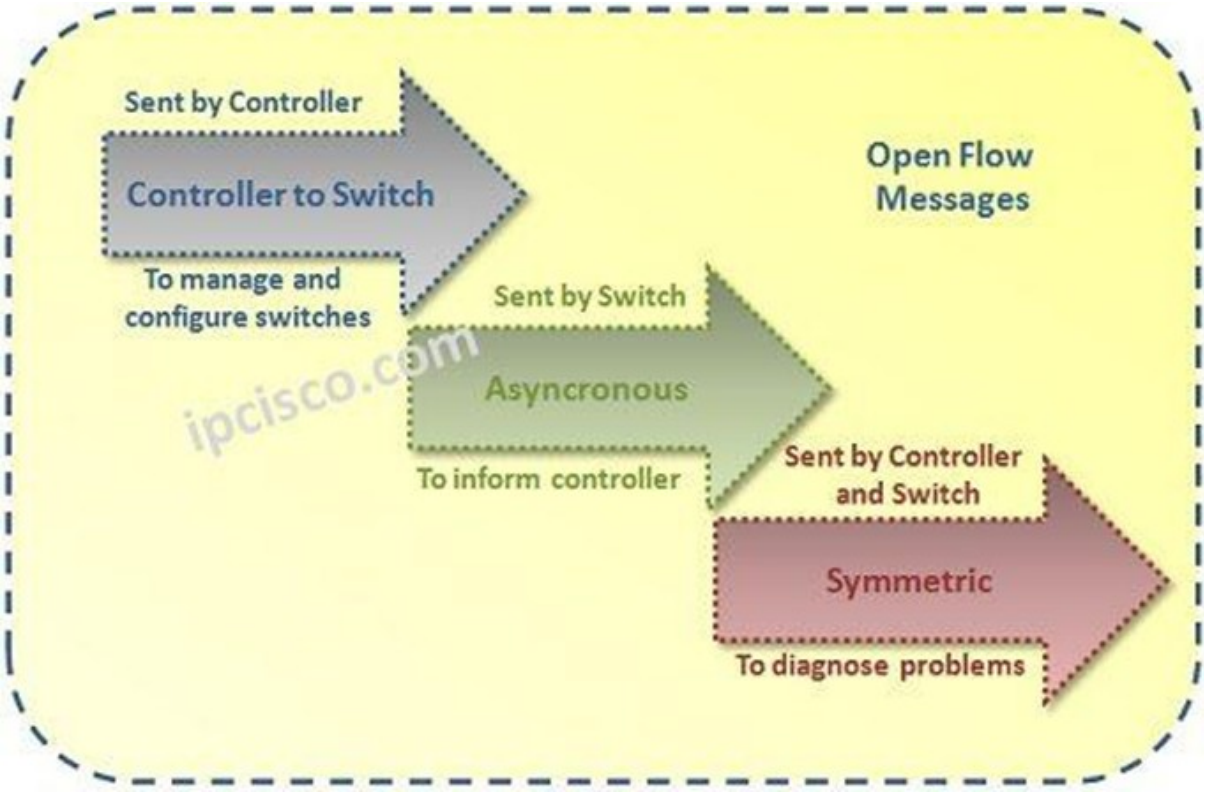


Figure 14: Les messages OpenFlow [35]

II.4.4.1 Messages Contrôleur-Commutateur

Les messages contrôle du commutateur sont initiés par le contrôleur dans le but de gérer et surveiller l'état du commutateur. Certains de ces messages nécessitent une réponse, d'autres non. On peut classer ce type de message en plusieurs catégories[34] :

- **Features** : Le contrôleur envoie une requête pour connaître les capacités de switch et son identité. [34]
- **Configuration** : Le contrôleur peut définir et interroger les paramètres de configuration de commutateur. [34]
- **Modify-state** : Ces messages permettent de gérer l'état des switch, notamment pour ajouter, modifier ou supprimer des entrées dans les tables de flux. [30]
- **Read-state** : Utilisé par le contrôleur pour collecter diverses informations sur le switch, telles que ses statistiques, ses capacités et sa configuration actuelle. [30]

- **Paquet-out** : Ces messages permettent au contrôleur de demander au switch de transmettre un paquet sur une interface définie, soit en incluant le paquet entier, soit en utilisant l'ID du paquet dans le buffer stocké dans le switch. [30]
- **Barrier** : Messages de type requête/réponse utilisés pour s'assurer les dépendances entre messages sont respectées et d'obtenir la confirmation de l'exécution des opérations demandées. [34]

II.4.4.2 Messages symétriques

Les messages symétriques peuvent être émis soit par le commutateur ou le contrôleur, sans qu'une requête préalable soit nécessaire. Ces messages sont divisés en trois types :

- **Hello** : Ces messages sont échangés lors de l'établissement initial de connexion entre le commutateur et le contrôleur afin de vérifier la communication. [34]
- **Echo** : Les messages Echo Request et Echo Reply peuvent être initiés par chacune des deux entités. Leur fonction consiste à vérifier l'état de la connexion en continu entre contrôleur et switch SDN et peuvent également donner des indices sur la latence ou la qualité de lien [34]. Chaque message ECHO_REQUEST doit être acquitté par un ECHO_REPLY [30].
- **Vendor** : Ces messages permettent d'ajouter des fonctionnalités supplémentaires spécifiques aux constructeurs, offrant ainsi une certaine flexibilité pour étendre les capacités de OpenFlow [34].

II.4.4.3 Message asynchrone

Les messages asynchrone sont transmis par le commutateur vers le contrôleur afin de l'informer des événements survenus dans le réseau. Ils servent notamment à notifier l'arrivée de paquets, des changements ou des erreurs. Les principaux types sont :

- **Packet-in**: Ce type de message est envoyé lorsqu'un paquet ne correspond à aucune entrée dans la table de flux ou lorsqu'une règle demande explicitement de transmettre le paquet au contrôleur. Selon ses capacités, le commutateur peut placer le paquet dans le buffer et transmettre l'identifiant du paquet au contrôleur ou transmettre paquet entier dans le message Packet-In[34].
- **Flow-Removed** : Ce message informe le contrôleur qu'une entrée de flux a été supprimée [30].

- **Port-status** : Le commutateur émet ce message afin de signaler tout modification relative à l'état ou la configuration d'un port (activation, désactivation, etc). Ces changements peuvent être également liés au protocole Spanning Tree (802.1D), qui gère la topologie du réseau [34].
- **Error** : Le commutateur peut informer le contrôleur qu'il existe des problèmes à l'aide de messages d'erreur [34].

II.4. État de l'art des solutions de sécurité SDN

Les réseaux SDN permettent d'implémenter des solutions de sécurité grâce à leur architecture centralisée et programmable. Cette technologie offre une meilleure visibilité et un contrôle dynamique du trafic. Cela facilite la détection et la prévention des cyberattaques.

Les applications de sécurité dans les SDN se regroupent principalement en trois catégories : le filtrage et blocage du trafic, la défense contre les malwares, et les mécanismes de tromperie. Ces approches permettent de répondre aux principales menaces telles que les attaques DDoS, les intrusions et les logiciels malveillants.

Rezaei et Hashemi [46], ainsi que Vempati et al. [47], ont proposé des mécanismes de filtrage dynamique basés sur le contrôleur SDN permettant d'analyser le trafic en temps réel et d'appliquer des règles de sécurité. Ces approches permettent de bloquer automatiquement les flux suspects et d'atténuer les attaques comme DDoS.

Javeed et al. [48] et Ullah et al. [49] ont intégré des algorithmes de deep learning dans le contrôleur SDN pour détecter les anomalies du trafic réseau. De plus, Kokila et al. [50] ont utilisé des modèles SVM pour distinguer le trafic légitime du trafic malveillant, permettant ainsi une détection proactive des attaques.

Xing et al. [51] ainsi que Lin, H. [52] ont proposé l'intégration de honeypots dans des environnement SDN afin d'attirer les attaquants vers des système leurres. Cette approche permet d'analyser des comportement malveillants sans impacter le réseau réel et d'améliorer les mécanismes de défense.

Kumar et al. [53] ont proposé une approche basée sur l'algorithme K-means pour analyser le dataset NSL-KDD dans le contexte de la détection d'intrusion. Leurs méthodes permettent de regrouper les flux réseau en clusters représentant les comportements normaux et les différents types d'attaques, démontrant ainsi l'efficacité des techniques de clustering non supervisé pour l'identification des anomalies réseau.

Cependant, malgré ces avancées, plusieurs limites persistent, surtout le fait de s'orienter vers la disponibilité des ressources, au lieu de protéger la confidentialité et l'intégrité, ainsi que les risques liés à la centralisation du contrôleur SDN. Cela nécessite le développement de solution plus complexes et intelligentes.[37]

II.5. Conclusion

SDN introduisent une nouvelle approche basée sur la séparation du plan de contrôle et du plan de données, offrant plus de flexibilité et une gestion centralisée du réseau. L'architecture SDN, associée au protocole OpenFlow, permet un contrôle dynamique a travers les tables de flux et les échanges de messages entre le contrôleur et les équipements. Malgré ces avantages, cette technologie présente des défis importants en matière de sécurité, ce qui nécessite la mise en place de mécanismes de protection adaptés pour garantir un fonctionnement fiable et sécurisé.

Chapitre III : Conception et implémentation d'une plateforme de sécurité programmable multi-services basée sur SDN

III.1. Introduction

Ce projet de fin d'étude porte sur la conception et l'implémentation d'une plateforme de sécurité multi-services basée sur SDN, visant à améliorer la détection et la mitigation dynamique des attaques réseau. La solution proposée s'appuie sur le protocole OpenFlow pour le contrôle des commutateurs et sur un contrôleur SDN centralisé, permettant l'installation dynamique de règles de sécurité en fonction du comportement du trafic. Pour réaliser ce projet, nous avons utilisé des outils et des technologies adaptés.

III.2. Outils

- **Oracle VM VirtualBox :**
  Oracle VM VirtualBox est un logiciel de virtualisation open source et multi-plateforme permettant d'exécuter simultanément plusieurs systèmes d'exploitation sur une seule machine. Celui-ci est utilisé par les développeurs pour tester et valider leurs applications dans différents environnements, et par les équipes informatiques pour réduire les coûts et accélérer le déploiement des solutions, que ce soit en local ou dans le cloud. compatible avec plusieurs système hôtes tels que Windows, macOS, Linux et solaris, il constitue un outil efficace pour le développement, les tests et la démonstration des solutions multi-plateformes.[38]
- **Ubuntu 22.04 :**
  est un système d'exploitation Linux open source, stable et facile à utiliser, adapté au développement, à l'administration et l'utilisation quotidienne. [39]
  Dans le cadre de ce projet, nous avons utilisé Ubuntu 22.04 comme environnement de travail principal, sur lequel nous avons installé et configuré les différents outils nécessaires, notamment Mininet ainsi que le contrôleur SDN, afin de réaliser les expérimentations et les tests.
- **Open vSwitch :**
  Open vSwitch (OVS) est un commutateur réseau logiciel multicouche open source, conçu pour fonctionner principalement dans des environnements virtuels. Il permet de gérer et de contrôler le trafic réseau entre les machines virtuelles ainsi qu'entre celles-ci et le réseau physique. OVS prend en charge des interfaces de gestion standard et offre la possibilité de programmer et d'automatiser le fonctionnement du réseau

grâce à des protocoles comme OpenFlow, ce qui le rend particulièrement adapté aux architectures SDN. [40]

- **Mininet :**

  Mininet est un émulateur permettant de créer et tester des architectures réseau sur un simple ordinateur ou une machine virtuelle. Il est principalement utilisé dans SDN et supporte le protocole OpenFlow. Cet outil permet d'exécuter des applications réseau réelles sur des équipements virtuels tels que les hots, les commutateurs et les contrôleurs.

  Celui-ci se caractérise par sa simplicité d'utilisation, sa flexibilité et sa capacité de reproduire des environnement réseau proche du réel. Il permet également de créer des topologies de grande taille et de tester facilement différents scénarios. Enfin, il constitue une solution idéale pour le prototypage rapide et l'expérimentation en réseau [41].

- **MiniEdit :**

  MiniEdit est une interface graphique de mininet qui permet de créer et configurer des topologies réseau de manière visuelle et intuitive. Il facilite la conception des réseaux SDN en utilisant des éléments graphiques comme les hôtes, les switchs et les liens, sans nécessiter de programmation [42].

- **Python :**

  Python est un langage de programmation open source très utilisé dans plusieurs domaines comme l'analyse de données, la gestion d'infrastructure et le développement logiciel. Il est apprécié pour sa simplicité, ce qui permet aux développeurs de coder plus rapidement en se concentrant sur la logique plutôt que sur la syntaxe.

  Il est aussi accessible aux débutants grâce à de nombreux tutoriels en ligne et une grande communauté qui facilite l'apprentissage et la résolution des problèmes. [43]

- **Pox :**

  POX est un contrôleur SDN basé sur OpenFlow, développé en Python. Il est principalement utilisé pour le prototypage et la recherche. Sa structure simple le rend facile à utiliser dans les tests SDN. Contrairement à d'autres contrôleurs, il n'est pas destiné aux réseaux de production.

  POX repose sur un modèle basé sur des composants indépendants. Ces composants peuvent être utilisés et réutilisés selon les besoins. Le contrôleur se place entre les applications et les équipements réseau. Il

assure la communication entre les applications et les switches. POX utilise le protocole OpenFlow pour gérer le trafic réseau. Ses modules Python permettent d'ajouter des fonctionnalités réseau facilement.[44]
POX est utilisé pour implémenter des pare-feu dans les réseaux SDN afin d'appliquer des règles de sécurité et filtrer le trafic.[45]

III.3. Simulation avec Mininet

```python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel

class MyTopo(Topo):
    def build(self):

        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')

        h1 = self.addHost('h1', ip='10.0.0.1/24')
        h2 = self.addHost('h2', ip='10.0.0.2/24')
        h3 = self.addHost('h3', ip='10.0.0.3/24')
        h4 = self.addHost('h4', ip='10.0.0.4/24')
        h5 = self.addHost('h5', ip='10.0.0.5/24')
        h6 = self.addHost('h6', ip='10.0.0.6/24')
        h7 = self.addHost('h7', ip='10.0.0.7/24')
        h8 = self.addHost('h8', ip='10.0.0.8/24')

        self.addLink(h1, s1)
        self.addLink(h2, s1)

        self.addLink(h3, s2)
        self.addLink(h4, s2)

        self.addLink(h5, s3)
        self.addLink(h6, s3)

        self.addLink(h7, s4)
        self.addLink(h8, s4)

        self.addLink(s1, s2)
        self.addLink(s2, s3)
        self.addLink(s3, s4)

def run():
    topo = MyTopo()
    net = Mininet(
        topo=topo,
        controller=lambda name: RemoteController(name, ip='127.0.0.1', port=6633)
    )
    net.start()

    print("=== TEST PING ===")
    net.pingAll()

    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    run()
```

Figure 15: Création d'une topologie personnalisée

III.3.1 Création de topologie dans SDN

Voici quelque explications du code :

- **Topo :** une classe de base pour les topologies Mininet.
- **addSwitch() :** ajouter un switch a la topologie, son paramètre correspond au nom de switch à créer.
- **addHost(nom, **params) :** elle permet de créer un hôte dans la topologie.
  - nom (obligatoire) : nom du host (h1, h2,...).
  - IP (optionnel) : définit l'adresse IP de host.
- **addLink(host, switch) ou addLink(switch,switch) :** permet d'ajouter des liens entre host et switch ou bien switch et switch.
- **RemoteController(...) :** permet de déclarer le contrôleur POX au quel les switch SDN doivent se connecter.
- **net.pingAll() :** teste la connectivité entre tous les hôtes.
- **CLI(net) :** ouvre l'interface de commandes Mininet.

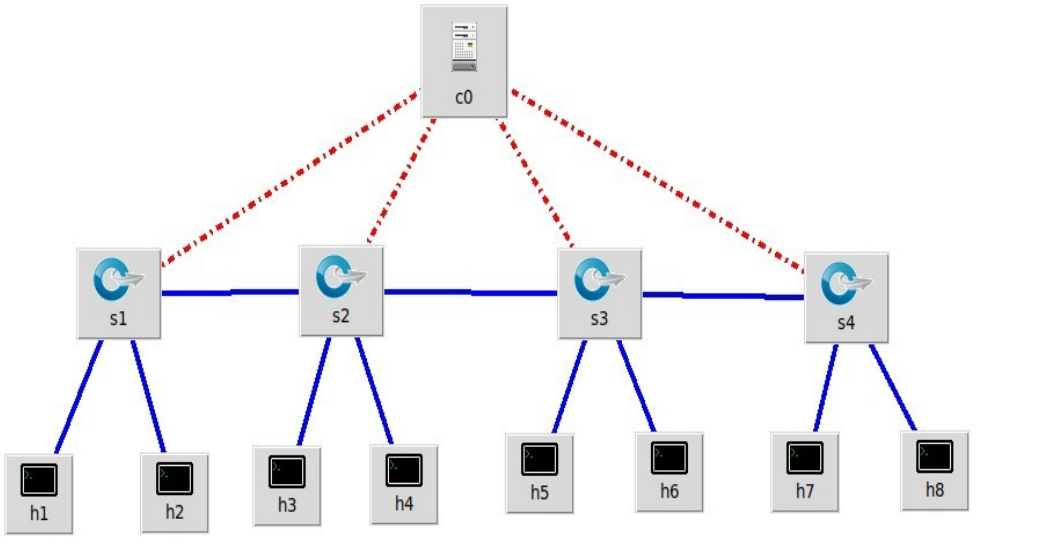


Figure 16: Topologie réseau émulée

Cette topologie présente une architecture de Data Center à trois niveaux (3-tier), composée de switchs interconnectée et d'un contrôleur SDN centralisé. Les hôtes (h1 à h8) sont connectés aux switchs d'accès (s1 à s4), formant la couche d'accès qui assure la communication avec les utilisateurs. Les switchs sont reliés entre eux pour constituer une couche d'agrégation, permettant de regrouper et de gérer le trafic réseau. Le contrôleur (c0) joue le rôle de couche de cœur en

centralisant le contrôle et en prenant les décisions de routage. Cette organisation améliore la performance, la scalabilité et la gestion du réseau.

III.3.2 Implémentation du code de contrôleur POX

```
  GNU nano 6.2
from pox.core import core
import pox.openflow.libopenflow_01 as of
from collections import defaultdict

log = core.getLogger()

mac_to_port = defaultdict(dict)

def _handle_ConnectionUp(event):
    log.info("Switch %s connecté" % event.dpid)

def _handle_PacketIn(event):
    packet = event.parsed
    if not packet.parsed:
        log.warning("Paquet non parsé")
        return

    dpid = event.connection.dpid
    src = str(packet.src)
    dst = str(packet.dst)
    in_port = event.port

    mac_to_port[dpid][src] = in_port

    if dst in mac_to_port[dpid]:
        out_port = mac_to_port[dpid][dst]
    else:
        out_port = of.OFPP_FLOOD

    msg = of.ofp_packet_out()
    msg.data = event.ofp
    msg.actions.append(of.ofp_action_output(port=out_port))
    msg.in_port = in_port

    event.connection.send(msg)

def launch():
    core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
    core.openflow.addListenerByName("PacketIn", _handle_PacketIn)
    log.info("Learning Switch Controller compatible Python 3.10 démarré")
```

Figure 17: Script de contrôleur POX

Voici quelque explications du code :

- **_handle_ConnectionUp(event) :** traite l'événement de connexion d'un switch au controlleur.
- **_handle_PacketIn(event) :** traite les message Openflow reçus de type Packet-In. Dans cet exemple, cette fonction apprend les adresse MAC pour simuler le fonctionnement d'un switch L2.
- **mac_to_port[src] = in_port :** créer une table MAC pour savoir par quel port renvoyer les paquets.
- **OFPP_FLOOD :** type d'action Openflow pour instruire le switch d'envoyer un paquet sur tous les ports sauf celui d'entrée.
- **launch() :** fonction appelée par POX pour démarrer le contrôleur.

III.4. Simulation d'attaque DDoS

L'objectif de cette simulation est de reproduire une attaque DDoS en générant un trafic massif à l'aide de l'outil hping3, afin d'observer son impact sur le réseau.

Nous avons lancé quatre hôtes (h2,h3,h4,h5) utilisés comme attaquants ciblant le serveur victime h1.

Avant de lancer l'attaque, nous avons testé la connectivité vers l'hôte h1 à l'aide de la commande ping. Les résultats montrent que le temps de réponse est faible et qu'aucune perte de paquets n'est observée.

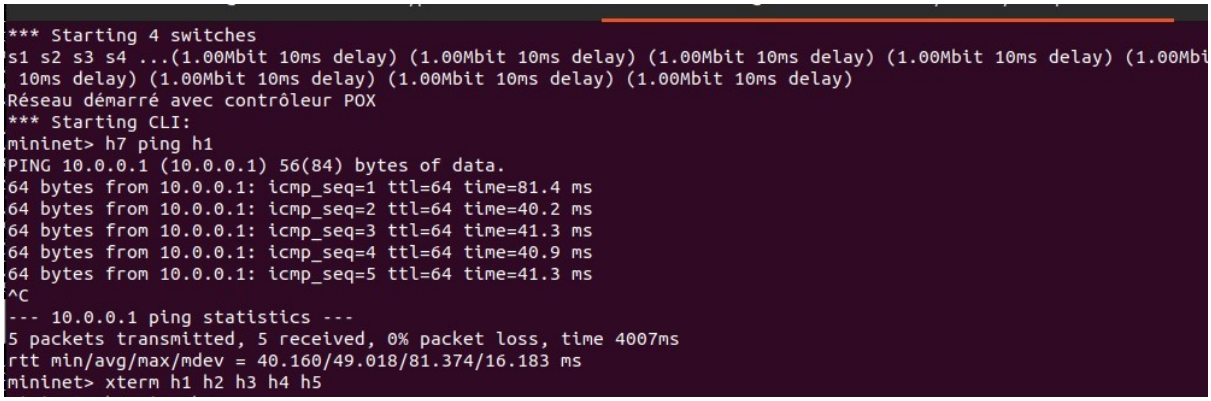


Figure 18: Test de ping avant l'attaque

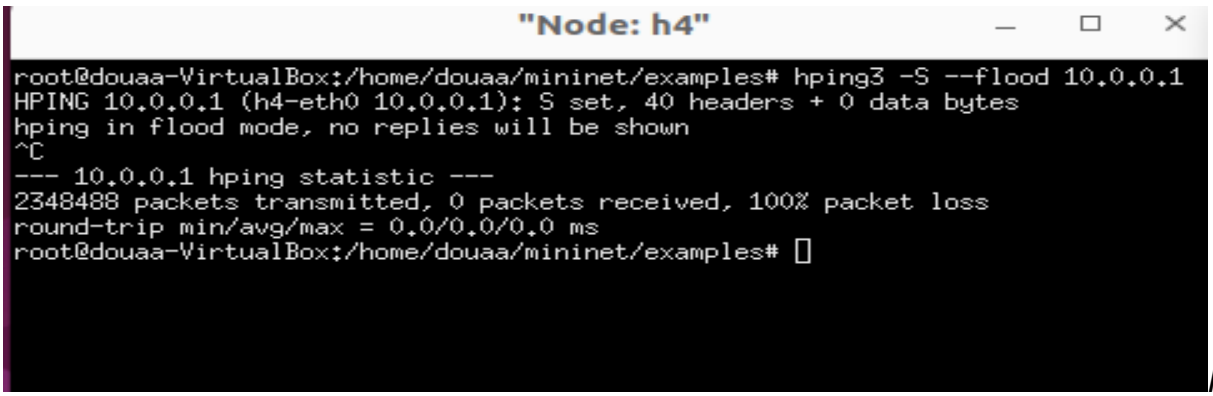


Figure 19: Lancement de l'attaque DDoS

hping3 est un outil de génération de paquets permettant de simuler des attaques comme le SYN flood.

- *-S* : envoi de paquets SYN.
- *–flood* : envoi massif de paquet.
- *10.0.0.1* : la victime.

En mode *flood*, l'outil génère un trafic massif en envoyant les paquets de manière continue et à très haute fréquence, ce qui permet de saturer les ressources du réseau et de la cible.

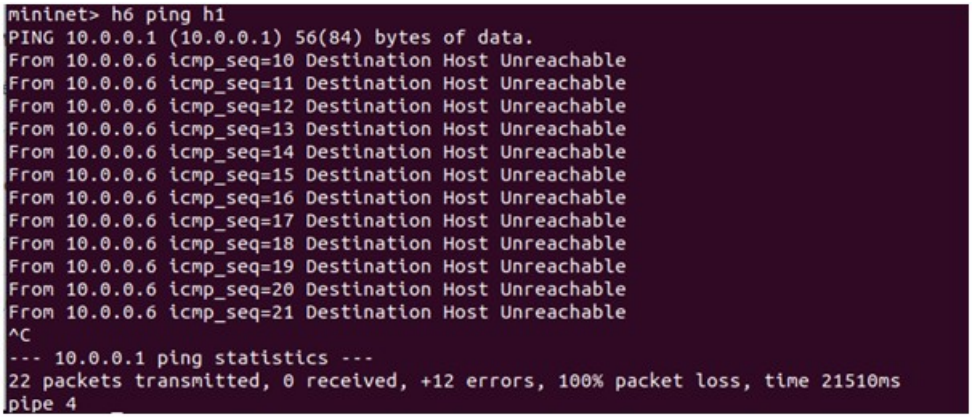
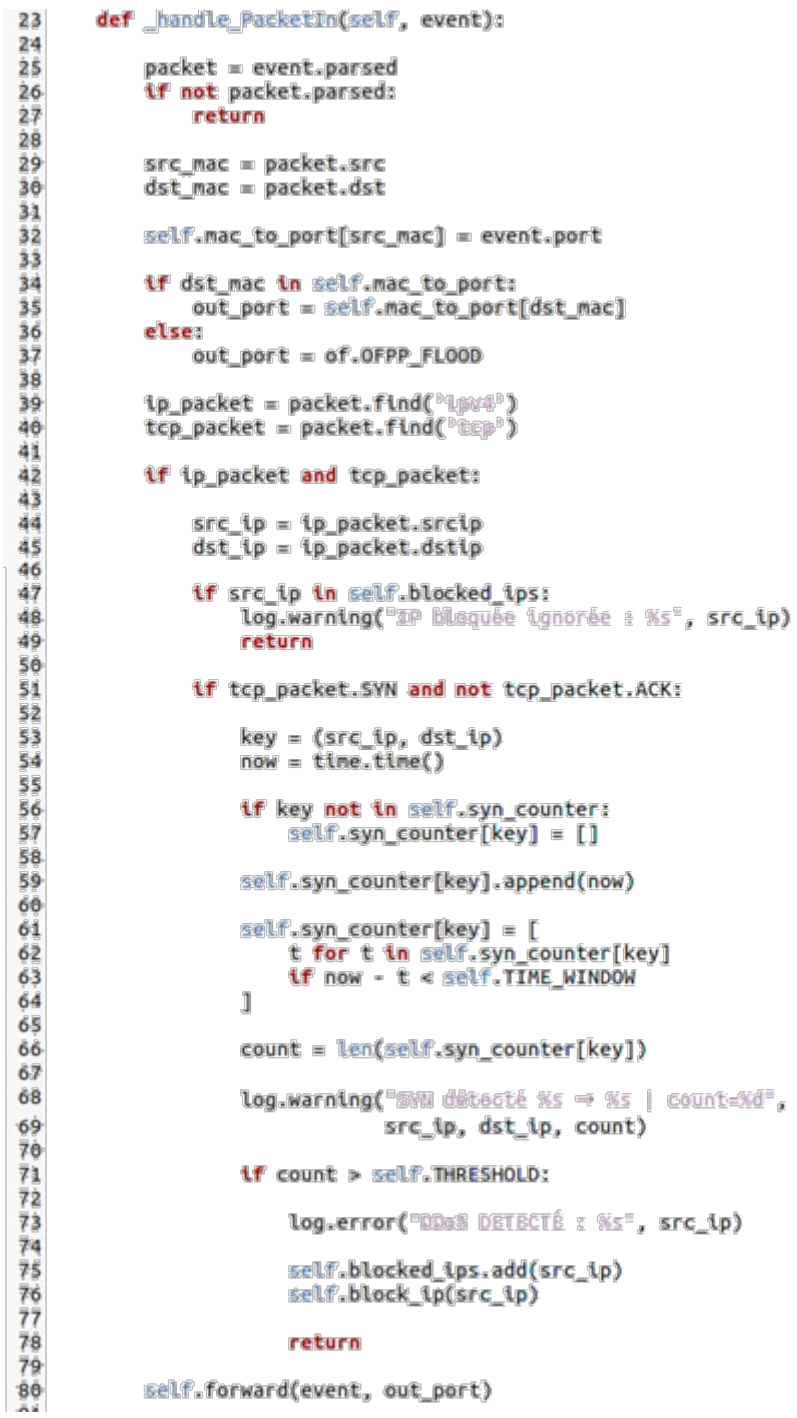


Figure 20: Ping pendant l'attaque DDoS

Pendant l'attaque DDoS de type SYN flood, la machine cible ne répond plus aux requêtes ping. Les messages Destination Unreachable et la perte de paquets à 100 % indiquent que la victime est devenue indisponible. Cela s'explique par l'envoi massif de requête SYN, ce qui empêche le traitement normal du trafic légitime. Cette situation confirme l'impact de l'attaque sur la disponibilité du service.

III.5. Détection et mitigation d'attaque DDoS

Dans cette partie, nous mettons en œuvre un module de sécurité au niveau du contrôleur POX pour détecter l'attaque DDoS qui s'appuient sur les requêtes TCP SYN. Le système réalise une analyse en temps réel du trafic réseau. Lorsque le volume de trafic dépasse un seuil prédéfini, un mécanisme de blocage est automatiquement activé.



```
    def _handle_PacketIn(self, event):

        packet = event.parsed
        if not packet.parsed:
            return

        src_mac = packet.src
        dst_mac = packet.dst

        self.mac_to_port[src_mac] = event.port

        if dst_mac in self.mac_to_port:
            out_port = self.mac_to_port[dst_mac]
        else:
            out_port = of.OFPP_FLOOD

        ip_packet = packet.find('ipv4')
        tcp_packet = packet.find('tcp')

        if ip_packet and tcp_packet:

            src_ip = ip_packet.srcip
            dst_ip = ip_packet.dstip

            if src_ip in self.blocked_ips:
                log.warning("IP bloquée ignorée : %s", src_ip)
                return

            if tcp_packet.SYN and not tcp_packet.ACK:

                key = (src_ip, dst_ip)
                now = time.time()

                if key not in self.syn_counter:
                    self.syn_counter[key] = []

                self.syn_counter[key].append(now)

                self.syn_counter[key] = [
                    t for t in self.syn_counter[key]
                    if now - t < self.TIME_WINDOW
                ]

                count = len(self.syn_counter[key])

                log.warning("SYN détecté %s => %s | count=%d",
                            src_ip, dst_ip, count)

                if count > self.THRESHOLD:

                    log.error("DDoS DETECTÉ : %s", src_ip)

                    self.blocked_ips.add(src_ip)
                    self.block_ip(src_ip)

                    return

        self.forward(event, out_port)
```

Figure 21: Module de sécurité dans POX

- Le module de sécurité analyse chaque evenement Packet-In reçu à travers la fonction *_handle_PacketIn(event)*
- Les segments avec drapeau SYN allumé sont enregistrés avec *self.syn_counter[key].append(time.time())* afin de compter le nombre de tentatives.
- Le contrôleur compare le nombre de tentatives avec le seuil défini en utilisant *if count > self.THRESHOLD.*

- En cas de dépassement du seuil, l'attaque est détectée et l'adresse IP est bloquée avec *self.block_ip(src_ip)*.
- Nous avons ajouté une règle de flux de type DROP dans le switch pour bloquer tout le trafic provenant de cette IP.

Après avoir lancé l'attaque, la journalisation suivante apparaissent dans POX:

```
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=1
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=2
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=3
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=4
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=5
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=6
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=7
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=8
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=9
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=10
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=11
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=12
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=13
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=14
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=15
WARNING:myController:SYN détecté 10.0.0.2 →10.0.0.1 | count=16
ERROR:myController:DDoS DETECTÉ : 10.0.0.2
ERROR:myController:IP BLOQUÉE : 10.0.0.2
```

Figure 22: L'affichage dans POX

Lors du lancement de l'attaque, le contrôleur POX affiche des messages indiquant la détection de paquets SYN répétés entre l'adresse source (l'attaquant) et destination (la victime) .

Le compteur (count) augmente progressivement, ce qui montre un trafic anormal. Dès que le seuil de détection est atteint, le système identifie une attaque DDoS, génère une alerte (DDoS détecté) et applique un mécanisme de mitigation consistant à bloquer les adresses IP source des attaquants.

```
mininet> h6 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=70.0 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.643 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.046 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.084 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.052 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.046 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.045 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.044 ms
^C
--- 10.0.0.1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7168ms
rtt min/avg/max/mdev = 0.044/8.869/69.992/23.103 ms
```

Figure 23: Ping avec le module de sécurité

Pendant l'attaque, nous avons testé la connexion vers la machine victime avec un ping. Les résultats montrent que tous les paquets sont reçus sans perte (0% de perte), ce qui signifie que le service fonctionne normalement. Cela montre que le mécanisme de détection et de blocage du contrôleur POX est efficace, car il limite l'impact de l'attaque et permet au service de rester fonctionnel.
Le graphique suivant illustre le temps de détection avec l'augmentation du nombre d'attaquants. Le temps de détection semble évolué d'une manière linéaire

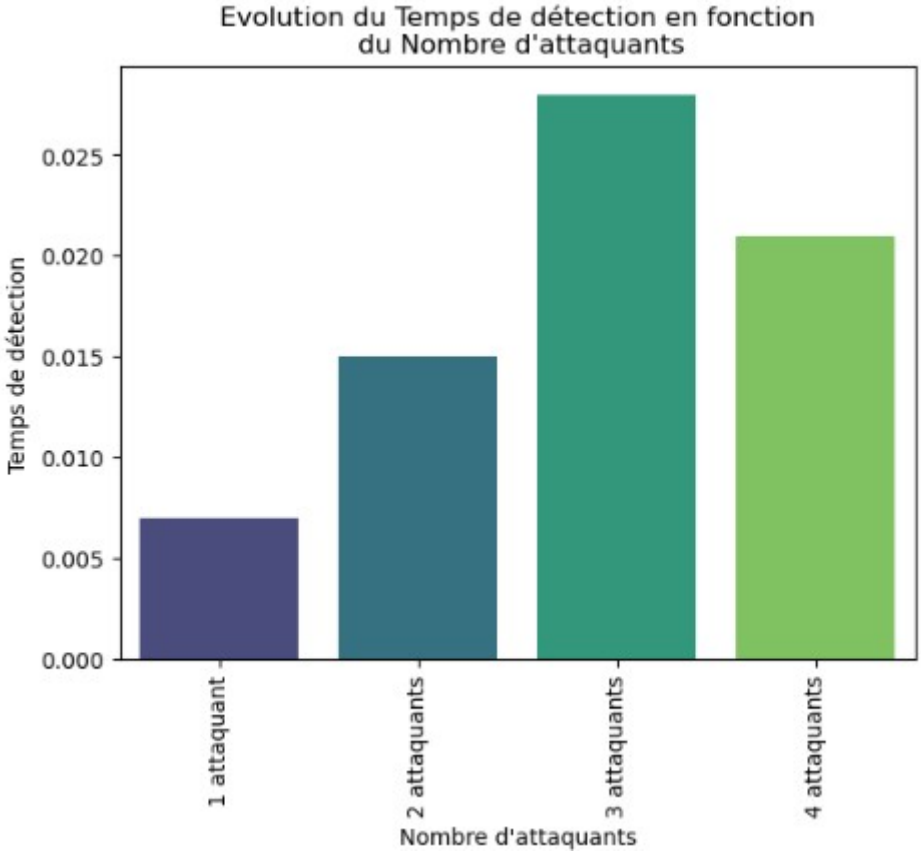


Figure 24: Evolution du temps de détection des attaques en fonction du nombre d'attaquants

La solution présentée dans ce travail représente un seul service de sécurité bâti avec la fonction de traitement des messages Packet-In. Cependant, d'autres services de sécurité peuvent être implémentés de la même façon. Particulièrement, le contrôle centralisé facilite la détection du protocole tunneling.

III.6. Conclusion

Dans ce chapitre, nous avons présenté la conception et l'implémentation d'une plateforme de sécurité programmable basée sur la paradigme SDN. Nous avons détaillé l'architecture proposée, ainsi que les différents mécanismes mis en place pour assurer la détection et la mitigation des attaques réseau, notamment les attaques de type DDoS. L'utilisation du contrôleur POX a permis de développer des fonctionnalités de supervision, d'analyse du trafic et de réaction dynamique face aux comportements anormaux. Les résultats expérimentaux obtenus, notamment à travers les tests de simulation sous Mininet, ont démontré l'efficacité du système, capable de détecter les attaques et de bloquer les sources malveillantes tout en maintenant la disponibilité du service.

Ainsi, cette approche basée sur SDN offre une solution flexible, évolutive et efficace pour renforcer la sécurité des réseaux modernes, tout en facilitant l'intégration de nouveaux mécanismes de protection.

Conclusion générale

Face à l'évolution rapide des réseaux informatiques et à l'augmentation des cyber-attaques, la sécurité des infrastructures réseau est devenue un enjeu majeur. Parmi ces menaces, les attaques DDoS peuvent fortement impacter la disponibilité des services en saturant les ressources réseau.

Dans ce contexte, le SDN propose une approche innovante qui améliore à la fois la gestion et la sécurité des réseaux grâce à la centralisation du contrôleur SDN permet d'avoir une vision globale du trafic et de mettre en place des mécanismes de détection et de réponse en temps réel.

Dans ce projet, nous avons travaillé sur un module de sécurité basé sur le SDN, en utilisant le contrôleur POX pour analyser le trafic réseau. Nous avons mis en place un système de détection et de blocage des attaques DDoS au niveau du contrôleur, permettant d'identifier les comportements anormaux et d'appliquer automatiquement des règles de filtrage afin de protéger le réseau.

Ce travail nous a permis de comprendre le rôle essentiel du contrôleur SDN dans la sécurité des réseaux, ainsi que l'importance de la programmabilité dans l'automatisation des mécanismes de défense.
Comme perspective future, il serait intéressant d'améliorer ce système en intégrant des techniques d'intelligence artificielle afin d'assurer une détection plus précise, proactive et capable de reconnaître plusieurs types d'attaques complexes.

Références

[1] Fortinet. (s.d.). *Definition TCP/IP (Transmission Control Protocol/Internet Protocol)*. Consulté sur : https://www.fortinet.com/fr/resources/cyberglossary/tcp-ip

[2] Gargan, R. (2024, août 12). *Network attacks*. Consulté sur : https://www.netmaker.io/resources/network-attacks

[3] BA Info. (2023). *Attaque DDoS* [Infographie]. Consulté sur : https://www.ba-info.fr/wp-content/uploads/2023/12/infographie-darticles-72-%C3%97-30-po7.png

[4] Sobrero, F., Clavarezza, B., Ucci, D., & Bisio, F. (2023). Towards a near-real-time protocol tunneling detector based on machine learning techniques. Journal of Cybersecurity and Privacy, 3(4), 794-807.

[6] Oliver Buxton, 05 mars 2025. [Internet]. Disponible sur : https://us.norton.com/blog/malware/what-is-ip-spoofing

[7] Figure de IP Spoofing. [Internet]. Disponible sur : https://www.zenarmor.com/docs/assets/images/ip-spoofing-181ecdcb3bd8364306cc5c2534764d10.png

[8] KristenS.(s.d.).Cross site Scripting (XSS). OWASP.https://owasp.org/www-community/attacks/xss/

[9] Alenzi, K. F., & Abbase, O. A. B. (2022). A defensive framework for reflected xss in client-side applications. *Journal of Web Engineering*, *21*(7), 2209-2229.

[10] Pramatarov, M. (2025, May 14). *DNS Spoofing (DNS poisoning)*. ClouDNS Blog.https://www.cloudns.net/blog/dns-spoofing-dns-poisoning/

[11] Proofpoint. (2025). Qu'est-ce que le session hijacking ou détournement de session. Proofpoint. Consulté le 12 février 2026, sur https://www.proofpoint.com/fr/threat-reference/session-hijacking

[12] Beschokov, M. (2025, 27 juin). What is Session Hijacking? Examples & Prevention. Consulté le 12 février 2026, sur https://www.wallarm.com/what/session-hijacking-attack

[13] Fortinet. (n.d.). Top 20 most common types of cyber attacks. Fortinet Cyber Glossary. Consulté February 12, 2026, sur https://www.fortinet.com/resources/cyberglossary/types-of-cyber-attacks

[14] Baitha, A. K., & Vinod, S. (2018). Session hijacking and prevention technique. International Journal of Engineering & Technology, 7(2.6), 193–198. https://www.sciencepubco.com/index.php/IJET

[15] Abbas, S., Naser, W., & Kadhim, A. (2023). Subject review: Intrusion detection system (IDS) and intrusion prevention system (IPS). *Global Journal of Engineering and Technology Advances*, *2*(14), 155-158.

[16] Dagorn, N. (2006). Détection et prévention d'intrusion: présentation et limites.

[17] Serbout, S., El Malki, A., Pautasso, C., & Zdun, U. (2023). *Uncovering rate limiting in the API wild: A pattern-based empirical study*. In Proceedings of the 2023 IEEE International Conference on Software Architecture Companion (ICSA-C). IEEE.

[18] Cloudflare. (s. d.). *What is rate limiting?* Cloudflare Learning Center. https://www.cloudflare.com/learning/bots/what-is-rate-limiting/

[19] Muzaki, R. A., Briliyant, O. C., Hasditama, M. A., & Ritchi, H. (2020, October). Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall. In *IWBIS* (pp. 85-90).
[20] Singh, K. K. V., & Gupta, H. (2016, March). A New Approach for the Security of VPN. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies* (pp. 1-5).
[21] Moussa, T. M., & O. L. S. (2018). *Implémentation du SDN dans une structure IP/MPLS* (Projet de fin d'étude, Université Mouloud Mammeri de Tizi-Ouzou). https://www.ummto.dz/dspace/handle/ummto/6287
[22] Chahed, B. (2015). *Mise en œuvre des aspects de gestion des réseaux définis par logiciels (réseaux SDN)* (Mémoire de maîtrise, Université de Montréal). https://publications.polymtl.ca/1924/
[23] Mir, S., & Chouteau, E. (2013). *Le SDN : ce qu'il faut savoir sur le « Cisco killer». Risk Insight* (Wavestone). https://www.riskinsight-wavestone.com/2013/10/le-sdn-ce-quil-faut-savoir-sur-le-cisco-killer/
[24] Kaljic, E., Maric, A., Njemcevic, P., & Hadzialic, M. (2019). *A Survey on Data Plane Flexibility and Programmability in Software-Defined Networking*. IEEE Access.
[25] Dawadi, B. R., Rawat, D. B., & Joshi, S. R. (2019). *Software Defined IPv6 Network: A New Paradigm for Future Networking*. **Journal of the Institute of Engineering**, 15(2), 1-13. https://doi.org/10.3126/jie.v15i2.27636
[26] Choukri, I., Ouzzif, M., & Bouragba, K. (2019, June). Software Defined Networking (SDN): Etat de L'art. In *Colloque sur les Objets et systèmes Connectés*.
[27] Torres, L. (2016, janvier 6). *Software Defined Network : la scalabilité des réseaux à travers l'automatisation.* Devoteam Rebirth. https://rebirth.devoteam.com/2016/01/06/software-defined-network/
[28] Liu, L., Tsuritani, T., Morita, I., Guo, H., & Wu, J. (2011, décembre). *Experimental validation and performance evaluation of OpenFlow-based wavelength path control in transparent optical networks.* https://www.researchgate.net/profile/Jian-Wu26/publication/221774501_Experimental_validation_and_performance_evaluation_of_OpenFlowbased_wavelength_path_control_in_transparent_optical_networks/links/54d3286d0cf28e069727a0ad/Experimental-validation-and-performance-evaluation-of-OpenFlow-based-wavelength-path-control-in-transparent-optical-networks.pdf
[29] EFORT. (2016). *Le protocole OpenFlow dans l'architecture SDN (Software Defined Network).* https://docplayer.fr/42864749-Le-protocole-openflow-dans-l-architecture-sdnsoftware-defined-network.html
[30] Aichaoui, A., & Aitbelkacem, Y. (2018). *Étude et implémentation d'une architecture SDN LAN* (Mémoire de master). Université Mouloud Mammeri de Tizi-Ouzou. https://www.ummto.dz/dspace/bitstream/handle/ummto/6707/AichaouiAnis_AitbelkacemYanis.pdf
[31] Open Networking Foundation. (2014). *OpenFlow switch specification (Version1.5.0).* https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.0.noipr.pdf

[32] Benamrane, F., Ben Mamoun, M., & Benaini, R. (2017). *Étude des performances des architectures du plan de contrôle des réseaux Software-Defined Networks.* https://doi.org/10.13140/RG.2.2.33883.57126
[33] Dargahi, T., Caponi, A., Ambrosin, M., Bianchi, G., & Conti, M. (2017). *A survey on the security of stateful SDN data planes.* IEEE Communications Surveys & Tutorials. https://doi.org/10.1109/COMST.2017.2689819
[34] Azodolmolky, S. (2013). *Software defined networking with OpenFlow* (Vol. 153). Birmingham: Packt Publishing.
[35] IPCisco. (2018, 7 juin). *Open Flow Messages*. Consulté le 26 mars 2026, sur https://ipcisco.com/lesson/open-flow-messages/
https://doi.org/10.1088/1757-899X/1022/1/012056
[37] Ayodele, B., & Buttigieg, V. (2024). *SDN as a defence mechanism: A comprehensive survey*. International Journal of Information Security, 23, 141–185. https://doi.org/10.1007/s10207-023-00764-1
[38] Virtualbox, O. V. (2011). Oracle vm virtualbox. *Change*, *107*, 1-287.
[39] Petersen, R. (2022). *Ubuntu 22.04 LTS Desktop: Applications and Administration*. surfing turtle press.
[40] Open vSwitch. (s.d.). *What is Open vSwitch?* Open vSwitch Documentation. https://docs.openvswitch.org/en/latest/intro/what-is-ovs/
[41] Kaur, K., Singh, J., & Ghumman, N. S. (2015). *Mininet as software defined networking testing platform*. Disponible à : https://www.researchgate.net/profile/Japinder-Singh/publication/287216738_Mininet_as_Software_Defined_Networking_Testing_Platform/links/5674461808aebcdda0de21cc/Mininet-as-Software-Defined-Networking-Testing-Platform.pdf
[42] Brian Linkletter. (2015). *How to use MiniEdit, Mininet's graphical user interface*. Disponible sur : https://brianlinkletter.com/2015/04/how-to-use-miniedit-mininets-graphical-user-interface/
[43] Journal du Net. (s.d.). *Python : définition et utilisation de ce langage informatique*. Consulté sur https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/
[44] Noman, H. M., & Jasim, M. N. (2020). *POX controller and OpenFlow performance evaluation in software defined networks (SDN) using Mininet emulator*. IOP Conference Series: Materials Science and Engineering, 881(1), 012102. https://iopscience.iop.org/article/10.1088/1757-899X/881/1/012102/pdf
[45] Deshmukh, S., Gawde, A., & Nagori, N. (2021). *Implementing software defined networking (SDN) based firewall using POX controller*. International Journal of Innovations in Engineering Research and Technology, 8(9), 168–174. https://repo.ijiert.org/index.php/ijiert/article/view/2899
[46] Rezaei, G., & Hashemi, M. R. (2021). *An SDN-based firewall for networks with varying security requirements*. In 2021 26th International Computer Conference (CSICC) (pp. 1–7). IEEE. https://doi.org/10.1109/CSICC52343.2021.9420571
[47] Li, J., Zhao, Z., & Li, R. (2018). Machine learning-based IDS for software-defined 5G network. *IET Networks, 7*(2), 53–60. https://doi.org/10.1049/iet-net.2017.0212

[48] Javeed, D., Gao, T., & Khan, M. T. (2021). *SDN-enabled hybrid DL-driven framework for the detection of emerging cyber threats in IoT*. **Electronics, 10**(8), 918. https://doi.org/10.3390/electronics10080918
[49] Ullah, I., Raza, B., Ali, S., Abbasi, I. A., Baseer, S., & Irshad, A. (2021). *Software defined network enabled fog-to-things hybrid deep learning driven cyber threat detection system*. **Security and Communication Networks, 2021**, 1–15. https://doi.org/10.1155/2021/6136670
[50] Chen, C. C., Chen, Y. R., Lu, W. C., Tsai, S. C., & Yang, M. C. (2017). *Detecting amplification attacks with software defined networking*. In **2017 IEEE Conference on Dependable and Secure Computing** (pp. 195–201). IEEE. https://doi.org/10.1109/DESEC.2017.8073807
[51] Xing, J., Yang, M., Zhou, H., Wu, C., & Ruan, W. (2019). *Hiding and trapping: A deceptive approach for defending against network reconnaissance with software-defined network*. In **2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)** (pp. 1–8). IEEE. https://doi.org/10.1109/IPCCC47392.2019.8958776
[52] Lin, H. (2019). *SDN-based in-network honeypot: Preemptively disrupt and mislead attacks in IoT networks*. arXiv preprint arXiv:1905.13254.
[53] Kumar, V., Chauhan, H., & Panwar, D. (2013). *K-means clustering approach to analyze NSL-KDD intrusion detection dataset*. International Journal of Soft Computing and Engineering, 3(4), 1–4.