RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ A. MIRA-BEJAIA

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

FACULTÉ DES SCIENCES EXACTES

DÉPARTEMENT D'INFORMATIQUE

MÈMOIRE

En vue de l'obtention du Diplôme de

Master en Informatique

Domaine : Mathématiques et Informatique **Filière :** Informatique

Spécialité : Administration et Sécurité des Réseaux

Présenté par

M. AHNIA Abderaouf

M. ALLOUTI Youcef

Thème

Réseaux définis par logiciel (SDN) : généralités, vulnérabilités et sécurisation intelligente

Soutenu le 29 Juin 2025. **Devant le jury composé de :**

Nom et Prénom	Grade	Université	Rôle
M. FARAH Zoubyr	MCB	Université de Béjaia	Président
M. MOKTEFI Mohand	MCB	Université de Béjaia	Rapporteur
M. YAZID Mohand	Professeur	Université de Béjaia	Examinateur
Mme. BACHIRI Lina	MCA	Université de Béjaia	Examinatrice
Mlle. SABRI Salima	MCA	Université de Béjaia	Examinatrice

Année Universitaire : 2024/2025

REMERCIMENT

A l'issue de notre cycle d'étude, nos vifs remerciements s'adressent :

Aux membres de jury, pour avoir accepté d'évaluer le présent mémoire.

A Dr. MOKTEFI Mohand, notre encadrant académique, pour son accompagnement constant. Sa rigueur scientifique et ses remarques pertinentes ont grandement contribué à réaliser ce mémoire.

Aux corps professionnel et administratif du Département Informatique de notre université pour la richesse des connaissances qu'ils nous ont transmis.

Aux personnels du service Informatique de l'entreprise Cevital qui nous ont bénéficié d'un stage de haut niveau et très adapté à la réalité de la vie professionnelle.

A nos familles, pour leur générosité dont ils ont su faire preuve.

DÉDICACE

A. Abderaouf

À mes parents, aucun hommage ne pourrait être à la hauteur de l'amour
et de la force dont ils ne cessent de me combler.

À tous ceux qui croient aux vertus de l'effort et du travail.

A. Abderaouf

A. Youcef

À Allah, le Très-Haut, pour sa grâce et son soutien infini.

À mes parents, sources de force et d'inspiration, pour leur amour inconditionnel et leur foi en moi.

À mon frère et ma sœur, pour leur affection et leur soutien.

À la mémoire de mes grands-parents, que Dieu les comble de Sa miséricorde.

À tous ceux qui ont semé en moi courage et persévérance,
je dédie ce travail avec gratitude et amour.

A. Youcef

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS ET ACRONYMES

Abréviation	**Définition**
ACL	Access Control List
AN	Active Network
API	Application Programming Interface
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
CSDN	Cellular Software Defined Network
CSV	Comma-Separated Values
DCAN	Devolved Control of ATM Networks
DdoS	Distributed Denial of Service
DoS	Denial of Service
FP	False Positive
FN	False Negative
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPsec	Internet Protocol Security

Abréviation	**Définition**
IPv4	Internet Protocol version 4
KNN	K-Nearest Neighbors
MAC	Media Access Control
MITM	Man-In-The-Middle
ML	Machine Learning
MPLS	Multiprotocol Label Switching
NVGRE	Network Virtualization using GRE
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OVS	Open vSwitch
PKL	Pickle (modèle ML sauvegardé)
PN	Programming Network
QoS	Quality of Service
RIP	Routing Information Protocol
RTT	Round Trip Time
SDN	Software Defined Networking
STP	Spanning Tree Protocol
SVM	Support Vector Machine
SYN-ACK	Synchronize-Acknowledge
TCP	Transmission Control Protocol
TCAM	Ternary Content Addressable Memory
TLS	Transport Layer Security
TP	True Positive
TN	True Negative
UI	User Interface
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VoIP	Voice over IP
VPN	Virtual Private Network
VRRP	Virtual Router Redundancy Protocol
VSDN	Video over Software Defined Network
VTP	VLAN Trunking Protocol
WSDN	Wireless Software Defined Network

INTRODUCTION GÉNÉRALE

Les réseaux informatiques jouent aujourd'hui un rôle central dans le fonctionnement des systèmes d'information modernes. Ils permettent d'assurer la transmission des données, la connectivité entre les équipements et le maintien des services en ligne. À ce titre, ils constituent la colonne vertébrale de toute infrastructure numérique.

Avec l'évolution rapide des technologies, l'augmentation massive du volume de données échangées et la diversification des usages, les exigences en matière de gestion, de performance et de sécurité des réseaux sont devenues de plus en plus complexes. Dans les architectures traditionnelles, chaque équipement prend ses décisions de manière locale, ce qui rend le contrôle du réseau globalement rigide et difficile à adapter, notamment dans les environnements étendus ou distribués. L'introduction de nouvelles politiques ou la modification de la topologie réseau requiert généralement des interventions manuelles sur plusieurs équipements, entraînant un risque accru d'erreurs de configuration et un ralentissement des processus de gestion.

Pour répondre à ces limites, une nouvelle approche a vu le jour : celle des Réseaux Définis par Logiciel (Software Defined Networking – SDN). Le SDN repose sur la séparation du plan de contrôle et du plan de données, permettant un pilotage centralisé du réseau par l'intermédiaire d'un contrôleur logiciel. Cette architecture apporte de nombreux avantages, notamment en matière de programmabilité, d'automatisation et de flexibilité, en rendant le réseau plus facile à gérer et à adapter aux besoins spécifiques des applications ou des politiques de sécurité.

Cependant, cette centralisation, tout en apportant plus d'intelligence et de contrôle, introduit de nouveaux risques. Le contrôleur SDN, véritable cerveau du réseau, devient une cible critique, notamment face aux attaques par déni de service distribué (DDoS), qui cherchent à submerger le réseau par un flot massif de requêtes malveillantes. Ces attaques peuvent ralentir, voire paralyser, le

réseau et compromettre la disponibilité des services.

Dans ce contexte, une question essentielle se pose : comment protéger efficacement une architecture SDN contre les attaques DDoS tout en maintenant ses performances et sa flexibilité opérationnelle ? C'est dans cette perspective que s'inscrit le présent mémoire. Notre travail vise à contribuer à la sécurisation des réseaux SDN à travers l'intégration de techniques d'apprentissage automatique (Machine Learning) pour la détection intelligente des attaques DDoS. Concrètement, nous avons simulé un environnement SDN sous Mininet et mis en œuvre plusieurs algorithmes d'apprentissage supervisé – notamment le modèle Random Forest – afin d'analyser, classer et distinguer le trafic réseau légitime du trafic malveillant. Cette approche permet d'envisager une détection dynamique et automatiser des comportements anormaux dans un réseau SDN.

Ce travail a été réalisé dans le cadre d'un stage d'observation effectué en 2025 au sein de l'entreprise Cevital, un acteur majeur du secteur agroalimentaire en Algérie. Ce stage nous a offert l'opportunité d'observer une infrastructure réseau traditionnelle en environnement industriel. Bien que cette topologie réelle n'ait pas pu être reproduite intégralement en raison de sa complexité, nous nous en sommes inspirés pour concevoir une topologie simplifiée, afin de la comparer à une architecture SDN simulée.

Le mémoire est structuré en quatre chapitres :

1. Le premier chapitre présente les fondements théoriques du **SDN**, ses composants, son mode de fonctionnement ainsi que ses avantages par rapport aux architectures traditionnelles.
2. Le deuxième chapitre est consacré à la **sécurité dans les réseaux SDN**, en identifiant les principales vulnérabilités et en présentant les mécanismes de défense existants.
3. Le troisième chapitre propose une **comparaison fonctionnelle et expérimentale** entre une topologie réseau classique et une topologie SDN simulée, à travers l'analyse de leurs performances.
4. Le quatrième chapitre détaille l'**implémentation de notre modèle de détection d'attaques DDoS par Machine Learning**, avec une présentation des outils utilisés (Python, Random Forest), des étapes de traitement des données et des résultats obtenus.

À travers cette étude, nous avons cherché à mieux comprendre les atouts et les limites des réseaux SDN, à analyser leurs vulnérabilités spécifiques face aux attaques DDoS, et à explorer des solutions intelligentes de détection basées sur le Machine Learning, afin de contribuer à la conception de réseaux plus sûrs, plus flexibles et plus adaptés aux enjeux numériques actuels.

CHAPITRE

I

GÉNÉRALITÉS SUR LES RÉSEAUX DÉFINIS PAR LOGICIEL (SDN)

I.1 Introduction

Avec l'évolution spectaculaire des technologies, les réseaux informatiques ont connu une transformation considérable afin de s'adapter avec les exigences croissantes notamment en termes de connectivité, de gestion et de performance.
Face à ces défis, une nouvelle approche a vu le jour : les Réseaux Définis par Logiciel. Cette nouveauté implique une séparation évidente entre le plan de contrôle et le plan de données, garantissant ainsi une gestion assez centralisée, dynamique et programmable des infrastructures réseau.

Ce chapitre a pour objectif de fournir une vue d'ensemble des réseaux SDN en commençant par leur évolution, les limites des architectures traditionnelles, puis en mettant en évidence leur fonctionnement, leur architecture, et les avantages qu'ils génèrent par rapport aux modèles classiques.

I.2 Évolution et Concepts Fondamentaux du SDN

Cette section présente les bases nécessaires pour comprendre le fonctionnement des réseaux définis par logiciel (SDN).

I.2.1 Aperçu historique

Avant que les réseaux définis par logiciel (SDN) ne deviennent populaires, plusieurs projets et idées avaient déjà conçu les bases de cette nouvelle approche. Cependant, des initiatives comme Active Network (AN), Programming Network (PN) et le projet DCAN (Devolved Control of ATM Networks) ont donné une distinction entre la gestion et le contrôle des données, ouvrant la voie à des concepts innovants en matière de réseau.

Le véritable déclic est venu avec le projet Ethane [6], lancé en 2006 à l'université de Stanford, ce projet a proposé une nouvelle façon de construire des réseaux d'entreprise en utilisant un contrôleur central pour gérer à la fois les règles de sécurité et la circulation des données.

Par conséquent, le projet **Ethane** se reposait sur deux éléments fondamentaux, un contrôleur qui décidait si un paquet devait passer ou non, et un switch spécialement conçu pour appliquer ces décisions.

Depuis cette étape charnière, l'évolution des SDN s'est déroulée en plusieurs phases :

— 2007-2011 : Les premières expérimentations du SDN

— 2012-2013 : Le concept se précise et se stabilise

— 2014-2015 : Les premières automatisations apparaissent dans les réseaux SDN

— 2015-2016 : l'apparition des déploiements dans divers environnements

— 2016-2019 : Le SDN se généralise dans le monde des réseaux

— 2020 : Le SDN s'impose comme une technologie majeure

Ces différentes étapes révèlent le développement du SDN d'une idée innovante à une technologie incontournable, offrant plus de flexibilité, une meilleure sécurité et une gestion simplifiée par rapport aux réseaux traditionnels.

I.2.2 Définition du SDN

Le Software-Defined Networking (SDN) est une approche innovante qui permet de gérer et de programmer un réseau de manière centralisée, en s'appuyant sur un contrôleur unique plutôt que sur la configuration manuelle de chaque équipement (voir la figure 1).

Contrairement aux architectures traditionnelles où le contrôle du trafic repose sur des routeurs et commutateurs dédiés, le SDN sépare le plan de contrôle (qui prend les décisions) du plan de données (qui exécute ces décisions), offrant ainsi une plus grande souplesse dans la gestion du réseau [30].

Grâce à cette séparation, le SDN peut aussi bien gérer des réseaux physiques que des réseaux virtuels [13], en simplifiant l'administration et l'automatisation des opérations. Il permet notamment de segmenter plusieurs réseaux virtuels au sein d'une même infrastructure physique ou encore de regrouper des équipements situés sur des réseaux physiques différents dans un même réseau virtuel.

Le contrôleur SDN joue un rôle clé en fournissant une vue d'ensemble du réseau et en facilitant l'application de politiques de sécurité, de qualité de service (QoS), de routage et de gestion du trafic. Cette approche centralisée améliore considérablement la flexibilité, la réactivité et l'efficacité des réseaux modernes, tout en réduisant les coûts et la complexité de leur administration. La figure 1 donne un aperçu sur un réseau SDN simple.

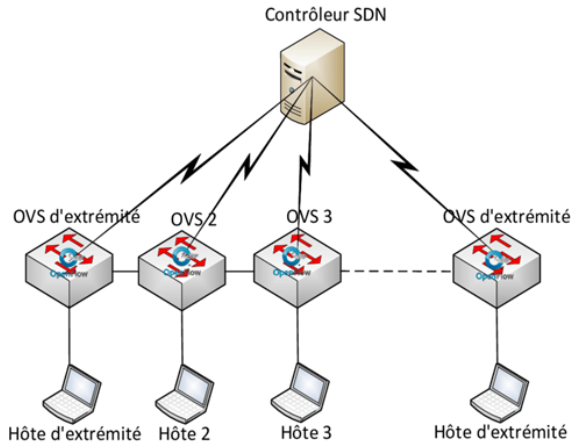

FIGURE I.1 – Topologie d'un réseau SDN simple

I.2.3 Différents Modèles d'Architecture SDN

Le SDN repose sur plusieurs architectures [1] adaptées à des besoins variés en matière de gestion, de flexibilité et de scalabilité des réseaux. Voici les principales approches utilisées :

SDN Superposé

Ce modèle crée des réseaux virtuels indépendants de l'infrastructure physique sous-jacente. Il repose sur des technologies comme **VXLAN**, **NVGRE** ou **GRE** pour encapsuler les paquets et les transporter à travers le réseau physique. Cette approche permet aux administrateurs de gérer les réseaux virtuels de manière logicielle sans modifier l'architecture physique simplifiant ainsi le déploiement des services et l'isolation des réseaux.

SDN Hybride

Le SDN hybride combine les avantages du SDN superposé et du SDN à contrôleur. Il permet d'introduire progressivement des fonctionnalités SDN sans perturber les réseaux traditionnels. Cette architecture est particulièrement utile dans les environnements où certains équipements ne prennent pas encore en charge **OpenFlow** ou d'autres protocoles SDN. Elle offre une transition en douceur vers une infrastructure SDN complète.

SDN à Contrôleur

Ce modèle repose sur un contrôleur centralisé qui supervise et orchestre l'ensemble du réseau. Le contrôleur possède une vision globale et en temps réel de l'état du réseau, ce qui lui permet de prendre des décisions intelligentes pour la gestion du trafic. Cette centralisation simplifie la configuration et l'optimisation du réseau, mais peut poser un problème de résilience en cas de défaillance du contrôleur.

SDN de Structure

Contrairement au modèle à contrôleur unique, le SDN de structure repose sur un plan de contrôle distribué entre plusieurs périphériques réseau. Cette approche améliore l'évolutivité et la tolérance aux pannes, car elle évite les goulots d'étranglement causés par un contrôleur central unique. Elle est souvent utilisée dans les grandes infrastructures, comme les datas center, pour garantir un trafic fluide et dynamique.

SDN Applicatif

Ce modèle est conçu pour répondre aux besoins spécifiques de certaines applications en intégrant des services avancés comme l'équilibrage de charge, la gestion de la qualité de service (QoS) et la sécurité des applications. Le contrôleur SDN applicatif prend en compte les besoins des applications en temps réel et adapte la configuration du réseau en conséquence garantissant une meilleure performance et une gestion optimisée des ressources.

SDN Cloud

Dans cette approche, le contrôleur SDN est hébergé dans le cloud, ce qui offre une gestion flexible et évolutive du réseau. Ce modèle est particulièrement utile pour les entreprises qui exploitent des infrastructures multi-clouds ou hybrides, car il permet d'orchestrer dynamiquement les ressources réseau selon la demande. Il favorise également l'automatisation et l'optimisation des performances dans les environnements cloud natifs.

SDN Distribué

Le SDN distribué décentralise le processus de prise de décision en répartissant le plan de contrôle entre plusieurs équipements réseau. Cette approche améliore la résilience du système, réduit la latence et assure une continuité de service en cas de panne d'un contrôleur. Elle est souvent adoptée dans les infrastructures critiques nécessitant une haute disponibilité et une gestion dynamique des flux réseau.

Architecture et Composants des Réseaux SDN

Le réseau SDN se distingue par sa structure modulaire, qui sépare clairement la logique de contrôle de la transmission des données. Cette séparation facilite une gestion centralisée une grande flexibilité et l'intégration rapide de nouvelles fonctionnalités. La présente section détaille d'abord l'architecture globale et les interfaces de communication du SDN, puis présente ses composants essentiels.

Architecture Fonctionnelle des Réseaux SDN

L'architecture des réseaux SDN [5] est basée sur une organisation simple, mais très efficace en trois couches. Chacune joue un rôle bien défini, et elles communiquent entre elles à l'aide d'interfaces de programmation (API). Cette séparation permet d'avoir un réseau plus facile à gérer, plus flexible et surtout plus intelligent.

La première couche, appelée couche infrastructure (data plane), regroupe les équipements comme les commutateurs (switches) et les routeurs. Ce sont eux qui s'occupent du transport réel des paquets à travers le réseau. Ils se contentent d'appliquer les règles qu'on leur envoie sans rien décider par eux-mêmes. On peut dire que c'est la partie physique ou matérielle du réseau SDN. Ces équipements peuvent aussi fournir des statistiques sur le trafic, ce qui sera utile aux couches supérieures.

Juste au-dessus, on trouve la couche de contrôle (control plane). C'est ici que se situe le ou les contrôleurs SDN. Le contrôleur agit comme le cerveau du réseau : il décide comment les paquets doivent circuler, il applique des règles de sécurité, de qualité de service, de routage etc. Il gère tout cela à distance, en envoyant ses instructions aux équipements de la couche en dessous via l'interfaces Southbound. Le protocole le plus utilisé pour cette interface est OpenFlow, mais on peut aussi trouver d'autres solutions comme ForCes ou OVSDB, selon les besoins du réseau [16].

Enfin, tout en haut se trouve la couche application (application plane). C'est dans cette couche que fonctionnent les applications réseau, comme celles qui permettent de surveiller le trafic de détecter des anomalies, d'optimiser les performances ou encore d'améliorer la sécurité. Ces applications ne communiquent pas directement avec les équipements, mais passent par le contrôleur, en utilisant l'interfaces Northbound. Cette interface est souvent basée sur des API RESTful, qui sont simples à utiliser, notamment pour les développeurs qui souhaitent créer leurs propres services réseau.

Il existe aussi une autre interface, un peu moins connue mais très importante lorsqu'on travaille avec plusieurs contrôleurs SDN répartis dans un réseau : **l'interfaces East/West.** Elle permet aux différents contrôleurs de communiquer entre eux, d'échanger des informations et de se synchroniser. Même si cette interface n'est pas encore vraiment standardisée, elle devient essentielle dans les grandes architectures distribuées où une vision globale du réseau est nécessaire.

La figure ci-dessous illustre cette architecture SDN, avec ses trois couches et les différentes interfaces

qui assurent la communication entre elles.

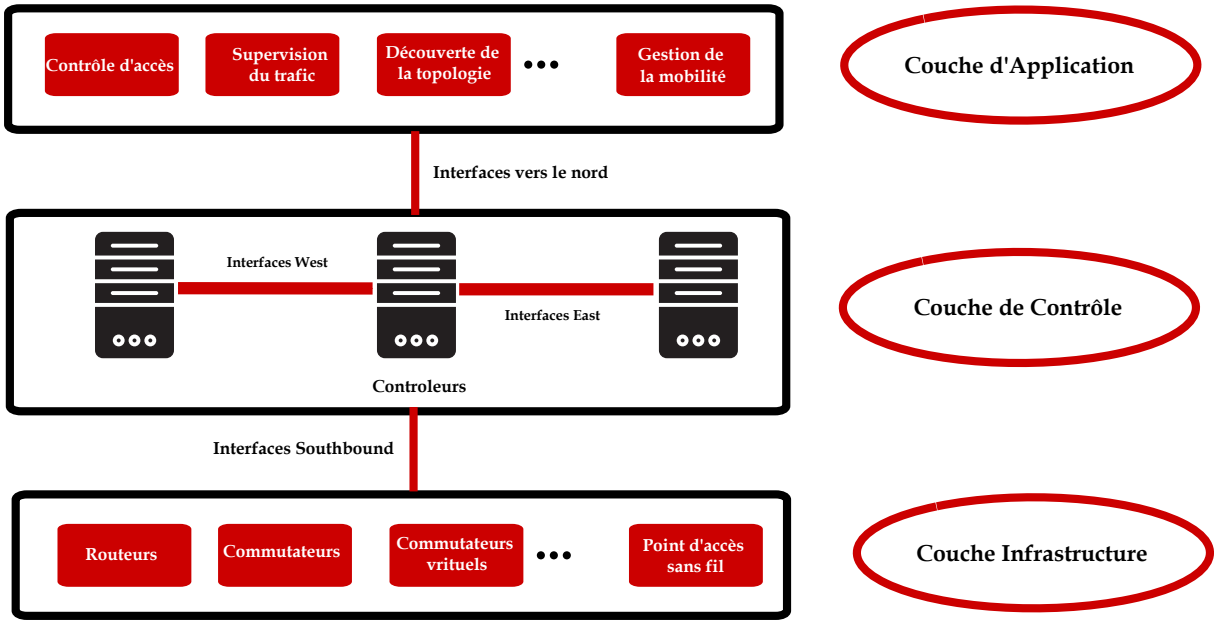

FIGURE I.2 – Architecture détaillée d'un réseau SDN

I.2.4 Composants des réseaux SDN

Les principaux composants du SDN, qui mettent en œuvre l'architecture décrite ci-dessus comprennent le contrôleur SDN, les commutateurs SDN???

I.2.5 Contrôleurs SDN

Le contrôleur SDN joue un rôle central dans l'architecture des réseaux SDN. Il agit comme le cerveau du réseau : il supervise les équipements, prend les décisions de routage et envoie les instructions aux commutateurs pour diriger les flux de données. Contrairement aux réseaux classiques où chaque équipement décide localement, ici c'est le contrôleur qui gère tout de manière centralisée, ce qui facilite l'administration et l'adaptation du réseau.

Il communique avec les équipements réseau (comme les switches OpenFlow) via des protocoles appelés **Southbound APIs**, le plus connu étant **OpenFlow**.
Pour interagir avec les applications réseau (comme la gestion du trafic ou la détection d'attaques), il utilise des Northbound APIs qui permettent aux développeurs d'automatiser ou de personnaliser le comportement du réseau.

Il existe plusieurs contrôleurs SDN [16], chacun avec ses particularités :

1. **NOX :** l'un des premiers contrôleurs, écrit en C++. Il est performant, mais son développement est plus complexe. Il est surtout utilisé dans le cadre de recherches.

2. **POX :** dérivé de NOX, mais écrit en Python. Plus simple à manipuler, il est idéal pour les projets académiques ou les tests.
3. **Beacon :** contrôleur en Java, conçu pour être modulaire et efficace. Il a servi de base à plusieurs autres contrôleurs.
4. **Floodlight :** basé sur Beacon, open source et bien documenté. Il convient pour des déploiements en production.
5. **OpenDaylight (ODL) :** un projet soutenu par la Linux Foundation, très complet, modulaire et compatible avec plusieurs protocoles (OpenFlow, NETCONF, BGP, etc.). Il est souvent utilisé dans des environnements professionnels.

En ce qui concerne le fonctionnement du contrôleur, on distingue deux modes :

1. **Mode réactif :** le switch ne sait pas comment traiter un paquet, donc il l'envoie au contrôleur. Ce dernier analyse le paquet, choisit le chemin, puis installe une règle dans le switch. Ce mode est flexible, mais un peu plus lent au premier envoi du paquet.
2. **Mode proactif :** ici, le contrôleur configure les règles de routage à l'avance dans les switches. Du coup, les paquets sont directement traités sans passer par le contrôleur ce qui réduit la latence et améliore les performances.

I.2.6 Commutateurs SDN

Dans un réseau SDN, les commutateurs jouent un rôle fondamental. Ils assurent l'acheminement des paquets au sein de l'infrastructure réseau, en appliquant rigoureusement les règles définies par le contrôleur. Contrairement aux commutateurs traditionnels qui prennent localement des décisions de commutation, ceux utilisés dans une architecture SDN se contentent d'exécuter les instructions reçues, garantissant ainsi une gestion centralisée cohérente et flexible du réseau. On distingue principalement deux types de commutateurs dans les environnements SDN **??** :

1. **Les commutateurs logiciels :** Il s'agit de programmes capables de simuler le comportement d'un commutateur physique, ils sont largement utilisés dans les environnements virtualisés ou pour des tests expérimentaux, grâce à leur facilité de déploiement et de configuration. Le plus connu est Open vSwitch (OVS), très répandu pour la gestion des connexions entre machines virtuelles. On peut également citer Indigo, proposé par Big Switch Networks.
2. **Les commutateurs matériels :** Ce sont des équipements physiques dédiés, optimisés pour les environnements exigeants comme les datacenters ou les réseaux backbone. Grâce à des composants spécialisés (comme la mémoire TCAM), ils permettent un traitement à très haut débit des paquets, tout en maintenant une faible latence.

Aujourd'hui, les commutateurs compatibles avec le protocole **OpenFlow** sont les plus utilisés dans les réseaux SDN. **OpenFlow**, en tant que norme largement adoptée, facilite l'interopérabilité

entre les équipements et le contrôleur, tout en assurant une communication standardisée. Ces commutateurs OpenFlow représentent ainsi l'élément de base dans la plupart des déploiements SDN actuels.

Le Commutateur OpenFlow

Le commutateur OpenFlow est un type spécifique de commutateur SDN qui fonctionne avec le protocole OpenFlow. Il possède une ou plusieurs tables de flux, chacune contenant des entrées de flux utilisées pour faire correspondre et traiter les paquets. Chaque entrée définit des critères de correspondance à partir de plusieurs champs réseau et inclut des compteurs pour suivre le traitement des paquets ainsi que des instructions à appliquer. Le commutateur OpenFlow communique avec le contrôleur via un canal dédié [7], assurant ainsi une gestion centralisée du réseau. Il est principalement composé de tables de flux et d'un canal sécurisé facilitant les échanges avec le contrôleur. La figure ci-dessous illustre les composants du commutateur OpenFlow

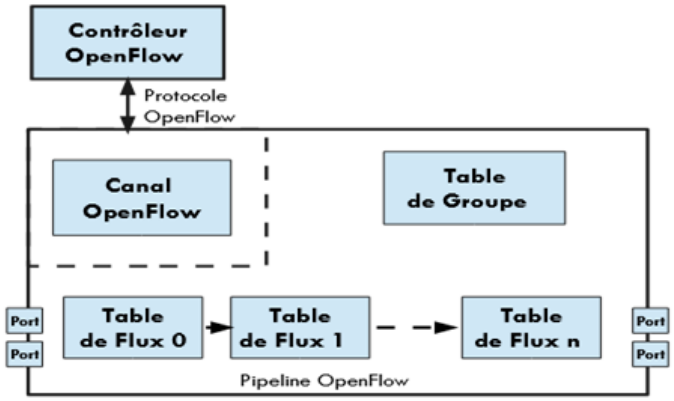

FIGURE I.3 – Commutateur OpenFlow

I.2.7 Protocole OpenFlow

OpenFlow est un protocole de communication ouvert qui opère principalement sur **la couche 2 du modèle OSI**. Il permet un accès direct au plan de transfert des commutateurs et routeurs ce qui facilite la gestion du trafic au sein d'un réseau SDN.

Grâce à ce protocole, un logiciel exécuté sur un ou plusieurs contrôleurs SDN peut définir de manière centralisée et dynamique, le chemin que doivent suivre les paquets dans un réseau composé de commutateurs compatibles OpenFlow. Le contrôleur envoie aux commutateurs des règles de traitement qui précisent comment les paquets doivent être traités.

L'un des grands avantages d'OpenFlow est sa capacité à dissocier la logique de contrôle du matériel réseau. Ainsi, il n'est plus nécessaire de configurer individuellement chaque commutateur : tout est piloté depuis un point central, le contrôleur. Cette flexibilité rend le réseau plus simple à gérer, plus réactif et facilement programmable [20].

Conçu pour garantir l'interopérabilité entre équipements de différents fabricants, OpenFlow est aujourd'hui l'un des protocoles les plus utilisés dans le domaine du SDN. Il a été développé après plusieurs années de recherche par l'Université de **Stanford et l'Université de Californie à Berkele**y, puis rendu public en 2011.

I.2.8 Canal OpenFlow

Le canal OpenFlow est une interface sécurisée assurant la communication entre un commutateur OpenFlow et son contrôleur. Il permet au plan de contrôle d'envoyer des instructions, de recevoir des requêtes et d'échanger des informations. L'ensemble des messages transmis via ce canal est crypté grâce à la sécurité de la couche de transport **(TLS)**. Chaque message inclut un en-tête OpenFlow, précisant la version du protocole, le type de message la longueur du message et l'ID de transaction. Ce canal prend en charge trois catégories de messages [20] :

- Messages Contrôleur-vers-Commutateur.
  - **Features :** Le contrôleur interroge le commutateur pour identifier ses capacités.
  - **Modify-state :** Permet de modifier, ajouter ou supprimer des entrées dans les tables de flux. Ce type inclut plusieurs messages comme Flow-Mod (règles de flux), Group-Mod (groupes d'actions), et Meter-Mod (contrôle du débit).
  - Configuration : Gère les paramètres du commutateur et répond aux requêtes du contrôleur.
  - **Read-state :** Collecte des statistiques, capacités et configurations du commutateur.
  - **Packet-out :** Transfère un paquet vers un port spécifique ou le détruit s'il n'a pas d'action définie.
  - **Barrier** : Confirme la fin d'une opération demandée par le contrôleur.
- Messages Symétriques.
  - **Hello :** Vérifie l'établissement de la connexion entre le contrôleur et le commutateur.
  - **Echo :** Mesure la latence et la bande passante de la connexion (chaque requête nécessite une réponse).
  - **Error :** Informe des problèmes de connexion.
- Messages Asynchrones
  - **Packet-in :** Le commutateur envoie un paquet au contrôleur lorsqu'aucune règle de flux ne lui correspond.
  - **Flow-removed :** Informe le contrôleur qu'une entrée de la table de flux a été supprimée.
  - **Port-status :** Signale un changement d'état sur un port.
  - **Role-status :** Informe le contrôleur lorsqu'un nouveau contrôleur prend le relais.

Ce canal est essentiel pour la gestion dynamique du trafic réseau et l'adaptabilité des infrastructures SDN. La figure ci-dessous illustre l'échange de communication entre un commutateur et un contrôleur.

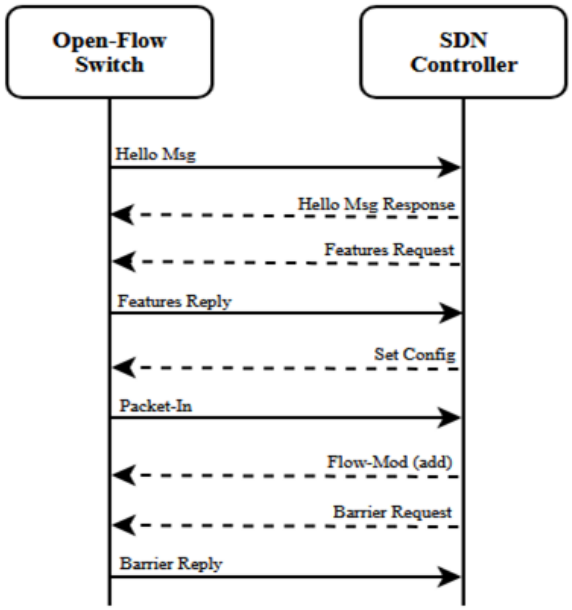

FIGURE I.4 – Schéma de communication entre un commutateur SDN et un contrôleur

NB : Les versions doivent être compatibles, Un commutateur OpenFlow et un contrôleur SDN doivent utiliser des versions compatibles du protocole OpenFlow pour pouvoir échanger des messages correctement.

I.2.9 Tables de flux

Les tables de flux (flow tables) d'un commutateur OpenFlow contiennent des règles permettant de traiter les paquets en fonction de critères spécifiques tels que les adresses MAC, les adresses IP, les ports et l'ID VLAN [20].
Chaque entrée de table de flux est composée de trois éléments :

1. un en-tête de correspondance définissant les caractéristiques du flux.
2. un ensemble d'actions précisant le traitement des paquets (transfert vers un port, envoi au contrôleur ou suppression).
3. des compteurs permettant de suivre le nombre de paquets, d'octets échangés et la durée de vie du flux.

Ces mécanismes assurent une gestion optimisée et dynamique du trafic réseau. La figure I.4 ci-dessous illustre la table de flux utilisée dans les réseaux SDN.

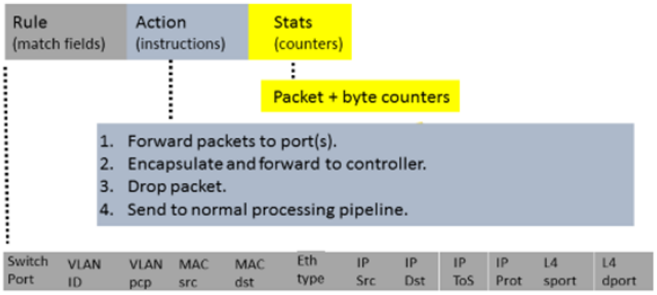

FIGURE I.5 – Table De Flux

I.3 Applications du SDN

Depuis son émergence en 2011, le SDN connaît une évolution rapide, portée par ses nombreux avantages dans divers secteurs. Grâce à sa flexibilité et à son approche centralisée, il s'intègre à différents domaines du réseau, offrant ainsi des solutions adaptées à divers besoins.

I.3.1 Amélioration de la qualité de service (QoS) sur Internet

L'architecture d'Internet n'a jamais garanti une stabilité absolue, ce qui peut poser des défis majeurs pour des services exigeants tels que le streaming vidéo, la VoD ou la visioconférence. Ces applications sont particulièrement sensibles aux délais et aux pertes de paquets, nécessitant ainsi une gestion optimisée du trafic. Grâce à la visibilité centralisée qu'offre le SDN, il devient possible d'optimiser le routage en fonction du débit et des conditions du réseau. C'est dans ce cadre qu'a émergé le concept de VSDN (Video over SDN), une approche qui tire parti de la centralisation du contrôle pour choisir dynamiquement le chemin optimal des flux multimédias [22].

Optimisation des centres de données

Dans les environnements de data centers, le SDN facilite la virtualisation et l'automatisation des infrastructures réseau. Un exemple notable est Oracle SDN, qui permet de relier dynamiquement les machines virtuelles aux serveurs réseau. Grâce à l'interface Oracle Fabric Manager, il devient possible de configurer et de surveiller le réseau à distance, tout en déployant à la demande des services tels que les pare-feu, l'équilibrage de charge ou encore le routage. Selon Oracle, cette technologie améliore considérablement les performances applicatives offrant un débit inter-serveur pouvant atteindre 80 Gb/s et multipliant par 30 l'efficacité des applications [21].

I.3.2 SDN dans les réseaux mobiles

Avec l'arrivée imminente de la 5G, les opérateurs télécoms doivent adapter leur infrastructure pour gérer l'explosion du volume de données tout en garantissant une qualité de service optimale.

Le SDN se révèle être un élément clé pour maximiser les performances et assurer une gestion dynamique des ressources réseau. Le concept de CSDN (Cellular SDN) [4] a ainsi été introduit pour exploiter pleinement les principes du SDN dans les réseaux mobiles.
Cette architecture collecte les données des utilisateurs et l'état du réseau afin d'optimiser la consommation énergétique, l'allocation des ressources et l'adaptation des services en fonction des besoins des abonnés.

I.3.3 Sécurité et protection des réseaux

L'architecture programmable du SDN a également été mise à profit pour renforcer la sécurité réseau. Une étude menée par Jafarian, Al-Shaer et Qi Duan a démontré qu'une approche basée sur des mutations fréquentes et aléatoires des adresses IP permettait de réduire de 99 % le risque d'interception des données par des attaquants utilisant l'analyse de paquets [12].
En outre, des solutions ont été proposées pour contrer les attaques par déni de service (DdoS). Une approche consiste à combiner des systèmes de détection d'intrusion (IDS) au sein des réseaux locaux avec les switches OpenFlow, afin d'isoler automatiquement les équipements compromis et ainsi limiter la propagation des attaques

I.4 Comparaison entre les Réseaux Traditionnels et les Réseaux SDN

Dans cette section, nous allons comparer les réseaux traditionnels et les réseaux SDN, puis présenter les principaux avantages et inconvénients de l'architecture SDN.

I.4.1 Différenciation entre les Réseaux SDN et les Réseaux Traditionnels

Les réseaux définis par logiciel (SDN) révolutionnent l'architecture réseau en dissociant le contrôle du trafic de son acheminement. Contrairement aux réseaux traditionnels, où chaque équipement fonctionne de manière autonome, le SDN repose sur un contrôleur centralisé qui orchestre l'ensemble du réseau. Cette approche facilite la gestion, améliore la flexibilité et optimise les performances en permettant une programmation dynamique des flux de données. De plus, elle simplifie l'intégration de nouvelles technologies comme la virtualisation et l'intelligence artificielle [16, 15]. La figure suivante compare l'architecture d'un réseau traditionnel à celle d'un réseau SDN.

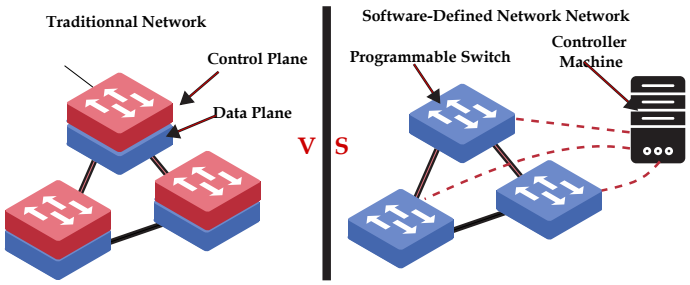

FIGURE I.6 – Comparaison entre architecture SDN et réseau traditionnel

Le tableau suivant résume les principales distinctions entre ces deux types d'architectures :

Critères	**Réseaux Traditionnels**	**Réseaux SDN**
Architecture	Intégration du contrôle et du transfert des données dans chaque équipement	Séparation entre le plan de contrôle et le plan de données
Gestion du réseau	Configuration manuelle et locale	Gestion centralisée via un contrôleur SDN
Programmabilité	Peu ou pas de possibilité de programmation	Programmable via des API et des protocoles comme OpenFlow
Flexibilité	Rigide, difficile à modifier	Dynamique, adaptable en temps réel
Évolutivité	Expansion complexe nécessitant du matériel supplémentaire	Facilement évolutif grâce à la virtualisation et l'automatisation
Sécurité	Sécurité répartie entre plusieurs équipements, difficile à unifier	Sécurité centralisée avec une meilleure visibilité des menaces
Coût	Coût élevé dû aux équipements propriétaires et à la maintenance	Réduction des coûts grâce à l'utilisation de solutions logicielles et de matériels ouverts

TABLE I.1 – Comparaison entre les réseaux traditionnels et les réseaux SDN

I.4.2 Avantages des réseaux SDN

Les réseaux SDN (Software Defined Networking) présentent plusieurs avantages qui les distinguent des architectures traditionnelles :

— **Gestion centralisée simplifiée :** En séparant le plan de contrôle du plan de données, le SDN permet une administration réseau plus simple. Le contrôleur central offre une vue d'ensemble du réseau, facilitant ainsi la configuration et la surveillance des équipements.

— Flexibilité et programmabilité accrues : Les équipements standard peuvent exécuter diverses règles réseau, offrant une plus grande souplesse dans la gestion du trafic. Cela permet d'adapter rapidement le réseau aux besoins changeants.

— **Automatisation et orchestration multi-fournisseurs :** Le SDN facilite l'autom]atisation de la gestion des réseaux issus de différents fournisseurs, permettant un déploiement rapide et à grande échelle des services tout en réduisant les erreurs humaines.

— **Optimisation du routage :** Grâce à une vision globale du réseau, le contrôleur SDN peut choisir les meilleurs chemins pour acheminer les données, remplaçant ainsi les protocoles de routage classiques comme OSPF ou RIP.

— **Réduction des coûts :** Contrairement aux équipements propriétaires coûteux, le SDN permet d'intégrer des solutions ouvertes et évolutives, réduisant ainsi les dépenses en matériel et en ressources humaines.

— **Services dynamiques et personnalisables :** Le SDN facilite la mise en place de politiques de routage adaptées aux besoins des entreprises, améliorant la gestion du trafic et des services.

— **Indépendance vis-à-vis des fournisseurs :** L'utilisation d'équipements et de logiciels ouverts permet aux administrateurs de sélectionner les meilleures solutions sans être liés à un seul fabricant.

— **Accélération de l'innovation :** Le SDN encourage le développement et le test de nouvelles applications réseau sans impacter les performances globales, favorisant ainsi l'émergence de nouvelles technologies.

I.4.3 Inconvénients des réseaux SDN

Malgré leurs nombreux avantages, les réseaux SDN présentent également certaines limitations :

— **Complexité de mise en œuvre :** L'adoption du SDN nécessite des compétences avancées en programmation et en gestion réseau, ce qui peut représenter un obstacle pour certaines organisations.

— **Problèmes d'interopérabilité :** Tous les équipements et protocoles ne sont pas toujours compatibles avec le SDN, ce qui peut limiter le choix du matériel et complexifier l'intégration avec les infrastructures existantes.

— **Risque de cyberattaques :** Comme tout passe par le contrôleur, il devient une cible privilégiée pour les attaques, comme les attaques DdoS. Si le contrôleur est compromis l'ensemble du réseau peut être exposé. Cela pose donc de vrais défis en matière de sécurité.

I.5 Conclusion

À travers ce chapitre, nous avons exploré les fondements des Réseaux Définis par Logiciel (SDN) en mettant en lumière leur évolution, leur architecture, ainsi que les différences fondamentales avec

les réseaux traditionnels. Nous avons également souligné les principaux avantages qu'offre cette nouvelle approche, notamment en termes de flexibilité de programmabilité et de gestion centralisée. Cependant, bien que le SDN simplifie grandement l'administration des réseaux et améliore leur efficacité, il introduit également de nouveaux défis de sécurité. La séparation du plan de contrôle et du plan de données, ainsi que la centralisation du contrôle, peuvent exposer ces réseaux à des attaques spécifiques nécessitant des mécanismes de protection adaptés. Dans le chapitre suivant, nous nous intéresserons aux aspects de sécurité du SDN en identifiant ses vulnérabilités potentielles et en explorant les différentes solutions et approches proposées pour renforcer la protection de ces infrastructures.

CHAPITRE

II

VULNÉRABILITÉS ET SÉCURISATION DES RÉSEAUX SDN

II.1 Introduction

Bien que l'architecture des réseaux définis par logiciel (SDN) apporte de nombreux avantages, notamment en matière de flexibilité, de centralisation du contrôle et de gestion intelligente des ressources, elle n'en demeure pas moins exposée à de nouvelles vulnérabilités.
En effet, les spécificités du modèle SDN, telles que l'ouverture des interfaces de communication et la centralisation des décisions au niveau du contrôleur, introduisent des surfaces d'attaque qui peuvent être exploitées pour compromettre la stabilité la confidentialité ou encore la disponibilité du réseau. Dès lors, la sécurité des environnements SDN devient une préoccupation majeure, nécessitant des stratégies de défense adaptées et évolutives.
Dans ce chapitre, nous allons d'une part analyser en profondeur les principales vulnérabilités propres aux réseaux SDN, en nous intéressant aux différentes couches de l'architecture, à savoir le plan d'infrastructure, le plan de contrôle et la couche applicative. D'autre part, nous étudierons les solutions mises en œuvre pour protéger ces environnements en passant par des mécanismes traditionnels de sécurisation, des approches innovantes comme Avant-Guard, Flood-Guard ou SDN-Guard, ainsi que l'intégration de l'intelligence artificielle à travers des techniques d'apprentissage automatique. Par ailleurs, une attention particulière sera portée à la sécurisation des communications et des règles de

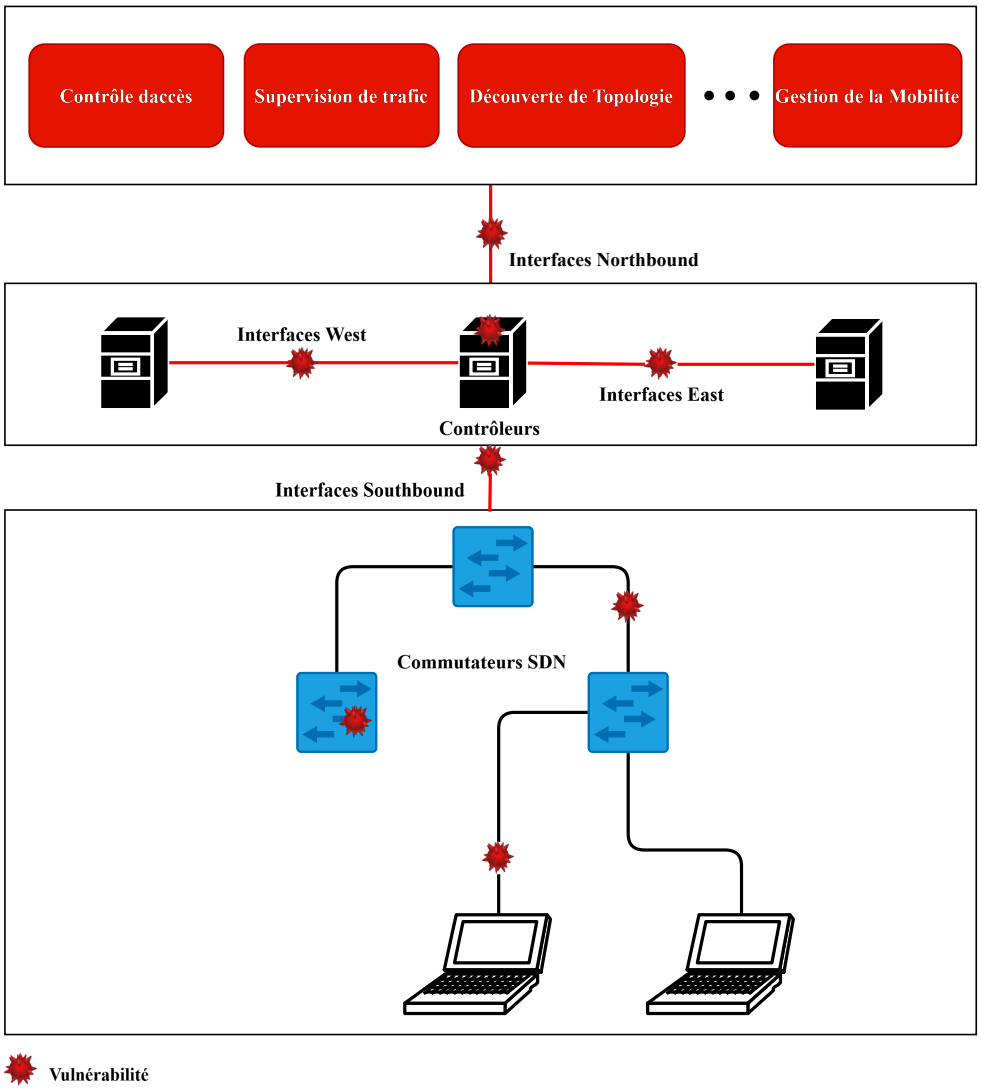

FIGURE II.1 – Vulnérabilités dans un réseau SDN

routage, éléments clés pour assurer la fiabilité et la résilience d'un réseau SDN face aux différentes menaces actuelles et émergentes.

II.2 Vulnérabilités des réseaux SDN

Les réseaux définis par logiciel offrent une flexibilité et une programmabilité améliorées par rapport aux architectures traditionnelles. Toutefois, cette nouvelle approche introduit également des vulnérabilités spécifiques [3] [26] pouvant être exploitées par des attaquants. La figure ci-dessous illustre les principaux points critiques du SDN, mettant en évidence les surfaces d'attaque potentielles. Cette figure met en évidence plusieurs vulnérabilités réparties au sein des trois plans fondamentaux de l'architecture SDN :

— **Le plan de contrôle** : vulnérabilités liées au contrôleur SDN et aux interfaces de communication.

— **Le plan d'infrastructure** : risques associés aux commutateurs SDN et aux équipements de réseau.

— **Le plan applicatif** : menaces provenant des applications exécutées au-dessus du SDN.

Dans cette section, nous procéderons à une analyse approfondie de ces vulnérabilités en identifiant leurs mécanismes, leurs impacts sur le réseau et les menaces qu'elles engendrent.

II.2.1 Vulnérabilités de la couche infrastructure

La couche Infrastructure, également appelée *Data Plane*, regroupe les dispositifs physiques ou virtuels responsables du transfert effectif des paquets dans le réseau SDN. Cette couche, tout en assurant la transmission des données, présente des vulnérabilités spécifiques que les attaquants peuvent exploiter pour compromettre la performance et la sécurité globale du réseau.

Saturation de la table de flux

Les réseaux SDN reposent sur l'utilisation de tables de flux dans les commutateurs pour gérer le traitement des paquets, ces tables contiennent des règles précises définissant les actions à effectuer pour chaque type de trafic. Toutefois, cette architecture présente des vulnérabilités critiques, notamment face aux attaques par « Flow Table Overflow ».
Ce type d'attaque consiste à saturer la table de flux en y injectant un nombre massif de requêtes dépassant ainsi ses capacités matérielles et logicielles.
Les attaquants exploitent cette faiblesse en générant de façon automatisée un volume élevé de requêtes de création de flux [27], souvent par le biais de scripts malveillants ou en profitant de failles dans la gestion de la mémoire du commutateur. Lorsque la table de flux est saturée, les nouveaux paquets nécessitant des règles spécifiques sont soit rejetés, soit transmis directement au contrôleur SDN. Cette surcharge provoque un goulot d'étranglement au niveau du contrôleur, augmentant la latence et les risques d'erreurs dans le traitement des paquets.
Le diagramme ci-dessous illustre l'impact d'une attaque par « Flow Table Overflow ». On y observe la progression du nombre de règles de flux expulsées au fil du temps, mettant en évidence comment les flux malveillants (en rouge) forcent l'expulsion des flux légitimes (en bleu). Cette saturation progressive compromet non seulement la performance mais aussi la sécurité du réseau SDN.

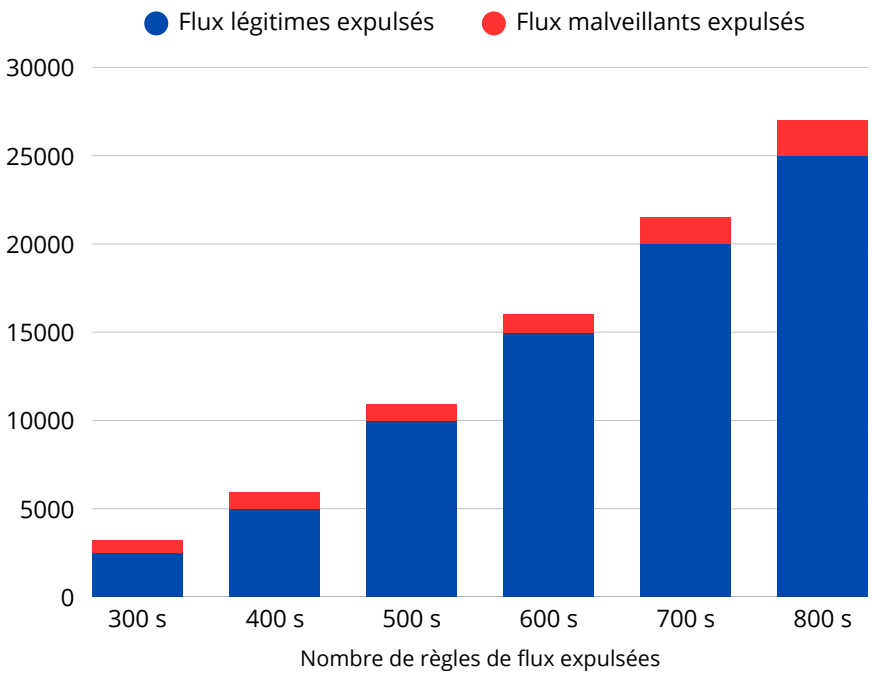

FIGURE II.2 – Expulsion des règles lors d'une attaque « Flow Table Overflow ».

Impacts sur la sécurité et la performance

La saturation des tables de flux entraîne plusieurs **conséquences majeures** :

— **Déni de service local ou global** : En empêchant l'installation de nouvelles règles, l'attaque par *Flow Table Overflow* peut paralyser totalement ou partiellement le réseau. Les paquets sont soit rejetés, soit redirigés vers le contrôleur, provoquant des délais importants dans le traitement.

— **Épuisement des ressources du contrôleur SDN** : La redirection massive de paquets vers le contrôleur engendre une surcharge qui affecte sa capacité à gérer efficacement les nouvelles requêtes. Cela expose le réseau à d'autres attaques, telles que les attaques par déni de service distribué (DDoS).

— **Masquage d'autres attaques** : Les attaquants peuvent utiliser cette saturation pour masquer d'autres intrusions, comme l'injection de paquets malveillants ou l'exfiltration de données. En exploitant la confusion créée, ils parviennent à contourner les mécanismes de détection classiques.

Attaques Man-in-the-Middle (MITM)

es attaques **Man-in-the-Middle** (MITM) [29] consistent à intercepter et manipuler les communications entre deux entités légitimes sans qu'elles s'en aperçoivent. Dans les réseaux SDN, ces attaques peuvent cibler les interfaces **Southbound** qui relient les contrôleurs aux commutateurs via des protocoles comme **OpenFlow**, La figure ci-dessous illustre la différence entre la connexion perçue par

l'hôte A et la connexion réelle manipulée par l'attaquant lors d'une attaque de type **Man-in-the-Middle** (MITM).

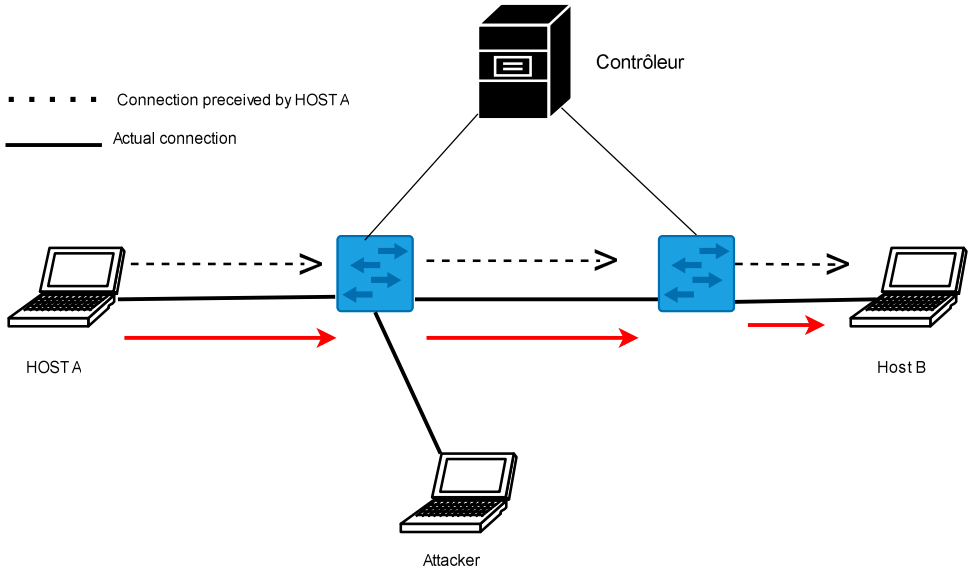

FIGURE II.3 – Attaque MITM dans un réseau SDN.

Cibles des attaques MITM dans les réseaux SDN

Les attaques de type *Man-in-the-Middle* peuvent cibler plusieurs segments clés du réseau SDN :

- **Entre les PC et les commutateurs** : Les attaquants peuvent intercepter les paquets circulant entre les utilisateurs et les commutateurs SDN. Ils peuvent :
  - *Écouter clandestinement* : Capturer des informations sensibles telles que des identifiants ou des mots de passe.
  - *Modifier des paquets* : Injecter des commandes malveillantes ou altérer les données en transit.
- **Entre les commutateurs (Switch-to-Switch)** : Les communications inter-commutateurs peuvent être compromises, permettant :
  - *Le reroutage malveillant* : Détourner le trafic vers des nœuds compromis.
  - *La suppression de paquets* : Perturber les applications sensibles en éliminant des paquets spécifiques.
- **Entre le contrôleur SDN et les commutateurs** : Les interfaces *Southbound* sont vulnérables aux attaques MITM. Un attaquant peut :
  - *Modifier les politiques de routage* : En interceptant les commandes du contrôleur, un attaquant peut modifier les règles de flux des commutateurs pour utiliser des routes non sécurisées.

— *Surcharger le contrôleur* : En saturant les tables de flux, l'attaquant peut provoquer une surcharge du contrôleur SDN.

II.2.2 Impacts des attaques MITM

— • **Compromission de la confidentialité** : Les attaquants peuvent accéder à des informations sensibles sans être détectés.

— • **Dégradation des performances** : La manipulation des routes et des tables de flux provoque une latence accrue et des congestions.

— • **Épuisement des ressources du contrôleur** : Les attaques MITM exploitent les interfaces Southbound pour diriger un volume massif de requêtes vers le contrôleur, épuisant rapidement ses ressources.

II.2.3 Attaques par Empoisonnement ARP

Address Resolution Protocol (ARP) [29] joue un rôle clé dans les réseaux en permettant d'associer une adresse IP à une adresse MAC. Cependant, ce protocole ne dispose d'aucune protection intégrée contre les attaques, ce qui le rend particulièrement vulnérable à l'empoisonnement ARP. Cette technique malveillante consiste à modifier frauduleusement les correspondances IP-MAC dans les tables ARP d'un commutateur ou d'un hôte, permettant ainsi à un attaquant de rediriger le trafic réseau vers une machine sous son contrôle.
Dans un environnement SDN, une telle attaque peut avoir des conséquences graves. En injectant des réponses ARP falsifiées, l'attaquant associe son adresse MAC à celle d'une victime, lui donnant la capacité d'intercepter, de modifier ou de bloquer les communications entre les équipements. Cette manipulation ouvre la voie à des activités malveillantes telles que l'espionnage, l'injection de paquets malveillants, ou même des attaques par déni de service (DoS) en saturant les tables ARP du commutateur. Comme illustré dans la figure ci-dessous, l'attaque par empoisonnement ARP exploite la gestion dynamique des flux dans un réseau SDN, permettant à un attaquant de manipuler les tables ARP et de rediriger le trafic de la victime.

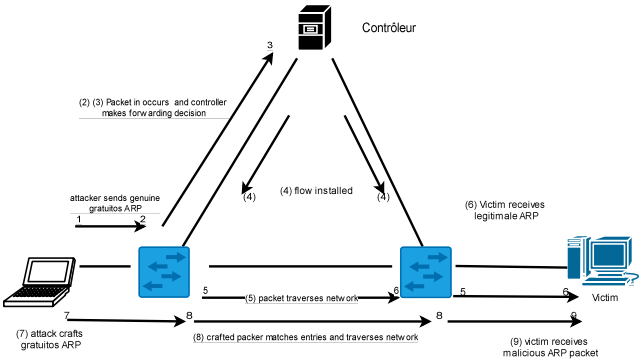

FIGURE II.4 – Schéma de l'attaque par empoisonnement ARP dans un réseau SDN

Impacts sur la sécurité et la performance

- **Compromission de la confidentialité :**

— L'attaque permet d'intercepter des informations sensibles (identifiants, mots de passe, communications privées).

— Les utilisateurs ne se rendent pas compte que leurs données sont détournées avant d'atteindre leur destination.

- **Altération des communications**

— L'attaquant peut modifier les paquets interceptés et y injecter du contenu malveillant.

— Il peut rediriger les utilisateurs vers des sites frauduleux en falsifiant les réponses ARP.

- **Dégradation des performances réseau**

— Un grand nombre de réponses ARP falsifiées sature les tables du commutateur, rendant les décisions de routage inefficaces.

— L'empoisonnement ARP peut provoquer des boucles réseau, entraînant une latence accrue et des interruptions de service.

II.3 Vulnérabilités de la couche de contrôle

La couche de contrôle ou **Control Plan** est le cœur du réseau SDN, centralisant les décisions via le contrôleur SDN. Cette centralisation améliore la gestion du réseau, mais expose le contrôleur à des attaques ciblées pouvant compromettre la disponibilité, l'intégrité et la sécurité du réseau. Une attaque réussie à ce niveau peut perturber le fonctionnement global du SDN. Cette couche est particulièrement vulnérable aux attaques suivantes :

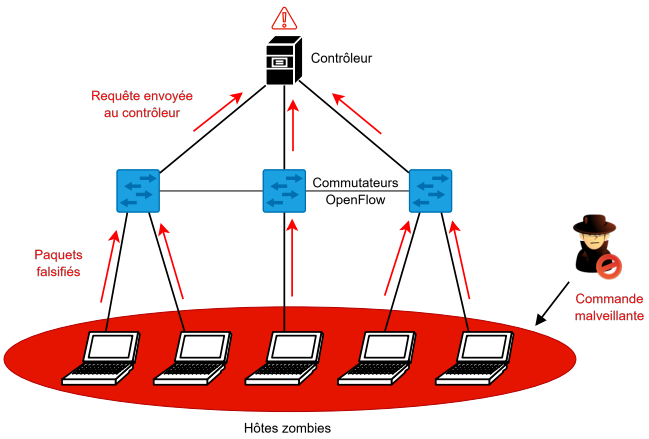

FIGURE II.5 – Illustration d'une attaque DdoS ciblant un contrôleur SDN

II.3.1 Attaques par déni de service

Les attaques par déni de service distribué (DdoS) constituent une menace critique pour la couche de contrôle des réseaux SDN. Ces attaques exploitent la centralisation du contrôleur qui gère la totalité des décisions de routage et de gestion du trafic.
En submergeant ce dernier sous un volume excessif de requêtes malveillantes, les attaquants parviennent à saturer ses ressources, notamment le processeur, la mémoire et la bande passante compromettant ainsi la stabilité et la disponibilité du réseau [8]. L'architecture centralisée du SDN le rend particulièrement vulnérable à ce type d'attaque car une seule entité doit traiter l'ensemble des requêtes des commutateurs du réseau. Lorsqu'un attaquant génère un trafic massif, le contrôleur devient incapable de répondre aux demandes légitimes, ce qui entraîne plusieurs conséquences :

— **Surcharge du contrôleur :** L'accumulation excessive de requêtes ralentit le traitement des flux et entraîne des pertes de paquets.

— **Perturbation des communications :** Les mises à jour des règles de routage sont retardées, provoquant des erreurs de transmission, des boucles réseau et une augmentation de la latence.

— **Interruption du service :** Dans les cas les plus graves, le contrôleur devient inopérant paralysant l'ensemble du réseau et empêchant tout acheminement du trafic.

En plus de perturber directement les performances du contrôleur, les attaques DdoS peuvent être utilisées comme diversion pour masquer d'autres types de cyberattaques. Une surcharge du contrôleur peut, par texemple, être exploitée pour rediriger le trafic vers des points d'écoute malveillants, mettant ainsi en péril la confidentialité et l'intégrité des données circulant sur le réseau.

II.3.2 Accès non autorisé au contrôleur SDN

Dans les réseaux SDN, le contrôleur joue un rôle central en orchestrant les communications entre les différentes couches du réseau. Toutefois, cette centralisation en fait une cible privilégiée pour les attaquants cherchant à obtenir un accès non autorisé. Lorsqu'un adversaire parvient à compromettre le contrôleur, il peut alors exercer un contrôle total sur l'ensemble du réseau, entraînant des conséquences graves sur la confidentialité, l'intégrité et la disponibilité des communications.
L'un des principaux vecteurs d'attaque repose sur l'exploitation des failles de sécurité [7]présentes dans les interfaces de communication du contrôleur, notamment les interfaces East/West.
Une mauvaise gestion des permissions ou l'absence de protocoles de sécurité robustes peut permettre à un attaquant d'envoyer des requêtes malveillantes directement au contrôleur, contournant ainsi les mécanismes de protection classiques. Une fois l'accès obtenu, il peut modifier ou supprimer les règles de routage, rediriger le trafic vers des destinations non autorisées, ou encore provoquer des boucles réseau qui entraînent une saturation des ressources.
Dans certains cas, un accès non autorisé peut également être utilisé pour intercepter des données sensibles en redirigeant les flux vers un serveur malveillant compromettant ainsi la confidentialité des échanges.

II.4 Vulnérabilités de la couche application

La couche application dans les réseaux SDN regroupe l'ensemble des applications et services qui interagissent avec le contrôleur via l'interface Northbound. Ces applications permettent d'enrichir la gestion du réseau en apportant des fonctionnalités avancées telles que la sécurité l'optimisation du trafic ou encore la virtualisation des ressources. Toutefois, cette flexibilité s'accompagne d'une vulnérabilité majeure pouvant être exploitée par des attaquants pour compromettre le fonctionnement du réseau. La principale vulnérabilité identifiée à ce niveau est la suivante :

II.4.1 Manipulation des API Northbound

Dans un environnement SDN, les applications communiquent avec le contrôleur par l'intermédiaire des API Northbound pour piloter le trafic et optimiser les ressources réseau. Cependant, si ces interfaces ne sont pas correctement sécurisées, elles peuvent représenter une faille critique.
Un attaquant peut alors exploiter une API [25] mal protégée pour envoyer des requêtes non autorisées, modifier ou supprimer des règles de routage, ou encore intercepter des données sensibles échangées avec le contrôleur.
Par exemple, une application malveillante pourrait rediriger le trafic réseau vers des serveurs compromis, facilitant ainsi des attaques de type homme-du-milieu (MITM). De plus, l'injection de règles

frauduleuses pourrait entraîner des boucles réseau ou provoquer un déni de service compromettant gravement la disponibilité et la stabilité du réseau SDN.
La protection des API Northbound est donc un enjeu essentiel pour garantir la sécurité de la couche application et assurer la fiabilité des communications au sein du réseau.

II.5 Sécurisation des réseaux SDN

L'architecture SDN expose le réseau à diverses menaces, nécessitant des stratégies de défense efficaces. Cette section présente les solutions clés permettant de protéger le contrôleur les règles de routage, les communications et d'intégrer des mécanismes intelligents de détection des attaques.

II.5.1 Protection contre les attaques DdoS et la surcharge du contrôleur

Les attaques DdoS et la surcharge du contrôleur sont des menaces majeures en SDN, pouvant perturber le réseau en saturant ses ressources. Pour y faire face, plusieurs solutions ont été développées afin de détecter et limiter ces attaques avant qu'elles n'impactent le système.

Avant-Guard

Avant-Guard [28] est une solution de sécurité innovante, développée pour renforcer la résilience du plan de contrôle des réseaux SDN contre les attaques par saturation en particulier les attaques de type TCP-SYN Flooding. Ces attaques visent à surcharger le contrôleur en envoyant un grand nombre de requêtes TCP incomplètes, ce qui entraîne un ralentissement important du réseau. Afin de contrer cette menace, Avant-Guard intègre une extension matérielle sur les commutateurs OpenFlow, incluant un proxy TCP qui joue un rôle crucial de filtre de sécurité. Lorsqu'un client tente d'établir une connexion TCP, ce proxy intercepte le paquet SYN, répond par un SYN-ACK, puis vérifie l'authenticité des deux entités avant d'autoriser la connexion. Seules les connexions validées sont transmises au contrôleur, ce qui permet de réduire la charge sur le plan de contrôle et de bloquer les tentatives d'inondation.

— Cette solution repose sur deux mécanismes fondamentaux :

— • **Un module de filtrage des connexions**, qui permet de bloquer les requêtes suspectes avant même qu'elles n'atteignent le contrôleur.

— • **Un mécanisme de gestion des sessions**, permettant de suivre l'état des connexions établies et d'identifier rapidement les comportements anormaux. Grâce à cette approche proactive, Avant-Guard améliore considérablement la protection contre les attaques DdoS basées sur le protocole TCP, et par conséquent, la stabilité du réseau. Cependant, elle présente certaines limites, telles qu'une inefficacité face aux attaques utilisant d'autres protocoles comme UDP et ICMP,

ainsi qu'une légère latence supplémentaire liée au processus de validation des connexions. Pour compenser ces limites, FloodGuard a été proposé comme une solution complémentaire, visant à optimiser la gestion des flux et à éviter la saturation du contrôleur, quelle que soit la nature du trafic malveillant.

Flood-Guard

— FloodGuard [31] est une solution de défense spécifiquement conçue pour protéger les réseaux SDN contre les attaques DdoS, en évitant la saturation des tables de commutation et la surcharge du contrôleur. Cette solution repose sur deux modules clés intégrés directement dans le contrôleur SDN :

— **Le module d'analyse proactive des règles de commutation**, qui permet de détecter et d'anticiper les attaques en ajustant dynamiquement les règles de commutation dès qu'une menace est identifiée. Cela permet de gérer efficacement le trafic réseau en temps réel.

— **Le module de migration de paquets**, dont le rôle est de rediriger temporairement le trafic suspect vers un cache dédié, tout en garantissant la fluidité des communications légitimes.
Lorsqu'une attaque DdoS est détectée, le module de migration isole les paquets malveillants et les dévie, tandis que le module d'analyse réajuste les règles de commutation en fonction de l'état actuel du réseau.
Ces règles optimisées sont ensuite attribuées aux commutateurs ce qui assure la stabilité du réseau, la continuité du service, et la neutralisation de la menace.
Ce mécanisme permet ainsi de maintenir une gestion fluide et efficace du trafic, tout en protégeant le réseau contre les attaques DdoS.

SDN-Guard

SDN-Guard [9] est une solution de sécurité avancée spécialement conçue pour protéger les réseaux SDN contre les attaques par déni de service, elle s'appuie sur une combinaison de mécanismes intelligents pour détecter, analyser et neutraliser les menaces en temps réel.
L'un de ses principaux atouts réside dans l'application de règles de pare-feu dynamiques, qui permettent de bloquer les adresses IP suspectes et de filtrer les paquets malveillants. Afin de réduire l'impact des attaques de type volumétrique, SDN-Guard intègre également des techniques de limitation de bande passante, contribuant à réguler le flux de trafic vers les cibles potentielles et à éviter la saturation des ressources réseau. Pensé pour être hautement évolutif, SDN-Guard est capable de s'adapter à des infrastructures de grande taille et de supporter efficacement un trafic important, tout en assurant la stabilité et les performances globales du réseau.

II.5.2 Apprentissage automatique

L'apprentissage automatique (Machine Learning) [2] constitue une approche intelligente de sécurisation, intégrant l'analyse du trafic et la détection automatique des comportements anormaux au sein des réseaux SDN.
En exploitant directement les statistiques collectées via le protocole OpenFlow, telles que le nombre de paquets échangés ou encore la vitesse d'apparition des adresses IP sources il devient possible d'identifier des flux suspects, de prédire les attaques et ainsi de renforcer la résilience du réseau.
À partir de ces données, des paramètres caractéristiques sont extraits, puis utilisés pour entraîner un modèle d'apprentissage automatique, tel qu'un modèle basé sur le deep learning, permettant ainsi de construire un système capable de classer les flux réseau de manière autonome.
Une fois l'entraînement terminé, le modèle analyse les flux en temps réel afin de distinguer le trafic légitime du trafic malveillant ; en cas de détection d'une anomalie, le contrôleur SDN intervient immédiatement en bloquant les flux d'attaque tout en maintenant la fluidité du trafic normal.

Le schéma ci-dessous illustre le processus de détection des attaques DdoS dans un réseau SDN à l'aide d'un modèle de Machine Learning, en passant par la collecte des flux, l'extraction des paramètres et la classification des attaques.

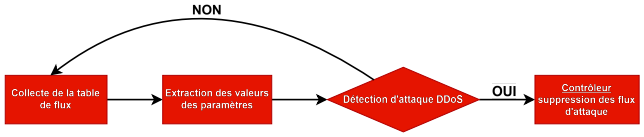

FIGURE II.6 – Processus de détection et mitigation des attaques DdoS en SDN via ML

II.5.3 Sécurisation des règles de routage et du contrôle d'accès

Le bon fonctionnement d'un réseau SDN repose sur l'intégrité des règles de routage et la gestion sécurisée des accès au contrôleur. Cependant, des attaques ciblant ces aspects peuvent entraîner des modifications malveillantes des règles, des détournements de flux ou des accès non autorisés.
Pour prévenir ces menaces, plusieurs solutions ont été mises en place afin de garantir l'authenticité des règles appliquées et de renforcer le contrôle des accès au sein du réseau SDN.

II.5.4 FortNOX

FortNOX [11] est une extension de sécurité conçue pour protéger les réseaux SDN contre l'injection de règles malveillantes et les accès non autorisés. Son rôle principal est d'empêcher toute modification indésirable des règles de routage en appliquant un contrôle strict des permissions et une gestion avancée des conflits.

Tout d'abord, FortNOX implémente un système d'autorisation basé sur les rôles, où chaque application Open Flow dispose de privilèges définis pour modifier ou ajouter des règles.
Ainsi, seules les applications autorisées peuvent interagir avec le contrôleur, réduisant ainsi le risque d'intrusion. Ensuite, il intègre un mécanisme de détection et de résolution des conflits qui analyse les nouvelles règles avant leur application. En cas de conflit avec une règle existante, FortNOX applique des politiques prédéfinies pour garantir la cohérence du réseau.
En complément, FortNOX repose sur une authentification des sources via des signatures numériques, permettant de vérifier l'identité des applications qui tentent d'installer des règles sur le contrôleur SDN. Ce mécanisme empêche ainsi tout acteur malveillant d'envoyer des requêtes frauduleuses.
Grâce à ces fonctionnalités, FortNOX améliore la sécurité des réseaux SDN en garantissant l'intégrité des règles de routage et en limitant les accès au contrôleur, ce qui le rend particulièrement efficace pour prévenir les attaques de manipulation des flux.

II.5.5 BroFlow

BroFlow [17]est une solution de détection et de prévention des attaques par déni de service (DoS) qui repose sur l'API OpenFlow et l'analyseur de trafic réseau, son objectif est d'identifier les anomalies dans le trafic et de déployer des contre-mesures adaptées pour bloquer les attaques avant qu'elles n'impactent le réseau.
Pour ce faire, BroFlow s'appuie sur un commutateur programmable Open vSwitch (OVS) fonctionnant comme un commutateur OpenFlow dont les règles de transfert sont mises à jour dynamiquement par un contrôleur SDN. Cette intégration permet une gestion centralisée et réactive des menaces. Plutôt que d'exiger un déploiement massif de capteurs sur chaque commutateur, BroFlow exploite la vue d'ensemble du réseau fournie par OpenFlow et utilise un nombre restreint de capteurs stratégiquement placés pour optimiser la surveillance et limiter les coûts d'infrastructure.
L'un des atouts majeurs de BroFlow réside dans l'utilisation du langage de politiques de sécurité de Bro, qui permet aux administrateurs réseau de définir des règles personnalisées pour la détection et la réponse aux incidents. Grâce à son analyse en temps réel, BroFlow est capable d'inspecter le trafic, générer des alertes et produire des rapports détaillés lorsqu'une activité suspecte est détectée. Cette approche renforce la résilience des réseaux SDN face aux cybermenaces en combinant analyse avancée du trafic et programmabilité du réseau.

II.5.6 SnortFlow

SnortFlow [32] est une solution qui intègre le système de détection d'intrusions Snort aux réseaux définis par logiciel (SDN) grâce au protocole OpenFlow, cette combinaison permet de surveiller le trafic réseau en temps réel et d'agir immédiatement en cas de détection de comportements suspects ou

d'attaques connues.
Lorsqu'une menace est identifiée, Snort transmet l'information au contrôleur SDN, qui met à jour dynamiquement les règles de flux afin de bloquer le trafic malveillant avant qu'il n'atteigne sa cible. En centralisant la gestion de la sécurité au niveau du contrôleur, SnortFlow offre une meilleure visibilité sur l'ensemble du réseau et facilite la coordination des réponses aux incidents. Sa conception évolutive lui permet également de s'adapter aux besoins croissants des infrastructures modernes.
En combinant la capacité d'analyse de Snort avec la flexibilité du SDN, SnortFlow propose ainsi une approche efficace et intelligente pour renforcer la sécurité des réseaux.

II.6 Sécurisation des communications et des échanges entre les équipements

Dans une architecture SDN, la sécurité des échanges entre les composants notamment entre le contrôleur, les commutateurs et les applications du plan de contrôle représente un enjeu majeur. La nature centralisée du SDN rend ces communications particulièrement sensibles à des menaces telles que l'interception de paquets ou les attaques de type Man-in-the-Middle.
Il est donc essentiel de mettre en place des mécanismes de protection adaptés, afin d'assurer la confidentialité, l'intégrité et l'authenticité des données échangées.
Parmi ces mécanismes, l'authentification mutuelle [19]constitue une première barrière de sécurité indispensable. Elle permet aux équipements réseau et au contrôleur de vérifier réciproquement leur identité avant toute tentative de communication.
Ce processus repose généralement sur l'utilisation de certificats numériques émis par une autorité de certification (CA), ainsi que sur des clés privées et signatures numériques.
L'authentification se déroule en trois étapes :

— **Échange des certificats** : chaque entité envoie son certificat numérique à l'autre.

— **Vérification des identités** : les certificats sont validés via la CA, et les signatures numériques permettent de confirmer l'authenticité de chaque partie.

— **Établissement d'un canal sécurisé** : une fois la vérification terminée, la session de communication peut démarrer de manière chiffrée.

Une fois l'identité des entités confirmée, il devient alors nécessaire de protéger le contenu même des échanges. C'est ici que les protocoles de chiffrement comme TLS et IPsec interviennent.
Le protocole TLS (Transport Layer Security) [23] est largement utilisé dans les réseaux SDN pour sécuriser les communications entre les composants logiciels. TLS assurela confidentialité et l'intégrité des données échangées en établissant un canal chiffré entre les entités.
En plus de cela, il prend en charge l'authentification mutuelle évoquée précédemment renforçant

ainsi la fiabilité des connexions dans l'infrastructure réseau.
Grâce à TLS les paquets échangés sont protégés contre toute tentative d'interception ou de modification.
Concernant IPsec (Internet Protocol Security) [18], il s'agit d'une suite de protocoles conçue pour sécuriser les communications au niveau de la couche réseau (couche 3 du modèle OSI). IPsec garantit plusieurs fonctions essentielles, telles que le chiffrement des paquets IP la vérification de leur intégrité et l'authentification de l'origine des données. Dans un environnement SDN distribué, où les commutateurs et le contrôleur peuvent être déployés sur des sites géographiquement séparés, IPsec constitue une solution efficace pour protéger les échanges interdomaines tout en maintenant un haut niveau de sécurité réseau. En combinant authentification mutuelle, chiffrement des flux via TLS, et protection au niveau IP avec IPsec, les réseaux SDN peuvent atteindre un niveau de sécurité élevé, tout en conservant la flexibilité et la programmabilité qui font leur force.

II.7 Conclusion

À travers ce chapitre, nous avons tout d'abord exploré les différentes vulnérabilités auxquelles les réseaux SDN peuvent être exposés, en abordant notamment les menaces pesant sur les couches infrastructure, contrôle et application.
Nous avons ensuite présenté plusieurs solutions de sécurisation, allant des protections classiques contre les attaques de type DdoS jusqu'aux mécanismes plus intelligents basés sur l'apprentissage automatique, ainsi que des stratégies spécifiques pour protéger les communications, les règles de routage et les accès critiques.
Ces différentes approches permettent de renforcer considérablement la résilience et la sécurité globale des environnements SDN. Dans le prochain chapitre, nous passerons à une étude comparative pratique entre un réseau traditionnel et un réseau SDN, en analysant leurs performances respectives à travers des scénarios de tests concrets.

CHAPITRE

III

ÉTUDE PRATIQUE SUR LES RÉSEAUX CLASSIQUES ET SDN

III.1 Introduction

À l'ère de la transformation numérique, les réseaux d'entreprise doivent évoluer pour répondre à des besoins croissants en agilité, performance et sécurité. Pourtant, de nombreuses organisations, comme le groupe industriel Cevital, reposent encore sur des architectures réseau traditionnelles, basées sur des commutateurs et routeurs physiques configurés manuellement. Ces infrastructures, bien que stables, peinent à s'adapter aux exigences modernes : gestion complexe des VLANs, latence accrue en cas de congestion, et difficulté à implémenter des politiques de sécurité dynamiques.
Dans ce contexte, le Software-Defined Networking émerge comme une solution prometteuse. En séparant le plan de contrôle du plan de données, le SDN permet une gestion centralisée, programmable et optimisée du trafic. Mais cette innovation est-elle réellement plus performante qu'un réseau classique ? Pour répondre à cette question, ce chapitre propose une analyse comparative approfondie entre l'architecture traditionnelle de Cevital et une topologie SDN équivalente, simulée avec Mininet et OpenDaylight. Nous évaluerons des indicateurs clés tels que la latence (RTT), le taux de perte de paquets et l'efficacité du routage, afin d'identifier concrètement les avantages et limites de chaque approche.

III.2 Présentation de l'entreprise Cevital

Groupe industriel majeur en Algérie, Cevital domine les secteurs agroalimentaire et industriel. Notre analyse se concentrera sur son organisation et sa direction des systèmes d'information élément clé de sa performance.

III.2.1 Historique de l'entreprise

Cevital a été fondée en 1998 par Issad Rebrab avec pour objectif initial de se positionner dans le secteur agroalimentaire. L'entreprise a rapidement connu une croissance remarquable notamment grâce à l'installation de ses premières unités de production spécialisées dans le raffinage de sucre, la fabrication d'huile et de margarine. Fort de son succès, cevital a progressivement diversifié ses activités en investissant dans l'industrie, le commerce et la grande distribution. Les étapes clés de son développement sont les suivantes :

— 1998 : Création de Cevital et lancement des premières unités de production agroalimentaire à Bejaïa.

— 2007 : Expansion industrielle avec l'ouverture de nouvelles unités, notamment une raffinerie de sucre et une usine d'embouteillage d'eau minérale.

— 2012 : Développement à l'international avec l'acquisition d'usines à l'étranger et l'extension des exportations vers l'Afrique et l'Europe.

— 2018 : Renforcement des investissements dans le secteur automobile et la logistique marquant une nouvelle phase de diversification et d'industrialisation. Aujourd'hui, Cevital est l'un des principaux groupes économiques en Algérie et continue d'élargir son influence à l'échelle internationale.

III.2.2 Présentation générale de l'entreprise

L'organisation de Cevital repose sur plusieurs directions stratégiques qui permettent une gestion efficace et une coordination optimale des activités. Parmi ces directions on distingue :

— Direction financière : Assure la gestion des ressources financières et la planification des investissements.

— Direction commerciale : Responsable de la stratégie de vente, du marketing et des relations avec les clients et fournisseurs.

— Direction des ressources humaines : Gère le recrutement, la formation et le bien-être des employés.

— Direction des systèmes d'information : Pilote les infrastructures informatiques et le développement des solutions technologiques.

III.2.3 Organigramme de Cevital

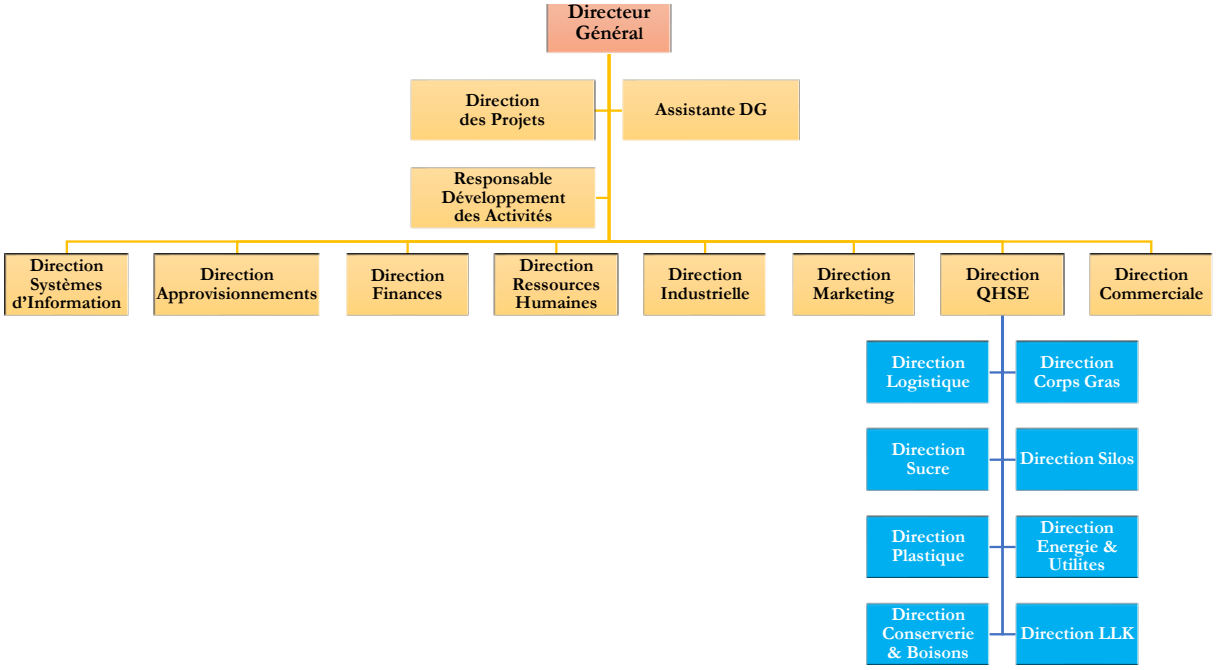

FIGURE III.1 – Organigramme général de CEVITAL

III.2.4 Organigramme de la direction système d'information

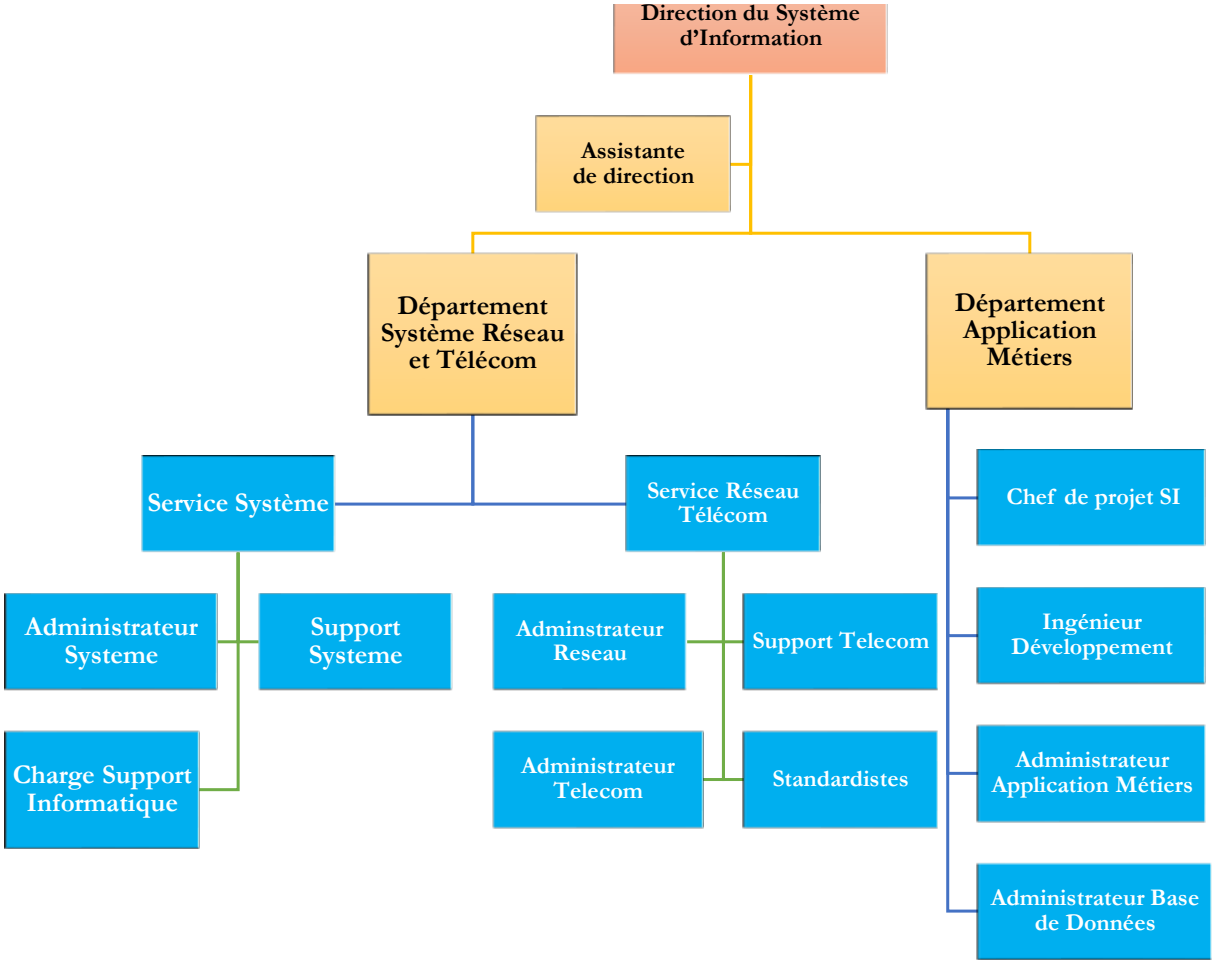

FIGURE III.2 – Organigramme de la direction informatique

III.3 Architecture du réseau traditionnel observé en entreprise

Cette section présente l'architecture du réseau traditionnel observé chez CEVITAL. Nous commencerons par décrire son infrastructure, puis nous aborderons brièvement sa configuration, notamment l'implémentation des VLANs, le routage inter-VLAN le protocole VTP et les mécanismes de sécurisation. Enfin, nous proposerons une étude d'une topologie simplifiée pour analyser et valider certaines configurations.

III.3.1 Description de l'infrastructure réseau de Cevital

L'architecture réseau de CEVITAL repose sur une infrastructure hiérarchique à trois couches pour assurer la connectivité, la sécurité et la performance du réseau. L'objectif est de garantir une communication fluide entre les sites de l'entreprise tout en sécurisant les échanges de données. **Couche Core** : Le Cœur du Réseau

— Responsable du routage principal entre les différentes zones du réseau.

— Utilisation de liens fibre optique à haut débit pour minimiser la latence.

— Protocoles de routage avancés comme OSPF et BGP pour optimiser l'acheminement du trafic.

— Redondance et haute disponibilité grâce à des mécanismes comme HSRP ou VRRP.

Couche Distribution : L'Interface Intermédiaire

— Regroupe les connexions entre la couche Core et la couche Accès.

— Applique des politiques de sécurité et de segmentation via ACLs et VLANs.

— Gestion centralisée des VLANs avec VTP (VLAN Trunking Protocol).

— Prévention des boucles réseau via STP (Spanning Tree Protocol).

Couche Accès : La Connexion des Équipements

— Assure la connexion des postes clients, serveurs et équipements IoT.

— Sécurisation avec 802.1X, port-security et BPDU Guard.

— Mise en place de priorités de trafic (QoS) pour la VoIP et la vidéosurveillance.

Routage et Interconnexion des Sites

L'interconnexion des sites de CEVITAL repose sur trois technologies principales : • **MPLS (Multiprotocol Label Switching) :**

— Fournit une connexion privée et performante entre les différents sites.

— Optimise l'acheminement des paquets en réduisant la latence.

— Assure une isolation du trafic entre les succursales grâce à la segmentation MPLS.

• **VPN IPsec (Internet Protocol Security) :**

— Sécurise les échanges de données entre les sites en les chiffrant.

— Permet une extension sécurisée du réseau privé sur Internet.

— Utilisé principalement pour les connexions externes et les accès distants.

• **Carte GDL (Global Data Link)** :

— Permet une connectivité alternative via des réseaux mobiles (4G).

— Sert de solution de secours en cas de panne des liens principaux.

— Utilisée pour les sites distants où la fibre optique n'est pas disponible.

III.3.2 Étude d'une topologie simplifiée

Présentation de la topologie simplifiée

L'architecture réseau de Cevital étant particulièrement vaste et complexe, il n'est pas envisageable de la reproduire intégralement dans ce mémoire. Pour cette raison, une topologie simplifiée a été conçue afin de représenter les principaux mécanismes mis en place dans l'entreprise. Cette version réduite du réseau conserve les mêmes principes fondamentaux notamment la segmentation en VLANs, le routage inter-VLAN, ainsi que diverses mesures de sécurisation du réseau. Cette topologie servira de base pour une étude approfondie de la configuration réseau permettant d'illustrer le fonctionnement des protocoles utilisés et de valider leur efficacité à travers des tests de connectivité et d'administration du trafic.

Outils et matériel utilisés

Logiciel de simulation

Pour la simulation et la configuration du réseau, nous avons utilisé Cisco Packet Tracer. Ce logiciel d'émulation permet de reproduire un environnement réseau avec des équipements configurables, facilitant ainsi les tests et la validation des configurations avant un déploiement réel.

Équipements réseau simulés

La topologie du réseau repose sur plusieurs équipements réseau, chacun jouant un rôle spécifique dans l'architecture globale. Avant de détailler les équipements utilisés, la figure ci-dessous présente la topologie réseau étudiée dans cette simulation :

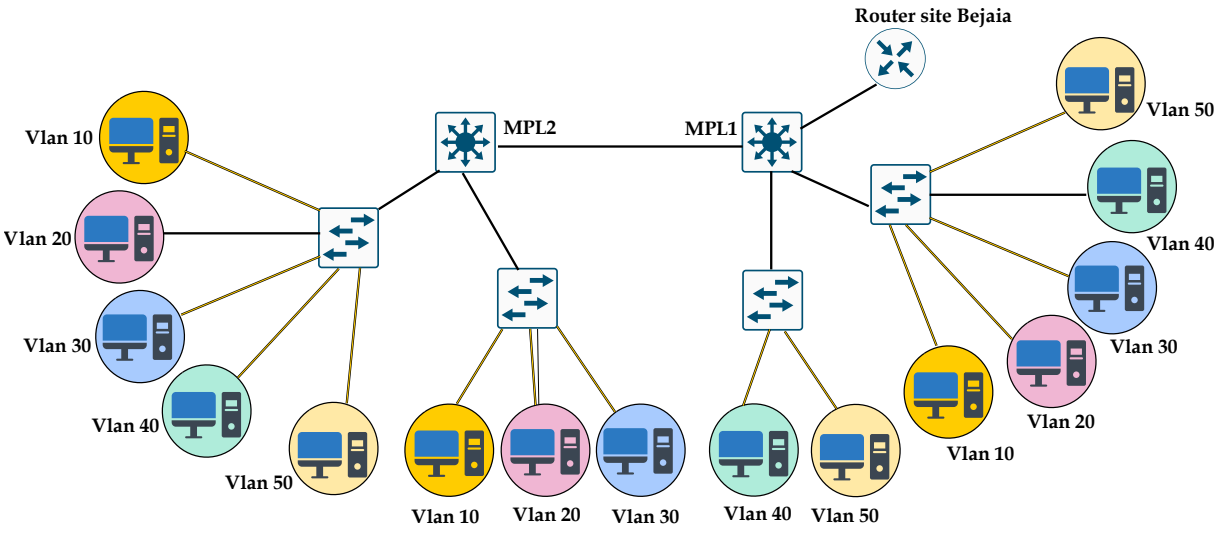

FIGURE III.3 – Topologie simplifiée du réseau simulé

Les équipements utilisés sont décrits comme suit :

— **Routeur** : Un routeur a été utilisé pour gérer l'interconnexion entre les différents VLANs et assurer le routage inter-VLAN.

— **Switches de niveau 3 (multi-layer switches)** : Deux switches ont été intégrés pour gérer la distribution du réseau et assurer le rôle de serveurs VTP facilitant la gestion centralisée des VLANs. L'utilisation de deux switches VTP permet d'assurer une redondance en cas de panne de l'un d'entre eux.

— **Switches d'accès** : Quatre switches d'accès ont été utilisés pour connecter les postes clients et segmenter le réseau en plusieurs VLANs.

— **Postes clients (PCs)** : Plusieurs PC ont été intégrés, chacun appartenant à un VLAN spécifique, simulant ainsi un réseau d'entreprise structuré et sécurisé.

III.3.3 Implémentation et Configuration du Réseau

Cette partie détaille l'adressage IP, la configuration des switches de niveau 3, la mise en place du protocole VTP, la création des VLANs dans les switches de distribution (serveurs VTP), l'attribution des interfaces aux switches d'accès (clients VTP), la configuration du routage inter-VLAN, ainsi que les mécanismes de sécurisation tels que le Spanning Tree Protocol (STP) et la Port Security.

Attribution des adresses IP aux postes clients

Chaque poste client a reçu une adresse IPv4 de classe C avec une passerelle par défaut correspondant à l'adresse du VLAN sur le switch de distribution. Le tableau ci-dessous présente l'attribution des adresses IP aux postes clients en fonction de leur VLAN et du switch auquel ils sont connectés.

Configuration des switches de niveau 3 et mise en place du protocole VTP

Pour assurer une gestion centralisée des VLANs et simplifier la configuration du réseau le protocole VTP a été mis en place. Les switches de distribution ont été configurés en mode serveur VTP, tandis que les switches d'accès ont été configurés en mode client VTP.
Les étapes suivantes ont été réalisées pour configurer le VTP :

1. Activation du mode serveur VTP sur les switches de distribution :

```
SD-DIS2# configure terminal
SD-DIS2(config)# vtp mode server
SD-DIS2(config)# vtp domain Cevital
SD-DIS2(config)# vtp password Cevital
SD-DIS2(config)# end
SD-DIS2# write memory
```

Switch	**VLAN**	**Adresse IP du poste client**	**Passerelle (Gateway)**
Switch 1	10	192.168.10.10	192.168.10.1
Switch 1	20	192.168.20.10	192.168.20.1
Switch 1	30	192.168.30.10	192.168.30.1
Switch 1	40	192.168.40.10	192.168.40.1
Switch 1	50	192.168.50.10	192.168.50.1
Switch 2	10	192.168.10.11	192.168.10.1
Switch 2	20	192.168.20.11	192.168.20.1
Switch 2	30	192.168.30.11	192.168.30.1
Switch 3	10	192.168.10.12	192.168.10.1
Switch 3	40	192.168.40.12	192.168.40.1
Switch 3	50	192.168.50.12	192.168.50.1
Switch 4	10	192.168.10.13	192.168.10.1
Switch 4	20	192.168.20.13	192.168.20.1
Switch 4	30	192.168.30.13	192.168.30.1
Switch 4	40	192.168.40.13	192.168.40.1
Switch 4	50	192.168.50.13	192.168.50.1

TABLE III.1 – Attribution des adresses IP aux postes clients

SD-DIS2> enable

2. Configuration du mode client VTP sur les switches d'accès :

```
Switch> enable
Switch# configure terminal
Switch(config)# vtp mode client
Switch(config)# vtp domain Cevital
Switch(config)# vtp password Cevital
Switch(config)# end
Switch# write memory
```

Les figures ci-dessous affichent l'état du protocole VTP après la configuration. La première figure montre la configuration du switch de niveau 3 en mode serveur, tandis que la seconde illustre la configuration du switch d'accès en mode client. Ces informations sont obtenues à l'aide de la commande « show vtp status », qui permet de vérifier le mode VTP le nom du domaine, ainsi que d'autres paramètres liés à la gestion des VLANs.

```
SD-DIS2>show vtp status
VTP Version capable             : 1 to 2
VTP version running             : 1
VTP Domain Name                 : cevital
VTP Pruning Mode                : Disabled
VTP Traps Generation            : Disabled
Device ID                       : 000A.F3E1.5900
Configuration last modified by 0.0.0.0 at 3-1-93 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN :
--------------
VTP Operating Mode                : Server
Maximum VLANs supported locally   : 1005
Number of existing VLANs          : 10
Configuration Revision            : 130
MD5 digest                        : 0x4C 0xB7 0x0D 0x29 0xE7 0x15 0xDB 0xBE
                                    0x05 0x13 0xBD 0xDD 0xC5 0x80 0xA8 0x0B
SD-DIS2>
```

FIGURE III.4 – Affichage du statut VTP sur le switch serveur

```
Switch>
Switch>show vtp status
VTP Version capable             : 1 to 2
VTP version running             : 1
VTP Domain Name                 : cevital
VTP Pruning Mode                : Disabled
VTP Traps Generation            : Disabled
Device ID                       : 0000.0C02.DD00
Configuration last modified by 0.0.0.0 at 3-1-93 00:00:00

Feature VLAN :
--------------
VTP Operating Mode                : Client
Maximum VLANs supported locally   : 255
Number of existing VLANs          : 10
Configuration Revision            : 130
MD5 digest                        : 0x4C 0xB7 0x0D 0x29 0xE7 0x15 0xDB 0xBE
                                    0x05 0x13 0xBD 0xDD 0xC5 0x80 0xA8 0x0B
Switch>
```

FIGURE III.5 – Affichage du statut VTP sur le switch client

Création des VLANs dans les switches de distribution

Les VLANs ont été créés uniquement sur les switches de distribution (serveurs VTP) afin qu'ils soient propagés automatiquement aux switches d'accès (clients VTP). La figure ci-dessous affiche la liste des VLANs configurés sur les switches de distribution, obtenue à l'aide de la commande « **show vlan brief** ».

```
SD-DIS2>show vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                Gig0/1, Gig0/2
10   DSI                              active
20   finance                          active
30   GRH                              active
40   logistique                       active
50   direction                        active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
SD-DIS2>
```

FIGURE III.6 – Affichage des VLANs configurés sur les switches de distribution

Dans cette étape, les interfaces des switches d'accès ont été attribuées aux VLANs correspondants en mode access pour les connexions aux postes clients et en mode trunk pour les liaisons entre switches ainsi que celles reliant les switches de distribution aux routeurs.
Plutôt que de détailler chaque configuration individuellement, la commande « show interfaces status » a été utilisée pour afficher un aperçu global de l'attribution des interfaces aux VLANs. Cette commande permet de vérifier que :

1. Chaque interface est correctement assignée à un VLAN spécifique en mode access.
2. Les interfaces assurant la liaison entre switches et vers le routeur sont bien configurées en mode trunk.

La figure ci-dessous montre le résultat de cette commande exécutée sur un switch d'accès mettant en évidence l'affectation des interfaces aux VLANs ainsi que les interfaces configurées en mode trunk.

```
Switch>en
Switch#show interfaces status
Port      Name               Status       Vlan       Duplex  Speed Type
Fa0/1                        connected    10         auto    auto  10/100BaseTX
Fa0/2                        notconnect   10         auto    auto  10/100BaseTX
Fa0/3                        connected    trunk      auto    auto  10/100BaseTX
Fa0/4                        notconnect   10         auto    auto  10/100BaseTX
Fa0/5                        notconnect   10         auto    auto  10/100BaseTX
Fa0/6                        connected    20         auto    auto  10/100BaseTX
Fa0/7                        notconnect   20         auto    auto  10/100BaseTX
Fa0/8                        notconnect   20         auto    auto  10/100BaseTX
Fa0/9                        notconnect   20         auto    auto  10/100BaseTX
Fa0/10                       notconnect   20         auto    auto  10/100BaseTX
Fa0/11                       connected    30         auto    auto  10/100BaseTX
Fa0/12                       notconnect   30         auto    auto  10/100BaseTX
Fa0/13                       notconnect   30         auto    auto  10/100BaseTX
Fa0/14                       notconnect   30         auto    auto  10/100BaseTX
Fa0/15                       notconnect   30         auto    auto  10/100BaseTX
Fa0/16                       connected    40         auto    auto  10/100BaseTX
Fa0/17                       notconnect   40         auto    auto  10/100BaseTX
Fa0/18                       notconnect   40         auto    auto  10/100BaseTX
Fa0/19                       notconnect   40         auto    auto  10/100BaseTX
Fa0/20                       notconnect   40         auto    auto  10/100BaseTX
Fa0/21                       connected    50         auto    auto  10/100BaseTX
Fa0/22                       notconnect   50         auto    auto  10/100BaseTX
Fa0/23                       notconnect   50         auto    auto  10/100BaseTX
Fa0/24                       notconnect   50         auto    auto  10/100BaseTX
Gig0/1                       notconnect   1          auto    auto  10/100BaseTX
Gig0/2                       notconnect   1          auto    auto  10/100BaseTX

Switch#
```

FIGURE III.7 – Affichage de l'attribution des interfaces aux VLANs et des liens trunk

Configuration du Routage Inter-VLAN

Le routage inter-VLAN a été configuré sur le routeur afin de permettre la communication entre les VLANs. Pour ce faire, l'interface GigabitEthernet0/0 a été subdivisée en plusieurs sous-interfaces, chacune correspondant à un VLAN spécifique. Chaque sous-interface a été associée à son VLAN respectif à l'aide de la commande encapsulation dot1Q et une adresse IP de passerelle a été attribuée pour assurer l'interconnexion des postes clients.
Les configurations ci-dessous assurent que chaque VLAN possède une passerelle, permettant aux appareils de communiquer entre eux via le routeur.

```
Router> enable
Router# configure terminal

Router(config)# interface GigabitEthernet0/0
Router(config-if)# no shutdown

Router(config)# interface GigabitEthernet0/0.10
Router(config-subif)# encapsulation dot1Q 10
Router(config-subif)# ip address 192.168.10.1 255.255.255.0

Router(config)# interface GigabitEthernet0/0.20
Router(config-subif)# encapsulation dot1Q 20
```

```
Router(config-subif)# ip address 192.168.20.1 255.255.255.0

Router(config)# interface GigabitEthernet0/0.30
Router(config-subif)# encapsulation dot1Q 30
Router(config-subif)# ip address 192.168.30.1 255.255.255.0

Router(config)# interface GigabitEthernet0/0.40
Router(config-subif)# encapsulation dot1Q 40
Router(config-subif)# ip address 192.168.40.1 255.255.255.0

Router(config)# interface GigabitEthernet0/0.50
Router(config-subif)# encapsulation dot1Q 50
Router(config-subif)# ip address 192.168.50.1 255.255.255.0

Router(config)# end
Router# write memory
```

La figure ci-dessous affiche la configuration des sous-interfaces et les adresses IP attribuées à l'aide de la commande show ip interface brief.

```
Router>show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0     unassigned      YES NVRAM  up                    up
GigabitEthernet0/0.10  192.168.10.1    YES manual up                    up
GigabitEthernet0/0.20  192.168.20.1    YES manual up                    up
GigabitEthernet0/0.30  192.168.30.1    YES manual up                    up
GigabitEthernet0/0.40  192.168.40.1    YES manual up                    up
GigabitEthernet0/0.50  192.168.50.1    YES manual up                    up
GigabitEthernet0/1     unassigned      YES NVRAM  administratively down down
GigabitEthernet0/2     unassigned      YES NVRAM  administratively down down
FastEthernet0/2/0      unassigned      YES unset  up                    down
FastEthernet0/2/1      unassigned      YES unset  up                    down
FastEthernet0/2/2      unassigned      YES unset  up                    down
FastEthernet0/2/3      unassigned      YES unset  up                    down
FastEthernet0/3/0      unassigned      YES unset  up                    down
FastEthernet0/3/1      unassigned      YES unset  up                    down
FastEthernet0/3/2      unassigned      YES unset  up                    down
FastEthernet0/3/3      unassigned      YES unset  up                    down
```

FIGURE III.8 – Résumé des sous-interfaces configurées sur le routeur

Sécurisation du réseau avec Spanning Tree Protocol et Port Security

Spanning Tree Protocol (STP) est configuré en mode Rapid PVST+ afin d'éviter les boucles dans le réseau en désactivant dynamiquement certains liens redondants. Ensuite, Port Security est mis en place sur les ports d'accès des switches pour limiter les adresses MAC apprises et renforcer la sécurité

contre les attaques par usurpation d'adresse MAC.

— Configuration de Spanning Tree Protocol (STP) en mode Rapid PVST+

```
Switch> enable
Switch# configure terminal
Switch(config)# spanning-tree mode rapid-pvst
Switch(config)# spanning-tree vlan 10-50 priority 24576
Switch(config)# exit
Switch# write memory
```

— Configuration de Port Security sur les ports d'accès

```
Switch> enable
Switch# configure terminal
Switch(config)# interface FastEthernet 0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security maximum 2
Switch(config-if)# switchport port-security violation restrict
Switch(config-if)# switchport port-security mac-address sticky
Switch(config-if)# exit
Switch(config)# exit
Switch# write memory
```

III.4 Implémentation d'une architecture SDN

Dans cette section, nous allons procéder à la conversion de la topologie réseau traditionnelle en une architecture SDN. Pour cela, nous présenterons tout d'abord les outils utilisés notamment Mininet et OpenDaylight, qui joueront respectivement le rôle d'émulateur de réseau et de contrôleur SDN. Ensuite, nous détaillerons les étapes d'installation et de configuration du contrôleur ainsi que la création de la topologie SDN.
Enfin, nous analyserons les performances du réseau SDN en réalisant des tests similaires à ceux effectués sur le réseau traditionnel.

III.4.1 Présentation des outils utilisés

Dans le cadre de cette implémentation, nous avons utilisé les outils suivants :

1. **Mininet :** un émulateur de réseau permettant de créer et tester des topologies SDN.

2. **OpenDaylight :** un contrôleur SDN open-source facilitant la gestion du réseau.
3. **Langages et frameworks nécessaires :** Python pour la création de la topologie et les commandes shell pour l'installation et la configuration.

Installation et configuration des outils

Mise à jour du système

Avant d'installer les outils, nous devons mettre à jour le système pour éviter tout conflit logiciel.

```
sudo apt update && sudo apt upgrade -y
```

Installation de Mininet

Nous installons Mininet avec la commande suivante :

```
sudo apt update && sudo apt upgrade -y
sudo apt install mininet -y
```

Nous testons ensuite son bon fonctionnement avec :

```
sudo mn --test pingall
```

Si le test est réussi, cela signifie que Mininet est correctement installé.

Installation et configuration d'OpenDaylight

Installation des dépendances requises

OpenDaylight nécessite Java 8, Git, Maven et unzip :

```
sudo apt install maven git openjdk-8-jre openjdk-8-jdk unzip -y
```

Nous définissons ensuite Java 8 comme version principale :

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
```

Téléchargement et installation d'OpenDaylight

Nous téléchargeons OpenDaylight (version 0.8.4) depuis le dépôt officiel :

```
sudo wget https://nexus.opendaylight.org/content/repositories/opendaylight
```

Une fois le fichier téléchargé, nous l'extrayons :

```
unzip karaf-0.8.4.zip
cd karaf-0.8.4
```

III.4.2 Démarrage et configuration d'OpenDaylight

Nous nous plaçons dans le dossier bin et lançons le contrôleur :

```
cd bin
./karaf
```

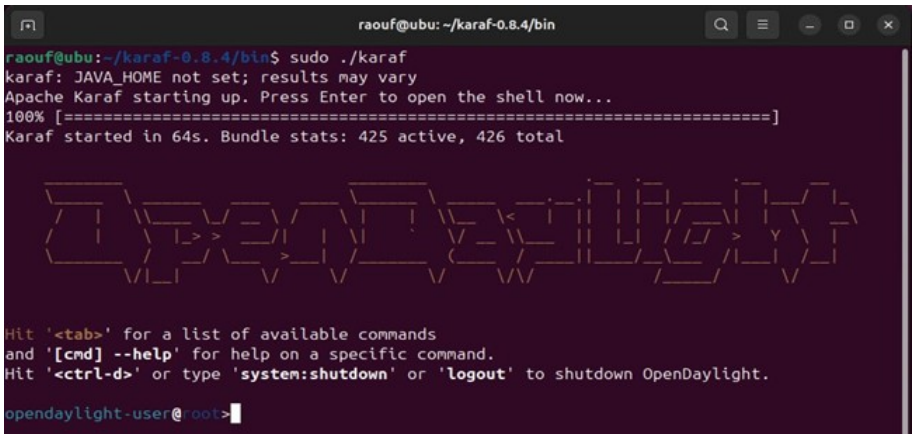

FIGURE III.9 – Lancement du contrôleur OpenDaylight

Activation des fonctionnalités OpenDaylight

Une fois le contrôleur démarré, nous devons activer plusieurs fonctionnalités essentielles :

```
feature:install odl-restconf
feature:install odl-l2switch-switch
feature:install odl-mdsal-apidocs
feature:install odl-dlux-core
feature:install odl-dluxapps-topology
feature:install odl-dluxapps-nodes
```

Ces modules permettent :

1. **odl-restconf :** InterfaceREST pour gérer le réseau.
2. **odl-l2switch-switch :** Support du protocole OpenFlow.
3. **odl-mdsal-apidocs :** Documentation API pour les services ODL.
4. **odl-dlux-core :** Interface graphique OpenDaylight (DLUX).
5. **odl-dluxapps-topology :** Visualisation des topologies réseau.
6. **odl-dluxapps-nodes :** Gestion des nœuds réseau.

Accès à l'interface graphique d'OpenDaylight

Pour accéder à l'interface de gestion du contrôleur, nous ouvrons un navigateur et entrons l'URL suivante : http ://172.20.10.2 :8181/index.html NB : Les identifiants par défaut sont admin / admin.

III.4.3 Création de la topologie SDN avec Mininet

La même topologie que le réseau classique est reproduite avec Mininet, en intégrant un contrôleur SDN (OpenDaylight). Elle comprend 6 switches, 15 hôtes répartis sur 5 VLANs et un routage inter-VLAN géré par le switch central. Voici le script permettant de déployer cette topologie :

```
from mininet.net import Mininet
from mininet.node import RemoteController, OVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink
import time

def create_network():
setLogLevel('info')

info('\n\n=== Démarrage de la création du réseau SDN ===\n')
info('*** Initialisation du réseau avec contrôleur OpenDaylight\n')
net = Mininet(controller=RemoteController,
switch=OVSSwitch,
link=TCLink,
autoSetMacs=True,
waitConnected=True)

info('\n=== PHASE 1: Configuration du contrôleur ===\n')
info('*** Ajout du contrôleur OpenDaylight [IP:172.20.10.2 Port:6633]\n')
c0 = net.addController('c0',
controller=RemoteController,
ip='172.20.10.2',
port=6633,
protocols='OpenFlow13')

info('\n=== PHASE 2: Création des switches ===\n')
```

```
info('*** Création de 6 switches (s1-s6)***\n')
switches = {}
for i in range(1, 7):
switches[f's{i}'] = net.addSwitch(f's{i}',
protocols='OpenFlow13',
stp=False)
info(f'    - Switch s{i} créé\n')

info('\n=== PHASE 3: Création des hôtes et VLANs ===\n')
info('''*** Configuration des 15 hôtes avec VLANs:
- VLAN 10: 192.168.10.0/24
- VLAN 20: 192.168.20.0/24
- VLAN 30: 192.168.30.0/24
- VLAN 40: 192.168.40.0/24
- VLAN 50: 192.168.50.0/24\n''')

hosts = [
# Switch s1
{'name': 'h1', 'ip': '192.168.10.10/24', 'gw': '192.168.10.1', 'switch': 's1', 'vla
{'name': 'h2', 'ip': '192.168.20.10/24', 'gw': '192.168.20.1', 'switch': 's1', 'vla
{'name': 'h3', 'ip': '192.168.30.10/24', 'gw': '192.168.30.1', 'switch': 's1', 'vla
{'name': 'h4', 'ip': '192.168.40.10/24', 'gw': '192.168.40.1', 'switch': 's1', 'vla
{'name': 'h5', 'ip': '192.168.50.10/24', 'gw': '192.168.50.1', 'switch': 's1', 'vla

# Switch s2
{'name': 'h6', 'ip': '192.168.10.11/24', 'gw': '192.168.10.1', 'switch': 's2', 'vla
{'name': 'h7', 'ip': '192.168.20.11/24', 'gw': '192.168.20.1', 'switch': 's2', 'vla
{'name': 'h8', 'ip': '192.168.30.11/24', 'gw': '192.168.30.1', 'switch': 's2', 'vla

# Switch s3
{'name': 'h9', 'ip': '192.168.10.12/24', 'gw': '192.168.10.1', 'switch': 's3', 'vla
{'name': 'h10', 'ip': '192.168.40.12/24', 'gw': '192.168.40.1', 'switch': 's3', 'vl
{'name': 'h11', 'ip': '192.168.50.12/24', 'gw': '192.168.50.1', 'switch': 's3', 'vl

# Switch s4
{'name': 'h12', 'ip': '192.168.10.13/24', 'gw': '192.168.10.1', 'switch': 's4', 'vl
```

```
{'name': 'h13', 'ip': '192.168.20.13/24', 'gw': '192.168.20.1', 'switch': 's4', 'vl
{'name': 'h14', 'ip': '192.168.30.13/24', 'gw': '192.168.30.1', 'switch': 's4', 'vl
{'name': 'h15', 'ip': '192.168.40.13/24', 'gw': '192.168.40.1', 'switch': 's4', 'vl
]

for h in hosts:
host = net.addHost(h['name'], ip=h['ip'])
net.addLink(host, switches[h['switch']])
info(f'    - Hôte {h["name"]} créé [IP:{h["ip"]} VLAN:{h["vlan"]} -> Switch {h["swi

info('\n=== PHASE 4: Connexion des switches ===\n')


net.addLink(switches['s1'], switches['s5'])
net.addLink(switches['s2'], switches['s5'])
net.addLink(switches['s3'], switches['s5'])
net.addLink(switches['s4'], switches['s5'])
net.addLink(switches['s5'], switches['s6'])
info('*** Tous les switches sont connectés avec s5 comme switch central\n')

info('\n=== PHASE 5: Démarrage du réseau ===\n')
net.build()

c0.start()
info('*** Contrôleur OpenDaylight démarré\n')

info('\n=== PHASE 6: Configuration des switches ===\n')
for name, sw in switches.items():
sw.start([c0])
sw.cmd('ovs-vsctl set bridge', name, 'stp_enable=false')
sw.cmd('ovs-vsctl set bridge', name, 'protocols=OpenFlow13')
info(f'    - Switch {name} démarré [STP désactivé, OpenFlow13 activé]\n')
time.sleep(1)

info('\n=== PHASE 7: Configuration du routage (s5) ===\n')
for vlan in [10, 20, 30, 40, 50]:
```

```
switches['s5'].cmd(f'ovs-vsctl␣add-port␣s5␣vlan{vlan}␣tag={vlan}␣--␣set␣interface␣
switches['s5'].cmd(f'ip␣addr␣add␣192.168.{vlan}.1/24␣dev␣vlan{vlan}')
switches['s5'].cmd(f'ip␣link␣set␣dev␣vlan{vlan}␣up')
info(f'␣␣␣␣-␣Interface␣VLAN{vlan}␣configurée␣[IP:192.168.{vlan}.1/24]\n')
time.sleep(0.5)

switches['s5'].cmd('sysctl␣-w␣net.ipv4.ip_forward=1')
info('***␣Routage␣IP␣activé␣sur␣s5\n')

info('\n===␣PHASE␣8:␣Configuration␣des␣hôtes␣===\n')
for h in hosts:
net.get(h['name']).cmd(f'ip␣route␣add␣default␣via␣{h["gw"]}')
info(f'␣␣␣␣-␣Route␣par␣défaut␣ajoutée␣pour␣{h["name"]}␣->␣{h["gw"]}\n')
time.sleep(0.1)



info('\n===␣RÉSEAU␣PRÊT␣===\n')
CLI(net)
net.stop()
info('\n===␣Réseau␣arrêté␣===\n')

if __name__ == '__main__':
create_network()
```

Pour exécuter le script et démarrer la simulation de notre topologie SDN, nous utilisons la commande suivante :

```
sudo python3 3.py
```

Une fois le script lancé, Mininet démarre le réseau et nous donne accès à son interface CLI où nous pouvons tester la connectivité entre les hôtes et observer le comportement du réseau.
La figure ci-dessous affiche l'interface CLI de Mininet après l'exécution du script.

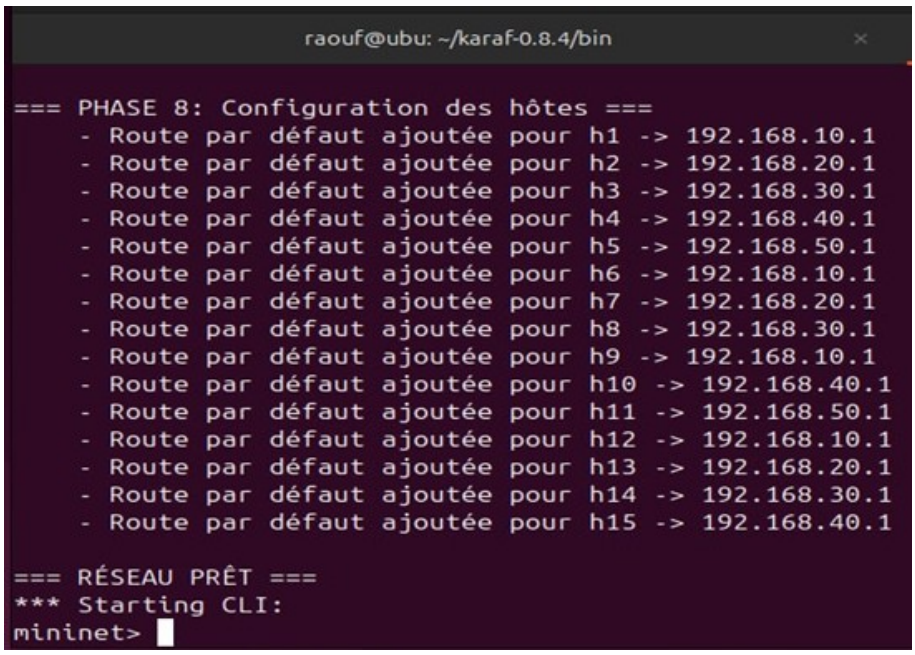

FIGURE III.10 – Interface CLI de Mininet après l'exécution du script

Visualisation de la topologie SDN

Pour mieux comprendre l'architecture de notre réseau, nous pouvons utiliser l'interface graphique d'OpenDaylight pour visualiser la topologie. La figure ci-dessous montre l'interface d'OpenDaylight ainsi que la topologie du réseau réalisé, permettant de visualiser les connexions entre les switches et les hôtes.

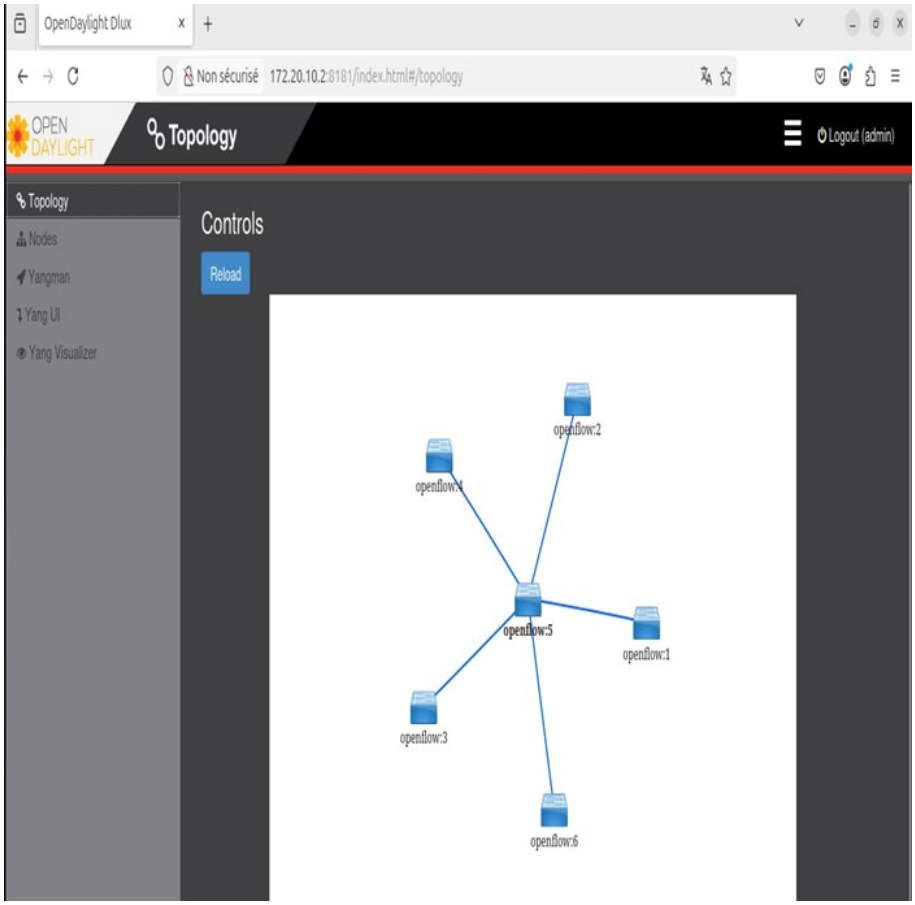

FIGURE III.11 – Interface d'OpenDaylight affichant la topologie du réseau SDN

III.5 Évaluation des Performances du Réseau

Cette section analyse les performances du réseau traditionnel et du réseau SDN en comparant des indicateurs comme le RTT, le taux de perte de paquets et la taille des paquets.

III.5.1 Test de Connectivité par Ping

Afin d'évaluer les performances des réseaux traditionnel et SDN, un test de connectivité a été effectué à l'aide de la commande ping. Ce test permet de mesurer trois indicateurs clés :

1. **Le Round-Trip Time (RTT) :** Temps nécessaire pour qu'un paquet atteigne sa destination et revienne à l'expéditeur.
2. **Le taux de perte de paquets :** Pourcentage de paquets envoyés mais non reçus en retour.
3. **La taille des paquets :** 32 octets dans le réseau traditionnel et 64 octets dans le réseau SDN.

Les Figures ci-dessous présentent les résultats obtenus après l'envoi de 100 paquets vers une machine cible dans chaque type de réseau. Les valeurs du RTT moyen, du RTT maximum et du taux de perte de paquets sont mises en évidence afin de faciliter l'analyse comparative.

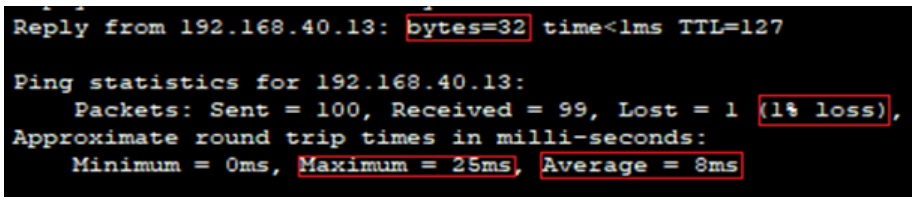

FIGURE III.12 – Résultats du test de connectivité par ping dans le réseau traditionnel

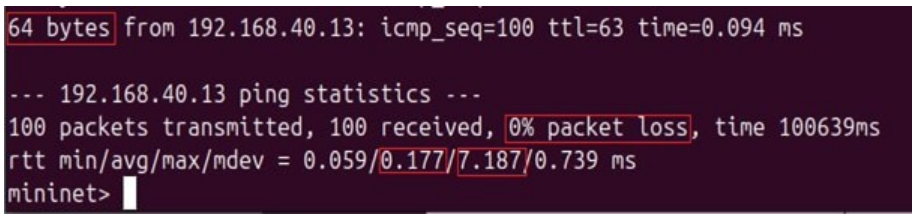

FIGURE III.13 – Résultats du test de connectivité par ping dans le réseau SDN

III.5.2 Analyse et Comparaison des Résultats

Les tests de connectivité effectués permettent d'évaluer les performances des deux architectures réseau en termes de latence, de taux de perte de paquets et de gestion du trafic. Le tableau ci-dessous résume ces mesures.

Paramètre	**Réseau Traditionnel**	**Réseau SDN**
Taille des paquets	32 octets	64 octets
RTT moyen	8 ms	0.177 ms
RTT maximum	25 ms	7.187 ms
Taux de perte	1%	0%

TABLE III.2 – Comparaison des performances réseau traditionnel vs SDN

Le SDN affiche un RTT moyen nettement inférieur à celui du réseau traditionnel, bien que les paquets transmis soient plus volumineux. Cette différence s'explique par l'optimisation du routage et la gestion centralisée du trafic, qui réduisent considérablement la latence et minimisent les pertes de paquets.

Dans le réseau traditionnel, la gestion distribuée introduit des délais supplémentaires notamment lors du traitement des paquets et de la prise de décision au niveau des équipements intermédiaires. En revanche, l'architecture SDN centralise ces décisions, permettant un acheminement plus efficace. Enfin, malgré l'augmentation de la taille des paquets dans le SDN, les performances restent supérieures, démontrant une meilleure gestion des ressources et une plus grande efficacité dans le traitement des flux.

III.6 Conclusion

Les résultats de cette étude confirment que le SDN apporte des avantages significatifs par rapport aux réseaux traditionnels. La centralisation du contrôle permet une réduction drastique de la latence (jusqu'à 45 fois moins) et une élimination quasi totale des pertes de paquets. De plus, la programmabilité des flux simplifie la gestion des VLANs et l'application de politiques de sécurité dynamiques, offrant une flexibilité inédite aux administrateurs réseau.

Cependant, cette transition vers le SDN n'est pas sans défis. La centralisation introduit un point de vulnérabilité critique : le contrôleur, cible privilégiée des attaques par déni de service (DoS). Pour garantir la robustesse de l'infrastructure, il devient indispensable d'y intégrer des mécanismes de protection avancés. C'est précisément l'objet du chapitre suivant, où nous exploiterons l'intelligence artificielle pour développer un système de détection d'intrusions en temps réel, combinant la puissance du SDN avec la réactivité du Machine Learning.

CHAPITRE

IV

DÉTECTION DES ATTAQUES DDOS PAR APPRENTISSAGE AUTOMATIQUE.

IV.1 Introduction

La sécurité dans les réseaux SDN représente aujourd'hui un véritable défi, notamment face aux attaques DdoS qui deviennent de plus en plus fréquentes et puissantes. Ce type d'attaque vise généralement à saturer le contrôleur SDN, qui est considéré comme l'élément central et le plus sensible dans cette architecture.

Dans le but de renforcer la sécurité et de mieux faire face à ces menaces, les techniques de Machine Learning sont de plus en plus exploitées. En effet, ces dernières offrent la possibilité d'analyser en profondeur le trafic réseau afin de détecter rapidement tout comportement suspect.

Dans ce chapitre, nous nous sommes donc intéressés à l'intégration du Machine Learning dans la détection des attaques DdoS au sein des réseaux SDN.

D'une part, nous avons abordé un volet théorique qui nous a permis de présenter les bases du Machine Learning ainsi que les principaux algorithmes utilisés pour la détection d'anomalies. D'autre part, nous avons mis en place l'environnement expérimental nécessaire au développement de notre solution.

Par la suite, nous avons détaillé le fonctionnement global de notre démarche à travers un workflow bien précis, composé de plusieurs étapes clés : la capture du trafic réseau l'entraînement du modèle,

et enfin la détection des attaques en temps réel.

Ainsi, ce chapitre regroupe l'ensemble des étapes pratiques réalisées pour la mise en œuvre de notre solution, en mettant en avant les résultats obtenus et les différents tests effectués afin de valider son efficacité.

IV.2 Comprendre le Machine Learning et son principe de fonctionnement

Cette section présente les fondements du Machine Learning, nous commencerons par définir ce concept et le comparer à la programmation classique. Ensuite, nous aborderons les types d'apprentissage, les concepts clés, puis les principaux algorithmes utilisés.

IV.2.1 Définition du Machine Learning

Le Machine Learning, ou apprentissage automatique, est un domaine clé de l'intelligence artificielle qui permet aux machines d'apprendre à partir des données, sans avoir à être explicitement programmées pour chaque tâche spécifique. L'idée principale est de permettre à un système d'identifier des modèles ou des comportements à partir d'exemples, afin de pouvoir faire des prédictions ou prendre des décisions de manière autonome [5, 10].

Contrairement à la programmation classique, où l'on définit manuellement les règles que l'ordinateur doit suivre pour produire un résultat à partir de données d'entrée, le **Machine Learning** adopte une approche inverse : on fournit à la machine un ensemble de données accompagnées des résultats attendus, et c'est elle qui va déduire les règles par elle-même. Ce changement de paradigme rend les systèmes plus flexibles et adaptatifs, en particulier dans des environnements complexes ou évolutifs [24].

Pour mieux illustrer cette différence, la figure ci-dessous montre de manière simplifiée la logique de fonctionnement dans les deux cas :

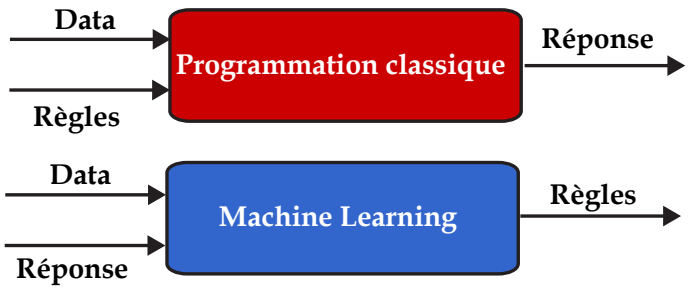

FIGURE IV.1 – Différence entre la programmation classique et le Machine Learning

IV.2.2 Types d'apprentissage en Machine Learning

Dans le domaine du Machine Learning, il existe principalement trois types d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Chaque approche a ses spécificités et est utilisée selon le type de problème à résoudre [5][10].

Apprentissage supervisé

L'apprentissage supervisé est sans doute la forme la plus utilisée du Machine Learning. Son principe est simple : on fournit à l'algorithme un ensemble de données d'entrée (features) ainsi que les réponses attendues (labels). Le but est que le modèle apprenne à faire le lien entre les entrées et les sorties, afin de pouvoir ensuite prédire les sorties sur de nouvelles données.
L'apprentissage supervisé permet de résoudre deux catégories de problèmes de Machine Learning :

1. **Les problèmes de régression :** il s'agit de prédire une valeur continue. Par exemple, prévoir le prix d'un produit en fonction de ses caractéristiques.
2. **Les problèmes de classification :** ici, le modèle prédit une catégorie ou une étiquette. Par exemple, déterminer si un e-mail est un spam ou non. La figure ci-dessous présente ces deux concepts

NB : Dans notre cas, la détection d'attaques DdoS dans un réseau SDN est un problème de classification, car le modèle doit déterminer si un trafic réseau est normal ou malveillant (attaque). L'utilisation de l'apprentissage supervisé est donc parfaitement adaptée [14].

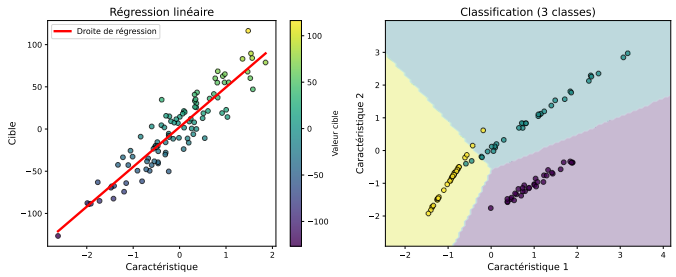

FIGURE IV.2 – Représentation schématique régression & classification

Apprentissage non supervisé

Dans l'apprentissage non supervisé, l'algorithme ne reçoit que les données d'entrée, sans étiquette ni résultat attendu. L'objectif est alors de découvrir des structures cachées ou des regroupements naturels dans les données.
Ce type d'apprentissage est souvent utilisé pour :

1. **Le clustering :** regrouper les données en différents groupes similaires.

2. **La détection d'anomalies :** identifier les comportements inhabituels sans savoir à l'avance ce qui est normal ou non.

Apprentissage par renforcement

L'apprentissage par renforcement repose sur un agent qui apprend à interagir avec un environnement en recevant des récompenses ou des pénalités selon ses actions. Le but est d'apprendre une stratégie optimale pour maximiser la récompense cumulée.
Ce type d'apprentissage est très utilisé dans les jeux, les robots autonomes ou encore la gestion dynamique des ressources.

IV.2.3 Concepts clés en Machine Learning

Avant de passer à l'implémentation, il est important de comprendre quelques notions fondamentales qui sont au cœur du fonctionnement du Machine Learning [10].

IV.2.4 Dataset

Le dataset est l'ensemble de données utilisé pour entraîner et tester un modèle. Il se présente sous forme d'un tableau contenant :

1. Des features (entrées : $x_1, x_2, ...$) qui représentent les caractéristiques observées.
2. Une target (sortie : y) qui correspond à ce qu'on veut prédire.

Par exemple, pour détecter une attaque DdoS, les features peuvent représenter le trafic réseau, et la target indique s'il s'agit d'un trafic normal ou malveillant.

IV.2.5 Modèle

Un modèle est une formule ou une structure mathématique que le système apprend à partir des données pour effectuer des prédictions. Plus les données sont pertinentes et bien préparées, plus le modèle sera performant.

IV.2.6 Paramètres et Hyperparamètres

— Les paramètres sont ajustés automatiquement par le modèle pendant l'apprentissage (ex. : poids dans une régression).

— Les hyperparamètres sont définis à l'avance (ex. : profondeur d'un arbre, nombre d'itérations) et influencent la qualité de l'entraînement.

IV.2.7 Training set et Test set

Le dataset est généralement divisé en deux parties :

— Training set : sert à entraîner le modèle (environ 70 %),

— Test set : permet d'évaluer ses performances sur des données qu'il ne connaît pas (environ 30 %).

IV.2.8 Biais et Variance

1. **Le biais :** désigne les erreurs que le modèle fait lorsqu'il n'arrive pas à bien apprendre à partir des données. Cela arrive souvent quand le modèle est trop simple : il généralise trop et ignore des détails importants.
2. **La variance :** au contraire, apparaît quand le modèle est trop sensible aux données d'entraînement. Il apprend « par cœur » ces données et a du mal à s'adapter à de nouvelles données qu'il n'a jamais vues. Cela entraîne des erreurs sur les données de test.

IV.2.9 Overfitting et Underfitting

1. **Overfitting :** quand le modèle s'adapte trop aux données d'entraînement, au point de perdre sa capacité à généraliser. Cela est souvent causé par une variance élevée.
2. **Underfitting** : quand le modèle est trop simple pour bien apprendre les relations entre les données. Il donne de mauvais résultats même sur les données d'entraînement. Cela est lié à un biais élevé.

IV.3 Principaux algorithmes de Machine Learning

Le Machine Learning utilise divers algorithmes [5] [10], adaptés à différents types de données et problèmes. Nous présentons ici une brève vue d'ensemble des méthodes supervisées et non supervisées les plus courantes.

IV.3.1 Algorithmes d'apprentissage supervisé

Ces algorithmes apprennent à partir d'un ensemble de données labellisées (avec les résultats connus à l'avance). Ils sont les plus adaptés à la détection des attaques, car ils peuvent être entraînés à reconnaître des modèles malveillants à partir d'exemples.

Régression linéaire

Cet algorithme cherche à établir une relation linéaire entre une variable d'entrée (x) et une variable de sortie (y). Il est souvent utilisé pour des problèmes de prédiction continue. La figure ci-dessous illustre un exemple simple de régression linéaire.

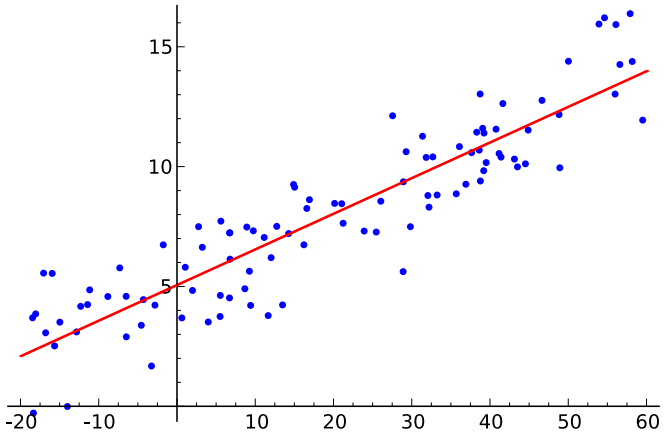

FIGURE IV.3 – Exemple de régression linéaire

Régression polynomiale

Lorsque la relation entre les données n'est pas linéaire, la régression polynomiale permet de modéliser des courbes plus complexes en introduisant des puissances de x. La figure suivante montrant une régression polynomiale appliquée à un jeu de données non linéaire.

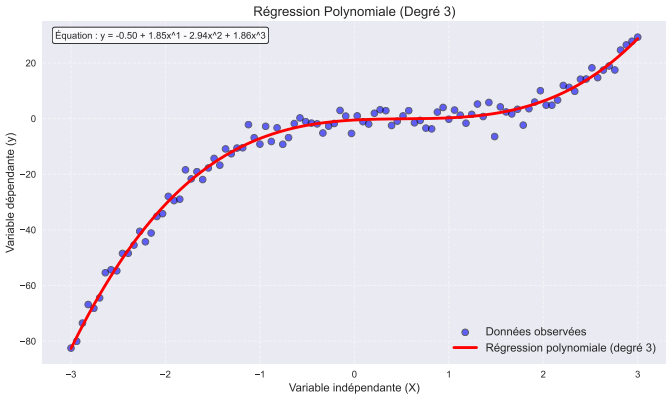

FIGURE IV.4 – Exemple de régression polynomial

Régression logistique

Contrairement aux précédentes, la régression logistique est utilisée pour des problèmes de classification binaire. Elle permet de prédire des classes comme "attaque" ou "normal" à partir des caractéristiques du trafic. La figure ci-dessus illustre un exemple avec une séparation claire entre deux

classes.

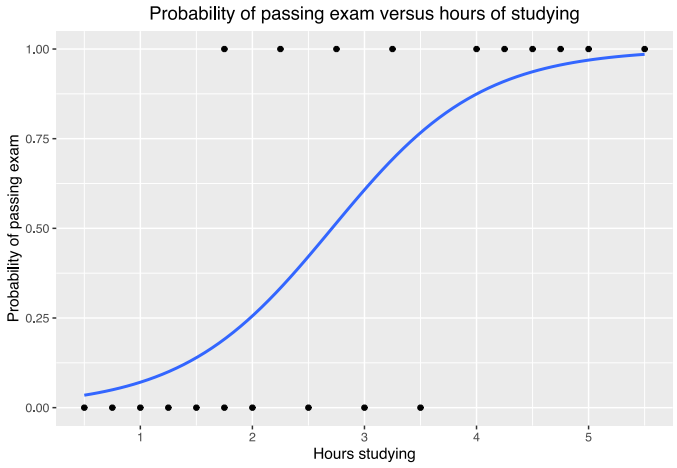

FIGURE IV.5 – Exemple de Régression logistique

Arbre de décision

Un arbre de décision est un modèle sous forme d'arbre, où chaque nœud représente une question ou un test sur une variable, et chaque branche mène à une prédiction. Il est très intuitif et facile à interpréter.

Random Forest

Le **Random Forest** est un algorithme d'apprentissage automatique qui repose sur le principe des arbres de décision. Son fonctionnement consiste à construire plusieurs arbres de décision de manière aléatoire, chacun étant entraîné sur un sous-ensemble différent des données disponibles. Chaque arbre effectue ensuite sa propre prédiction, et la prédiction finale est obtenue en combinant les résultats de l'ensemble des arbres, généralement en faisant la moyenne (pour les problèmes de régression) ou en utilisant un vote majoritaire (pour les problèmes de classification).
Le schéma ci-dessous illustre le fonctionnement de l'algorithme **Random Forest** appliqué à un problème de classification

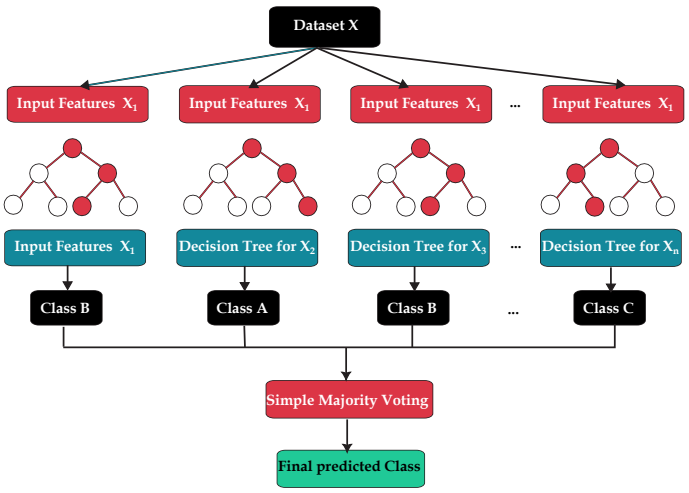

FIGURE IV.6 – Fonctionnement du Random Forest dans un problème de classification

K-Nearest Neighbors (KNN)

KNN classe une nouvelle donnée selon la majorité des "k" voisins les plus proches dans l'espace des données. C'est un algorithme simple mais parfois lent avec de grands datasets.

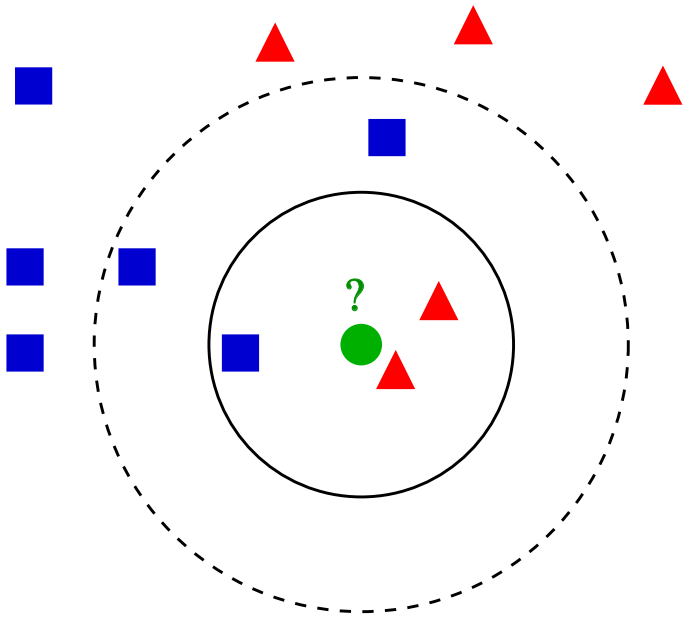

FIGURE IV.7 – Exemple de KNN

Support Vector Machine (SVM)

Le Support Vector Machine (SVM) est un algorithme d'apprentissage supervisé largement utilisé pour des tâches de classification et de régression. Son principe consiste à trouver une frontière optimale permettant de séparer les différentes classes d'observations tout en maximisant la distance (appelée marge) entre cette frontière et les points de données les plus proches. Ces points particuliers, situés au plus près de la frontière de séparation sont appelés vecteurs de support, car ils jouent un

rôle clé dans la construction du modèle. La Figure IV.8 illustre ce mécanisme avec un exemple visuel de frontière de décision et de marge maximale.

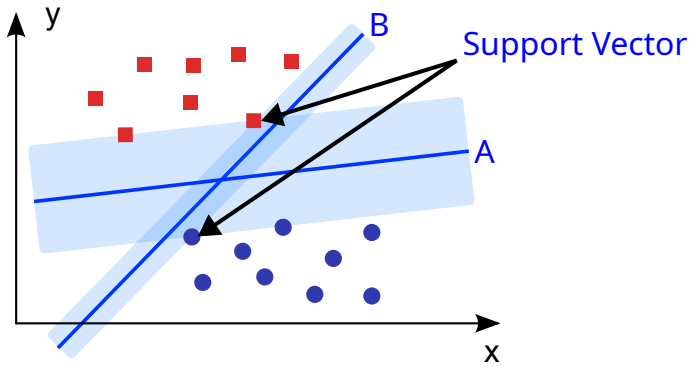

FIGURE IV.8 – Algorithme de Support Vecteur Machine

IV.3.2 Algorithmes d'apprentissage non supervisé

Ces algorithmes travaillent avec des données non étiquetées. Ils sont utilisés pour détecter des regroupements naturels ou des comportements inhabituels dans les données (anomalies).

1. **K-Means :** K-Means regroupe les données en **k clusters** selon leur similarité. Chaque donnée est affectée au centre de cluster le plus proche. Il est utilisé dans l'analyse exploratoire et la détection d'anomalies.
2. **Mean Shift :** Contrairement à K-Means, Mean Shift ne nécessite pas de spécifier le nombre de clusters. Il recherche automatiquement les zones de haute densité dans l'espace de données.

IV.4 Environnement Expérimental et Outils de Développement

Dans cette section, nous présenterons la topologie réseau utilisée, puis nous procéderons à l'installation des bibliothèques et des outils nécessaires à notre étude. 2.1 Présentation de la topologie réseau Dans le cadre de cette partie pratique, nous avons poursuivi notre travail en réutilisant la même topologie réseau que celle mise en place précédemment dans le chapitre 3.

Cette topologie est basée sur une architecture SDN (Software Defined Network) contrôlée par le contrôleur ***OpenDaylight***, version **Karaf 0.8.4**. Cette infrastructure nous a permis de simuler un environnement réseau flexible et adapté aux besoins de notre expérimentation, notamment pour la capture du trafic réseau et la détection d'attaques Ddos.

IV.4.1 Installation des bibliothèques Python

Dans le cadre de la mise en œuvre de notre système de détection d'attaques DdoS dans un réseau SDN, nous avons eu recours à plusieurs bibliothèques Python. Ces bibliothèques jouent un rôle essentiel dans les différentes étapes de notre travail, que ce soit pour la capture des paquets réseau, le

traitement des données ou encore la mise en place de l'algorithme de Machine Learning.
Voici les principales bibliothèques utilisées, avec pour chacune son rôle spécifique :

— **Scapy** : Bibliothèque de manipulation de paquets réseau permettant la capture, l'analyse et l'injection de trafic.

— **Pandas** : Bibliothèque d'analyse et manipulation de données. Offre des structures de données efficaces comme les DataFrames pour organiser et traiter les métriques réseau collectées.

— **Scikit-learn :** Bibliothèque de machine learning complète. Fournit des algorithmes comme le Random Forest, utilisés pour classifier le trafic et identifier les attaques DdoS.

— **Joblib :** Permet de sauvegarder et recharger rapidement des modèles de machine learning entraînés.

— **Matplotlib :** Librairie de visualisation de données. Génère des graphiques et diagrammes pour représenter visuellement les résultats d'analyse.

— **Termcolor :** Module simple pour l'affichage coloré dans le terminal. Ajoute des codes couleurs aux messages pour distinguer visuellement les différents types d'alertes.

L'installation de ces bibliothèques se fait à l'aide de la commande suivante :

```
sudo apt install python3-scapy python3-pandas python3-sklearn python3-joblib python
```

IV.4.2 Installation des outils de simulation d'attaque

Pour simuler des attaques réseau, nous utiliserons deux outils principaux :

— **Hping3 :** Permet de générer du trafic réseau personnalisé, particulièrement utile pour simuler des attaques DdoS.

— **Xterm :** Fournit une interface terminal externe pour exécuter plus facilement les commandes ping et hping3.

L'installation de ces outils se fait à l'aide de la commande suivante :

```
sudo apt install hping3 xterm
```

IV.5 Workflow ML : Capture de Trafic, Entraînement du Modèle et Détection en Temps Réel

Dans le but de détecter une attaque de déni de service (DdoS) dans un réseau SDN à l'aide du Machine Learning, il est nécessaire de suivre un processus bien défini.
Ce processus se compose principalement de trois étapes essentielles. Premièrement, la capture du trafic réseau. Ensuite, l'entraînement d'un modèle de Machine Learning. Et enfin, la détection des

attaques en temps réel.
Avant d'entamer ces différentes étapes, il est important de démarrer au préalable le contrôleur SDN ainsi que la topologie du réseau sous ***Mininet***.
Une fois cela fait, nous allons exécuter successivement trois scripts Python, chacun ayant un rôle bien précis dans ce processus de détection. Dans cette section, nous allons donc voire successivement :

1. La création du script de capture du trafic réseau.
2. L'entraînement du modèle de Machine Learning.
3. La détection des attaques DdoS en temps réel.

IV.5.1 Création du script de capture du trafic réseau

Objectif du script

La première étape de notre démarche consiste à créer un fichier Python nommé capture_traffic.py. L'objectif principal de ce script est de capturer le trafic réseau généré dans notre topologie SDN.
Ce trafic capturé doit contenir deux types de flux :

1. Un trafic légitime (trafic normal), généré par des commandes comme ping.
2. Un trafic malveillant (trafic anormal), représenté par des attaques de type DdoS, généré à l'aide de l'outil hping3.

Les données collectées à travers ce script seront ensuite enregistrées dans un fichier CSV. Ce dernier jouera un rôle essentiel puisqu'il sera utilisé par la suite comme jeu de données d'entraînement pour le modèle de Machine Learning.

Code source du script "capture_traffic.py"

```
from scapy.all import sniff, IP, TCP, UDP
import pandas as pd

# Liste pour stocker les caractéristiques des paquets
data = []

def packet_callback(packet):
if packet.haslayer(IP):
src_ip = packet[IP].src
dst_ip = packet[IP].dst
proto = packet[IP].proto
pkt_len = len(packet)
```

```
        # Vérification du type de protocole
        protocol = "TCP" if packet.haslayer(TCP) else "UDP" if packet.haslayer(UDP)

        # Ajout des caractéristiques extraites
        data.append([src_ip, dst_ip, proto, pkt_len, protocol])
        print(f"Paquet capturé : {src_ip} -> {dst_ip}, Protocole : {protocol}, Tai

        print("Démarrage de la capture des paquets...")
        sniff(iface="s1-eth1", prn=packet_callback, count=5000)

        # Sauvegarde des caractéristiques extraites dans un fichier CSV
        df = pd.DataFrame(data, columns=["IP source", "IP destination", "Protocole
        df["Type de protocole"] = df["Type de protocole"].map({"TCP": 1, "UDP": 2,
# Encodage du type de protocole
        df.to_csv("traffic_data.csv", index=False)

        print("Extraction des caractéristiques terminée. Données enregistrées dans
```

Explication du code

Le script nommé capture_traffic.py a pour objectif de capturer le trafic réseau circulant au niveau de l'interface s1-eth1, en extrayant les principales caractéristiques des paquets interceptés. Pour cela, il utilise principalement la bibliothèque Scapy pour la capture des paquets et Pandas pour le traitement et l'enregistrement des données. Lorsqu'un paquet est détecté, la fonction packet_callback () est automatiquement appelée afin de récupérer des informations essentielles comme l'adresse IP source, l'adresse IP destination, le protocole utilisé (TCP, UDP ou autre) ainsi que la taille du paquet, ces informations sont ensuite stockées dans une liste, puis organisées sous forme d'un tableau grâce à un DataFrame Pandas.
Enfin, l'ensemble des données collectées est sauvegardé dans un fichier CSV nommé traffic_data.csv

Exécution du script

Après avoir sauvegardé le script, nous l'exécutons à l'aide de la commande suivante :

```
sudo python3 capture_traffic.py
```

Durant l'exécution de ce script, nous avons généré deux types de trafic au sein de notre réseau SDN :

— Un trafic légitime à l'aide de la commande :

```
pingall
```

— Un trafic malveillant à l'aide de la commande :

```
sudo hping3 -su--flood --udp -p 80 192.168.10.10
```

La figure ci-dessus illustre l'exécution du script de capture du trafic réseau ainsi que l'enregistrement des données dans un fichier CSV.

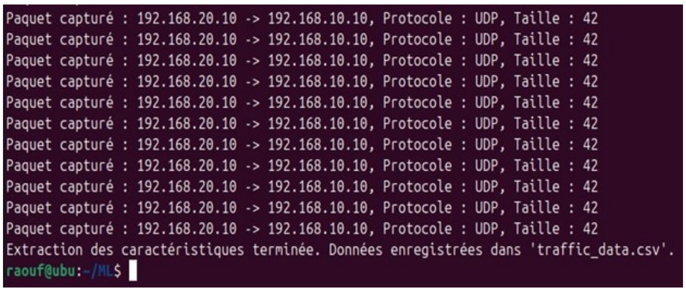

FIGURE IV.9 – Exécution du script de capture du trafic réseau

IV.5.2 Création du script pour l'entraînement du modèle de Machine Learning

Objectif du script

Dans cette partie, nous avons choisi d'utiliser l'algorithme **Random Forest** comme méthode d'apprentissage automatique, ce choix est motivé par ses bonnes performances en classification, particulièrement pour la détection d'attaques dans les réseaux SDN.

L'objectif est de développer un modèle de Machine Learning capable de différencier le trafic légitime du trafic malveillant. Le modèle est entraîné à partir des fichiers CSV obtenus lors de la capture de trafic, qui contiennent les caractéristiques réseaux nécessaires à son apprentissage.

Après l'entraînement, le modèle est enregistré sous le nom *trafic_modele.pkl*. Il pourra ensuite être utilisé pour identifier en temps réel si un nouveau trafic est légitime ou malveillant.

Code source du script modele_ddos.py

```python
import pandas as pd
from sklearn.model\_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib


# Charger le jeu de données
```

```
data = pd.read_csv("traffic_data.csv")

# Définir les caractéristiques et les étiquettes
X = data[["Protocole", "Taille du paquet"]]
y = data["Type de protocole"].apply(lambda x: 1 if x == 2 else 0)  # UDP (DoS) = 1,
    Normal = 0

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Entraîner le modèle
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Évaluer le modèle
y_pred = clf.predict(X_test)
print(f"Précision : {accuracy_score(y_test, y_pred):.2f}")
print("Rapport de classification :\n", classification_report(y_test, y_pred))

# Sauvegarder le modèle
joblib.dump(clf, "traffic_model.pkl")
print("Modèle sauvegardé sous 'traffic_model.pkl'.")
```

Explication du code

Ce script sert à créer un modèle d'apprentissage automatique avec l'algorithme Random Forest. D'abord, il importe les bibliothèques nécessaires et charge les données depuis le fichier "traffic_data.csv".
Ensuite, il sélectionne les caractéristiques "Protocole" et "Taille du paquet", et transforme les étiquettes : 1 pour le trafic malveillant et 0 pour le trafic légitime, les données sont ensuite divisées en deux parties : 70% pour l'entraînement du modèle et 30% pour les tests.
Le modèle Random Forest est ensuite entraîné avec 100 arbres de décision, une fois le modèle entraîné, il est évalué en calculant sa précision, et un rapport détaillé est généré avec des métriques comme la précision, le rappel et le F1-score.
Enfin, le modèle est sauvegardé sous le nom "traffic_model.pkl" pour pouvoir l'utiliser plus tard dans la détection en temps réel.

Exécution du script

Après avoir sauvegardé le script sous le nom modele.py, nous procédons à son exécution en utilisant la commande suivante :

```
sudo python3 modele\_ddos.py
```

Une fois le script exécuté avec succès, un fichier contenant le modèle entraîné est généré sous le nom « traffic_model.pkl ».
La figure ci-dessus illustre l'exécution du script d'entraînement du modèle.

Figure IV.10 : Exécution du script de l'entraînement du modèle de Machine Learning

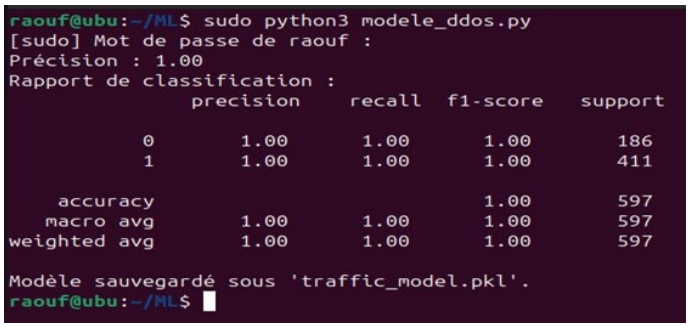

FIGURE IV.10 – Exécution du script de l'entraînement du modèle de Machine Learning

IV.5.3 Création du script pour la détection des attaques DdoS en temps réel

Objectif du script

Dans cette partie, nous avons réalisé un script permettant d'effectuer la détection des attaques en temps réel dans un réseau SDN.
Après avoir entraîné et sauvegardé notre modèle de Machine Learning sous le nom "trafic_modele.pkl", nous allons l'utiliser ici pour analyser le trafic réseau circulant dans notre topologie SDN.
L'objectif principal de ce script est de capturer les paquets réseau en temps réel et d'analyser chacun de ces paquets à l'aide du modèle déjà entraîné, afin de distinguer entre un trafic légitime et un trafic malveillant.
Ainsi, lorsqu'un trafic malveillant est détecté, une alerte sera automatiquement affichée à l'écran mentionnant les informations détaillées du paquet suspect (protocole, taille, source et destination). Sinon, le script indique que le trafic est légitime.

Code source du script detection_temps_reel.py

```
1 from scapy.all import sniff, IP
2 import joblib
3 import pandas as pd
4 from termcolor import colored
5
6 # Charger le modèle entraîné
```

```
clf = joblib.load("traffic_model.pkl")

def analyze_packet(packet):
    if packet.haslayer(IP):
        pkt_len = len(packet)
        proto = packet[IP].proto
        src_ip = packet[IP].src
        dst_ip = packet[IP].dst

        # Préparer les caractéristiques sous forme de DataFrame
        features = pd.DataFrame([[proto, pkt_len]], columns=["Protocole", "Taille du paquet"])

        # Prédire le type de trafic
        prediction = clf.predict(features)

        if prediction[0] == 1:
            # Alerte en cas d'attaque
            print(colored("\n[ALERTE] Détection d'une activité suspecte", "red", attrs=["bold"]))
            print(f"Type : Anomalie détectée")
            print(f"Protocole : {proto}")
            print(f"Taille du paquet : {pkt_len} octets")
            print(f"Source : {src_ip}")
            print(f"Cible : {dst_ip}")
            print("-" * 50)

        else:

            # Trafic normal
            print(colored(f"[INFO] Trafic normal | Protocole={proto} | Taille={pkt_len} octets | {src_ip} → {dst_ip}", "green"))

print(colored("\nDémarrage de l'analyse du trafic...", "blue", attrs=["bold"]))
sniff(iface="s1-eth1", prn=analyze_packet, count=5000)
```

Explication du code source

Tout d'abord, nous avons procédé à l'importation des bibliothèques nécessaires. Ensuite, nous avons chargé le modèle de classification nommé traffic_model.pkl qui a été précédemment généré lors de la phase d'entraînement. La fonction principale analyze_packet(packet) est responsable de l'analyse de chaque paquet capturé :

— Elle vérifie si le paquet contient une couche IP.

— Elle récupère certaines caractéristiques importantes :

  — Le protocole utilisé.

  — La taille du paquet.

  — L'adresse IP source et destination.

Ces caractéristiques sont ensuite structurées dans un DataFrame avant d'être envoyées au modèle pour prédire le type de trafic :

— Si la prédiction est 1, cela signifie qu'une activité suspecte (attaque) a été détectée. Le script affiche alors une alerte avec les détails du paquet.

— Sinon, si la prédiction est 0, le trafic est considéré comme normal et un message d'information est affiché.

Enfin, la fonction sniff () de Scapy permet de capturer les paquets en temps réel depuis l'interface réseau s1-eth1 en appelant à chaque fois la fonction "analyze_packet()" pour les analyser.

Exécution du script

Une fois le script sauvegardé sous le nom detection_temps_reel.py, son exécution se fait à l'aide de la commande suivante :

```
sudo python3 detection_temps_reel.py
```

Dès son exécution, le script procède automatiquement à l'analyse en temps réel du trafic circulant sur le réseau, comme l'illustre la figure ci-dessous.

FIGURE IV.11 – Lancement de l'analyse en temps réel

Dans un premier test, nous avons effectué un simple ping entre deux hôtes pour générer un trafic légitime.

```
ping 192.168.10.10
```

La figure ci-dessous montre l'affichage obtenu suite à ce test, indiquant que le trafic est légitime.

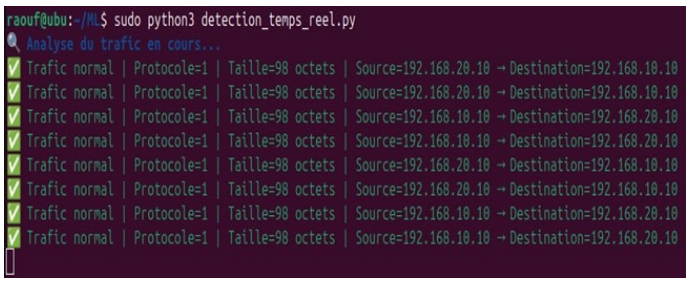

FIGURE IV.12 – Affichage du trafic légitime détecté par le script

Ensuite, un autre test a été réalisé en utilisant l'outil hping3 afin de générer un grand nombre de paquets et simuler un trafic malveillant.

```
sudo hping3 -su--flood --udp -p 80 192.168.10.10
```

La figure ci-dessous illustre l'affichage du script déclenchant une alerte suite à la détection d'un trafic malveillant.

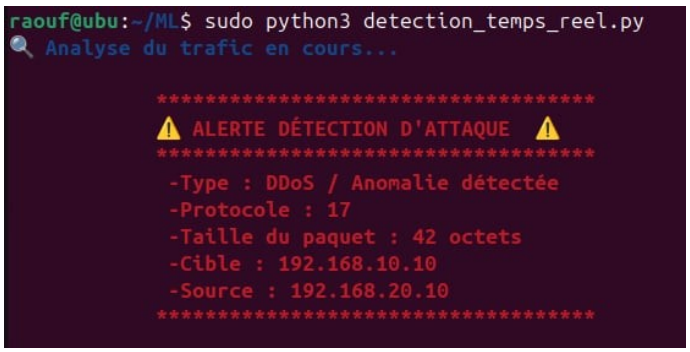

FIGURE IV.13 – Affichage d'une alerte suite à la détection d'un trafic malveillant

Métrique	**Valeur (%)**
Précision (Accuracy)	98.00
Rappel (Recall)	97.50
F1-score	97.80

TABLE IV.1 – Tableau IV.X – Résultats des métriques du modèle Random Forest

Les résultats obtenus par le modèle **Random Forest** sont très encourageants et montrent une performance élevée dans la détection des attaques DDoS. Voici une explication des principales métriques utilisées :

— **Précision (Accuracy)** : correspond au pourcentage de bonnes classifications (attaques et trafic normal) sur l'ensemble des données testées. Une précision de 98 % signifie que le modèle a correctement prédit 98 % des cas, toutes classes confondues.

— **Rappel (Recall)** : mesure la capacité du modèle à détecter les attaques réelles. Un rappel de 97,5 % indique que la grande majorité des attaques DDoS présentes dans les données ont été identifiées.

— **F1-score** : représente une moyenne harmonique entre la précision et le rappel. Il est particulièrement utile lorsqu'il y a un déséquilibre entre les classes (trafic normal vs attaque). Un F1-score de 97,8 % montre un bon compromis entre exactitude et couverture.

Ces métriques démontrent l'efficacité du modèle proposé pour classifier le trafic réseau dans un environnement SDN simulé. Grâce à ses performances, le modèle Random Forest permet de détecter les comportements anormaux avec fiabilité, tout en limitant les faux positifs. Cette approche par *apprentissage automatique* constitue ainsi une solution prometteuse pour renforcer la sécurité des réseaux SDN contre les attaques DDoS.

IV.6 Conclusion

Pour conclure, à travers ce chapitre, nous avons pu mettre en place une approche pratique et efficace basée sur le Machine Learning pour la détection des attaques DdoS dans les réseaux SDN. Grâce à cette méthode, il est possible d'identifier en temps réel les comportements anormaux et de générer des alertes pour réagir rapidement face aux tentatives d'attaques.
En somme, ce travail nous a permis non seulement de mieux comprendre les concepts du Machine Learning, mais également de les appliquer concrètement dans le domaine de la cybersécurité. Bien sûr, cette solution peut encore être améliorée et optimisée pour traiter des attaques plus complexes ou pour fonctionner sur des réseaux de plus grande taille.

CONCLUSION ET PERSPECTIVES

Pour en conclure, il convient de faire rappeler que ce mémoire nous a permis de mettre en évidence plusieurs aspects fondamentaux liés aux réseaux définis par logiciel (SDN) en mettant en lumière leurs avantages, leurs limites, et les enjeux de sécurité qui y sont liés.
En effet, l'approche SDN transforme profondément la manière dont un réseau est administré : en séparant la logique de décision du transfert de données, on gagne en flexibilité en automatisation, et en visibilité globale. Tout au long de ce travail, nous avons cherché à comprendre ce qui distingue réellement le SDN des architectures réseau traditionnelles. Nous avons étudié les vulnérabilités spécifiques à ce nouveau modèle, et les solutions proposées dans la littérature pour les contrer. Nous avons ensuite réalisé une comparaison concrète entre deux topologies l'une traditionnelle et l'autre basée sur le SDN afin de mieux assimiler les différences techniques et opérationnelles. Enfin, nous avons mis en œuvre une solution pratique basée sur le Machine Learning avec un algorithme de type Random Forest, pour détecter automatiquement les comportements suspects dans un réseau SDN simulé. Bien entendu, cette étude reste une première étape. Le système présenté peut encore être amélioré, notamment en testant d'autres algorithmes d'apprentissage, en enrichissant le jeu de données, ou en intégrant la réponse automatique aux attaques. Cette expérience a non seulement renforcé notre compréhension des réseaux SDN et des menaces actuelles, mais elle a aussi montré l'importance d'associer les technologies d'intelligence artificielle à l'administration réseau. Enfin, et dans un contexte où les infrastructures deviennent de plus en plus dynamiques et complexes, cette combinaison s'impose comme une piste propice pour construire des réseaux plus sécurisé et plus autonomes.

BIBLIOGRAPHIE

[1] Manageengine sdn tech topic. https://www.manageengine.com/network-monitoring/tech-topics/sdn.html. [Accès le 19 Fevrier 2025].

[2] Tauseef Ali, Chong Yung-Wey, and Selvakumar Manickam. Machine learning techniques to detect a ddos attack in sdn : A systematic review. *Applied Sciences*, 13(15) :1–4, 2023.

[3] Kevin Benton, L. Jean Camp, and Chris Small. Openflow vulnerability assessment. In *Proceedings of the ACM HotSDN (Hot Topics in Software Defined Networking)*, Hong Kong, China, 2013.

[4] Afef Bradai, Kushal Singh, Toufik Ahmed, and Tinku Rasheed. Cellular software defined networking : a framework. *IEEE Communications Magazine*, 53(6) :36–44, 2015.

[5] F. Camps. Intelligence artificielle. LAAS-CNRS (Laboratoire d'Analyse et d'Architecture des Systèmes), Toulouse, France, 2025.

[6] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane : Taking control of the enterprise. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007.

[7] Colin Dixon, David Olshefski, Vikram Jain, Casimer DeCusatis, Wes Felter, John Carter, Mohammad Banikazemi, Vijay Mann, John M. Tracey, and Renato Recio. Software defined networking to support the software defined environment. *IBM Journal of Research and Development*, 58(2/3) :3–4, 2014.

[8] L. Dridi. *Mitigation des attaques de déni de service dans les réseaux définis par logiciel*. PhD thesis, Montréal, 2017.

[9] L. Dridi and M. Zhani. Sdn-guard : Dos attacks mitigation in sdn networks. In *IEEE International Conference on Cloud Networking (CloudNet)*, 2016.

[10] Astrid Genevay. *Introduction au machine learning*. Dunod, Paris, 2020.

[11] Muhammad Iqbal, Farhat Iqbal, Farhan Mohsin, Muhammad Rizwan, and Farhan Ahmad. Security issues in software defined networking (sdn) : Risks, challenges and potential solutions. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 10(10) :301, 2019.

[12] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Openflow random host mutation : Transparent moving target defense using software defined networking. 2012.

[13] Ahmed Kammoun. *Gestion des réseaux virtuels dans un contexte SDN/NFV*. PhD thesis, Paris, 2019.

[14] P. Karthika and K. Arockiasamy. Simulation of sdn in mininet and detection of ddos attack using machine learning. *Bulletin of Electrical Engineering and Informatics*, 12(3) :1797–1805, June 2023.

[15] Hyojoon Kim, Nick Feamster, and Santosh Vempala. Sdn vs. traditional networking : a performance evaluation. 2013.

[16] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking : A comprehensive survey. *Proceedings of the IEEE*, 103(1) :50, 2015.

[17] M. A. Lopez and O. Duarte. *Providing Elasticity to Intrusion Detection Systems in Virtualized Software Defined Networks*. PhD thesis, Rio de Janeiro, 2016.

[18] Gabriel Lopez-Millan, Rafael Marin-Lopez, and Fernando Pereniguez-Garcia. Towards a standard sdn-based ipsec management framework. *Computer Standards & Interfaces*, 66 :54–62, 2019.

[19] Shikha Midha, Shalini Verma, Kavita, Mohit Mittal, Noor Zaman Jhanjhi, Mehedi Masud, and Mohammed A. AlZain. A secure multi-factor authentication protocol for healthcare services using cloud-based sdn. *Computers, Materials & Continua*, 74(1) :3712–3723, 2023.

[20] Open Networking Foundation (ONF). Openflow switch specification version 1.4.0 (wire protocol 0x05). http://www.opennetworking.org, October 2013. [Accès le 31 janvier 2025].

[21] Oracle. Oracle sdn. http://www.oracle.com/us/products/networking/virtual-networking/sdn/overview/index.html, 2016. [Accès le 2 Fevrier 2025].

[22] John Owens, II and Arjan Durresi. Video over software-defined networking (vsdn). In *16th International Conference on Network-Based Information Systems (NBiS)*, 2013.

[23] Alireza Ranjbar, Miika Komu, Pekka Salmela, and Tuomas Aura. An sdn-based approach to enhance the end-to-end security : Ssl/tls case study. 2016.

[24] F. Rida. Smart business : repenser le business de votre entreprise avec l'ia, le machine learning et le deep learning. https://blog.cellenza.com/data/smart-business-repenser-le-business-de-votre-entreprise-avec-lia-le-machine-le September 2022. [Accès le 1 mai 2025].

[25] M. Salim. *Gestion dynamique et évolutive de règles de sécurité pour l'Internet des Objets*. PhD thesis, Reims, 2019.

[26] Sandra Scott-Hayward, Gemma O'Callaghan, and Sakir Sezer. Sdn security : A survey. In *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, Italy, 2013.

[27] Yu Shen, Wu Chunming, Deyu Kong, and Qi Cheng. Flow table saturation attack against dynamic timeout mechanisms in sdn. *Applied Sciences*, 13(12) :14–19, 2023.

[28] Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Gu Guofei. Avantguard : Scalable and vigilant switch flow management in software-defined networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2013.

[29] Daniel Smyth, Victor Cionca, Sean McSweeney, and Donna O'Shea. Exploiting pitfalls in software-defined networking implementation. In *IEEE International Workshop on Secure and Dependable Internet of Things (SDIoTS)*, 2016.

[30] Carlos Vicentini, Altair Santin, Eduardo Viegas, and Vilmar Abreu. Sdn-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming. *Journal of Network and Computer Applications*, pages 2–4, 2018.

[31] Haopei Wang, Xu Lei, and Gu Guofei. Floodguard : A dos attack prevention extension in software-defined networks. In *45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2015.

[32] Tianyi Xing, Dijiang Huang, Lei Xu, Chin-Jen Chung, and Pankaj Khatkar. Snortflow : An openflow-based intrusion prevention system in a cloud environment. In *Second GENI Research and Educational Experiment Workshop (GREE)*, 2013.

RÉSUMÉ

LES réseaux classiques, aussi appelés réseaux à routage réparti (RR), sont utilisés depuis longtemps dans les systèmes informatiques. Même s'ils ont prouvé leur efficacité, ils présentent aujourd'hui certaines limites, notamment en matière de sécurité, de gestion et d'adaptabilité. Pour répondre à ces défis, une nouvelle architecture a vu le jour : les Réseaux Définis par Logiciel (SDN), qui offrent plus de souplesse grâce à un contrôle centralisé. Mais cette centralisation peut aussi créer de nouvelles failles. Dans ce mémoire, nous avons d'abord abordé les concepts fondamentaux des SDN, puis analysé leurs vulnérabilités ainsi que les principales méthodes pour les sécuriser. Nous avons ensuite comparé un réseau classique à une topologie SDN simulée, avant de mettre en œuvre une solution de détection d'attaques DdoS basée sur le Machine Learning, afin de renforcer la sécurité de ce type de réseau.

Mots clés :Réseaux classiques, Réseaux définis par logiciel, SDN, OpenFlow, sécurité réseaux,DDoS, Détection d'attaques DDoS,Routage réparti, Machine Learning,Topologie réseau , Contrôle centralisée

ABSTRACT

TRADITIONAL networks, also known as distributed routing networks (RR), have been used for many years in IT systems. Although they have proven to be reliable, they now show limitations in terms of security, management, and adaptability. To address these challenges a new architecture has emerged : Software Defined Networking (SDN), which provides more flexibility through centralized control. However, this centralization can also introduce new vulnerabilities.
In this thesis, we first introduced the fundamental concepts of SDN, then discussed their security weaknesses and the main strategies used to address them. We carried out a comparison between a traditional network and a simulated SDN topology, and finally implemented a DdoS attack detection solution based on Machine Learning to enhance the security of SDN environments.

Keywords :Traditional networks, Software Defined Networking, SDN,OpenFlow, Network Security, DDoS Attack Detection,Distributed routing, Machine Learning ,Network topology, Centralized control.