



RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE



Ministère de l'Enseignement Supérieur et de la Recherche

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE

Faculté d'Electronique et d'Informatique

Département Informatique

MÉMOIRE DE MASTER

Spécialité

RÉSEAUX ET SYSTÈMES DISTRIBUES

Thème

Détection d'attaques dans le réseau SDN
par une approche ensembliste de
machine learning

Proposé et Dirigé par :

Prof. BOUGHACI Dalila

Réalisé et présenté par :

Mr. KISSOUM Mohand Ouali

Devant le jury composé de :

Mme Président

Mr Membre

Soutenu le : /09/2021

Binôme N° : RSD016/2021

REMERCIEMENTS

Avant tout nous tenons à remercier **ALLAH** tout puissant qui a illuminé nos cœurs et nous avoir donné le courage et la force d'établir ce mémoire dans les meilleures conditions.

Nous tenons à exprimer toute notre profonde gratitude à notre promotrice, Professeur **D. BOUGHACI**, pour sa patience et ses précieux conseils qui n'ont d'égale que sa gentillesse et sa bonne humeur.

Nous remercions aussi les membres du jury, d'avoir pris le temps de lire notre mémoire.

Nous remercions tout particulièrement nos familles qui ont toujours su nous soutenir, nous conseiller et sans lesquelles ce travail n'aurait pas pu être possible, et avec émotion nous leur dévoilons le fruit de nos efforts en espérant être à la hauteur de leur fierté inconditionnelle.

Un grand merci également à nos amis pour leurs soutiens, leurs sincères amitiés, leurs confiances et leurs présences à tout moment.

Nos derniers remerciements, s'adressent aux enseignants qui nous ont offert le privilège d'avoir suivi leurs enseignements durant notre formation universitaire.

Racim MALOUM

&

Mohand Ouali KISSOUM

Table des matières

Table des figures	IV
Liste des tableaux	V
Introduction générale	1
I Software Defined networking SDN	3
I.1 Introduction	4
I.2 Le SDN	4
I.2.1 Définition	4
I.2.2 Architecture	4
I.2.3 Open Flow et Switch OpenFlow	6
I.2.4 Table des flux	7
I.2.5 Fonctionnement Openflow	8
I.2.6 Différence et avantage entre un réseau SDN et un réseaux traditionnel	9
I.2.7 La sécurité dans SDN	10
I.3 Conclusion	10
II Détection d'intrusion dans les réseaux SDN	11
II.1 Introduction	12
II.2 Définition d'intrusion	12
II.3 Catégorie d'intrusion :	12
II.3.1 Attaque par déni de service (DOS)	12
II.3.2 Attaque de l'utilisateur à la racine (U2R)	12
II.3.3 Attaque de type "Remote to Local" (R2L)	12
II.3.4 Attaque par sondage	12
II.4 La sécurité dans SDN	13
II.4.1 Analyse des menaces de sécurité dans SDN	13
II.4.2 Exemple d'attaques DOS/DDOS	15
II.4.3 Attaque DOS/DDOS spécifiques au réseau SDN	16
II.5 Mécanismes de sécurité informatique	17
II.5.1 Parefeu	17
II.5.2 Antivirus	17
II.5.3 Systèmes de détection d'intrusions IDS	17
II.5.4 Système de prévention d'intrusion IPS	17
II.5.5 Cryptographie	17
II.5.6 VPN	17
II.6 Mécanismes de Détection des attaques DOS/DDOS dans un réseau SDN : .	18
II.6.1 Entropie	18
II.6.2 Apprentissage automatique	18
II.6.3 Analyse des modèles de trafic	18
II.6.4 Taux de connexion	18

II.6.5	Intégration de SNORT et d'OpenFlow	18
II.7	Conclusion	19
III	Machine learning	20
III.1	Introduction	21
III.2	Apprentissage automatique :	21
III.2.1	Définition :	21
III.2.2	Types d'apprentissages :	21
III.3	Process de machine Learning	22
III.4	Techniques ensemblistes d'apprentissage automatique	23
III.5	Les modèles de classification	25
III.5.1	Arbre de Décision	25
III.5.2	Random Forest	25
III.5.3	AdaBoost	26
III.6	Conclusion	27
IV	Conception de la solution	28
IV.1	Introduction	29
IV.2	Problématique	29
IV.3	DOS lent dans http	30
IV.4	Synthèse sur les travaux existants	31
IV.5	Motivation de travail	32
IV.6	Solution Proposée	33
IV.7	Architecture de la solution	33
IV.8	Construction d'un modèle de Machine Learning	34
IV.8.1	Dataset	35
IV.8.2	Phase d'entraînement	36
IV.8.3	Évaluation du modèle	39
IV.9	Fonctionnement de notre système dans le cadre SDN	40
IV.10	Pseudo code de l'implémentation de la solution	41
IV.11	Conclusion	43
V	Implémentation de la solution	44
V.1	Introduction	45
V.2	Environnement de travail	45
V.2.1	Contrôleur Ryu	45
V.2.2	Mininet	45
V.3	Langage et librairies utilisés	45
V.4	Processus d'implémentation de la solution	46
V.4.1	Implémentation des deux modules collection de paquets et extraction d'attributs	46
V.4.2	Implémentation du module de classification de flux	47
V.4.3	Implémentation du module d'atténuation d'attaque	50
V.5	Simulation	51
V.6	Conclusion	59
	Conclusion générale	60
	Bibliographie	I

Table des figures

Figure I.1	: Architecture SDN	5
Figure I.2	: Commutateur OpenFlow	6
Figure I.3	: Structure de la table de flux	7
Figure I.4	: Processus de transmission d'un paquet avec Openflow	8
Figure III.1	: Processus machine learning	22
Figure III.2	: Principe de fonctionnement de Bagging	23
Figure III.3	: Principe de fonctionnement de Boosting	24
Figure III.4	: Principe de fonctionnement de Stacking	24
Figure III.5	: Principe de fonctionnement de RF	26
Figure IV.1	: Illustration du trafic normal, du DOS lent et du DOS volumétrique	29
Figure IV.2	: Architecture générale de la solution	34
Figure IV.3	: Processus Machine Learning	35
Figure IV.4	: Processus d'entraînement et d'évaluation du modèle	39
Figure IV.5	: Pseudo code de la détection d'attaque	42
Figure IV.6	: Pseudo code de l'atténuation d'attaque	43
Figure V.1	: Matrices de confusions des algorithmes non-ensemblistes	48
Figure V.2	: Matrices de confusions des algorithmes ensembliste	49
Figure V.3	: Résultat d'analyse des performances de tous les algorithmes	50
Figure V.4	: Architecture de la simulation	52
Figure V.5	: Lancement du contrôleur Ryu	53
Figure V.6	: Lancement de la topologie	53
Figure V.7	: Lancement des serveurs w1 et W2	54
Figure V.8	: Lancement de la détection de trafic	54
Figure V.9	: Lancement de la prévention d'attaque	55
Figure V.10	: Simulation d'un trafic légitime	55
Figure V.11	: Détection de trafic légitime	56
Figure V.12	: Simulation de trafic d'attaque	56
Figure V.13	: Détection de trafic d'attaque	57
Figure V.14	: Atténuation de trafic d'attaque	57
Figure V.15	: Activation de trafic venant de la source d'attaque	57
Figure V.16	: Statistiques des paquets TCP malveillants avant la détection	58
Figure V.17	: Statistiques des paquets TCP malveillants après la détection	59

Liste des tableaux

Table I.1 : Comparaison entre SDN et un réseau traditionnel	9
Table II.1 : Vecteur d'attaque dans SDN	13
Table IV.1 : Descriptions des attributs sélectionnées	38
Table IV.2 : Matrice de confusion pour un problème de classification à 2 classes .	40
Table V.1 : Performances des algorithmes non ensemblistes	47
Table V.2 : Performances des algorithmes ensemblistes	49
Table V.3 : Rôle des entités de la topologie	52

Liste des abréviations

API	Application Programming Interface
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CIC	The Canadian Institute for Cybersecurity
CLI	Command Line Interface
CPU	Central Processing Unit
CSE	Communications Security Establishment
DARPA	Defense Advanced Research Projects Agency
DDOS	Distributed Denial of Service
DOS	Denial of Service
DT	Decision Tree
FN	Faux Négative
FP	Faux Positive
HTTP	Hyper-Text Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IP	Internet Protocol
ISO	International Organization for Standardization
KDD	Knowledge Discovery and Data Mining
KNN	K-Nearest Neighbors
LBNL	Lawrence Berkeley National Laboratory
LLDP	(Link Layer Discovery Protocol
MLP	Multi-layer Perceptron
ML	Machine Learning

NBI NorthBound Interface

NB Naive Bayes

ONF Open Networking Fondation

OSI Open Systems Interconnection

OSPF Open Shortest Path First

R2L Remote to Local

RF Random Forest

SBI SouthBound Interface

SDN Software-defined networking

SGDC Stochastic Gradient Descent Classifier

SQL Structured Query Language

SSL Secure Socket Layer

STP Spanning Tree Protocol

SVM Support Vector Machines

TCAM Ternary Content-Addressable Memory

TCP Transport Control Protocol

TLS Transport Layer Security

U2R User to Root

UDP User Datagram Protocol

UNB University of New Brunswick's

VM Virtual Machine

VN Vrai Négative

VPN Virtual Private Network

VP Vrai Positive

XML Extensible Markup Language

Introduction générale

Durant ces dernières années, la prolifération des appareils mobiles, et l'apparition de nouvelles technologies, telles que internet des objets, informatique en nuage, etc... fait constat d'augmentation d'un nombre de dispositifs internet à un rythme sans précédent. Ces évolutions ont entraîné une croissance significative de la complexité des réseaux à grande échelle, et nécessitent généralement une mise à jour des infrastructures concernées. Toutefois, la mise à jour des réseaux traditionnels est réalisée en configurant manuellement chaque équipement qui peut se traduire par des erreurs et incohérences. De plus, ces configurations prennent beaucoup de temps qui ne permet pas de faire face à l'expansion rapide du monde des technologies d'information. Par conséquent les technologies et infrastructures de réseau traditionnel n'offrent pas de solutions évolutives et faciles à gérer pour des réseaux aussi vastes et complexes.

C'est dans ce contexte qu'apparut le paradigme des réseaux définis par logiciels (SDN), approuvé pour relever les défis inhérents d'un réseau complexe et étendu. Cela est possible grâce à l'une des caractéristiques les plus marquantes de SDN qui est la division du plan de données et du plan de contrôle. En effet, l'intelligence et le contrôle du réseau sont logiquement centralisés et déployés sur une entité logique appelée contrôleur SDN, ce dernier communique avec l'infrastructure sous-jacente abstraite telle que des routers et commutateurs via des interfaces ouvertes, permettant ainsi de programmer dynamiquement des applications et de les déployer dans le contrôleur pour déterminer des politiques de gestion et de sécurité d'un réseau. Cette technologie a permis ainsi d'offrir une architecture prometteuse pour les réseaux futures.

Bien que SDN offre une flexibilité et une gestion efficace à faible coût, néanmoins cette technologie souffre de nombreux problèmes de sécurité et elle reste vulnérable à de nombreux types d'attaques, notamment, les attaques de type DOS. Ce type d'attaque a pour objectif d'empêcher les utilisateurs légitimes d'accéder au service client pendant une longue période, ciblant particulièrement le contrôleur dû à la nature centralisée de l'architecture, mais aussi les équipements réseau, comme cela était dans les réseaux traditionnels. Ces attaques destructrices visent à utiliser essentiellement la force brute et génèrent un volume massif de trafic en tort d'une part et d'autre part l'attaque DOS lente, un autre type d'attaque DOS vise la couche applicative est également très dangereuse et capable de provoquer des dégâts importants, Cette attaque DOS lente génère un trafic à un taux très faible épuisant les ressources de la victime dans le respect des normes du protocole. Donc, c'est essentiel de bien sécuriser l'architecture SDN, en commençant d'abord par la détection de ces attaques à travers un IDS, par la suite, penser à établir des mécanismes pour atténuer ou stopper ces menaces.

Nous dénombrons à l'heure actuelle environ une centaine de systèmes de détections d'intrusions (ou IDS), ces derniers jouent un rôle crucial dans la sécurité des systèmes informatiques. Pourtant, il reste des désavantages dans les IDS courants, en effet le déploiement

croissant et multiple de ces IDS dans des environnements, de plus en plus complexes, a mis en évidence plusieurs problèmes qui sont actuellement fortement posés, entre autres, en produisant souvent un gros volume de fausses alertes dues à des processus d'attaques automatisés et fréquents. Cela rend l'analyse embarrassante et compliquée. Il est donc intéressant de développer de nouvelles techniques de détection d'attaque.

Ces dernières années on a constaté l'utilisation d'un nouveau paradigme qui représente la tendance actuelle dans le monde informatique pour détecter les intrusions. Ce concept s'agit bien de l'apprentissage automatique, dont le principe est de définir en premier lieu le comportement normal d'un réseau informatique avec un modèle formel qui permet, lors de la phase de détection, d'identifier toute activité réseau comme étant anormale si la déviation par rapport au modèle dépasse un certain seuil. Par ailleurs les méthodes ensemblistes d'apprentissage automatique qui reposent sur la combinaison de multiples modèles, permettent d'accroître les performances de détection d'intrusion et d'obtenir des résultats meilleurs que si on prend chacun des modèles séparément.

Dans ce projet, nous proposons un système de détection agrégé par un module d'atténuation d'attaque en utilisant une approche ensembliste de machine learning dans le réseau SDN, et ceci en se concentrant sur les attaques DOS plus particulièrement DOS lente. Pour cela notre solution se traduit par trois parties, la première a pour objectif de détecter les attaques en se basant sur les informations des paquets échangés dans le réseau. Dans la deuxième partie, un modèle de machine learning est construit à l'aide de deux dataset CICIDS2017 et CSE-CIC-IDS2018. Random forest est l'algorithme de classification favorisé après une évaluation et comparaison des performances avec d'autres algorithmes qui existent dans la littérature. La troisième partie consiste à exploiter la couche application du contrôleur SDN pour pouvoir atténuer ces attaques.

Ce mémoire est divisé en cinq chapitres organisés comme suit :

- ▷ Le premier chapitre sera consacré à la description générale des réseaux SDN et la présentation de son principe de fonctionnement.
- ▷ Dans le deuxième chapitre nous présenterons et analyserons les intrusions dans le réseau SDN et donnerons un aperçu sur les différents mécanismes de détection d'attaques.
- ▷ Le troisième chapitre sera consacré aux explications et définition des concepts liés à l'apprentissage automatique.
- ▷ Le quatrième chapitre sera dédié à la conception de la solution où on va expliquer le fonctionnement de notre solution.
- ▷ Le cinquième et dernier chapitre sera destiné à l'implémentation, réalisation et la validation de la solution en simulant notre solution sous mininet.

Ce mémoire s'achèvera par une conclusion, suivie par les perspectives de recherche induites par notre travail

Chapitre I

Software Defined networking SDN

I.1 Introduction

Durant ces dernières années, dans le but d'être gérées efficacement, les réseaux informatiques traditionnels ont connu de grands défis pour faire face aux progressions de l'informatique en nuage (cloud), et à l'évolution des réseaux en termes de flexibilisation et de virtualisation. C'est dans ce contexte qu'est apparu le paradigme SDN.

L'idée de réseau défini par le logiciel connu sous SDN est de séparer le plan de contrôle du plan des données, libérant ainsi des cycles d'innovation de logiciels pour devenir indépendant des cycles d'innovation matérielles. Cela permet de rendre le réseau programmable, c'est-à-dire de permettre aux applications d'interagir directement avec les réseaux [1].

Dans ce chapitre, nous présenterons les informations nécessaires pour comprendre la technologie et l'architecture du SDN ainsi que le protocole OpenFlow.

I.2 Le SDN

I.2.1 Définition

SDN est une architecture concentrée sur la séparation des fonctions de contrôle et de transfert des données du réseau. L'abstraction de ces deux fonctions permet de créer le contrôleur SDN, un point central de haute performances qui gère le plan de contrôle, cela assure aux administrateurs réseaux une visibilité totale de tout le réseau et permet d'initialiser, contrôler, d'ajuster et gérer les ressources du réseau d'une manière dynamique[2].

I.2.2 Architecture

Comme représentée sur la figure I.1, l'architecture d'un réseau SDN est divisée en trois couches à savoir la couche infrastructure, la couche de contrôle et la couche application. La communication entre ces différentes couches est faite à travers les Southbound, Northbound et East-Westbound APIs. Dans cette partie, nous allons détailler, étape par étape, ces différentes couches et APIs pour mieux expliquer l'architecture de SDN.

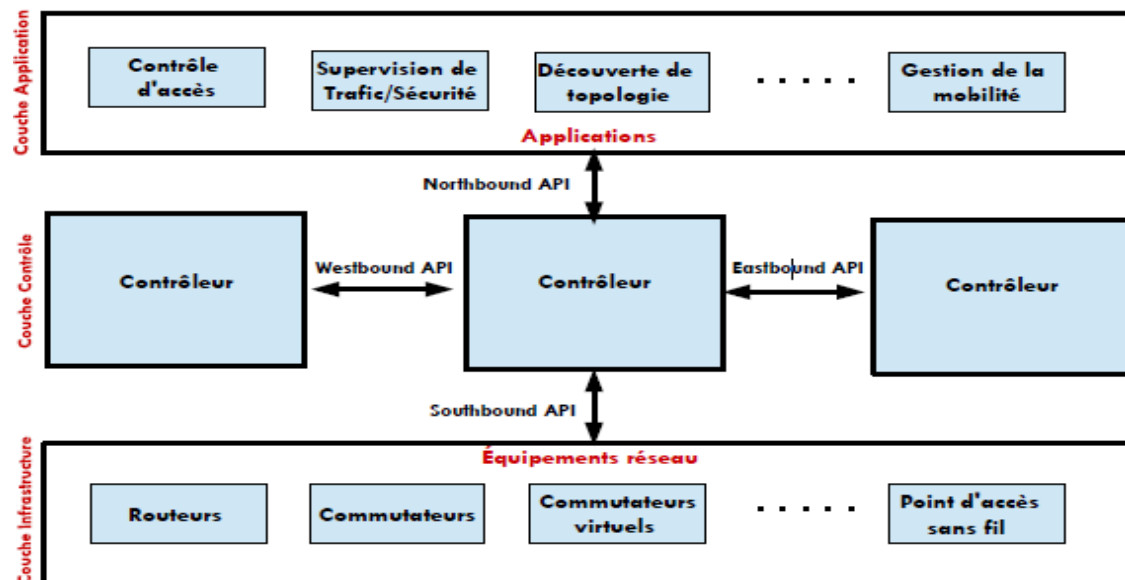


FIGURE I.1 – Architecture SDN [3]

I.2.2.1 Plan de données

C'est la couche la plus basse, constituée des équipements réseaux tels que commutateurs, routeurs..., responsable uniquement de transfert de données de la source vers la destination, en se basant sur des informations contenues dans des tables de flux et reçues du plan de contrôle [4].

I.2.2.2 Plan de contrôle

La couche de contrôle est le cœur de l'architecture SDN, elle est responsable de la configuration, la gestion et la maintenance du réseau. Comme son nom l'indique, cette couche, contrôle le plan de données en établissant les règles qu'elle devra suivre. Parmi les protocoles qui participent à ce plan, on peut citer par exemple OSPF, STP, ARP, ou BGP [5].

Cette couche utilise une entité logique appelée contrôleur SDN qui est le cerveau du réseau. Ce contrôleur est chargé de découvrir et maintenir la topologie, d'informer les périphériques du plan de transmission sur la façon de gérer les paquets dans le réseau, basculer la charge des chemins et aussi communiquer avec les services et les applications offertes par le réseau.

I.2.2.3 Plan applicatif

La couche la plus haute de l'architecture présente le plan applicatif de l'architecture SDN. Elle apporte les automatisations des applications à travers le réseau fournit par le contrôleur, qui se trouve au niveau du plan de contrôle en ayant recours à la NBI associée [6]. Cela permet très rapidement aux gestionnaires de réseau de configurer, de gérer, de sécuriser et d'optimiser les ressources du réseau via des programmes dynamiques automatisés SDN. Ces applications SDN pourraient inclure la surveillance de réseau, la découverte de la topologie, la réservation des chemins, les statistiques et l'analyse de réseau, la sécurité de réseau, le contrôle d'accès et la virtualisation de fonction de réseau (VNF), etc.

I.2.2.4 Interface vers le sud

Communément appelée SBI, sa fonction principale est de permettre la communication entre le contrôleur SDN et les nœuds du réseau (routeurs et commutateurs). Ces interfaces facilitent le contrôle du réseau et permettent au contrôleur SDN de connaître la topologie du réseau existante et d'effectuer des changements dynamiques dans le plan de données en temps réel. Open Flow est le standard le plus utilisé pour cette interface [5].

I.2.2.5 Interface vers le Nord

Communément appelée NBI, décrit la zone de communication couverte par le protocole entre le contrôleur et les applications. Il n'existe aucun standard entre la couche de contrôle et celle d'application. Ce qui signifie qu'il peut y avoir plusieurs interfaces NBI pour servir tous les différents cas d'utilisation, contrairement à la SBI[5].

I.2.2.6 Interface Est/Ouest

Les interfaces côté Est/Ouest sont nécessaire au réseau qui est géré par plusieurs contrôleurs. Ces interfaces sont utilisées pour l'échange d'informations entre les contrôleurs pour synchroniser les états du réseau. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible[5].

I.2.3 Open Flow et Switch OpenFlow

La séparation de plan de donnée, du plan de contrôle nécessite un protocole de communication entre ces deux plans. L'ONF a publié le protocole OpenFlow pour transmettre au commutateur des instructions qui permettent de programmer le plan de données et d'obtenir des informations de ces commutateurs afin que le contrôleur puisse disposer d'une vue globale, logique du réseau physique. Cette vue est utilisée pour toutes les décisions que doit prendre le plan de contrôle (routage, filtrage de trafic, ...etc)[7]. Un commutateur OpenFlow est un programme logiciel ou un périphérique matériel qui transfère les paquets dans un environnement SDN. La Figure I.2 montre un commutateur OpenFlow et ses trois parties essentielles :

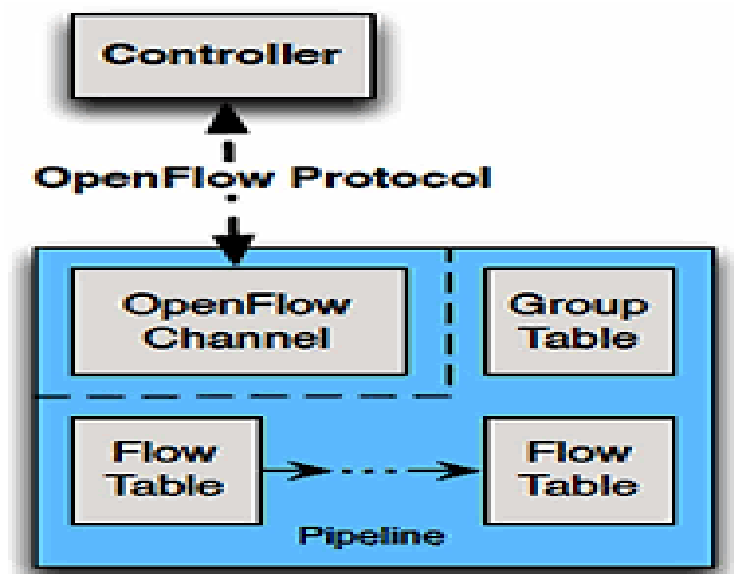


FIGURE I.2 – Commutateur OpenFlow [8]

Table de flux : est utilisée pour faire correspondre les paquets entrants aux critères de sélection des flux dans la table des flux pour appliquer un certain nombre d'actions. Un flux entrant peut passer par une ou plusieurs tables de flux (pipeline)[9].

Table de groupe : est utilisé pour envisager une autre façon de traiter les paquets entrants en leur appliquant un ensemble d'actions[9].

Canal sécurisé : est l'interface qui connecte chaque commutateur OpenFlow à un contrôleur. Grâce à cette interface, le contrôleur peut recevoir des messages par le commutateur et de pouvoir le gérer à travers le réseau. Pour assurer le bon déroulement des communications entre le commutateur et le contrôleur, le canal est sécurisé par le protocole TLS ou SSL[9].

Protocole OpenFlow : permet au contrôleur de gérer les commutateurs. En utilisant ce protocole, le contrôleur peut ajouter, effacer et mettre à jour les entrées dans les tables de Flux[9].

I.2.4 Table des flux

Un commutateur OpenFlow doit contenir une ou plusieurs tables de flux, chaque table de flux contiennent plusieurs entrées qui correspondent à des règles, où chacune est constituée principalement des trois champs illustrés dans la figure ci-dessous :

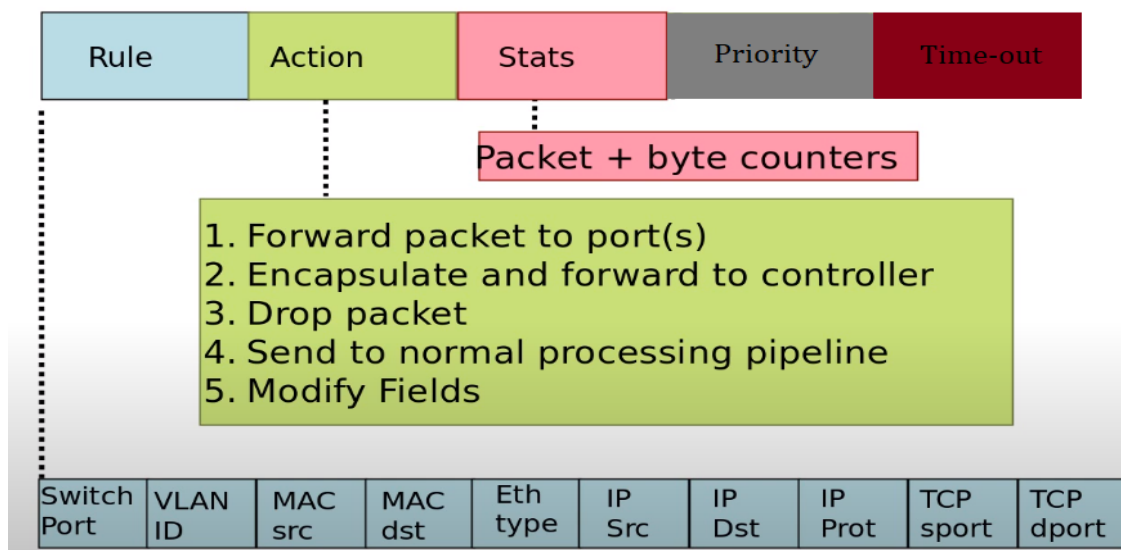


FIGURE I.3 – Structure de la table de flux [10]

- **Match (Champ de correspondance) :** il définit le flux de données, il contient les informations nécessaires pour déterminer le paquet auquel une règle sera appliquée. L'identification peut se faire à base d'une multitude d'informations contenues dans l'entête du paquet, allant de la couche 1 à la couche 4 du modèle ISO dépendamment de la spécification d'OpenFlow déployé[9].
- **Action (Instructions) :** spécifie comment les paquets d'un flux seront traités. Une action peut être l'une des suivantes : Transférer le paquet vers un ou plusieurs ports, supprimer le paquet, transférer le paquet vers le contrôleur, ou modifier le champ d'entête de paquet[9].

- **Counter (Compteurs)** : il est réservé à la collecte des statistiques de flux. Ils enregistrent le nombre de paquets de chaque flux, et le temps écoulé depuis le dernier transfert de flux[9].
- **Priority(priority)** : indique le niveau de priorité de l'application de la règle. Plus la priorité est haute, plus elle sera utilisée pour traiter des paquets[9].
- **Time Out** : indique le temps d'attente maximal entre deux traitements de flux avant que la règle de ces deux flux soit effacée[9].

I.2.5 Fonctionnement Openflow

Dans un réseau SDN basé sur le protocole OpenFlow. Lorsqu'un paquet arrive à un commutateur, le commutateur vérifie s'il y'a une entrée dans la table de flux qui correspond à l'en-tête de paquet. Si c'est le cas, le commutateur exécute l'action correspondante dans la table de flux. Dans le cas contraire, c'est-à-dire s'il n'y a pas une entrée correspondante, le commutateur génère un message *Packet_in* et l'envoie vers le contrôleur demandant une règle de transmission, puis le contrôleur décide selon sa configuration une action pour ce paquet, et envoie une nouvelle règle de transmission sous la forme d'un *Packet_out* et *Flow-mod* au commutateur, et enfin, la table de flux du commutateur est actualisée, pour prendre en compte la nouvelle règle installée par le contrôleur[11]. la figure I.4 résume le fonctionnement de OpenFlow.

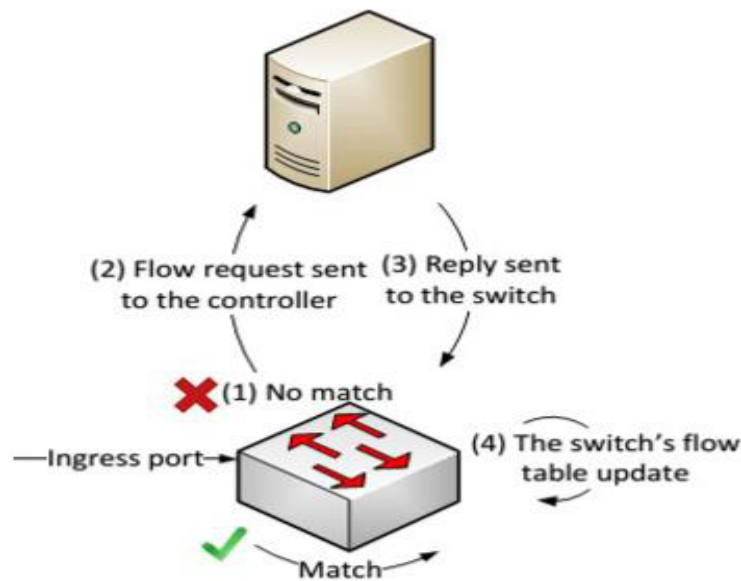


FIGURE I.4 – Processus de transmission d'un paquet avec Openflow [11]

Chaque règle de flux a deux valeurs de temporisation timeout associées qui contrôlent la suppression des règles de la table de commutation, le idle timeout et le hard timeout. Le idle timeout est une valeur de temporisation d'inactivité qui indique quand l'entrée doit être supprimée de la table de commutation en raison d'un manque d'activité de cette règle de commutation pendant cette période. Il se déclenche lorsque le flux reste inactif, alors que le hard timeout est une valeur qui indique quand l'entrée doit être supprimée, quelle que soit l'activité de la règle de commutation. En recevant la nouvelle règle de commutation, le commutateur l'installe dans sa table de commutation et l'utilise jusqu'à l'expiration de l'un de ces temporisateurs. Dans ce cas, le commutateur supprime l'entrée de la règle correspondante

de sa table de commutation et envoie un message *OFPT_FLOW_REMOVED* au contrôleur pour l'informer que la règle a été supprimé[11].

I.2.6 Différence et avantage entre un réseau SDN et un réseaux traditionnel

Bien que le point décisif dans la différence entre un réseau basé sur l'approche SDN et un réseau traditionnel est la séparation du matériel et du logiciel et l'intégration du contrôleur. Le SDN marque son existence par l'intégration des avantages suivants[12] :

- **Agile** : L'abstraction de la topologie réseau permet aux administrateurs d'ajuster dynamiquement le trafic réseau.
- **Management centralisé** : L'intelligence du réseau est centralisée dans un contrôleur SDN, qui maintient une vue globale du réseau.
- **Basé sur des standards ouverts et vendeurs-indépendants** : L'implémentation à travers des standards ouverts permet de simplifier l'architecture réseau car les instructions sont fournies par un ou plusieurs contrôleurs au lieu de multiple équipements propriétaires.
- **Réduction des dépenses opérationnelles** : Le matériel réseau est simplifié par la suppression de la fonction de contrôle et la mettre dans un contrôleur ce qui réduit les coûts d'exploitation.
- **Amélioration de l'efficacité du réseau** : Le contrôle et la gestion centralisés augmentent l'automatisation.
- **La vue globale du réseau par le contrôleur** : Permet de remplacer les protocoles de routage distribués (OSPF, EIGRP, ...) par des mécanismes plus simples.

Le tableau suivant résume donc les principales différences entre le SDN et le réseau traditionnel

Réseaux définis par logiciel (SDN)	Réseau traditionnel
Autorité de contrôle centralisée	Instances de contrôle spécifiques à l'appareil
Séparation claire entre le matériel et le niveau de contrôle	Le contrôle du matériel est intégré dans le matériel
Plan de commande librement programmable	Plan de commande spécifique à l'appareil
Protocoles standardisés (par ex. via OpenFlow)	Protocoles spécifiques au fabricant
Accès logiciel à la couche de données possible	L'accès à la couche de données doit se faire directement sur le matériel
Architecture flexible et facilement évolutive	Architecture statique et difficile à adapter

TABLE I.1 – Comparaison entre SDN et un réseau traditionnel

I.2.7 La sécurité dans SDN

En général, les menaces de sécurité sur les architectures de type SDN sont similaires aux réseaux traditionnels. Cependant, le profil de ces menaces change avec le SDN, par exemple un contrôleur SDN compromis pourrait permettre à un hacker de prendre le contrôle de tout un réseau.

Le SDN est confronté à ces nouveaux challenges de sécurité, notamment sur la question de comment sécuriser l'architecture SDN elle-même. C'est pourquoi il est nécessaire d'intégrer la sécurité et la fiabilité dans la conception même du réseau SDN avant toute utilisation. Étant basée sur une architecture en trois couches et des interfaces de programmation, la sécurité de SDN doit être assurée à tous ces niveaux, ce qui constitue plusieurs défis à relever et qui devront s'allonger avec le déploiement progressif de cette technologie[13].

I.3 Conclusion

Ce chapitre a servi à la présentation de quelques notions qui permettent de comprendre les réseaux SDN. Nous avons fourni une base théorique sur SDN, en présentant son architecture, ainsi le protocole OpenFlow. Il était question de montrer que cette approche a permis de rendre les réseaux programmables, évolutifs et faciles à innover. Dans le chapitre suivant, on va présenter les menaces de sécurité ainsi que les techniques de détection d'intrusion dans les réseaux SDN.

Chapitre II

Détection d'intrusion dans les réseaux SDN

II.1 Introduction

À l'heure actuelle, les infrastructures réseaux notamment ceux basé SDN subissent des attaques qui peuvent entraîner des pertes conséquentes. Les outils de sécurité tels que les IDS et les IPS sont indispensables pour lutter contre les différents types d'intrusions, mais ces derniers restent toujours impuissant faces à ces intrusions qui ne cesses d'augmenter et être plus dévastatrices.

Ce chapitre sera consacré pour parler d'une manière générale des intrusions dans le réseau SDN, l'analyse des attaques DOS et nous terminerons enfin par présenté les différents mécanismes de détection d'attaques DOS dans le réseau SDN.

II.2 Définition d'intrusion

Une intrusion est l'opération qui consiste à accéder aux données d'un système informatique, sans autorisation et en contournant les dispositifs de sécurité mis en place afin d'extraire des informations sensibles.

II.3 Catégorie d'intrusion :

Dans le domaine informatique il existe plusieurs types d'attaques qu'on peut répertorier selon quatre catégories[14] :

II.3.1 Attaque par déni de service (DOS)

Une attaque DOS est une tentative de rendre des ressources ou services indisponibles pour les utilisateurs en créant un trafic important dans les ressources de calcul ou de mémoire des machines victimes, les rendant saturer et donc incapable de traiter les demandes par les utilisateurs légitimes du système, ou en exploitant Une faille d'un système cible afin de le rendre inutilisable.

II.3.2 Attaque de l'utilisateur à la racine (U2R)

Il s'agit d'une catégorie d'attaque dans laquelle l'attaquant exploite les vulnérabilités du système pour obtenir des privilèges de super-utilisateur ; L'intrus essaie d'accéder aux ressources du réseau en tant qu'utilisateur normal, obtenu par exemple en reniflant des mot de passe ou par ingénierie sociale, et tente par la suite d'obtenir les privilèges d'un super utilisateur.

II.3.3 Attaque de type "Remote to Local" (R2L)

Cette classe d'attaque se produit lorsqu'un attaquant envoie des paquets au réseau cible avec une intention de s'intéresser à leurs vulnérabilités pour obtenir un accès local aux ressources qui existent sur ce réseau.

II.3.4 Attaque par sondage

Le sondage est une classe d'attaque qui consiste à scanner un réseau ou une machine afin de recueillir des informations permettant d'identifier les vulnérabilités potentielles, telle que le balayage de ports afin de connaître l'architecture du réseau (OS, topologie du réseau, protections déployées, ...).

II.4 La sécurité dans SDN

II.4.1 Analyse des menaces de sécurité dans SDN

La sécurité est l'un des facteurs freinant le déploiement des architectures SDN qui sont principalement centralisées. L'aspect centralisé du plan de contrôle présente, en effet, beaucoup d'avantages par rapport aux réseaux traditionnels, mais elle induit au SDN de nouveaux risques de sécurité qui lui sont propres, dont SDN ne peut être adopté si ces défis de sécurité ne sont pas relevés. Ces défis doivent donc être disposés pour apporter des mesures de sécurité appropriées.

Différents vecteurs de menaces ont déjà été identifiés dans SDN. Cependant, seules les menaces ciblant la communication entre le plan de contrôle et le plan de données sont spécifiques au SDN ; les autres vecteurs de menaces n'affectent que les réseaux traditionnels. Comme le montre le tableau II.1, il existe au moins sept vecteurs de menace identifiés dans l'architecture SDN. Le premier indique la possibilité que des attaquants génèrent un faux trafic pour submerger les dispositifs du plan de données et le contrôleur. Le second permet à un attaquant d'exploiter les vulnérabilités des dispositifs de transfert pour perturber le réseau. La troisième est spécifique au SDN et exploite l'interface vers le sud pour écouter et surveiller la communication. La quatrième est importante et spécifique au SDN car elle exploite les vulnérabilités des contrôleurs pour prendre le contrôle du contrôleur ce qui entraîne la compromission de l'ensemble du réseau. La cinquième est spécifique au SDN et concerne les applications malveillantes développées et déployées sur les contrôleurs SDN. La sixième est liée aux attaques et aux vulnérabilités des stations d'administrations. Enfin la dernière menace est le manque de ressources de confiance pour l'analyse légale et la remédiation dans les réseaux SDN, ce qui empêche la récupération rapide pour remettre le réseau en état de fonctionnement [15].

Vecteur d'attaque	Spécifique au SDN ?	Conséquences sur SDN
Vecteur1	Non	Porte ouverte aux attaques DDOS
Vecteur2	Non	Augmentation potentielle d'attaque
Vecteur3	Oui	L'exploitation du contrôleur
Vecteur4	Oui	Un contrôleur compromis peut compromettre tout le réseau
Vecteur5	Oui	Développement et déploiement des applications malveillante sur le contrôleur
Vecteur6	Non	Augmentation potentielle d'attaque
Vecteur7	Non	Impact négatif sur la récupération rapide et le diagnostic des défauts

TABLE II.1 – Vecteur d'attaque dans SDN[12]

Selon l'analyse de la sécurité par plusieurs auteurs [15],[16],[17],[18] dans la littérature nous allons discuter en détails les attaques sur les différentes couches de l'architecture SDN :

- **Attaques sur le plan de données :** Une machine hôte malveillante peut attaquer n'importe quel commutateur ou contrôleur de SDN en générant des paquets réseaux falsifiés. Autrement dit, un attaquant peut injecter un volume important de données en faisant passer une machine malveillante pour une machine autorisée du réseau, pour lancer une attaque de déni de service DOS. De même, pour chaque nouveau paquet

falsifié les commutateurs génèrent le message *Packet_In* au contrôleur, ce qui peut entraîner une baisse des performances de ce dernier.

Un commutateur SDN peut être attaqué par un hacker en manipulant son comportement pour perturber les échanges dans le réseau. Cela peut se traduire par l'introduction d'un faux trafic ou règle de flux sur le commutateur dont les conséquences sont de détournés et d'abandonné le trafic légitime ce qui va interrompre la communication entre les dispositifs SDN ou la génération de nombreux messages à destination du contrôleur SDN, pouvant provoquer un déni de service.

- **Attaques sur le plan contrôle :** L'attaquant s'intéresse davantage au plan de contrôle en raison des rôles principales qu'a le contrôleur dans le réseau SDN. Il existe différents types d'attaques qui peuvent être réalisées sur le contrôleur.

Les applications malveillantes développées et déployées sur les contrôleurs SDN sont l'un des vecteurs d'attaques dans le plan de contrôle. Par exemple, le gestionnaire de topologie stocke les informations relatives aux périphériques tels que les commutateurs et les hôtes du réseau et utilise le protocole LLDP pour découvrir les liens interconnectés entre les commutateurs. Un attaquant peut, par exemple, exploiter la vulnérabilité de découverte de liens en générant de faux liens entre les commutateurs.

Les contrôleurs peuvent également être distribués et échangent des informations de temps en temps pour mettre à jour leurs états. Le problème se pose lorsque l'un des contrôleurs agit de manière malveillante et partage des informations erronées avec les autres contrôleurs afin de provoquer un dysfonctionnement du réseau.

L'une des vulnérabilités bien connues de cette couche est la compromission du contrôleur par une attaque par déni de service DOS, qui peut avoir un effet catastrophique sur l'ensemble du réseau, car ce genre de menaces sont très dangereuses et peuvent rendre une ressource réseau indisponible aux utilisateurs légitimes.

- **Attaques sur le plan Applications :** Le plan d'application comprend différents types d'applications qui sont généralement utilisées par les attaquants pour prendre la main sur le contrôleur, et d'affecter l'ensemble du réseau. Il existe plusieurs attaques possibles dans le plan d'application ,par exemple, un accès non autorisé à ces applications peut aider les attaquants à contourner le niveau de sécurité du contrôleur, et accéder à diverses parties du réseau, et prendre le contrôle du réseau en introduisant ses propres règles sur le contrôleur.
- **Attaques sur l'interface vers le sud :** Le plan de données devient hors service si un commutateur ne reçoit pas les règles de transmission des flux du contrôleur. Par conséquent, le lien entre le commutateur et le contrôleur doit être protégé car il peut potentiellement devenir une cible privilégiée des hackers pour attaquer le réseau. L'absence de chiffrement des données et la faible authentification à ce niveau permet de prendre le contrôle de cette interface. Par la suite, l'attaquant peut exploiter cette vulnérabilité pour obtenir un accès non autorisé aux données, par exemple la principale attaque sur cette interface est l'accès malveillant aux règles de flux et aussi d'autres attaques de type *Man-in-The-Middle* et l'écoute clandestine. De plus, un attaquant peut générer de faux paquets vers les commutateurs pour les forcer à générer un grand

nombre de messages *Packet_In* afin de surcharger le canal utilisé entre les commutateurs et le contrôleur.

- **Attaques sur l'interface vers le nord** : Contrairement à l'interface sud, l'interface nord n'est pas normalisée, donc cela la rend vulnérable à des menaces. En effet l'attaquant peut utiliser cette interface pour interférer avec la communication entre l'application et le contrôleur pour obtenir un accès non autorisé, par conséquent une attaque par usurpation d'identité supprime certaines informations, ce qui peut entraîner une falsification des résultats de l'application. Aussi, l'attaquant peut utiliser une application malveillante pour informer le contrôleur qu'il doit déconnecter d'autres applications, de même, l'application malveillante peut envoyer de nombreuses requêtes afin d'occuper la bande passante disponible sur l'interface nord.
- **Attaques sur l'interface Est/Ouest** : Les interfaces est et ouest sont également sujettes à diverses attaques profitant de la communication non cryptée des données entre les contrôleurs pour partager les informations de mise jour du réseau.

Les attaques DOS sont les menaces les plus populaires et inévitables pour les réseaux SDN comme ils l'ont toujours été pour les réseaux traditionnels. Cependant dans notre travail, nous allons nous concentrer sur les attaques DOS, en raison des graves répercussions sur les performances du réseau que peuvent avoir ces attaques notamment sur le plan de contrôle, le plan de données et la liaison entre le contrôleur et les commutateurs.

II.4.2 Exemple d'attaques DOS/DDOS

Au fil des ans, les cybercriminels ont mis au point un certain nombre de techniques pour s'attaquer aux cibles en ligne par le biais des attaques DOS/DDOS. Les experts ont réparti ces différentes techniques en trois grandes catégories : les attaques volumétriques, qui utilisent un trafic élevé pour inonder la bande passante du réseau. les attaques protocolaires, qui exploitent des faiblesses des couches 3 et 4 de la pile protocolaire pour rendre la cible inaccessible. Enfin les attaques applicatives, visant la couche application du modèle OSI en ciblant les applications WEB, ces attaques sont considérées comme le type d'attaques le plus sophistiqué et le plus grave. Voici quelques exemples concernant les attaques DOS/DDOS avec leurs principes de fonctionnements.

- **HTTP Flood** : est un type d'attaque basé application qui a pour objectif de submerger un serveur WEB ciblé avec des requêtes HTTP. Une fois que la cible a été saturée de demandes et qu'elle est incapable de répondre au trafic normal, un déni de service se produira pour les demandes supplémentaires des utilisateurs légitimes[19].
- **Ping Flood** : pour ce type d'attaque, les hackers envoient des paquets ICMP. Sachant que la machine cible répond par un nombre égal de demande d'écho ICMP, alors le hacker tente de la submerger avec ces paquets ICMP [19].
- **SYN Flood** : cette attaque est utilisée en exploitant la faiblesse du protocole TCP. L'attaquant envoie un grand nombre de demande SYN au serveur avec une adresse source frauduleuse et inexistante. Quant au serveur il répond à la demande en envoyant le paquet SYN-ACK (accusé de réception) et attend la réception du paquet ACK du client. Puisque le serveur attend dans ce cas un ACK inexistant et qu'il a un grand nombre de connexions semi-ouvertes sans message ACK, alors la file d'attente de cette connexion sera pleine rendant le serveur indisponible pour le trafic légitime [19].

- **UDP Flood** : Cette attaque consiste à envoyer un grand nombre de données utilisant le protocole UDP sans demande particulière à une cible. Dans le but de neutraliser la capacité de traitement et de réponse de ce dispositif[20].
- **Attaque Smurf** : la technique de l'attaque Smurf ou attaque par réflexion en français, consiste à l'envoi d'une multitude de paquets ICMP à des serveurs de diffusion avec l'adresse source de la machine cible, et de faire planter cette dernière suite à la réception d'une multitude de réponses à des requêtes qu'il n'a pas posées[20].
- **Attaque Land** : cette attaque consiste à démarrer une ouverture de session TCP via un SYN à destination d'un port ouvert de la machine cible. L'astuce de l'attaque est de préciser l'adresse IP source identique à l'IP de destination ainsi que le port source identique au port de destination. Le serveur répond alors à la requête en envoyant à lui-même une réponse sous la forme d'un paquet SYN/ACK. Ceci peut donc être interprété comme une nouvelle demande de connexion qui doit ainsi à nouveau être répondu avec un paquet SYN /ACK. La conséquence est donc une surcharge de capacité car le système tente de répondre constamment aux requêtes, ce qui finalement paralyse le système[20].

II.4.3 Attaque DOS/DDOS spécifiques au réseau SDN

La surcharge des contrôleurs et commutateurs, la congestion de la bande passante entre les commutateurs et le contrôle sont les menaces DOS/DDOS les plus préoccupantes pour le SDN. On discute ces principales menaces [21] :

- **Surcharge des commutateurs et débordement des tables de flux** : Les attaques DOS/DDOS génèrent une grande quantité de paquets malveillants qui sont inondés dans un commutateur. Pour ces paquets malveillants, le commutateur ne trouvera pas de règle correspondante dans la table des flux, du coup Il doit mettre en mémoire tampon les flux de données jusqu'à l'obtention d'une règle de flux du contrôleur SDN. Cependant, la taille de la TCAM d'un commutateur étant limitée, il n'est pas toujours possible de traiter tous les paquets entrants. De même un commutateur dispose d'un nombre limité de tables de flux dans lesquelles les règles de flux sont installées et utiliser jusqu'à l'expiration des temporisateurs, le idle timeout et hard timeout. Ce qui fait un commutateur OpenFlow est vulnérable aux attaques par saturation.
- **Saturation des ressources du contrôleur** : le contrôleur est le cœur du réseau SDN qui contrôle et gère l'ensemble des fonctionnalités du réseau. Par conséquent, un contrôleur compromis entrave les performances globales du réseau. Les ressources du contrôleur, telles que l'unité centrale et la mémoire, seront épuisées par le traitement des demandes inondées des attaques DOS/DDOS menant à des cas où le contrôleur devient totalement paralysé ainsi incapable de prendre une décision du routage. Cela réduit les performances de l'ensemble du réseau, car les demandes légitimes ne sont pas traitées en temps voulu.
- **Congestion de la bande passante entre le commutateur et le contrôleur** : Due à la communication agressive entre le contrôleur SDN et les commutateurs demandant des décisions de routage ; l'or d'arrivé de paquet au commutateur, si il ne trouve pas de règle correspondante dans la table des flux, alors il enregistre les informations nécessaires pour la demande de règle approprié contenant un ID et des informations partielles de l'en-tête du paquet. Mais lorsque ce tampon est plein, le paquet complet est attribué au contrôleur, ce qui conduit à la congestion du canal OpenFlow et par

conséquent, de nombreux paquets entrent en collision avec l'interface sud. Cela cause la perte de plusieurs messages *paquet-in* ainsi que le retard dans le temps de réponse des messages échangés entre le contrôleur et les commutateurs.

II.5 Mécanismes de sécurité informatique

Aujourd'hui, la sécurité est un enjeu majeur pour les entreprises ainsi que pour l'ensemble des acteurs qui l'entourent. Nous trouvons plusieurs mécanismes de sécurité :

II.5.1 Parefeu

Le parefeu ou Firewall en anglais, est un mécanisme indispensable dans la sécurité informatique des entreprises. Le Pare-feu propose donc un véritable contrôle sur le trafic réseau de l'entreprise. Il permet d'analyser, de sécuriser et de gérer le trafic réseau, en fixant un ensemble de règles et autorisations pour limiter les accès et les activités inutiles dans le réseau [22].

II.5.2 Antivirus

Un antivirus est un programme ayant pour objectif de détecter et supprimer les virus et les logiciels malveillants présents sur un ordinateur. La détection s'appuie sur des techniques tels que : technique basée sur la signature et technique basée sur la Méthode heuristique.

II.5.3 Systèmes de détection d'intrusions IDS

Un système de détection d'intrusion IDS est un mécanisme destiné à repérer des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte). Il permet ainsi d'avoir une connaissance sur les tentatives d'intrusions réussies ou échouées [23].

II.5.4 Système de prévention d'intrusion IPS

Un système de prévention d'intrusion IPS est un outil similaire aux IDS, sauf que ce système peut prendre des mesures afin de diminuer les risques d'impact d'une attaque. C'est un IDS actif, il détecte les intrusions et bloque les ports automatiquement [23] .

II.5.5 Cryptographie

La cryptographie est une science basée sur les mathématiques et la cryptologie s'attachant à protéger des messages (assurant la confidentialité, l'authenticité et l'intégrité) avec un algorithme de chiffrement qui consiste à rendre un message inintelligible, sauf pour celui qui possède le moyen (une clé) de le déchiffrer[24]. Parmi les principales fonctions on distingue le hachage, la Signature électronique

II.5.6 VPN

Un Réseau Privé Virtuel est une connexion inter-réseau permettant de relier 2 réseaux locaux différents de façon sécurisée par un protocole de tunnelisation. La tunnelisation est un protocole permettant aux données passant d'une extrémité à l'autre du VPN d'être sécurisées par des algorithmes de cryptographie[25].

II.6 Mécanismes de Détection des attaques DOS/DDOS dans un réseau SDN :

Dans cette section nous exposerons quelques techniques qui sont employées pour détecter les attaques DOS/DDOS au sein d'un réseau SDN.

II.6.1 Entropie

Technique basée sur la théorie de l'information, comme entropie de Shannon, utilisée pour mesurer le caractère aléatoire d'un attribut dans une période de temps donnée en se basant sur plusieurs caractéristiques telles que les flux réseau, les adresses IP et le nombre de paquets. En raison du succès de cette approche dans les réseaux traditionnels pour la détection des attaques DOS/DDOS, elle sera largement employée dans les réseaux SDN [26].

II.6.2 Apprentissage automatique

Les méthodes basées sur l'apprentissage automatique utilisent des techniques pour détecter les intrusions dans un environnement réseau, basées sur des modèles construits statistiquement et mathématiquement. Généralement elles sont appliquées efficacement dans la détection des attaques DOS/DDOS en discriminant des flux réseaux moyennant certaines caractéristiques liées au trafic réseau, ensuite en les classifiant comme malveillants ou non grâce à un modèle de classification déjà construit [27].

II.6.3 Analyse des modèles de trafic

Ces techniques reposent sur l'hypothèse que les hôtes infectés présentent des schémas comportementaux similaires et différents de ceux des hôtes non infectés. Par conséquent, elles analysent le trafic relatif aux métriques des réseaux de modèles d'attaque afin d'identifier l'attaquant ou la cible attaquée. Cette technique est très utilisée pour la détection des attaques botnets [28].

II.6.4 Taux de connexion

Les méthodes de détection des anomalies basées sur le taux de connexion peuvent être classées en deux types :

- Taux de réussite des connexions : ces techniques reposent sur le fait que la probabilité qu'une tentative de connexion réussisse doit être beaucoup plus élevée pour un hôte légitime contrairement à un hôte malveillant. Dès que le rapport de probabilité pour un hôte dépasse un certain seuil, il est déclaré comme infecté [29].
- Nombre de connexions établies : ces méthodes reposent sur le fait d'utiliser un seuil pour limiter le nombre de nouvelles tentatives de connexion dans une certaine période de temps, en raison qu'un attaquant peut tenter de connecter de nombreuses machines différentes dans un court laps de temps [30].

II.6.5 Intégration de SNORT et d'OpenFlow

Snort est le premier système de prévention des intrusions (IPS) Open Source au monde, il utilise une série de règles qui aident à définir l'activité réseau malveillante et génère des alertes pour les utilisateurs en cas de détection d'intrusion. Cependant, la large acceptation

de SNORT dans la communauté des réseaux et la croissance du SDN ont conduit au développement de systèmes qui mettent en œuvre la combinaison d'OpenFlow et de SNORT pour permettre aux systèmes de détecter les intrusions et de déployer des contremesures en cas d'intrusion dans un réseaux SDN [31].

Dans notre étude, nous allons utiliser les techniques d'apprentissage automatique pour détecter les attaques DOS, dans ce qui suit nous allons aborder en détail les méthodes d'apprentissage automatique utilisées dans notre approche proposée.

II.7 Conclusion

Ce chapitre est le résultat d'une recherche théorique sur la sécurité dans le réseau SDN. Ainsi nous avons parlé d'une façon globale sur les attaques informatiques et plus particulièrement sur les attaques dans SDN tels que les attaques DOS/DDOS, au final nous avons clôturé ce chapitre par quelques mécanismes de détection et protection contre les attaques informatiques. Dans le chapitre suivant en va présenter la technique d'apprentissage automatique qui représente actuellement la tendance dans la détection des attaques.

Chapitre III

Machine learning

III.1 Introduction

Les techniques d'apprentissage automatique ont montré leurs efficacités et leurs implications dans plusieurs domaines, notamment dans le domaine de la détection d'intrusion. Ce chapitre se concentre sur la présentation des différents concepts liés à l'apprentissage automatique en particulier les méthodes ensemblistes, et l'énumération de quelques algorithmes supervisés avec leurs principes de fonctionnements.

III.2 Apprentissage automatique :

III.2.1 Définition :

L'apprentissage automatique (ML en anglais) est une catégorie d'algorithme qui permet aux applications logicielles de prédire plus précisément les résultats sans être explicitement programmées. Le principe de base de l'apprentissage automatique est de créer des algorithmes capables de recevoir des données d'entrée et d'utiliser une analyse statistique pour prédire une sortie tout en les mettant à jour à mesure que de nouvelles données deviennent disponibles.

L'apprentissage automatique peut également être défini comme le processus de résolution d'un problème pratique par la collecte d'un jeu de données (dataset) et la construction d'un modèle statistique basé sur ce jeu de données [32].

III.2.2 Types d'apprentissages :

On distingue différents types d'algorithmes d'apprentissage automatique. Généralement, ils peuvent être répartis en trois catégories : supervisés, non supervisés et semi-supervisé.

III.2.2.1 Apprentissage automatique supervisé

L'apprentissage supervisé est une catégorie de ML dans laquelle les algorithmes apprennent des variables d'entrée (X), qui servent de superviseur ou d'enseignant, afin de prédire les variables de sortie (Y) [33].

Dans ce type d'apprentissage, les données utilisées pour l'entraînement sont déjà étiquetées. Par conséquent, le modèle de ML sait déjà ce qu'elle doit chercher (motif, élément...) dans ces données. À la fin de l'apprentissage, le modèle ainsi entraîné sera capable de retrouver les mêmes éléments sur des données non étiquetées.

Parmi les algorithmes supervisés, on distingue les algorithmes de classification (prédictions non-numériques) et les algorithmes de régression (prédictions numérique). En fonction du problème à résoudre, on utilisera l'un de ces deux archétypes.

III.2.2.2 Apprentissage automatique non supervisé

Ce type d'apprentissage, consiste à entraîner le modèle sur des données sans étiquettes. La machine apprend sur la base de ses propres informations, sans avoir besoin d'être guidé pour découvrir des modèles, des similarités et des différences. Cette approche est couramment utilisée dans certains domaines, comme la cybersécurité [34].

Parmi les modèles non-supervisés, on distingue les algorithmes de clustering (pour trouver des groupes d'objets similaires), d'association (pour trouver des liens entre des objets) et de réduction dimensionnelle (pour choisir ou extraire des caractéristiques).

III.2.2.3 Apprentissage automatique semi-supervisé :

L'apprentissage semi-supervisé repose sur la combinaison d'un ensemble de données étiquetées pour une grande quantité de données, et d'un ensemble de données non étiquetées pour une petite quantité de données pendant la phase de formation [35]. Ce type de ML se situe quelque part entre l'apprentissage supervisé, qui implique un ensemble de données étiquetées, et l'apprentissage non supervisé, qui implique un ensemble de données non étiquetées [36]. Parmi les exemples pratiques d'apprentissage semi-supervisé, citons l'analyse de la parole, la classification du contenu d'Internet et la classification des séquences de protéines.

III.3 Process de machine Learning

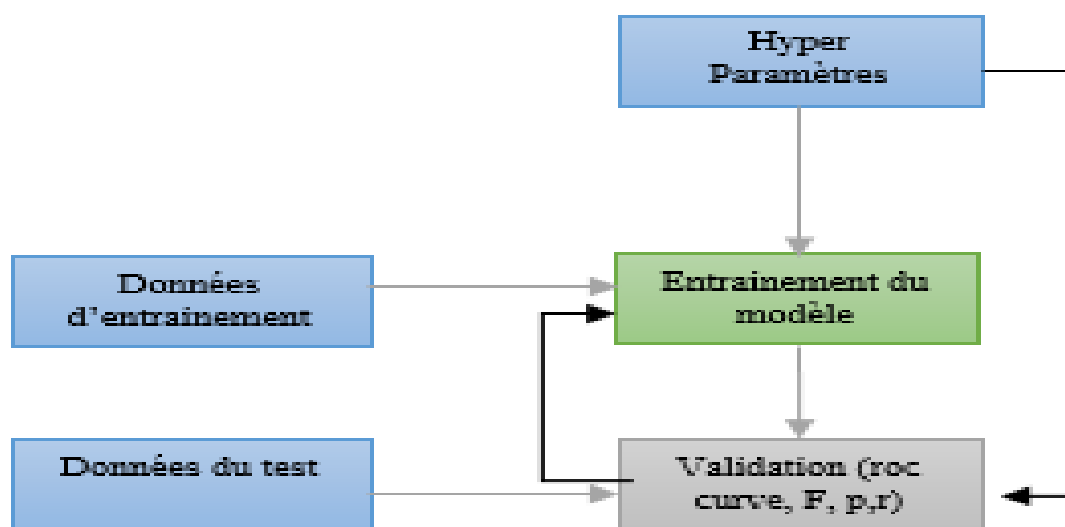


FIGURE III.1 – Processus machine learning [37]

Afin de mieux expliquer le processus de ML, nous commençons par la définition de quelques concepts de base :

- **Dataset** : c'est un ensemble de données qu'on fournit à une machine sous forme d'un couple d'exemples (x, y) dans l'apprentissage supervisé, où x représente les questions (variables ou attribut) et y les réponses au problème (résultat) que la machine doit résoudre. Dans l'apprentissage non supervisé, le dataset contient que les variables x .
- **Modèle** : un modèle de ML est le résultat généré lors de l'entraînement de l'algorithme d'apprentissage automatique avec des données. Après la formation, lorsqu'on fournit des données en entrée à un modèle, on reçoit un résultat en sortie.
- **Les hyper-paramètres** les hyper-paramètres sont les valeurs de réglage du modèle : le nombre d'itérations, les valeurs de seed (valeur aléatoire initiales), et les autres paramètres spécifiques des différents modèles testés.
- **La phase de validation** : établit la performance du modèle en termes de taux de faux positifs (les fausses alertes) et de faux négatifs (les ratés) que l'on doit réduire simultanément.

La première étape consiste à sélectionner et à préparer un ensemble de données d'entraînement, utilisées pour nourrir le modèle de ML. Puis dans la deuxième étape on sélectionne un algorithme d'apprentissage à exécuter sur l'ensemble des données d'entraînement et associe les hyper-paramètres au modèle. Enfin, Un entraînement du modèle sera effectué et passera par la phase de validation. Il s'agit d'un processus itératif où l'efficacité du modèle est étudiée et les résultats sont comparés avec ceux qu'il aurait dû produire. Donc l'algorithme d'apprentissage est exécuté pour réajuster le modèle jusqu'à ce qu'il produise des résultats corrects où les performances du modèle sont améliorées.

III.4 Techniques ensemblistes d'apprentissage automatique

Les méthodes d'apprentissage d'ensemble sont un ensemble de méta-algorithmes qui combinent différentes techniques d'apprentissage automatique dans un modèle prédictif afin d'améliorer la qualité des résultats dans ce modèle. Cette technique ensembliste repose sur le concept de *the wisdom of the crowd*; Les modèles de machine learning réunis ensemble sont plus forts qu'un modèle de machine Learning tout seul à condition que les modèles soient un minimum compétent et suffisamment diversifiés. ci-dessous une description des différentes techniques ensembliste :

Bagging un nom dérivé de l'agrégation Bootstrap, Il s'agit de la première méthode efficace d'apprentissage d'ensemble qui améliore les modèles en termes de stabilité et de précision. Elle réduit la variance et aide à éviter l'overfitting¹.

La méthode consiste à créer plusieurs copies d'un même modèle en entraînant chaque copie sur une partie aléatoire du dataset en utilisant une technique de bootstrapping dans le but de former des modèles différents. Les résultats de ces modèles sont combinés par moyenne ou vote pour obtenir un seul résultat (voir Figure III.2) [38].

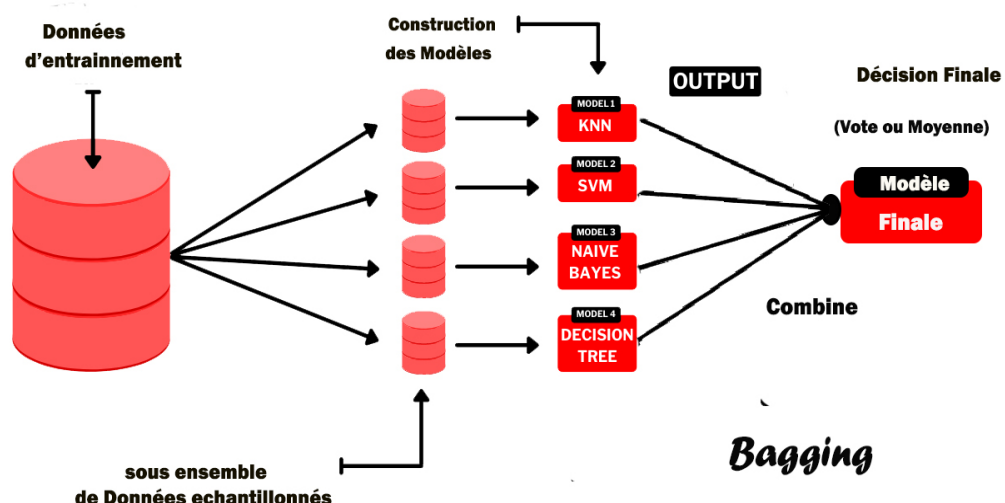


FIGURE III.2 – Principe de fonctionnement de Bagging [39]

1. L'overfitting (parfois appelé surapprentissage en français) survient lorsqu'un modèle, trop proche de données particulières, ne peut plus être généralisé. L'analogie est faite avec un humain qui apprend par cœur sans comprendre. Il lui est alors impossible de répondre à une question qu'il n'a encore jamais vue.

Boosting : consiste à construire une famille d'estimateurs qui sont ensuite agrégés par une moyenne pondérée des estimations (en régression) ou un vote à la majorité (en discrimination). Les estimateurs sont construits de manière récursive ; chaque estimateur est une version adaptative du précédent en donnant plus de poids aux observations mal ajustées ou mal prédites. L'estimateur construit à l'étape k concentrera donc ses efforts sur les observations mal ajustées par l'estimateur à l'étape $k - 1$. D'une manière fonctionnelle le terme boosting s'applique à des méthodes générales capables de produire des décisions très précises à partir de règles peu précises (voir Figure III.3).[40].

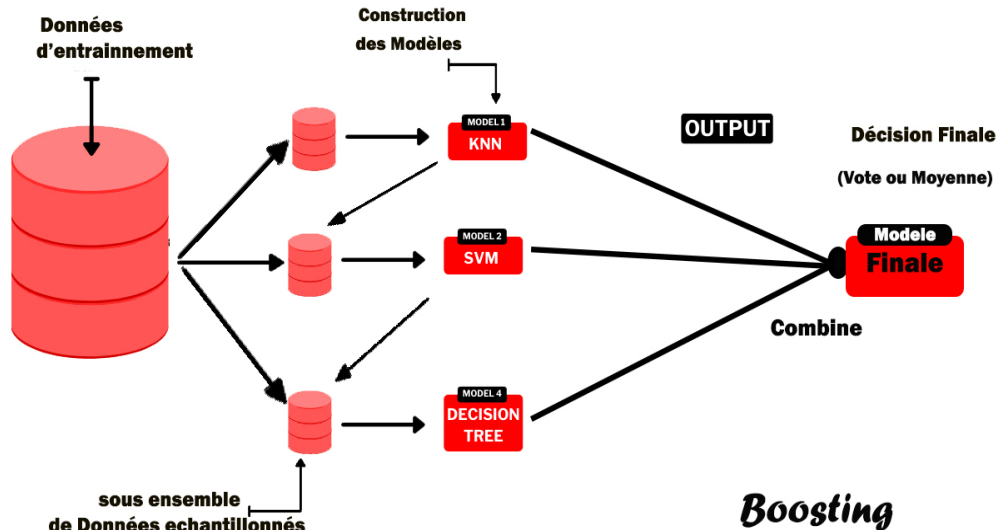


FIGURE III.3 – Principe de fonctionnement de Boosting[39]

Stacking : s'agit d'un procédé qui consiste à combiner plusieurs modèles très différents dans le but d'améliorer la qualité de la prédiction finale. L'idée générale est d'entraîner plusieurs modèles, et plutôt de choisir le meilleur modèle, tous les modèles sont agrégés à l'aide d'un autre modèle, qui effectue la prédiction finale (voir Figure III.4).[41].

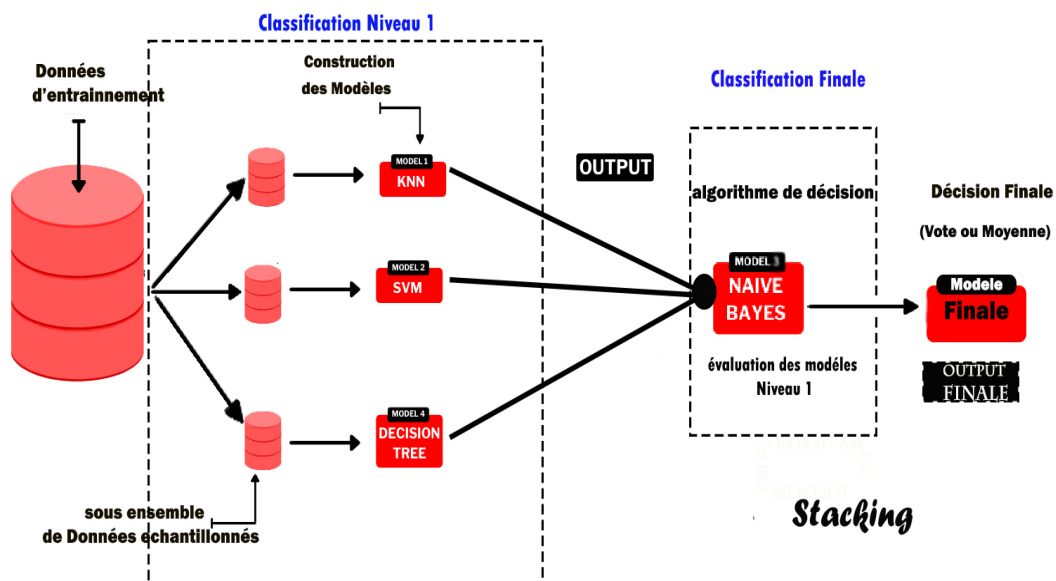


FIGURE III.4 – Principe de fonctionnement de Stacking[39]

III.5 Les modèles de classification

Il existe plusieurs algorithmes de classification ensembliste de technique bagging ou Boosting, parmi eux on cite respectivement Random Forest ou Adaboost. Dont l'algorithme arbre de décision est nécessaire pour Random Forest, et généralement appliquer dans AdaBoost. Voici leurs principes de fonctionnement :

III.5.1 Arbre de Décision

Les arbres de décision sont très utilisés en classification à cause de leurs simplicité d'interprétation et leurs qualité de précision relative. Leurs nature instable les rend aussi de bons candidats pour l'application des méthodes ensemblistes. En effet, beaucoup de travaux dans le Bagging et Boosting sont réalisés à base de cette technique.

L'arbre de décision est un schéma représentant les résultats possibles d'une série de choix interconnectés. Ces séries de choix représentent les feuilles de l'arbre qui correspondent à des classifications et les interconnexions représentent les branches de l'arbre [42]. Ainsi ce schéma d'arbres classifie les instances en commençant par le nœud racine de l'arbre, et testant l'attribut spécifié par ce nœud, puis en descendant la branche de l'arbre correspondant à la valeur de l'attribut. Ce processus est ensuite répété pour le sous-arbre dont la racine est le nouveau nœud jusqu'à arriver au nœud feuille qui correspond à la classification de l'instance.

III.5.2 Random Forest

La forêt aléatoire est l'un des algorithmes les plus puissants utilisés pour la modélisation prédictive. Le principe est la construction de plusieurs arbres de décision à partir d'un ensemble de données donné et les résultats sont combinés pour faire des prédictions. Pour construire des arbres de décision multiples, l'ensemble de données est divisé à plusieurs reprises en sous-arbres en changeant la combinaison des variables[43]. La précision de l'algorithme Random Forest dépend dans ce cas de la combinaison de variables. En effet, chaque arbre de décision individuel généré fait sa propre prédiction. Certaines peuvent être justes et d'autres fausses. Les arbres individuels qui ont produit des prédictions correctes se renforcent mutuellement, tandis que les prédictions erronées sont annulées. Pour que cela se produise, les arbres individuels générés doivent être non corrélés. C'est là qu'intervient la technique de Bagging, qui permet de générer des arbres de décision avec une corrélation minimale. Les étapes ci-dessous et la figure III.5 montre le fonctionnement de l'algorithme RF :

1. On prend un nombre X d'observations du jeu de données de départ (avec remise).
2. On prend un nombre K des M variables disponibles.
3. On entraîne un arbre de décision sur ce jeu de données.
4. On répète les étapes 1. à 4. N fois de sorte à obtenir N arbres.
5. Pour une nouvelle observation dont on cherche la classe on descend les N arbres. Chaque arbre propose une classe différente. La classe retenue est celle qui est la plus représentée parmi tous les arbres de la forêt (Vote majoritaire).

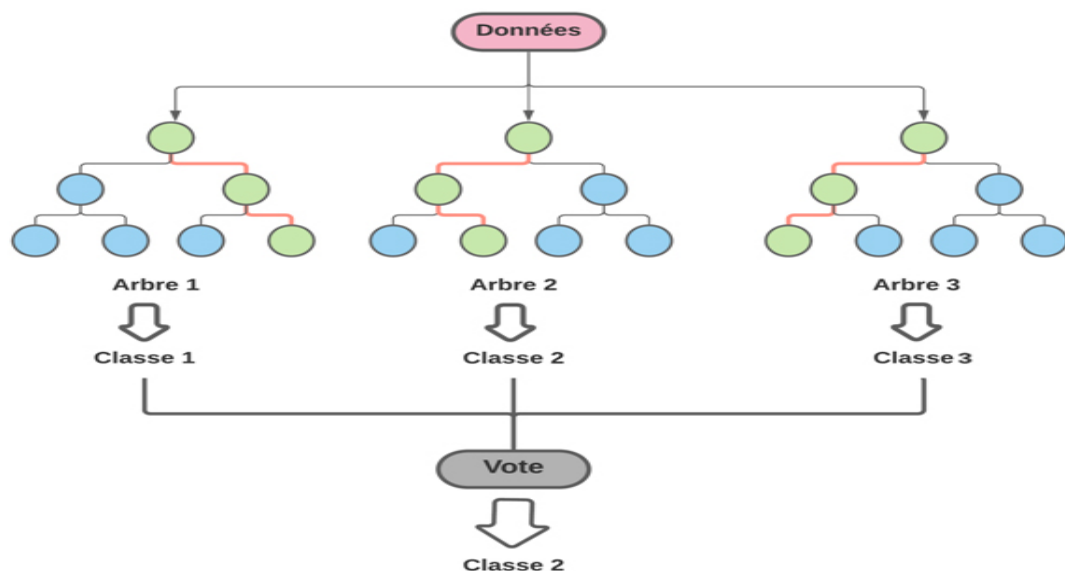


FIGURE III.5 – Principe de fonctionnement de RF

III.5.2.1 Avantages de Random Forest :

- L'algorithme Random Forest peut traiter de grands ensembles de données.
- La précision est élevée par rapport aux autres algorithmes d'apprentissage automatique.
- L'implémentation est plus facile et plus rapide que n'importe quel autre algorithme.
- Il surmonte le problème de l'overfitting.

III.5.2.2 Inconvénients de Random Forest :

- Utilise plus de mémoire pour construire une forêt.
- Les modèles de forêt aléatoire sont difficiles à interpréter.
- Valeurs extrêmes souvent mal estimées dans le cas d'un problème de régression.

III.5.3 AdaBoost

Il s'agit de l'algorithme le plus populaire appartenant à la famille des algorithmes boosting. Cette méthode repose sur le principe de sélection itérative de classificateurs faibles en fonction des exemples d'apprentissage. Un poids est associé à chaque exemple pour l'erreur que commet le classificateur final à son sujet. Le poids des exemples est utilisé dans le calcul de l'erreur de classification des classificateurs faibles. Plus un exemple est actuellement mal classé, plus son poids est fort et plus son importance dans le calcul de l'erreur des classificateurs faibles sera prépondérante. Les classificateurs faibles sont choisis pour minimiser leurs erreurs de classification. Au cours des itérations, l'algorithme va donc se concentrer sur les exemples qu'il a du mal à bien classer et se désintéresser progressivement des exemples qui sont toujours bien classés. Les exemples mal classés prennent de plus en plus d'importance dans le choix des futurs classificateurs faibles sélectionnés, ainsi le classificateur finalement généré est doté d'une meilleure précision que les classificateurs faibles[44].

III.5.3.1 Avantages de AdaBoost :

- Rapide, Simple et facile à implémenter.
- Applicable à de nombreux domaines par un bon choix de classificateur faible.
- Il prévient l'overfitting.

III.5.3.2 Inconvénients de Adaboost :

Adaboost inadapté dans les cas suivant :

- Pas assez de données.
- Comité de classificateurs faibles trop restreints.
- Classificateurs faibles trop stables.
- Classificateurs faibles trop forts.

III.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à quelques notions de base de l'apprentissage automatique, et leurs différents types, ainsi que les différentes méthodes ensemblistes. Nous avons aussi introduit quelques algorithmes ensemblistes de classification les plus répandus tels que Random Forest et adaboost, et aussi leurs principes de fonctionnements.

Chapitre IV

Conception de la solution

IV.1 Introduction

Ce chapitre va s'étaler sur la proposition d'un plan de mise en œuvre de notre solution. Nous présenterons notre contexte de travail ensuite nous allons décrire notre solution de détection et d'atténuation d'attaque, où nous allons présenter l'architecture générale ainsi que ses principaux modules, sans oublier les détails sur l'approche machine learning adaptée pour notre solution.

IV.2 Problématique

Traditionnellement, l'attaque DOS/DDOS volumétrique consomme, d'une façon drastique, la bande passante du réseau entre le service ciblé et les clients. Pour réaliser cette attaque, les pirates injectent un énorme volume de trafic à l'aide d'hôtes ou d'ordinateurs compromis. De nos jours, les attaques DOS/DDOS ciblent des environnements comme l'infrastructure en nuage[45], les réseaux mobiles et sans fil[46].

Cependant une autre catégorie d'attaque DOS, connue sous le nom de DOS lente, cible les ressources des applications et des serveurs en injectant un faible volume de trafic légitime à un rythme très lent. Ces attaques nécessitent très peu de bande passante et peuvent être difficiles à atténuer, car elles génèrent un trafic très difficile à distinguer du trafic normal. De plus, comme elles ne nécessitent pas beaucoup de ressources, les attaques DOS lentes peuvent être lancées avec succès en utilisant un seul ordinateur.

La Figure IV.1 illustre la différence entre le trafic normal, les attaques DOS/DDOS volumétriques et les attaques DOS lentes. Comme le montre la figure, en considérant le volume de trafic et la vitesse de transmission, le trafic normal et le trafic DOS lente se chevauchent. Il est donc difficile de distinguer les attaques DOS lentes du trafic normal et de les prévenir.

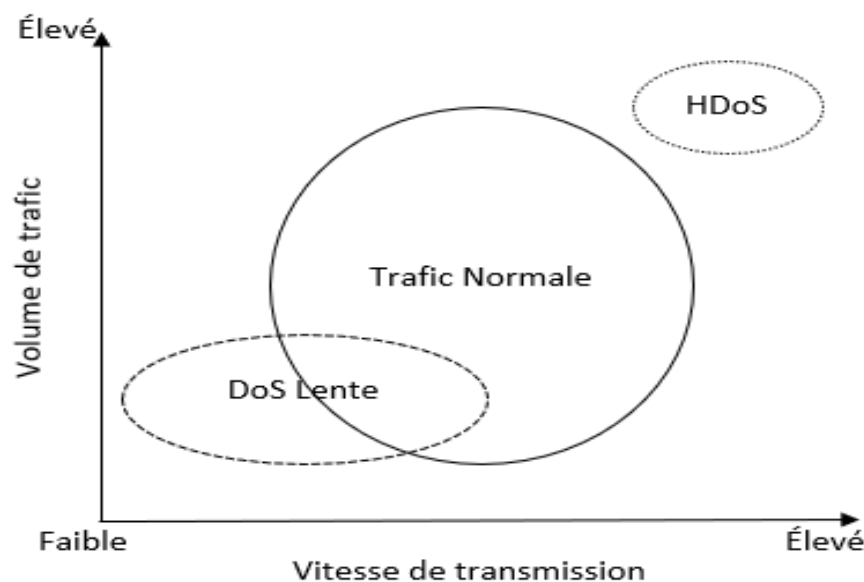


FIGURE IV.1 – Illustration du trafic normal, du DOS lent et du DOS volumétrique

De nos jours, les services basés sur le WEB sont des méthodes courantes utilisées et fournies par les prestataires de services tels que les services aux citoyens, les services en nuage, les services bancaires et financiers. Par conséquent, une attaque lente de type DOS sur les serveurs web dans ces environnements peut avoir un effet catastrophique sur le fonctionnement

du service.

Une étude a observé que la plupart des serveurs WEB modernes sont vulnérables aux attaques DOS lentes[47]. Une autre étude sur l'impact des attaques DOS sur des serveurs WEB est présentée dans[48]. Des études récentes montrent que HTTP/2, la version actualisée du protocole HTTP, est également vulnérable à de nombreuses attaques DOS lentes [49]. Par conséquent, la détection et la prévention des attaques DOS lentes ciblant HTTP a une importance primordiale dans l'internet d'aujourd'hui.

IV.3 DOS lent dans http

Les attaques DOS lent cible la couche application du modèle OSI en envoyant un trafic légitime à un taux très faible. La propriété commune de ces attaques est que les serveurs semblent avoir un grand nombre de clients connectés, mais la charge de traitement réelle est très faible. Le protocole HTTP étant un protocole de couche d'application utilisé sur internet, il est devenu l'une des cibles courantes des attaques DOS lentes. Les différents types d'attaques DOS lentes ciblant HTTP sont expliqués ci-dessous :

IV.3.0.1 Slowloris

Les serveurs WEB qui sont vulnérables à l'attaque SlowLoris commencent à traiter la requête seulement après l'avoir reçu complètement du client. En sachant cela, l'attaquant envoie au serveur web vulnérable des requêtes HTTP partielles pour ouvrir des connexions et de les maintenir en vie le plus longtemps possible. le serveur cible ne disposera que d'un nombre limité de threads pour gérer des connexions simultanées. Chaque thread du serveur tentera de rester en vie en attendant la fin de la demande lente, qui ne se produit jamais. Lorsque le nombre maximal de connexions possibles du serveur a été dépassé, chaque connexion supplémentaire ne reçoit pas de réponse, ce qui entraîne un déni de service [50]. On peut résumer cette attaque comme suit :

1. L'attaquant ouvre d'abord plusieurs connexions avec le serveur cible en envoyant plusieurs en-têtes de requête HTTP partielles.
2. La cible ouvre un thread pour chaque requête entrante, avec l'intention de fermer le thread une fois la connexion établie. Afin d'être efficace, si une connexion prend trop de temps, le serveur fermera la connexion excessivement longue, ce qui libère le thread pour la prochaine requête.
3. Pour empêcher la cible d'expirer les connexions, l'attaquant envoie périodiquement des en-têtes de demandes partielles à la cible afin de maintenir la requête en vie.
4. Le serveur ciblé n'est jamais en mesure de libérer l'une des connexions partielles ouvertes qu'une fois la demande prendra fin. Une fois que tous les threads disponibles sont utilisés, le serveur n'est plus en mesure de répondre aux requêtes supplémentaires effectuées à partir du trafic normal, ce qui entraîne un déni de service.

IV.3.0.2 POST lent

Dans une attaque POST lent, un attaquant commence par envoyer une en-tête HTTP POST légitime à un serveur WEB en fixant la taille exacte du corps du message qui suivra. À la réception de cette demande, le serveur alloue les ressources nécessaires au traitement de la donnée et de la longueur de son contenu spécifiées. Ensuite, le client envoie les données

à un rythme extrêmement lent, ce qui a pour effet de maintenir la connexion ouverte et d'épuiser toutes les ressources du serveur pour rendre impossible une connexion légitime[51].

IV.3.0.3 Lecture lente

Dans le cas de la lecture lente, le client envoie des requêtes HTTP légitimes au serveur et lit la réponse à un rythme très lent. Cela est réalisable en envoyant un paquet avec une taille de fenêtre nulle. A la réception de ce paquet, le serveur jugera que le client est entrain de lire les données et maintient donc la connexion ouverte. Par conséquent la cumulation de ces requêtes a pour effet de consommer les ressources du serveur, empêchant ainsi les demandes légitimes de passer[52].

IV.4 Synthèse sur les travaux existants

Un certain nombres d'études antérieures ont examiné l'utilisation de diverses approches ML, ainsi que d'autre techniques pour lutter contre les attaques DOS/DDOS. Parmi les principaux travaux qui ont abordée le contexte d'attaques DOS lent, sont présentés dans cette section selon l'ordre chronologique de leur publications :

S.Shafieian et al. (2015) [53] :La solution proposée par ces auteurs est basée sur l'algorithme Random forest pour la détection des attaques DOS de type lecture lente. Ils créent leur propre dataset, plutôt que d'utiliser des ensembles de données existantes car il juge que les attaques DOS à lecture lente n'existent pas sur des jeux de données ultérieures. Leurs expériences montrent que cette approche offre une précision de classification élevée et un faible taux de faux positive et négative.

T.Hirakawa et al. (2016) [54] : Les auteurs ont proposé une méthode efficace pour atténuer les attaques de type DOS lente en bloquant les connexions d'attaque d'une manière sélective relativement à la durée et au nombre de connexion de chaque adresse IP connectée au réseau.

N.Tripathi et al. (2016) [55] : Les auteurs ont présenté un système de détection des attaques HTTP DOS lentes plus spécifiquement POST lent et lecture lente basé sur la mesure des anomalies statistiques qui mesure la distance de Hellinger entre deux distributions de probabilité générées pendant la phase de formation et l'autre à la phase de test. Ces deux distributions comprennent des requêtes complètes et incomplètes GET et POST. Pendant la période d'attaque la proportion de requêtes incomplètes augmente dans le trafic global, ce qui fait que la distribution de probabilité correspondante s'écarte de la distribution de probabilité normale (requête complètes), et entraîne une anomalie. Les résultats expérimentaux montrent que le système proposé détecte les attaques DOS lentes.

K. Hong et al. (2018) [56] : Les auteurs ont proposé un système de défense contre les attaques DOS lente, assisté par un réseau défini par logiciel qui peut détecter et atténuer ces attaques dans le réseau. Ils proposent une méthode qui teste si le nombre de requêtes incomplètes est supérieur à un certain seuil, un système de défense est déclenché pour abandonner les requêtes incomplètes. Les résultats de simulation montrent que la méthode proposée protège avec succès les serveurs WEB touchés par cette attaque.

A.Dhanapal et al. (2019) [57] : Les auteurs ont discuté dans cet article les différents types d'attaques DOS lentes, ainsi leurs détection et atténuation dans un environnement cloud. Ils utilisent une technique de pré-monotoring du réseau pour détecter les clients qui émettent des requêtes lentes. Dans le cas d'une détection d'attaque, le client sera immédiatement mis dans la zone des listes de clients bloqués.

C.Calvert et al. (2019) [58] : Les auteurs ont mis en œuvre une expérience de collecte de données basée sur Netflow, ils ont proposé une approche ML pour la détection d'attaque DOS lente dans un environnement réel en utilisant plusieurs classificateurs, parmi eux, on cite K-NearestNeighbor, SVM et Random Forest. Avec la sélection de 14 attributs, les meilleures performances de détection sont obtenues par le classificateur Random Forest à un taux de précision de 0.9989.

I. Sumantra et al. (2020) [59] : Les auteurs exploitent la capacité de programmation et contrôle centralisé de SDN pour détecter les attaques telles que TCP SYN flood, Ping flood et les attaques DOS lente en se basant sur la technique de l'entropie. Cette approche statistique utilise l'IP source dans le réseau et divers attributs des drapeaux TCP à des fins de calcul d'entropie. Les résultats expérimentaux montrent que la méthode proposée a amélioré les performances pour servir une requête légitime en présence d'une attaque.

C.Kemp et al. (2020) [60] : Dans cet article, les auteurs ont donné une approche pour la détection des attaques de type lecture lente. Leur expérience évalue les ensembles de données collectées qui proviennent d'un environnement réseau réel afin de déterminer l'exactitude et l'efficacité de plusieurs modèles de détection d'attaques de lecture lente. Ils intègrent quatre classificateurs qui sont Décision tree, Random Forest et SVM dont les meilleurs résultats sont obtenus par le classificateur Random Forest. L'expérience démontre que les attributs collectés sont suffisamment discriminantes pour détecter de telles attaques.

J.Arturo et al. (2020) [61] : Dans cet article, les auteurs ont proposé une approche de détection et d'atténuation des attaques de type DOS faible et DOS lente en se basant sur une technique d'apprentissage automatique s'appuyant sur le dataset CICDoS 2017 et six algorithmes suivant J48, Random Tree, REPTree, MLP, SVM et RF. Selon eux l'algorithme des réseaux de neurones MLP est le plus favorisé pour classifier et distinguer le trafic légitime et malveillant.

IV.5 Motivation de travail

Comme nous l'avons fait remarquer dans la sous-section précédente, nous constatons que les serveurs WEB actuels restent vulnérables à de nombreuses attaques de type DOS lente et sont difficile à détecter.

Par ailleurs, nous avons trouvé quelques travaux concernant ce type d'attaques, ces derniers consistent entre autres, à présenter plusieurs techniques de détection et prévention d'attaques, notamment celle de ML qui s'est distinguée en offrant une grande flexibilité dans le processus de classification, ce qui a amélioré la détection du trafic malveillant. [62][63]. Cependant, la plupart des travaux cités traitent un ensemble restreint de types d'attaques DOS lente, et la majorité de ces études ont abordé la problématique seulement dans le cadre d'un environnement réseau traditionnel ou spécifique.

IV.6 Solution Proposée

Dans ce mémoire, nous proposons un système complet pour la détection et l'atténuation des attaques DOS lente dans un environnement SDN en se basant sur une approche ensembliste de machine learning. Notre étude vise en particulier dans un premier temps à collecter des informations sur les flux TCP issus des requêtes HTTP, étant donné que le protocole TCP est utilisé dans les attaques DOS lente. Dans un second temps, à travers des données émanant de l'étape précédente, un modèle d'apprentissage automatique est lancé pour classer un flux comme bénin ou malveillant. Dans un troisième temps, pour protéger le réseau le flux malveillant sera atténué.

IV.7 Architecture de la solution

Pour réaliser un système de défense, l'approche que nous avons proposé a pour but de construire un système à quatre modules :

- Un module de collection de paquets.
- Un module d'extraction d'attributs.
- Un module de classification de flux.
- Un module d'atténuation d'attaques.

Le rôle et fonctionnement de chaque module sont expliqués ci-dessous :

Module de collection de paquets : ce module s'occupe de capturer et d'analyser tout le trafic réseaux en se mettant à l'écoute des paquets échangés sur tous les ports du Switch OpenFlow, et redirige une copie de chaque paquet sur un port dédié. Ces paquets sont filtrés de telle sorte à garder seulement les paquets TCP et éliminer les autres paquets OpenFlow. Ce module capture des paquets pendant 15 seconde, et transmet les paquets capturés au module d'extraction d'attributs.

Module d'extraction d'attributs : s'appuyant sur les informations d'entête de chaque paquet reçu du module précédant, ce module identifie d'une façon unique les paquets de même flux TCP, et calcule les attributs de chaque flux TCP nécessaires à la classification pour les transmettre au prochain module.

Module de classification de flux : pour détecter d'éventuelles attaques dans le réseau SDN, ce module met en œuvre le modèle de classification. Il prend en entrée les attributs de flux TCP calculés dans l'étape précédente afin de déterminer si un flux appartient à la classe normal ou malveillante. Lorsqu'il s'agit d'une classe d'attaque, le flux sera enregistré dans un fichier journal(stock des instances d'attaque), où le module d'atténuation prévient pour atténuer le trafic d'attaque.

Module d'atténuation d'attaque lorsque le trafic d'attaque est détecté par le module de classification et enregistré dans le fichier journal, ce module implémenté par le contrôleur sera prévenu pour consulter ce fichier et bloquer le trafic d'attaque associé instantanément, en fixant de nouvelles règles de flux qui refusent le trafic venant de la source d'attaque. Une fois que le trafic a été bloqué, le contrôleur est réglé sur un temps qui dure 10 secondes, après quoi il autorise le trafic venant de la source élue comme attaquant. Mais si l'attaque est toujours active, il détecte à nouveau ce trafic et le bloque pendant 10 secondes supplémentaire.

La solution proposée est caractérisée par une surveillance permanente du réseaux, qui sera lancée, d'une part par le module collection de paquets et d'autre part par les deux modules, l'un d'extraction d'attributs et l'autre de classification dans deux thread différent.

la figure montre l'architecture générale de notre solution

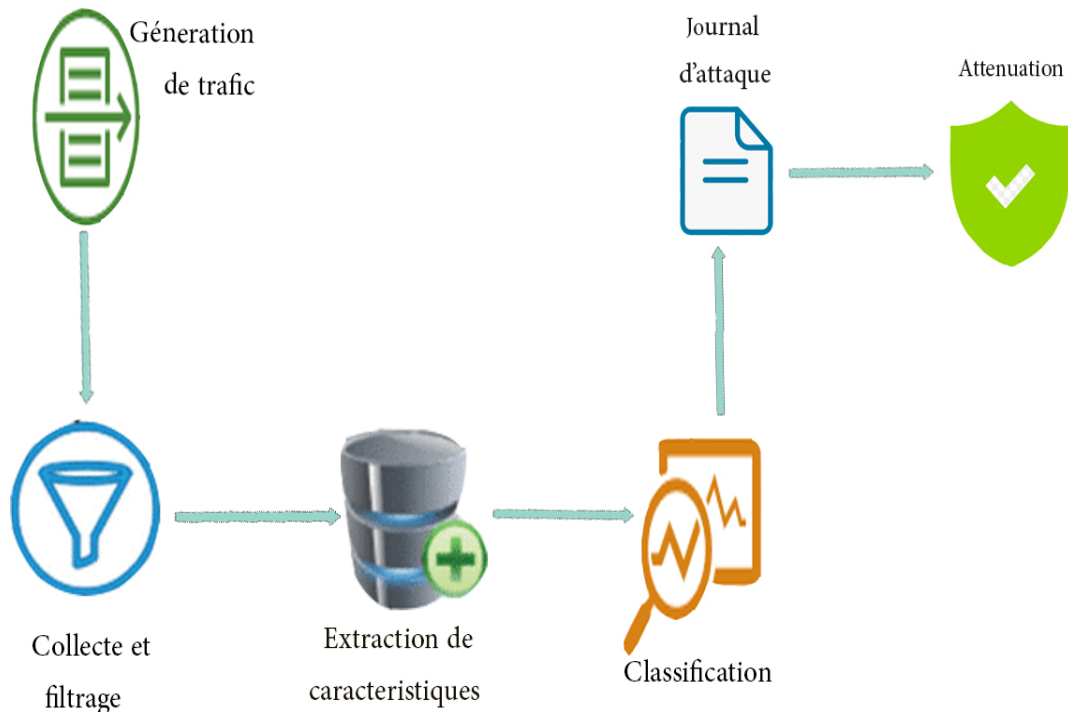


FIGURE IV.2 – Architecture générale de la solution

IV.8 Construction d'un modèle de Machine Learning

Vu que les attaques DOS lentes peuvent se présenter sous différentes formes, il est nécessaire d'apprendre de l'expérience d'apprentissage automatique en construisant un modèle ML capable d'analyser le trafic afin de reconnaître une attaque. Les étapes de base de l'apprentissage automatique, que nous implémenterons et discuterons par la suite, peuvent être résumées comme suit :

1. Collection de données (dataset).
2. Prétraitement d'ensemble de donnée.
3. Sélection d'attributs.
4. Création d'un sous-ensemble de données avec les attributs les plus pertinentes.
5. Entraînement de classificateur.

6. Évaluation du modèle.



FIGURE IV.3 – Processus Machine Learning

IV.8.1 Dataset

La construction des données d'apprentissage est l'étape la plus importante dans la mise en œuvre de l'approche d'apprentissage automatique, car c'est la quantité des données collectées qui détermineront la qualité du modèle à venir. Nous devons donc sélectionner un ensemble de données adéquat pour former notre modèle d'apprentissage automatique.

De nombreux jeux de données tels que DARPA, KDD'99, et LBNL ont été utilisés par les chercheurs pour évaluer la performance de leurs approches de détection et de prévention d'intrusions. Cependant, beaucoup de ces ensembles de données sont obsolètes et peu fiables [64]. Dans cette étude, les ensembles de données CICIDS2017, CSE-CIC-IDS2018 ont été utilisés, car ils incluent des attaques DOS modernes. En voici une description de ces deux ensembles de données :

CICIDS2017 : créé par l'Université du Nouveau Brunswick (UNB) en coopération avec l'Institut canadien pour la cybersécurité (CIC). L'ensemble de données CICIDS2017 ne contient pas seulement les scénarios d'attaque réseau les plus récents, mais aussi remplit tous les critères des cyberattaques du monde réel [64]. L'ensemble de données contient du trafic réseau bénin (normal) et anormal (différents types d'attaques), capturé pendant cinq jours consécutifs, dans lequel la capture du trafic est divisée en 8 fichiers différents. Pour chaque jour, un type d'attaque différent a été déployé. Le jeu de données est divisé en

15 classes avec 14 classes d'attaque et une classe de trafic normal. L'ensemble de données contient 83 attributs qui ont été extraits du trafic réseau à l'aide de l'outil CICFlowMeter, cet outil génère des flux bidirectionnels (Biflow), où le premier paquet détermine le flux forward (source à destination) et le flux backward (destination à source).

Notre travail se concentre sur les activités malveillantes DOS de la Capture du Mercredi 5 juillet 2017, fichier de capture, qui consiste en cinq attaques DOS/DDOS web : DOS GoldenEye, DOS Hulk, DOS Slowhttptest, DOS Slowloris, Heartbleed et d'une grande variété de trafic réseau bénin.

CSE-CIC-IDS2018[65] : représente un projet de collaboration entre le centre de la sécurité des communications (CSE) et l'institut canadien pour la cyber sécurité (CIC). L'ensemble des données comprend du trafic normal et trafic d'attaque dont 23 différentes attaques classées en 6 catégories : force brute, Heartbleed, Botnet, DOS, DDOS, attaques web. La construction de ce dataset a été effectuée en 16 jours dans la période entre le 14 février et le 2 mars 2018. Finalement, ils ont abouti à un dataset contenant également 83 attributs à l'instar de CICIDS2017, collecté en utilisant l'outil CICFlowMeter.

Dans notre cas, vu que notre travail s'intéresse seulement aux attaques DOS lente, alors pour les deux datasets CICIDS2017 et CSE-CIC-IDS2018 on a juste sélectionné les instances correspondantes à cette attaque nommé : DOS Slowhttptest, DOS Slowloris.

IV.8.2 Phase d'entraînement

Avec un ensemble de données d'entraînement et des attributs sélectionnées, nous pouvons former le modèle d'apprentissage automatique :

IV.8.2.1 Prétraitement d'ensemble de donnée

Pour obtenir de meilleurs résultats à partir du modèle appliqué dans les projets d'apprentissage automatique, et mettre les données dans un format propice au machine learning, il y'a la nécessité de prétraiter les données. Effectivement, à cause de la grande taille des bases de données actuelles et leurs natures brutes, ils sont généralement de faible qualité. Elles peuvent être incomplètes (valeurs manquantes ou agrégées), bruitées (valeurs erronées ou aberrantes) ou incohérentes (divergence entre attributs). L'application d'algorithmes d'apprentissage sur de telles données nuit à la performance ainsi qu'à la fiabilité du modèle. Donc Le prétraitement des données est une étape cruciale dans le processus d'apprentissage automatique.

La première étape de prétraitement dans notre travail consiste à obtenir les deux ensembles de données d'entrée CICIDS2017 et CSE-CIC-IDS2018, puis à les traiter. C'est-à-dire que les attributs ou les colonnes qui ont des valeurs infinies ou nulles doivent être modifiés ou supprimés en fonction de l'importance des données, et les données redondantes seront éliminées. De même, les valeurs contenant des caractères doivent être converties en valeurs numériques afin que les données puissent être traitées par les algorithmes d'apprentissage automatique. L'étape suivante consiste à sélectionner les attributs qui sont pertinents pour la détection des attaques. Nous formons ensuite les modèles à l'aide des algorithmes d'apprentissage automatique, et enfin, nous sauvegarderons les modèles formés et analyserons les résultats.

IV.8.2.2 Sélection d'attribut

L'objectif ultime de la sélection des attributs est de réduire la quantité de données en minimisant le nombre des attributs utilisés, car Il est fréquent qu'une partie de celles-ci ne

contienne que des informations non pertinentes, redondantes ou inutiles à la tâche de classification. Ainsi cette sélection d'attribut présente divers avantages, entre autres en améliorant la précision et la performance du classificateur, et réduisant le temps de calcul. Il est donc nécessaire, lors de la construction de notre système de classification, de limiter le nombre d'attributs pris en compte de manière à optimiser ses performances.

Les deux ensembles de données CICIDS2017 et CSE-CIC-IDS2018, présentent une variété d'attributs. Cependant, tout les attributs ne sont pas utilisés dans notre approche pour classer un flux réseau. Par conséquent, nous avons suivis un processus de sélection d'attributs qui se distingue en deux étapes : la génération du sous-ensemble, et l'évaluation du sous-ensemble.

- La génération du sous-ensemble est une stratégie de recherche utilisée pour déterminer des sous-ensembles d'attributs candidats pour l'évaluation. La recherche est effectuée en sélectionnant des attributs dont le score de Gini¹ est élevé.
- L'évaluation du sous-ensemble : un certain critère d'évaluation est estimé pour mesurer la qualité du sous-ensemble candidat. Ensuite il est comparé avec le meilleur sous ensemble précédent pour déterminer si ce sous-ensemble est convenable ou non. Si le nouveau sous-ensemble candidat est meilleur, il remplace le précédent meilleur. En répétant ce processus le sous-ensemble associé à la meilleure valeur du critère est sélectionné.

A la fin de l'exécution de ce processus de sélection d'attributs, nous aurons abouti à un sous-ensemble de 14 attributs induisant une performance de classification similaire à celle induite par l'ensemble initiale mais en améliorant le temps de calcul ou d'apprentissage. L'ensemble d'attributs sélectionné est décrit ci-dessous :

1. Score de Gini également connue sous le nom d'indice d'impureté de Gini, il permet de mesurer l'importance des attributs.

Attribut	Description
Total Length of Fwd Packets	Taille totale des paquets transférés de la source à la destination
Fwd Packet Length Mean	Taille moyenne des paquets transférés de la source à la destination
Bwd Packet Length Mean	Taille moyenne des paquets transférés de la destination à la source
Flow Bytes/s	Nombre d'octets transférés par seconde
Flow Packets/s	Nombre de paquets transférés par seconde
Bwd Header Length	Nombre total d'octets utilisés pour les en-têtes de la source à la destination
Bwd Packets/s	Nombre de paquets de la destination à la source par seconde
Max Packet Length	Longueur maximale d'un packet d'un même flux
PSH Flag Count	Nombre de paquets avec PUSH
ACK Flag Count	Nombre de paquets avec ACK
Average Packet Size	Taille moyenne des paquets
Avg Bwd Segment Size	Taille moyenne observée des segments dans de la destination à la source
Subflow Fwd Bytes	Le nombre moyen d'octets dans un sous-flux de la source à la destination.
fwd seg min	La taille minimum des segment observer dans le sens source à la destination.

TABLE IV.1 – Descriptions des attributs sélectionnées

IV.8.2.3 Entraînement de classificateur

Pour réaliser un modèle de classification, nous avons opté pour l'algorithme Random Forest, car il nous a permis d'obtenir de meilleures performances contrairement à d'autres algorithmes ensemblistes en raison de différents avantages qu'il présente.

Nous savons que les hyper-paramètres sont des valeurs de réglage du modèle jouant un rôle important pour obtenir des résultats précis. Dans notre cas, étant donné l'augmentation du nombre d'arbres de décision et comme Le nombre d'arbres de décision est un hyper-paramètre spécifique, la précision augmente. Alors, nous avons généré 100 arbres de décision afin de fournir une meilleure précision de détection sans introduire de frais de traitement importants. Aussi vu que les arbres de décision de la forêt aléatoire sont générés à partir d'échantillons de données d'entraînement et d'attributs aléatoires, cela pourra affecter la précision de la prédiction d'une manière négative, et pour y remédier, la technique de Bootstrapping est mise en œuvre, comme hyper-paramètres qui permet de construire des arbres de décision avec une corrélation minimale. Enfin le classificateur RF est appliqué aux données d'entraînement

préparées pour construire le modèle capable de classer le trafic réseau en tant que trafic normal ou d'attaque.

IV.8.3 Évaluation du modèle

Pour évaluer les performances de nos modèles de classifications, nous avons fixé l'ensemble de test à 20 % de l'ensemble de données initiales, et 80 % de données pour l'apprentissage (données d'entraînement). Cette répartition de l'ensemble de données initiales est généralement utilisé et choisi pour avoir un ensemble de données d'entraînement plus enrichi mais sans imposer un énorme coût de calcul lors de la formation ou lors de l'évaluation. la figure IV.4 ci-dessous illustre le processus d'entraînement et d'évaluation du modèle.

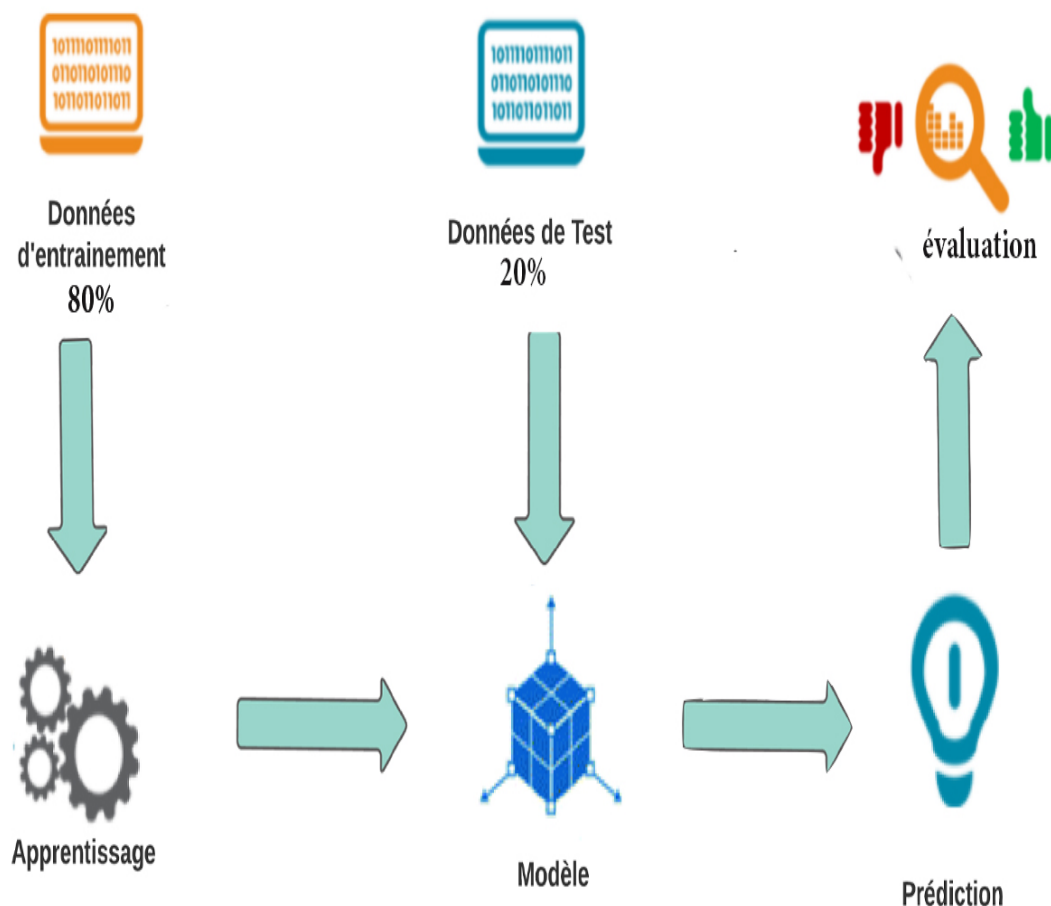


FIGURE IV.4 – Processus d'entraînement et d'évaluation du modèle

Les performances de l'algorithme d'apprentissage automatique sont évaluées en utilisant les paramètres suivants : Recall, Précision, F1-score (F1), Accuracy. Nous utilisons une matrice de confusion pour calculer ces mesures de performance.

Matrice de confusion : est une matrice $N \times N$ tel que N est le nombre de classes. Dans notre cas, nous avons 2 classes (attaque et normal), où les colonnes de la matrice représentent les classes prédites et les lignes représentent les classes réelles. La matrice de confusion donne le nombre de résultats correctement et incorrectement prédits par le modèle. Le tableau IV.2 montre la matrice de confusion de notre approche proposée.

		classe prédite	
		Positive	Négative
classe réelles	Positive	<i>VP</i>	<i>FP</i>
	Négative	<i>FN</i>	<i>VN</i>

TABLE IV.2 – Matrice de confusion pour un problème de classification à 2 classes

Avec :

Vrai positif (VP) : est le nombre de cas où le modèle prédit correctement la classe positive.

Vrai négatif (VN) : est le nombre de cas où le modèle prédit correctement la classe négative.

Faux positif (FP) : est le nombre de cas où le modèle prédit incorrectement la classe positive.

Faux négatif (FN) : est le nombre de cas où le modèle prédit incorrectement la classe négative.

Recall : le nombre de flux correctement attribués à la classe *i* au regard du nombre de flux appartenant à la classe *i*.

$$R = \frac{VP}{VP+FN}$$

Précision : le nombre de flux correctement attribués à la classe *i* au regard du nombre de flux attribués à la classe *i*.

$$P = \frac{VP}{VP+FP}$$

F1-score (F1) : une mesure qui combine la précision et le rappel et leur moyenne harmonique.

$$F1_score = 2 * \frac{\text{rappel} * \text{précision}}{\text{rappel} + \text{précision}}$$

Accuracy : nombre de prédictions correctes par rapport au nombre total de prédictions.

$$P = \frac{VP+VN}{VP+FP+FN+VN}$$

IV.9 Fonctionnement de notre système dans le cadre SDN

Des contre-mesures peuvent être mises en place pour atténuer les menaces de sécurité dans le réseau SDN. Parmi eux, la limitation du débit et l'élimination des paquets, peuvent être appliqués pour éviter les attaques DOS sur le plan de contrôle ou sur le plan de données. Cependant, ces techniques ne sont pas encore prises en charge ou mises en œuvre dans les déploiements SDN, mais ultérieurement elles peuvent être mises en œuvre dans différents dispositifs, tels que les contrôleurs ou les dispositifs de transfert [12].

Dans notre cas grâce au contrôle centralisé du contrôleur, on a pu atténuer les attaques DOS lente sur le plan de donné moyennant des règles OpenFlow installées au niveau des commutateurs par le contrôleur. Concrètement si un flux est reconnu comme anormal par notre système de détection, celui ci va l'enregistrer dans le fichier journal, et c'est finalement à la charge du contrôleur de déceler l'attaque courante dans ce fichier afin de la bloquer, par la création et l'ajout dans la table des flux du commutateur cible d'un flux IP, dont l'action est de supprimer les paquets provenant de la source d'attaque. Dans notre cas, le trafic malveillant sera bloqué temporairement afin de s'assurer que les flux malveillants faux positifs ont la chance d'atteindre la destination ultérieurement. Les étapes de notre système de détection et d'atténuation sont résumées dans l'algorithme ci-dessous :

Algorithme 1 Système de détection et d'atténuation d'attaque

Entrées : dataset T , trafic de flux F

- 1: Obtenir les flux de trafic
 - 2: **pour** Flow $f \in F$ **faire**
 - 3: Obtenir les statistiques de flux
 - 4: Extraire les attributs
 - 5: **fin pour**
 - 6: Prétraitement de T
 - 7: Créez un sous-ensemble de données T' avec les attributs pertinents de T
 - 8: Entraîner le modèle RF sur T'
 - 9: Classification des flux à l'aide du modèle RF
 - 10: **si** Le classificateur classe le flux comme une anomalie **alors**
 - 11: Afficher une alerte de trafic d'attaque
 - 12: Ajouter la règle R_i qui refuse le trafic entre l'attaquant et la victime
 - 13: Attendre 10 seconde
 - 14: supprimer la règle R_i
 - 15: **si non**
 - 16: afficher le trafic comme normal
 - 17: **fin si**
-

IV.10 Pseudo code de l'implémentation de la solution

Dans cette section nous allons présenter un pseudo code qui résume les modules constituant notre système de détection et d'atténuation d'attaque. La figure IV.5 nous montre le pseudo code de la détection d'attaque.

```
var paquets[] : liste_paquet;
procedure collection_paquets()
debut
    //caputrer seulement les paquets TCP sur le port sniffer chaque 10 seconde
    tant que Vrai faire
        paquets = capture(port_sniffer, time_out = 10, display_filter='tcp')
    fait;
fin;

procedure extractionAttributs_classificationFlux()
var p:paquet, idFlow[]:cahine de caractere, mat_flux[][]:liste_flux, f:flux, attributs:liste_attribut, m:modele, prediction:cahine de caractere;
debut
    tant que Vrai faire
        pour p paquets faire
            //identifier tout les paquets d'un même flux bidirectionnel
            idFlow = str(min(p.ipsrc ,p.ipdest))+ '-' +str(max(p.ipsrc, p.ipdest))+ '-' +str(min (p.portSrc ,p.portDest))+ '-' +str(max(p.portSrc ,p.portDest))
            //enregistrer chaques paquets 'p' d'un même flux bidirectionnel 'idFlow' dans la structure de donnée 'mat_flux'
            enregistrer_paquets(mat_flux, p, idFlow)
        fait;
        pour f mat_flux faire
            //calculer les attributs pour chaque flux
            attributs = calcule_d'attributs(f)
            //charger le modele de clacification
            m = charger_modele_classification()
            //classifier le flux
            prediction = predire(m, attributs)
            afficher(prediction)
            //si le flux f est mallvaient alors il sera enregistré dans le fichier journal
            si(prediction == 'attaque')alors
                maj_journal(f)
        fait;
    fait;
fin;

algorithme:
debut
    faire en parallele
    debut
        //collection de paquets
        collection_paquets()
        //extraction d'attributs et classification de flux
        extractionAttributs_classificationFlux()
    fin;
fin;
```

FIGURE IV.5 – Pseudo code de la détection d'attaque

Dans la procédure *extractionAttributs_classificationFlux()*, l'identification de l'ensemble des paquets appartenant à un même flux est calculé de la manière suivante :

$$\text{idFlow} = \text{str}(\min(\text{ipsrc}, \text{ipdest})) + '-' + \text{str}(\max(\text{ipsrc}, \text{ipdest})) + '-' + \text{str}(\min(\text{portSrc}, \text{portDest})) + '-' + \text{str}(\max(\text{portSrc}, \text{portDest}))$$

idFlow est un identifiant qui permet de regrouper les paquets venant de la même source vers la même destination ou inversement, dans un même flux.

La figure IV.6 montre la manière dont le contrôleur procède pour atténuer les attaques au sein du réseau SDN.

```
algorithme:
debut
    activationPareFeu()
    //Ajout de règles autorisant la communication TCP entre les machines de notre réseau SDN.
    ajoutRegle()
    tant que Vrai faire
        //surveiller la presence d'eventuelle nouvelles attaques enregistré dans le fichier journal
        attaque = lire_journal()
        si(attaque != NULL)
        {
            //recuperer l'adresse ip de la source d'attaque et de la victime
            ipSrc = recuperer_ipSrc()
            ipDest = recuperer_ipDest()
            //Ajout de règles refusant la communication TCP entre la machine attaquante et la machine victime
            ajoutRegleBloquante(ipSrc, ipDest)
            //attendre 10 seconde avant de supprimer les regles bloquantes
            attendre(10)
            //suppression de règles refusant la communication TCP entre la machine attaquante et la machine victime
            suppressionRegleBloquante(ipSrc, ipDest)
        }
    fait;
fin;
```

FIGURE IV.6 – Pseudo code de l'atténuation d'attaque

IV.11 Conclusion

Nous avons présenté dans ce chapitre les détails relatifs à la conception de notre système de détection des attaques de type DOS lentes en s'inspirant des méthodes de Machine Learning du chapitre précédent. Plus précisément, nous avons décrit l'architecture générale de l'approche proposé ainsi que les différents modules nécessaires à sa réalisation .

Juste après cette étape, nous pouvons passer à la phase d'implémentation de notre solution ainsi que les résultats obtenus qui feront l'objet du prochain chapitre.

Chapitre V

Implémentation de la solution

V.1 Introduction

L'objectif de ce chapitre est d'expliquer, étape par étape, la mise en place d'un système permettant la détection et l'atténuation d'attaque dans un environnement SDN. Nous présenterons les outils logiciels utilisés pour réaliser ce projet, nous décrirons l'implémentation des différents modules de l'approche proposée, nous terminerons ce chapitre par une simulation de notre solution dans un environnement SDN.

V.2 Environnement de travail

V.2.1 Contrôleur Ryu

Pour le choix du contrôleur, nous avons opté pour Ryu[66]. Il s'agit d'une plateforme de développement open source, utilisée pour créer un contrôleur SDN en utilisant le langage de programmation python, conçu de telle sorte à augmenter l'agilité du réseau en facilitant la gestion du trafic. Il fournit des composants logiciels avec des API bien définis qui permettent aux développeurs de créer facilement de nouvelles applications de gestion et de contrôle du réseau, parmi les applications existantes qu'on peut exécuter il y a un hub, un commutateur, un équilibreur de charge et un pare-feu. En ce qui concerne les protocoles de gestion des périphériques, Ryu prend en charge divers protocoles, tels qu'OpenFlow, Netconf, OF-config, etc.

V.2.2 Mininet

Mininet est un émulateur réseau open source qui émule un réseau SDN composé des commutateurs OpenFlow, un contrôleur SDN, les hôtes IP et des liens. Il permet de créer un réseau suivant une topologie réseau prédéfinie par Mininet (e.g. topologies linéaire, arbre,...), ou bien personnalisées selon les besoins de l'utilisateur en termes de nombre d'équipements, leurs types et la configuration des liens. Mininet fonctionne sur le noyau de Linux ce qui donne la possibilité aux hôtes d'exécuter des programmes python à leurs niveaux[67].

La principale raison d'utiliser Mininet est qu'il prend en charge le protocole OpenFlow, qui est essentiel pour la configuration et la communication du réseau SDN, et qu'il fournit également une plateforme peu coûteuse pour développer, tester et créer des topologies personnalisées dans le réseau.

V.3 Langage et bibliothèques utilisés

Python : Le choix du langage de programmation est Python [10]. Il s'agit d'un langage orienté objet, facile à prendre en main, l'un des plus utilisés pour faire du machine learning. Python possède un grand nombre de bibliothèques : d'analyse de données, de calcul scientifique, de visualisation et d'apprentissage automatique[68].

Scikit-learn : est une bibliothèque d'apprentissage automatique gratuite pour le langage de programmation Python. Elle contient de nombreux outils pour l'apprentissage automatique et la modélisation statistique, notamment la classification supervisée et la classification non-supervisée, la régression, le clustering et la réduction de dimensionnalité et des méthodes de validations[69].

Numpy : est une bibliothèque open source considérée comme fondamentale pour le calcul scientifique avec python. Elle est une bibliothèque très puissante pour la création et

la manipulation de tableaux et de matrices multidimensionnelles. Elle permet aussi de créer directement un tableau depuis un fichier ou le contraire, de sauvegarder un tableau dans un fichier[70].

Panda : c'est une librairie python qui permet d'utiliser et de manipuler des dataframes. Cette classe fournit un nombre important de méthodes très utiles pour manipuler des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles comme la manipulation des tables SQL ou feuilles de calcul Excel[71].

Matplotlib : est une bibliothèque python de visualisation de données multiplateformes. Elle permet de tracer et de visualiser des données sous plusieurs formes de graphiques (graphique à barres, carte thermique, histogramme, ...) [72].

Pyshark : est un wrapper (enveloppe) Python pour Tshark (utilitaire en ligne de commande de wireshark). il existe de nombreux modules d'analyse de paquets en python, celui-ci est différent car il n'analyse pas réellement les paquets, il utilise simplement la capacité de tshark à exporter des XMLs pour utiliser sa propre analyse. Cette bibliothèque qui analyse à partir d'un fichier de capture ou d'une capture direct, dispose de quelques objets Capture (Live, Remote, File, InMem). Chacun d'entre eux lit depuis sa source respective et peut ensuite être utilisé comme un itérateur pour obtenir ses paquets. Chaque objet de capture peut également recevoir divers filtres afin que seuls certains des paquets entrants soient enregistrés[73].

Iptraf : est une console logicielle qui fournit des statistiques réseau. Il fonctionne en collectant des informations à partir des connexions TCP, telles que les statistiques et les interfaces d'activité, par exemple il est utilisé pour surveiller l'activité sur une interface et donner ,en temps réel, le nombre de kilo octets qui transite, aussi bien, en entrée qu'en sortie.

V.4 Processus d'implémentation de la solution

Dans cette section nous allons aborder en détail les étapes d'implémentation des différents modules qui compose notre solution :

V.4.1 Implémentation des deux modules collection de paquets et extraction d'attributs

L'implémentation des deux modules collection de paquets et extraction d'attributs programmés avec python sont réalisés comme suit :

Pour réaliser le module de collection de paquets, nous avons utilisé le Port Mirroring. Cette fonction consiste à dupliquer les paquets qui entrent et sortent d'un port du commutateur spécifique (port source), et de les transmettre à un autre port spécifique (port destination). Ainsi cette fonction, nous permettra d'obtenir une copie de chaque paquets qui transite dans le réseau. Mais pour capturer les paquets dupliqués, nous avons utilisé Pyshark afin d'analyser et extraire les informations sur chaque paquet.

Vu que chaque instance de notre dataset représente un flux donné, et que chaque flux est bidirectionnel, alors le module d'extraction d'attributs est programmé avec Pyshark de telle sorte à récupérer la liste des paquets capturés par le module précédant, et regrouper les paquets venant de la même source vers la même destination ou inversement, dans un même flux identifier d'une manière unique. Enfin, pour calculer les attributs nécessaires à la

détection d'un flux donné, nous exploitons les informations de l'entête IP et TCP de chaque paquet de ce flux.

V.4.2 Implémentation du module de classification de flux

L'implémentation du module classification de flux est la phase la plus importante de construction de notre solution, sachant qu'elle va servir à l'obtention d'un modèle capable de classifier le flux réseau. Son implémentation est divisée en deux étapes : une étape d'apprentissage (entraînement) et une étape de classification (prédiction).

Dans l'étape d'apprentissage, on va procéder au test de plusieurs algorithmes d'apprentissage, afin de construire un modèle qui sera en mesure de classifier un flux réseau dans le cas réel en flux légitime ou d'attaque, ensuite à chaque entraînement d'un algorithme, nous allons entamer la dernière étape du processus de construction d'un modèle d'apprentissage automatique, qui est l'analyse et l'évaluation des performances du modèle construit à base de chacun de nos algorithmes.

Dans notre cas, nous avons choisi 6 algorithmes d'apprentissage supervisés ensemblistes et non ensemblistes qui sont : Decision Tree, k-nearest neighbors, AdaBoost, Naive Bayes, Random Forest et un algorithme à base de l'approche Stacking. Malgré que notre travail consiste, dès le début, de proposer une approche ensembliste nous avons intégré des algorithmes non ensemblistes dans le but de mener une étude comparative entre les algorithmes ensemblistes et non ensemblistes pour défendre notre choix d'algorithme.

Après l'entraînement de nos algorithmes, nous avons évalués les modèles obtenus selon les critères de performances suivants : Précision, Recall, F1_score et Accuracy.

Dans un premier temps, nous avons choisi un ensemble d'algorithmes de classification non ensembliste tels que DT, NB, KNN. Le tableau v.1 montre les différentes performances obtenues par chacun des classificateurs non ensemblistes .

Métrique \ Algorithme	DT	NB	KNN
Précision	0.99872	0.28693	0.99783
Recall	0.99951	0.95801	0.99601
Accuracy	0.99951	0.33754	0.99832
F1-Score	0.99911	0.44159	0.996926

TABLE V.1 – Performances des algorithmes non ensemblistes

Après analyses des performances obtenues de chaque algorithme, on constate que les meilleures performances sont obtenues par les deux classificateur DT et KNN avec des pourcentages qui dépassent les 99%. Pour aller plus loin dans la comparaison, la figure V.1 suivante nous montre les Matrices de confusions.

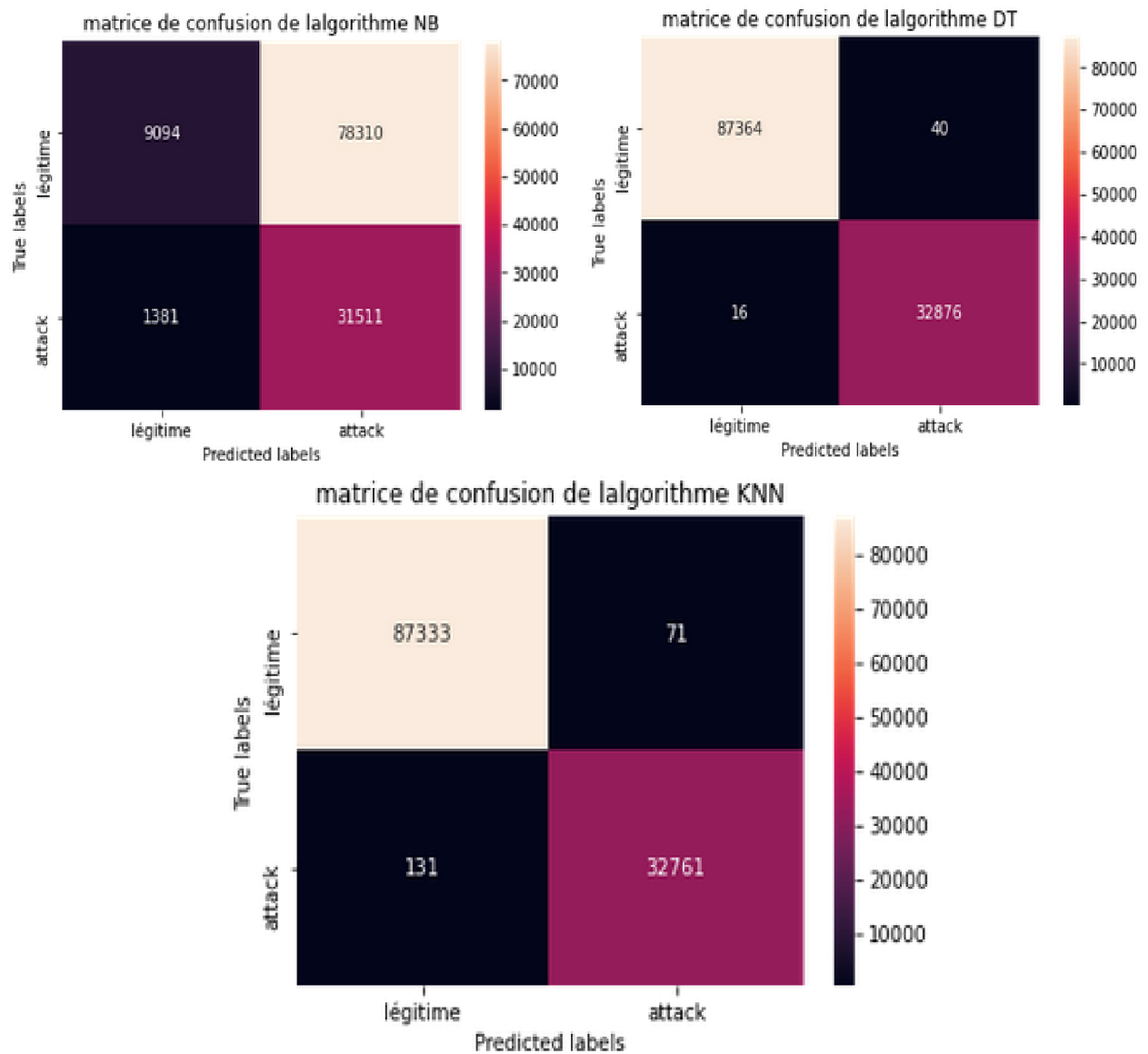


FIGURE V.1 – Matrices de confusions des algorithmes non-ensemblistes

Après l'observation de la figure V.1, on remarque que le classificateur DT est meilleur en terme de nombre VP et VN, en effet il a réussi à prédire 99,954% de flux légitime et 99,951 % de flux d'attaque.

A la suite de notre comparaison et dans le but d'améliorer plus les résultats et les performances de nos algorithmes entraînés précédemment, on passe à l'évaluation de l'approche d'apprentissage ensembliste avec l'utilisation des 3 techniques les plus connus, bagging, boosting et stacking dont les algorithmes sont respectivement Adaboost, RF et une combinaison de KNN, DT, SGDC avec l'approche de stacking. Le tableau V.2 nous montre les résultats obtenus par rapport aux critères de performance des différents algorithmes d'apprentissages ensemblistes.

Métrique \ Algorithme	AdaBoost	RF	Stacking
Précision	0.99400	0.99914	0.02779
Recall	0.99379	0.99972	0.00647
Accuracy	0.99666	0.99969	0.66640
F1-Score	0.99390	0.99943	0.01050

TABLE V.2 – Performances des algorithmes ensemblistes

Après la première vue, on observe bien que les algorithmes RF et AdaBoost sont les plus performants. Vérifiant cela à partir des matrices de confusion correspondantes à chacun de ces algorithmes comme indiquer dans la figure V.2 ci-dessous.

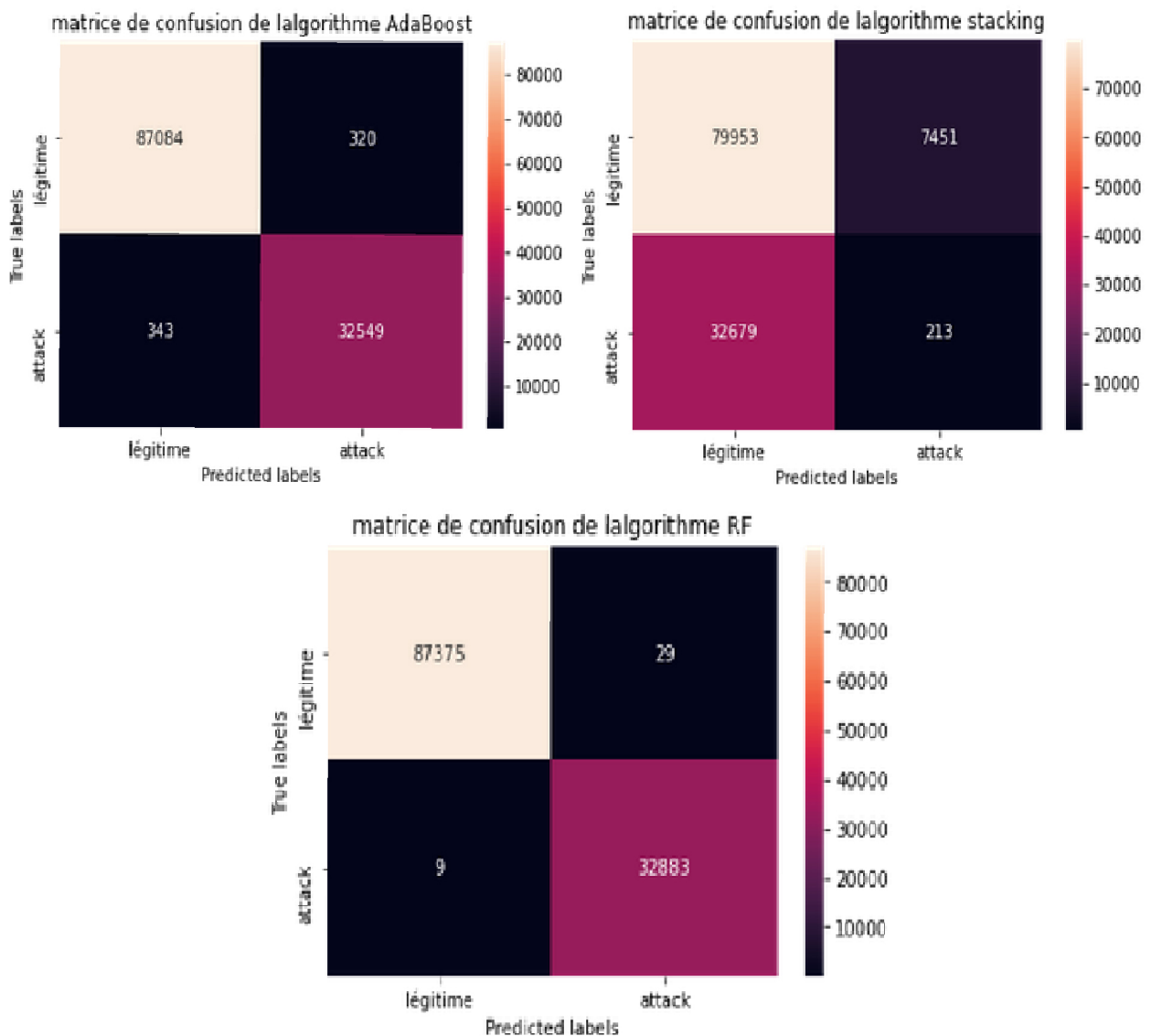


FIGURE V.2 – Matrices de confusions des algorithmes ensembliste

La figure V.2 montre que l'algorithme RF possède le plus grand nombre de VP et VN, tandis que l'approche de stacking a eu des mauvaises performances avec des nombres FP et FN très élevés.

Passons maintenant à la comparaison entre les deux approches supervisées non ensembliste et ensembliste. La figure V.3 nous donne une vue d'ensemble sur les résultats d'analyse des performances de tous les algorithmes qu'on a choisi.

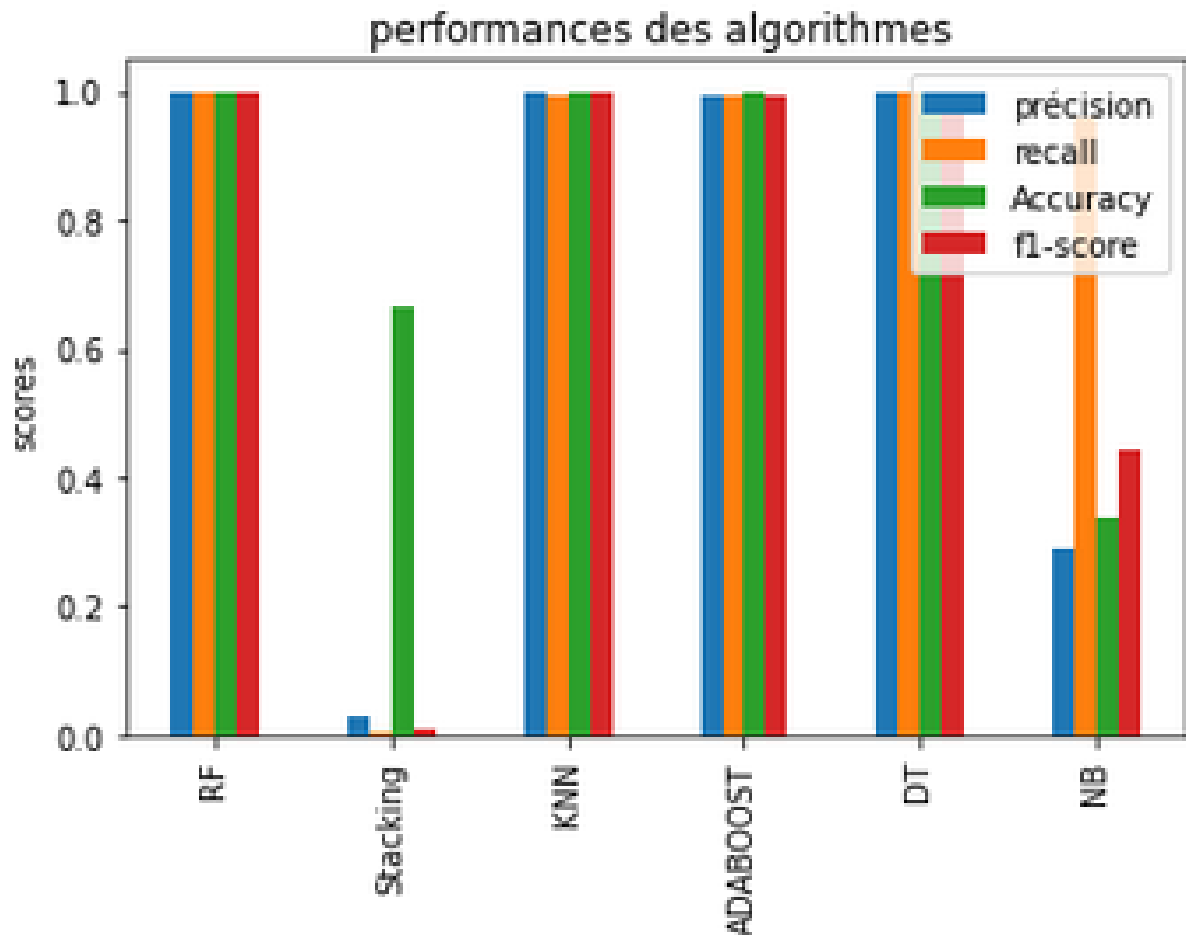


FIGURE V.3 – Résultat d'analyse des performances de tous les algorithmes

Après analyse et observation des figures et tableaux, on constate un rapprochement entre les performances obtenues des 4 algorithmes RF, KNN, DT et AdaBoost, avec des performances qui atteignent 99%, mais après l'analyse de la matrice de confusion, on remarque une légère différence qui apparaît au niveau des instances FN et FP, d'où les meilleures performances reviennent aux algorithmes RF, DT et AdaBoost.

A la fin de cette comparaison on a orienté notre choix de classificateur vers l'algorithme ensembliste RF le plus performant en terme de précision, recall, accuracy et F1-score, avec des performances qui dépassent les 99.99% pour chacune des métriques.

V.4.3 Implémentation du module d'atténuation d'attaque

Maintenant que nous avons pu constater que les APIs permettent d'interagir avec le contrôleur RYU, nous avons implémenté une fonction python sur notre contrôleur en utilisant les connaissances acquises dans le premier chapitre, et une application existante sur ce contrôleur, pour mettre en place un par-feu chargé d'atténuer les attaques DOS lentes. Grâce à l'ensemble des commandes, permettant notamment d'accéder aux APIs de l'application par-feu, nous parvenons à son implémentation en créant un script shell :

- **A l'initialisation :**

- Puisque le firewall n'est pas activé par défaut, alors nous l'activons grâce à la commande suivante :

```
curl -X PUT http://localhost:8080/firewall/module/enable/00000000000001
```

- Ajout de règles autorisant la communication TCP entre les machines de notre réseau SDN. La commande suivante permet d'ajouter une règle autorisant la communication TCP depuis la machine dont l'adresse IP est *ip_src* vers la machine destination dont l'adresse IP est *ip_dest* :

```
curl -X POST -d '{"nw_src": ip_src, "nw_dst": ip_dest, "nw_proto": "TCP"}' http://localhost:8080/firewall/rules/000000000000000001
```

- **A la détection d'une attaque :**

- Ajout de règles refusant la communication TCP entre la machine attaquante et la machine victime qui sont plus prioritaire que celle autorisant la communication. La commande qui permet d'ajouter une telle règle entre la machine dont l'adresse IP est *ip_src* et la machine destination dont l'adresse IP est *ip_dest* est la suivante :

```
curl -X POST -d '{"nw_src": ip_src, "nw_dst": ip_dest, "nw_proto": "TCP", "actions": "DENY", "priority": "10"}' http://localhost:8080/firewall/rules/000000000000000001'
```

- Après 10 seconde, suppression des règles refusant la communication TCP entre la machine attaquante et la machine victime. La commande suivante permet de supprimer une règle grace à son identifiant *idRule* :

```
curl -X DELETE -d '{"rule_id": idRule}' http://localhost:8080/firewall/rules/000000000000000001
```

V.5 Simulation

Pour notre simulation, nous avons crée une topologie personnalisée à l'aide du simulateur mininet, comprenant les unités suivante :le contrôleur RYU, un commutateur OpenFlow et 5 hôtes.

La topologie est composée de deux serveur WEB, l'un étant victime d'une attaque DOS lente par un client attaquant et l'autre qui assure la réception de requête d'un client légitime. En outre les différents modules de notre solution sont mis en oeuvre dans cette topologie. La figure V.4 et le tableau V.3 ci-dessous résume la topologie construite.

Hôte et contrôleur	@IP	Rôle
H1	10.0.0.1	Serveur Web recevant des requêtes HTTP légitimes
H2	10.0.0.2	Serveur Web recevant des requêtes HTTP malveillantes
H3	10.0.0.3	Client envoyant des requêtes HTTP légitimes
H4	10.0.0.4	Client envoyant des requêtes HTTP malveillantes
H5	10.0.0.5	Système de détection
c1	127.0.0.1	prévention d'attaque DOS lente

TABLE V.3 – Rôle des entités de la topologie

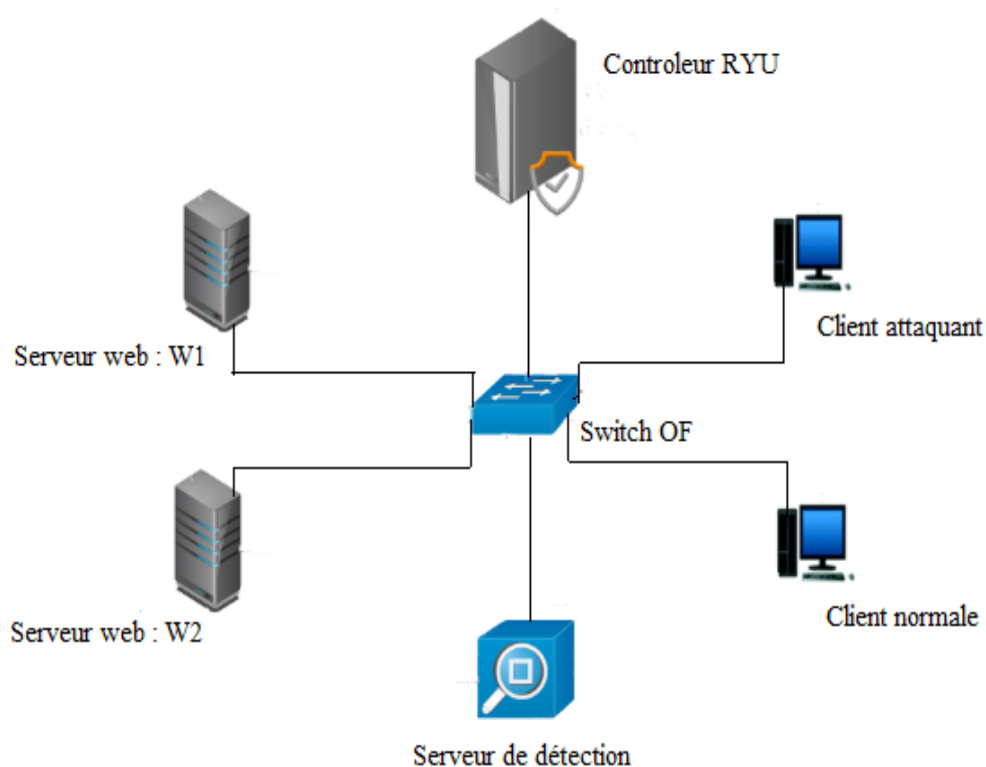


FIGURE V.4 – Architecture de la simulation

Afin de tester la solution proposée, nous utilisons l'outil et scripts python suivant :

Script web : ce script écrit en python crée un serveur web fonctionnant sur le port 80 en utilisant la bibliothèque HTTP, il assure la fonction principale d'un serveur web par le stockage et la délivrance d'un simple fichier HTML.

Slowhttpstest : est un outil hautement configurable qui simule certaines attaques par déni de service de la couche application. Il fonctionne sur la majorité des plateformes Linux, il met en œuvre les attaques de déni de service, les plus courantes, de la couche applicative à faible bande passante, telles que Slowloris, POST lent et Lecture lente

en épuisant le pool de connexions concurrentes, et provoquant une utilisation très importante de la mémoire et du CPU sur le serveur.

Script de trafic régulier : ce script écrit en python génère du trafic HTTP normale par l'envoi des requêtes légitimes au serveur WEB et la réception des réponses.

V.5.0.1 Test et analyse des résultats

Cette partie est consacrée à la présentation et la visualisation des résultats fournis par les différents modules de la solution proposée :

1. Lancement de la topologie :

Préalablement on lance le contrôleur RYU avec l'application préexistante par-feu.

```
racim@racim-Inspiron-3542: ~  
Fichier Édition Affichage Rechercher Terminal Aide  
racim@racim-Inspiron-3542:~$ ryu-manager Ryu/ryu/ryu/app/rest_firewall.py  
loading app Ryu/ryu/ryu/app/rest_firewall.py  
loading app ryu.controller.ofp_handler  
instantiating app None of DPSet  
creating context dpset  
creating context wsgi  
instantiating app ryu.controller.ofp_handler of OFPHandler  
instantiating app Ryu/ryu/ryu/app/rest_firewall.py of RestFirewallAPI  
(16874) wsgi starting up on http://0.0.0.0:8080  
[FW][INFO] dpid=0000000000000001: Join as firewall.  
█
```

FIGURE V.5 – Lancement du contrôleur Ryu

Ensuite on crée notre topologie avec Mininet.

```
racim@racim-Inspiron-3542: ~/projet1  
Fichier Édition Affichage Rechercher Terminal Aide  
racim@racim-Inspiron-3542:~/projet1$ sudo python topo.py  
Connecting to remote controller at 127.0.0.1:6653  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3 h4 h5  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)  
*** Configuring hosts  
h1 h2 h3 h4 h5  
*** Running terms on :0  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet> █
```

FIGURE V.6 – Lancement de la topologie

La figure V.6 montre que la topologie a bien été créée. Nous observons la création des différents hôtes, switch et liens, ainsi que l'établissement d'une connexion avec le contrôleur lancé au préalable. Par ailleurs, nous avons activé la fonction Port Mirroring pour rediriger tout les paquets venant des ports 1 jusqu'à 4 vers le port 5 du switch.

2. Installation des rôles :

On lance les serveurs W1 et W2 sur les hôtes h1 et h2 respectivement.

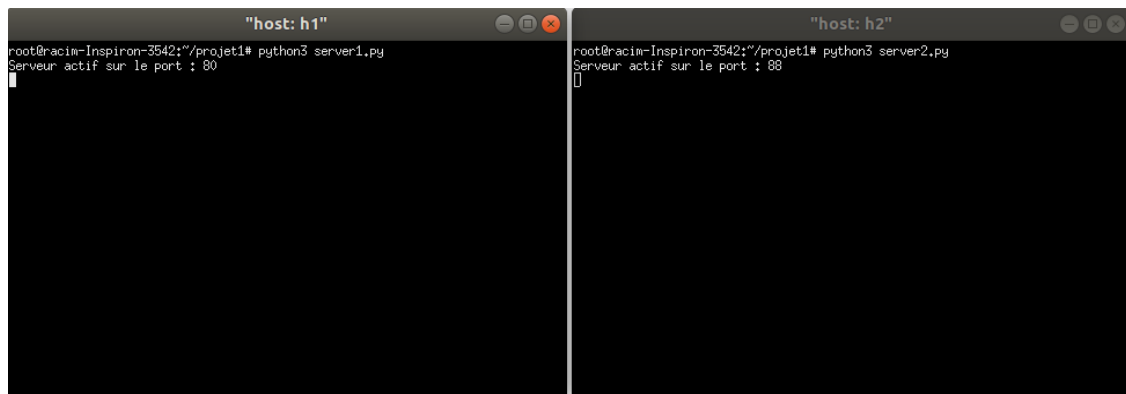


FIGURE V.7 – Lancement des serveurs w1 et W2

On lance le script *détection.py* sur h5 pour installer les modules collection de paquets, extraction d'attributs et classification de flux.

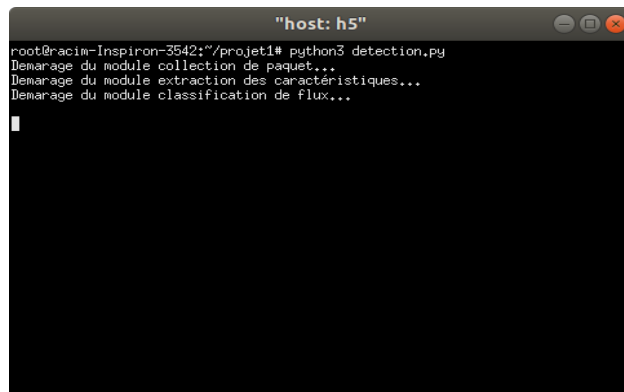
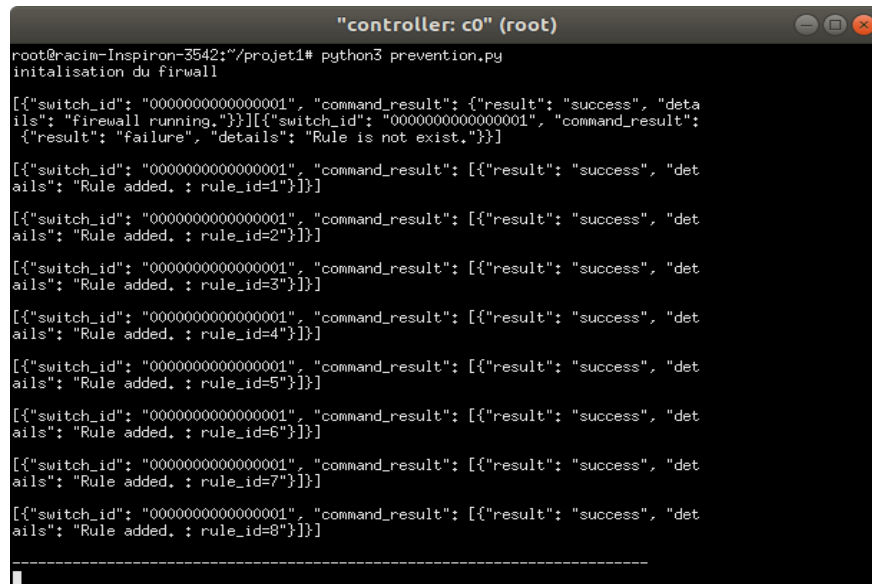


FIGURE V.8 – Lancement de la détection de trafic

Enfin, on lance le script *prévention.py* sur le contrôleur pour installer le module d'atténuation d'attaque.



```

"controller: c0" (root)
root@racim-Inspiron-3542:~/projet1# python3 prevention.py
initialisation du firewall
[{"switch_id": "0000000000000001", "command_result": {"result": "success", "details": "Firewall running."}}][{"switch_id": "0000000000000001", "command_result": {"result": "failure", "details": "Rule is not exist."}}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=1"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=2"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=3"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=4"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=5"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=6"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=7"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=8"}]}]

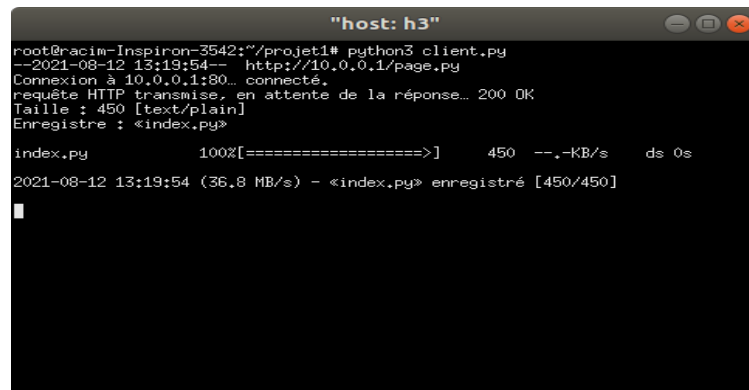
```

FIGURE V.9 – Lancement de la prévention d’attaque

A l’initialisation la figure V.9 indique l’activation du par-feu sur le contrôleur et l’ajout des règles autorisant la communication TCP entre clients et serveurs Web.

3. Génération de trafic légitimes :

Le script *client.py* est exécuté au niveau de l’hôte h3 pour envoyer des requêtes HTTP chaque 10 secondes au serveur web W1.



```

"host: h3"
root@racim-Inspiron-3542:~/projet1# python3 client.py
--2021-08-12 13:19:54-- http://10.0.0.1/page.py
Connexion à 10.0.0.1:80_ connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 450 [text/plain]
Enregistre : <index.py>

index.py          100%[=====]          450 --.-KB/s   ds 0s
2021-08-12 13:19:54 (36.8 MB/s) - <index.py> enregistré [450/450]

```

FIGURE V.10 – Simulation d’un trafic légitime

On observe sur la figure V.10 que la réponse à la première requête HTTP envoyée par l’hôte h3 a été effectué avec succès en téléchargeant le fichier *index.py*.

```

"host: h5"
root@racim-Inspiron-3542:~/projet1# python3 detection.py
Demarage du module collection de paquet...
Demarage du module extraction des caractéristiques...
Demarage du module classification de flux...
trafic legitime...
-----
█

```

FIGURE V.11 – Détection de trafic légitime

La figure V.11 montre q'après l'échange requête/réponse entre le client h3 et le serveur h1, le script *detection.py* a pu classer le trafic capturé comme bénin.

4. Génération de trafic d'attaque :

Pour générer notre attaque DOS lente, nous avons lancer l'outil *Slowhttptest* au niveau de l'hôte h4. Dans notre cas de test on a simulé l'attaque DOS Slowloris, rappelons nous qu'elle s'occupe d'envoyer des requêtes HTTP partielles, en essayant d'obtenir un déni de service du serveur web cible. le serveur W2 qui est la victime reçoit des requêtes HTTP partielles avec un taux de transfert très faible, lui obligeant à ouvrir 750 connexions. Ces connexions ouvertes étant maintenues, occupent les ressources du serveur W2 et provoquent un déni de service.

```

"host: h4"
slowhttptest version 1.6
- https://code.google.com/p/slowhttptest/ -
test type: SLOW HEADERS
number of connections: 750
URL: http://10.0.0.2:88/
verb: GET
Content-Length header value: 4096
Follow up data max size: 52
interval between follow up data: 200 seconds
connections per seconds: 50
probe connection timeout: 2 seconds
test duration: 240 seconds
using proxy: no proxy

Thu Aug 12 15:08:39 2021:
slow HTTP test status on 5th second:
initializing: 0
pending: 1
connected: 0
error: 0
closed: 246
service available: NO
█

```

FIGURE V.12 – Simulation de trafic d'attaque

Suivant les indications de La figure V.12, nous observons que l'attaque en cours d'exécution est bien effectuée car le service W2 n'est pas disponible.


```

"host: h5"
root@racim-Inspiron-3542:~/projet1# python3 detection.py
Demarage du module collection de paquet...
Demarage du module extraction des caractéristiques...
Demarage du module classification de flux...
trafic d'attaque...
La victime est l'hote: h2

```

FIGURE V.13 – Détection de trafic d'attaque

La figure V.13 montre que le trafic envoyé de la source h4 vers la victime W2, pour établir et maintenir des connexions HTTP, est considéré comme un trafic d'attaque.

```

"controller: c0" (root)
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule added. : rule_id=39"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule added. : rule_id=40"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule added. : rule_id=41"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule added. : rule_id=42"}]}]
-----
Desactivation du trafic d'attaque pour la source : 10.0.0.4

```

FIGURE V.14 – Atténuation de trafic d'attaque

La figure V.14 montre que le trafic venant de la source d'attaque h4 dont l'adresse est 10.0.0.04 est bloqué en ajoutant deux nouvelles règles qui le désactive.

```

"controller: c0" (root)
tails": "Rule deleted. : ruleID=44"}]}]
-----
Desactivation du trafic d'attaque pour la source : 10.0.0.4
Ajout de regles:
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule added. : rule_id=45"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule added. : rule_id=46"}]}]
Activation du trafic pour la source : 10.0.0.4
Ajout de regles:
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule deleted. : ruleID=45"}]}]
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "de
tails": "Rule deleted. : ruleID=46"}]}]

```

FIGURE V.15 – Activation de trafic venant de la source d'attaque

Après 10 seconde, depuis que le trafic d'attaque est bloqué, La figure V.15 montre que le trafic venant de la source d'attaque h4, dont l'adresse est 10.0.0.4, est activé en supprimant les règles qui bloquent le trafic.

Dans la prochaine étape, afin de vérifier que le trafic a été bloqué nous avons lancé l'outil *Iptraf* pour capturer tout les paquets TCP qui transitent sur l'interface *s1-eth2* du commutateur causalement lié au serveur web victime W2.

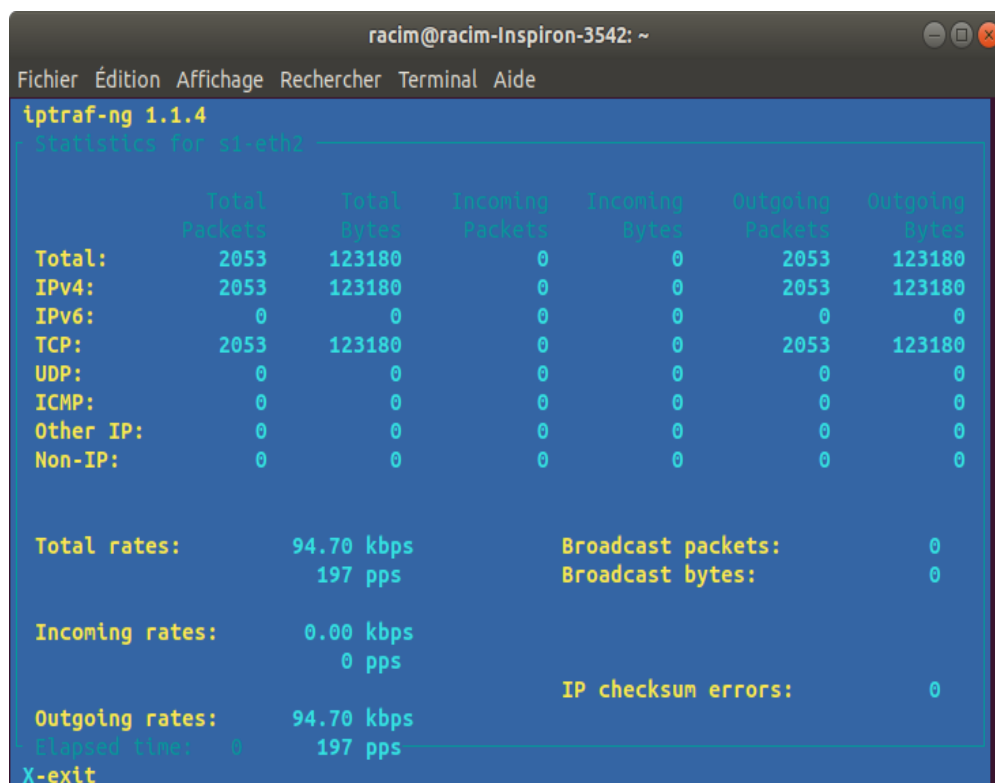


FIGURE V.16 – Statistiques des paquets TCP malveillants avant la détection

Avant la détection d'attaque La figure V.16 indique, à mesure qu'un nombre de paquet malveillant reçu par le serveur w2, le nombre de paquet TCP capturé augmente à partir du début de la prise et le ratio nombre de paquet ou octet par seconde capturé en temps réel devient non nul.

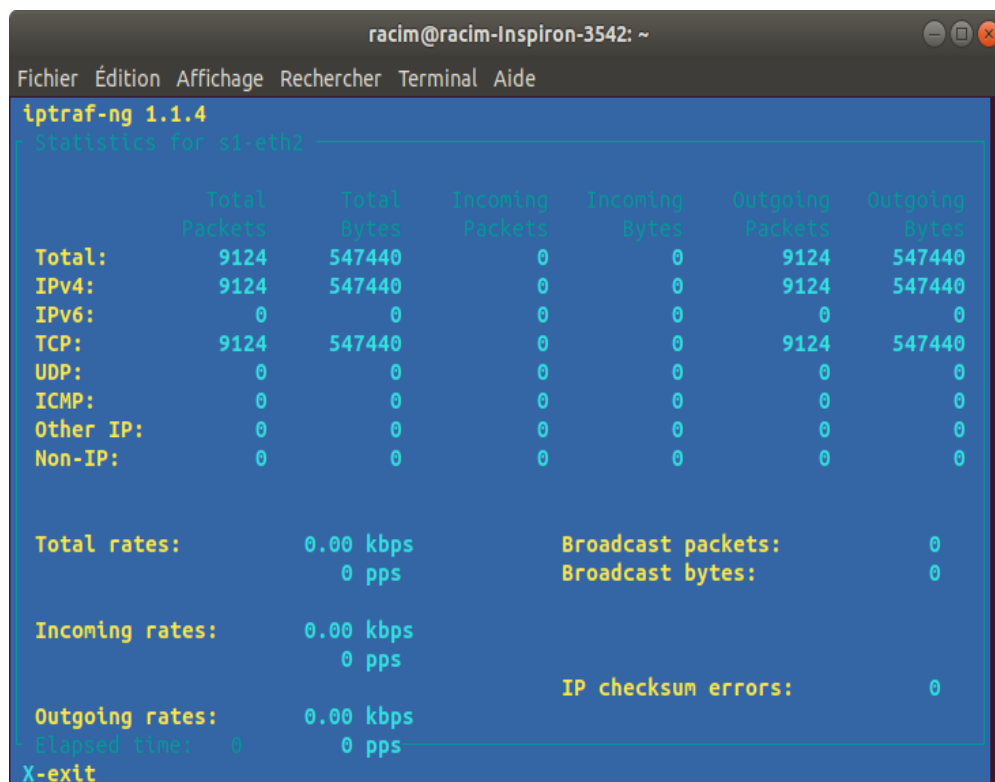


FIGURE V.17 – Statistiques des paquets TCP malveillants après la détection

Après la détection et la prévention d'attaque, suivant la figure V.17, nous observons que le ratio nombre de paquet ou octet par seconde capturé en temps réel est nul.

V.6 Conclusion

Dans ce chapitre, nous avons présenté l'environnement et les outils utilisés pour le développement de notre système de détection et d'atténuation d'attaques DOS lentes. Puis, nous avons décrit, pas à pas, les étapes de la réalisation des différents modules et les tests effectués pour décider de l'algorithme de classification à utiliser dans le module de détection. Avant de terminer par la visualisation du fonctionnement de notre solution proposé à travers un test de simulation.

Conclusion générale

Dans notre projet, l'objectif principal était de fournir de nouvelles idées concernant le manque de solutions sécurisées dans l'environnement SDN et de proposer une solution de détection et de prévention des intrusions dans ce réseau. Dans un premier temps, le travail est concentré sur l'analyse et recherches existantes dans le domaine de SDN, permettant ainsi de comprendre avant tout l'architecture, le protocole OpenFlow, son fonctionnement, le défi de développer des solutions de réseaux définis par logiciel et le grand intérêt qu'elle suscite de la part des chercheurs. Cependant on a constaté que les attaques DOS lentes contre la couche de données ne fait pas l'objet de nombreuses recherches, du coup notre intention c'est orienter vers ce type d'attaques DOS. Dans un second temps, nous nous sommes intéressés aux technologies de détection d'attaques, plus précisément celle de machine learning qui s'avère prometteuse et dans laquelle nous avons assimilé plusieurs notions qui en découlent.

Pour atteindre notre objectif, nous avons combiné la puissance de programmation de SDN avec l'intelligence du ML, afin d'atténuer les attaques DOS lentes dans les réseaux SDN. Pratiquement un modèle d'apprentissage automatique choisi parmi tant d'autres, suite à une comparaison entre eux, est construit grâce à deux dataset combinés CICIDS2017 et CSE-CIC-IDS2018, nous donne une précision de classification satisfaisante de 99,99%, concluant ainsi que les algorithmes de type ensemblistes constituent une solution efficace pour obtenir des prédictions d'une meilleure qualité dû à la combinaison de plusieurs classificateurs en même temps. L'application de cette technologie de machine learning, en tant que IDS, nous a induit après tout, à simuler une topologie SDN et capturer les paquets de données transités dans le réseau, pour les classer comme trafic normal ou d'attaque. Par ailleurs, les applications prédéfinies dans le contrôleur Ryu nous a permis, non seulement, de simuler une topologie mais aussi de parvenir à atténuer le trafic d'attaque grâce à la manipulation simplifiée et la facilité de programmation de ce contrôleur.

L'implémentation de notre approche a été, pour nous, l'occasion d'acquérir, une expérience pratique avec le SDN et énormément de connaissances en apprentissage automatique, qui sont très utiles pour le développement de notre solution de sécurisation des échanges dans un réseau. Au cours de cette étude, nous nous sommes familiarisés avec les techniques d'installation des réseaux SDN en utilisant OpenFlow, qui est bénéfique et favorable à l'amélioration de l'intégration des règles de sécurité de manière automatisée.

Dans la continuité de nos travaux de mémoire de Master, nous proposons les perspectives suivantes :

- ▷ Modifier notre approche de telle sorte que le module de détection envoie des signaux d'alerte d'attaque via l'interface sud de l'architecture SDN.
- ▷ Servir des requêtes légitimes en présence d'attaques.
- ▷ Proposition d'un système hybrides plus sécurisé permettant de traiter d'autres vulnérabilités dans l'environnement SDN.

- ▷ Exploiter d'autres volets de l'apprentissage automatique comme le Deep Learning.
- ▷ Réaliser une plateforme WEB pour avoir une vue globale des événements produits et consulter les statistiques pour comprendre la source des menaces.

Bibliographie

- [1] J. Doherty, *SDN and NFV Simplified*. Addison-Wesley Professional, 2016.
- [2] Wenfeng.X and Yong.W, “A survey on software-defined networking,” *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, 2014.
- [3] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. F. ans D. Lake, J. Finnegan, M. M. N. Viljoen, , and N. Rao, “Openflow inventor martin casado on sdn, vmware, and software defined networking hype,” *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, July 2013.
- [4] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, , and T. Turetletti, “A survey of software-defined networking : Past, present, and future of programmable networks,” *IEEE Communications Surveys Tutorials*, vol. 16, pp. 1617–1634, 2014.
- [5] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, , and W. Kellerer, “Interfaces, attributes, and use cases : A compass for sdn,” *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, June 2014.
- [6] Karmakar.K, Varadharajan.V, and Tupakula, “Mitigating attacks in software defined network (sdn),” in *2017 Fourth International Conference on Software Defined Systems (SDS)*, pp. 112–17, 2017.
- [7] C. Göransson.P, Black.C, *Software Defined Networks A Comprehensive Approach*. Elsevier Science, 2014.
- [8] Mukherjee, Amitava, Saeed.R, S. Dutta, and Naskar.M, *Fault Tracking Framework for Software-Defined Networking (SDN)*. In Resource Allocation in Next-Generation Broadband Wireless Access Networks, 2017.
- [9] O. N. Foundation, “Openflow switch specification v1.3.0.” Juin 2012,[Available online] : <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflowspec-v1.3.0.pdf>>.(consulté le 06 octobre 2021).
- [10] L. Ivan, D. R. Massimo, C. Pasquale, and C. Dajana, “Performance of botnet detection by neural networks in software-defined networks,” in *2nd Italian Conference on Cyber Security, ITASEC 2018*, 2018.
- [11] F. Benamrane, M. B. mamoun, and R. Benaini, “Performances of openflow-based software-defined networks : An overview,” *JOURNAL OF NETWORKS*, vol. 10, no. 6, pp. 329–337, June 2015.
- [12] D. Kreutz, F. M. V. Ramos, P. V., C. E. R. Verissimo, and S. U. Siamak A., “Software defined networking : A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, January 2015.

- [13] D. Kreutz, F. M. V. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks.” in in Proc. 2nd ACM Workshop Hot Topics in Software Defined Networks(HotSDN), 2013.
- [14] Sapna, K. S, and al, “Detection of attacks in an intrusion detection system,” (*IJCSIT*) *International Journal of Computer Science and Information Technologies*, vol. 2, no. 3, pp. 982–986, 2011.
- [15] A. L. Aliyu, P. Bull, and A. Abdallah, “Detection of attacks in an intrusion detection system,” tech. rep., School of Computing, Telecommunications and Networks Faculty of Computing, Engineering and the Built Environment, 2015.
- [16] A. Feghali, R. Kilany, and M. Chamoun, “Sdn security problems and solutions analysis,” in *.International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, pp. 1–5, 2015.
- [17] S. Khan, A. Gani, A. W. A. Wahab, M. Guizani, and M. K. Khan, “Topology discovery in software defined networks : Threats, taxonomy, and state-of-the-art,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 303–324, 2017.
- [18] Z. Shu, J.Wan, D. Li, J. Lin, A. V. Vasilakos, and M. Imran, “Security in software-defined networking : Threats and countermeasures,” in *journal Mobile Networks and Applications*, vol. 21, no. 5, pp. 764–776, 2017.
- [19] ulshan Kumar, *Denial of service attacks – an updated perspective*. Systems Science Control Engineering, 2016.
- [20] Z. ST, J. J, and T. D, “A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks,” *IEEE Commun Surv Tut*, 2013.
- [21] R. Swam, M. Dave, and V. Ranga, “Software defined networking based ddos defense mechanisms,” *Journal of ACM Compute .surv*, vol. 52, pp. 28–36, 2019.
- [22] W. Weber, “Firewall basics,” in *4th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, pp. 300–301, 1999.
- [23] E. Jebri, *Introduction à la sécurité*. support de cours, 2008.
- [24] H. Schauer, *Introduction à la cryptographie*. HSC Hervé Schauer Consultants, 2001.
- [25] R. Fisli, *Secure Corporate Communications over VPN-Based WANs*. PhD thesis, Royal Institute of Technology sweden, 2005.
- [26] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, “ombining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments,” *Comput. NETWORKS*, vol. 62, p. 122–136, 2013.
- [27] Y. Xu and Y. Liu, “Ddos attack detection under sdn context,” pp. 1–9, 2016.
- [28] S. R, S. D, S. W, R. A, and M. E, “achine learning algorithms to detect ddos attacks in sdn. concurrency computat pract exper,” *PROCC, Federal University of Sergipe, Brazil*, 2019.
- [29] E. Schechter, J. Jung, , and A. W. Berger, “Fast detection of scanning worm infections,” vol. 3224, 2004.

- [30] M. M. Williamson, H. P. L. Bristol, F. Road, and S. Gifford, "Throttling viruses : Restricting propagation to defeat malicious mobile code," p. 51–68, 2002.
- [31] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," vol. 35, no. 4, pp. 217–228, 2005.
- [32] Z. ISMAILI, "Apprentissage supervisé vs. non supervisé," 2019. <https://analyticsinsights.io/apprentissage-supervise-vs-non-supervise>, consulté le :2021-07-29.
- [33] A. Dey, "Machine learning algorithms : A review," *Sci. Inf. Technol.*, vol. 7, no. 3, pp. 1174–1179, 2016.
- [34] K. L. K. K. Mak and C. Park, "Applications of machine learning in addiction studies : A systematic review," *Psychiatry Res.*, vol. 275, pp. 53–60, May 2019.
- [35] G. N. T. Sakai, M. C. Plessis and M. Sugiyama, "Semi-supervised classification based on classification from positive and unlabeled data," 2016. [online] Available : <http://arxiv.org/abs/1605.06955>, consulté le :2021-06-29.
- [36] A. Mehnaz and K. S. Shreedhara, "Performance analysis of semi-supervised machine learning approach for ddos detection," *Int. J. Innov. Res. Technol.*, vol. 6, no. 2, pp. 3193–3208, Jul 2019.
- [37] S. B. et C. Benavent, "Les techniques du nlp pour la recherche en sciences de gestion." 2019.
- [38] L. Breiman, "Bagging predictors. machine learning," vol. 24, no. 2, pp. 123–140, 1996.
- [39] "Difference between bagging and boosting| ensemble method." [Available online] : <https://buggyprogrammer.com/difference-between-bagging-and-boosting/> (consulté le 25 septembre 2021).
- [40] Y. Freund and R. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Science*, vol. 55, pp. 119–139, 1996.
- [41] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.
- [42] D. X. Dua. S, *Data Mining and Machine Learning in Cybersecurity*. Auerbach Publications, 2011.
- [43] B. L., "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [44] Y. F. et Robert E. Schapire, "A short introduction to boosting," *Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.
- [45] a. A. Almomani, "An online intrusion detection system to cloud computing based on neucube algorithms," *Int. J. Cloud Appl. Comput.*, vol. 8, no. 2, pp. 96–112, 2018.
- [46] Y. S. S.M. Kasongo, "A deep long short-term memory based classifier for wireless intrusion detection system," *ICT Express*, vol. 6, no. 2, pp. 98–103, 2020.

- [47] Y. S. N. Tripathi, N. Hubballi, “How secure are web servers ? an empirical study of slow http dos attacks and detection for wireless intrusion detection system,” *11th International Conference on, in : Availability, Reliability and Security (ARES), IEEE*, pp. 454–463, 2016.
- [48] a. H.H. Jazi, “Http-based application layer dos attacks on web servers in the presence of sampling,” *Computer Networks*, vol. 121, pp. 25–36, 2017.
- [49] N. H. N. Tripathi, “Slow rate denial of service attacks against http/2 and detection,” *Comput. Secur*, vol. 72, pp. 255–272, 2018.
- [50] M. A. E. Cambiaso, G. Papaleo, “Taxonomy of slow dos attacks to web applications,” *International Conference on Security in Computer Networks and Distributed Systems, Springer, Berlin, Heidelberg*, pp. 195–204, 2012.
- [51] a. M.M. Najafabad, “Rudy attack : Detection at the network level and its important features,” *FLAIRS Conference*, pp. 288–293, 2016.
- [52] H. T. T. K. J. Park, K. Iwai, “Analysis of slow read dos attack and countermeasures,” *The International Conference on Cyber-Crime Investigation and Cyber Security (IC-CICS2014)*, pp. 37–49, 2014.
- [53] M. Z. S. Shafieian and A. Haque, “Cloudzombie : Launching and detecting slow-read distributed denial of service attacks from the cloud,” *2015 IEEE International Conference on Computer and Information Technology*, pp. 1733–1740, 2015.
- [54] B. B. B. T. Hirakawa, K. Ogura and T. Takata, “A defense method against distributed slow http dos attack,” *2016 19th International Conference on Network-Based Information Systems (NBIS)*, pp. 152–158, 2016.
- [55] N. H. N. Tripathi and Y. Singh, “How secure are web servers ? an empirical study of slow http dos attacks and detection,,” *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pp. 454–463, 2016.
- [56] K. H. et al., ““sdn-assisted slow http ddos attack defense method,,” *IEEE Communications Letters*, vol. 22, p. 688–691, April 2018.
- [57] D. A. Ph.D and P. Nithyanandam, “The slow http distributed denial of service attack detection in cloud,” *calable Computing : Practice and Experience*, 2019.
- [58] K. C. K. T. . N. M. Calvert, C., “A detecting slow http post dos attacks using netflow features,” *FLAIRS Conference*, 2019.
- [59] Sumantra and S. I. Gandhi, “Ddos attack detection and mitigation in software defined networks,,” *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pp. 1–5, 2020.
- [60] C. C. C. Kemp and T. M. Khoshgoftaar, “Detection methods of slow read dos using full packet capture data,,” *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, pp. 9–16, 2020.
- [61] K. K. R. C. J. A. Pérez-Díaz, I. A. Valdovinos and D. Zhu, “A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning,” *in IEEE Access*, vol. 8, pp. 155859–155872, 2020.

- [62] J. V. Phan and M. Park, “Efficient distributed denial-of-service attack defense in sdn-based cloud,” *sdn-based cloud, IEEE Access*, vol. 7, pp. 18701–18714, 2019.
- [63] Z. G. Y. Gu, K. Li and Y. Wang, “Semi-supervised k-means ddos detection method using hybrid feature selection algorithm,” *IEEE Access*, vol. 7, pp. 64351–64365, 2019.
- [64] I. Sharafaldin., A. Habibi Lashkari., and A. A. Ghorbani., “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*,, pp. 108–116, INSTICC, SciTePress, 2018.
- [65] “A collaborative project between the communications security establishment (cse) the canadian institute for cybersecurity (cic).,” 2018. [Available online] : <https://www.unb.ca/cic/datasets/ids-2018.html> (consulte le 13 juin 2021).
- [66] “Ryu controller.” [Available online] : <https://ryu-sdn.org/> (consulte le 13 juillet 2021).
- [67] “Mininet : An instant virtual network on your laptop (or other pc) – mininet.” [Available online] : <http://mininet.org/> (consulte le 13 juillet 2021).
- [68] (s. d.), “What is python? executivesummary.” [Available online] : <https://www.python.org/doc/essays/blurb/> (consulte le 13 juillet 2021).
- [69] A. Vidhya, “Sklearn | scikit-learn in python.,” 2015. [Available online] : <https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/> (consulte le 13 juillet 2021).
- [70] GeeksforGeeks, “Python numpy.,” 2015. [Available online] : <https://www.geeksforgeeks.org/python-numpy/> (consulte le 15 juillet 2021).
- [71] “getting started with pandas.” [Available online] : https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html.(consulté le 05août 2021).
- [72] GeeksforGeeks, “What is matplotlib ? (s. d.). educative : Interactive courses for software developers.” [Available online] : <https://www.educative.io/edpresso/whatismatplotlib> (consulté le 15 juillet 2021).
- [73] “Pyshark, project website.” [Available online] : <http://kiminewt.github.io/pyshark/> (consulté le 02 août 2021).