RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPERIEURE
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ SAAD DAHLEB BLIDA

FACULTÉ DES SCIENCES

DÉPARTEMENT INFORMATIQUE

MÈMOIRE DE FIN D'ÉTUDES

Pour l'obtention

D'un diplôme Master en informatique

Spécialité : Systèmes Informatiques et Réseaux

THÈME

Système de détection des intrusions avec l'apprentissage automatique sur SDN

Réalisé par :

BENHAMADI Mohamed Nadhir, HANOUNE Mohamed

Promoteur :

M. OULD KHAOUA Mohamed, USDB

Encadreur :

M. TAKHERIST Ayoub, SONATRACH

Jury :

Président :

Mme. GHEBGHOUB Yasmine, USDB

Examinateur :

Mme. MIDOUN Khadidja, USDB

Promotion : 2023/2024

Remerciements

Nous tenons tout d'abord à remercier Allah le tout puissant et miséricordieux qui nous a donné la force et la patience d'accomplir ce modeste travail.

En second lieu, nous tenons à remercier très chaleureusement notre encadreur, Monsieur ***Takherist Ayoub*** *pour ses aides précieuses, pour sa spontanéité et ses compétences professionnelles incontestables, durant toute cette période d'encadrement.*

Nous remercions également notre promoteur, Monsieur ***OULD KHAOUA Mohamed*** *pour la confiance placée en nous et pour avoir accepté de diriger ce travail ainsi que pour ses conseils et son assistance inestimables tout au long de la période de travail.*

Aussi, nos vifs remerciements aux membres du jury pour l'intérêt accordé à notre travail en l'examinant minutieusement et avec attention.

Nous tenons à exprimer nos sincères remerciements à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu pour la réussite dans nos études.

À nos familles et nos amis qui, par leurs prières et leurs encouragements, on a pu surmonter tous les obstacles.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

الملخص

كنموذج شبكة ناشئ تم تطويره لتقليل تعقيد الشبكات، أصبحت الشبكات المعرفة بالبرمجيات (SDN) تُنفذ على نطاق واسع من خلال فصل الطبقة التحكمية عن الطبقة التنفيذية، مما يوفر إدارة مركزية وبرمجة مرنة. ومع ذلك، تُقدم هذه البنية تحديات أمنية جديدة، خصوصًا في اكتشاف التطفل والتخفيف من حدته بشكل فعال. باستخدام تقنيات التعلم الآلي، وخاصة خوارزمية الغابة العشوائية RFومجموعة بيانات-IN SDNالتي تحاكي حركة المرور الحقيقية فيSDN ، قمنا بتطوير نموذج قادر على تحديد التطفليات الشبكية بدقة مع تقليل الإنذارات الكاذبة فيSDN .

هذا يعزز فهمنا لأنظمة اكتشاف التطفل (IDS) ضمن الشبكات المعرفة بالبرمجيات(SDN) ، مسلطًا الضوء على إمكانيات التعلم الآلي في معالجة تحديات الأمان.

كلمات أساسية: نظام اكتشاف التطفل(IDS) ، الشبكات المعرفة بالبرمجيات(SDN) ، الأمان، التعلم الآلي

Résumé

En tant que paradigme de réseau émergent développé pour réduire la complexité des réseaux, les réseaux définis par logiciel (SDN) sont largement implémentés en séparant le plan de contrôle du plan de données, offrant ainsi une gestion centralisée et une programmabilité flexible. Cependant, cette architecture introduit de nouveaux défis en matière de sécurité, en particulier dans la détection et l'atténuation des intrusions de manière efficace. En utilisant des techniques d'apprentissage automatique, notamment l'algorithme de la forêt aléatoire et le jeu de données IN-SDN qui simule le trafic SDN du monde réel, nous avons développé un modèle capable d'identifier avec précision les intrusions réseau tout en minimisant les fausses alertes dans les SDN.Cela améliore notre compréhension des systèmes de détection d'intrusion (IDS) au sein des réseaux définis par logiciel (SDN), mettant en évidence le potentiel de l'apprentissage automatique pour relever les défis de sécurité.

Mots-clés :
Système de détection d'intrusion (IDS), réseau défini par logiciel (SDN), sécurité, apprentissage automatique.

Abstract

As an emerged network paradigm that was developed to reduce network complexity, Software-defined networks (SDN) became widely implemented by separating the control plane from the data plane, offering centralized management and programmability. However, this architecture introduces new security challenges, particularly in detecting and mitigating intrusions effectively. Using machine learning techniques specifically the Random Forest algorithm and the IN-SDN dataset, which simulates real-world SDN traffic, we developed a model capable of accurately identifying network intrusions while minimizing false alarms in SDN. This enhances our understanding of Intrusion Detection Systems (IDS) within Software Defined Networks (SDN), highlighting the potential of machine learning to address security challenges.

Keywords :
Intrusion Detection System (IDS), Software Defined Network (SDN), Security, Machine Learning.

Table des Matières

Liste des Figures

Liste des Tables

Liste des Abréviations

SDN : Software-defined network

ML : Machine Learning

DL : Deep Learning

DOS : Denial of Service

DDOS : Distributed Denial of Service

IDS : Intrusion Detection System

API : Application Programming Interface

U2R : User to Root

VM : Virtual Machine

HIDS : Host Intrusion Detection system

NIDS : Network Intrusion Detection system

OSI : Open Systems Interaction

TCP/IP : Transmission Control Protocol/Internet Protocol

SYN/ACK : Synchronization / Acknowledgment

UDP : User Datagram Protocol

FE : Forwarding Element

HTTP : HyperText Transfer Protocol

RF : Random Forest

SVM : Support Vector Machine

OFP : Open Flow Protocol

FS : Feature Selection

ANN : Artificial Neural Network

OSI : Open Systems Interaction

AI : Artificial Intelligence

Introduction Générale

Contexte du Travail

Un réseau défini par logiciel (SDN) est un nouveau paradigme qui a vu le jour ces dernières années pour résoudre les problèmes et les limites du réseau conventionnel. Contrairement au réseau traditionnel qui implémente le trafic réseau et configure les politiques de trafic sur chaque appareil indépendamment telles que le routage, la commutation et la qualité de service, le SDN sépare le plan de contrôle et le plan de données afin que la gestion du réseau soit effectuée par le contrôleur central qui a la capacité de contrôler et d'appliquer des politiques réseau à l'ensemble du réseau à partir d'un seul point. Le découplage des plans de données et de contrôle a permis au SDN de rendre le réseau plus flexible et plus facile à gérer, grâce au contrôleur centralisé qui constitue un avantage clé. Cette nature flexible permet d'améliorer les mesures de sécurité telles que la détection et la prévention des menaces, car elle accélère la recherche d'innovation par rapport aux réseaux conventionnels. Cependant, malgré tous les avantages que le SDN peut offrir, comme tout autre réseau, il est sujet à plusieurs problèmes de sécurité et exposé à des attaques spécifiques, le déploiement de techniques IDS est considéré comme un élément essentiel afin de détecter les intrusions malveillantes dans le trafic réseau SDN. Par conséquent, avec les récentes avancées dans le domaine de l'intelligence artificielle et de l'apprentissage automatique, nous assistons à de plus en plus de recherches sur la sécurité SDN en utilisant techniques d'apprentissage automatique pour la détection et la prévention des attaques.

Objectifs du Travail

L'entreprise SONATRACH a identifié la nécessité d'améliorer la gestion de son système réseau en adoptant une approche centralisée open source. Dans cette optique, elle a choisi d'adopter une architecture SDN, une alternative moderne, économique, flexible et plus facile à administrer que l'architecture traditionnelle. SONATRACH accorde une grande importance à la

sécurité de son réseau. Alors l'objectif de notre projet est l'intégration de système de détection et prévention d'intrusion (IDS) avec l'utilisation des techniques d'apprentissage automatique (ML) pour le SDN afin d'assurer et de renforcer la sécurité globale de SONATRACH, Ils agissent comme une couche de défense supplémentaire en contrôlant et en filtrant le trafic réseau avec des techniques d'apprentissage automatique (ML), ce qui permet de protéger efficacement les infrastructures contre les attaques potentielles et nous avons suivi les étapes suivantes :

1. Création de modèle à l'aide des techniques d'apprentissage automatique
2. Création de l'application de contrôleur SDN
3. L'importation de modèle ML dans l'application de contrôleur
4. Simulation en temps réel à l'aide de simulateur Mininet
5. Test et validation de système de détection et de prévention des intrusions

Organisation du Mémoire

Ce travail est structuré en cinq chapitres :

• **Chapitre 1** : Présente la revue de littérature du software Defined Network.

• **Chapitre 2** : Présente une vue d'ensemble sur l'apprentissage automatique

• **Chapitre 3** : Présente une vue sur le système de détection d'intrusion traditionnel et avec l'apprentissage automatique

• **Chapitre 4** : Présente les différents travaux connexes avec une comparaison entre les études.

• **Chapitre 5** : Présente les phases de conceptions et l'implémentation de notre système de détection et prévention d'intrusion avec les tests en temps réel

Nous terminerons par une conclusion et quelques perspectives pour des travaux futurs.

Chapitre 1

Réseau Défini par Logiciel

1.1 Introduction

Il y a un nouveau paradigme de réseau appelé SDN, qui apporte de nouvelles techniques pour améliorer divers aspects de la gestion et de la configuration des réseaux. Le SDN est une idée qui sépare le plan de données du plan de contrôle en faisant des Switchs des composants de commutation de paquet dont l'intelligence a été extraite pour devenir un programme placé dans une entité réseau appelée contrôleur. Dans ce chapitre, nous allons décrire la technologie SDN, en identifiant ses différents modèles. Ensuite, nous allons présenter son architecture, ses avantages et ses domaines d'applications.

1.2 Définition du SDN

Le SDN (Software Defined Networking) ; En français, le réseau défini par logiciel, est un modèle d'architecture réseau qui permet aux administrateurs réseaux de gérer les services de réseaux par abstraction de fonctionnalités. Consiste à voir le SDN comme une architecture qui découplait les fonctions de contrôle et de transfert des données du réseau afin d'avoir une infrastructure physique complètement exempte de tout service réseau. Dans ce modèle, les équipements réseau se contentent d'implémenter des règles, injectées par les applications, de traitement des flux de données. Une entité intelligente, appelée « contrôleur » voit le réseau dans sa globalité et injecte directement les règles de traitement des données sur chaque équipement du réseau [15].

1.3 Comparaison entre SDN et Réseaux Traditionnels

La structure de l'internet et les réseaux informatiques se composent généralement de différents dispositifs de réseau tels que routeur, commutateur et différents types de middlebox qui sont intégrés verticalement et conçus par des puces et ASIC (circuits intégrés spécifiques à l'application) avec un débit élevé et une fonction spécifique [27] Pour la gestion et la configuration de ces périphériques réseau, un ensemble de commandes de ligne spécifiques et prédéfinies basées sur un système d'exploitation intégré est utilisé. Par conséquent, on peut faire valoir que la gestion d'un grand nombre de périphériques réseau est un grand défi qui est sujet à de nombreuses erreurs. Ainsi, les réseaux traditionnels souffrent d'importantes lacunes en matière de recherche et d'innovation, la fiabilité, l'extensibilité, la flexibilité et gestion. Depuis la naissance de l'internet, les réseaux se développent et de nouvelles technologies telles que le cloud, les réseaux sociaux et la virtualisation ont vu le jour, le besoin de réseaux avec une bande passante plus large, une plus grande accessibilité et une gestion dynamique plus élevée est devenu un problème critique [13].Pour résoudre les problèmes et les limites des réseaux traditionnels, une structure a été proposée, connue sous le nom de SDN, où le contrôle du réseau est séparé du mécanisme de transmission et peut être programmé et contrôlé directement [27] La Figure 1.1

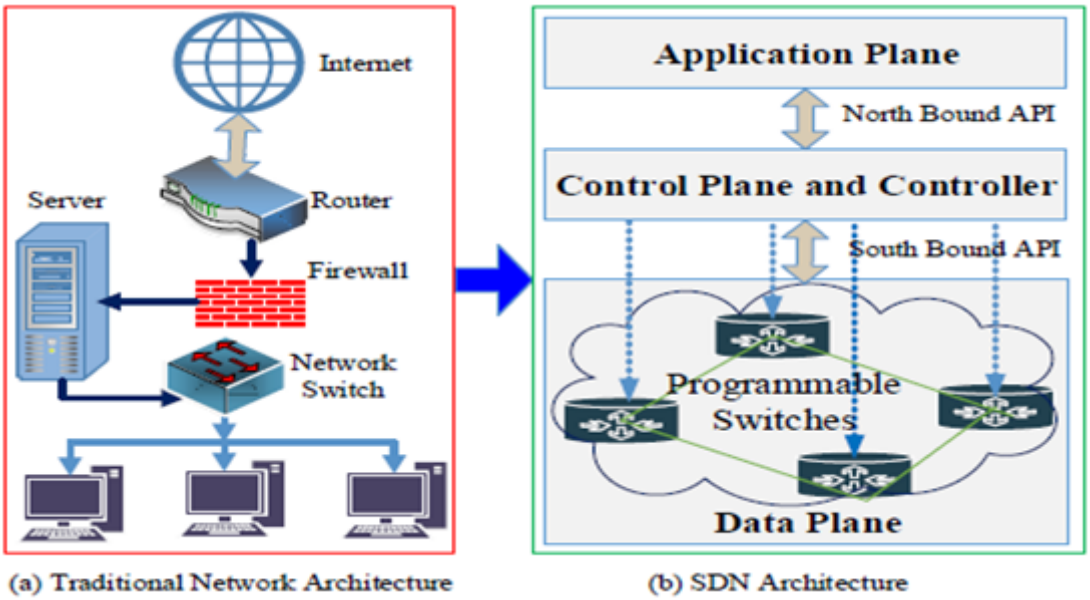


FIGURE 1.1 – Comparaison entre l'architecture des réseaux traditionnels (a) et celle de SDN (b) [33]

illustre les différences architecturales entre l'Internet traditionnel et le SDN. Il montre clairement comment le contrôle est centralisé logiquement et comment le plan de données est simplifié en simples éléments de transfert. Les commutateurs programmables du plan de données peuvent être utilisés dans le matériel ou le logiciel à condition qu'ils soient capables de communiquer et de configurer le protocole OpenFlow avec le contrôleur.

Caractéristiques	SDN	Réseau Traditionnel
Contrôle du réseau Centralisé	OUI	NON
La programmabilité	OUI	NON
La flexibilité du réseau	OUI	NON
Réseau de contrôle complexe	NON	OUI
Amélioration de la performance	OUI	NON
Configuration d'erreur	NON	OUI
Gestion améliorée	OUI	NON
Efficacité de la configuration	OUI	NON
Facile à utiliser et à mettre en œuvre	OUI	NON

TABLE 1.1 – Tableau de comparaison entre l'architecture des réseaux traditionnels et celle de SDN

1.4 Architecture du SDN

l'architecture de « réseaux défini par logiciel », ou SDN, a vu le jour. Elle est composée de trois couches principales et d'interfaces de communication (Figure 1.2). Nous décrivons dans la suite ces couches ainsi que les interfaces de communication entre les couches. La couche la plus basse est la couche de transmission, aussi appelée « plan de données ». Elle contient les équipements de transmission (FEs : Forwarding Element) tels que les switchs physiques et virtuels. Son rôle principal est de transmettre les données, surveiller les informations locales et collecter les statistiques. la couche de contrôle, également appelée « plan de contrôle », est constituée d'un ou de plusieurs logiciels de contrôle (Contrôleurs), ces contrôleurs utilisent des interfaces sud ouvertes pour contrôler le comportement des FEs et communiquent via des APIs nord avec la couche supérieure pour superviser et gérer le réseau. la couche d'application (la plus haute dans la figure) héberge les applications qui peuvent introduire de nouvelles fonctionnalités réseau, comme la sécurité, la configuration dynamique et la gestion. La couche d'application exploite la vue globale et distante du réseau offerte par le plan de contrôle pour fournir des directives appropriées à la couche de contrôle. Trois types d'interfaces permettent aux contrôleurs de communiquer avec leur environnement : interface, sud, nord et est/ouest.[18]

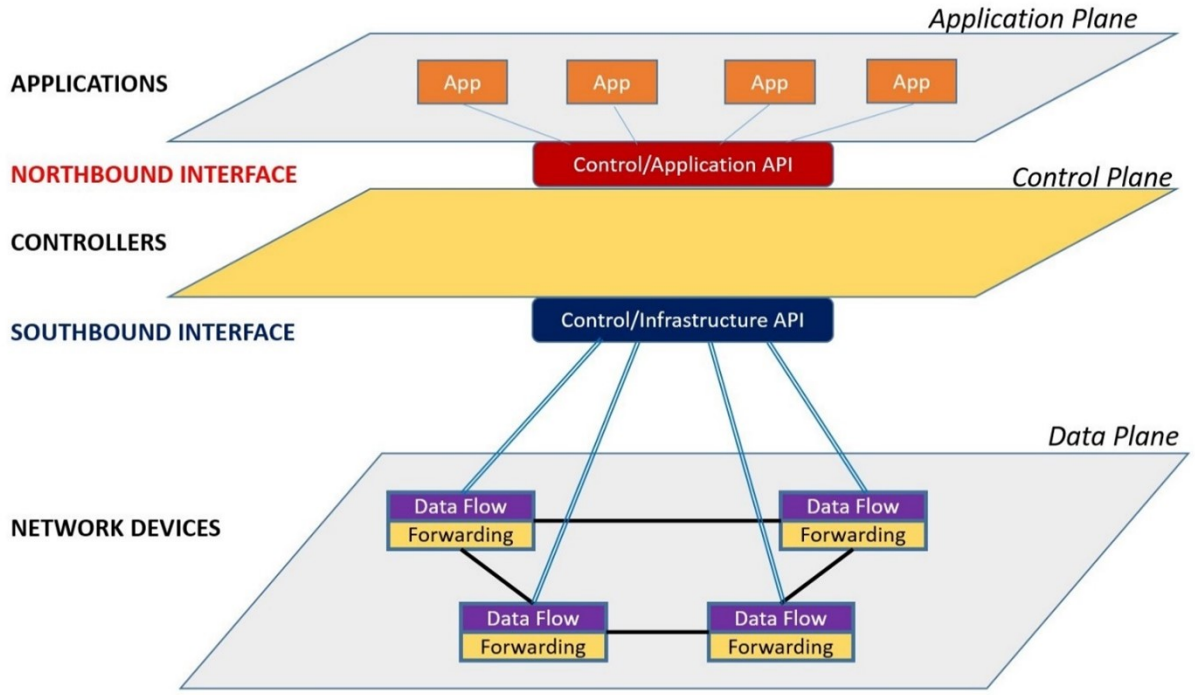


FIGURE 1.2 – Architecture bref du SDN [21]

1.4.1 Plan de Transmission (Data Plane)

C'est la couche inférieure. Elle est utilisée pour afficher les dispositifs de commutation, tels que les commutateurs, les routeurs, dans le plan de donnée. Ces appareils remplissent deux fonctions distinctes. Pour commencer, ils sont chargés de collecter les statuts du réseau, de les stocker temporairement dans les dispositifs locaux et de les transmettre au contrôleur. Les statuts du réseau peuvent inclure des informations telles que la topologie du réseau, la statistique du trafic et l'utilisation du réseau. Deuxièmement, selon les règles fournies par le contrôleur, ils sont responsables du traitement des paquets.

1.4.2 Plan de Contrôle (Contrôle Plane)

Il relie la couche infrastructure à la couche application via ses deux interfaces (southbound interface et northbound interface). En traduisant les requêtes des applications en langage compréhensible par les composants d'infrastructure réseau et inversement, en remontant aux applications SDN les événements détectés par les composants actifs, le plan de contrôle est chargé d'indiquer au plan de transmission comment traiter les paquets du réseau.

1.4.3 Plan d'Application (Application Plane)

représente les applications qui permettent de déployer de nouvelles fonctionnalités réseau, comme l'ingénierie de trafic, QoS, la sécurité, etc. Ces applications sont construites moyennant une interface de programmation appelée north-bound API [13] Les fonctionnalités du plan de gestion sont généralement initiées sur la base d'une vue globale du réseau, et sont traditionnellement centrées sur l'homme. Cependant, ces derniers temps, les algorithmes remplacent la plupart des interventions humaines. Les fonctionnalités du plan de gestion comprennent généralement les éléments suivants :

— La gestion de la configuration.

— La gestion des pannes et de la surveillance.

1.4.4 Les Interfaces de Communications

Il existe principalement trois types d'interfaces permettent aux contrôleurs de communiquer avec leur environnement : interface sud, nord et est/ouest.

— **Interface Nord**

  — Les API nord sont utilisées par la couche applicative pour communiquer avec le contrôleur. Elles constituent la partie la plus critique de l'architecture du contrôleur SDN, servant à programmer les équipements de transmission en exploitant l'abstraction du réseau fourni par le plan de contrôle. L'avantage le plus précieux du SDN provient de sa capacité à prendre en charge et à permettre des applications innovantes.

— **Interface Sud**

  — Les interfaces de communication, également connues sous le nom d'interfaces sud, permettent au contrôleur SDN d'interagir avec les équipements de la couche d'infrastructure, tels que les interrupteurs et les routeurs. Le protocole OpenFlow, standardisé par l'ONF, est le protocole le plus utilisé et le plus déployé comme interface sud. La version la plus récente du protocole est 1.5, et plus d'informations sur ce protocole sont fournies dans la section suivante. Le protocole OpenFlow est actuellement le standard de facto, largement accepté et répandu dans les réseaux SDN, bien qu'il existe plusieurs alternatives d'interface sud, telles que Forces ou OpenVswitch, Data base (OVSDB).

— **Interfaces Est/Ouest**

  — Les interfaces Est/Ouest sont des interfaces de communication qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs, pour synchroniser l'état du réseau. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible[13].

1.5 Protocole OpenFlow dans l'Architecture SDN

OpenFlow c'est le protocole qui lie le plan de contrôle avec le plan de données. L'échange des messages se fait au cours d'une session TCP (Transmission Contrôle Protocole) établie via le port 6633 du serveur contrôleur. Openflow est une composante du SDN, délivré par l'université de Stanford et l'université de Californie à Berkeley, un Switch OpenFlow contient une ou plusieurs tables de flux (flow-table). Lorsqu'un paquet parvient aux Switch, les valeurs contenues dans ses en-têtes sont comparés aux différentes règles enregistrées dans la table de flux de Switch [12] OpenFlow a été créé comme un projet à l'université de Stanford lorsqu'un groupe de chercheurs cherchait à tester de nouveaux protocoles dans le monde IP (créer un réseau expérimental avec le réseau de production) sans arrêter le trafic du réseau de production pendant les tests. Les chercheurs de Stanford ont trouvé un moyen de séparer le trafic de recherche du trafic du réseau de production qui utilise le même réseau IP dans cet environnement. Le résultat de l'équipe de recherche à Stanford a été OpenFlow, qui fournit un protocole ouvert (openprotocol) qui permet aux administrateurs de réseau de programmer les tables de flux (flow tables) dans leurs différents routeurs IP et commutateurs Ethernet dans un but (dans le cas de Stanford) de séparer le trafic de recherche du trafic de production, chacun avec son ensemble de fonctionnalités et caractéristiques de flux.[6]

1.6 Différents Modèles pour le SDN

Afin de faciliter l'interaction avec les applications, le SDN comprend toutes les solutions de programmation du réseau. Il existe une variété de solutions qui peuvent être utilisées en fonction des besoins des utilisateurs. On comprend que les solutions pour simplifier le déploiement d'une application dans un centre de données ne sont pas les mêmes que celles pour mieux contrôler l'éclairage d'une ville à travers le réseau. Divers modèles de SDN peuvent être observés.

1.6.1 Programmabilité Individuelle de chaque Équipement

Dans ce modèle, une application interagit directement avec chaque équipement via des API. L'application est centralisée ou peut être localisée directement sur l'équipement réseau pour réaliser des tâches spécifiques.[15]

1.6.2 Programmabilité via un Contrôleur

Dans ce modèle, une application donne un ordre abstrait et global à un contrôleur, qui à son tour traduit cette requête en une suite d'ordres auprès des équipements du réseau concerné. Ce modèle est certainement le plus populaire puisqu'il permet de simplifier le réseau. Le contrôleur masque la complexité du réseau. On peut distinguer plusieurs cas selon le type d'ordres échangés entre le contrôleur et les équipements. Si, au départ, il était question d'avoir une programmabilité

du plan de données (via Openflow par exemple), des modèles plus récents implémentent des modèles dans lesquels des ordres plus abstraits sont donnés aux équipements, ces derniers restant libres de les implémenter au mieux. On parle dans ce cas d'un modèle « policy-intent ». La suite de l'article reviendra sur ces différentes alternatives.[15]

1.6.3 Création d'un Réseau Virtuel au-dessus du Réseau Physique

Dans ce modèle, les applications créent leur propre réseau « overlay », s'affranchissant des contraintes du réseau physique sous-jacent. Ce dernier n'a pour mission que la simple connectivité entre les nœuds d'extrémité des tunnels et le réseau d'overlay assure l'intégralité des services. On parle également de virtualisation des fonctions réseau (NFV – Network Function Virtualization) quand les routeurs, commutateurs, firewalls, etc. sont des éléments virtualisés sur des serveurs.[15]

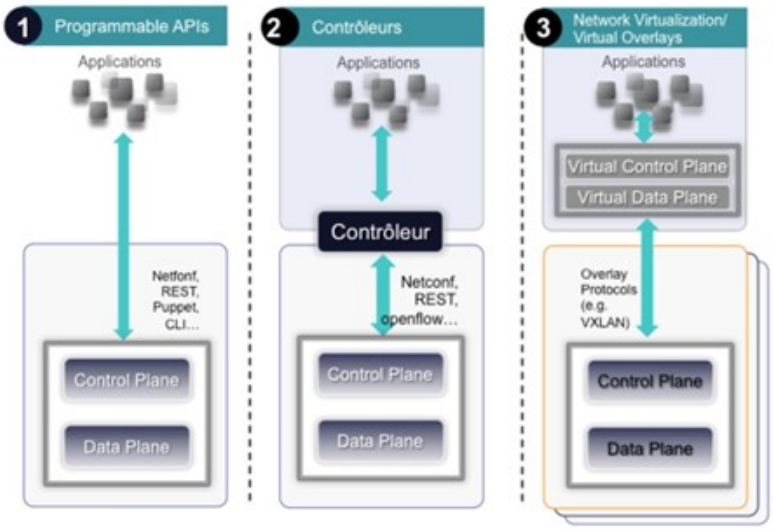


FIGURE 1.3 – Les principaux modèles du SDN [15]

1.7 Liste des Contrôleurs SDN

Il existe plusieurs contrôleurs SDN, tel que :

1.7.1 POX

POX est un contrôleur SDN qui permet le développement et le prototypage rapides du réseau. Il suit le protocole OpenFlow, et qui sert a joué le rôle d'un framework entre les commutateurs OpenFlow.[23]

1.7.2 Floodlight

Floodlight est un contrôleur OpenFlow basé sur Java, pris en charge par BigSwitch Networks, il est sous licence Apache. C'est un contrôleur qui est facile à configurer. Avec ses fonctionnalités, Floodlight est considéré comme une solution complète[1]

1.7.3 OpenDaylight

OpenDaylight est un projet de la fondation linux pris en charge par l'industrie. C'est un framework de source ouverte (open-source). Comme Floodlight, il peut également être considéré comme une solution complète[3]

1.7.4 Ryu

RYU est un contrôleur SDN appelé aussi un 'Framework' SDN fournissant des composants qui facilitent la gestion des réseaux et la création des applications de contrôle. RYU est basé sur Python et supporte la majorité des versions d'OpenFlow **(voir la figure 1.4)**[2]

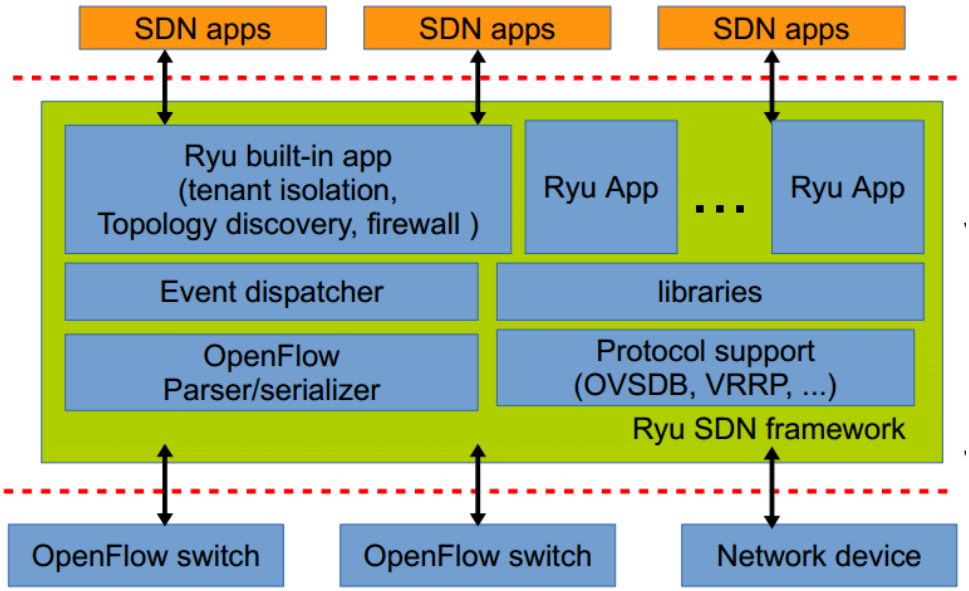


FIGURE 1.4 – Architecture du contrôleur RYU [4]

Nous allons utiliser sur notre projet IDS avec l'apprentissage automatique le contrôleur RYU sur SDN en raison de sa facilité à intégrer les applications existante et il est bien documenté et open-source.

1.8 Conclusions

Dans ce chapitre, nous avons fourni une base théorique sur la nouvelle technologie SDN qui élimine la nature complexe et statique des architectures de réseau distribuées traditionnelles,

nous avons présenté ses déférents modèles, son architecture et ses avantages ce qui nous donne une bonne compréhension de cette nouvelle technologie. Dans ce qui suit, nous nous intéressons à présenter les étapes à suivre afin de mettre cette technologie en réalité.

Chapitre 2

Apprentissage Automatique

2.1 Introduction

Parmi les branches de l'intelligence artificielle (IA) c'est L'apprentissage automatique qui donne aux Machines la capacité d'apprendre sans être programmée. Les chercheurs dans ce domaine doivent comprendre comment fonctionne le cerveau humain et comment le traitement de l'information est observé dans le système nerveux biologique pour donner aux machines la capacité d'apprendre à partir de jeu de données, de les interpréter et de prendre des décisions éclairées. Dans ce chapitre, nous allons présenter en détaille l'apprentissage automatique avec les différents algorithmes qu'ils existent.

2.2 Approche d'Apprentissage Automatique

L'apprentissage automatique est une branche spéciale de l'intelligence artificielle qui acquiert des connaissances à partir de données d'entraînement basées sur des faits connus. L'apprentissage automatique est défini comme une étude qui permet aux ordinateurs d'apprendre des connaissances sans être programmés, mentionnés par Arthur Samuel en 1959.L'apprentissage automatique se concentre principalement sur la prédiction. Les techniques d'apprentissage automatique sont classées en trois grandes catégories telles que l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement.[20]

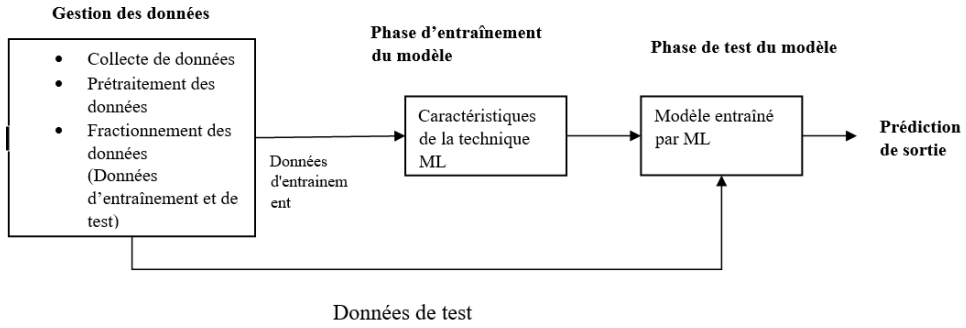


FIGURE 2.1 – Les phases d'apprentissage automatique

2.3 Types d'Algorithmes d'Apprentissage Automatique

Il existe quatre types d'algorithmes d'apprentissage automatique : supervisé, non supervisé, semi-supervisé, et des algorithmes d'apprentissage par renforcement.

2.3.1 Apprentissage Supervisé

L'apprentissage automatique supervisé exige à ses algorithmes d'utiliser des exemples étiquetés pour appliquer ce qu'ils ont appris dans le passé aux nouvelles données afin de prédire des événements futurs à partir de l'analyse d'un ensemble de données déjà connues, dont il produit une fonction déduite pour faire des prédictions sur la valeur de sortie, il peut également comparer sa sortie aux sorties correctes et prédites et trouver des erreurs pour modifier le modèle en conséquence. En apprentissage supervisé, on distingue entre deux types de tâches : [28]

— **Classification** : Quand la variable cible (à prédire) est discrète. Ce qui revient à accorder une classe (ou étiquette) à chaque entrée. C'est le cas si on cherche à prédire la tendance d'un mouvement futur d'un actif (haut, neutre, bas).

— **La régression** Quand la variable cible à prédire est continue. Voici quelques algorithmes d'apprentissage supervisé les plus importants [10] : — K plus proches voisins — Régression linéaire — Régression logistique — Machines à vecteurs de support (SVM) — Arbres de décision et forêts aléatoires.

2.3.2 Apprentissage non Supervisé

Pour ce type d'apprentissage, la base de données d'apprentissage ne contient pas de variable cible (comme on l'a vu en apprentissage supervisé). Il y a seulement un ensemble de données collectées en entrée. L'algorithme doit découvrir par lui-même la structure en fonction des

données [28].Nous citons quelques algorithmes d'apprentissage non supervisé les plus importants [10] :

— Clustering : — K-Means — Analyse des clusters hiérarchiques (HCA) — Maximisation des attentes.

— Visualisation et réduction de la dimensionnalité : — Analyse en composantes principales (ACP) — Kernel PCA — L'encastrement linéaire local (LLE). — T-distribué Stochastic Neighbor Embedding (t-SNE).

— Apprentissage des règles d'association : — Apriori — Eclat.

2.3.3 Apprentissage par Renforcement

Aucun résultat explicite n'est établi dans l'apprentissage par renforcement (RL), et l'agent apprend via le retour d'information après avoir interagi avec l'environnement. Il prend des mesures et prend des décisions en fonction de la récompense qu'il reçoit. De plus, les modèles d'apprentissage humains et animaux l'influencent. De telles caractéristiques en font un est une robotique hautement dynamique où le système apprend à faire des tâches sans programmation explicite. Il est également crucial de choisir la bonne fonction de récompense[40] Les techniques d'apprentissage par renforcement sont généralement utilisées dans le scénario suivant :[22]

— Lorsque les données historiques et les exemples passés ne sont pas disponibles pour l'entraînement du modèle

— L'objectif global est connu, et l'environnement peut être détecté pour maximiser les deux résultats à court et à long terme.

— lorsque les valeurs exactes correctes et erronées pour un scénario particulier sont inconnues a priori

2.3.4 Apprentissage semi-Supervisé

Dans les deux types précédents, soit toutes les observations de Le jeu de données n'a pas d'étiquettes ou toutes les observations ont des étiquettes. Apprentissage semi-supervisé se situe plutôt au milieu. Dans de nombreux cas pratiques, le coût de l'étiquetage est relativement élevé parce qu'elle nécessite des experts humains qualifiés. Par conséquent, les algorithmes semi-supervisés sont les meilleures options pour l'élaboration de modèles lorsque les étiquettes sont absentes dans la plupart des observations, mais présent dans quelques[22]

2.4 Algorithmes d'Apprentissage Supervisé

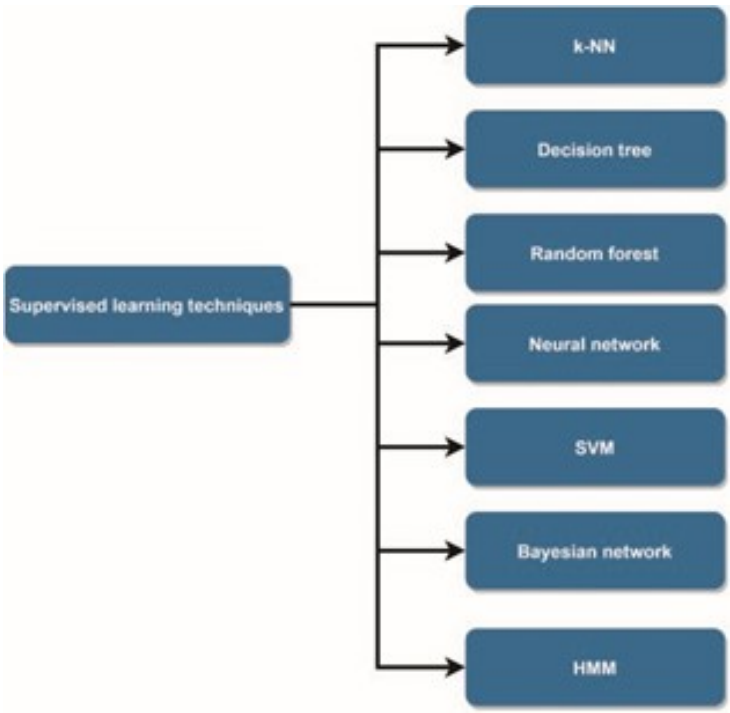


FIGURE 2.2 – Les techniques de l'apprentissage supervisée

2.4.1 k-Nearest Neighbor (k-NN)

L'un des algorithmes d'apprentissage automatique les plus utilisés est k-NN. Il s'agit d'une approche supervisée, non paramétrique et basée sur la distance qui a été introduite pour la première fois en 1951[17] Diverses techniques de mesure de distance sont utilisées dans le voisin le plus proche. Le voisin K le plus proche trouve k nombre d'échantillons dans les données d'apprentissage qui sont les plus proches de l'échantillon de test, puis il attribue l'étiquette de classe la plus fréquente parmi les échantillons d'apprentissage considérés à l'échantillon de test. Pour la classification des échantillons, le K plus proche voisin est connu comme l'approche la plus simple et la plus non paramétrique[11]K-plus proche voisin peut être mentionné comme un apprenant basé sur l'instance, et non comme un apprenant inductif.

2.4.2 Arbre de Décision

La création d'un classifier permettant de prédire la valeur d'une classe cible pour une instance de test invisible, basée sur plusieurs instances déjà connues, est la tâche de l'arbre de décision (DT). Par le biais d'une séquence de décisions, une instance de test invisible est classée par un arbre de décision[38] L'arbre de décision est très populaire en tant que classificateur unique en raison de sa simplicité et de sa mise en œuvre plus facile. L'arbre de décision peut être développé

en 2 types : (I) l'arbre de classification, avec une plage d'étiquettes de classe symboliques, et (II) l'arbre de régression, avec une plage d'étiquettes de classe à valeurs numériques[38]

2.4.3 Bayes Naïf

Sur la base de l'étiquette de classe donnée, Bayes naïf suppose que les attributs sont conditionnellement indépendants et tente donc d'estimer la probabilité conditionnelle de classe[32] La méthode bayésienne naïve produit souvent de bons résultats dans la classification lorsqu'il existe des relations plus simples. Le bayésien naïf ne nécessite qu'un seul scan des données d'entraînement, ce qui facilite considérablement la tâche de classification.[20]

2.4.4 Réseau de Neurones Artificiels

Le réseau de neurones artificiels (ANN) est une unité de traitement de l'information qui s'inspire de la fonctionnalité du cerveau humain. En règle générale, les réseaux neuronaux sont organisés en couches composées d'un certain nombre de nœuds interconnectés qui contiennent une fonction d'activation. Les motifs sont présentés au réseau via la couche d'entrée, qui communique avec une ou plusieurs couches cachées où, via un système de connexions pondérées, le traitement réel est effectué. Les couches masquées sont ensuite liées à une couche en sortie pour produire le résultat de la détection en sortie.[20]

2.4.5 Support Vecteur Machines

La machine à vecteurs de support (SVM) a été introduite au milieu des années 1990. Le concept derrière SVM pour la détection d'intrusion est essentiellement d'utiliser les données d'apprentissage comme une description de la classe normale d'objets ou qui est connue sous le nom de non-attaque dans le système de détection d'intrusion, et donc de supposer le reste comme des anomalies. Le classifier construit par la méthodologie des machines à vecteurs de support discrimine l'espace d'entrée dans une région finie où les objets normaux sont contenus et où tout le reste de l'espace est supposé contenir les anomalies.[20]

2.4.6 Forêt Aléatoire

Random Forest, ou Forêt Aléatoires, fonctionne en créant plusieurs arbres de décision et en faisant une prédiction avec chaque arbre puis en votant pour déterminer le résultat final, chaque arbre est créé à partir d'un échantillon de données et de caractéristiques différentes. L'idée principale d'utiliser plusieurs arbres est de réduire l'overfitting et obtenir un modèle plus précis.

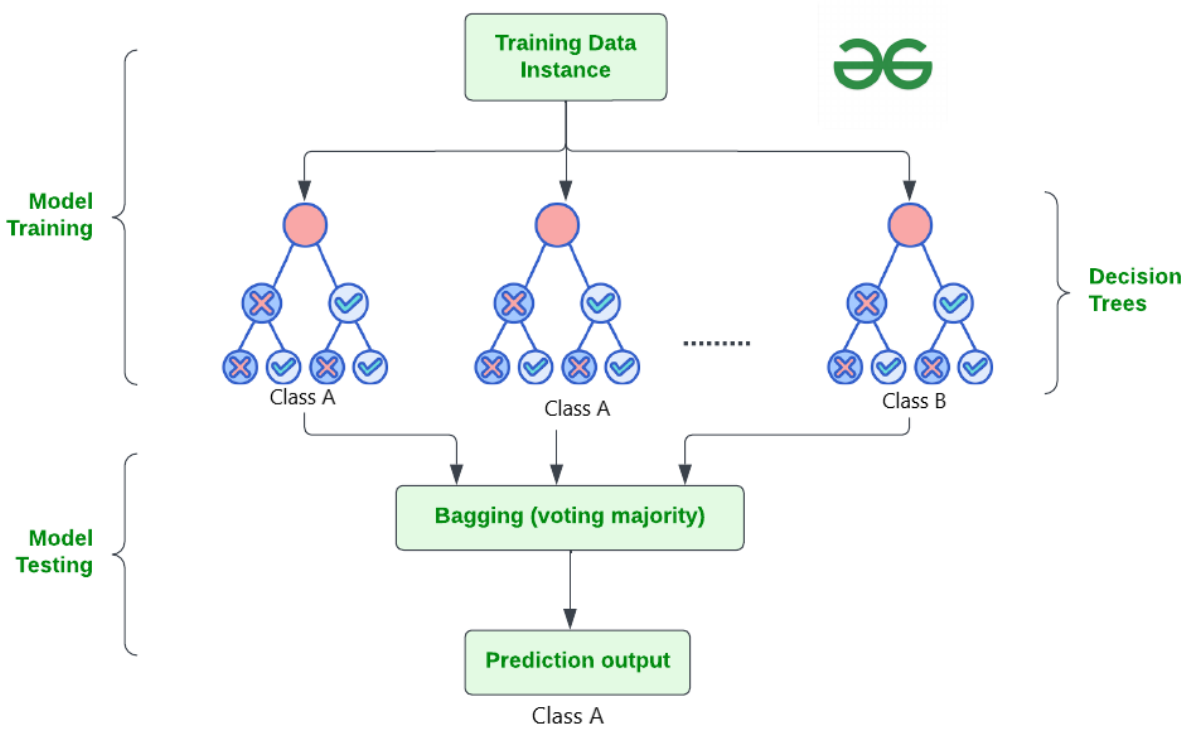


FIGURE 2.3 – Forêt aléatoire [19]

Nous allons exploiter l'algorithme foret aléatoire dans l'entraînement de notre modèle IDS en raison, de sa simplicité et de sa flexibilité, ainsi l'algorithme foret aléatoire fait la diminution du taux du surajustemen dans l'entrainement et le test du modèle.

Technique d'apprentissage automatique	**Avantages**	**Inconvénients**
Arbre de décision	— Interprétation facile Sélection de caractéristiques discriminantes Moins gourmande en ressources CPU Fonctionne avec des données continues et discrètes	— Instable, soumis aux données d'entraînement Problème de surajustement
Forêt aléatoire (RF)	— Convient aux données d'entraînement volumineuses Comparativement moins d'instabilité Évite les problèmes de surajustement	— Processus d'entraînement lent Résultats biaisés en cas de données déséquilibrées
SVM	— Convient aux données de grande dimension Convient aux données séparables linéairement et non linéairement	— Coûteux en termes de calcul pour les données volumineuses Évite les problèmes de surajustement
k-NN	— Mise en œuvre facile Choix des fonctions de distance	— Gourmand en ressources CPU en raison du calcul de la distance, Gourmand en mémoire pour stocker tous les jeux de données d'entraînement
Réseau bayésien	— Mise en œuvre facile De bons résultats pour une petite quantité de données d'entraînement	— Hypothèse d'indépendance Difficile de traiter des données continues
ANN	— Prédiction rapide après l'entraînement Convient aux données de grande dimension	— Nécessite une puissance de calcul élevée pour l'entraînement Difficile d'interpréter les résultats

TABLE 2.1 – Avantages et inconvénients des techniques d'apprentissage automatique supervisées

2.5 Conclusions

Dans ce chapitre, nous avons présenté un aperçu de l'apprentissage automatique en nous concentrant sur l'approche de classification supervisée et ses modèles les plus connus, avec une comparaison en présentant les avantages et les inconvénients de chaque méthode.

Chapitre 3

Système de Détection d'Intrusions avec l'Apprentissage Automatique

3.1 Introduction

La sécurité en ligne est aujourd'hui l'une des principales préoccupations. Étant donné qu'internet est exposé à différentes attaques, il est primordial de mettre en place un système de protection de ces données et des utilisateurs qui les utilisent. L'invention du système de détection d'intrusion (IDS) vise donc à satisfaire à cette exigence. Le système de détection d'intrusion est adapté par les administrateurs réseau pour éviter les attaques malveillantes. Par conséquent, il est devenu un élément essentiel de la gestion de la sécurité et il signale toute tentative d'intrusion ou d'utilisation abusive sur le réseau. Ce chapitre mettra en lumière le IDS en tant que solution de sécurité Traditionnelle et sécurité programmable à l'aide d'apprentissage automatique, en examinant en détail ses fondements, ses avantages et ses fonctionnalités clés en matière de sécurité. De plus, nous étudierons les approches du système de détection d'intrusion avec les méthodes d'apprentissage automatique pour un objectif de réduire les fausses alarmes et augmenter le taux de détection.

3.2 Système de Détection d'Intrusions

Un IDS est défini comme « une technologie de sécurité efficace, capable de détecter, de prévenir et éventuellement de réagir. aux attaques informatiques » est l'un des composants standards des infrastructures de sécurité[25]. Il surveille les sources d'activités cible, telles que les données d'audit et de trafic réseau dans un ordinateur ou des systèmes réseau, et déploie

diverses techniques pour fournir des services de sécurité. L'objectif principal de l'IDS est de détecter efficacement toutes les intrusions. La mise en œuvre d'IDS permet aux administrateurs réseau de détecter les violations des objectifs de sécurité. Ces violations de l'objectif de sécurité vont des attaquants externes qui tentent d'obtenir[24]un accès non autorisé à l'infrastructure de sécurité du réseau ou rendant les ressources indisponibles aux internes, qui abusent de leur accès aux ressources du système.

3.3 Types des Systèmes de Détection d'Intrusions

Il existe deux types d'IDS :

3.3.1 Système de Détection d'Intrusion Basé sur l'Hôte

L'objectif du HIDS est de contrôler l'état et le comportement dynamique du système informatique, Ce système de détection vérifie toutes les activités des paquets inspectés sur un réseau. Les HIDS reconnaissent quelles ressources sont utilisées et quel programme accède à ces ressources. Si, dans le réseau, des changements ou des ajustements, l'administrateur du système reçoit des alertes sur le réseau.Le HIDS devient progressivement pour garantir les cadres de l'ordinateur hôte et ses activités de réseau. HIDS avec des informations basées sur l'hôte est incorporé dans les cadres informatiques pour identifier les activités anormales de l'intrus, le comportement nocif, les anomalies de l'application et préserver les systèmes d'information et signaler les occasions à l'administrateur du système HIDS.[34]

3.3.2 Système de Détection d'Intrusion Basé sur le Réseau

NIDS est utilisé pour surveiller et analyser le trafic réseau sur un segment de réseau spécifique afin de détecter les activités suspectes. La figure 3.1 représente les NIDS et ses étapes de détection des attaques. NIDS utilisés dans l'analyse au niveau des paquets pour tous les systèmes du segment réseau en vérifiant les activités au niveau de l'IP, du réseau de transport et du protocole d'application et les en-têtes de paquets pour détecter de nombreuses attaques DOS basées sur IP telles que les attaques TCP SYN, les attaques par paquets de fragments. Le NIDS se concentre davantage sur l'abus des vulnérabilités, tandis que le HIDS se concentre sur l'abus de privilèges. Le NIDS est moins coûteux et plus rapide que le HIDS, car il n'est pas nécessaire de maintenir la programmation des capteurs au niveau de l'hôte, et il surveille le trafic en temps réel ou en temps quasi réel. Par conséquent, le NIDS peut détecter les attaques au fur et à mesure qu'elles se produisent. Cependant, NIDS n'indique pas si ces attaques réussissent ou non, car il n'analyse pas le système de journalisation. Le problème avec NIDS est qu'il a une visibilité restreinte à l'intérieur de la machine hôte, et il n'y a pas de moyen efficace d'analyser le trafic réseau chiffré pour détecter les attaques. Par conséquent, jusqu'à présent, de nombreuses recherches ont progressé pour développer des moyens efficaces pour le NIDS de détecter les attaques. Il existe

plusieurs produits de détection d'intrusion sur le réseau, tels que Snort et NetSTAT, qui est un outil destiné au NIDS en temps réel [31]

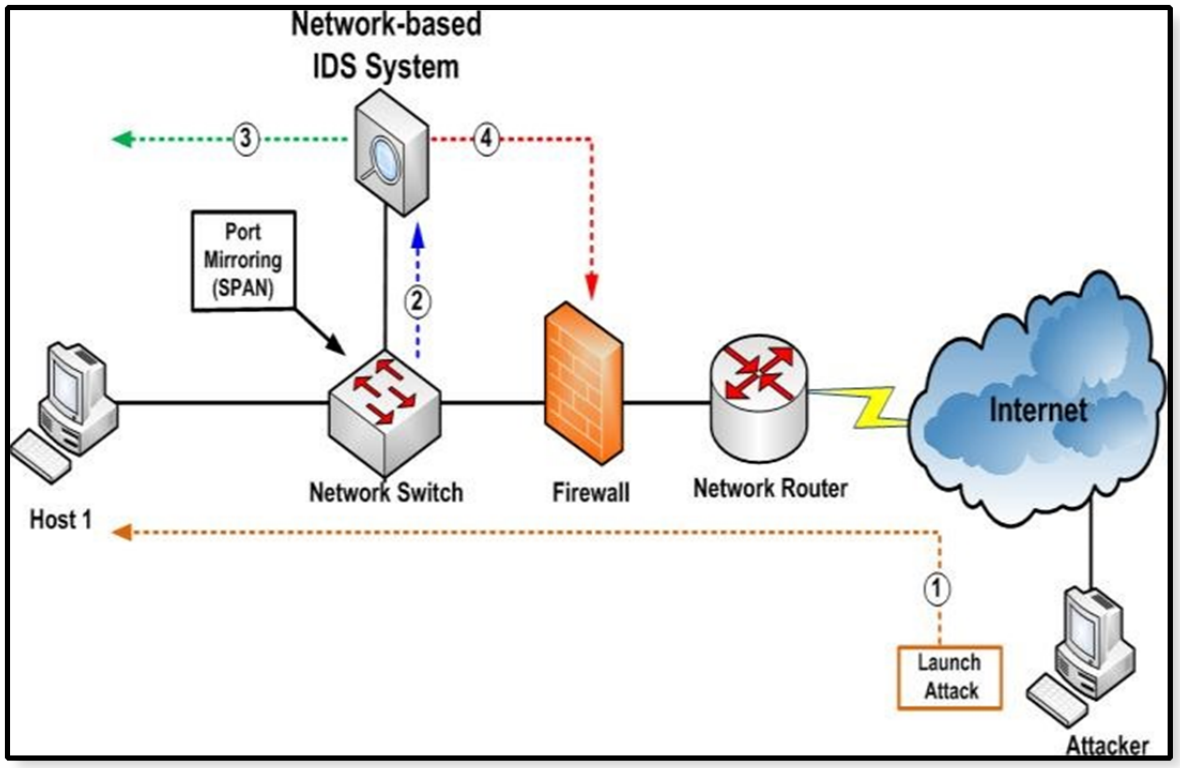


FIGURE 3.1 – IDS basé sur le réseau (NIDS)[31]

Le NIDS consiste en deux classifications essentielles :

3.3.2.1 Système de Détection d'Intrusion Basée sur des Signatures

Identifie les anomalies en les comparant aux fichiers journaux ou aux signatures précédents pour détecter les intrusions, parmi ses avantages :

— Les attaques connues comme tentatives d'intrusion sont souvent évaluées et identifiées de manière efficace.

— L'administrateur système utilise moins de temps pour gérer le taux de faux positifs, ce qui permet de réduire la consommation de temps.

Inconvénients :

— La détection des tentatives d'intrusion correspondant à une fonctionnalité connue de la base de données est limitée par les systèmes basés sur l'utilisation abusive, mais n'est pas possible pour les attaques inconnues.

— Cette méthode détecte les nouvelles attaques, mais il contraint les concessionnaires à mettre à jour leurs modèles dans les bases de données.

3.3.2.2 Système de Détection d'Intrusion Basée sur des Anomalies

Identifier l'utilisation abusive ainsi que les anomalies informatiques, puis classer comme normal ou attaque dépend de l'heuristique des signatures. Il s'agit principalement de trois types de méthodes, l'apprentissage automatique, les méthodes statistiques et les méthodes basées sur les connaissances, la méthode la plus efficace, c'est avec l'apprentissage automatique. Le système d'identification basé sur les anomalies comprend trois sections telles que la découverte d'anomalies semi-supervisées, supervisées et gérées. Le système de détection d'anomalies supervisé tente de construire le modèle et construit séparément les enregistrements anormaux et les enregistrements normaux. Les méthodes semi-supervisées de détection d'anomalies tentent de construire le modèle en fonction des données normales. Si les enregistrements ne correspondent pas au modèle, il s'agit d'une anomalie. Maintenant, les méthodes semi-supervisées peuvent construire le modèle dépend de données anormales, mais très difficile d'identifier toutes les anomalies. À cette fin, la détection semi-supervisée nécessite une base de données étiquetée et représente souvent le taux élevé de faux positifs. Pour résoudre ce problème et trouver de nouvelles anomalies, un système de détection d'anomalies non supervisé est utilisé. Cette méthode n'a pas besoin d'un jeu de données étiqueté et installé dans un système sans modification.[34].Parmi ses Avantages :

— Il n'est pas indispensable de mettre à jour la base de données afin de repérer les nouvelles attaques.

— En même temps, il surveille le fonctionnement du réseau et élabore des profils d'activités réseau.

— Identifiez de manière optimale les dangers dans un système plus large.

Inconvénient

— Lorsque le réseau présente un comportement anormal dans le trafic normal, il ne communiquera pas l'alerte à l'administrateur.

— Dans une configuration fondée sur les anomalies, il est possible que les faux positifs augmentent.

3.4 Architecture du IDS

Avec le passage du temps et la multiplication des attaques informatiques, plusieurs architectures d'IDS ont été proposées. Axelsson [9]a proposé une architecture commune pour l'IDS, comme le montre la figure 4.

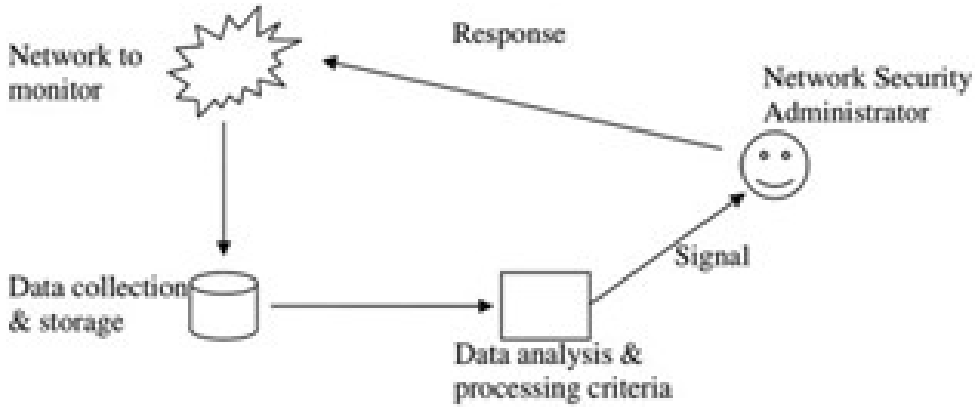


FIGURE 3.2 – Architecture IDS[24]

Selon Axelsson [9], les composants standard de l'IDS sont les suivants. Le réseau à surveiller est l'identité à surveiller pour détecter les intrusions. Il peut s'agir d'un hôte unique ou d'un réseau, L'unité de collecte et de stockage des données est responsable de la collecte des données de divers événements et de leur conversion dans un format approprié et de leur stockage sur disque. L'unité d'analyse et de traitement des données est le cerveau d'IDS et il contient toutes les fonctionnalités permettant de détecter le comportement suspect du trafic d'attaque. Lors de la détection d'une attaque, un signal est généré. En fonction du type d'IDS, le système peut déclencher l'action pour atténuer le problème ou un signal est transmis à l'administrateur réseau pour qu'il prenne les mesures appropriées. Signal, Cette partie du système gère toutes les sorties de l'IDS, Il peut s'agir d'une réponse automatisée à une intrusion ou d'une alerte d'activité malveillante pour un administrateur de sécurité réseau.[24]

3.5 IDS avec l'Apprentissage Automatique

Comme l'indique le tableau détaillé d'architecture IDS avec ML Figure 3.3 l'objectif est de classer le trafic entrant à l'aide des méthodes ML en trafic normal ou d'attaque (DoS/DDoS).

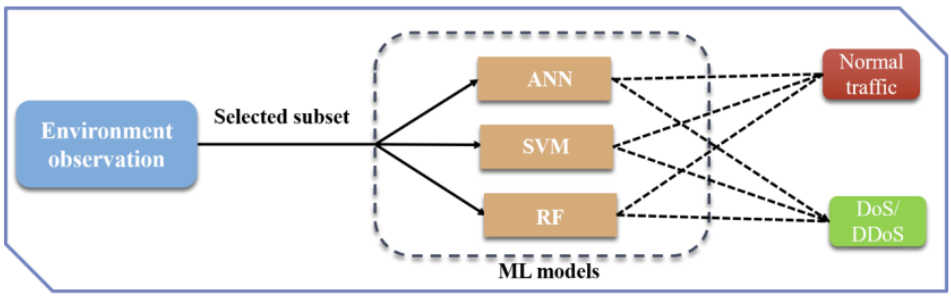


FIGURE 3.3 – Architecture détaillée du modèle IDS [30]

Le processus de réalisation du Modèle IDS passe par les étapes et étapes suivantes, comme le montré dans la figure 3.4, qui sont appliquées à tous les algorithmes ML utilisés pour obtenir

les meilleures performances et les meilleurs résultats possibles :

1. Préparation et prétraitement des données.
2. Traitement du modèle.
3. Évaluation du modèle.

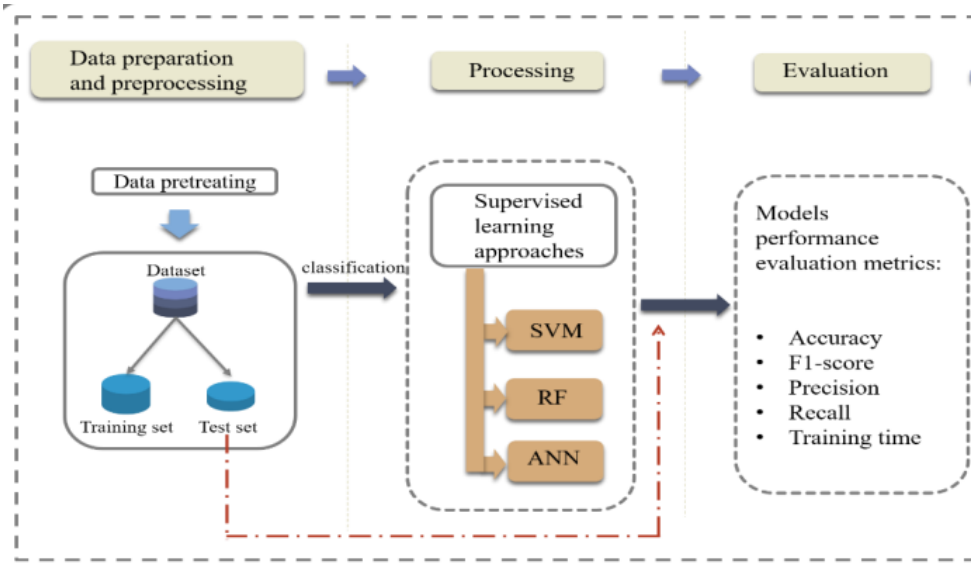


FIGURE 3.4 – Les étapes de processus [30]

3.5.1 Préparation et Prétraitement des Données

la première étape consiste à obtenir suffisamment de données pour représenter le problème à résoudre. Cela ne s'avère pas toujours simple. Il est plus coûteux d'obtenir certaines informations que d'autres. Par exemple, il est plus facile d'obtenir un header d'un paquet réseau que d'obtenir une information dans la partie data lorsque celle-ci est chiffrée. La prochaine étape consiste à nettoyer, également connu sous le nom de prétraitement, les données collectées, c'est-à-dire à réduire ce qui est strictement intéressant, ainsi qu'à les traduire. L'objectif de cette étape est d'améliorer la précision du modèle, de maximiser son temps d'exécution et son apprentissage, ainsi que de réduire sa dimension.

3.5.1.1 Sélection des Caractéristiques (Feature Selection)

L'objectif principal de l'utilisation de l'étape de sélection d'entités est de sélectionner les entités qui généreraient une erreur de classification minimale, il existe trois types des techniques et des méthodes de sélection des caractéristiques, on a le filtre méthodes, les Wrapper methods et les Embedded méthodes. Ces méthodes peuvent être combinées pour obtenir de meilleurs résultats .

3.5.2 Traitement du Modèle

Dans la partie de traitement, on a la phase d'apprentissage, Les données d'entrainement sont constituées d'un sous-ensemble de l'ensemble des informations nettoyées, ce qui permet de mener à bien la phase d'apprentissage du modèle. Cela permet de personnaliser les réglages du modèle. en particulier, il existe plusieurs algorithmes d'apprentissage automatique tel que les supervisés et les non supervisés qu'ils ont bien détaillés dans le chapitre précèdent ensuite, on a la phase de validation, Durant cette phase, on va tester et valider le modèle et ses paramètres selon des critères se basant sur ses résultats. Il permet d'obtenir le meilleur modèle généralisant les données obtenues lors de la phase d'apprentissage. Pour cela, on a un ensemble de donnée pour l'apprentissage et un autre pour les tests.

3.5.3 Évaluation des Performances du Modèle

L'évaluation des algorithmes de ML après leur application est une étape essentielle pour s'assurer qu'ils fonctionnent correctement à l'aide des métriques d'évaluation des performances. Dans le modèle IDS, il faut appuyer sur l'utilisation des mesures de performance les plus importantes pour évaluer le modèle et effectuer une analyse comparative telle que la précision, le rappel, le score F1 ainsi que l'accuracy.

3.6 Travaux Connexes sur l'IDS avec L'Apprentissage Automatique dans SDN

Nous avons des résumés détaillés sur des différentes études sur le système de détection d'intrusion à l'aide d'apprentissage automatique sur SDN.

3.6.1 Étude 1 Song et al [36]

Dans cet article, ils ont présenté un nouveau système de détection et de réponse aux intrusions sur SDN utilisant des techniques d'apprentissage automatique et le routage réactif. Il se compose de trois sous-systèmes le prétraitement des données, la modélisation prédictive des données et le sous-système de prise de décision et de réponse Ils ont développé une nouvelle méthode pour traiter les données et l'alerte indécises compte tenu de la haute résilience du SDN, le système Eunoia proposé, sélectionne soigneusement les ensembles de caractéristiques les plus importants pour atteindre une grande précision, ils ont utilisé 10 et 15 caractéristiques importantes sur l'ensemble des données (DATASET) KDD99 à l'aide de l'algorithme l'arbre de décision pour classer le jeu de données Les quatre mesures de performances (Recall, Précision, accuracy,F1 Score) du jeu de données ont des valeurs d'environ 0,824 0,884 Pour les 10 caractéristiques et d'environ 0,911 0,946 pour 15 caractéristiques. En utilisant le routage réactif, le système proposé réagit à l'incertitude du SDN, Pour évaluer le système ils ont généré différentes tailles

de trafic à l'intérieur du Simulateur réseau Mininet et mesurent l'heure de début et l'heure de fin de l'envoi du trafic de la source à la destination, si la taille du trafic augmentait, le temps d'exécution total augmentait également légèrement.

3.6.2 Étude 2 Tang et al [37]

Dans cet article, ils ont présenté un IDS dans l'environnement SDN à l'aide de l'algorithme GRU-RNN Dans l'apprentissage profond qu'est une technique puissante qui permet de représenter la relation entre les événements actuels et précédents et d'améliorer le taux de détection des anomalies, ils ont 3 composants de système essentiel collecteur de flux collecte les statistiques de flux et les envoie au détecteur d'anomalies et détecteur d'anomalies utilise GRU-RNN pour déterminer les anomalies et enfin L'atténuation d'anomalie agit en fonction des résultats du détecteur d'anomalies sur le côté de Performance GRU-RNN surpasse les algorithmes SVM, RNN et NB Tree et il atteint une précision de détection à 89% en utilisant un nombre minimal de caractéristiques par rapport à d'autres approches de pointe en utilisant l'ensemble des données (DATASET) NSL-KDD, ils ont utilisé l'architecture GRU pour résoudre le problème de RNN traditionnel de gradient disparition/explosion et ils ont implémenté en python au sein d'un contrôleur POX et évalue les performances du contrôleur SDN à l'aide de l'outil Cbench.

3.6.3 Étude 3 Nanda et al [29]

Dans cet article, ils ont utilisé des données historiques collectées à partir de 32 pots de miel (honeyPot) du monde entier et des algorithmes d'apprentissage automatique Naïve-Bayes, un arbre de décision, C4.5, BayesNet pour entraîner leur modèle. Leur objectif est de prédire l'hôte qui sera attaqué. Cette prédiction est alors utilisée par le contrôleur SDN pour définir des règles de sécurité, ils proposent de bloquer l'ensemble du sous-réseau pour empêcher de futures attaques plutôt que de restreindre l'accès pour une adresse IP d'attaquant spécifique, ils ont également choisi un niveau de seuil alpha en pourcentage comme la probabilité minimale requise pour considérer un hôte comme vulnérable, ils ont remarqué que l'augmentation de la valeur alpha diminue la précision de la prédiction. Globalement, la précision de prédiction moyenne la plus élevée 91,68% a été obtenue Avec le réseau bayésien.

3.6.4 Étude 4 DaSilva et al [14]

ATLANTIQUE est Un framework pour la détection, la classification et l'atténuation des anomalies du trafic dans SDN le framework atlantique est composé de 2 phases. La première est la phase légère, elle est plus rapide et exécutée plus souvent. Elle est responsable de la surveillance du trafic et de la détection des anomalies à l'aide de l'analyse entropique pour détecter les changements dans les caractéristiques du trafic si une anomalie est détectée, la deuxième phase appelée phase lourde est appelée, elle utilise l'algorithme svm pour la classification des flux,

si des flux malveillants sont détectés, l'atténuation est appliquée en installant de nouvelles règles dans le pare-feu et en supprimant tous les flux et paquets malveillants, sinon les flux sont enregistrés dans l'interface administrateur pour une enquête plus approfondie par l'humain, dans leurs expériences, ils ont évalué le cadre en utilisant l'attaque ddos et ont remarqué que la phase légère ne prend qu'environ 0,07 et la phase lourde prend 3 secondes et une précision globale de 88,7 à partir de ces résultats, nous pouvons apprécier les avantages de l'utilisation de la phase légère au lieu de toujours utiliser la phase lourde pour réduire les frais généraux de détection des anomalies.

3.6.5 Étude 5 Wang et al [39]

Dans ce travail, ils ont utilisé l'algorithme d'arbre de décision id3 pour détecter le meilleur sous ensemble de fonctionnalités (features) de l'ensemble de données kdd d'origine (KDD dataset contient des flux de données de plusieurs catégories d'attaques DOS,R2L,U2R,Probing et ainsi des flux non malicieux) , car moins de fonctionnalités pour les signatures d'attaque permettent une détection plus rapide des attaques, mais cela pourrait augmenter le taux de fausses alarmes, l'algorithme sélectionne les meilleures fonctionnalités en utilisant le gain d'informations de chaque attribut, puis l'algorithme d'apprentissage automatique svm est utilisé pour la classification (flux normal ou malveillant) et ils ont obtenu une précision moyenne de 97,5 en utilisant 23, 29, 41 fonctionnalités, en addition de ce modèle, ils ont également utilisé Snort et sFlow pour une meilleure détection des intrusions.

3.6.6 Étude 6 Liu et al [26]

Cette étude propose une méthode de détection DDoS basée sur l'ingénierie des caractéristiques et l'apprentissage automatique en SDN. La méthode est divisée en deux parties, le module d'extraction de caractéristiques et de sélection de modèles, et le module de détection des attaques DDoS. Dans le module 1, ils ont utilisé un algorithme d'optimisation binaire amélioré du loup gris (Binary Grey Wolf Optimization Algorithm) pour l'extraction de caractéristiques sur l'ensemble de données (DATASET CSE6CIC6IDS2018). Ils ont utilisé cinq modèles d'apprentissage automatique (RF, SVM, XGBoost, Decision Tree et k-NN) pour évaluer et sélectionner le meilleur classificateur pour l'ensemble de données d'origine et qui sont extraits de caractéristiques, respectivement. Les résultats ont montré que la précision la plus élevée de XGBoost était de 0,969 sur l'ensemble de données d'origine, Après l'extraction des entités, le nombre d'entités sur l'ensemble de données est passé de 79 à 26, et bien qu'il y ait moins d'entités, tous les classificateurs se sont améliorés dans toutes les mesures. Parmi eux, le classificateur RF a obtenu les meilleurs résultats en termes d'exactitude, de précision, de rappel et de f1-scores métriques avec 0,9913, 0,9843, 0,9992 et 0,9913, respectivement. Dans le module 2, et ils ont déployé le meilleur classificateur sélectionné dans le module 1 sur le contrôleur et effectué la détection DDoS, Les résultats montrent que la méthode proposée permet de détecter les attaques DDoS

et d'alerter les utilisateurs. Ils ont Mesuré les performances évaluées à l'aide de la validation croisée K-fold.

3.7 Discussions

Le tableau 3.1 montre la comparaison entre les travaux connexes précédents en ce qui concerne les mesures suivantes : les méthodes et le modèle d'apprentissage automatique utilisés, les ensembles de données et le nombre de caractéristiques utilisées, la précision, obtenus par les algorithmes d'apprentissage automatique appliqués avec l'avantage et inconvénient de chaque étude.

Les études présentées offrent plusieurs méthodes d'apprentissage automatique utilisées à la détection d'intrusions dans SDN. Song et al.[36] ont mis en avant un système de détection réactif qui utilise un nombre limité de caractéristiques pour une haute précision, tandis que Tang et al.[37] ont exploité la puissance des RNN pour améliorer la détection des anomalies. Nanda et al.[29] ont adopté une approche prédictive basée sur des données historiques pour prévenir les attaques, et da Silva et al.[14] ont proposé un framework bimodal (phase légère et phase lourde) pour une détection et une atténuation rapides des anomalies. Wang et al.[39] ont concentré leurs efforts sur la sélection de caractéristiques pour accélérer la détection en maintenant une précision élevée, et Liu et al.[26] ont combiné l'ingénierie des caractéristiques avec des techniques d'apprentissage automatique pour une détection efficace des attaques DDoS. Chaque étude apporte sa contribution unique à la sécurité des réseaux SDN, avec des taux de précision allant de 82% à 97,55%.

Étude	Méthode d'apprentissage	Avantages	Inconvénients	Jeu de données (DATA Set)	Précision moyenne (%)
[36] 2017	Arbre de décision (DT)	Utilisation de réactifs Routage pour Installation de Flow Règlement Correspondant à Type d'écoulement	Ignorance des informations contextuelles pertinentes	KDD	82
[37] 2018	Réseau de neurones récurrent (RNN)	Détection d'anomalies basée sur le modèle DL-NN (Deep Learning)	Fonctionnalités limitées utilisées	NSL-KDD	89
[26] 2023	Plusieurs méthodes (RF, SVM, XGBoost, Decision Tree et k-NN)	Plusieurs techniques d'apprentissage automatique appliquées pour détection DDoS	Fonctionnalités limitées utilisées et détection non flexible	CSECICIDS2018	98(RF limited features) 96 (XGBoost All data set)
[29] 2016	Réseau bayésien (BayesNet)	Plusieurs techniques d'apprentissage automatique appliquées pour détecter les connexions malveillantes et les hôtes vulnérables	Blocage de l'ensemble du réseau pour éviter les attaques	Synthétique	91
[14] 2016	Machine à vecteurs de support (SVM)	Détection du trafic intrusif à l'aide de la SVM	Inspection manuelle des flux de trafic inconnus	Synthétique	88.7
[39] 2016	Hybride de DT et SVM	Utilisation de caractéristiques réduites à l'aide de l'arbre de décision pour une classification précise par SVM	Résultat comparatif non fourni	KDD	97.55

TABLE 3.1 – Tableau comparatif entre les travaux connexes

3.8 Conclusions

Nous avons présenté dans ce chapitre la structure générale et détaillée de IDS avec l'apprentissage automatique à laquelle nous avons expliqué les types et l'architecture du système de détection d'intrusion, et un résumé sur les phases du modèle IDS à l'aide de l'apprentissage automatique et, nous avons présenté un résumé des travaux connexes de chaque étude qui traitaient des problèmes similaires et du problème de détection des attaques d'intrusion comme DoS/DDoS en SDN. Le chapitre suivant présentera la conception et l'implémentation de notre Projet IDS avec l'apprentissage automatique dans SDN.

Chapitre 4

Conception et Implémentation

4.1 Introduction

Dans ce chapitre, nous allons explorer la conception et l'implémentation de notre solution proposée ainsi faire des tests sur notre projet IDS avec prévention en temps réel dans SDN.dans la partie de la conception, nous allons décrire notre conception générale d'un IDS avec l'apprentissage automatique sur SDN en présentant l'architecture globale ainsi que ses principaux modules.

4.1.1 Création d'un Modèle ML

Nous avons sur la figure 4.1 les phases importantes pour la création du nôtre modèle d'apprentissage automatique pour obtenir des meilleures performances telles que la précision, rappel et f1-score et ensuite, on a le déploiement du modèle ML sur le SDN.

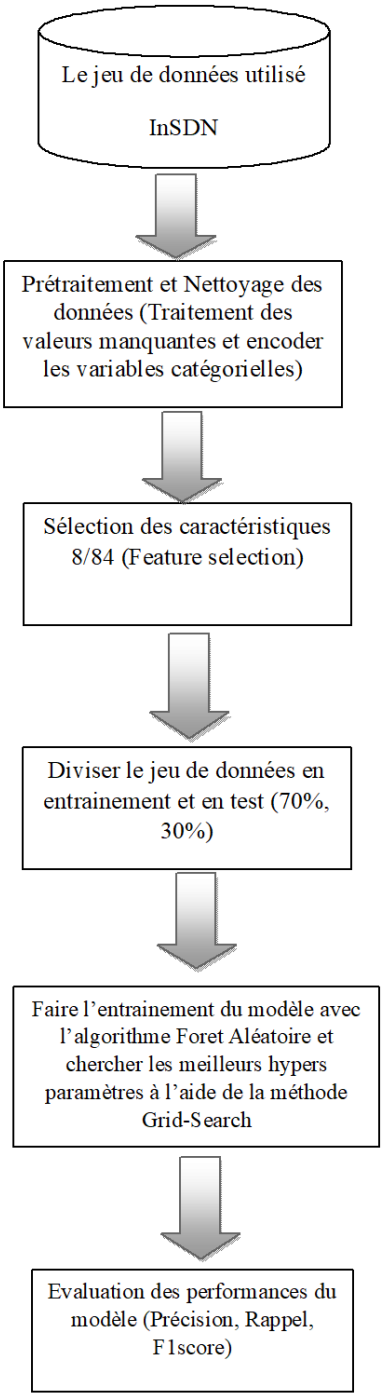


FIGURE 4.1 – Les étapes de la création du modèle ML

4.1.2 Schéma Général de IDS avec Prévention

Nous avons sur la figure 4.2 la conception et le fonctionnement détaillé de notre IDS avec prévention sur SDN.

L'application du contrôleur RYU B.1 fonctionne comme ceci, le contrôleur est programmé pour envoyer périodiquement une demande de statistique de flux (flow stats request) à tous les

switches, puis le switch répond et l'événement flow stat reply est déclenchée et la fonction dans l'événement est programmée pour effectuer l'extraction de fonctionnalités pour les flux un par un. Ces caractéristiques sont transmises à notre classificateur ml pré-entraîné pour prédire s'il s'agit d'une intrusion (type d'attaque) ou d'un trafic normal s'il s'agit d'une intrusion, la fonction drop flow est appelée pour installer une nouvelle règle de flux dans le switch pour bloquer les futures connexions à partir de ce flux (protocole IP src IP dst) et le flux sera également supprimé du switch et les informations sur le flux malveillant sont envoyées au serveur Web via API REST pour afficher les journaux malveillants dans une interface web.

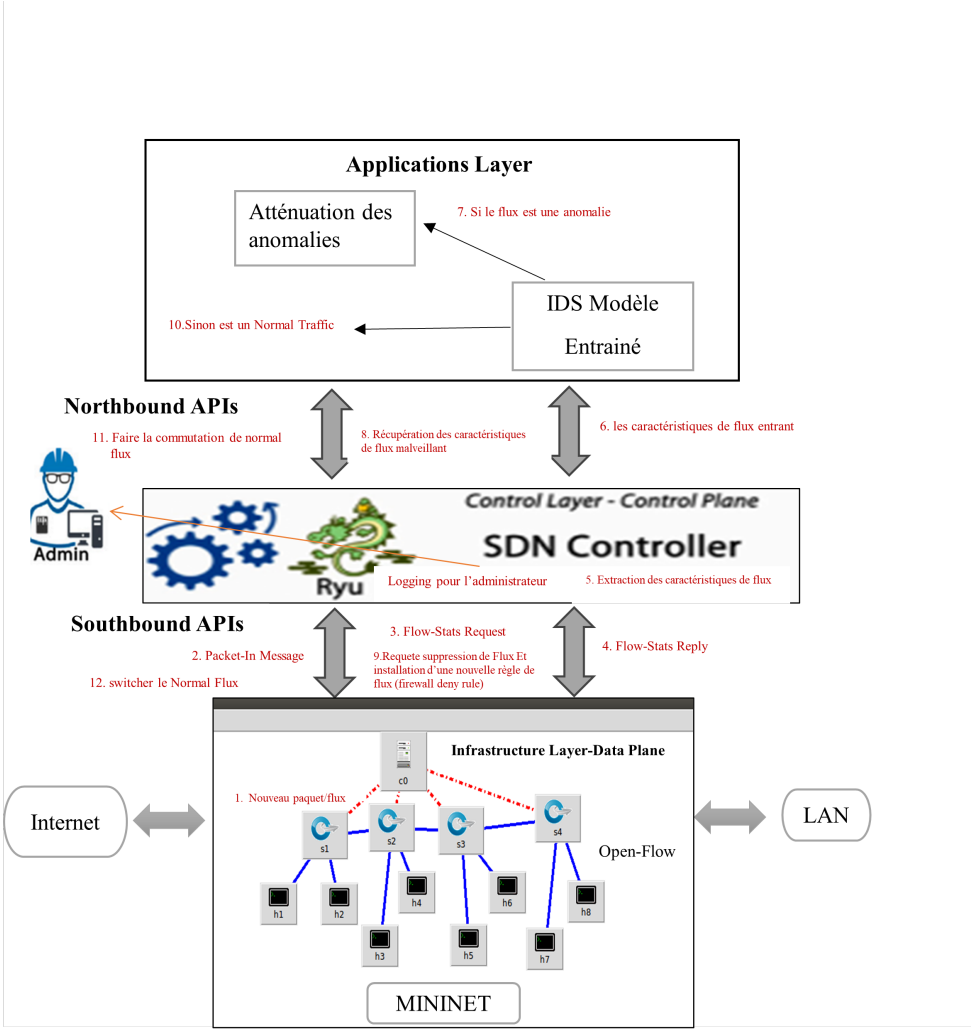


FIGURE 4.2 – Schéma d'un système de détection d'intrusion sur SDN

4.1.2.1 Descriptions de chaque étape

— **Packet-in message** : c'est un message asynchrone envoyé depuis le switch au contrôleur quand un nouveau paquet arrive au switch et il ne sait pas où l'envoyer (il n'y a aucune correspondance dans la table de flux).

— **Flow stat request** : est un message asynchrone envoyé depuis le contrôleur à tous les

switches Périodiquement (chaque 20 secondes) pour récupérer tous les flux dans la table de flux.

— **Flow stat reply** : c'est la réponse à flow stat request et c'est envoyé depuis le switch vers le contrôleur.

— **IDS modèle** : après la réponse de switch le contrôleur extraire les caractéristiques nécessaires depuis les flows un par un et faire passer ces caractéristiques au modèle pré-entrainé pour détecter s'il y a intrusion.

— **Atténuation des anomalies** : si le flow est détecté malicieux, une requête est envoyée depuis le Contrôleur au switch pour faire la suppression de flux et l'installation d'une nouvelle règle de flux dans le switch pour bloquer les futurs paquets (firewall deny rule).

— **Logging pour l'administrateur** : si le flow est détecté malicieux, des informations sur le flow sont Envoyé à l'administrateur et sont affichés sur une interface web (source IP, destination IP, date et Heure).

4.2 Implémentation d'un IDS avec Prévention dans SDN

Dans cette partie, nous allons expliquer comment nous avons implémenté les principaux modules de notre système de détection d'intrusion intelligent à l'aide de l'apprentissage automatique et avec prévention sur le réseau SDN.

4.2.1 Bibliothèque

Les bibliothèques que nous avons importées dans notre projet sont définies dans ce qui suit :

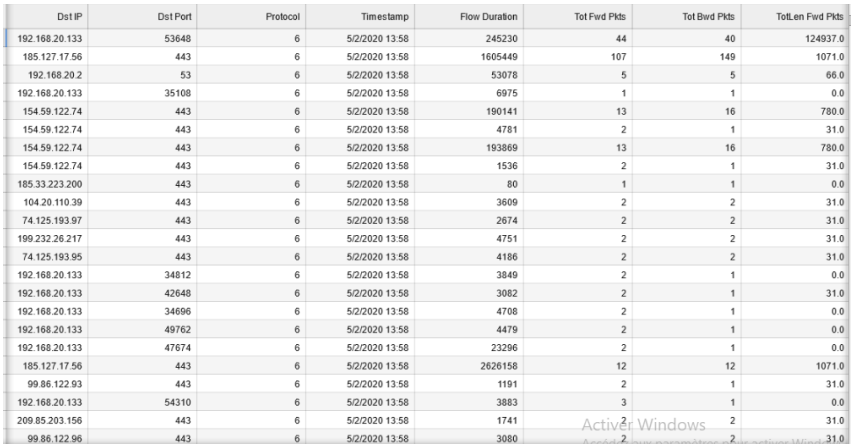
— **Numpy** : est une bibliothèque python qui fournit un objet tableau pour faciliter l'opération avec les listes, il peut être 50 fois plus rapide que la liste python standard. Puisque Numpy rend l'opération sur les tableaux beaucoup plus facile, ceci aidera beaucoup avec les grandes structures de données. L'objet tableau dans NumPy est appelé ndarray, il fournit beaucoup de fonctions qui rendent le travail très facile[5]

— **Pandas** : Pandas est une bibliothèque python à code source ouvert, utilisée pour analyser les données, dotée de nombreuses fonctions d'analyse, de nettoyage et de manipulation des données. Les outils de Pandas sont très utiles pour l'analyse et les structures de données. Le nom "Pandas" fait référence à la fois à "Panel Data" et à "Python Data Analysis" [7].

— **Scikit-Learn** : Scikit-Learn est également appelé "sklearn", il s'agit d'une bibliothèque python, qui propose de nombreux algorithmes de classification et de régression pour l'apprentissage supervisé, tels que la forêt aléatoire et les arbres de décision, ainsi que des regroupements pour l'apprentissage non supervisé, tels que k-means.[8]

4.2.2 Ensemble de Données (DATASET)

Nous avons utilisé pour l'entrainement et le test de notre modèle d'apprentissage automatique l'ensemble de données InSDN[16]. Data-Set InSDN a été utilisé, car il a été collecté dans un SDN et renferme une combinaison de trafic normal et malveillant. Le trafic courant comprend des demandes HTTP, des pings, des demandes DNS, et ainsi de suite. Les attaques telles que les probes, le DDoS, le DoS, les attaques web, les botnets et les attaques par force brute (BFA) et U2R (user to root) font partie du trafic malveillant. Chaque flux collecté dans le data-set possède de nombreuses caractéristiques (84), et il est employé dans le développement de systèmes de détection d'intrusion (IDS).



Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts
192.168.20.133	53648	6	5/2/2020 13:58	245230	44	40	124937.0
185.127.17.56	443	6	5/2/2020 13:58	1605449	107	149	1071.0
192.168.20.2	53	6	5/2/2020 13:58	53078	5	5	66.0
192.168.20.133	35108	6	5/2/2020 13:58	6975	1	1	0.0
154.59.122.74	443	6	5/2/2020 13:58	190141	13	16	780.0
154.59.122.74	443	6	5/2/2020 13:58	4781	2	1	31.0
154.59.122.74	443	6	5/2/2020 13:58	193869	13	16	780.0
154.59.122.74	443	6	5/2/2020 13:58	1536	2	1	31.0
185.33.223.200	443	6	5/2/2020 13:58	80	1	1	0.0
104.20.110.39	443	6	5/2/2020 13:58	3609	2	2	31.0
74.125.193.97	443	6	5/2/2020 13:58	2674	2	2	31.0
199.232.26.217	443	6	5/2/2020 13:58	4751	2	2	31.0
74.125.193.95	443	6	5/2/2020 13:58	4186	2	2	31.0
192.168.20.133	34812	6	5/2/2020 13:58	3849	2	1	0.0
192.168.20.133	42648	6	5/2/2020 13:58	3082	2	1	31.0
192.168.20.133	34696	6	5/2/2020 13:58	4708	2	1	0.0
192.168.20.133	49762	6	5/2/2020 13:58	4479	2	1	0.0
192.168.20.133	47674	6	5/2/2020 13:58	23296	2	1	0.0
185.127.17.56	443	6	5/2/2020 13:58	2626158	12	12	1071.0
99.86.122.93	443	6	5/2/2020 13:58	1191	2	1	31.0
192.168.20.133	54310	6	5/2/2020 13:58	3883	3	1	0.0
209.85.203.156	443	6	5/2/2020 13:58	1741	2	2	31.0
99.86.122.96	443	6	5/2/2020 13:58	3080	2	2	31.0

FIGURE 4.3 – Affichage d'une partie de jeu de données

Attaques	**Description**
DOS	Déni de Service est une attaque visant à rendre un service indisponible en le submergeant de trafic à partir d'une seule source.
DDOS	Déni de Service Distribué est une attaque visant à rendre un service indisponible en le submergeant de trafic depuis plusieurs sources.
BFA	Brute Force Attack est une attaque informatique où l'attaquant essaye de deviner un mot de passe en essayant différentes combinaisons de manière automatisée.
Probe	Une sonde (probe) est une tentative de reconnaissance d'un système informatique pour identifier ses vulnérabilités ou ses caractéristiques. Le port-scan est l'un de ses types, consistant à scanner les ports d'une machine pour détecter les services ouverts.
U2R	est un type de cyberattaque où un attaquant commence avec l'accès à un compte utilisateur normal sur le système et est capable d'exploiter la vulnérabilité pour obtenir un accès root ou administrateur.
Web-Attack	Tentative de compromettre un site Web ou une application Web en exploitant ses vulnérabilités.
BOTNET	Un botnet est un ensemble d'ordinateurs infectés contrôlés par l'attaquant pour lancer des attaques telles que DDoS ou spam.

TABLE 4.1 – Description des attaques

4.2.3 Prétraitement et Nettoyages des Données

Nous avons traité les valeurs manquantes et encoder les variables catégorielles avec la suppression des valeurs vides sur notre DATA-SET.

```
label_to_encoded = {
    'BFA': 0,
    'BOTNET': 1,
    'DDoS': 2,
    'DoS': 3,
    'Normal': 4,
    'Probe': 5,
    'U2R': 6,
    'Web-Attack': 7
}


result['encoded_label'] = result['Label'].map(label_to_encoded)


result.drop(columns=['Label'], inplace=True)
```

FIGURE 4.4 – Prétraitement d'ensemble de données

```
NaN_rows = result.isnull().any(axis=1)
result = result.drop(NaN_rows[NaN_rows].index)
```

FIGURE 4.5 – Nettoyage de données

4.2.4 Sélection des Caractéristiques

Nous avons réduit le nombre de caractéristiques de 84 à 8 en utilisant la fonction feature-importance sur notre modèle foret aléatoire entrainé sur l'ensemble complet de DATA-SET pour sélectionner les meilleures caractéristiques parmi les 84 disponibles ainsi, nous avons créé une nouvelle colonne forwarded bytes/seconds, et aussi, on a vérifié la possibilité d'extraire ces caractéristiques depuis le protocole OFP (open flow protocol).

```python
feature_importance = rf_model.feature_importances_
feature_names = X.columns

features_with_importance = list(zip(feature_names, feature_importance))

sorted_features = sorted(features_with_importance, key=lambda x: x[1], reverse=True)

for feature_name, importance in sorted_features:
    print(f"Feature {feature_name}: Importance = {importance:.4f}")
```

```
Feature Dst Port: Importance = 0.0915
Feature Bwd Header Len: Importance = 0.0906
Feature Src Port: Importance = 0.0844
Feature Protocol: Importance = 0.0765
Feature Init Bwd Win Byts: Importance = 0.0602
Feature Pkt Len Max: Importance = 0.0381
Feature Flow IAT Max: Importance = 0.0324
Feature Pkt Len Std: Importance = 0.0263
Feature Pkt Len Var: Importance = 0.0222
Feature Bwd IAT Mean: Importance = 0.0218
Feature Bwd IAT Max: Importance = 0.0214
Feature Tot Fwd Pkts: Importance = 0.0214
```

FIGURE 4.6 – Sélection des caractéristiques (All features)

Nous avons créé une nouvelle colonne d'une caractéristique importante (Fwd-Byte/seconde) et Nous l'avons ajouté dans le jeu de données, pour l'amélioration de performance du modèle comme il s'affiche sur le code, la figure 5.7.

```python
result['fwd_bytes/s'] = (result['TotLen Fwd Pkts'] / result['Flow Duration']) * 1000000
#creation d'une nouvelle colonne nommé fwd_bytes/s calculé en divisant forwarded packet length sur flow duration
columns_to_keep = ['Src Port','Dst Port','Protocol', 'Flow Duration','Tot Fwd Pkts','TotLen Fwd Pkts','fwd_bytes/s','Fwd Pkts/s','Label

#avoir liste de toutes les colonnes
all_columns = result.columns.tolist()


columns_todrop = list(set(all_columns) - set(columns_to_keep))

result = result.drop(columns=columns_todrop)
print(result)
```

FIGURE 4.7 – Sélection des caractéristiques importantes (8 features)

Caractéristiques	**Description**
Src-port	utilisé pour identifier le service de l'émetteur
Dst-port	utilisé pour identifier le service du récepteur
Protocol	TCP, UDP, ICMP
Total fwd packets	Nombre total de paquets envoyé depuis source aux destinations
Total length of fwd packets	Nombre total de bytes envoyé depuis source aux destinations
Fwd bytes/s	Nombre de bytes envoyés par seconde
Fwd packets/s	Nombre de paquets envoyés par seconde
Label	Résultat de la classification

TABLE 4.2 – Description des caractéristiques

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	TotLen Fwd Pkts	\
0	4444	6	269709	4	48.0	
1	4444	6	268599	2	0.0	
2	3632	6	22194	5	53.0	
3	8180	6	9556	4	30.0	
4	8180	6	8782	4	30.0	
...	...	...	...	...	...	
343884	80	6	296	1	0.0	
343885	80	6	3431	2	633.0	
343886	80	6	4121	1	0.0	
343887	80	6	5887	2	671.0	
343888	80	6	150	1	0.0	

	Fwd Pkts/s	Label	fwd_bytes/s
0	14.830799	U2R	177.969589
1	7.446044	U2R	0.000000
2	225.286113	U2R	2388.032802
3	418.585182	BFA	3139.388866
4	455.477112	BFA	3416.078342
...	...	...	...
343884	3378.378378	Web-Attack	0.000000
343885	582.920431	Web-Attack	184494.316526
343886	242.659549	Web-Attack	0.000000
343887	339.731612	Web-Attack	113979.955835
343888	6666.666667	Web-Attack	0.000000

FIGURE 4.8 – Jeu de données après prétraitement et sélection des caractéristiques

4.2.5 Entrainement du Modèle d'Apprentissage Automatique

D'abord, nous avons divisé l'ensemble des données sur deux l'entrainement 70% et le test 30% Ensuite, nous avons créé notre modèle classificateur avec l'algorithme Forêt aléatoire (Random Forest) et la méthode Grid-Searchcv pour trouver les meilleurs hyper-paramètres c'est une méthode de recherche sur grille fournie par la bibliothèque scikit-learn combiné avec la validation croisée.

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import GridSearchCV

X = result.drop('encoded_label', axis=1)
y = result['encoded_label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

grid_search = GridSearchCV(RandomForestClassifier(), param_grid=param_grid, cv=skf, scoring='accuracy')

grid_search_rf.fit(X_train, y_train)

print("Best Hyperparameters:", grid_search_rf.best_params_)
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
Best Hyperparameters: {'max_depth': 20, 'max_features': 'log2', 'n_estimators': 200}
```

FIGURE 4.9 – Entrainement de modèle ML avec l'algorithme RF et la méthode Grid-Search

4.2.6 Évaluation des Performances

Les résultats de l'évaluation des performances du classificateur RF sur l'ensemble de données réservée pour le test sont présentés dans les figures 5.10 où ils représentent respectivement les paramètres d'évaluation et la matrice de confusion.

```
Accuracy: 0.99443620537575
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       411
           1       1.00      1.00      1.00        41
           2       1.00      1.00      1.00     36673
           3       0.99      0.97      0.98     15934
           4       1.00      1.00      1.00     20557
           5       0.99      1.00      0.99     29499
           6       1.00      0.75      0.86         4
           7       1.00      0.94      0.97        48

    accuracy                           0.99    103167
   macro avg       1.00      0.96      0.97    103167
weighted avg       0.99      0.99      0.99    103167
```

FIGURE 4.10 – Rapport de classification

Voici notre matrice de confusion de modèle IDS dans la figure 5.11.

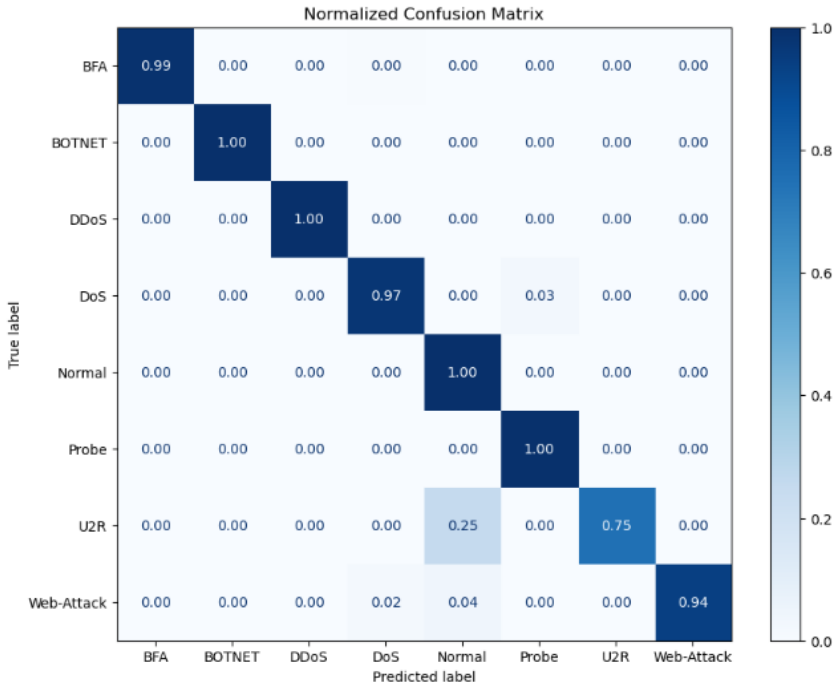
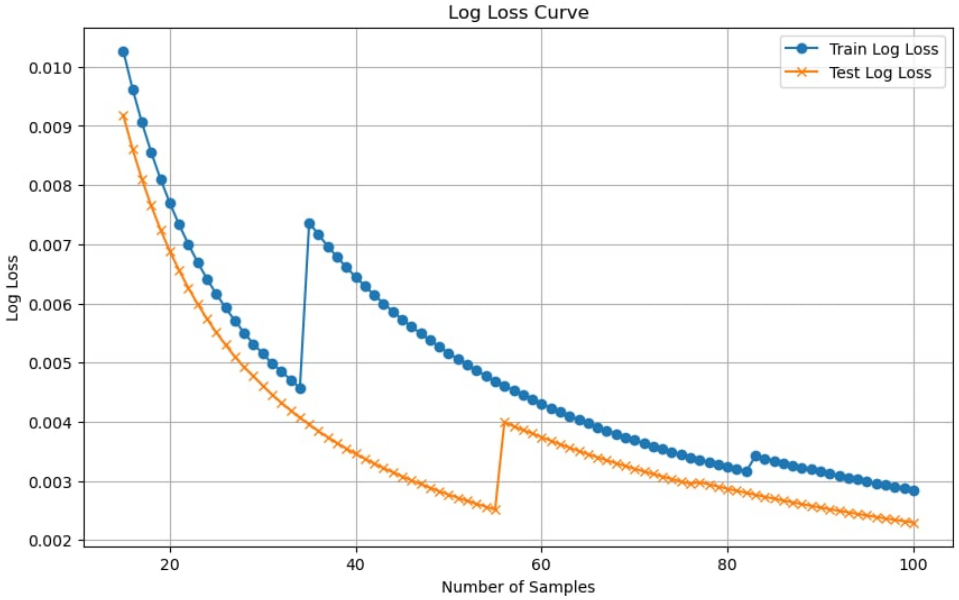


FIGURE 4.11 – Matrice de confusion

La figure 4.12 montre la perte logarithmique qui diminue progressivement à chaque itération, indiquant une amélioration continue de la performance du modèle sur les données d'entraînement et du test, ce qui est un bon indicateur de la généralisation du modèle.



cv

FIGURE 4.12 – Courbes de perte pour l'ensemble de données de l'entrainement et de test

4.2.7 Validation Croisée

Il est utilisé en apprentissage automatique pour évaluer la capacité de généralisation d'un modèle à un ensemble de données indépendant. C'est une technique pour évaluer les modèles de ML en entrainant plusieurs modèles de ML sur des sous-ensembles des données d'entrée disponibles et en les évaluant sur le sous-ensemble complémentaire des données. Elle aide à diminuer le surajustement (overfiting) et donne un aperçu de la façon dont le modèle se généralisera a un jeu de données indépendant.

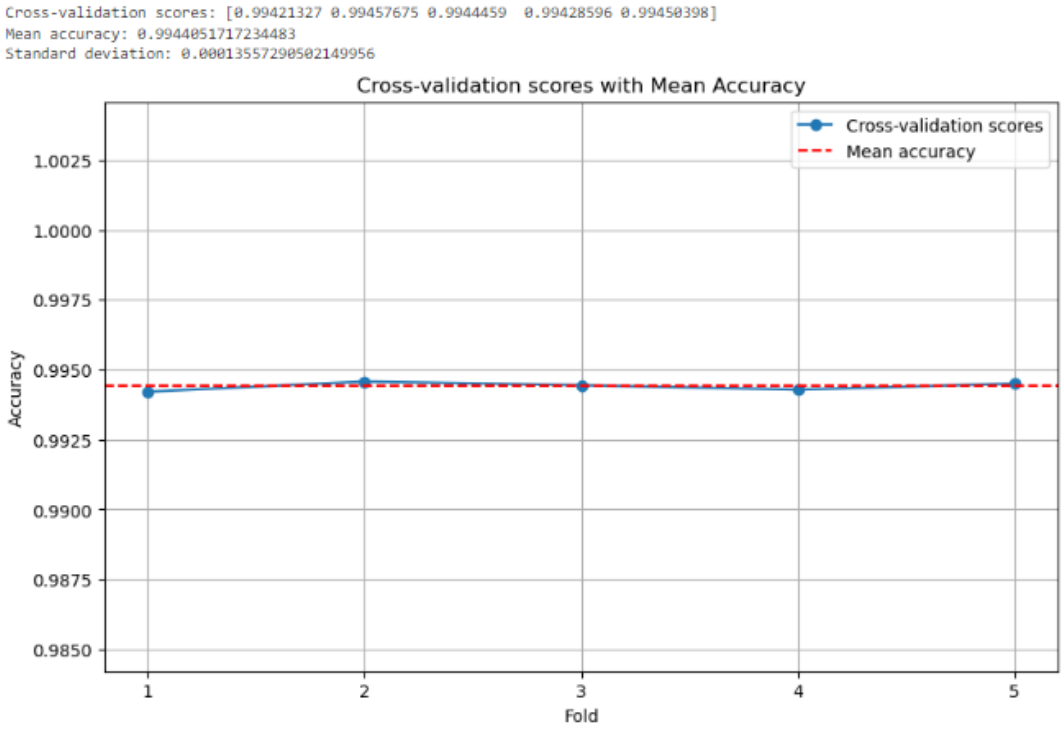


FIGURE 4.13 – Résultats de validation croisée avec 5 échantillons (accuracy scores)

	precision	recall	f1-score	support
BFA	0.00	0.00	0.00	0
BOTNET	0.00	0.00	0.00	0
DDoS	0.00	0.00	0.00	0
Normal	0.99	0.91	0.95	328699
Probe	0.98	0.92	0.95	79494

FIGURE 4.14 – Test avec un jeu de données contenant 2 types de trafique (Normal, Probe)

4.3 Test et Validation IDS avec Prévention

4.3.1 Mininet

Mininet est un logiciel open-source qui est utilisé pour simuler des réseaux définis par logiciel SDN. Ce logiciel utilise un système de contrôleurs, des Switch et d'hôtes, son fonctionnement repose sur le noyau de Linux et utilise Python comme API. Un outil peut être utilisé pour simuler une configuration réseau en fonction complète, il offre l'avantage de l'individuation, et la création des topologies établie sur le choix de l'utilisateur au lieu d'être limité, par exemple par le paramètre du nombre de commutateurs achetés ou espace physique.

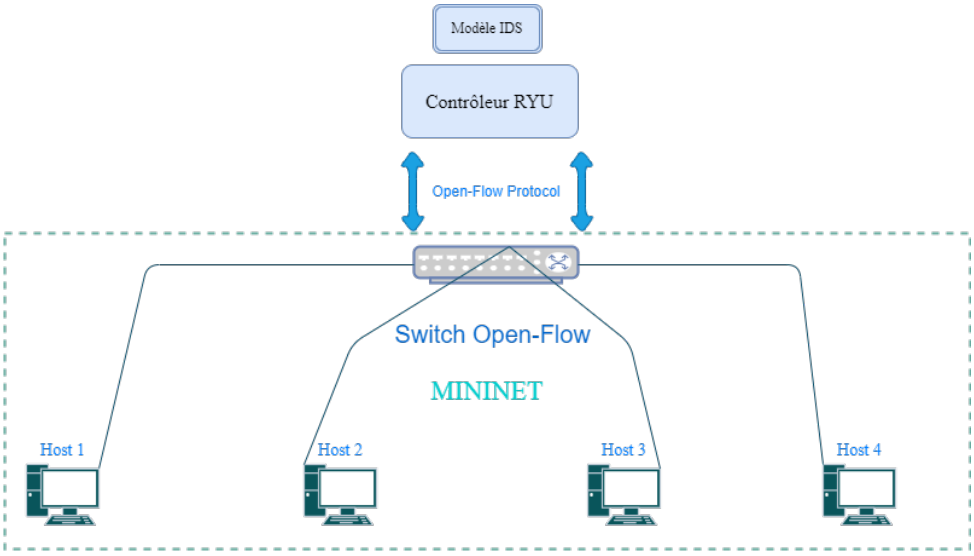


FIGURE 4.15 – Topologie de test sur mininet

Nous avons simulé sur Mininet avec une topologie juste pour le test de quatre hôtes et une Switch Open-Flow figure 4.15

4.3.2 Importation de Modèle IDS

Nous avons importé notre modèle apprentissage automatique à l'aide de la bibliothèque PICKLE pour le charger sur notre application RYU et l'utiliser dans l'architecture SDN pour la détection des intrusions. Cette approche vous permet de sérialiser l'objet modèle dans un fichier et de le désérialiser ultérieurement.

```
import pickle
pickle.dump(rf_model, open(r'C:\Users\HP\Downloads\Compressed\InSDN_DatasetCSV\InSDN_DatasetCSV\modele3.pkl', 'wb'))
```

FIGURE 4.16 – Importation de modèle IDS

Nous avons créé la topologie sur le simulateur MININET et faire la configuration nécessaire pour ajouter les Switchs et les Hôtes et configurer la connexion avec le contrôleur RYU et lancer le contrôleur avec le script python (ids1.py) qui contient notre application responsable de la détection des intrusions utilisant notre modèle, ainsi que des mécanismes de prévention. Il est programmé en Python.

FIGURE 4.17 – Création de la topologie sur mininet

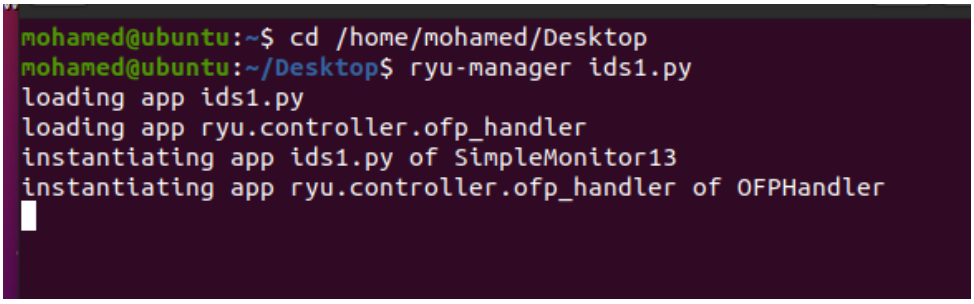


FIGURE 4.18 – Lancement de Ryu avec notre application IDS

4.3.3 Test de Trafique Normal

On a simulé le trafique normal avec des pings et avec des requêtes HTTP entre les hôtes

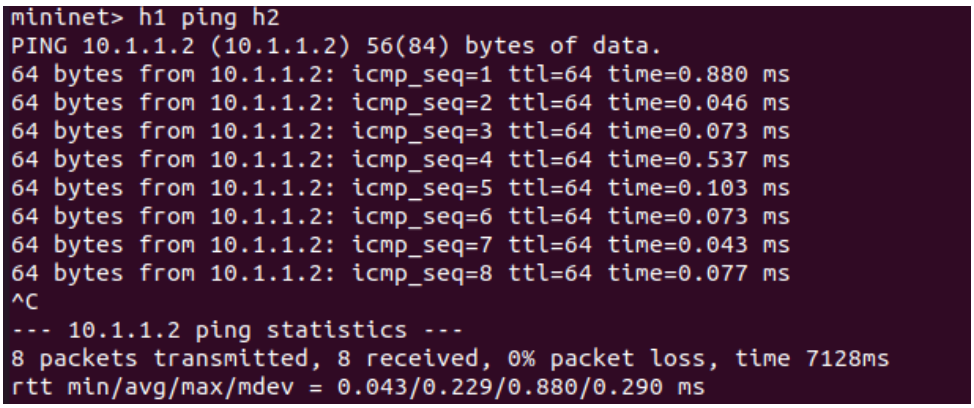


FIGURE 4.19 – Test de Ping entre H1 et H2

Nous avons ouvert un serveur web HTTP sur host H1 et puis on a accédé depuis l'hôte H2 avec la commande CURL

FIGURE 4.20 – Serveur web HTTP sur l'hôte

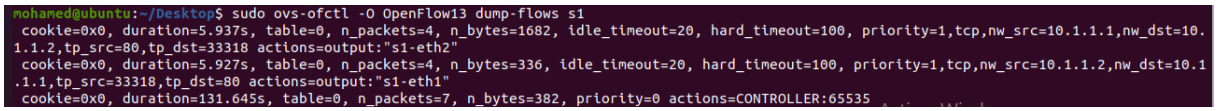


FIGURE 4.21 – Table de flux après la requête HTTP

Ensuite, on a le résultat de détection de notre modèle IDS sur les paquets du trafique normal après la requête de statistique et sa réponse sur le contrôleur et l'extraction des caractéristiques des flux.

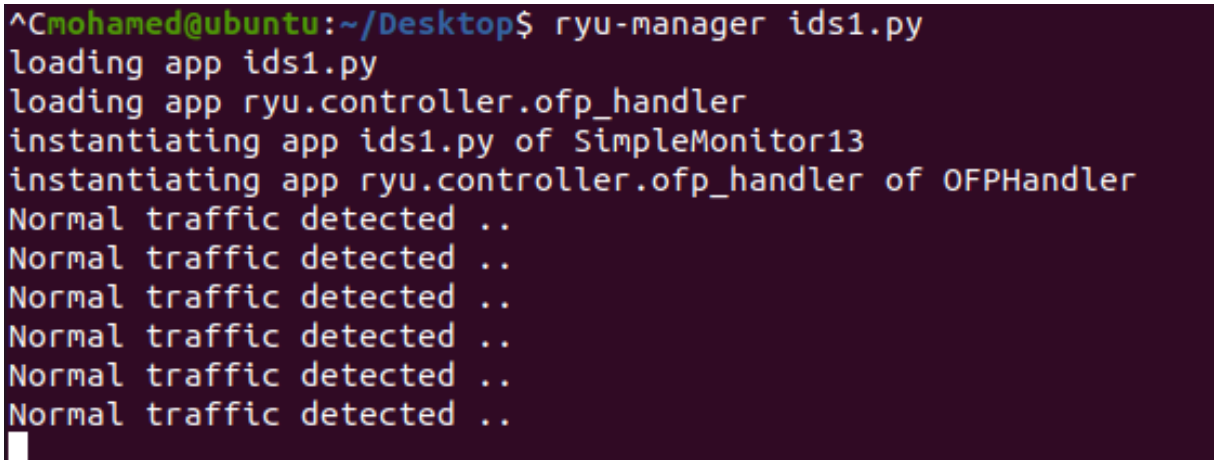


FIGURE 4.22 – La détection de trafique normal

4.3.4 Test de Trafique Malveillant

4.3.4.1 Les Outils Utilisés pour le Test des Attaques

— **Hping3 :** est un outil utilisé pour simuler des attaques DDOS et elle peut être configuré pour utiliser des sources IP adresse aléatoires (IP spoofing).

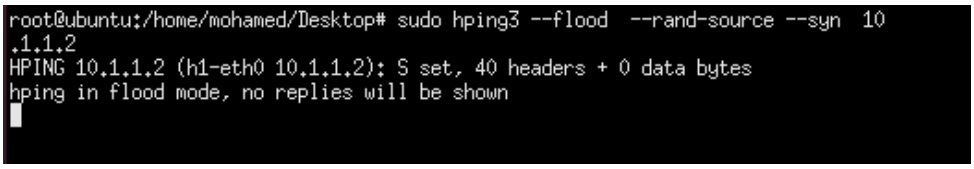


FIGURE 4.23 – Commande de Hping3

— **Nmap :** est un outil puissant qui a été créé pour des fins légitimes, mais peut être utilisé pour simuler des attaques comme le port scan pour découvrir les ports ouvertes dans la machine cible et c'est un type d'attaques de sondage (Probe attack).

FIGURE 4.24 – Commande de Nmap

— **Nping :** outil pour simuler des attaques dos, il offre contrôle total sur la génération des paquets, on peut spécifier le type, la taille des paquets envoyé

4.3.4.2 Résultat de Détection de Trafique Malveillant

Après la détection des attaques, on a les résultats suivant les figures :

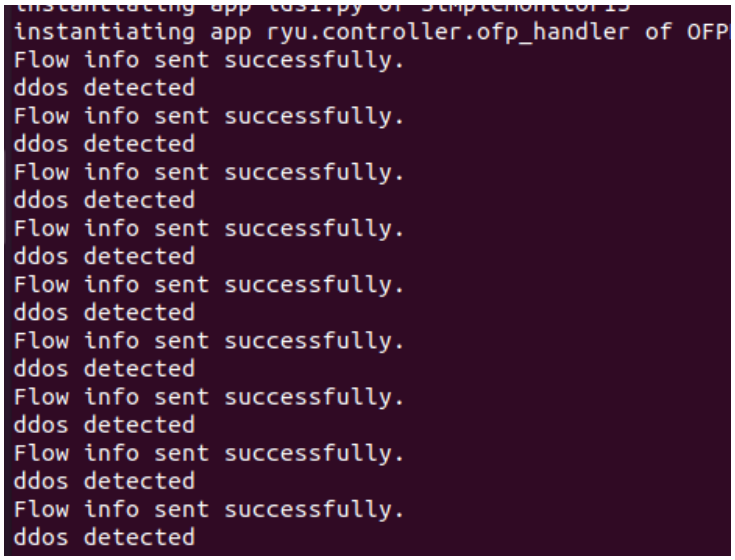


FIGURE 4.25 – Résultat de détection DDoS

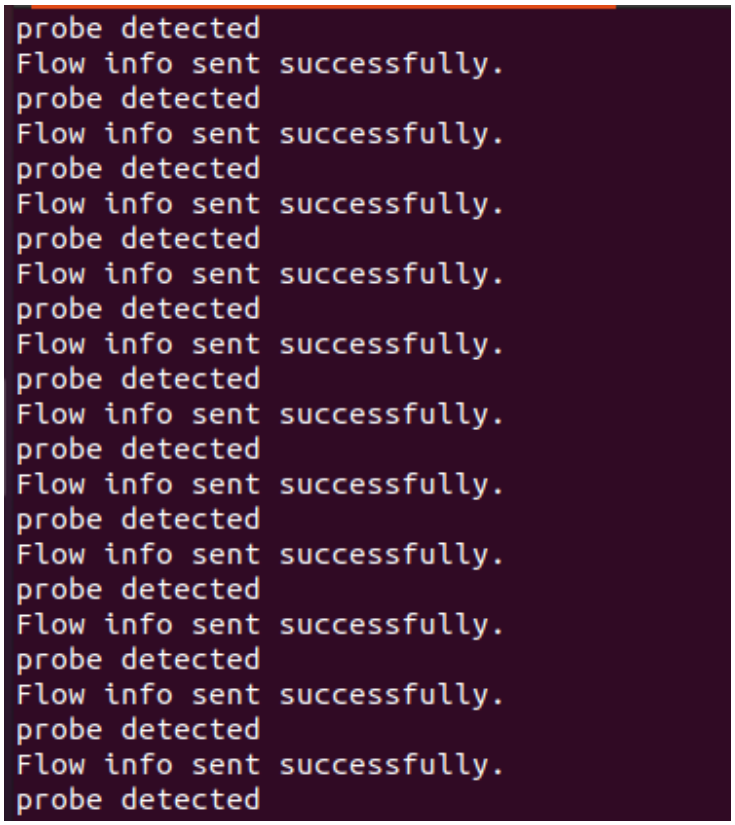


FIGURE 4.26 – Résultat de détection PROBE

4.3.5 Prévention des Attaques Détectées

Après un flux est détecté malveillant, une nouvelle règle de flux est installée au niveau de switch pour bloquer les futurs paquets de flux (Deny Trafic) donc le switch va acter comme un firewall ainsi le flux est supprimé de la table des flux.

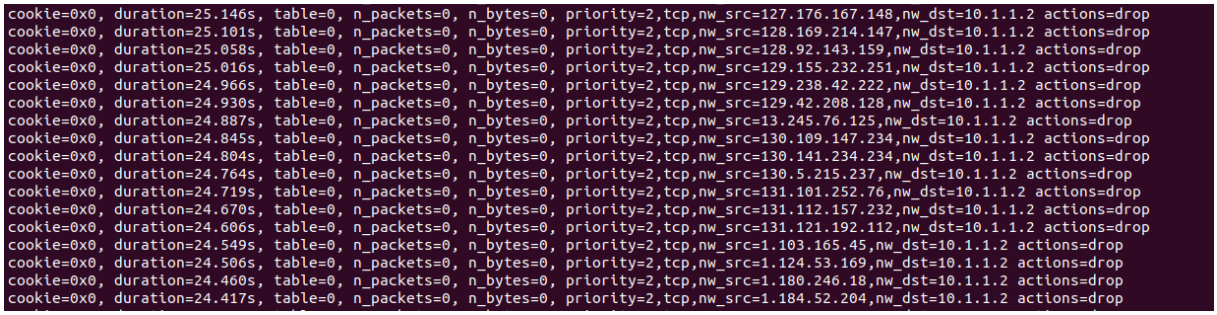


FIGURE 4.27 – Table de flux après la détection des attaques

4.3.6 Logs des Flux Détecté Malveillant

Cette interface affiche les informations sur les flux détectés malveillants (IP-Source, IP-Destination) avec date et l'heure de détection en temps réel Pour la même ligne de log, on affiche

à l'administrateur le résultat de détection (Type d'attaque), ce qui permet à l'administrateur de prendre d'autres décisions de prévention, Cette interface offre aussi à l'administrateur un ensemble de fonctionnalités avancées de filtrage.

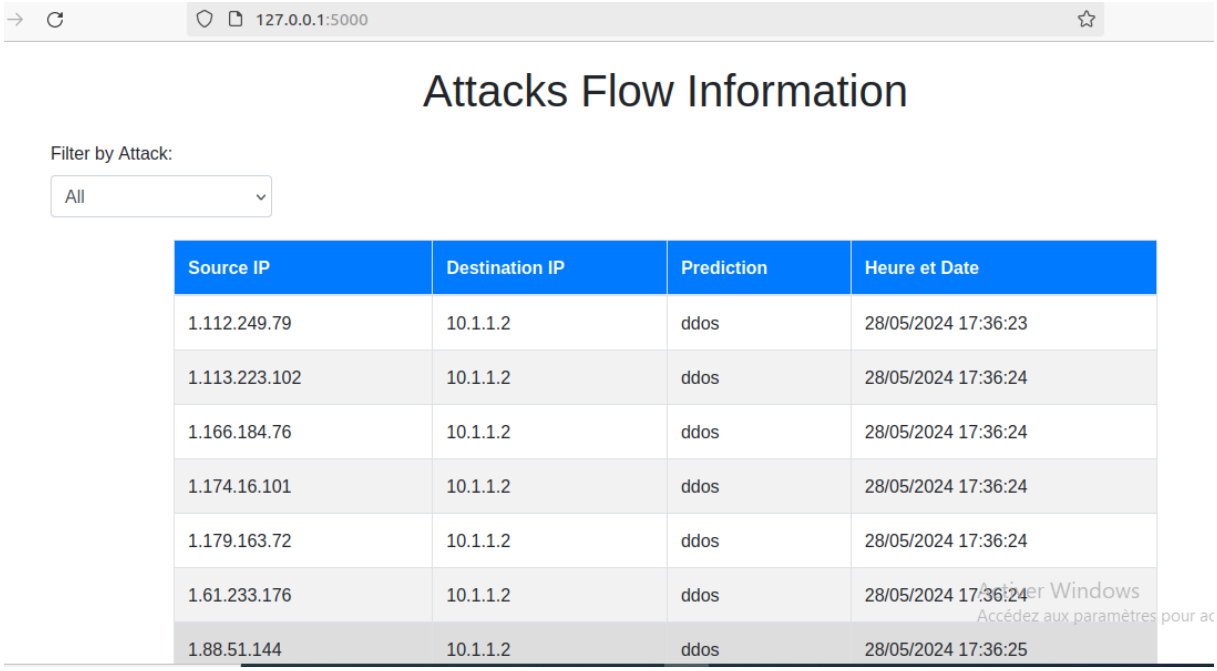


FIGURE 4.28 – Interface web pour les Logs

4.4 Conclusions

Dans ce chapitre, nous avons expliqué les fonctionnalités des différentes parties de notre architecture du système et comment elles sont mises en œuvre ensemble pour donner les meilleurs résultats en matière de détection des attaques et des anomalies, d'abord, nous avons créé un modèle d'apprentissage automatique ensuite, nous avons chargé ce modèle sur le Contrôleur RYU À la fin du chapitre, nous avons exposé les diffèrent tests sur notre projet IDS avec prévention.

Conclusion et Perspectives

4.5 Conclusion Générale

Cette thèse démontre avec succès le développement et l'implémentation d'un système de détection d'intrusion (IDS) avec prévention de haute précision utilisant des techniques d'apprentissage automatique (ML) et des technologies de SDN. En utilisant l'algorithme foret aléatoire (RF) et la méthode de recherche par grille combinée avec la validation croisée (GridSearchCV), le système a atteint une précision de 99%, soulignant son efficacité dans l'identification des intrusions. Le modèle a été entrainé à l'aide de jeu de données INSDN et, malgré l'utilisation de seulement 8 caractéristiques, il a réussi à maintenir des performances élevées. L'intégration de cet IDS basé sur ML dans un environnement SDN permet une détection et une réponse en temps réel aux intrusions sur le réseau. Lors de la détection d'une intrusion, le système est capable de déployer des mesures préventives, telles que l'installation dynamique de règles de pare-feu et alerter l'administrateur, pour atténuer les dommages potentiels. La flexibilité et la programmabilité du SDN étaient essentielles pour intégrer la détection par apprentissage automatique avec des réponses en temps réel, renforçant la sécurité du réseau.

4.6 Futurs Travaux

Pour nos futurs travaux, notre objectif est d'utiliser des méthodes d'apprentissage profond ou semi-supervisé afin de créer notre IDS. Cela permettra au modèle de détecter de nouvelles intrusions qu'il n'avait pas rencontrées pendant son entraînement, améliorant sa capacité à s'adapter à de nouvelles menaces et augmentant la performance du modèle tel que la précision rappel et F1-Score.

Bibliographie

[1] Floodlight controller - confluence. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller Consulté le 20-04-2024.

[2] Getting started — ryu 4.34 documentation. https://ryu.readthedocs.io/en/latest/getting_started.html Consulté le 20-04-2024.

[3] OpenDaylight. https://www.opendaylight.org/ Consulté le 20-04-2024.

[4] Ryu sdn framework | PPT. https://fr.slideshare.net/yamahata/ryu-sdnframeworkupload Consulté le 20-04-2024.

[5] NumPy -, 2024. https://numpy.org./ Consulté le 03-06-2024.

[6] Openflow efort PDF | PDF | protocoles internet | ethernet, 2024. https ://fr.scribd.com/document/478676025/OPENFLOW-EFORT-pdf Consulté le 15-04-2024.

[7] Pandas - python data analysis library, 2024. https://pandas.pydata.org/ Consulté le 03-06-2024.

[8] Scikit-learn : machine learning in python — scikit-learn 1.5.0 documentation, 2024. https://scikit-learn.org/stable/ Consulté le 03-06-2024.

[9] Stefan Axelsson. Intrusion detection systems : A survey and taxonomy. 2000.

[10] Hacene Bellahmer. *Implémentation et évaluation d'un modèle d'apprentissage automatique pour l'estimation de la valeur marchande de propriétés immobilières*. PhD thesis, Université Mouloud Mammeri, 2020.

[11] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[12] E Canceres. Le protocole openflow dans l'architecture sdn. http://www.efort.com/r_tutoriels/OPENFLOW_EFORT.pdf, 2016. Consulté le 20-04-2024.

[13] Ihssane Choukri, Mohammed Ouzzif, and Khalid Bouragba. Software defined networking (SDN) : Etat de l'art. In *Colloque sur les Objets et systèmes Connectés*, 2019.

[14] Anderson Santos Da Silva, Juliano Araujo Wickboldt, Lisandro Zambenedetti Granville, and Alberto Schaeffer-Filho. Atlantic : A framework for anomaly traffic detection, classification, and mitigation in sdn. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pages 27–35. IEEE, 2016.

[15] Jérôme Durand. Le SDN pour les nuls. 2015. https://hepia.infolibre.ch/Virtualisation-Reseaux/Le_SDN_pour_Les_Nuls_Jerome_Durand_JRES_2015.pdf Consulté le 03-13-2024.

[16] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D. Jurcut. Insdn : A novel sdn intrusion dataset. *IEEE Access*, 8 :165263–165284, 2020.

[17] Evelyn Fix. *Discriminatory analysis : nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

[18] Benamrane Fouad and M. Mouad. Etude des performances des architectures du plan de contrôle des réseaux 'software-defined networks.', 2017.

[19] GeeksforGeeks. Random forest algorithm in machine learning, 2024. Section : AI-ML-DS.

[20] Nutan Farah Haq, Abdur Rahman Onik, Md Avishek Khan Hridoy, Musharrat Rafni, Faisal Muhammad Shah, and Dewan Md Farid. Application of machine learning approaches in intrusion detection system : a survey. *IJARAI-International Journal of Advanced Research in Artificial Intelligence*, 4(3) :9–18, 2015.

[21] Doan Hoang. Software defined networking-shaping up for the next disruptive step ? *Journal of Telecommunications and the Digital Economy*, 3(4) :48–62, 2015.

[22] Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. Machine learning in iot security : Current solutions and future challenges. *IEEE Communications Surveys & Tutorials*, 22(3) :1686–1721, 2020.

[23] Sukhveer Kaur, Japinder Singh, and Navtej Ghumman. Network programmability using POX controller. 2014.

[24] Gulshan Kumar and Hamed Alqahtani. Machine learning techniques for intrusion detection systems in SDN-recent advances, challenges and future directions. 134(1) :89–119, 2022. Publisher : Tech Science Press.

[25] Gulshan Kumar, Krishan Kumar, and Monika Sachdeva. The use of artificial intelligence based techniques for intrusion detection : a review. *Artificial Intelligence Review*, 34 :369–387, 2010.

[26] Zhenpeng Liu, Yihang Wang, Fan Feng, Yifan Liu, Zelin Li, and Yawei Shan. A DDoS detection method based on feature engineering and machine learning in software-defined networks. 23(13) :6176, 2023.

[27] Rahim Masoudi and Ali Ghaffari. Software defined networks : A survey. 67 :1–25, 2016.

[28] Rachid Mifdal. *Application des techniques d'apprentissage automatique pour la prédiction de la tendance des titres financiers*. PhD thesis, École de technologie supérieure, 2019.

[29] Saurav Nanda, Faheem Zafari, Casimer DeCusatis, Eric Wedaa, and Baijian Yang. Predicting network attack patterns in sdn using machine learning approach. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 167–172. IEEE, 2016.

[30] Lehat Narimen. Generalities ids avec ml. 2022.

[31] Suad Othman, Taha Nabeel, Fadl Ba-Alwi, and Ammar Zahary. Survey on intrusion detection system types. 2018.

[32] Chowdhury Mofizur Rahman, Dewan Md Farid, and Mohammad Zahidur Rahman. Adaptive intrusion detection based on boosting and naïve bayesian classifier. 2011.

[33] Suchismita Rout, Kshira Sagar Sahoo, Sudhansu Sekhar Patra, Bibhudatta Sahoo, and Deepak Puthal. Energy efficiency in software defined networking : A survey. 2(4) :308, 2021.

[34] Rafath Samrin and D Vasumathi. Review on anomaly based network intrusion detection system. In *2017 international conference on electrical, electronics, communication, computer, and optimization techniques (ICEECCOT)*, pages 141–147. IEEE, 2017.

[35] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc : a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[36] Chungsik Song, Younghee Park, Keyur Golani, Youngsoo Kim, Kalgi Bhatt, and Kunal Goswami. Machine-learning based threat-aware system in software defined networks. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–9. IEEE, 2017.

[37] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep recurrent neural network for intrusion detection in sdn-based networks. In *2018 4th*

IEEE Conference on Network Softwarization and Workshops (NetSoft), pages 202–206. IEEE, 2018.

[38] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning : A review. *expert systems with applications*, 36(10) :11994–12000, 2009.

[39] Ping Wang, Kuo-Ming Chao, Hsiao-Chung Lin, Wen-Hui Lin, and Chi-Chun Lo. An efficient flow control approach for sdn-based network threat detection and migration using support vector machine. In *2016 IEEE 13th international conference on e-business engineering (ICEBE)*, pages 56–63. IEEE, 2016.

[40] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136) :1–46, 2017.

Annexe A

Critères d'évaluation des modèles ML

Faces aux diverses méthodes d'apprentissage, plusieurs critères d'évaluation et de comparaison des performances des modèles ont été créés. Cette évaluation consiste à une évaluation empirique des modèles. Pour valider ou tester un classifieur, l'ensemble de données est généralement divisé en deux sous ensemble : l'ensemble d'apprentissage sur lequel le classificateur fait son apprentissage et l'ensemble de test sur lequel nous pouvons évaluer sa performance. La performance du classifieur peut être analysée en considérant les erreurs commises sur l'ensemble de test. Car ce dernier contient les données dont on connaît à l'avance les classes auxquelles elles devraient appartenir. Nous pourrons donc comparer les décisions prises par le classificateur automatique à celles des experts humains et calculer un score de performance. Dans ce contexte, nous définissons les métriques que nous avons utilisées pour évaluer nos modèles :

A.1 Matrice de confusion

Il s'agit d'une matrice qui est souvent utilisée pour décrire le résultat des prédictions d'un modèle d'IA (classificateur qui impliquent deux classes ou plus en sortie) dans l'ensemble de test, avec des détails sur chaque classe en utilisant les paramètres suivants.[35]

- **Vrais Positifs (VP)** : prédits positifs et réellement positifs.
- **Faux Positifs (FP)** : prédits positifs et réellement négatifs.
- **Vrais Négatifs(VN)** : prédits négatifs et réellement négatifs.
- **Faux Négatifs (FN)** : Prédits négatifs et en réalité positifs.

La matrice indique le nombre de prédictions correctes et incorrectes pour chaque classe en fonction de la classe prédite. Le tableau 4 montre un exemple de matrice de confusion pour la classification binaire.

	Prédictions	
	Positives (P)	**Négatives (N)**
Réellement Positives (P)	Vrais Positifs (VP)	Faux Négatifs (FN)
Réellement Négatives (N)	Faux Positifs (FP)	Vrais Négatifs (VN)

TABLE A.1 – Matrice de confusion

A.2 Accuracy

L'accuraccy, est la fréquence à laquelle le classificateur (le modèle) fait la bonne prédiction, dans l'ensemble de test, en utilisant l'ensemble d'entrainement (c'est-à dire que le modèle peut généraliser de nouvelles données en utilisant seulement les données d'entrainement) [35]. Il s'agit de la proportion de classifications correctes par rapport au nombre total de classifications effectuées.

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \tag{A.1}$$

A.3 Précision

Elle évalue le pouvoir prédictif d'un modèle d'IA en estimant la valeur prédictive d'une étiquette, autrement dit la capacité d'un modèle à ne pas identifier de faux positifs. C'est le rapport entre les instances positives (négatives) et le total des instances positives (négatives) prédites[35].

$$\text{Précision} = \frac{VP}{VP + FP} \tag{A.2}$$

A.4 Rappel

Évalue l'efficacité du modèle dans une classe spécifique en calculant la probabilité que l'étiquette positive soit vraie[35]. Différemment dit, c'est la capacité du modèle à ne pas rater de vrais positifs. C'est le rapport entre les instances positives et le total des instances positives réelles.

$$\text{Rappel} = \frac{VP}{VP + FN} \tag{A.3}$$

A.5 F1 Score

Il prend en compte la contribution de la précision et du rappel, de sorte que plus la valeur F1 est élevée, meilleur est le résultat [35]. Le modèle obtient un F1 élevé lorsque les prédictions positives sont réellement positives (précision) et que le modèle ne manque pas beaucoup de valeurs positives (rappel).

$$F1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \tag{A.4}$$

Annexe B

Code de l'Application RYU Contrôleur

Voici notre script python de l'application RYU en trois parties :

```python
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub
import switch
import pickle
import numpy as np
from sklearn.ensemble import RandomForestClassifier
import requests
import pandas as pd
from datetime import datetime


class SimpleMonitor13(switch.SimpleSwitch13):

    def __init__(self, *args, **kwargs):
        super(SimpleMonitor13, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self._monitor)
        self.loaded_model = None

    @set_ev_cls(ofp_event.EventOFPStateChange,
                [MAIN_DISPATCHER, DEAD_DISPATCHER])
    def _state_change_handler(self, ev):
        datapath = ev.datapath
        if ev.state == MAIN_DISPATCHER:
            if datapath.id not in self.datapaths:
                self.logger.debug('register datapath: %016x', datapath.id)
                self.datapaths[datapath.id] = datapath
        elif ev.state == DEAD_DISPATCHER:
            if datapath.id in self.datapaths:
                self.logger.debug('unregister datapath: %016x', datapath.id)
                del self.datapaths[datapath.id]

    def _monitor(self):
        while True:
            for dp in self.datapaths.values():
                self._request_stats(dp)
            hub.sleep(20)

    def _load_model(self):
        # Load the pickled model
        loaded_model = pickle.load(open('/home/mohamed/Desktop/modele2.pkl', 'rb'))
        return loaded_model
```

FIGURE B.1 – Script Ids1.py partie 1

```
    def _request_stats(self, datapath):
        self.logger.debug('send stats request: %016x', datapath.id)
        parser = datapath.ofproto_parser
        req = parser.OFPFlowStatsRequest(datapath)
        datapath.send_msg(req)
    def delete_flow(self, datapath, eth_type, ip_src, ip_dst, protocol):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser

    # Define the match criteria for the flow
        match = parser.OFPMatch(eth_type=eth_type,
                                ipv4_src=ip_src, ipv4_dst=ip_dst,
                                ip_proto=protocol)

    # Create a flow mod message to delete the flow
        flow_mod = parser.OFPFlowMod(
        datapath=datapath,
        command=ofproto.OFPFC_DELETE,
        out_port=ofproto.OFPP_ANY,
        out_group=ofproto.OFPG_ANY,
        match=match
        )

    # Send the flow mod message to the switch
        datapath.send_msg(flow_mod)
    def add_drop_flow(self, datapath, priority, eth_type, ip_src, ip_dst, protocol):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser

        match = parser.OFPMatch(
        eth_type=eth_type,
        ipv4_src=ip_src,
        ipv4_dst=ip_dst,
        ip_proto=protocol
          )

        instructions = []

        mod = parser.OFPFlowMod(
        datapath=datapath,
        priority=priority,
        match=match,
        instructions=instructions
           )

        datapath.send_msg(mod)
```

FIGURE B.2 – Script Ids1.py partie 2

```
    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
    def _flow_stats_reply_handler(self, ev):
        body = ev.msg.body
        datapath = ev.msg.datapath
        for stat in sorted([flow for flow in body if (flow.priority == 1) and ('eth_type' in
flow.match)],
                           key=lambda flow: (flow.match.get('eth_type', 0), flow.match.get('ipv4_src',
'0.0.0.0'),
                                             flow.match.get('ipv4_dst', '0.0.0.0'),
flow.match.get('ip_proto', 0))):
            dp_id = ev.msg.datapath.id
            ip_src=stat.match.get('ipv4_src')
            ip_dst=stat.match.get('ipv4_src')
            protocol=stat.match.get('ip_proto')
            eth_type=stat.match.get('eth_type')
            ip_proto=stat.match.get('ip_proto')
            if ip_proto == 6:
                tp_src = stat.match.get('tcp_src')
                tp_dst = stat.match.get('tcp_dst')
            elif ip_proto == 17:
                tp_src = stat.match.get('udp_src')
                tp_dst = stat.match.get('udp_dst')
            else:
                tp_src = 0
                tp_dst = 0
                ip_proto = 0

            try:
                packet_count_per_second = stat.packet_count / stat.duration_sec
            except ZeroDivisionError:
                packet_count_per_second = 0
            try:
                byte_count_per_second = stat.byte_count / stat.duration_sec
            except ZeroDivisionError:
                byte_count_per_second = 0
            duration_sec = stat.duration_sec
            duration = duration_sec * 1000000

            flow_data = [tp_src, tp_dst, ip_proto, duration, stat.packet_count, stat.byte_count,
                         packet_count_per_second, byte_count_per_second]
            model = self._load_model()
            flow_data1 = np.array(flow_data)
            flow_data2 = flow_data1.reshape(1, -1)
            data_for_prediction = pd.DataFrame(flow_data2, columns=model.feature_names_in_)
            prediction = model.predict(data_for_prediction)

            prediction1 = int(prediction)
            now = datetime.now()
            time = now.strftime("%d/%m/%Y %H:%M:%S")
            if prediction1 in [0, 1, 2, 3, 5, 6, 7]:
                if prediction1 == 0:
                    prediction2 = 'bfa'
                elif prediction1 == 1:
                    prediction2 = 'botnet'
                elif prediction1 == 2:
                    prediction2 = 'ddos'
                elif prediction1 == 3:
                    prediction2 = 'dos'
                elif prediction1 == 4:
                    prediction2 = 'Normal'
                elif prediction1 == 5:
                    prediction2 = 'probe'
                elif prediction1 == 6:
                    prediction2 = 'u2r'
                elif prediction1 == 7:
                    prediction2 = 'web attack'
                flow_info = {
                    'ip_src': stat.match.get('ipv4_src'),
                    'ip_dst': stat.match.get('ipv4_dst'),
                    'prediction': prediction2 ,
                    'prediction1': prediction1 ,
                    'date' : time
                }
                self._send_flow_info(flow_info)
                self.logger.warning("%s detected",prediction2)
                self.delete_flow(datapath, eth_type, stat.match.get('ipv4_src'),
stat.match.get('ipv4_dst'), protocol)
                self.add_drop_flow(datapath, 2, eth_type, stat.match.get('ipv4_src'),
stat.match.get('ipv4_dst'), protocol)
            else:
                self.logger.info("Normal traffic detected")
    def _send_flow_info(self, flow_info):
        try:
            response = requests.post('http://127.0.0.1:5000/api/flow_info', json=flow_info)
            if response.status_code == 200:
                self.logger.info("Flow info sent successfully.")
            else:
                self.logger.error("Failed to send flow info. Status code: %s", response.status_code)
        except requests.exceptions.RequestException as e:
            self.logger.error("Error sending flow info: %s", e)
```

FIGURE B.3 – Script Ids1.py partie 3