Royaume du Maroc
UNIVERSITÉ MOHAMED V - RABAT

ECOLE NATIONALE SUPÉRIEURE D'INFORMATQUE
ET D'ANALISE DES SYSTÈMES

Projet de fin de deuxième année

Smart IDS

Filière : Sécurité des systèmes d'information (SSI)

Réalisé par :

AGOULZI Imane
JOUIJATE Rim

Encadré par :

M. BERQIA Amine

Membre de Jury :

M. ERRADI Mohamed

Année universitaire : 2022 - 2023

Remerciements

Louange à ALLAH seul, que ses bénédictions soient sur notre seigneur et maitre Mohamed et sur les siens.

Tout d'abord nous souhaitons témoigner notre immense gratitude et notre profonde reconnaissance à Monsieur **M. BERQIA Amine** , notre encadrant, qui nous a accompagné durant la réalisation du projet et qui n'a épargné aucun effort afin d'assurer une bonne qualité de travail. Leur conseils avisés, leur large expérience et surtout leur encouragement incessant nous ont été d'une aide précieuse.

Nos gratifications sont aussi adressées à tout le corps professoral et à toute l'équipe pédagogique de l'École Nationale Supérieur d'Informatique et d'Analyse des Systèmes de nous avoir passé une meilleure formation qui nous a permis d'exécuter nos missions de manière juste et efficace.

Et pour terminer, nous n'oublions pas d'adresser nos vifs remerciements à nos familles qui nous offrent l'énergie et le soutient nécessaire pour mener à bien nos projets.

Résumé

Ce travail vise à découvrir les systèmes de détection d'intrusion et leur fonctionnement lors de la surveillance d'un réseau.

Pour étudier de près cet outil de surveillance, une solution simple et efficace sera proposée en combinant le célèbre nano-ordinateur : le Raspberry Pi et le fameux système de détection d'intrusion : Suricata. Il est vrai que cette solution concerne les petits réseaux mais elle peut s'étendre après aux réseaux à grande échelle.

Ensuite, on examinera l'impact de l'intelligence artificielle et les techniques de Machine Learning sur ces systèmes qui peuvent devenir plus intelligents et plus performants. En essayant finalement de construire un modèle basé ces nouvelles technologies pour une atteindre un niveau élevé de sécurité réseau.

Abstract

The aim of this project is to discover intrusion detection systems and how they work when monitoring a network.

To take a closer look at this monitoring tool, we'll be proposing a simple and effective solution combining the famous nano-computer : Raspberry Pi and the famous intrusion detection system : Suricata .Although this solution is designed for small-scale networks, it could be extended to large-scale networks in the future.

In addition, we will look at the impact of artificial intelligence and Machine Learning techniques on these systems, which can become smarter and more efficient. Finally, we will try to build a model based on these new technologies for achieving a high level of network security.

Table des matières

Table des abréviations

1	DDoS	Distributed Denial of Service
2	DoS	Denial of Service
3	FTP	File Transfer Protocol
4	HIDS	Host-based Intrusion Detection System
5	HTTP	Hypertext Transfer Protocol
6	ICMP	Internet Control Message Protocol
7	IDS	Intrusion Detection System
8	IMAP	Internet Message Access Protocol
9	IPS	Intrusion Prevention System
10	IP	Internet Protocol
11	JSON	JavaScript Object Notation
12	NIDS	Network-based Intrusion Detection System
13	POP	Post Office Protocol
14	SMTP	Simple Mail Transfer Protocol
15	SSH	Secure Shell
16	TCP	Transmission Control Protocol
17	UDP	User Datagram Protocol

A

Table des figures

Introduction générale

L'évolution rapide de la technologie a conduit à de nouvelles formes d'innovation dans le domaine des IDS. Les cyberattaques sont devenues de plus en plus sophistiquées, exploitant les vulnérabilités des systèmes et les techniques d'attaque avancées pour contourner les mesures de sécurité traditionnelles.

En réponse à cette menace en constante évolution, les IDS ont dû s'adapter et évoluer pour rester efficaces. De nouvelles techniques et technologies, telles que l'intelligence artificielle, le machine learning et l'analyse comportementale, ont été intégrées aux IDS pour améliorer leur capacité à détecter les attaques complexes et à réduire les faux positifs. Ces avancées permettent aux IDS de mieux comprendre les modèles de comportement des utilisateurs, d'identifier les anomalies et de détecter les attaques ciblées.

En combinant ces innovations avec une surveillance continue, une analyse en temps réel et une réponse rapide, les IDS jouent un rôle essentiel dans la protection des systèmes informatiques contre les cyberattaques et la préservation de la confidentialité, de l'intégrité et de la disponibilité des données.

Notre aidée est de faire l'exploitation du Raspberry Pi pour la surveillance réseau avec Suricata qui un système de détection d'intrusion (IDS) largement utilisé. Le premier chapitre aborde les bases des systèmes de détection d'intrusion, en définissant leur concept et en expliquant la différence entre les IDS et les systèmes de prévention d'intrusion (IPS). Une classification des IDS est également présentée. Sans oubliant de faire une introduction générale de Suricata, en mettant l'accent sur ses fonctionnalités et ses règles.

Le deuxième chapitre se concentre sur l'exploitation du Raspberry Pi pour la surveillance réseau avec Suricata. En prmier lieu on trouve une présentation du Raspberry Pi, en mettant en évidence ses caractéristiques et sa configuration. Et en deuxième lieu, la mise en place de Suricata en effectuant plusieurs tests sur lui afin de juger sa performance.

Le troisième chapitre aborde le rôle de l'intelligence artificielle (IA) dans les IDS. En commençant par une explication de l'intelligence artificielle et de l'apprentissage automatique (Machine Learning), suivie d'une discussion sur les IDS intelligents qui utilisent ces techniques. Après, on passe à l'implémentation d'un IDS basé sur l'apprentissage automatique qui sera par la suite combiné avec Suricata.

Chapitre 1

Les systèmes de détection d'intrusion

1.1 Définition

Les système de détection d'intrusion (intrusion detection system IDS en anglais) est un outil utilisé dans la sécurité informatique pour surveiller les réseaux et les systèmes informatiques, et détecter les activités suspectes ou malveillantes.

Il vise à détecter une menace avant qu'elle ne s'infiltre dans un réseau, donc en cas d'une intrusion ou une violation il le signale à l'administrateur ou au personnel de sécurité en générant des alertes et en envoyant des notifications, ce qui aide à enquêter sur l'incident signalé et à prendre des mesures appropriées. En plus de détecter les violations de politique, il peut se prémunir contre les menaces telles que les fuites d'informations, les accès non autorisés, les erreurs de configuration, les chevaux de Troie et les virus.

D'où, l'objectif principal d'un IDS est d'identifier les tentatives d'intrusion, les attaques ou les comportements anormaux qui pourraient compromettre la sécurité du système, Il est par conséquent une solution de surveillance passive qui peut alerter de la détection d'une menace, mais qui ne peut pas prendre de mesures directes contre celle-ci. [1]

1.2 La différence entre IDS et IPS

D'abord, un système de prévention d'intrusion (intrusion prevention system IPS en anglais) est une solution logicielle qui surveille les activités d'un système ou d'un réseau à la recherche d'incidents malveillants, enregistre des informations sur ces activités, les signale à l'administrateur ou au personnel de sécurité et tente de les arrêter ou de les bloquer. [2]

La principale différence entre un IDS et un IPS est que le premier est un système de surveillance, alors que le deuxième est un système de contrôle.

Il est vrai que l'IPS a une fonction similaire à l'IDS dans la détection des activités suspectes ou malveillantes, mais il va plus loin en prenant des mesures actives pour prévenir et bloquer les activités malveillantes détectées. En plus de générer des alertes, l'IPS peut également prendre des mesures telles que le blocage d'adresses IP, la modification des règles de pare-feu, etc.

Au contraire de l'IDS qui fonctionne en mode de détection passive, l'IPS fonctionne en mode de prévention actif, il inspecte le trafic en temps réel et prend des mesures pour bloquer ou neutraliser les activités suspectes dès qu'elles sont détectées.

1.3 Classification des IDS

On peut classer les IDS selon l'endroit où se produit la détection de la menace comme les HIDS et les NIDS, ou bien selon la méthode de détection utilisée comme les IDS basés sur des

signatures et ceux qui sont basés sur des anomalies.[3]

1.3.1 IDS basé sur l'hôte

un IDS basé sur l'hôte est déployé sur un point d'extrémité particulier et conçu pour le protéger contre les menaces internes et externes. Ce type d'IDS peut avoir la capacité de surveiller le trafic réseau en provenance et à destination de la machine, d'observer les processus en cours et d'inspecter les journaux du système. Ils analysent les journaux d'événements, les fichiers système, les connexions réseau, etc., pour détecter les comportements malveillants ou anormaux. La visibilité d'un IDS basé sur l'hôte est limitée à la machine hôte, ce qui réduit le contexte disponible pour la prise de décision, mais il dispose d'une visibilité approfondie sur les parties internes de l'ordinateur hôte. Ce type d'IDS ne détecte les attaques qu'après leur occurrence.

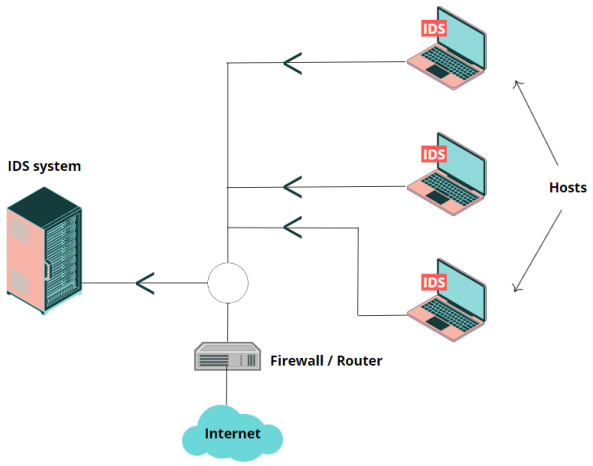

FIGURE 1.1 – HIDS

1.3.2 IDS basé sur le réseau

Cet IDS est positionné sur le réseau et surveillent le trafic réseau en temps réel. Il analyse les paquets de données, les flux de communication et les protocoles réseau pour détecter les activités suspectes. Les NIDS sont généralement déployés sur des points stratégiques du réseau, tels que des commutateurs ou des concentrateurs.

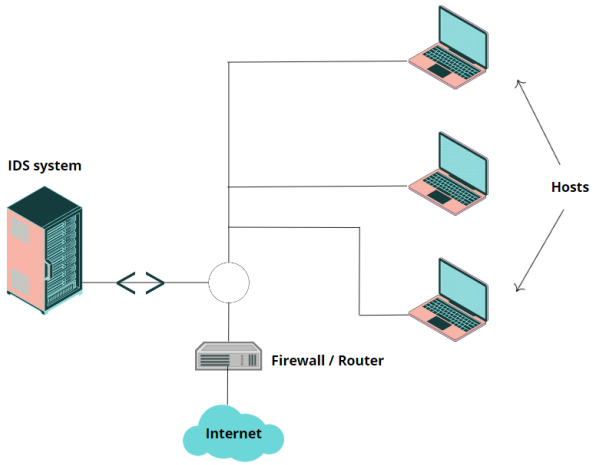

FIGURE 1.2 – NIDS

1.3.3 IDS basé sur les anomalies

Les solutions IDS basées sur les anomalies construisent un modèle du comportement normal du système protégé pour détecter les activités qui s'écartent de la norme. Tous les comportements futurs sont comparés à ce modèle et toute anomalie est qualifiée de menace potentielle et génère des alertes. Si cette approche permet de détecter les nouvelles menaces ou les menaces de type "zero-day", la difficulté de construire un modèle précis de comportement normal signifie que ces systèmes doivent équilibrer les faux positifs (alertes incorrectes) et les faux négatifs (détections manquées).

1.3.4 IDS basé sur les signatures

Les solutions IDS basées sur des signatures utilisent les empreintes digitales des menaces connues pour les identifier. Autrement dit, il utilise une base de données de signatures ou de règles préétablies pour détecter les intrusions. Une fois qu'un logiciel malveillant ou un autre contenu malveillant a été identifié, une signature est générée et ajoutée à la liste utilisée par la solution IDS pour tester le contenu entrant. Cela permet à un système IDS d'atteindre un taux élevé de détection des menaces sans faux positifs, car toutes les alertes sont générées sur la base de la détection d'un contenu malveillant connu. Cependant, un IDS basé sur des signatures est limité à la détection des menaces connues et est aveugle aux vulnérabilités de type "zero-day".

1.4 L'IDS Suricata

1.4.1 Généralités

Suricata, Zeek et Snort sont tous des systèmes de détection d'intrusion (IDS) largement utilisés pour surveiller et protéger les réseaux contre les activités malveillantes.

Chacun de ces outils a ses propres forces et faiblesses. Suricata se distingue par ses capacités de prévention des intrusions, tandis que Zeek se concentre sur l'analyse approfondie du trafic réseau. Snort est connu pour sa vaste base de règles et sa large adoption. [4] [5]

Le tableau suivant montre une comparaison entre les caractéristiques de chacun d'eux :

Caractéristiques	Snort	Suricata	Zeek
Type d'outil	IDS	IDS	Network Security Monitor
Détection basée sur	Signatures, règles	Signatures, règles	Analyse du trafic réseau
Support des protocoles	Large gamme de protocoles	Large gamme de protocoles	Protocoles réseau courants
Mode de déploiement	En ligne ou hors ligne	En ligne ou hors ligne	Hors ligne
Performances	Performances élevées	Performances élevées	Performances élevées
Prise en charge de la DPI	Oui	Oui	Non
Capacités de prévention	Oui (Snort-inline)	oui (Suricata-Inline)	Non
Analyse du contenu	Oui	Oui	Oui
Flexibilité et extensibilité	Moins flexible	Plus flexible	Très flexible
Support de la communauté	Grande communauté	Grande communauté	Grande communauté
Documentation	Bien documenté	Bien documenté	Bien documenté
Mises à jour et maintien	Mises à jour régulières	Mises à jour régulières	Mises à jour régulières
Licence	GPL	GPL	BSD

TABLE 1.1 – Tableau comparatif de Suricata, Snort et Zeek

Nous avons décidé de travailler avec Suricata qui se distingue par ses capacités de prévention des intrusions en temps réel, ainsi que sa haute performance, sa flexibilité et son extensibilité et son utilisation optimale des ressources.

1.4.2 Suricata

Suricata est un IDS open source, conçu pour être une alternative évolutive et performante à Snort. Il utilise également des règles et des signatures pour détecter les activités malveillantes, mais offre une syntaxe de règles plus expressive et une meilleure gestion des ressources. Suricata propose des fonctionnalités avancées telles que l'analyse du contenu, la prise en charge de protocoles étendus, la détection d'anomalies et la prévention des intrusions via le mode Suricata-Inline. Comme Snort, Suricata bénéficie d'une large communauté d'utilisateurs et de contributeurs actifs, assurant un support régulier et des mises à jour fréquentes. Suricata prend en charge plusieurs moteurs de détection grâce au multithreading, ce qui lui permet de gérer davantage de trafic réseau que Snort, qui ne prend en charge qu'un seul moteur de détection. En termes de détection basée sur les signatures, Suricata utilise le même format que celui employé par Snort pour la déclaration des règles et dispose également d'algorithmes de détection similaires. Comme Snort, Suricata prend également en charge les modes IDS et IPS, tandis

qu'au niveau de la sortie, les alertes peuvent être stockées soit dans un simple fichier texte, soit au format JSON.[4]

1.4.3 Les règles de Suricata

Les signatures/les règles jouent un rôle majeur dans Suricata, et surtout pour déclencher les alertes. On peut trouver un ensemble de règles déjà existantes dans Suricata-update, comme on peut modifier ou créer des nouvelles règles.

Une règle dans Suricata peut prednre plusieurs formes mais elle est toujours constituée de trois éléments fondamentaux : The action, the header et the options.

La figure suivante représente un exemple d'une règle de Suricata, telle que la partie rouge correspond à l'action, la verte correspond à l'entete et la bleue aux options. [6]

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +..)";
flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]
{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124;
rev:2;)
```

FIGURE 1.3 – Une règle de Suricata

Examinons maintenant chaque partie e la règle plus en détails :

— **The action** : détermine le comportement à adopter en cas de détection d'intrusion. Il existe quatre types d'action : **Pass**, **Drop**, **Reject** et **Alert**. Les actions Drop et Reject correspondent plus au fonctionnaité IPS de Suricata.

— **The header** : définit le protocole (TCP, UDP, ICMP,...), les adresses IP (ou des plages IP) définissant la source et la destination du trafic, la direction du trafic : entrant ->, sortant <- ou bidirectionnel <>, ainsi que les ports source et destination.

— **The options** : définit les spécificités de la régle.
Elles sont placées entre parenthèses et séparées par des points-virgules. Certaines options ont des paramètres (comme msg), qui sont spécifiés par le mot-clé de l'option, suivi de deux points, puis des paramètres. D'autres n'ont pas de paramètres et sont simplement mot-clé.
Prmi les options les plus utilisées, on trouve : le **sid** (pour Signature Identifier) permet à l'IDS de comparer son analyse de paquets avec une base de données. Si les signature correspondent, il effectuera alors l'action. Et **rev** qui est un mot-clé tout le temps utilisé avec le mot-clé sid pour signifier sa révision, il représente la version de la signature.
Les options de la règle ont un ordre spécifique et le fait de changer leur ordre modifierait la signification de la règle.

Chapitre 2

Exploitation du Raspberry Pi pour la surveillance réseau avec Suricata

2.1 Le Raspberry Pi

Notre idée est de mettre en œuvre un IDS au sein d'un Raspberry Pi pour surveiller un réseau. En fait, nous avons choisi d'utiliser le Raspberry Pi car il offre une solution économique par rapport à d'autres options matérielles grâce à son coût abordable d'une part, ainsi qu'il permet de simuler le comportement d'un serveur exécutant un IDS pour surveiller un petit réseau domestique ou un environnement à petite échelle d'autre part. Cependant, pour des réseaux plus importants ou des environnements professionnels, il est généralement recommandé de mettre en place l'IDS sur un serveur dédié disposant de ressources plus puissantes pour gérer efficacement la charge de travail de surveillance et d'analyse du trafic réseau. Mais lors de notre travail, nous allons essayé de tester le bon fonctionnement de cette solution qui pourra par la suite s'étendre à des réseaux plus grands avec des serveurs plus puissant. [7]

2.1.1 Présentation du Raspberry Pi

Raspberry Pi est le nom d'une série d'ordinateurs à carte unique lancée en 2012 par la Raspberry Pi Foundation, une organisation caritative britannique qui encou- rage l'apprentissage, l'expérimentation et l'innovation pour les élèves. Cet ordinateur monocarte de petite taille et d'un faible coût, offre une plateforme abordable et polyvalente pour réaliser une variété d'applications.

Le cœur d'une carte Raspberry est constitué d'un circuit intégré appelé SoC (System on a Chip : tout le système sur une puce), Il regroupe les composants essentiels de tout ordinateur, c'est-à-dire un CPU, une mémoire RAM, des ports d'entrée/sortie et de connectivité réseau.

Comme le Raspberry Pi n'est pas livré avec un système d'exploitation préinstallé. Cela signifie qu'on peut choisir parmi une large sélection de systèmes d'exploitation comme Arch Linux ARM, Kali Linux, Free BSD, Pidora,... N'importe lequel d'entre eux peut être flashé sur la carte SD de Raspberry Pi.

Il existe plusieurs types de Raspberry qui se diffèrent en termes de performances matérielles, notamment au niveau de la mémoire vive où on trouve que les modèles les plus récents sont généralement dotés d'une quantité de RAM supérieure par rapport aux anciennes versions ainsi qu'au niveau de processeur. De plus, certaines variantes plus récentes intègrent également une carte réseau directement sur la carte, offrant une connectivité Ethernet sans avoir besoin d'un adaptateur supplémentaire. [8]

2.1.2 Configuration et prise en main de Raspberry Pi

Au cours de ce travail, nous allons utiliser le Raspberry Pi 4 Model B 8 Go RAM avec une carte SD de 32 Go.

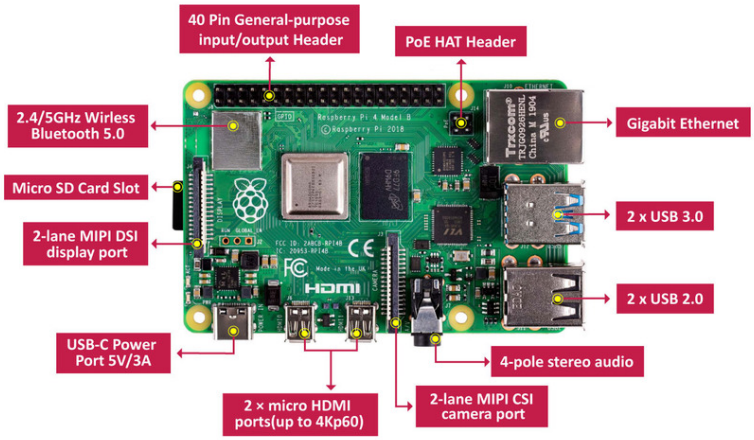

FIGURE 2.1 – Composants de Raspberry Pi 4 model B

Concernant le OS, nous avons décidé d'installer le Raspbian qui est le système d'exploitation standard officiel de Raspberry Pi. On commence par télécharger l'image de Raspberry Pi OS de site officiel, puis le logiciel Raspberry Pi Imager qui va nous permettre de faire modification de quelques paramètres nécessaires (comme la modification du nom d'ulisateur et du mot de passe par défaut, le choix du réseau auquel on veut se connecter et l'activation de l'option SSH) avant l'écriture sur la carte SD formatée.

Après l'insertion de la carte SD dans notre Raspberry Pi on le démarre et commence à le manipuler, que ce soit à travers un écran et un clavier ou bien via SSH comme dans notre cas.

```
PS C:\Users\HP> ssh riusers@raspberrypi.local
riusers@raspberrypi.local's password:
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST 2023 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jun  4 16:43:19 2023 from 10.1.5.214
riusers@raspberrypi:~ $
```

FIGURE 2.2 – Accès au Raspberry Pi via SSH

2.2 Déploiement de Suricata dans le Raspberry Pi

Le Raspberry Pi 4 dispose des spécifications matérielles adéquates pour l'installation et l'utilisation de Suricata sans problèmes. On commence par le téléchargement de la dernière version Suricata 6.0.1 à partir du site officiel puis on suit les étapes de l'installation. [9] [10]

Ensuite, on importe les règles standards préconfigurées sur le site https ://rules.emerging-threats.net/open/suricata/rules/. Elles permettent notamment d'avoir des règles pré-établies

pour les requêtes gérées par les protocoles ICMP, FTP, IMAP, SMTP, POP, NetBios, Telnet, HTTP...

Une fois Suricata installé, nous passerons à sa configuration à travers le fichier de configuration par défaut de Suricata "**/etc/suricata/suricata.yaml**", dans lequel nous spécifierons les paramètres essentiels tels que : l'interface réseau à surveiller que nous obtenons par la commande ifconfig et en l'affectant à la variable **$HOME_NET**, et les règles de détection à utiliser en spécifiant les noms des fichiers contenant ces règles dans la fin de fichier de configuration, par défaut sont déjà présentes les règles préconfigurées sous la section rule-files, elles sont placées par défaut dans le default-rule-path qui est **/etc/suricata/rules**. On peut rajouter une ligne - mesRegles.rules pour ajouter le fichier contenant nos règles personnalisées.

```
# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://suricata.readthedocs.io/en/latest/configuration/suricata-yaml.html

##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[10.1.33.230/16]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"
```

```
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /etc/suricata/rules

rule-files:
  - suricata.rules
  - scapy.rules
  - dos.rules
##
## Auxiliary configuration files.
##

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config
```

FIGURE 2.3 – Configuration de Suriata - suricata.yaml

Enfin, on lance Suricata en tant que service sur le Raspberry Pi :

```
riusers@raspberrypi:~ $ sudo systemctl status suricata
● suricata.service - Suricata IDS/IDP daemon
     Loaded: loaded (/lib/systemd/system/suricata.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2023-05-31 15:19:08 BST; 20h ago
       Docs: man:suricata(8)
             man:suricatasc(8)
             https://suricata-ids.org/docs/
    Process: 549 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid >
   Main PID: 603 (Suricata-Main)
      Tasks: 10 (limit: 4915)
        CPU: 3min 37.395s
     CGroup: /system.slice/suricata.service
             └─603 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid

May 31 15:19:07 raspberrypi systemd[1]: Starting Suricata IDS/IDP daemon...
May 31 15:19:07 raspberrypi suricata[549]: 31/5/2023 -- 15:19:07 - <Notice> - This is Suricata version 6.0.1 RELEASE ru>
May 31 15:19:08 raspberrypi systemd[1]: Started Suricata IDS/IDP daemon.
lines 1-16/16 (END)
```

FIGURE 2.4 – Lancement de Suricata - Service

2.3 Test de Suricata

Pour vérifier le bon fonctionnement des règles de Suricata, nous allons effectuer plusieurs tests pour des différents protocoles (ICMP, TCP). Pour ce faire, on ajoute nos règles personnalisées dans un fichier dans le répertoire **/etc/suricata/rules**, puis on ajoute les noms de ces fichiers dans le fichier de configuration **suricata.yaml**, ensuite on enregistre les changements effectués en mettant à jours suricata par la commande **sudo suricata-update**, et finalement on lance suricata et on ouvre le fichier des logs **/var/log/suricata/fast.log** où vont apparaitre les résultats des détections.

2.3.1 ICMP Echo (Ping)

Pour tester la détection du protocole ICMP, nous avons essayé d'envoyer des requtes ICMP echo (Ping) vers notre Raspberry Pi à partir d'une autre machine pour voir s'il seront détecter. Nous avons commencé par configurer notre propre règle sur Suricata : **alert icmp any any -> $HOME_NET any (msg : "ICMP Ping" ; sid :1 ;rev :1 ;)** Dans un fichier dans le répertoire /suricata/rules et en l'ajoutant dans le fichier de cinfiguration.

cette règle permet de déclencher une alerte en affichant le message "ICMP Ping" à chaque fois qu'on reçoit un paquet ICMP ping sur l'adresse IP de notre Raspberry spécifiée dans la variable $HOME_NET et de provenance de n'importe quelle adresse et de n'importe quel port.

Depuis notre machine, on envoie un Ping vers l'adresse IP de notre Raspberry Pi :

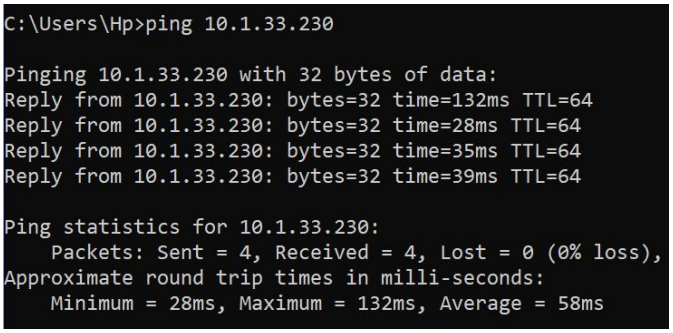

FIGURE 2.5 – Ping le Raspberry Pi

Et voilà les résultats dans le fichier des logs :

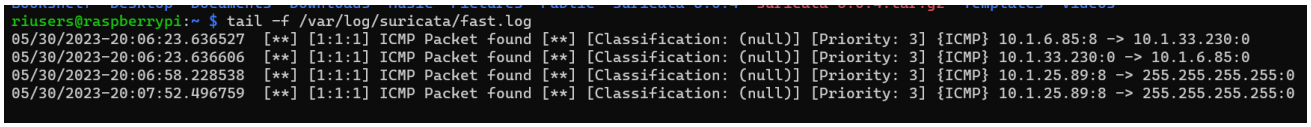

FIGURE 2.6 – Les logs de Suricata après le ping

2.3.2 Simulation d'une attaque DoS

Nous avons essayé de simuler une attaque par déni de service (DoS) avec l'outil HPING3 DDoS.

Donc comme déjà expliqué précédemment on ajoute la règle suivante :

alert tcp any any -> any any (flags : S ; msg : "Possible TCP DoS HPING3 DDoS attack" ; flow : stateless ; detection_filter : track by_dst, count 100, seconds 5 ; sid :10001 ; rev :1 ;)

qui consiste à déclencher une alerte lorsqu'un paquet TCP est détecté, indépendamment de l'adresse IP source et le port source, et qui a le flag SYN (S) activé (qui est généralement utilisé pour initialiser une connexion TCP), en étant dans l'état "stateless" c-à-d sans tenir compte de l'état de la connexion TCP. Par ailleurs, on applique un filtre consistant à filtrer les paquets TCP ayant la même adresse IP de destination, et en les comptant jusqu'à 100 dans un intervalle du temps de 5 secondes, c'est à ce moment là que l'alerte sera déclechée.

On envoie les paquets TCP depuis une machine virtuelle Kali sur notre PC vers le Raspberry Pi par la commande **hping3 -S -p 443 10.1.33.230 –flood** qui à travers l'outil de test de réseau "hping3" envoie des paquets TCP ayant le flag SYN levé, vers le port 443 de notre Raspberry Pi. L'option "flood" rend l'envoi de ces paquets très rapide sans attendre de réponse, ce qui peut entraîner une surcharge du réseau cible.

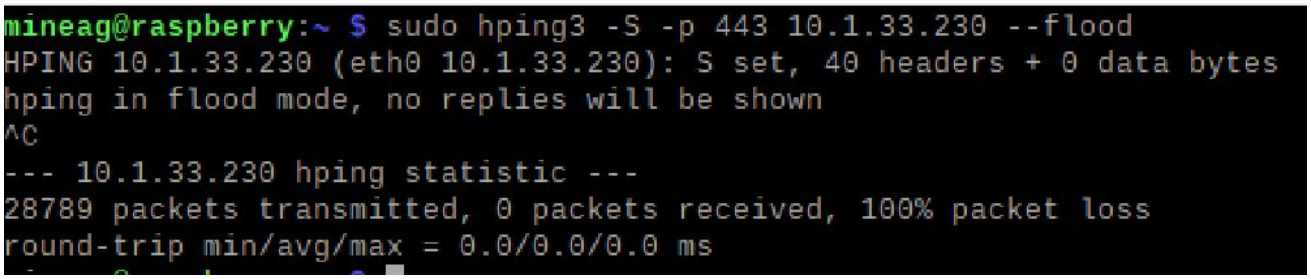

FIGURE 2.7 – L'attaque DoS

L'alerte est bien déclenchée sur le fichier des logs :

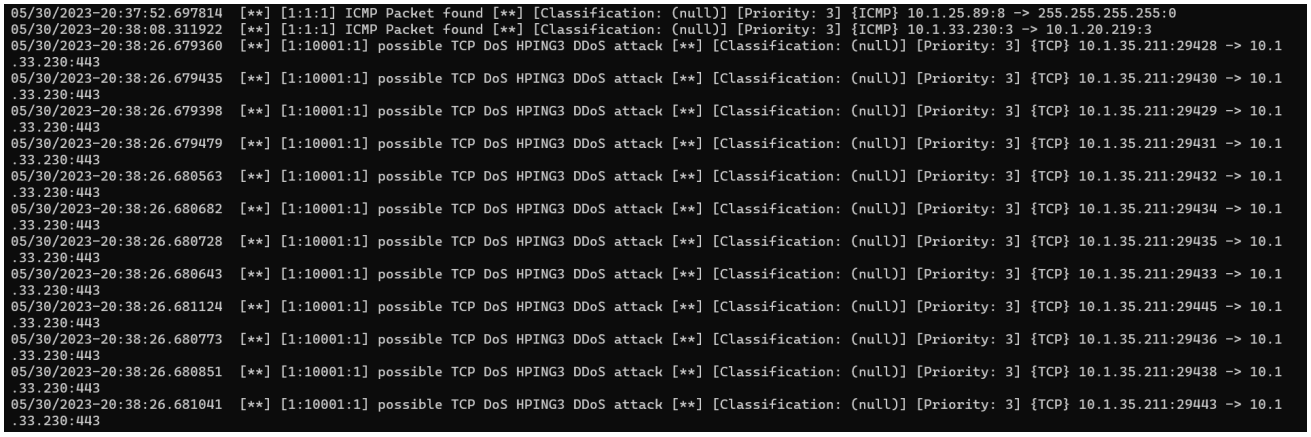

FIGURE 2.8 – Les logs de Suricata après l'attaque DoS

2.3.3 Visite d'un site web malveillant

Ici, nous essyaons de visiter un site web depuis notre Raspberry Pi, qui cherche à récupérer des informations propres au système. Ce site en ligne est dédié aux tests et à l'évaluation des systèmes de détection d'intrusion basé sur le réseau : **curl http ://testmynids.org/uid/index.html** [13]

La visite de ce site permet d'évaluer les performances des IDS en termes de détection d'attaques réelles et de fausses alertes.

FIGURE 2.9 – Accés au site web malveillant

Suricata a réussi à déclencher l'alerte, dans le fichier des logs on trouve :

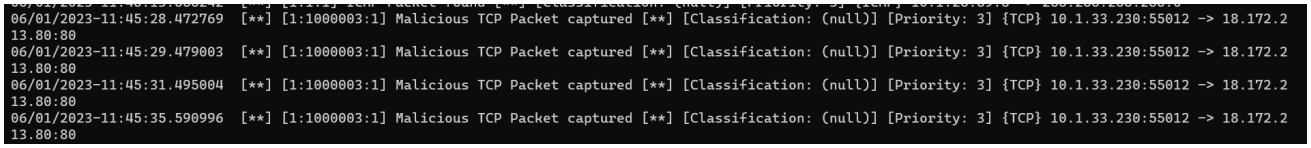

FIGURE 2.10 – Les logs de Suricata après accès au site malveillant

2.3.4 Trafic malveillant

Dans ce test, nous commence par télécharger un fichier contenant une caprture de trafic malveillant, puis

à travers la commande suivante **sudo suricata -r 2015-11-24-traffic-analysis-exercise.pcap -c /etc/suricata/suricata.yaml** on exécute suricata en utilisant le fichier de capture réseau (PCAP) à analyser, qui est nommé "2015-11-24-traffic-analysis-exercise.pcap", ainsi que le chemin du fichier de configuration "suricata.yaml". [11]

Suricata analysera par la suite le contenu de ce fichier pour détecter les éventuelles activités suspectes ou malveillantes. Même si il n'a pas réusé de détecter tous les attaques.

```
riusers@raspberrypi:~/paquets $ wget https://www.malware-traffic-analysis.net/2015/11/24/2015-11-24-traffic-analysis-exercise.pcap.zip
--2023-06-01 11:50:50--  https://www.malware-traffic-analysis.net/2015/11/24/2015-11-24-traffic-analysis-exercise.pcap.zip
Resolving www.malware-traffic-analysis.net (www.malware-traffic-analysis.net)... 199.201.110.204
Connecting to www.malware-traffic-analysis.net (www.malware-traffic-analysis.net)|199.201.110.204|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11726615 (11M) [application/zip]
Saving to: '2015-11-24-traffic-analysis-exercise.pcap.zip'

2015-11-24-traffic-analysis-exercise.p 100%[=========================================================================>]  11.18M   269KB/s    in 79s

2023-06-01 11:52:09 (145 KB/s) - '2015-11-24-traffic-analysis-exercise.pcap.zip' saved [11726615/11726615]

riusers@raspberrypi:~/paquets $ ls
2015-11-24-traffic-analysis-exercise.pcap.zip
riusers@raspberrypi:~/paquets $ unzip 2015-11-24-traffic-analysis-exercise.pcap.zip
Archive:  2015-11-24-traffic-analysis-exercise.pcap.zip
[2015-11-24-traffic-analysis-exercise.pcap.zip] 2015-11-24-traffic-analysis-exercise.pcap password:
  inflating: 2015-11-24-traffic-analysis-exercise.pcap
riusers@raspberrypi:~/paquets $ ls
2015-11-24-traffic-analysis-exercise.pcap  2015-11-24-traffic-analysis-exercise.pcap.zip
riusers@raspberrypi:~/paquets $ sudo suricata -r 2015-11-24-traffic-analysis-exercise.pcap -c /etc/suricata/suricata.yaml
1/6/2023 -- 11:54:29 - <Notice> - This is Suricata version 6.0.1 RELEASE running in USER mode
1/6/2023 -- 11:54:29 - <Notice> - all 5 packet processing threads, 4 management threads initialized, engine started.
1/6/2023 -- 11:54:29 - <Notice> - Signal Received.  Stopping engine.
1/6/2023 -- 11:54:29 - <Notice> - Pcap-file module read 1 files, 24240 packets, 14093286 bytes
riusers@raspberrypi:~/paquets $ ls
2015-11-24-traffic-analysis-exercise.pcap  2015-11-24-traffic-analysis-exercise.pcap.zip  eve.json  fast.log  stats.log  suricata.log
riusers@raspberrypi:~/paquets $ cat fast.log
11/24/2015-16:14:20.436868  [**] [1:1000003:1] Malicious TCP Packet captured [**] [Classification: (null)] [Priority: 3] {TCP} 10.1.25.119:49163 -> 191.234.
5.80:80
11/24/2015-16:14:20.057825  [**] [1:1000003:1] Malicious TCP Packet captured [**] [Classification: (null)] [Priority: 3] {TCP} 10.1.25.119:49159 -> 191.234.
5.80:80
11/24/2015-16:14:21.090950  [**] [1:1000003:1] Malicious TCP Packet captured [**] [Classification: (null)] [Priority: 3] {TCP} 10.1.25.119:49166 -> 191.234.
5.80:80
11/24/2015-16:14:20.436079  [**] [1:1000003:1] Malicious TCP Packet captured [**] [Classification: (null)] [Priority: 3] {TCP} 10.1.25.119:49162 -> 191.234.
5.80:80
11/24/2015-16:14:20.779180  [**] [1:1000003:1] Malicious TCP Packet captured [**] [Classification: (null)] [Priority: 3] {TCP} 10.1.25.119:49164 -> 191.234.
5.80:80
11/24/2015-16:14:22.632563  [**] [1:1000003:1] Malicious TCP Packet captured [**] [Classification: (null)] [Priority: 3] {TCP} 10.1.25.119:49168 -> 74.125.2
26.165:80
```

FIGURE 2.11 – Analyse d'un trafic malveillant

Finalement, on remarque que les alertes sont déclenchées par Suricata sur le fichier des logs.

2.4 Limitations de Suricata

Comme mentionné précédemment, Suricata se base principalement sur des signatures pour détecter les attaques. Cela signifie que s'il y a une attaque qui n'est pas incluse dans les règles de Suricata, elle ne pourra pas la détecter. Les signatures utilisées par Suricata sont basées sur des modèles préexistants d'attaques connues. Si une attaque utilise des techniques ou des vulnérabilités inconnues, Suricata ne pourra pas la reconnaître dès que les règles ne soient mises à jour pour inclure ces nouvelles menaces. Il peut donc y avoir un délai entre l'émergence d'une nouvelle attaque et la disponibilité des règles correspondantes dans Suricata.

Bien que Suricata puisse générer un fichier de sortie appelé "Anomaly" dans lequel il stocke les alertes pour le trafic considéré comme anormal ou inconnu, il n'est pas généralement utilisé de manière efficace car il génère trop de faux positifs. Il ne peut donc pas être considéré comme un bon choix pour détecter de nouvelles attaques ou des attaques "zero day". [12]

Donc, Suricata tous seul ne peut pas bien surveiller notre réseau efficacement. Pour ce faire, il est généralement recommandé de combiner Suricata avec des solutions plus intelligentes apportées par l'intelligence artificielle et surtout par le machine learning.

Chapitre 3

Smart IDS

3.1 l'intelligence artificielle dans les IDS

3.1.1 L'intelligence artificielle et Machine Learning

L'intelligence artificielle (IA) est un « ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine ». Elle entre en jeu là où l'informatique classique atteint ses limites. En effet, tout problème pour lequel il n'existe pas d'algorithme connu ou réalisable pour le résoudre est considéré comme relevant de l'IA.

le machine learning (ML) est une sous-branche de l'IA, qui consiste à laisser l'ordinateur apprendre quel calcul effectuer, plutôt que de lui donner ce calcul (c'est-à-dire le programmer de façon explicite). C'est la définition du Machine Learning selon son inventeur Arthur Samuel, un mathématicien américain qui a développé un programme pouvant apprendre tout seul comment jouer aux Dames en 1959.

Le ML regroupe des algorithmes qui analysent un ensemble de données afin de déduire des règles et des nouvelles connaissances permettant ainsi d'analyser des situations nouvelles.

Il comprend trois différents types d'algorithmes pour analyser et interpréter des données

- **L'apprentissage supervisé** utilise des données étiquetées pour former des modèles. Il repose sur des exemples où les entrées sont associées à des sorties connues, permettant aux algorithmes de prédire des sorties pour de nouvelles entrées. Les exemples d'apprentissage sont utilisés pour guider le modèle dans sa phase d'apprentissage.
- **L'apprentissage non supervisé** explore les structures et les relations intrinsèques des données non étiquetées pour en extraire des informations significatives.
- **L'apprentissage par renforcement** est basé sur l'idée d'un agent qui interagit avec un environnement et apprend à prendre des actions appropriées pour maximiser une récompense donnée. L'agent explore différents états de l'environnement, effectue des actions et reçoit des récompenses ou des pénalités en fonction de ses choix.

3.1.2 Les IDS intelligents

L'intégration de l'intelligence artificielle dans les systèmes de détection d'intrusion et surtout l'utilisation des techniques de l'apprentissage automatique apporte plusieurs avantages significatifs, On commence donc à parler d'un système de Détection d'Intrusion Intelligent "Smart IDS" qui se distingue d'un IDS traditionnel par les points suivants :

La détection s'est améliorée et les IDS peuvent analyser plus grandes quantités de données.

Les IDS classiques (traditionnels) reposent souvent sur des règles préétablies pour détecter les intrusions connues, alors que l'IA leur permet d'apprendre et de s'adapter en temps réel à de nouvelles menaces et à des techniques d'attaque émergentes. ce qui garde leur efficacité face

à une évolution constante des attaques.

La détection des faux positifs posent des problématiques pour les IDS, car ils entraînent des alertes inutiles et une surcharge de travail pour les équipes de sécurité. L'IA peut aider donc à les réduire en identifiant les modèles de trafic normaux et en affinant les mécanismes de détection pour améliorer la précision des alertes.

3.2 Implémentation d'un smart IDS

3.2.1 Description

Il est vrai qu'il peut être difficile de détecter les attaques zero-day par Suricata, puisque c'est un IDS basé principalement sur les signatures, et il se peut que ces derniers ne sont pas répertoriées dans sa base de données de signatures comme déjà expliqué précédemment. Pour faire face à cette problématique, nous envisageons de créer un modèle de machine learning qui complètera Suricata. Ce modèle pourra nous permettre de construire un IDS basé sur anomalies, qui va etre par la suite combiné avec suricata pour une détection plus avancée des anomalies. [14] [15]

3.2.2 analyse exploratoire des données

3.2.2.1 Source de données

Nous avons utilisé l'ensemble de données CICIDS2017 [16] proposé par l'institut canadien de cybersécurité. Il contient à la fois des attaques bénignes et des attaques récentes qui sont similaires aux données réelles capturées sous la forme de fichiers PCAP.

3.2.2.2 Exploration et préparation de données

Le jeu de données est constitué de 2214468 enregistrements et 79 attributs dont 77 de types float64 et 2 de type object.

Plusieurs traitements ont été effectués sur le jeu de données pour le rendre prêt à être utilisé dans les modèles de classification.

Avant de traiter les attributs représentant une forte corrélation entre eux, et ceux qui sont nulles ainsi que les enregistrements redondants, nous avons remarqué qu'il existe des colonnes ayant des valeurs constantes (une seul valeur unique) dont nous décidons de les supprimer car ils n'ajoutent aucune information supplémentaire.

Matrice de corrélation

La matrice de corrélation permet de comprendre et d'analyser les relations linéaires entre les variables (attributs) d'un ensemble de données. Elle offre un aperçu global des corrélations entre les différentes variables.

Deux variables qui sont fortement corrélées contiennent des informations similaires donc le fait de les utiliser tous les deux dans notre modèle n'ajoute pas de nouvelles informations et conduit à une redondance et surcharge du modèle.

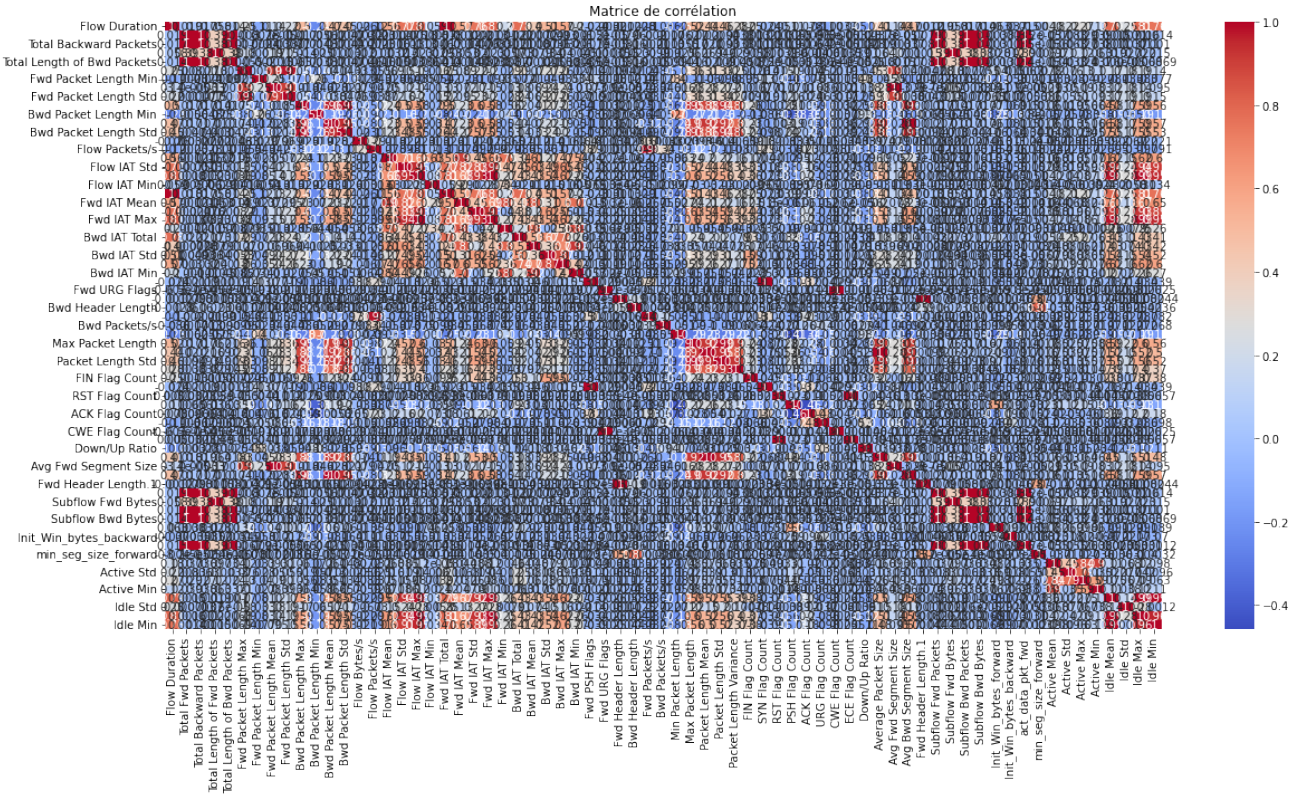

FIGURE 3.1 – Martice de corrélation

Puisque le jeu de données contient 79 variables, la matrice de corrélation est presque illisible, mais on peut remarquer beaucoup de petits carrés rouges représentant la forte corrélation (=1) entre quelques attributs.

Par ailleurs nous avons décidé d'éliminer les variables ayant une forte corrélation entre eux (qui dépassent 0.8)

En éliminant ces attributs, nous nous retrouverons finalement avec 33 colonnes significatives.

```
Out[27]: Index(['Destination Port', 'Flow Duration', 'Total Fwd Packets',
                'Total Length of Fwd Packets', 'Fwd Packet Length Max',
                'Fwd Packet Length Min', 'Bwd Packet Length Max',
                'Bwd Packet Length Min', 'Flow Bytes/s', 'Flow Packets/s',
                'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Min', 'Bwd IAT Total',
                'Bwd IAT Std', 'Fwd PSH Flags', 'Fwd URG Flags', 'Fwd Header Length',
                'Bwd Header Length', 'Bwd Packets/s', 'Min Packet Length',
                'FIN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count',
                'URG Flag Count', 'Down/Up Ratio', 'Init_Win_bytes_forward',
                'Init_Win_bytes_backward', 'Active Mean', 'Active Std', 'Idle Std',
                'Label'],
               dtype='object')
```

FIGURE 3.2 – les attributs restants après élimination des forts corrélés

Ainsi la matrice de corrélation devient :

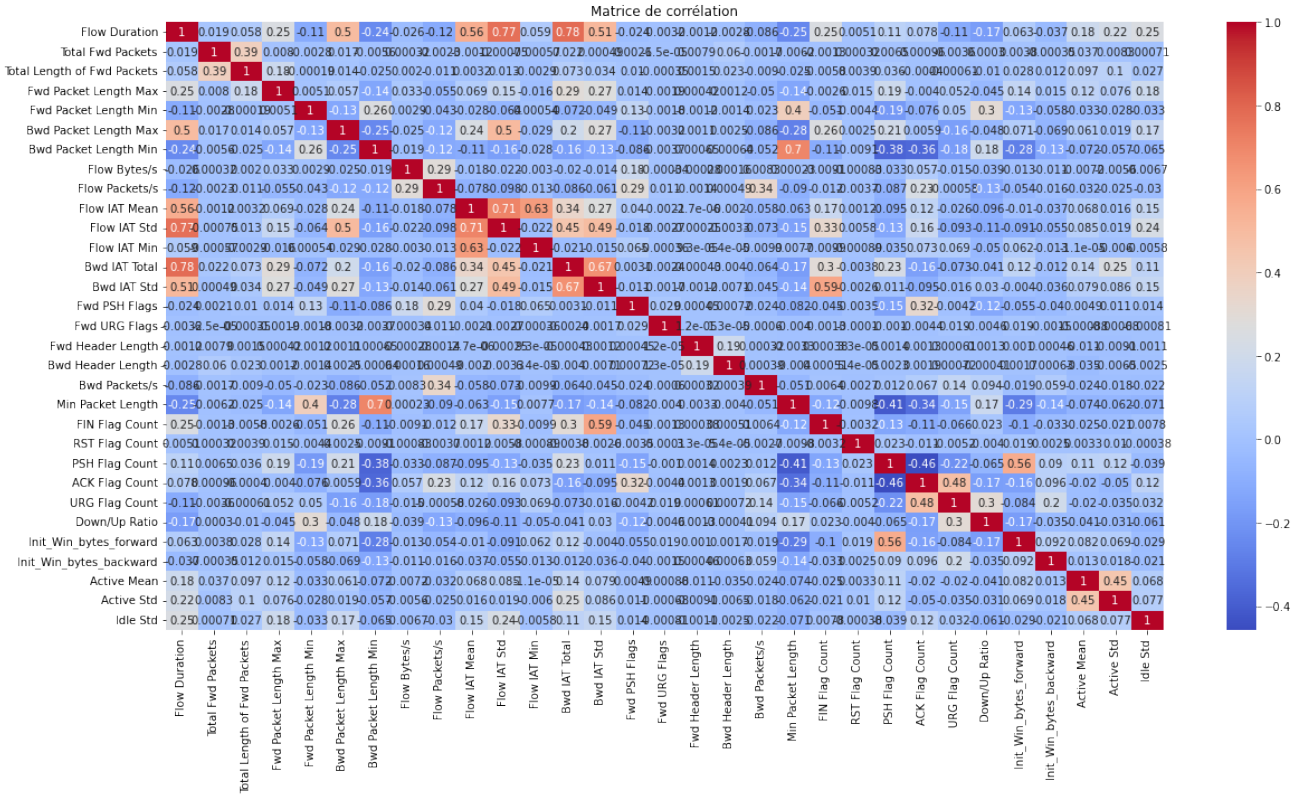

FIGURE 3.3 – Martice de corrélation après élimination des attributs fortement corrélés

De plus, Comme mentionné précédemment, nous souhaitons créer un modèle qui effectue l'apprentissage sur le comportement normal du trafic. Pour la cible "Label", nous allons regrouper différentes valeurs ['DDoS', 'PortScan', 'Bot', 'Infiltration','DoS slowloris', 'DoS Slowhttptest', 'DoS Hulk', 'DoS GoldenEye', 'Heartbleed'] dans une seule valeur appelée "Anormal", et pour l'autre valeur "BENIGN" sera appelée"Normal".

Les valeurs manquantes

2432 valeur manquante ont été identifier dans ce jeu de données et qui sont remplacée par la valeur moyenne de chaque colonne.

Les valeurs redondantes

271598 Les lignes redondantes ont été éliminées pour éviter toute répétition ou duplication des données.

Equilibrage de jeu de données

Nous avons remarqué qu'il y a un déséquilibre extrême entre les classes du jeu de données. Donc pour résoudre ce problème il exite deux solutions :

— **Oversampling** : consiste à augmenter le nombre d'instances de la classe minoritaire. Cependant, en raison des contraintes de capacité de notre machine, le traitement des données prendrait beaucoup de temps.

— **Undersampling** : consiste à réduire le nombre d'instances de la classe majoritaire. Cela permettrait d'équilibrer les classes, mais pourrait entraîner une perte d'informations.

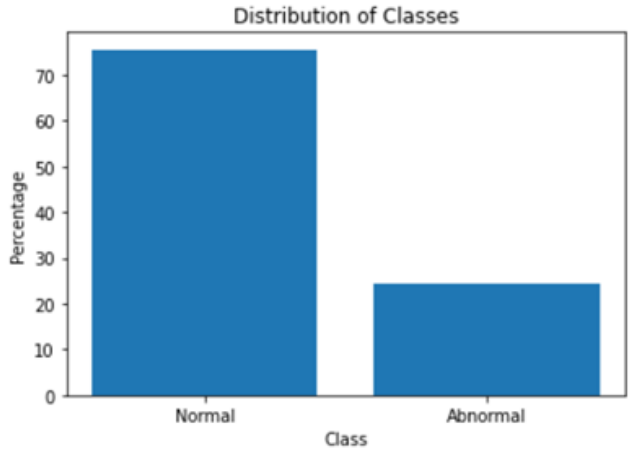

FIGURE 3.4 – le déséquilibre de données

Dans notre cas, nous avons opté vers la deuxième solution qui est le undersampling. car, le oversampling nécéssite l'ajout de nouvelles données ce qui rend très lent le temps de traitement de données ainsi qu'il nécéssite des contraintes matérielles qui dépassent celles de notre machine.

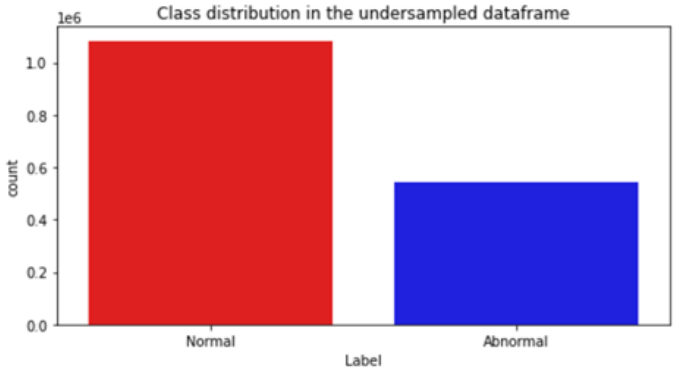

FIGURE 3.5 – Après l'équilibrage de données

3.2.3 Entrainement de modèle

Après avoir optimisé l'ensemble de données CICIDS2017, on passe maintenant à construire notre modèle. Nous avons choisi un algorithme de l'apprentisage supervisé qui s'appelle Arbre de décision (Decision Tree).

Un arbre de décision est une structure de graphe arborescente, où chaque nœud représente un test sur un attribut. Chaque branche représente le résultat du test et les nœuds feuilles représentent l'étiquette de classe obtenue après toutes les décisions prises via cette branche. Les chemins de la racine à la feuille représentent des règles de classification.

Dans un premier temps, on donne à notre modèle les données d'apprentissage labélisées pour qu'il puisse construire l'arbre, puis on lui donne les données de test sans label afin de pédire leurs classes qu'on va les comparer par la suite avec celles de notre jeu de données.

Les données d'apprentissage constituent 80% notre jeu de données, et Les données de test constituent 20%.

3.2.4 Evaluation de modèle

Pour l'evaluation de ce modèle, on génére d'abord la matrice de confusion pour les données de test :

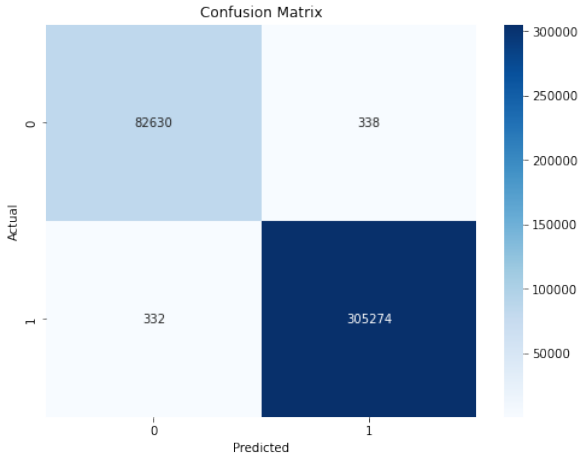

FIGURE 3.6 – Matrice de confusion pour les données de test

D'après cette matrice on obtient les informations suivantes :
— Les vrais positifs : 82633
— Les faux positifs : 335
— Les faux négatifs : 324
— Les vrais négatifs : 305282
En se basant sur ces informations, on calcule les métriques représentées dans le tableau ci-dessous permettant de juger la performance de modèle :

Accuracy	Precision	Recall	F1-score
99,8%	99,6%	99,6%	99,6%

TABLE 3.1

L'algorithme arbre de décision a pu de classer le trafic avec une précision de 99,8% ainsi qu'un faible taux de faux positifs.
Pour s'assurer qu'il n y a pas de surajustement (overfitting) dans notre modèle, on génère la matrice de confusion pour les données d'entrainement :

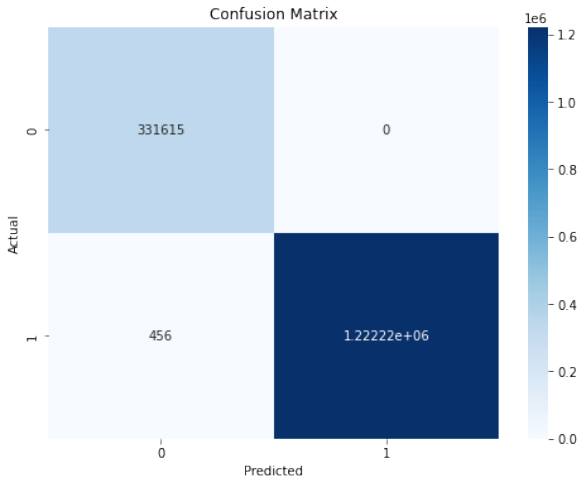

FIGURE 3.7 – Matrice de confusion pour les données d'entrainement

L'accuracy dans ce cas vaut 99,9% et qui est proche de l'accuracy des données de test clculée précedemment, ce qui affirme que le modèle ne présente par de surajustement. ‘

3.2.5 Combinaison de Suriata et le modèle réalisé

Vu les limites de Suricata déjà expliquées précédemment, nous explorons de nouvelles approches pour améliorer ses performances, ceci par la création d'un IDS hybride basé sur Suricata et ses signatures d'une part, et sur le modèle d'IDS par anomalies basé sur le Machine Learning d'autre part.

Dans ce cadre, Suricata effectue une surveillance continue de notre réseau, et lorsqu'il détecte un comportement inconnu anormal qque ses signature n'ont pas pu l'identifier, il l'enregistre dans les journaux d'anomalies contenant des informations sur le trafic réseau anormal. Parallèlement, nous enregistrons les fichiers .pcap du trafic réseau afin de les utiliser comme données d'entrée pour notre modèle d'apprentissage automatique.

En résumé, lors de la surveillance de réseau, si Suricata détecte une anomalie il se base sur ses signatures pour déclencher les alertes, mais dans le cas où suricata n'a pas déclenché un alerte il donne la main au deuxième IDS qu'on a construit pour qu'il décide un autre fois s'il s'agit d'un trafic normal ou anormal.

Deux scénarios se présentent alors : soit le trafic est considéré comme normal, soit il est jugé anormal. En cas de trafic anormal, le modèle génère une alerte pour l'administrateur afin qu'il effectue une vérification approfondie. L'administrateur peut ensuite rédiger une signature pour cette anomalie et l'ajouter à la liste des signatures de Suricata. Et de cette façon que notre IDS hybride fonctionne.

Cela offre une très bonne surveillance de réseau ainsi qu'un niveau de sécurité plus élevé.

Conclusion

Dans notre monde connecté, la sécurité des réseaux est une préoccupation majeure car ils sont exposés à une variété de menaces en ligne qui sont devenues de plus en plus sophistiquées et omniprésentes. Alors, l'utilisation des solutions comme des systèmes de détection d'intrusion est primordiale pour la surveillance d'un réseau et sa protection contre ces menace.

En fait, il existe plusieurs types d'IDS qui se diffèrent soit au niveau de l'endroit où se produit la détection de la menace comme les HIDS et les NIDS, soit au niveau de la méthode de détection utilisée comme les IDS basés sur des signatures et ceux basés sur des anomalies. Le choix entre ces IDS est relatif au contexte dans lequel ils vont etre utilisés ainsi qu'aux besoins des utilisateurs. Parmi les IDS qui sont largement utilisés on touve Zeek, Snort et Suricata. Mais tout au long de projet, nous avons décidé de travailler avec Suricata grace à son utilisation optimale des ressources matérielles.

L'idée proposée dans ce travail consiste à déployer un IDS aussi performant que Suricata au sein d'un Raspberry Pi pour surveiller un réseau. Elle reste une solution à petit cout mais à grande efficacité qui offre une réponse efficace aux attaques réseau. En effet, Suricata a réussi à déclencher les alertes après chaque test que nous avons effectué sur lui, ce qui prouve sa robustesse et sa bonne performance. Mais, on peut toujours améliorer les performances avec des solutions plus intelligentes grâce à l'intégration de l'intelligence artificielle et de l'apprentissage automatique, qui permettent aux IDS (dans notre cas : Suricata), de devenir capables de détecter les attaques complexes et sophistiquées, et de réduire aussi les faux positifs.

Et Donc la combinaison des IDS existants avec des technique de l'IA permet une surveillance continue, une détection précise des intrusions et une réponse rapide aux menaces. Dans un contexte où la sécurité des systèmes informatiques est devenue cruciale, les IDS jouent un rôle essentiel dans la protection des données et la préservation de l'intégrité des systèmes.

Références

[1] Intrusion Detection System (IDS), [https://www.geeksforgeeks.org/intrusion-detection-system-ids/]

[2] IDS vs IPS : un guide complet des solutions de sécurité réseau, [https://geekflare.com/fr/ids-vs-ips-network-security-solutions/]

[3] Classification of Intrusion Detection Systems, [https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/]

[4] Open Source IDS Tools : Comparing Suricata, Snort, Bro (Zeek), Linux, [https://cybersecurity.att.com/blogs/security-essentials/open-source-intrusion-detection-tools-a-quick-overview]

[5] Which open-source IDS ? Snort, Suricata or Zeek, [https://www.sciencedirect.com/science/article/abs/pii/S1389128622002420]

[6] Documentation de Suricata, [https://docs.suricata.io/en/suricata-6.0.12/]

[7] Documentation de Raspberry Pi, [https://www.raspberrypi.com/documentation/]

[8] Comparatif Raspberry Pi : quel modèle choisir ?, [https://framboisepi.fr/comparatif-raspberry-pi/]

[9] IDS on Raspberry Pi - A Performance Evaluation, Thommy SIMONSSON and Andreas ASPERNÄS, [https://www.eecis.udel.edu/ cshen/367/Assignments/IDS-RPi.pdf]

[10] Site officiel de Suricata, [https://suricata.io/]

[11] Installation de trafic réseau malveillant, [https://www.malware-traffic-analysis.net/about.html]

[12] Les différents types de sortie de Suricata, [https://docs.panther.com/data-onboarding/supported-logs/suricata]

[13] Le site testmynids, [https://github.com/3CORESec/testmynids.org]

[14] What Is Anomaly Detection in Machine Learning ?, Yulia Gavrilova, [https://serokell.io/blog/anomaly-detection-in-machine-learning]

[15] Suricata and Machine Learning Based Hybrid Network Intrusion Detection System, [https://www.researchgate.net/publication/357785493]

[16] Jeu de données CICIDS2017, [https://www.kaggle.com/datasets/cicdataset/cicids2017]