RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 55 02 80 70
http://www.ifri-uac.bj Courriel : secretariat@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Sécurité Informatique

Présenté par :
Yves-Christophane N. **MEDAGBE**

Conception d'un framework de détection d'intrusion en temps réel basé sur les arbres de décision et l'IDS SNORT

Sous la supervision :
Professeur Eugène C. **EZIN**
Enseignant-chercheur à l'UAC

Membres du jury :

M. **EZIN** C. Eugène	Professeur Titulaire	IFRI	Président
M. **HOUNDJI** Vinasétan Ratheil	Maître-assistant	IFRI	Examinateur
M. **AHOUANDJINOU** Arnaud	Maître de conférences	IFRI	Rapporteur

Année Académique : **2022-2023**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Table des figures vi

Liste des tableaux vii

Liste des acronymes viii

Introduction générale 1

1 Etat de l'art 4

2 Présentation de l'architecture de l'approche hybride proposée 20

3 Mise en place de la solution et évaluation de performances 27

Conclusion générale et perspectives 38

Bibliographie 40

Webographie 42

Table des matières 43

Dédicace

A ma merveilleuse famille pour son soutien inébranlable.

Remerciements

En préambule, permettez-nous d'exprimer notre profonde gratitude ainsi que notre reconnaissance envers toutes les personnes qui de près ou de loin, ont contribué à la réalisation de ce mémoire. Nos remerciements s'adressent particulièrement au/à :

- Professeur Eugène C. EZIN, Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI) pour nous avoir permis de suivre cette excellente formation ;
- Professeur Gaston EDAH, Directeur Adjoint de l'Institut de Formation et de Recherche en Informatique (IFRI) pour tous les efforts consentis en notre faveur particulièrement la gestion des emplois du temps et des rattrapages ;
- l'ensemble du corps des enseignants et du personnel administratif de l'IFRI pour la qualité des cours dispensés ;
- Professeur Eugène C. EZIN, cette fois-ci en qualité de maitre de mémoire, pour son implication dans tous les aspects du travail ;
- M. Modeste Parfait BESSAN notre maître de stage qui nous a accordé beaucoup d'attention durant et après notre stage ;
- tous les membres du jury pour avoir accepté sacrifier leur temps à l'appréciation de ce travail ;
- Centre des Activités Educatives du Bénin (CAEB) ainsi qu'à la fondation Vallet pour leur généreuse contribution financière à notre formation ;
- nos parents, frères, sœurs, et proches qui nous ont soutenu physiquement, moralement et financièrement ;
- tous nos aînés avec qui nous avons acquis des connaissances.

Résumé

Malgré l'utilisation d'antivirus et de pare-feux, les attaques informatiques restent un fléau mondial en constante augmentation. Contrer ces cyber-menaces devenant de plus en plus sophistiquées représente un défi de taille. Face à cette problématique, plutôt que de chercher à les éviter, des systèmes de détection d'intrusions (IDS) ont été développés pour les identifier lorsqu'elles surviennent, en exploitant différentes approches. Dans ce contexte, nous avons conçu un modèle de détection comportementale que nous avons couplé à l'outil libre Snort3 au sein d'un nouveau programme Python. Baptisé SHIDS (Smart Hybrid Intrusion Detection System), ce framework hybride combine la puissance de Snort et l'apprentissage automatique. Nous l'avons également doté d'un système d'alertes par SMS et email. Pour évaluer les performances de SHIDS, nous avons simulé une attaque informatique. Les résultats se sont avérés concluants, avec la réception effective des notifications d'alerte, démontrant ainsi l'efficacité de notre solution de détection d'intrusions intelligente et modulaire.

Mots clés : IDS, Apprentissage automatique, Cyberattaque

Abstract

Despite the use of antivirus and firewalls, cyberattacks remain a global scourge that is constantly increasing. Countering these increasingly sophisticated cyber threats represents a significant challenge. In response to this issue, rather than trying to avoid attacks, intrusion detection systems (IDS) have been developed to identify them when they occur, using different approaches. In this context, we have designed a behavioral detection model that we have coupled with the open-source tool Snort3 within a new Python program. Named SHIDS (Smart Hybrid Intrusion Detection System), this hybrid framework combines the power of Snort with machine learning. We have also equipped it with a system for sending alerts via SMS and email. To evaluate the performance of SHIDS, we simulated a cyberattack. The results were conclusive, with the effective reception of alert notifications, thus demonstrating the effectiveness of our intelligent and modular intrusion detection solution.

Key words: IDS, Machine Learning, Cyberattack

Table des figures

2.1 Fonctionnement global du Smart Hybrid Intrusion Detection System dans un réseau . 21

3.1 Réussite de l'installation de Snort 29
3.2 Vérification de l'effectivité des changements sur la carte réseau 30
3.3 Paramétrage des adresses de réseaux hostiles et à protéger 32
3.4 Intégration des règles téléchargées 33
3.5 Réussite du test de configuration 33
3.6 Lancement de l'API de de prédiction 34
3.7 Initialisation du Smart Hybrid Intrusion Detection System 34
3.8 Notification sms de l'attaque 35
3.9 Notification d'alerte par mail 36
3.10 Requêtes du module d'analyse comportementale vers l'API de prédiction 36

Liste des tableaux

1.1 Tableau comparatif de quelques solutions IDS 18

2.1 Tableau descriptif des caractérisques du dataset 22
2.2 Performances des modèles ML entrainés 24

3.1 Les étapes d'installation des dépendances. 28
3.2 Les étapes d'installation de SNORT3. 28

Liste des acronymes

API : Application Programming Interface

ARP : Address Resolution Protocol

CART : Classification And Regression Trees

DMZ : DeMilitarized Zone

DNS : Domain Name System

DVWA : Damn Vulnerable Web Application

HIDS : Host-based Intrusion Detection System

ICMP : Internet Control Message Protocol

IDS : Intrusion Detection System

IP : Internet Protocol

IPS : Intrusion Prevention System

NIDS : Network-based Intrusion Detection System

RFC : Requests For Comments

SGBD : System de Gestion de Base de Données

SIEM : Security Information Management System

SSH : Secure Shell

TCP : Transmission Control Protocol

UDP : User Datagram Protocol

Introduction Générale

Dans un monde de plus en plus connecté, la sécurité informatique est devenue une préoccupation essentielle pour les organisations et les individus. Les cyberattaques, visant à voler des informations, perturber des services ou causer des dommages, sont en constante augmentation et gagnent en sophistication. Traditionnellement, les antivirus et les pare-feux ont été utilisés pour protéger les systèmes contre ces menaces. Cependant, ces mesures de défense se révèlent souvent insuffisantes face à des attaques de plus en plus élaborées. Pour relever ce défi, les systèmes de détection d'intrusions (IDS) ont été développés pour identifier les attaques lorsqu'elles se produisent. Ces systèmes analysent le comportement du réseau et des systèmes pour détecter des activités suspectes ou malveillantes, permettant ainsi une réponse rapide et appropriée. Il est essentiel de continuer à perfectionner ces solutions de sécurisation des réseaux. Dans ce travail, nous proposons une approche innovante visant à renforcer l'efficacité des systèmes de détection d'intrusions.

Problématique

Face à la recrudescence et à la sophistication des attaques cyber-criminelles, les solutions tradtionnelles de sécurisation des systèmes d'information telles que les pare-feux, les antivirus et les VPNs s'avèrent limitées. D'où la nécessité d'ajouter une ligne de défense supplémentaire afin de surveiller en temps réel son réseau.

En étudiant les travaux antérieurs relatifs au sujet [1], [2], notamment les recherches menées conjointement par Eugène C. Ezin et Hervé Akakpo [3], nous nous sommes rendus compte que :

- l'approche de détection par recherche de motifs est certes efficace mais impuissante face aux attaques inconnues ;
- l'administrateur doit nécessairement être au service et en face de l'IDS avant d'être informé d'une quelconque intrusion ;
- l'utilisation des méthodes statistiques et probabilistes produit un taux non négligeable de faux positifs ;
- les codes sources des IDS basés sur le l'apprentissage en profondeur sont pour la plupart confidentiels

Ces faits évoquent deux principales questions qui font l'objet de la présente étude.

- Comment améliorer la performance et la fiabilité des systèmes de détection d'intrusion ?
- Comment s'assurer que les alertes générées par les IDS parviennent aux administrateurs même en dehors de l'entreprise ?

Contexte et Justification

Pour une entreprise connectée à l'internet, la question n'est plus de s'avoir si elle sera attaquée, mais plutôt le moment où l'intrusion aura lieu. En d'autres termes, la connexion quasi-permanente à l'Internet a pour corollaire leur exposition à des menaces toujours croissantes et plus élaborées. Conscientes de ce fait, les entreprises ne lésinent plus sur le financement nécessaire pour se doter d'outils capables d'améliorer le niveau de sécurité de leur système d'information. Les solutions conventionnelles de protection telles que les antivirus, pare-feux, IDS et de monitoring réseau suivent immédiatement l'installation de l'infrastructure d'une entreprise. Toutefois, ces moyens sont limités à la détection d'attaques connues. Ils sont moins efficaces face aux nouvelles attaques dites attaques zero-day.

Face au défi d'amélioration de la portée et de l'efficacité de détection d'attaques informatiques, certains chercheurs concentrent beaucoup plus leurs efforts sur l'analyse de profils grâce à des algorithmes d'apprentissage automatique. Cependant, bien que prometteurs, ces algorithmes ont aussi leurs limites et ne permettent pas à eux seuls une détection optimale des menaces émergentes. Il est donc nécessaire d'adopter un IDS avec une approche plus optimale de détection.

La mise en place du système proposé vise à renforcer la sécurité des systèmes d'information dans le sens où les cyber-attaques répertoriées ou non seront automatiquement détectées avec une marge réduite d'erreur. De plus, l'administrateur en sera informé quel que soit l'endroit où il se trouve.

Objectifs

L'objectif de cette étude est de concevoir un système capable d'identifier des attaques inconnues avec un faible taux de fausses alarmes à travers les bases de signatures et les algorithmes d'analyse de comportements.
De manière spécifique, il s'agit de :

- mettre en place un environnement représentant l'architecture réseau d'une entreprise ;
- mettre au point un algorithme de machine learning optimal ;
- extraire des paquets réseau les informations nécessaires pour la prédiction avec les algorithmes de machine learning ;
- intégrer un NIDS existant d'analyse par scénario ;
- créer un système de génération d'alertes avec notification par mail et sms ;
- tester la performance d'un système de détection d'intrusion hybride.

Organisation du mémoire

Le présent document est structuré en trois chapitres. Dans le premier qui concerne l'état de l'art, nous parlons des concepts liés à la sécurité informatique, aux systèmes de détection d'intrusion et au machine learning. Le second chapitre présente la solution choisie ainsi que les matériels et la méthodologie de création et d'implémentation de cette dernière. Enfin, le dernier chapitre est consacré à la mise en place de la solution proposée, à l'analyse et de l'évaluation des performances de l'IDS.

Chapitre 1

Etat de l'art

Introduction

Dans ce chapitre, nous abordons les connaissances existantes afin de mettre en évidence la pertinence du sujet abordé. Pour ce faire, nous commençons par donner une vue globale de la sécurité informatique et des attaques informatiques, et passons en revue les systèmes de détection d'intrusion.

1.1 La sécurité informatique

1.1.1 Définition

La sécurité informatique est l'ensemble des moyens humains, juridiques, techniques et organisationnels mis en oeuvre pour assurer la protection des données d'un système d'information contre des menaces accidentelles ou intentionnelles.

1.1.2 Objectifs de la sécurité informatique

Les systèmes d'informations constituent l'ensemble des données de l'organisation ainsi que les infrastructures matérielles et logicielles qui assurent le traitement, le stockage et la transmission de ces données. La sécurité informatique d'une manière générale vise à s'assurer que les ressources de l'entreprise sont utilisées par les personnes légitimes, aux heures prescrites et dans le cadre prévu. Ainsi les objectfis de la sécurité informatique peuvent être déclinés comme suit [4] :

- **l'intégrité** : l'information ne doit pas être altérée ou modifiée en dehors des personnes autorisées durant la communication ;
- **la confidentialité** : s'assurer que l'information soit inintelligible aux personnes sans le niveau d'accréditation requis ;
- **la disponibilité** : garantir un accès sans interruption à un service ou à des ressources ;
- **la non répudiation** : permet de garantir qu'une transaction ne puisse être niée ;

- **l'authentification** : consiste à s'assurer que seules les personnes autorisées aient accès aux ressources.

1.1.3 Mise en place d'une politique de sécurité informatique

La Politique de Sécurité des Systèmes d'Information (PSSI) définit l'intégralité de la stratégie de sécurité informatique de l'entreprise. Elle se traduit par la réalisation d'un document qui regroupe l'ensemble des règles de sécurité à adopter ainsi que le plan d'actions ayant pour objectif de maintenir le niveau de sécurité de l'information dans l'organisme. [5]

1.1.3.1 Différents aspects de la sécurité

L'élaboration d'une politique de sécurité doit être effectuée suivant les axes suivants : [6]

- la gestion de l'accès aux données logiques notamment les données d'entreprise, les applications ou encore les sytèmes d'exploitation (mécanismes d'authentification, contrôle d'intégrité) ;
- la gestion des accès physiques aux équipements en fonction de leur vitalité pour système d'information ;
- la sécurité des télécommunications (technologies réseau, serveurs d'entreprise, réseaux d'accès, etc.) ;
- la gestion des ressources humaines, maillon faible de tout système (sensibilisation aux notions basiques et essentielles de sécurité telles que le choix et la gestion des mots de passe).

1.1.4 Etapes d'élaboration d'une politique de sécurité

Une politique de sécurité doit garantir les droits d'accès aux données et aux ressources d'un système à travers la mise en place de mécanismes d'authentification et de contrôle. Les utilisateurs ne doivent posséder que les droits qui leur ont été octroyés. Cependant ces règles ne doivent pas constituer un frein au bon usage des ressources informatiques dans la génération de gains pour l'entreprise. L'élaboration d'une politique de sécurité passe donc par quatre étapes : [7]

- identification des besoins de sécurité, des risques informatiques pesant sur l'entreprise et l'étendue de leurs impacts ;
- élaboration des règles et procédures à mettre en oeuvre dans les divers services de l'organisation pour les risques identifiés ;
- surveillance et détection des vulnérabilités du sytème d'information et veille technologique sur les nouvelles vulnérabilités et outils disponibles afin de combler toute brèche éventuelle ;
- définition des actions à entreprendre et de l'équipe d'intervention en cas de détection d'une menace.

Afin de mieux assurer la sécurité des Systèmes d'informations, il est primordial de comprendre le fonctionnement des attaques informatiques.

1.2 Les attaques informatiques

Une attaque informatique est toute tentative d'accès non autorisé à un ordinateur, un système informatique ou un réseau informatique dans le but de causer des dommages. Les attaques informatiques visent à désactiver, perturber, détruire ou contrôler des systèmes informatiques ou à modifier, bloquer, supprimer, manipuler ou voler les données contenues dans ces systèmes. [8]

1.2.1 Méthodologie générale d'une attaque

La plupart des attaques informatiques se mènent suivant la même démarche. En théorie, une attaque se déroule à travers 5 phases exprimées par cinq verbes anglais (les 5P) : Probe, Penetrate, Persiste, Propagate et Paralyze [9]. Dans la pratique, le pirate choisit s'arrêter à une phase en fonction de ses objectifs.

Probe : consiste à cartogaphier le réseau cible afin d'obtenir des détails systèmes tels que les ports ouverts, les noms de domaines, les adresses IP, les technologies et dispositifs de sécurité utilisés. Il existe de nombreux outils pour effectuer cette reconnaissance. Les logiciels whois, DNS lookup, maltego par exemple, permettent d'obtenir dans les bases de données publiques les noms de domaines, les adresses ip, sites web, numéros de téléphone et bien d'autres informations utiles. Il est également possible de rechercher les ports ouverts sur une ou plusieurs machines d'un réseau avec le puissant outil Nmap. Par ailleurs, les logiciels tels que Nessus, Nikto permettent d'effectuer un scan de vulnérabilités à exploiter. [9]

Penetrate : utilisation des informations récoltées pour s'introduire dans un réseau grâce à des connaissances de techniques d'attaques. Des techniques comme le bruteforce ou les attaques par dictionnaires peuvent être utilisées pour outrepasser les protections par mot de passe. Le plus simple est de consulter les bases de données des vulnérabilités existantes dont ExploitDB et de les exploiter. [9]

Persist : consiste à s'approprier la machine infiltrée par l'installation d'un dispositif permettant de revenir aisément sur le système. Ce dispositif encore appelé porte dérobée (backdoor en anglais), n'est rien d'autre qu'un programme de contrôle à distance via TCP ou UDP. Netcat est un programme permettant de mettre en place rapidement et aisément des backdoors. Le pirate corrige ensuite la vulnérabilité exploitée afin de s'assurer qu'aucun autre pirate ne puisse s'introduire sur son territoire. [9]

Propagate : cette étape consiste à évaluer les ressources et services accessibles à partir du réseau piraté afin d'étendre son contrôle sur plusieurs systèmes. Le framework Metasploit est souvent utilisé pour cette tâche. [9]

Paralyze : Le pirate peut utiliser le serveur pour mener une attaque sur une autre machine, détruire des données ou encore endommager le système d'exploitation dans le but de planter le serveur. [9]

Après ces cinq étapes, le pirate peut éventuellement tenter d'effacer ses traces, bien que cela ne soit rarement utile. En effet, les administrateurs réseaux sont souvent surchargés de logs à analyser.

De plus, il est très difficile de supprimer entièrement des traces. [9]

1.2.2 Présentation de quelques attaques informatiques

Malgré la diversité des cyber attaques, nous retenons deux grandes familles : les attaques réseaux et les attaques applicatives.[10]

1.2.2.1 Les attaques réseaux

1. **Les techniques de scan**

   Le but des scans est de déterminer quels sont les ports ouverts afin de déduire les services qui sont exécutés sur la machine cible. Il existe un nombre important de techniques de scan. Idéalement, la meilleure technique de scan est celle qui est la plus furtive afin de ne pas éveiller les soupçons de la future victime. Voici une description des techniques de scan les plus répandues :

   Le scan simple : aussi appelé scan connect(), il consiste à établir une connexion TCP complète sur une suite de ports. S'il arrive à se connecter, le port est ouvert ; sinon, il est fermé. Cette méthode de scan est très facilement détectable.

   Le scan furtif : encore appelé scan SYN, il s'agit d'une amélioration du scan simple. Ce scan essaie également de se connecter sur des ports donnés, mais il n'établit pas complètement la connexion : pas de commande ACK (acquittement) après avoir reçu l'accord de se connecter. Grâce à ceci, la méthode est bien plus furtive que le scan normal.

   Les scans XMAS, NULL et FIN : ils se basent sur des détails de la RFC du protocole TCP pour déterminer si un port est fermé ou non en fonction de la réaction à certaines requêtes. Ces scans sont moins fiables que le scan SYN mais ils sont un peu plus furtifs. La différence entre ces trois types de scan se situe au niveau des flags TCP utilisés lors de la requête.

   Le scan à l'aveugle : : il s'effectue via une machine intermédiaire et avec du spoofing (voir plus bas). Le système attaqué pense que le scan est réalisé par la machine intermédiaire et non par le pirate.

   Le scan passif : c'est la méthode la plus furtive. Elle consiste à analyser les champs d'en-tête des paquets (TTL, TOS, MSS, ...) et les comparer avec une base de signatures qui pourra déterminer les applications qui ont envoyé ces paquets.

2. **Usurpation d'identité (spoofing)**

   Cette attaque consiste à se faire passer pour une autre machine en falsifiant ses identifiants de connexion. Parmi les attaques par usurpation les plus connues sont : l'IP spoofing, l'ARP spoofing et le DNS spoofing.

   IP Spoofing : technique qui consiste à remplacer l'adresse IP du pirate par celle d'une autre machine du réseau. Cela lui permet d'envoyer des paquets anonymement afin de compromettre le système cible. Pour ce faire, il forge ses propres paquets IP en modifiant l'adresse IP source. Les réponses envoyées sont récupérées soit par spécification du chemin de routage (source routing) ou par modification de la table des routeurs utilisant le protocole RIP (reroutage).[10]

ARP Spoofing : consiste à rediriger le trafic réseau d'une ou plusieurs machines vers la machine du pirate. Pour ce faire, ce dernier envoie des paquets ARP indiquant une adresse MAC qui lui est accessible en réponse à une requête ARP broadcast. Cela entraine la corruption du cache de la machine victime par la liaison de l'adresse MAC de l'attaquant avec l'adresse IP d'une machine légitime du réseau local. Ainsi, le pirate sera en mesure de recevoir tous les paquets en destination de la machine légitime. [10]

DNS Spoofing : assure la résolution de nom en son adresse IP correspondante et inversement (résolution inverse). Le DNS spoofing consiste à faire parvenir de fausses réponses aux requêtes du serveur DNS cible. Ces réponses permettront de faire correspondre l'adresse IP de l'attaquant au nom de domaine recherché ou de son choix. Pour ce faire, il peut usurper l'adresse IP du serveur DNS ayant autorité sur le nom de domaine et/ou le numéro d'identification DNS. Grâce à cette fausse redirection, l'utilisateur peut envoyer ses identifiants en toute confiance par exemple.[10]

3. **Dénis de service**

Cette attaque vise à rendre indisponible un service (application spécifique) ou machine cible par saturation des ressources allouées. Cela est possible en raison d'un bug ou d'une mauvaise implémentation d'un protocole. Il existe différentes techniques d'attaque de déni de service dont le SYN Flooding, l'UDP Flooding, le Smurfing et le Déni de service distribué.

SYN Flooding : une connexion TCP s'établit en trois phases (TCP Three Handshake) : envoi d'SYN, réception d'un SYN-ACK, et envoi d'un ACK. Le SYN Flooding consiste à laisser un nombre important de connexions TCP en attente sur une machine cible par l'envoi de paquets TCP SYN sans envoyer la réponse ACK. Etant donné que les connexions semi-ouvertes consomment des ressources mémoires, la victime n'est plus en mesure d'accepter les connections ni même d'en établir. [10]

UDP Flooding : cette technique exploite le mode non connecté du protocole UDP. Elle consiste à générer une grande quantité de paquets UDP à destination d'une machine cible ou entre deux machines d'un réseau. Cela entraine la saturation des machines victimes et une congestion du réseau. Etant donné que les paquets UDP sont prioritaires sur les paquets TCP, le trafic UDP sature la bande passante. [10]

Smurfing (attaque par réflexion) : cette méthode consiste à envoyer en broadcast un nombre important de paquets ICMP echo-request sur un réseau en usurpant l'adresse IP d'une machine cible. La réception de tous les paquets ICMP echo-reply de chacune des machines du réseau par la cible entraîne inévitablement la saturation de cette dernière et la congestion du réseau. [10], [11]

Dénis de service distribué (DDoS) : le principe de fonctionnement d'une telle attaque repose sur l'utilisation de plusieurs sources (daemons) contrôlées par des maîtres qui discutent en échangeant des messages spécifiques à l'outil utilisé. Le pirate applique les techniques de bases (SYN Flooding, UDP Flooding, Smurfing) à travers cette architecture afin de saturer ou de congestionner rapidement sa cible tout en masquant son identité. [10]

4. **Détournement de session TCP**

Le but de cette attaque est de rediriger un flux TCP afin de pouvoir outrepasser une protection par mot de passe. Dans un premier temps, le pirate écoute le réseau puis désynchro-

nise la session entre l'utilisateur et le serveur après un délai de n secondes. Pour parvenir à cette désynchronisation, il forge un paquet avec comme adresse IP source, celle de la machine de l'utilisateur et le numéro d'acquittement TCP attendu par le serveur. Etant donné que le contrôle d'authentification s'effectue uniquement à l'ouverture de session, la réussite de cette attaque permet de prendre possession de la connexion pendant toute la durée de la session. [10]

5. **Fragmentation de paquets**

   Le but de cette attaque est de passer outre les protections des équipements de filtrage IP. Ce faisant, un pirate peut par exemple s'infiltrer dans un réseau pour effectuer des attaques ou récupérer des informations confidentielles. On distingue deux types d'attaque sur les fragments IP :

   Fragments overlapping : lorsqu'un message est émis sur un réseau, il est fragmenté en plusieurs paquets IP. Afin de pouvoir reconstruire le message, chaque paquet possède un offset. Le but de l'attaque est de réaliser une demande de connexion et de faire chevaucher des paquets en spécifiant des offsets incorrects. La plupart des filtres analysant les paquets indépendamment ne détectent pas l'attaque. Cependant, lors de la défragmentation, la demande de connexion est bien valide et l'attaque a lieu. [10]

   Tiny fragments : le but de l'attaque est de fragmenter une demande de connexion sur deux paquets IP : le premier paquet de taille minimum (68 octets selon la RFC du protocole IP) ne contient que l'adresse et le port de destination. Le deuxième paquet contient la demande effective de connexion TCP. Le premier paquet est accepté par les filtres puisqu'il ne contient rien de suspect. Quand le deuxième paquet arrive, certains filtres ne le vérifient pas pensant que si le premier paquet est inoffensif, le deuxième l'est aussi. Mais lors de la défragmentation sur le système d'exploitation, la connexion s'établit ! [10]

1.2.2.2 Les attaques applicatives

Les attaques applicatives se basent sur des failles dans les programmes utilisés, ou encore des erreurs de configuration. Toutefois, à l'instar des attaques réseaux, il est possible de classifier ces attaques selon leur provenance.

Les problèmes de configuration : il est très rare que les administrateurs réseaux configurent correctement un programme. En général, ils se contentent d'utiliser les configurations par défaut. Celles-ci sont souvent non sécurisées afin de faciliter l'exploitation du logiciel (ex : login/password par défaut d'un serveur de base de données).

De plus, des erreurs peuvent apparaître lors de la configuration d'un logiciel. Une mauvaise configuration d'un serveur peut entraîner l'accès à des fichiers importants, ou mettant en jeu l'intégrité du système d'exploitation. C'est pourquoi il est important de bien lire les documentations fournies par les développeurs afin de ne pas créer de failles. [10]

Les bugs : liés à un problème dans le code source, ils peuvent amener à l'exploitation de failles. Il n'est pas rare de voir l'exploitation d'une machine suite à une simple erreur de programmation. On ne peut toutefois rien faire contre ce type de problèmes, si ce n'est attendre un correctif de la part du développeur. [10]

Les buffer overflows : les buffer overflows, ou dépassement de la pile, sont une catégorie de bug particulière. Issus d'une erreur de programmation, ils permettent l'exploitation d'un shellcode3 à distance. Ce shellcode permettra à une personne mal intentionnée d'exécuter des commandes sur le système distant, pouvant aller jusqu'à sa destruction.

L'erreur de programmation est souvent la même : la taille d'une entrée n'est pas vérifiée et l'entée est directement copiée dans un buffer dont la taille est inférieure à la taille de l'entrée. On se retrouve donc en situation de débordement, et l'exploitant peut ainsi accéder à la mémoire. [10]

Crackage d'identifiants : cette technique permet de contourner les mécanismes d'authentification des applications et services ou du système lui même. Elle consiste à faire de nombreux essais jusqu'à trouver la bonne combinaison nom d'utilisateur/mot de passe. Pour y arriver le pirate peut soit utiliser un **dictionnaire** ou une **méthode brute (bruteforce)**. Un dictionnaire contient une liste prédéfinie contenant les noms d'utilisateur ou les mots de passe les plus courants. La réussite en utilisant ces listes est incertaine. Mais elle est quasiment certaine avec la méthode brute. L'inconvénient avec cette dernière est qu'elle prend du temps et nécessite une quantité importante de ressources (puissance de calcul et mémoire). [3], [11], [12]

Scripts : cette attaque web consiste à exploiter les failles de scripts dévoloppés dans des langages web afin d'avoir accès à des données confidentielles et critiques d'un site. L'exemple le plus populaire est l'injection de codes malicieux dans les zones de saisie classiques présentées à l'utilisateur. Ces données sont interprétées par l'un des composants de l'application web (SGBD, JS, PHP, ASP, Perl) [10].

1.3 Systèmes de Détection d'Intrusion

1.3.1 Définition générale des IDS

Un Système de Détection d'Intrusion (IDS en anglais) est un dispositif permettant de surveiller et d'analyser les évènements survenant sur un système ou un réseau afin de découvrir les attaques compromettant sa confidentialité, son intégrité ou sa disponibilité. Il alerte le personnel qualité dès qu'il détecte toute activité malveillante (volontaire ou non) ayant pu échapper à la vigilance des pare-feux et des antivirus. Lorsqu'il est en mesure de prendre des dispositions contre les menaces afin de les prévenir, il est appelé Intrusion Prevention System (IPS). [1], [13]

1.3.2 Classification des alertes d'un IDS

- **Vrai positif :** situation dans laquelle l'IDS parvient à détecter effectivement une intrusion qui a eu lieu. C'est le fonctionnement normal et recherché.
- **Vrai négatif :** situation dans laquelle l'IDS ne détecte aucune activité suspecte pour un trafic normal. C'est le fonctionnement normal d'un IDS.
- **Faux positif :** situation dans laquelle l'IDS génère une alerte d'intrusion bien qu'il n'y en ait aucune activité malveillante. Il s'agit d'une erreur de détection

- **Faux négatif :** l'IDS ne détecte pas une attaque quand bien même celle-ci survient dans le système d'information. Il s'agit d'une erreur critique de détection. [1]

1.3.3 Les différents types d'IDS

Selon la littérature, il existe plusieurs classifications des IDS. Elles varient en fonction de leur emplacement, de leur architecture, de la source d'informations et du comportement après détection.[14] La classification basique distingue les IDS en fonction de leur emplacement dans un système d'information. Il s'agit des Host-based IDS, des Network-based IDS et des IDS hybrides.

1.3.3.1 Host-based IDS

Un système de détection hôte (HIDS en anglais) surveille exclusivement les activités au niveau de l'hôte sur lequel il est installé, afin de déceler toute tentative d'intrusion système. Pour ce faire, il analyse les logs d'audit de sécurité, le trafic réseau de l'hôte, les appels systèmes, les processus en cours d'exécution, les changements de configurations des applications, les accès aux fichiers, etc. Le plus souvent, ce type d'IDS est déployé sur les hôtes critiques fournissant des services tels que les serveurs informatiques.

Les HIDS ont l'avantage de détecter avec précision une variété d'attaques et même dans un trafic chiffré. Ils sont cependant plus vulnérables aux attaques de dénis de service. Par ailleurs, l'analyse des logs requiert d'importantes ressources en raison de leurs volumes de données. [3], [13], [15]

1.3.3.2 Network-based IDS

Les IDS réseaux (Network-based Intrusion Detection System (NIDS)) surveillent et protègent tout le réseau informatique. A travers des sondes, ils récupèrent les paquets circulant sur le réseau qu'ils analysent minutieusement afin de déceler tout contenu suspect. Cette catégorie d'IDS est capable de détecter les paquets malveillants furtifs non filtrés par les pare-feux. En effet, les trames ethernet passées au peigne fin, sont disséquées sur toutes leurs couches (réseau, transport, application). Les NIDS peuvent être déployés entre le réseau interne et le pare-feu, le pare-feu et la DMZ ou le pare-feu et internet. Le déploiement d'un NIDS à chacun de ces points stratégiques est certes idéale mais aussi extrêmement coûteux.

Ce type d'IDS est plus facile à protéger contre les attaques qui le ciblent. Cependant il n'est pas en mesure d'écouter les trafics chiffrés. Par ailleurs, l'analyse du trafic de tout un réseau nécessite une bande passante proportionnelle à l'importance du trafic.[3], [13], [15]

1.3.3.3 Les IDS Hybrides

Les IDS de type hybride rassemblent les caractéristiques des HIDS et NIDS. Il s'agit d'une nouvelle tendance en plein essor. Ils sont plus flexibles et augmentent le niveau de sécurité. Les sondes sont placées à des endroits stratégiques du réseau et/ou au niveau de l'hôte. Ils agissent comme des NIDS et/ou des HIDS suivant leurs emplacements. Les données collectées par les sondes sont exportées vers une machine centrale chargée d'effectuer les analyses afin de détecter aussi bien les attaques visant les éléments particuliers que celles visant l'ensemble du système. [3], [15]

1.3.4 Approches de détection d'intrusions

Les IDS utilisent différentes méthodes afin de détecter les intrusions dans un réseau ou un système informatique.

1.3.4.1 Approche par signature ou scénario

Cette technique de détection repose sur la création d'une base de signatures qui caractérisent les attaques déjà produites, publiées et connues de la communauté de cybersécurité. Le système analyse les données collectées par la sonde placée à un niveau stratégique du réseau ou tout simplement au niveau d'une machine en les comparant aux caractérisques des enregistrements de la base. C'est donc un système de reconnaissance de motifs qui permet de révéler ou non la présence d'une intrusion connue par la base de signature. [1]Cette approche est réputée pour son efficacité contre les scénarios d'attaque connus.

Il existe différents types de méthodes de détection par recherche de profil connu d'attaque :

1. **Reconnaissance de motifs (pattern matching)**
   Cette technique est basée sur le principe d'identification d'une série d'évènements ou de marques propre à une attaque via la recherche d'une suite spécifique de caractères dans une chaîne de caractères. En effet, un trafic réseau peut être considéré comme un ensemble de chaines de caractères (représentant les paquets). Détecter une attaque dans un paquet revient donc à trouver la suite de caractères caractérisque de cette attaque. [1], [15]

2. **Systèmes experts**
   Dans cette approche, le profil est établi en observant le flux d'évènements pour en déduire un certain nombre de règles qui décrivent le comportement normal du système. Pendant la phase de détection, le système applique les règles au flux d'évènements et vérifie si ce dernier respecte ou non les règles apprises. [1], [16]

3. **Analyse heuristique**
   Cette approche consiste à détecter à travers une analyse intelligente, toute attaque effectuée par une suite d'essais donnant des résultats approchés avec contrôle de la progression. Elle est par exemple très utile dans la détection d'attaques bruteforces.[17]

1.3.4.2 Approche comportementale

Cette méthode consiste à détecter une intrusion en fonction des activités passées de l'utilisateur. Son principe de base est de dresser le profil normal du système sur la base de la politique de sécurité, du fonctionnement naturel des applications et des habitudes pendant une période dite phase d'apprentissage. Au moment de la phase de détection, tous les évènements avec déviation significative du profil normal sont considérés comme tentative d'intrusion. [15] Cette technique peut être appliquée aussi bien aux applications qu'aux services. L'inconvénient majeur de cette approche réside dans le fait que des attaques pourraient survenir pendant la phase d'apprentissage. De plus, leurs taux de fausses alarmes sont généralement très élevés. Toutefois, elle permet de détecter de nouveaux types d'attaques.

Il existe une variété de techniques de détection comportementale.

1. **Méthode probabiliste**
   L'utilisation courante d'une application ou d'un protocole est représentée grâce à des probabilités établies. Toute activité ne respectant pas le modèle probabiliste provoquera la génération d'une alerte. [1]

2. **Méthode statistique**
   Les profils normaux sont construits à partir des variables aléatoires échantillonnées à intervalles réguliers et on attribue des valeurs statistiques aux différentes variables utilisées. On utilise donc des propriétés et des tests statistiques pour construire la distribution de chaque variable et mesurer le taux de déviation entre le comportement observé et celui attendu.[1], [15], [16]

3. **Immunologie**
   Cette technique fait analogie avec le système immunitaire biologique basé sur le principe de reconnaissance de cellules étrangères. En effet, les services sont observés pendant un temps suffisamment représentatif de manière à établir une base des séquences d'appels normales. [1], [16]

4. **Approche basée sur le machine learning**
   Le Machine Learning est un concept important de l'Intelligence Artificielle. C'est un ensemble de techniques utilisant des données statistiques afin de permettre à une machine d'apprendre à effectuer une tâche de manière autonome et d'améliorer ses performances au fil des expériences. En effet, pendant la phase d'entrainement, les algorithmes de machine learning apprennent à découvrir les caractérisques récurrentes dans les ensembles de données. Les modèles ainsi obtenus sont utilisés sur de nouvelles acquisitions de données. L'une de ses nombreuses applications est la détection d'intrusion par classification des paquets soit en profil normal ou en suspect. Plusieurs algorithmes de Machine Learning sont utilisés dans la détection d'intrusions.

   - **Logistic Regression**
     La régression logistique est un algoritme de classification statistique et probabiliste qui prédit la probabilité d'occurence d'un évènement. Il modèle la relation existant entre l'ensemble des variables d'entrées et une variable dépendante binaire (normal ou anormal). Pour ce faire, il utilise l'équation de la droite de régression des variables indépendantes afin de prédire la probabilité d'occurence d'une classe. Les valeurs prédites sont contraintes dans l'intervalle [0,1] grâce à la fonction logistique (sigmoid). La variable dépendante est déterminée par définition d'une probabilité arbitraire délimitant les zones de chacune des deux classes. [18], [19]

     La formule de la fonction logistique est donnée par :

$$\mathcal{P}(Y|X = 1, 2, 3, ...n) = \frac{\exp^{\beta_0 + \sum_{k=1}^{n} \beta_k X_k}}{1 + \exp^{\beta_0 + \sum_{k=1}^{n} \beta_k X_k}}$$

     [20]
     avec $logit(\mathcal{P}(Y|X = 1, 2, 3, ...n)) = \beta_0 + \sum_{k=1}^{n} \beta_k X_k$ l'équation de la droite de régression à n-1 variables.

- **K-Nearest Neighboors**

  K-Nearest Neighboors est un algorithme de classification de 2 ou plusieurs classes. Pour ce faire, chaque entrée dans l'ensemble de données représente un point. A chacun de ces points correspond une classe connue. Pour déterminer la classe d'une nouvelle entrée, il suffit de chercher les K (nombre à choisir) points plus proches en calculant la distance euclidienne avec les tous les points aux alentours. La classe majoritaire dans l'ensemble des K entrées est celle de la nouvelle entrée.[18], [21]

- **Support Vector Machines**

  C'est un algorithme dont le but est de trouver un hyperplan d'équation : $f(x) = w^T x+b$ qui sépare les données en deux catégories. Il est particulièrement conçu pour la classification binaire. Lorsque les données ne sont pas linéairement séparables, elles sont transformées dans un espace comparatif de paramètres à dimensionnalités plus élevées au moyen d'une fonction noyau. L'hyperplan de décision est déterminé par des vecteurs supports au lieu que par l'ensemble des données à travers la résolution d'un problème d'optimisation mathématique. Pour ce faire, les SVMs utilisent souvent la méthode des multiplicateurs lagrangiens. Les fonctions gaussienne, sigmoide, et polynomiale sont les noyaux utilisés la plupart du temps.[15], [18], [19]

- **Decision Trees**

  Un arbre de décision est un algorithme très simple. Son objectif est de mettre en place un arbre constitué de noeuds et de feuilles. La décision de séparation de l'ensemble de données ou de l'une de ses instances s'effectue de façon récursive à partir d'une condition. Le choix des conditions est basé sur une des caractérisques des données d'entrée et certains paramètres. L'opération récursive s'arrête dès que les feuilles sont homogènes. Il existe plusieurs algorithmes pour la construction d'un tel arbre. Ils diffèrent en fonction de la métrique de séparation utilisée et d'autres facteurs. Néanmoins, ces différentes métriques sont calculées à partir de l'entropie ou de l'indice de gini (uniquement Algorithme CART). Ces deux notions (entropie et indice de gini) sont en quelques sortes les métriques de base. Elles désignent la mesure de l'impureté d'un ensemble. [18], [19]

- **Neural Networks**

  Les réseaux de neurones sont des techniques largement utilisées dans la résolution de problèmes complexes tels que la détection d'intrusion. Ils simulent le fonctionnement du cerveau humain en utilisant des couches de neurones interconnectés pour apprendre à déterminer une variable dépendante suivant une approche supervisée ou non. L'apprentissage supervisé spécifie les données d'entrée et la sortie désirée tandis que l'apprentissage non supervisé ne requiert que l'entrée.

  Ainsi, l'approche basée sur les systèmes neuronaux permet d'apprendre à anticiper les prochaines actions en se basant sur les habitudes propres à un utilisateur du système informatique, sans nécessiter un modèle utilisateur explicite. Cela a conduit à une réduction significative des faux positifs, avec environ une fausse alarme par jour, et à une augmentation du taux de détection à environ 80%. De plus, cette méthode est capable de détecter les anciennes attaques ainsi que celles qui ne sont pas incluses dans l'ensemble de données

d'apprentissage.[15], [16], [18], [22]

- **Techniques d'Ensemble learning**

  L'ensemble learning est une combinaison de plusieurs modèles de machine learning dans la résolution d'un problème. Ces modèles à eux seuls n'ont pas un bon score de performance. L'idée de base repose sur le fait que la combinaison de modèles à faible performance pourrait améliorer le score global. Les techniques baiques d'ensemble learning consistent à entrainer les modèles de base et à combiner les résultats soit par la moyenne (régression) ou par vote (classification). [23]
  Il existe cependant des techniques plus avancées.

  Le Stacking : processus de combinaison de de plusieurs estimateurs dans le but de réduire leur biais. Les prédictions de chaque modèle de base à partir des données d'entrainement sont utilisés comme ensemble de données d'entrée d'un estimateur final appelé meta-model qui détermine la prédiction finale.[23]

  Le Bagging : méthode d'ensemble learning qui sélectionne aléatoirement des échantillons de l'ensemble de données d'apprentissage sur lesquelles sont construits parallèlement des algorithmes. Les résultats des modèles obtenus sont combinés pour obtenir un résultat plus généralisé.[23]

  Le Boosting : technique consistant à réduire le biais et la variance en faisant converger des modèles à faibles performances en estimateurs plus robustes. Les algorithmes sont entrainés de façon séquentielle. En effet, un premier algorithme est construit à partir de l'ensemble des données d'apprentissage. Les algorithmes restants sont entrainés en essayant de minimiser les erreurs générés par le précédent modèle.[23]

1.3.4.3 Approche hybride

Cette approche est la combinaison d'une ou plusieurs méthodes par signature avec une ou plusieurs techniques de détection comportementale. C'est une parade contre les limites respectives des ces approches. Elle renforce la fiabilité et la performance de la détection.

Dans le cadre de notre travail, nous avons proposé une approche qui consiste à utiliser les systèmes experts (les règles snort) pour la recherche de motifs d'attaque dans chacun des paquets réseau qui sont ensuite organisés en flux. Ces derniers sont utilisés pour la détection de profils suspects avec des algorithmes d'apprentissage automatique dont les réseaux de neurones.

1.3.5 Présentation de quelques exemples d'IDS

Il existe une variété de systèmes de détection d'intrusion. Certains sont propriétaires tandis que d'autres sont open-source. En guise d'exemples nous pouvons énumérer les IDS suivants :

- **Symantec Endpoint Security**
  Il s'agit d'une solution all-in-one multi-plateforme. Intégrant un antivirus, un pare-feu et un système de prévention d'intrusion, elle protège les terminaux et le réseau lui-même contre les virus, les pirates et les menaces combinées à travers une gestion centralisée. Le déploiement commun de ces différentes solutions de sécurisation du système d'information des entreprises permet non seulement une protection complète, mais également une administration beaucoup

plus facile. La composante IPS fonctionne au niveau hôte aussi bien qu'au niveau réseau. Elle utilise une base de signatures régulièrement mise à jour afin de bloquer la source de toute activité suspecte. [1], [24]

- **Zeek**
  Aussi connu sous le nom de Bro, c'est un logiciel de détection d'intrusion open-source orienté réseau (NIDS) fonctionnant uniquement sur les systèmes unix. Il est conçu et maintenu par des centres de recherches. En effet, il effectue l'analyse des flux du trafic réseau en temps réel à travers l'analyse de protocoles et le filtrage de motifs d'attaques connus. Cette analyse est effectuée par comparaison de la cartographie des flux de données du réseau à une cartographie modèle. Toute déviation déclenche une alerte. [1], [3], [25], [26]

- **OSSEC**

  L'Open Source SECurity (OSSEC) est un logiciel de détection d'intrusion hôte open source multi-plateforme. Grâce à des règles et au machine learning, il effectue l'analyse des fichiers logs, la vérification de l'intégrité des fichiers, le monitoring du registre Windows, la détection de rootkit. Toute activité malveillante génère automatiquement une alerte qui enclenche une action en fonction de la gravité de l'attaque. Il dispose également d'un moteur d'analyse de différents formats de logs tels que les journaux systèmes, Snort et Apache. [1], [3], [26], [27]

- **Suricata**

  Développé par la fondation OISF (Open Information Security), Suricata est un logiciel open source de détection d'intrusion (IDS), de prévention d'intrusion (IPS), et de supervision de sécurité réseau (NSM) multi-plateforme. Orienté réseau, il permet l'inspection de paquets en profondeur (DPI) provenant d'une ou plusieurs interfaces réseaux en fonction des règles activées. Les fichiers logs sont générés aux formats Unified2, Json et Prelude. Des analyses avancées peuvent être effectuées via des scripts en Lua. [26], [28]

- **Prelude SIEM**

  Propriétaire et publié sous licence GNU GPLv2, Prelude est un Security Information Management System (SIEM) sous Linux, qui permet de superviser la sécurité des systèmes d'information. Cette version est composée des modules ALERTE, ANALYSE, ARCHIVE et ADMIN. A travers le module ALERTE, il collecte, normalise, trie, agrège, corrèle et notifie en temps réel tous les événements de sécurité indépendamment des types d'équipements ou systèmes surveillés. Les données collectées sont analysées suivant l'approche par recherche de motifs et la méthode comportementale. Les journaux et les flux sont ensuite archivés et indexés pour le forensic. Il est également doté d'une interface d'administration et de configuration. [1], [3], [29]

- **Snort**

  Snort est un système de détection d'intrusion libre, multi-plateforme, orienté réseau et hôte, publié sous licence GNU GPL. C'est l'IDS à signatures le plus utilisé aujourd'hui dans le monde. Ce logiciel capture les paquets circulant sur le réseau et les enregistre au format tcpdump ou simplement au format texte. Ces paquets sont inspectés et analysés grâce à un langage de définition de règles intégré. Cette approche de détection pesonnalisée lui permet d'identifier rapidement les attaques et de générer les alertes. Snort peut être aussi utilisé comme un simple analyseur de paquets. [1], [3], [26], [30]

1.3.6 Etude comparative de quelques IDS

Le tableau 1.1 présente les avantages et inconvénients de quelques solutions IDS en vogue.

TABLEAU 1.1 : Tableau comparatif de quelques solutions IDS

[1], [27]-[31]

Nom	**Type**	**Approche de détection**	**Fonctionnement**	**Déploiement**	**Facilité d'utilisation**	**Personnalisation**	**Nouvelles menaces**	**Périmètre**	**Licence**
SNORT	IDS Réseau	Signatures/-Règles	Temps réel	Capteur	Moyenne	Règles personnalisées	Limité	Réseau	GPL
SURICATA	IDS Réseau	Signatures/-Règles	Temps réel	Capteur	Moyenne	Règles personnalisées	Limité	Réseau	GPL
OSSEC	IDS Hôte	Signatures/-Règles, Corrélation hôte	Temps réel	Agent sur hôtes	Moyenne	Règles personnalisées	Limité	Hôte	GPL
PRELUDE SIEM	SIEM	Corrélation multi-sources	Analyse a posteriori	Serveur central	Complexe	-	-	Réseau & Hôte	Commerciale
ZEEK	IDS Réseau	Signatures/-Règles, Scripts	Temps réel	Capteur	Complexe	Scripts personnalisés	Limité	Réseau	BSD
IDS basés sur les modèles de machine learning	IDS Réseau	Modèles d'apprentissage automatique	Temps réel	Capteur	Complexe (Entraînement)	Ré-entraînement des modèles	Possible	Réseau	Variée (Open source majoritaire)

De cette étude, il ressort que les IDS réseau traditionnels comme Snort, Suricata et Zeek excellent dans la détection en temps réel des menaces connues grâce à des signatures et règles prédéfinies. Cependant, leur capacité à détecter les nouvelles formes d'attaques zero-day reste limitée. C'est là que les IDS basés sur l'apprentissage machine peuvent potentiellement apporter un avantage, en modélisant les comportements "normaux" pour détecter les anomalies suspectes, sans dépendre de signatures spécifiques. Néanmoins, leur mise en œuvre reste complexe pour obtenir un bon compromis détection/faux positifs. Dans un autre registre, OSSEC propose une approche complémentaire axée sur l'analyse des logs hôtes et la corrélation d'événements sur plusieurs systèmes. Cette visibilité au niveau des terminaux constitue un atout par rapport aux IDS réseaux. Prelude SIEM va encore plus loin en agrégeant et corrélant de multiples sources d'alertes (IDS, antivirus, firewalls, etc.) pour une vue d'ensemble de la posture de sécurité, mais en mode analyse différé.

Conclusion

Dans ce chapitre, nous avons présenté de manière concise les principaux aspects de la sécurité informatique. De plus, nous avons abordé en détail les systèmes de détection d'intrusion, en mettant en lumière les différents types de systèmes et les approches de détection existants. Cette revue de littérature fournit ainsi une vue complète de la détection d'intrusion dans le contexte de la sécurité informatique. Nous avons également mené une étude comparative approfondie sur quelques IDS/-SIEM open source populaires. Cette analyse a permis de mettre en évidence leurs caractéristiques respectives. L'ensemble de ces éléments établit les fondations nécessaires pour une compréhension approfondie et une analyse ultérieure dans le cadre de ce projet. Dans le prochain chapitre, nous exposerons l'architecture de l'approche hybride proposée.

Chapitre 2

Présentation de l'architecture de l'approche hybride proposée

Introduction

Du chapitre précédent, il ressort clairement que les IDS représentent des moyens nécessaires de sécurisation d'un système d'information. Dans cette partie, nous présentons l'architecture de l'approche hybride proposée.

2.1 Choix du type d'IDS et Méthodologie

Compte tenu de l'étude comparative du chapitre précédent, nous avons opté pour le NIDS open source SNORT auquel nous avons ajouté un module d'analyse comportementale. La flexibilité de ce logiciel est l'une des principales raisons ayant guidé notre choix. C'est d'ailleurs aussi ce qui le rend si populaire. Le module de détection d'anomalies fait appel à des algorithmes de classification polychotomique. Les attaques détectées sont notifiées à l'administrateur par adresse électronique ou/et SMS.

Le fonctionnement global du système proposé dans un réseau informatique typique est illustré par la figure 2.1.

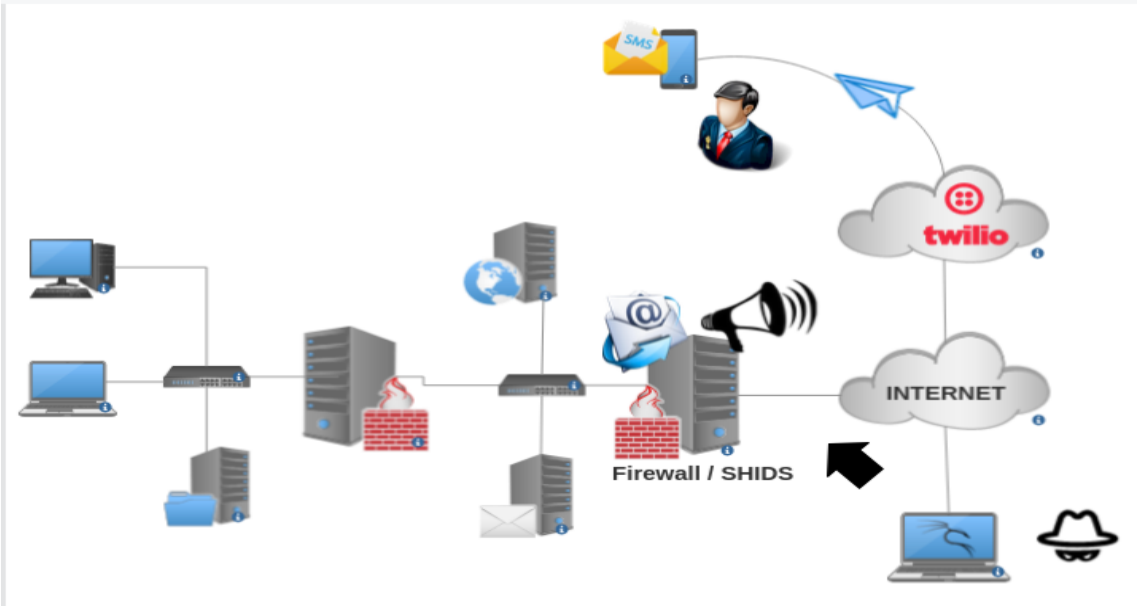


FIGURE 2.1 : Fonctionnement global du Smart Hybrid Intrusion Detection System dans un réseau [32]

2.1.1 Méthodologie

Pour mettre en place ce système hybride, les efforts ont été essentiellement concentrés sur le développement du module comportemental et du module d'envoi d'alertes ainsi que sur l'analyseur des fichiers journaux de l'IDS Snort.

2.1.1.1 Conception du module d'analyse comportementale

La construction de nos algorithmes d'apprentissage automatique s'est effectuée en différentes étapes :

1. **Le choix du dataset**

   Un dataset est un ensemble de données cohérent qui peuvent se présenter sous forme de textes, chiffres, images, vidéos, etc. Ce sont des données statistiques collectées dans un domaine donné afin de déterminer une corrélation à l'aide de modèles de machine learning. En fonction du type d'apprentissage pour lequel il est destiné, il peut être labelisé ou non. En dépit de la difficulté de collecte de données pour raison de confidentialité, il existe plusieurs datasets dans le domaine de la détection d'intrusion. Les plus populaires sont :

   - KDD Cup 1999 ;
   - NSL-KDD ;
   - UNSW-NB15 ;
   - CSE-CIC-IDS2017 ;
   - CSE-CIC-IDS2018

   Ces différents datasets regroupent différents types de données dont les paquets réseau sous forme de flux réseau. Afin de permettre la détection des attaques, les flux sont classés deux différents groupes : les flux normaux et les flux suspects. Ces flux sont générés de façon synthétique dans des environnements simulés avec des outils qui extraient des caractéristiques

propres à chaque dataset.

Notre choix s'est porté sur le dataset CICIDS2018[33]. En effet, ce dernier a été généré du 14 au 23 février 2018, le 28 du même mois, ainsi que les 01 et 02 mars 2018 sous forme de fichiers pcap. Ensuite ces fichiers de capture de paquets ont été transformés en flux de données enregistrés au format csv suivant 80 caractéristiques extraites décrits dans le tableau 2.1 [34]

Ce jeu de données regroupe une gamme variée d'attaques informatiques notamment les attaques par bruteforce, les attaques web, les dénis de service distribués et non distribués, les botnets et les infiltrations. Contrairement aux autres datasets, il est plus aisé d'extraire ses caractéristiques en situation réelle.

TABLEAU 2.1 : Tableau descriptif des caractérisques du dataset

Caractéristiques	**Description**
dst_port	Port de destination
protocol	Protocole réseau utilisé
timestamp	Horodatage
flow_duration	Durée du flux
tot_fwd_pkts	Nombre total de paquets envoyés
tot_bwd_pkts	Nombre total de paquets reçus
totlen_fwd_pkts	Taille totale des paquets envoyés
totlen_bwd_pkts	Taille totale des paquets reçus
fwd_pkt_len_max	Taille maximale des paquets envoyés
fwd_pkt_len_min	Taille minimale des paquets envoyés
fwd_pkt_len_mean	Taille moyenne des paquets envoyés
fwd_pkt_len_std	Écart-type de la taille des paquets envoyés
bwd_pkt_len_max	Taille maximale des paquets reçus
bwd_pkt_len_min	Taille minimale des paquets reçus
bwd_pkt_len_mean	Taille moyenne des paquets reçus
bwd_pkt_len_std	Écart-type de la taille des paquets reçus
flow_byts_s	Débit en octets
flow_pkts_s	Débit en paquets
flow_iat_mean	Temps inter-arrivée moyen des paquets
flow_iat_std	Écart-type du temps inter-arrivée des paquets
flow_iat_max	Temps inter-arrivée maximal des paquets
flow_iat_min	Temps inter-arrivée minimal des paquets
fwd_iat_tot	Temps inter-arrivée total des paquets envoyés
fwd_iat_mean	Temps inter-arrivée moyen des paquets envoyés
fwd_iat_std	Écart-type du temps inter-arrivée des paquets envoyés
fwd_iat_max	Temps inter-arrivée maximal des paquets envoyés
fwd_iat_min	Temps inter-arrivée minimal des paquets envoyés
Suite à la page suivante...	

Caractéristiques	Description
bwd_iat_tot	Temps inter-arrivée total des paquets reçus
bwd_iat_mean	Temps inter-arrivée moyen des paquets reçus
bwd_iat_std	Écart-type du temps inter-arrivée des paquets reçus
bwd_iat_max	Temps inter-arrivée maximal des paquets reçus
bwd_iat_min	Temps inter-arrivée minimal des paquets reçus
fwd_psh_flags	Drapeaux PSH des paquets envoyés
bwd_psh_flags	Drapeaux PSH des paquets reçus
fwd_urg_flags	Drapeaux URG des paquets envoyés
bwd_urg_flags	Drapeaux URG des paquets reçus
fwd_header_len	Taille d'en-tête des paquets envoyés
bwd_header_len	Taille d'en-tête des paquets reçus
fwd_pkts_s	Débit des paquets envoyés
bwd_pkts_s	Débit des paquets reçus
pkt_len_min	Taille minimale des paquets
pkt_len_max	Taille maximale des paquets
pkt_len_mean	Taille moyenne des paquets
pkt_len_std	Écart-type de la taille des paquets
pkt_len_var	Variance de la taille des paquets
fin_flag_cnt	Nombre de drapeaux FIN
syn_flag_cnt	Nombre de drapeaux SYN
rst_flag_cnt	Nombre de drapeaux RST
psh_flag_cnt	Nombre de drapeaux PSH
ack_flag_cnt	Nombre de drapeaux ACK
urg_flag_cnt	Nombre de drapeaux URG
cwe_flag_count	Nombre de drapeaux CWE
ece_flag_cnt	Nombre de drapeaux ECE
down_up_ratio	Ratio download/upload
pkt_size_avg	Taille moyenne des paquets
fwd_seg_size_avg	Taille moyenne des segments envoyés
bwd_seg_size_avg	Taille moyenne des segments reçus
fwd_byts_b_avg	Débit moyen en octets envoyés
fwd_pkts_b_avg	Débit moyen en paquets envoyés
fwd_blk_rate_avg	Taux moyen de blocage envoyé
bwd_byts_b_avg	Débit moyen en octets reçus
bwd_pkts_b_avg	Débit moyen en paquets reçus
bwd_blk_rate_avg	Taux moyen de blocage reçu
subflow_fwd_pkts	Paquets envoyés des sous-flux
subflow_fwd_byts	Octets envoyés des sous-flux
subflow_bwd_pkts	Paquets reçus des sous-flux
subflow_bwd_byts	Octets reçus des sous-flux

Suite à la page suivante...

Caractéristiques	Description
init_fwd_win_byts	Octets initiaux de la fenêtre envoyée
init_bwd_win_byts	Octets initiaux de la fenêtre reçue
fwd_act_data_pkts	Paquets de données actifs envoyés
fwd_seg_size_min	Taille minimale des segments envoyés
active_mean	Temps moyen actif
active_std	Écart-type du temps actif
active_max	Temps maximal actif
active_min	Temps minimal actif
idle_mean	Temps moyen inactif
idle_std	Écart-type du temps inactif
idle_max	Temps maximal inactif
idle_min	Temps minimal inactif

2. **La préparation et le pré-traitement des données**

   Les fichiers du dataset étant volumineux et contenant des catégories d'attaque très mal distribuées, nous avons procédé à une sélection. Puisque les profils anormaux étaient en minorité, nous les avons tous sélectionnés exceptés ceux du fichier csv du 20 février. Afin d'obtenir un échantillon des profils normaux, nous avons concatené tous les flux normaux des fichiers csv utilisés précédemment, puis en avons choisi un certain nombre de manière aléatoire.

   Après échantillonnage, nous les avons concatenés tout en regroupant les sous-catégories d'attaque de telle sorte que les classes soient désormais : Benign, DDoS, DoS, BruteForce, Web-Attack, Bot et Infiltration. Les données ont ensuite été normalisées et les labels encodés.

   Ce n'est qu'après cela que l'ensemble de données a été divisé en deux sous-ensembles : les données d'entrainement et de validation.

3. **L'entrainement et l'évaluation de différents modèles**

   Quatre modèles de d'apprentissage automatique ont été utilisés : Decision Tree, XGBOOST, Random Forest et Extra Trees avec la bibliothèque scikit-learn.

   Les modèles Decision Tree, Random Forest et Extra Trees ont été entrainés avec les paramètres par défaut.

   XGBoost a été entraîné avec 100 arbres de décision dans l'ensemble, utilisant une méthode d'histogramme pour construire les arbres, et bénéficiant de l'accélération matérielle CUDA.

   Voici un résumé de l'évaluation des performances de chacun des modèles sur l'ensemble des données de validation :

TABLEAU 2.2 : Performances des modèles ML entrainés

Modèle ML	Précision	Recall	F1-score
Decision Tree	0.961	0.961	0.961
Random Forest	0.973	0.972	0.972
Extra Trees	0.967	0.966	0.966
XGBoost	**0.979**	**0.976**	**0.976**

Sur la base de ces résultats, notre choix s'est porté sur **XGBOOST**, en raison de ses performances supérieures en termes de précision, rappel et score F1.

4. **Le déploiement et l'utilisation des modèles**

   Les algorithmes développés ont été intégrés à une Application Programming Interface (API) REST créée avec le module python **flask** que nous avons déployé en local. Les données du trafic générées et envoyées en temps réel à l'api par l'outil cicflowmeter[35] réécrit en python[36] et adapté à notre contexte. L'Application Programming Interface (API) se charge de leur prétraitement après le quel elles sont envoyées en entrée au modèle de classification XGBOOST entrainé. La réponse est envoyée au module de détection comportementale avec les précisions suivantes : **adresse IP source**, **port de destination**, **prédiction** et **probabilité**

2.1.1.2 Conception du parser et analyseur de fichiers journaux SNORT

Le module d'analyse par scénario utilise les fichiers journaux d'alertes de l'IDS SNORT. Afin de l'intégrer à notre système, nous avons personnalisé l'outil snort log reader [37]. De base, il permet de lire les logs snort et de générer des notifications par email et par script. Nous avons modifié le code de telle sorte que les alertes snort soient ajoutées à la queue des alertes qui seront notifiées à l'administrateur via le module de notification d'alertes.

2.1.1.3 Conception du module d'alertes

Les alertes sont envoyées par 3 différents canaux : l'adresse électronique, le numéro de téléphone de l'administrateur et par l'émission d'alarme vocale. Cette diversification permet de s'assurer que l'administrateur reçoit en temps réel les alertes quel que soit son emplacement. Processus exécuté en parallèle avec ceux des autres modules, il se charge de récupérer les données de la queue d'alertes et d'envoyer un message électronique à partir d'un compte GMAIL (devmail.chris@gmail.com). Par ailleurs, les alertes sont notifiées par SMS grâce à twilio[38]. Il s'agit d'une plate-forme de communications via le Cloud qui permet de créer et de recevoir des appels téléphoniques par programmation et également d'envoyer et de recevoir des messages texte ou SMS, et même des MMS (Multimedia Messaging System).

2.2 Environnement et matériels

2.2.1 Machine de développement

Notre système a été entièrement codé en Python, utilisant plusieurs modules. La conception des modules du système a nécessité l'utilisation de divers matériels, en particulier un ordinateur portable ASUS Zenbook 15 Pro UX535 équipé d'un processeur Intel® Core™ i7-10750H avec 12 processeurs logiques, ainsi qu'une carte graphique NVIDIA® GeForce® GTX 1650 Ti avec Max-Q Design et 4 Go de GDDR6. L'accélération matérielle CUDA a été configurée pour exploiter la puissance de calcul de la carte graphique NVIDIA.

2.2.2 Environnement de tests

Pour la mise en place du laboratoire de test, nous avons opté pour l'utilisation de deux machines Linux. La première est une machine virtuelle exécutée via l'hyperviseur VirtualBox. Sur cette machine, nous avons installé la distribution Ubuntu et configuré un service web ainsi que notre IDS.

La deuxième machine est notre machine physique exécutant la distribution Linux spécialisée dans les tests d'intrusion, Kali Linux. C'est depuis cette machine que nous avons effectué les attaques.

Conclusion

Dans ce chapitre, nous avons présenté en détail l'architecture de l'approche hybride que nous proposons. Ensuite, nous avons exposé la méthodologie adoptée. Elle s'articule autour des la conception des modules d'anomalies, du lecteur de journaux et d'un module d'alerte. Ces éléments constituent les fondements de notre IDS, offrant une solution robuste la détection d'intrusion.

Chapitre 3

Mise en place de la solution et évaluation de performances

Introduction

Dans ce chapitre nous abordons la mise en place du système choisi. Ensuite nous nous sommes appliqués à effectuer les configurations basiques nécessaires. Au final, nous avons testé notre configuration en lançant quelques attaques et essayé de les détecter.

3.1 Configuration de l'environnement de test

La configuration de l'environnement test a été réalisée à travers la virtualisation de l'architecture du réseau-cible sur notre machine physique faisant office de machine du hacker. Cette dernière héberge la version 2023.1 du système Kali Linux.

3.1.1 Mise en place de l'architecture réseau cible

L'architecture du réseau cible a été mise en place en utilisant le logiciel VirtualBox, installé sur notre unique machine physique. Une machine virtuelle a été créée, le système d'exploitation y a été installé puis mis à jour. Son adaptateur réseau a été configuré en mode **Accès par pont** afin de permettre la communication avec la machine hôte tout en conservant l'accès à Internet.

Les différents services ont été installés et configurés sur les machines adéquates. Plus précisément, le logiciel apache2 a été installé et configuré sur le serveur web. Le service Secure Shell (SSH) y a été aussi configuré.

3.1.2 Mise en place du Système de Détection d'Intrusion

3.1.2.1 Installation des dépendances

Le fonctionnement de notre système de détection requiert l'installation de quelques dépendances comme suit :

TABLEAU 3.1 : Les étapes d'installation des dépendances.

Commandes	Remarques
sudo apt update	Mise à jour de la liste des paquets disponibles dans les dépôts APT
sudo apt install gcc perl make sox libsox-fmt-mp3 net-tools build-essential libpcap-dev libpcre3-dev libnet1-dev zlib1g-dev luajit hwloc libdnet-dev libdumbnet-dev bison flex liblzma-dev openssl libssl-dev pkg-config libhwloc-dev cmake cpputest libsqlite3-dev uuid-dev libcmocka-dev libnetfilter-queue-dev libmnl-dev autotools-dev libluajit-5.1-dev libunwind-dev python3-pip	Installation des paquets prérequis
- git clone https://github.com/snort3/libdaq.git - cd libdaq - ./bootstrap - ./configure - make && sudo make install - cd ../	Installation librairie DAQ (Data Acquisition)
- git clone https://github.com/gperftools/gperftools - cd gpertftools - ./autogen.sh - ./configure - make && sudo make install - cd ../	Installation de TCMalloc
- pip3 install numpy pandas joblib flask scipy scapy twilio xgboost –upgrade - pip3 install scikit-learn	Installation des modules python pour la détection comportementale

3.1.2.2 Installation et configuration de SNORT3

Dans un premier temps, nous avons installé le logiciel SNORT3 [39]. Ensuite nous avons procédé à son paramétrage.

- **Installation**

  L'installation de l'outil s'est effectuée suivant les étapes présentées dans le tableau 3.2 :

  TABLEAU 3.2 : Les étapes d'installation de SNORT3.

Commandes	Remarques
git clone https://github.com/snort3/snort3.git	Téléchargement de la dernière version de snort3
- cd snort3 - ./configure_cmake.sh –enable-tcmalloc - cd build - make && make install	Compilation des fichiers sources et installation

  Une fois toutes ces étapes effectuées, nous avons exécuté la commande **ldconfig** afin de mettre à jour les librairies partagées.
  La dernière étape consiste à s'assurer de la réussite de l'installation en vérifiant la version de SNORT3 installée grâce à la commande **snort -V** (voir figure 3.1)

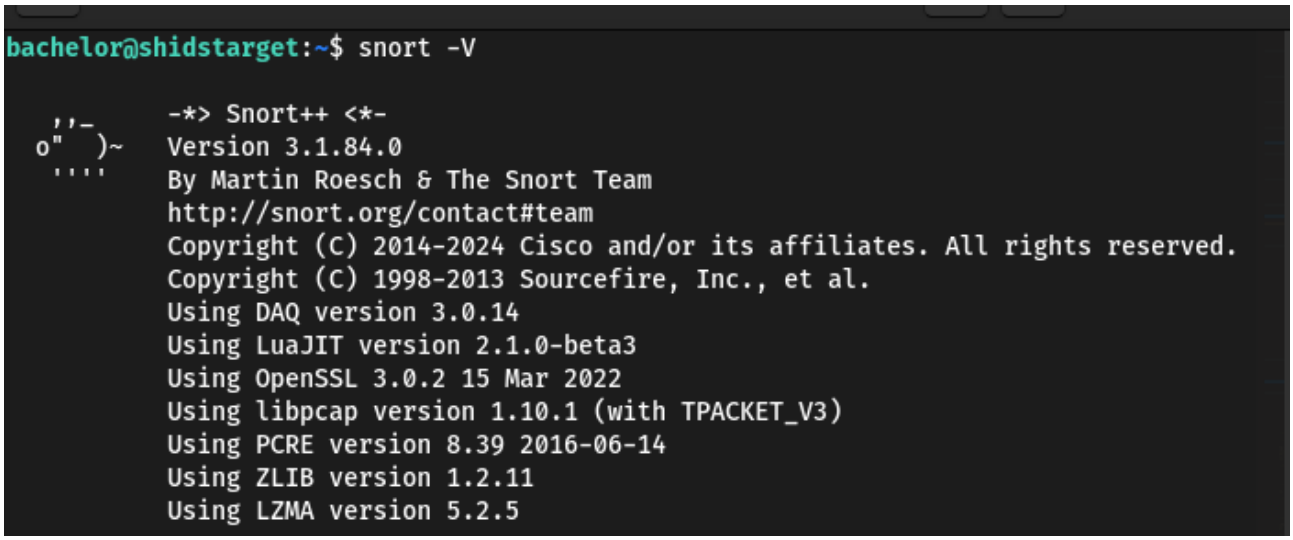


FIGURE 3.1 : Réussite de l'installation de Snort

Ainsi s'achève l'installation du système de détection d'intrusions réseau SNORT3 [40]. Nous sommes passés à sa configuration.

- **Configuration**

  Les cartes réseau modernes utilisent l'offloading (LRO par exemple) pour gérer le réassemblage des paquets réseau au niveau du matériel plutôt que du logiciel. Dans la plupart des cas, cette solution est préférable car elle réduit la charge sur le système. Pour un NIDS, il est nécessaire de désactiver les fonctionnalités LRO (large-receive-offload) et GRO (generic-receive-offload) de l'interface d'écoute, car cela peut tronquer des paquets plus longs.
  Par ailleurs, nous devons mettre l'interface en mode promiscuous afin que snort puisse voir tout le trafic réseau qui lui est envoyé même si il n'en est pas le destinataire. Pour effectuer ces configurations systèmes, nous avons créé un service systemD afin de s'assurer que les changements persistent au redémarrage du système.

Création du script du service systemD

```
$ sudo vi /etc/systemd/system/snort3-nic.service
```

Nous allons ensuite entrer les informations suivantes :

Script snort3-nic.service

```
[Unit]
Description=Set Snort 3 NIC in promiscuous mode and Disable GRO, LRO on boot
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/sbin/ip link set dev enp0s3 promisc on
ExecStart=/usr/sbin/ethtool -K enp0s3 gro off lro off
TimeoutStartSec=0
RemainAfterExit=yes

[Install]
WantedBy=default.target
```

Le script étant créé, nous avons rechargé les paramètres de configuration de systemd puis activé et démarré notre service.

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable –now snort3-nic
```

Nous avons vérifié ensuite que les changements ont été pris en compte.

```
bachelor@shidstarget:~$ sudo ip add sh enp0s3
2: enp0s3: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f4:db:dd brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.143/24 metric 100 brd 192.168.1.255 scope global dynamic enp0s3
       valid_lft 41578sec preferred_lft 41578sec
    inet6 fe80::a00:27ff:fef4:dbdd/64 scope link
       valid_lft forever preferred_lft forever
bachelor@shidstarget:~$
bachelor@shidstarget:~$ sudo ethtool -k enp0s3 |grep receive-offload
generic-receive-offload: off
large-receive-offload: off [fixed]
bachelor@shidstarget:~$
```

FIGURE 3.2 : Vérification de l'effectivité des changements sur la carte réseau

Par la suite, nous avons créé des dossiers requis pour l'intégration des règles et la journalisation.

```
$ sudo mkdir /usr/local/snort/etc/rules
$ sudo mkdir /usr/local/snort/etc/so_rules/
$ sudo mkdir /var/log/snort/
```

L'étape suivante est le téléchargement des règles snort. Afin d'assurer leurs mises à jour régulières, nous avons utilisé le plugin PulledPork3 [41], une adaptation en python du logiciel de téléchargement automatique de règles snort2 PulledPork codé en Perl. Les nouvelles règles sont automatiquement prises en compte sans nécessité de redémarrage.

Installation PulledPork3

```
$ git clone https ://github.com/shirkdog/pulledpork3.git
$ cd pulledpork3/
$ sudo mkdir /usr/local/snort/etc/pulledpork3/
$ sudo cp etc/pulledpork.conf /usr/local/snort/etc/pulledpork3/
$
$ sudo mkdir /usr/local/snort/bin/pulledpork3/
$ sudo cp -r lib/ /usr/local/snort/bin/pulledpork3/
$ sudo cp pulledpork.py /usr/local/snort/bin/pulledpork3
$ sudo chmod +x /usr/local/snort/bin/pulledpork3/pulledpork.py
```

Avant de continuer, nous devons nous enregistrer sur le site de snort afin d'obtenir un oinkcode qui nous permettra d'obtenir un ensemble de règles récent connu sous le nom de "LightSPD ruleset".

Il faudra ensuite éditer notre fichier pulledpork.conf.

```
$ sudo vi /usr/local/snort/etc/pulledpork3/pulledpork.conf
```

Ce fichier est subdivisé en plusieurs sections avec un certain nombre de paramètres optionnels. Nous avons opté pour une configuration basique. Nous avons activé le téléchargement de l'ensemble de règles LightSPD en changeant la valeur du paramètre LightSPD_ruleset à **true** :

```
3 community_ruleset = false
4 registered_ruleset = false
5 LightSPD_ruleset = true
```

Après avoir renseigné l'oinkcode obtenu suite à l'inscription, nous avons commenté la ligne 19 afin de désactiver effectivement le téléchargement de listes noires :

```
19 # blocklist_path = /usr/local/snort/etc/lists/default.blocklist
```

Par ailleurs, nous devons décommenter la ligne 74 afin d'inclure toutes les règles téléchargées :

```
74 include_disabled_rules = true
```

Il faudra également décommenter la ligne 40 pour permettre à snort de charger automatiquement les nouvelles règles téléchargées :

```
40 pid_path=/var/log/snort/snort.pid
```

Ensuite, nous avons lancé pulledpork3 pour procéder au téléchargement :

```
$ sudo /usr/local/snort/bin/pulledpork3/pulledpork.py -c /usr/local/snort/etc/pulledpork3/pulledpork.conf
```

Pour finaliser la configuration de PulledPork3, nous avons créé une tâche cron.

```
$ sudo crontab -e
```

Nous avons ensuite ajouté la ligne suivante :

```
01 04 * * * /usr/local/snort/bin/pulledpork3/pulledpork.py -c /usr/local/snort/etc/pulledpork3/pulledpork.conf
```

Cette commande planifie le lancement de PulledPork quotidiennement à 04 : 01 am.

Nous pouvons à présent passer à la configuration proprement dite de snort lui même. Ainsi donc, nous avons édité les lignes 24 et 28 pour spécifier l'adresse réseau interne à protéger et celles considérées comme hostiles. Ici, notre réseau à protéger est celui d'adresse 192.168.1.0/24. Toutes les adresses ip sont considérées comme hostiles.

```
$ sudo vi /usr/local/snort/etc/snort/snort.lua
```

```
-- 1. configure defaults

-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = '192.168.1.0/24'

-- set up the external network addresses.
-- (leave as "any" in most situations)
EXTERNAL_NET = 'any'

include 'snort_defaults.lua'
```

FIGURE 3.3 : Paramétrage des adresses de réseaux hostiles et à protéger

Dans la section ips, nous avons intégré les règles téléchargées par pulledpork3 en ajoutant de nouvelles lignes. Après modification, la section ips se présente comme suit :

```
ips =
{
    -- use this to enable decoder and inspector alerts
    --enable_builtin_rules = true,

    -- use include for rules files; be sure to set your path
    -- note that rules files can include other rules files
    -- (see also related path vars at the top of snort_defaults.lua)
    include = RULE_PATH .. "/pulledpork.rules",
    variables = default_variables
}
```

FIGURE 3.4 : Intégration des règles téléchargées

Au niveau de la section 7 enfin, nous avons configuré le format de la journalisation des alertes générées par snort en ajoutant de nouvelles lignes. Après modifications, la section ips se présente comme suit :

```
253  alert_json = {
254      file = true,
255      limit = 10,
256      fields = 'seconds timestamp action class b64_data dir dst_addr dst_ap dst_port
eth_dst eth_len eth_src eth_type gid icmp_code icmp_id icmp_seq icmp_type iface ip_id
ip_len msg mpls pkt_gen pkt_len pkt_num priority proto rev rule service sid src_addr
src_ap src_port target tcp_ack tcp_flags tcp_len tcp_seq tcp_win tos ttl udp_len vlan',
257      separator = "," ,
258  }
```

Il ne nous reste plus qu'à tester notre configuration. Nous devons spécifier l'emplacement des fichiers de règles compilés avec l'option **–plugin-path** ainsi que le chemin vers le fichier de configuration de snort avec l'option **-c** :

```
$ snort -c /usr/local/snort/etc/snort/snort.lua –plugin-path /usr/local/s-
nort/etc/so_rules/
```

```
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
bachelor@shidstarget:~$
```

FIGURE 3.5 : Réussite du test de configuration

Ce message de succès marque la fin de cette partie.

3.1.2.3 Initialisation de l'IDS

Après l'installation du module de détection par signatures, nous avons initialisé notre système. Mais pour ce faire, nous avons d'abord dû effectuer quelques étapes supplémentaires. Afin que le module comportemental puisse fonctionner correctement, il nous faut lancer le serveur de flask sur lequel est hébergé notre API de prédiction.

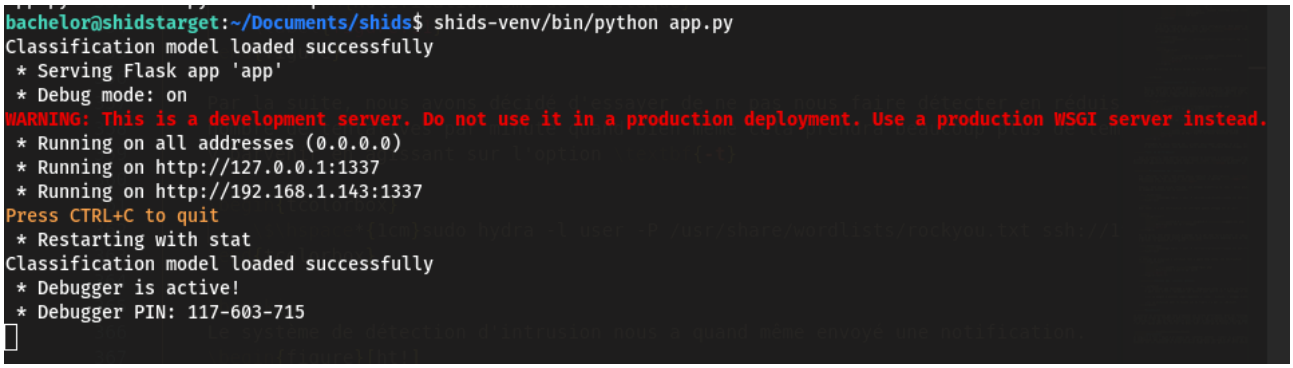


FIGURE 3.6 : Lancement de l'API de de prédiction

Ce n'est qu'après cela que nous avons démarré l'IDS en tant qu'utilisateur root.

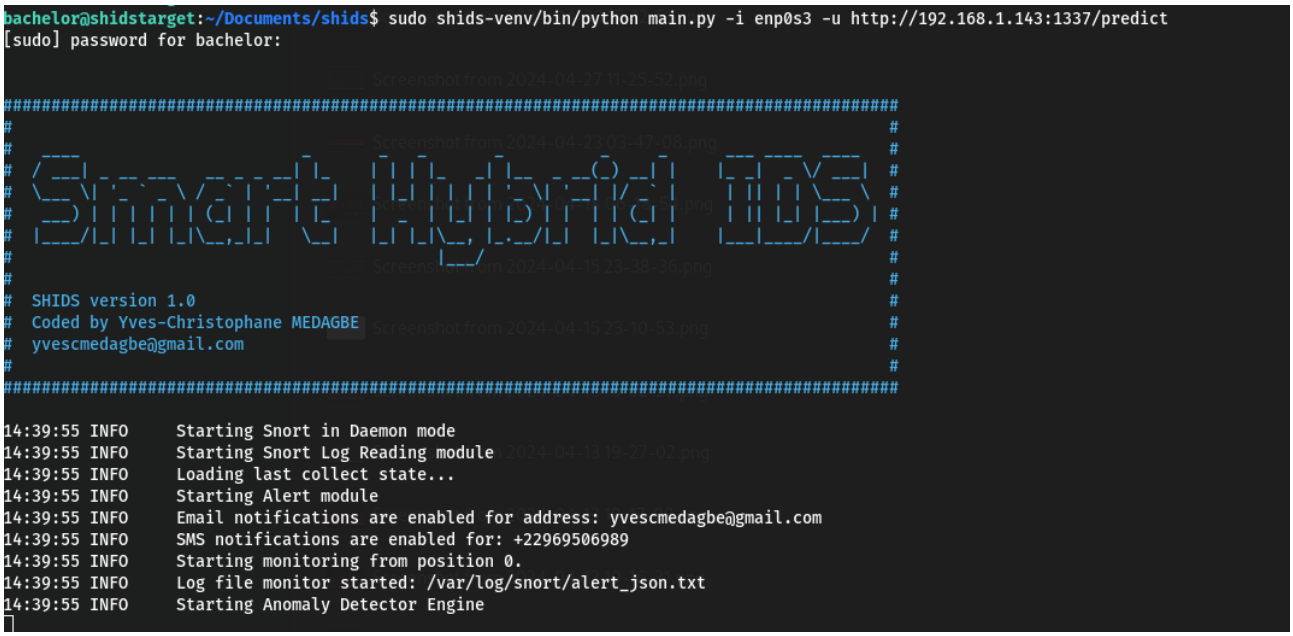


FIGURE 3.7 : Initialisation du Smart Hybrid Intrusion Detection System

L'ensemble de notre système est à présent entièrement configuré et prêt à être testé. Dans la prochaine et dernière section, nous mettons notre système à l'épreuve d'une attaque informatique.

3.2 Tests de performances

Dans ce chapitre, nous mettons notre solution de détection d'intrusions à l'épreuve. L'objectif n'étant pas d'éprouver le niveau de sécurité du système d'informations mais plutôt les performances de notre solution, nous avons laissé trainer volontairement quelques vulnérabilités dans le système.

3.2.1 Présentation des résultats

Pour effectuer le test, nous avons choisi exploiter la vulnérabilité **File inclusion** de notre application web. Pour ce faire, nous avons injecté le payload **?page=/etc/passwd** dans l'url. Nous avons découvert l'existence d'un utilisateur **bachelor**.

Ensuite, nous avons décidé de mener une attaque bruteforce avec l'outil ghidra afin de retrouver le mot de passe de ce dernier. La commande utilisée est la suivante :

```
$ sudo hydra -l user -P /usr/share/wordlists/rockyou.txt ssh ://192.168.1.143 -t 3
```

Nous avons reçu une notification sms informant qu'une attaque est en cours :

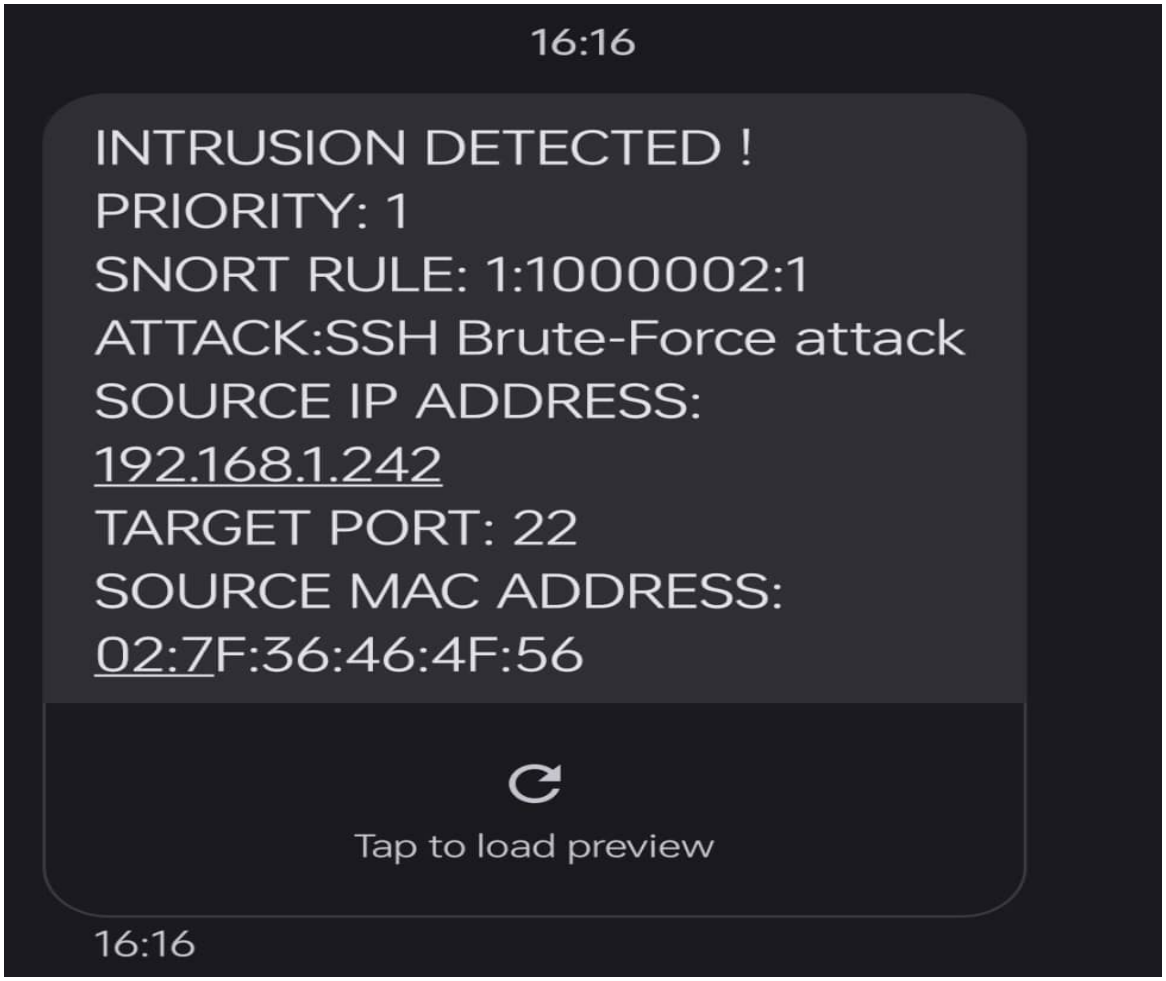


FIGURE 3.8 : Notification sms de l'attaque

Nous avons ensuite procédé à la désactivation des règles de détection des attaques par force brute (bruteforce) au sein de l'outil Snort3. Cette décision était basée sur l'hypothèse que les règles spécifiques à ce type d'attaques pourraient ne pas être présentes (écrites). Après cette modification, nous avons relancé la même attaque afin d'évaluer les capacités de détection de notre solution.

```
$ sudo hydra -l user -P /usr/share/wordlists/rockyou.txt ssh ://192.168.42.27 -t 3
```

Le système de détection d'intrusion nous a quand même envoyé une notification.

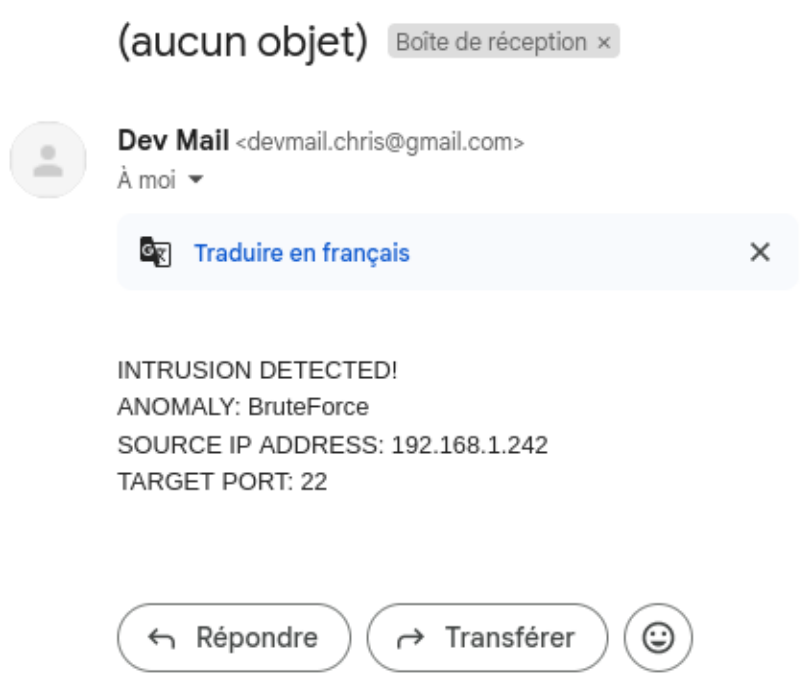


FIGURE 3.9 : Notification d'alerte par mail

La figure ci-dessous atteste que des requêtes ont effectivement été transmises à l'API de prédiction de notre module, confirmant son rôle clé dans la détection de cette attaque par force brute.

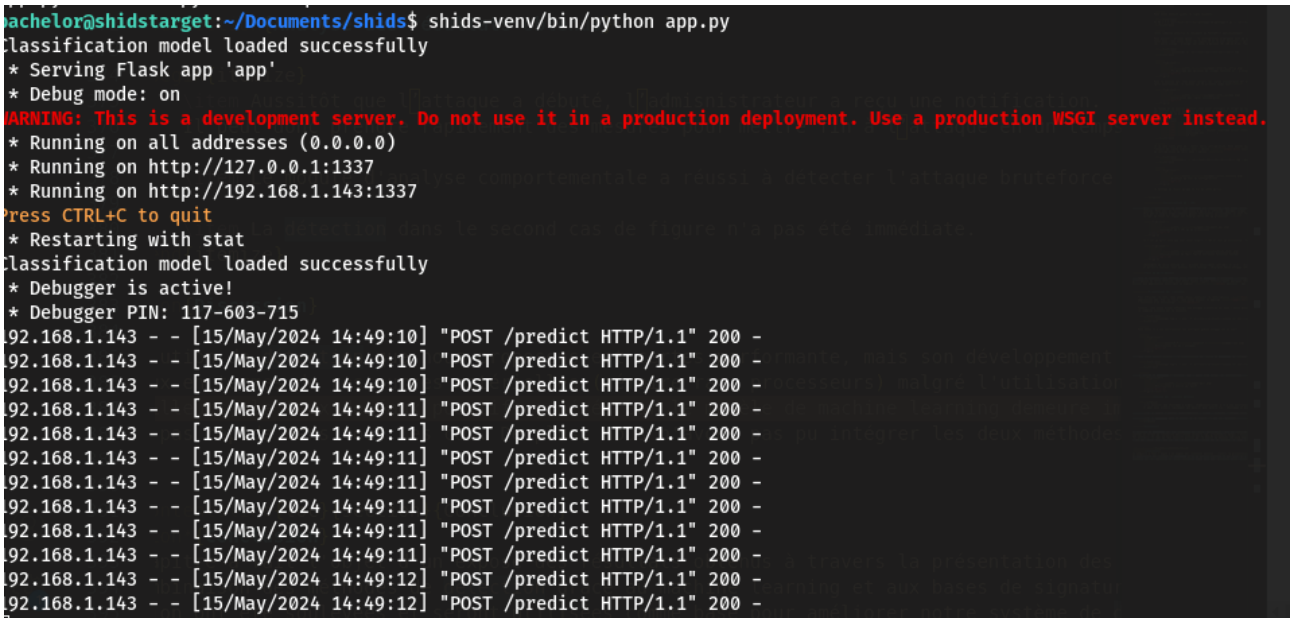


FIGURE 3.10 : Requêtes du module d'analyse comportementale vers l'API de prédiction

3.2.2 Analyse des résultats obtenus

Dès le lancement de l'attaque par force brute, l'administrateur a reçu une notification d'alerte. Cela lui a permis de réagir promptement en prenant les mesures appropriées pour rapidement mettre un terme à cette intrusion.

Malgré la désactivation de la règle Snort dédiée à la détection des attaques de bruteforce contre le service SSH, notre module d'analyse comportementale a réussi à identifier cette activité malveillante. Ceci démontre que c'est bien ce composant d'apprentissage automatique qui est à l'origine de la notification d'alerte envoyée.

3.3 Discussion

La solution de détection que nous proposons est certes performante, mais son développement a été très couteux en termes de ressources matérielles (mémoire ram, processeurs) surtout parce que le dataset est volumineux. Par ailleurs, le taux de faux positifs générés par le modèle de machine learning demeure non négligeable. De plus, nous n'avons pas pu intégrer les deux méthodes de détection basiques de façon enchainée.

Conclusion

Ce chapitre a fait l'objet d'un exposé des résultats obtenus à travers la présentation des performances d'une approche de combinaison des méthodes de détection grâce au machine learning et aux bases de signatures. Les insuffisances liées à la solution ont été soulevées et seront utilisées comme base pour améliorer notre système de détection d'intrusion. Le code source de l'application est disponible sur github à l'adresse suivante : `https://github.com/ychrism/SHIDS`

Conclusion générale et Perspectives

La solution proposée est un puissant détecteur d'intrusion hybride alliant astucieusement les techniques de détection par signatures et par modèles d'apprentissage automatique. Cette solution permet de tirer parti des forces complémentaires de ces deux méthodes. D'une part, la détection par signatures excelle pour identifier efficacement les menaces connues. D'autre part, les modèles d'apprentissage machine permettent de détecter des comportements anormaux et anomalies potentiellement révélateurs de nouvelles cyberattaques inédites. Les expérimentations menées ont validé les bonnes performances de détection de notre système hybride, tout en démontrant une relative simplicité de déploiement dans les environnements réseaux existants.

Si notre solution apporte déjà une brique supplémentaire à l'édifice de cyberdéfense, nous sommes conscients que des améliorations restent possibles. Nos futurs travaux viseront à :

- améliorer le score de détection avec une meilleure technique d'apprentissage automatique ;
- intégrer une interface d'administration web ;
- optimiser la synergie entre les approches signatures, comportementale et par scénarios d'attaque ;
- aller au-delà de la détection en programmant des actions d'auto-défense du système.

Bibliographie

[1] S. WARAKPE, "Implémentation de SNORT/BASE pour la détection d'intrusion avec alerte par SMS : Cas de la FECECAM-Bénin", Mémoire de Licence, Institut de Formation et de Recherche en Informatique, Université d'Abomey-Calavi, 2018-2019.

[2] S. P. THIRIMANNE, L. JAYAWARDANA, L. YASAKETHU, P. LIYANAARACHCHI et C. HEWAGE, "Deep Neural Network Based Real-Time Intrusion Detection System", *SN Computer Science*, t. 3, nº 145, 2022.

[3] E. C. EZIN et H. A. DJIHOUNTRY, "Java-Based Intrusion Detection System in a Wired Network", *IJCSIS*, t. 9, nº 11, 2011.

[6] N. BAUDOIN et M. KARLE, *NT Réseaux –IDS et IPS*, 2003-2004.

[9] D. BURGERMEISTER et J. KRIER, "Les systèmes de détection d'intrusions", rapp. tech., 2006.

[10] E. DETOISIEN, "Les attaques externes", *LinuxFocus*, nº 282, 2003.

[13] I. KADRIU, "Utilisation de Snort dans une PME", Mémoire de Licence, Haute École de Gestion de Genève, 2019.

[14] K. K. GUPTA, "Robust and efficient intrusion detection systems", Ph.D. dissertation, Department of Computer Science et Software Engineering, The University of Melbourne, 2009.

[15] P. ILLY, *Les systèmes de détection d'intrusion (IDS)*, avr. 2018. DOI : 10.13140/RG.2.2.10055.04001.

[16] F. MAJORCZYK, "Détection d'intrusions comportementale par diversification de COTS : application au cas des serveurs web", version non-définitive de la thèse, présentée lors de la soutenance., Theses, Université Rennes 1, déc. 2008. adresse : https://theses.hal.science/tel-00355366.

[17] A. V. D. ELEGBA, "Mise en place d'un système de détection d'intrusion et simulation d'attaque : cas de Bro Ids", Mémoire de Licence, Institut de Formation et de Recherche en Informatique, Université d'Abomey-Calavi, 2018-2019.

[18] A. HUČ, J. ŠALEJ et M. TREBAR, "Analysis of Machine Learning Algorithms for Anomaly Detection on Edge Devices", *Sensors (Basel)*, t. 21, nº 4946, p. 14, 2021.

[19] A. R. SONULE, M. KALLA, A. JAIN et D. S. CHOUHAN, "Unsw-Nb15 Dataset and Machine Learning Based Intrusion Detection Systems", *International Journal of Engineering and Advanced Technology (IJEAT)*, t. 9, 3 2020.

[21] T. TSIGKRITIS, G. GROUMAS et M. SCHNEIDER, "On the Use of k-NN in Anomaly Detection", *Journal of Information Security*, t. 9, p. 70-84, 2018.

[22] S. P. THIRIMANNE, L. JAYAWARDANA, L. YASAKETHU, P. LIYANAARACHCHI et C. HEWAGE, "Deep Neural Network Based Real-Time Intrusion Detection System", *SN Computer Science*, t. 3, nº 145, 2022.

[31] M. D.-D. C. HOUNKPEVI, "Mise en place d'un système de détection d'intrusion par analyse comportementale dans un réseau d'objets connectés", Mémoire de Licence, Institut de Formation et de Recherche en Informatique, Université d'Abomey-Calavi, 2021-2022.

[34] I. SHARAFALDIN, A. H. LASHKARI et A. A. GHORBANI, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4. I. C. on INFORMATION SYSTEMS SECURITY et P. (ICISSP), éd., jan. 2018.

Webographie

[4] *La sécurité informatique*, http://xenod.free.fr/0_La_securite_informatique.htm, consulté le 15/12/2022.

[5] *Mettre en oeuvre une politique de sécurité informatique (PSSI)*, https://gplexpert.com/securite-informatique/mise-oeuvre-politique-securite-informatique-pssi/, consulté le 15/12/2022.

[7] *Mise en place d'une politique de sécurité des systèmes informatiques*, https://www.promosoft.fr/mettre-en-place-politique-securite-systemes-informatiques/, consulté le 15/12/2022.

[8] *Attaques informatiques : Tout savoir sur les différentes menaces*, https://www.cyberuniversity.com/post/attaque-informatique-en-quoi-ca-consiste, consulté le 17/12/2022.

[11] E. FICHTNER, *Common Types of Cyber Attacks*, https://www.datto.com/fr/blog/common-types-of-cyber-security-attacks, consulté le 19/03/2023.

[12] E. FICHTNER, *Common Types of Cyber Attacks*, https://www.datto.com/fr/blog/common-types-of-cyber-security-attacks, consulté le 19/12/2022.

[20] M. AWAD et R. KHANNA, *Efficient Learning Machines : Theories, Concepts, and Applications for Engineers and System Designers*. Apress Berkeley, CA, 2015, p. XIX, 268.

[23] A. SINGH, *A comprehensive Guide to Ensemble Learning (with Python Codes)*, https://www.analyticsvidhya.co guide-for-ensemble-models/, consulté le 09/04/2023, 2018.

[24] *Endpoint Security website*, urlhttps://www.broadcom.com/products/cybersecurity/endpoint, consulté le 10/04/2023.

[25] *Zeek : An Open Source Network Security Monitoring Tool*, https://zeek.org/, consulté le 10/04/2023.

[26] J. SCHREIBER et R. LANGSTON, *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, consulté le 10/04/2023, 2020.

[27] *OSSEC*, urlhttps://www.ossec.net/, consulté le 10/04/2023.

[28] *Suricata website*, https://suricata.io/, consulté le 10/04/2023.

[29] *Prelude SIEM*, urlhttps://www.prelude-siem.com/, consulté le 10/04/2023.

[30] *Snort website*, urlhttps://snort.org/, consulté le 10/04/2023.

[32] *Mise en place de snort IDS*, http://jacquesgoueth.blogspot.com/2018/08/comment-mettre-en-place-un-systeme-de.html, consulté le 10/04/2023.

[33] *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)*, https://registry.opendata.aws/cse-cic-ids2018/, consulté le 10/10/2022.

[35] https://www.unb.ca/cic/research/applications.html#CICFlowMeter, consulté le 20/11/2022.

[36] *Python CICFlowMeter*, https://github.com/datthinh1801/cicflowmeter, consulté le 04/04/2023.

[37] *Projet Snort Alert Log Reader*, https://github.com/goooroooX/snort_reader/tree/master, consulté le 05/04/2023.

[38] *Communication APIs for SMS, Voice, Video & Authentication*, https://www.twilio.com/en-us, consulté le 29/12/2022.

[39] *Snort 3.1.18.0 on Ubuntu 18*, https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/012/147/original/Snort_3.1.8.0_on_Ubuntu_18_and_20.pdf, consulté le 15/01/2023.

[40] *Install and Configure Snort 3* NIDS *on Ubuntu 20.04*, https://kifarunix.com/install-and-configure-snort-3-nids-on-ubuntu-20-04/, consulté le 17/03/2023.

[41] *Projet PulledPork*, https://github.com/shirkdog/pulledpork3, consulté le 05/04/2023.

Table des matières

Dédicace **ii**

Remerciements **iii**

Résumé **iv**

iv

Abstract **v**

v

Table des figures **vi**

Liste des tableaux **vii**

Liste des acronymes **viii**

Introduction générale **1**

1 Etat de l'art **4**

Introduction . . . 4

1.1 La sécurité informatique . . . 4

1.1.1 Définition . . . 4

1.1.2 Objectifs de la sécurité informatique . . . 4

1.1.3 Mise en place d'une politique de sécurité informatique . . . 5

1.1.3.1 Différents aspects de la sécurité . . . 5

1.1.4 Etapes d'élaboration d'une politique de sécurité . . . 5

1.2 Les attaques informatiques . . . 6

1.2.1 Méthodologie générale d'une attaque . . . 6

1.2.2 Présentation de quelques attaques informatiques . . . 7

1.2.2.1 Les attaques réseaux . . . 7

1.2.2.2 Les attaques applicatives . . . 9

1.3 Systèmes de Détection d'Intrusion . . . 10

1.3.1 Définition générale des IDS . . . 10

1.3.2 Classification des alertes d'un IDS . . . 10

1.3.3 Les différents types d'IDS . . . 11

1.3.3.1 Host-based IDS . . . 11

1.3.3.2 Network-based IDS . . . 11

1.3.3.3 Les IDS Hybrides . . . 11

1.3.4 Approches de détection d'intrusions . . . 12

1.3.4.1 Approche par signature ou scénario . . . 12
1.3.4.2 Approche comportementale . . . 12
1.3.4.3 Approche hybride . . . 15
1.3.5 Présentation de quelques exemples d'IDS . . . 15
1.3.6 Etude comparative de quelques IDS . . . 17
Conclusion . . . 19

2 Présentation de l'architecture de l'approche hybride proposée 20
Introduction . . . 20
2.1 Choix du type d'IDS et Méthodologie . . . 20
2.1.1 Méthodologie . . . 21
2.1.1.1 Conception du module d'analyse comportementale . . . 21
2.1.1.2 Conception du parser et analyseur de fichiers journaux SNORT . . . 25
2.1.1.3 Conception du module d'alertes . . . 25
2.2 Environnement et matériels . . . 25
2.2.1 Machine de développement . . . 25
2.2.2 Environnement de tests . . . 26
Conclusion . . . 26

3 Mise en place de la solution et évaluation de performances 27
Introduction . . . 27
3.1 Configuration de l'environnement de test . . . 27
3.1.1 Mise en place de l'architecture réseau cible . . . 27
3.1.2 Mise en place du Système de Détection d'Intrusion . . . 27
3.1.2.1 Installation des dépendances . . . 27
3.1.2.2 Installation et configuration de SNORT3 . . . 28
3.1.2.3 Initialisation de l'IDS . . . 33
3.2 Tests de performances . . . 34
3.2.1 Présentation des résultats . . . 34
3.2.2 Analyse des résultats obtenus . . . 37
3.3 Discussion . . . 37
Conclusion . . . 37

Conclusion générale et perspectives 38

Bibliographie 40

Webographie 42

Table des matières 43