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**

01 BP 526 Cotonou-Bénin Tel : +229 55 028 070 / 53 973 060
https://www.ifri-uac.bj Courriel : contact@ifri.uac.bj

# MÉMOIRE

pour l'obtention du

**Diplôme de Licence en Informatique**

**Option :** Sécurité Informatique

**Présenté par** :
Morel Ebénézer **TALON**

# Etude et mise en place d'un outil open source pour la détection d'intrusion avec alerte par mail : Cas de l'APDP

**Sous la supervision** :
Prof Eugène C. **EZIN**
Enseignant-Chercheur à l'**UAC**

| | | | |
|---|---|---|---|
| Eugène C. EZIN | Enseignant-Chercheur | IFRI | Président |
| Hervé ACAKPO | Ingénieur | IFRI | Examinateur |
| Arnaud AHOUANDJINOU | Docteur | IFRI | Rapporteur |

Année Académique : **2020-2021**

# Sommaire


Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Table des figures vi

Liste des tableaux vii

Liste des acronymes viii

Introduction 1

1 Revue de littérature 3

2 Etude comparative des IDS Open Source et choix technique 15

3 Implémentation, résultats et discussions 21

Conclusion 39

Bibliographie 40

Webographie 41

Table des matières 43

# Dédicace

A

Mon père Abel **TALON**

Ma mère Gisèle **AMOUSSOUGA**

Mes frères.

# Remerciements

En préambule, permettez-nous d'exprimer nos profondes gratitudes et reconnaissances 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 :

- **Dieu**, pour nous avoir donné la santé, la force et la persévérance pour avoir accompli ce travail ;
- Professeur Eugène C. EZIN, Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI), pour nous avoir accepté dans son Institut d'une part et à toute son administration pour nous avoir permis de suivre cette excellente formation ;
- Docteur 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 ;
- Mr Toundé GREGOIRE, mon maitre de stage et ses collègues Mr Rémi HOUNKPATIN et Mr Hervé KPONON qui m'ont accordé beaucoup d'attention durant et après mon stage ;
- L'ensemble du corps des enseignants et du personnel administratif de l'IFRI pour la qualité des cours dispensés ;
- Nos parents,frères et proches qui nous ont soutenus physiquement, moralement et financièrement.

# Résumé

Dans ce document, nous avons présenté le déploiement d'un outil de détection d'intrusion open source au sein de l'Autorité de Protection des Données à caractère personnel (APDP). Pour réaliser le travail, nous avons tout d'abord procédé à une revue de littérature, une analyse des besoins et une étude comparative des différentes solutions de détection existant sur le marché afin de choisir la mieux adapté aux besoins de la structure. Nous avons porté notre choix sur SNORT tout en ressortant ses caractéristiques qui répondent aux attentes et réalités de l'entreprise. Nous avons enfin procédé au déploiement de la solution et effectué quelques tests de fonctionnalité. Lors de la phase de mise en place, nous avons configuré les services SSH et FTP. Nous avons ensuite installé SNORT sur la machine devant servir pour la détection. Nous avons par la suite écrit un code python qui nous a permis de surveiller en permanance le fichier d'alerte pour pouvoir l'envoyer à l'administrateur en cas de détection d'intrusion. Une discussion et une conclusion clôturent ce document.

**Mots clés** :Attaques, Pare-feux, Detection d'intrusion, Antivirus, SNORT.

# Abstract

This document, we presented the deployment of an open source intrusion detection tool within the Autorité de Protection des Données à caractère personnel (APDP). To carry out the work, we first carried out a literature review, a needs analysis and a comparative study of the various detection solutions existing on the market in order to choose the one best suited to the needs of the structure. We have chosen SNORT while highlighting its characteristics that meet the expectations and realities of the company. We finally proceeded to deploy the solution and performed some functionality tests. During the setup phase, we configured the SSH and FTP services. We then installed SNORT on the machine to be used for detection. We then wrote a python code which allowed us to permanently monitor the alert file in order to be able to send it to the administrator in the event of an intrusion detection. A discussion and a conclusion close this document.

**Key words**: Attack, Firewall, intrusion's detection, Anti-virus, SNORT .

# Table des figures


1.1 ways Handshake [3] . . . . . 7

3.1 Architecture de SNORT [23] . . . . . 22
3.2 Decodeur de paquets [5] . . . . . 22
3.3 Architecture déployée pour la simulation . . . . . 24
3.4 Installation de Snort . . . . . 26
3.5 Interface wlan0 en mode promiscuous . . . . . 26
3.6 Interface de déchargement . . . . . 27
3.7 GRO off . . . . . 27
3.8 Unité de service systemd . . . . . 27
3.9 Configuration du réseau à protéger . . . . . 28
3.10 Community-rules inclus . . . . . 29
3.11 Emplacement des bibliothèques OpenAppID . . . . . 29
3.12 Règles locales . . . . . 30
3.13 Inclusion des règles locales . . . . . 30
3.14 Configuration des sorties . . . . . 30
3.15 Configuration de Snort réussie . . . . . 31
3.16 Lancement de Scapy . . . . . 34
3.17 Création des paquets . . . . . 34
3.18 Enregistrement des alertes dans le fichier alert_fast.txt . . . . . 35
3.19 Lancemennt de l'attaque par brute force . . . . . 35
3.20 Enregistrement de nouveaux alertes au fichier d'alerte . . . . . 35
3.21 Réception du mail . . . . . 36
3.22 Contenu du mail . . . . . 37

# Liste des tableaux

1.1 Comparaison entre types d'IDS . . . . . 10

2.1 Etude comparative entre plusieurs outils IDS . . . . . 18

# Liste des acronymes

**APDP :**

Autorité de Protection des Données à caractère personnel iv, v, 1, 2, 39

**CGI :**

Common Gateway Interface 15

**CNIL :**

Commission Nationale de l'Informatique et des Libertés 2

**CPU :**

Central Processing Unit 12

**DHCP :**

Dynamic Host Configuration Protocol 8

**DMZ :**

Demilitarized zone 8, 9, 43

**DNS :**

Domain Name Server 5, 7, 8, 17

**DoS :**

Denial of Service 10, 13

**FTP :**

File Transfer Protocol iv, v, 30, 33, 34

**HIDS :**

Host Intrusion Detection Systems 9–11, 16, 18

**HIPS :**

Host Intrusion Prevention Systems 13, 44

**HTTP :**

Hypertext Transfer Protocol 8, 17, 23

**ICMP :**

Internet Control Message Protocol 6

**IDMEF :**

Intrusion Detection Message Exchange Format 16, 18

**IDS :**

Intrusion Detection Sytem vii, 1, 9–21, 23, 43, 44

**IP :**

Internet Protocol 6–9, 21–23, 34

**IPS :**

Intrusion Prevention Sytem 13, 14, 19, 44

**KIPS :**

Kernel Intrusion Prevention System 13, 44

**MAC :**

Media Access Control 14

**NIDS :**

Network Intrusion Detection Systems 10, 11, 17, 18, 21

**NIPS :**

Network Intrusion Prevention Systems 13, 44

**NSM :**

Network Security Monitoring 17

**OS :**

Operating System 13, 15

**PGP :**

Pretty Good Privacy 17

**PoE :**

Power Over Ethernet 24

**POSIX :**

Portable Operating System Interface 17

**RFC :**

Request for Comment 31

**SE :**

Système d'Exploitation 10

**SMB :**

Server Message Block 15, 23

**SMS :**

Short Message System 23

**SMTP :**

Simple Mail Transfer Protocol 31

**SNMP :**

Simple Network Management Protocol 5, 23

**SSH :**

Secure Shell iv, v, 30, 33

**SUID :**

Set User ID 17

**TCP :**

Transmission Control Protocol 6, 9, 17, 22, 23

**TLS :**

Transport Layer Security 17

**URI :**

Uniform Resource Identifier 23

**VPN :**

Virtuel Private Network 1

**XML :**

Extensible Markup Language 16, 23

# Introduction générale

## 1. Problématique

Dans la plupart des organisations ou entreprises comme remarqué à notre lieu de stage, pour protéger les actifs informatiques, on a recours au système de protection traditionnel, c'est à dire les firewall, les antivirus, Virtuel Private Network (VPN), etc. Cependant avec les techniques de piratage qui deviennent de plus en plus sophistiquées, ces technologies s'avèrent être limitées. Le système d'information étant au cœur même des activités de l'entreprise, doit être protégé au maximum pour garantir sa fiabilité. De ce fait, l'on doit pouvoir être au maximum informé de tout ce qui se passe d'anormal dans le réseau, être informé à temps réel, où que l'on soit des différentes attaques que subit le réseau afin de pouvoir mettre en œuvre les mesures adéquats pour les contrer. Devant ces exigences de sécurité et dans le cadre de ce mémoire, nous avons fixé comme objectif, d'étudier et de mettre en place un outil de détection d'intrusion pour l'APDP. Nous jugeons ceci comme étant une entité supplémentaire incontournable chargée de mieux garantir la protection et la sécurité du système informatique de l'entreprise.

## 2. Contexte et justification

Avec les avancées technologiques observées ces dernières années, les réseaux informatiques sont devenus indispensables à l'optimisation des performances d'une organisation. Ceci entraînant avec elles, plusieurs risques d'attaques. La sécurité des systèmes d'information n'apparait plus donc comme un paramètre de choix mais une nécessité. Puisque la survie de l'organisation en dépend. Une réalité qui n'épargne pas l'Autorité de Protection des Données à caractère personnel (APDP). Conscient de ce fait, l'on ne lésine plus sur les moyens pour protéger les infrastructures informatiques avec des antivirus et des firewalls. Toutefois, ces derniers ne suffisent plus ; car se basant sur des signatures ou empreintes d'attaques connues, ils deviennent obsolètes face à de nouvelles attaques (de types zero-day par exemple). D'où la nécessité de compléter à ces systèmes traditionnels des IDS pour détecter à temps réels ces attaques. Cependant pour qu'un IDS soit efficace, il faudrait non seulement qu'il détecte les attaques mais qu'il puisse mettre en place une intervention aisée et rapide 24h/24 pour pouvoir alerter à temps l'administrateur, mais également que sa capacité de détection puisse s'améliorer de manière autonome.

## 3. Objectifs

L'objectif général de notre travail est l'étude et la mise en place d'un outil de détection d'intrusion pour le système d'information de l'APDP. De façon spécifique, cela reviendra à :

- étudier quelques attaques des systèmes d'informations ;
- étudier l'intrusion dans un système d'information ;
- proposer une solution de détection de l'intrusion et implémenter cette solution ;
- simuler une intrusion dans le système d'information et analyser la réaction de l'outil.

## 4. Environnement de stage

Créée en 2009 - 2010, autrefois appelée Commission Nationale de l'Informatique et des Libertés (CNIL), l'APDP rebaptisée **Autorité de Protection des Données à caractère Personnel** en 2018, est une institution administrative indépendante chargée de protéger l'individu contre tout usage abusif des données le concernant. Elle a pour devise de « Protéger les données personnelles au Bénin ». Autrement dit, elle représente la "Police" des données personnelles. Cette institution est structurée en deux grandes divisions ; une première dont le rôle est de mener des sanctions contre les "criminels des données personnelles" ou contre les actions visant à détourner ces données conformément à l'article 461 du livre cinquième du code du numérique en République du Bénin, relatif aux peines. Une seconde dédiée à l'informatique. C'est ce secteur qui est en charge du traitement de ces données. L'APDP a pour missions :

- d'informer les personnes, des droits que leur reconnaît le code du numérique ;
- de protéger les données dans n'importe quel domaine d'activité ;
- de s'assurer du bon fonctionnement du traitement des fichiers et d'autoriser si nécessaire le traitement de données sensibles ;
- de contrôler les fichiers et vérifier si les responsables de traitement respectent le code du numérique ;
- de sanctionner les responsables qui n'observent pas le code du numérique ;
- d'anticiper les développements technologiques et de comprendre en quoi elles peuvent porter atteinte aux données personnelles.

Sa Direction Générale qui est la structure d'accueil de notre stage académique, est située dans le sixième arrondissement de Cotonou, quartier Aidjèdo dans la rue du centre d'accueil.

## 5. Organisation du mémoire

Le présent mémoire est organisé en trois chapitres. Le chapitre 1 est une revue de littéraire consacrée à l'état de l'art. Le chapitre 2 présente une étude des différentes solutions de détection existant, le choix de la solution implémentée ainsi que les raisons de ce choix. Quant au chapitre 3, il regroupe une description détaillée de la solution retenue, la mise en production de cette dernière, la présentation et l'analyse des résultats obtenus après les différents tests.

Chapitre 1

# Revue de littérature

## Introduction

Il s'agit dans ce chapitre de présenter une revue de littérature. Nous allons commencer par donner une vue d'ensemble de la sécurité informatique, par la suite nous plonger dans l'univers des systèmes de détection et de prévention d'intrusion.

## 1.1 La sécurité informatique

### 1.1.1 Définition

La notion de sécurité informatique couvre l'ensemble des moyens, outils, techniques et méthodes pour garantir que seules les personnes ou autres systèmes autorisés, interviennent sur le système et ont accès aux données, sensibles ou non [11].

Elle peut être definie comme l'ensemble des moyens mis en œuvre pour minimiser la vulnérabilité d'un système contre des menaces accidentelles [1].

### 1.1.2 Objectifs de la sécurité informatique

Le système d'information est généralement défini comme l'ensemble des données et des ressources matérielles et logicielles de l'entreprise permettant de les stocker ou de les faire circuler. Le système d'information représente un patrimoine essentiel de l'entreprise, qu'il convient de protéger. La sécurité informatique, d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'une organisation soient uniquement utilisées dans le cadre prévu [12].

La sécurité informatique vise généralement cinq principaux objectifs :

- l'intégrité, c'est-à-dire garantir que les données soient bien celles que l'on croit être ;
- la confidentialité, consistant à assurer que seules les personnes autorisées aient accès aux ressources échangées ;
- la disponibilité, permettant de maintenir le bon fonctionnement du système d'information ;
- la non répudiation, permettant de garantir qu'une transaction ne peut être niée ;

- l'authentification, consistant à assurer que seules les personnes autorisées aient accès aux ressources.

### 1.1.3 La mise en place d'une politique de sécurité

La sécurité des systèmes informatiques se cantonne généralement à garantir les droits d'accès aux données et ressources d'un système en mettant en place des mécanismes d'authentification et de contrôle permettant d'assurer que les utilisateurs desdites ressources possèdent uniquement les droits qui leur ont été octroyés.

Les mécanismes de sécurité mis en place peuvent néanmoins provoquer une gêne au niveau des utilisateurs et les consignes et règles deviennent de plus en plus compliquées au fur et à mesure que le réseau s'étend. Ainsi, la sécurité informatique doit être étudiée de telle manière à ne pas empêcher les utilisateurs de développer les usages qui leur sont nécessaires, et de faire en sorte qu'ils puissent utiliser le système d'information en toute confiance.

C'est la raison pour laquelle il est nécessaire de définir dans un premier temps une politique de sécurité, dont la mise en œuvre se fait selon les quatre étapes suivantes [12] :

- identifier les besoins en terme de sécurité, les risques informatiques pesant sur l'entreprise et leurs éventuelles conséquences ;
- élaborer des règles et des procédures à mettre en œuvre dans les différents services de l'organisation pour les risques identifiés ;
- surveiller et détecter les vulnérabilités du système d'information et se tenir informé des failles sur les applications et matériels utilisés ;
- définir les actions à entreprendre et les personnes à contacter en cas de détection d'une menace ;

La politique de sécurité est donc l'ensemble des orientations suivies par une organisation (à prendre au sens large) en terme de sécurité. A ce titre, elle se doit d'être élaborée au niveau de la direction de l'organisation concernée, car elle concerne tous les utilisateurs du système [13].

A cet égard, il ne revient pas aux seuls administrateurs informatiques de définir les droits d'accès des utilisateurs mais aux responsables hiérarchiques de ces derniers. Le rôle de l'administrateur informatique est donc de s'assurer que les ressources informatiques et les droits d'accès à celles-ci soient en cohérence avec la politique de sécurité définie par l'organisation [13].

De plus, étant donné qu'il est le seul à connaître parfaitement le système, il lui revient de faire remonter les informations concernant la sécurité à sa direction, éventuellement de conseiller les décideurs sur les stratégies à mettre en œuvre, ainsi que d'être le point d'entrée concernant la communication à destination des utilisateurs sur les problèmes et recommandations en terme de sécurité [13].

La sécurité informatique de l'entreprise repose sur une bonne connaissance des règles par les employés, grâce à des actions de formation et de sensibilisation auprès des utilisateurs, mais elle doit aller au-delà et notamment couvrir les champs suivants [13] :

- un dispositif de sécurité physique et logique, adapté aux besoins de l'entreprise et aux usages des utilisateurs ;
- une procédure de management des mises à jour ;

- une stratégie de sauvegarde correctement planifiée ;
- un plan de reprise après incident ;
- un système documenté à jour ;

### 1.1.4 Les attaques informatiques

Une attaque est le résultat de l'exploitation d'une faille d'un système informatique (système d'exploitation, logiciel, erreur de configuration, etc.) à des fins non connues par l'exploitant du système et il est généralement préjudiciable. De nos jours, en moyenne, un ordinateur connecté est attaqué au bout de 5 minutes [14].

#### 1.1.4.1 Anatomie d'une attaque

Fréquemment appelés « les 5 P » dans la littérature, ces cinq verbes anglophones constituent l'ossature de toute attaque informatique : Probe, Penetrate, Persist, Propagate, Paralyze. Observons le détail de chacune de ces étapes [2] :

**Probe :** consiste en la collecte d'informations par le biais d'outils comme whois[1], DNS lookup. La collecte d'informations sur le système cible peut s'effectuer de plusieurs manières, comme par exemple un scan de ports grâce au programme Nmap pour déterminer la version des logiciels utilisés, ou encore un scan de vulnérabilités à l'aide du programme Nessus. Pour les serveurs web, il existe un outil nommé Nikto qui permet de rechercher les failles connues ou les problèmes de sécurité. Des outils comme firewall, hping ou SNMP Walk permettent quant à eux de découvrir la nature d'un réseau.

**Penetrate :** utilisation des informations récoltées pour pénétrer un réseau. Des techniques comme la brute force ou les attaques par dictionnaire, peuvent être utilisées pour outrepasser les protections par mot de passe. Une autre alternative pour s'infiltrer dans un système est d'utiliser des failles applicatives.

**Persist :** une fois le réseau infiltré, le pirate cherchera à y revenir facilement. Pour cela, il installera par exemple des back doors[2]. Cependant, en général, il corrigera la faille par laquelle il s'est introduit afin de s'assurer qu'aucun autre pirate n'exploitera sa cible.

**Propagate :** cette étape consiste à observer ce qui est accessible et disponible sur le réseau local.

**Paralyze :** cette étape peut consister en plusieurs actions. 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.

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[3] à analyser. De plus, il est très difficile de supprimer entièrement des traces.

[1] est un outil permettant d'interroger des bases d'informations (appelées registres) concernant les noms de domaines et adresses IP

[2] Un backdoor (ou porte dérobée) est un programme informatique malveillant utilisé pour donner aux pirates un accès à distance non autorisé à un ordinateur infecté en exploitant les vulnérabilités du système.

[3] log désigne un type de fichier, ou une entité équivalente, dont la mission principale consiste à stocker un historique des événements.

#### 1.1.4.2 Quelques attaques informatiques

Parmi les attaques les plus connues, on peut citer les techniques de déni de service (DoS), l'usurpation d'adresse IP, l'usurpation du DNS Domain Name Server, l'ingénierie sociale, Man in the middle.

- **Le déni de service**

  D'une manière générale, on parle de déni de service quand une personne ou une organisation est privée d'un service utilisant des ressources qu'elle est en droit d'avoir normalement [15]. On trouvera par exemple des dénis de service touchant le service de courrier électronique, d'accès à Internet, de ressources partagées (pages Web), ou tout autre service à caractère commercial comme Yahoo ou EBay. Quoiqu'il en soit, le déni de service est un type d'attaque qui coûte généralement très cher puisqu'il interrompt le cours normal des transactions pour une entreprise ; les sommes et les enjeux sont énormes et cela ne peut aller qu'en s'aggravant tant que des parades réellement efficaces n'auront pas été trouvées. Il existe plusieurs moyens pour parvenir à un déni de service. Nous ne parlerons que de trois types dont l'attaque avec les paquets XMAS, l'attaque Smurf, et l'attaque par Syn flooding.

  - ***L'attaque avec paquets XMAS*** : Un paquet[4] XMAS ou Christmas Tree est un paquet dans lequel les drapeaux (flag) de tout protocole sont définis. Les drapeaux FIN, URG et PSH dans l'en-tête TCP sont activés. Ce paquet s'appelle paquet Christmas Tree, car tous les champs d'en-tête sont éclairés c'est à dire activés comme un arbre de noël. Ce type de paquet nécessite beaucoup plus de traitements que les paquets habituels, de sorte que le serveur alloue un grand nombre de ressources pour ce paquet. Par conséquent, cela peut être utilisé pour effectuer une attaque DoS sur le serveur [3].
  - ***L'attaque Smurf*** : L'attaquant ici envoie un grand nombre de paquets de diffusion d'écho ICMP, avec des adresses IP source falsifiées par rapport à l'adresse IP de la cible. Toutes les machines du réseau reçoivent ce message de diffusion et répondent à la cible avec un paquet de réponse d'écho [3].
  - ***Le syn flooding*** : Lors de l'initialisation d'une connexion TCP entre un client et un serveur, un échange de messages a lieu. Le principe est celui du three-way handshake[5], qui dans le cas d'une connexion normale sans volonté de nuire, se déroule en trois étapes [3] :
    * le client demande une connexion en envoyant un message SYN (pour synchronize) au serveur ;
    * le serveur accepte en envoyant un message SYN-ACK (synchronize-acknowledgment) vers le client ;
    * le client répond à son tour avec un message ACK (acknowledgment) ; la connexion est alors établie.

[4] Le paquet est l'unité de données qui est acheminée entre une origine et une destination sur un réseau.
[5] Mécanisme d'établissement de connexion TCP reposant sur 3 étapes

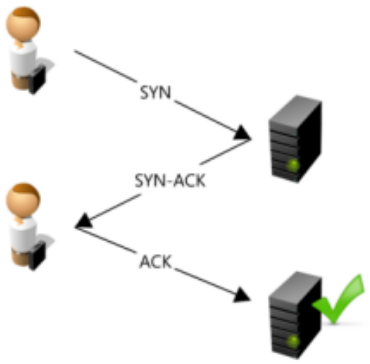



FIGURE 1.1 : ways Handshake [3]

L'attaquant peut créer un grand nombre de requêtes SYN forgées qui ont leurs adresses IP source falsifiées et l'envoyer à la cible. La cible répond avec SYN-ACK et alloue ses ressources pour la connexion, mais ne reçoit jamais la réponse ACK. Les ressources de la machine cible sont épuisées et elles ne permettent pas de répondre à d'autres demandes de machines légitimes.

- **L'usurpation d'adresse IP** L'« usurpation d'adresse IP » (également appelée mystification ou en anglais IP Spoofing) est une technique consistant à remplacer l'adresse IP de l'expéditeur d'un paquet IP par l'adresse IP d'une autre machine. Cette technique permet ainsi à un pirate d'envoyer des paquets de façon anonyme. Il ne s'agit pas pour autant d'un changement d'adresse IP, mais d'une mascarade de l'adresse IP au niveau des paquets émis [9].

- **L'usurpation DNS**

  Il s'agit de corrompre le cache[6] du serveur DNS Domain Name Server afin de faire croire à une machine qu'un nom de machine est relatif à une fausse adresse donnée. Ainsi donc les communications vers la machine ciblée sont redirigées vers celle dont l'adresse est portée en avant.

- **Ingénierie sociale**

  L'ingénierie sociale est une attaque qui s'appuie essentiellement sur les relations humaines pour inciter de façon détournée à enfreindre les procédures de sécurité. L'ingénierie sociale fait partie des techniques les plus simples, mais aussi les plus efficaces pour entrer dans un système sécurisé à grande échelle. En effet, le système aura beau être sécurisé avec les moyens les plus sophistiqués, une seule information divulguée d'un utilisateur aura suffi à mettre le système à terre. Il s'agit d'une arme très forte résultant de l'étude de l'être humain, de ses passions, de ses désirs. Cette forme d'attaque est très dévastatrice en fonction de l'importance des informations tirées [10].

- **Man in the middle**

  Moins connue, mais tout aussi efficace, cette attaque permet de détourner le trafic entre deux stations. Imaginons un client C communiquant avec un serveur S. Un pirate peut détourner le trafic du client en faisant passer les requêtes de C vers S par sa machine P, puis transmettre les

[6] une mémoire ou partie de mémoire dans laquelle sont stockés de façon temporaire les données ou les programmes les plus fréquemment ou les plus récemment utilisés, que l'ordinateur peut interroger afin de réduire les temps de réponse.

requêtes de P vers S. Et inversement pour les réponses de S vers C. Totalement transparente pour le client, la machine P joue le rôle de proxy. Il accédera ainsi à toutes les communications et pourra en obtenir les informations sans que l'utilisateur s'en rende compte [1].

Face à ces différentes menaces, différentes solutions sont nées. Nous allons en présenter quelques-unes.

### 1.1.5 Les dispositifs de protection

La variété et la disponibilité des outils d'attaque augmentent le risque des intrusions. Par conséquent, les administrateurs s'appuient sur diverses solutions dans le but de maintenir la protection du réseau informatique.

#### 1.1.5.1 Un antivirus

Logiciel censé protéger un ordinateur contre les logiciels (ou fichiers potentiellement exécutables) néfastes. Ne protège pas contre un intrus qui emploie un logiciel légitime, ou contre un utilisateur légitime qui accède à une ressource alors qu'il n'est pas autorisé à le faire [4].

#### 1.1.5.2 Un pare-feu

Un pare-feu est un logiciel et/ou matériel qui filtre et protège un système en bloquant les connexions venant de l'extérieur (entrées) ou de l'intérieur (sorties) pour empêcher ou autoriser l'accès à des services Web. Il permet aussi de faire de la translation d'adresse pour servir de routeur [7].

#### 1.1.5.3 Serveur proxy :

Un proxy (de l'anglais procuration) est un composant matériel informatique qui joue le rôle d'intermédiaire dans l'échange entre deux hôtes. L'objectif de base d'un serveur proxy est de prendre les requêtes du client à la place d'un serveur et de les transférer avec sa propre adresse IP à l'hôte cible. Pour cela, il n'existe pas de connexion directe entre l'émetteur et le destinataire. Parfois, le système qui reçoit la requête et l'hôte cible ne savent pas qu'ils ont affaire à un proxy. Ces serveurs vont dans deux directions. Le proxyhttp sert de protection d'un réseau de client face aux activités provenant du Web. Si le système cible (un serveur Web par exemple) est protégé par un serveur proxy, on parle alors de proxy inverse (reverse en anglais) [16].

#### 1.1.5.4 La DMZ :

Une DMZ (Demilitarized zone) est une zone tampon d'un réseau d'entreprise, située entre le réseau local et Internet, derrière le pare-feu. Il s'agit d'un réseau intermédiaire regroupant des serveurs publics (HTTP, DHCP, mails, DNS, etc.). Ces serveurs devront être accessibles depuis le réseau interne de l'entreprise et, pour certains, depuis les réseaux externes. Le but est ainsi d'éviter toute connexion directe au réseau interne.

L'informatique évolue, les applications sont de plus en plus complexes et les délais laissés aux programmeurs et administrateurs sont souvent très courts. Les risques de failles applicatives sont, de ce fait, très grands et peuvent s'avérer dangereux pour des applications largement répandues. Ce qui nous amène à notre seconde partie : comment détecter et empêcher ces attaques ?

## 1.2 Système de Détection d'intrusion

### 1.2.1 Définition générale des IDS

Un système de détection d'intrusion IDS est un outil de sécurité aussi bien matériel que logiciel permettant de surveiller l'activité d'un réseau ou d'un hôte donné, afin de détecter toute tentative d'intrusion [5]. Les systèmes de détection d'intrusions surveillent et analysent les activités d'un réseau, analysent ses configurations et ses vulnérabilités, et vérifient l'intégrité des fichiers. Ils peuvent reconnaître des schémas d'attaque classiques. Ils passent au peigne fin le traffic (entrant et sortant) à la recherche de signatures numériques suspectes et génèrent une alarme quand une correspondance est trouvée [1].

### 1.2.2 Les fonctions principales d'un IDS

Les principales fonctionnalités d'un IDS peuvent se résumer comme suit [1] :

- recueillir et analyser les informations à l'intérieur d'un ordinateur ou d'un réseau, afin de détecter des potentielles violations de la politique de sécurité, des accès non autorisés, aussi bien que des abus (pour des accès autorisés) ;
- se comporter en sniffeur[7] de paquets pour intercepter les paquets qui circulent à travers les médias et protocoles de communication, généralement le TCP/IP ;
- les paquets sont analysés une fois qu'ils ont été capturés ;
- un IDS scrute le trafic pour y déceler de possibles intrusions et produire une alarme après détection.

### 1.2.3 Emplacement de l'IDS dans le réseau

L'une des places les plus ordinaires pour déployer un IDS est juste à côté du firewall. En fonction du traffic, pour être un contrôleur, l'IDS est placé coté extérieur/intérieur du firewall. Placé côté intérieur, il est idéal de positionner l'IDS proche de la DMZ. Cependant, la meilleure pratique consiste à utiliser une défense en couche[8] en déployant un IDS devant le firewall et un autre derrière le firewall dans le réseau [1].

Avant de déployer l'IDS, il est essentiel d'analyser la topologie du réseau, comprendre le flux du trafic et identifier les composants critiques qui seront une cible potentielle pour de nombreuses attaques contre le réseau. Après avoir décidé de la position de l'IDS, il faut procéder à sa configuration minutieuse pour maximiser l'efficacité de la protection du réseau.

### 1.2.4 Les différents types d'IDS

Les différents IDS se caractérisent par leur domaine de surveillance. Il existe trois grandes familles distinctes d'IDS :

1. **Système de détection basé sur l'hôte (HIDS)**

[7] un programme qui capture tous les paquets circulant dans le réseau et vous permet d'en modifier le contenu.
[8] La défense en couche est une doctrine de cybersécurité dans laquelle une série de mécanismes défensifs sont superposés afin de protéger des données et des informations précieuses.

L'IDS Système ou Host Based IDS (HIDS) surveille le trafic sur une seule machine. Il analyse les journaux systèmes, les appels systèmes et enfin vérifie l'intégrité des systèmes de fichiers. Les HIDS sont de part leur principe de fonctionnement, dépendant du système sur lequel ils sont installés. Ce système peut s'appuyer ou non sur le système propre au système d'exploitation pour en vérifier l'intégrité et générer des alertes. Il peut aussi capturer les paquets réseaux entrant/sortant de l'hôte pour y déceler des signaux d'intrusions (Déni de Services, Backdoors, chevaux de Troie, tentatives d'accès non autorisés, exécution de codes malicieux, etc.) [17].

2. **Système de détetion sur le réseau (NIDS)**

Les systèmes de détection d'intrusion au niveau du réseau sont chargés d'analyser de manière passive le trafic réseau et d'y rechercher les activités malveillantes. Ils comportent généralement une sonde[9] (machine par exemple) qui écoute et surveille en temps réel tout le trafic réseau, puis analyse et génère des alertes s'il détecte des intrusions ou des paquets qui semblent dangereux.

3. **Système de détection d'intrusion Hybride**

Les IDS hybrides rassemblent les caractéristiques des NIDS et HIDS. Ils permettent de surveiller le réseau et les terminaux. Les sondes sont placées en des points stratégiques et agissent comme NIDS et/ou HIDS suivant leurs emplacements. Toutes ces sondes remontent alors les alertes à une machine qui va centraliser le tout, et lier les informations d'origines multiples. Ainsi, on comprend que les IDS hybrides sont basés sur une architecture distribuée[10], où chaque composant unifie son format d'envoi. Cela permet de communiquer et d'extraire des alertes plus exactes [1]

### 1.2.5 Comparaison entre les types d'IDS

TABLE 1.1 : Comparaison entre types d'IDS

| Types d'IDS | Avantages | Inconvenients |
|---|---|---|
| HIDS | • Détecter des attaques impossibles à détecter avec des IDS réseau puisque le trafic est souvent crypté<br>• Marcher dans un environnement dans lequel le trafic de réseau est encrypté.<br>• Observer les activités sur l'hôte avec précision<br>• Lorsque HIDS marche sur la traînée de l'audit des SE, ils peuvent détecter les Chevaux de Troie ou les autres attaques concernant la brèche d'intégrité de logiciel. | • HIDS est difficile à gérer, et des informations doivent être configurées et gérées pour chaque host surveillé.<br>• Ils ont moins de facilité à détecter les scans<br>• HIDS peuvent être neutralisés par certaines attaques de DoS.<br>• Ils consomment beaucoup de ressources CPU. |

[9] un équipement qui permet de gérer la qualité des flux réseau.

[10] un système d'information ou un réseau pour lequel l'ensemble des ressources disponibles ne se trouvent pas au même endroit ou sur une même machine.

| | | |
|---|---|---|
| NIDS | • Les capteurs peuvent être bien sécurisés puisqu'ils se contentent d'observer le trafic.<br>• Détecter plus facilement les scans grâce aux signatures.<br>• Filtrage de trafic.<br>• Assurer la sécurité contre les attaques puisqu'il est invisible. | • Il est difficile de contrôler le réseau entier.<br>• Ils doivent principalement fonctionner de manière cryptée d'où une complication de l'analyse des paquets.<br>• A l'opposé des IDS basés sur l'hôte, ils ne voient pas les impacts d'une attaque. |
| Hybride | • Bénéficiant du support des HIDS, ils sont insensibles aux problèmes rencontrés par les NIDS.<br>• Meilleure corrélation (la corrélation permet de générer de nouvelles alertes à partir de celles existantes)..<br>• Il est plus difficile de contourner les protections car ils disposent de plusieurs mécanismes de protection. | • Trop d'alertes révélés fausses après vérification |

### 1.2.6 Méthodes de détection des intrusions par les IDS

Les IDS utilisent deux méthodes pour détecter les intrusions dans le réseau.

#### 1.2.6.1 Approche par signature ou scénario

Dans cette approche, les détecteurs d'intrusion reposent sur la création d'une base de motifs qui représente des scénarios d'attaque connus au préalable et qui sera utilisée par la suite, le plus souvent en temps réel, sur les informations fournies par les sondes de détection. C'est donc un système de reconnaissance de motifs qui permet de mettre en évidence dans ces informations, la présence d'une intrusion connue par la base de signature. Les IDS à signature utilisent trois familles de méthodes qui se basent toutes sur la recherche d'un profil connu d'attaque [1] :

- Reconnaissance de forme (pattern matching) : Cette méthode consiste à identifier une suite d'événements ou des marques d'une attaque connue dans les paquets analysés. En fait, le trafic réseau peut être vu comme une chaîne de caractères principales et les scénarios d'attaque comme des sous-suites qu'on veut identifier.

- Système expert : L'idée consiste à coder les attaques sous forme de règles condition-action où les conditions portent sur l'état du système surveillé ainsi que la nature des événements analysés, tandis que les actions permettent, soit de mémoriser le nouvel état du système, soit de conclure la présence d'une attaque.

- Algorithme génétique : Les algorithmes génétiques utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles à un problème difficile, pour trouver une solution approchée dans un temps raisonnable.

#### 1.2.6.2 L'approche comportementale (anomaly detection)

Cette technique consiste à détecter une intrusion en fonction du comportement passé de l'utilisateur. Il faut préalablement dresser un profil utilisateur à partir de ses habitudes et déclencher une alerte lorsque des évènements hors profil se produisent. Cette technique peut etre appliquée non seulement à des utilisateurs mais aussi à des applications et services. Plusieurs métriques (paramètres) sont possibles : la charge CPU[11], le volume de données échangées, le temps de connexion sur des ressources, la répartition statistique des protocoles et applications utilisés, les heures de connexion, etc [6].

- Approche probabiliste : Des probabilités sont établies permettant de représenter une utilisation courante d'une application ou d'un protocole. Toute activité qui ne respecte pas le modèle probabiliste provoquera la génération d'une alerte.
- Approche statistique : Cette méthode consiste en un calcul de la variation (en fonction du temps) de l'utilisation d'un certain nombre de ressources. La constatation d'un changement dans la variation conduit au déclenchement d'une alerte.
- Les autres méthodes :
  - l'immunologie qui fait l'analogie avec le système immunitaire biologique basé sur le principe de reconnaissance de cellules étrangères ;
  - les modèles à base de graphes ainsi que
  - les systèmes experts
  - et le data mining[12].

### 1.2.7 Les différents types d'alertes d'un IDS

L'IDS détecte et produit des alertes en raison d'un certain nombre de facteurs qui sont classifiés dans une des limites suivantes :

- vrai positif : Une situation dans laquelle une signature met le feu correctement quand le trafic intrusif est détecté sur le réseau ; ceci représente l'opération normale et optimale.
- faux positif : Une situation dans laquelle l'utilisation d'une activité normale déclenche une alerte ou une réponse ; ceci représente une erreur.
- vrai négatif : Une situation dans laquelle, une signature ne met pas le feu pendant l'utilisation normal de trafic sur le réseau. Aucune activité malveillante. Ceci représente une opération normale et optimale.
- faux négatif : Une situation dans laquelle, le système de détection ne détecte pas le trafic intrusif bien qu'il y ait une activité malveillante, mais le système de sécurité ne réagit pas. Ce cas représente une erreur.

[11] l'intensité avec laquelle le processeur traite les programmes et les processus en cours d'exécution.
[12] désigne le processus d'analyse de volumes massifs de données et du Big Data sous différents angles afin d'identifier des relations entre les data et de les transformer en informations exploitables.

## 1.3 Système de Prévention d'intrusion

### 1.3.1 Définition

L'IPS est un outil de protection et sécurité des systèmes d'information contre les intrusions, similaire aux IDS, permettant de prendre des mesures afin de diminuer les impacts d'une attaque. C'est un IDS actif, il empêche toute activité suspecte détectée au sein d'un système [7].

### 1.3.2 Les types d'IPS

#### 1.3.2.1 La prévention d'intrusion basée sur l'hôte HIPS

Les HIPS installés sur le système permettant de surveiller le poste de travail à travers différentes techniques, surveillent les processus, les drivers[13], etc. En cas de détection de processus suspect, le HIPS peut bloquer les comportements anormaux tels que [7] :

- lecture / écriture des fichiers protégés ;
- comportement de certains applicatifs ;
- accès à des ports non autorisés ;
- tentative d'exploitation de débordement de pile (détection de Shell code) ;
- accès à certaines zones de la base de registre ;
- connexions suspectes.

#### 1.3.2.2 La prévention d'intrusion basée sur le réseau(NIPS)

Les NIPS permettant de surveiller le trafic réseau, l'identification et le blocage du trafic malicieux, sont parfois utilisés pour évoquer la protection des réseaux sans-fil [7]. Nous disposons de deux types de NIPS :

- un système par analyse comportementale (Content Based IPS qui se base sur la détection des anomalies protocolaires, la détection des comportements anormaux (scan de ports, DoS, etc.), etc.
- un système par détection des anomalies

#### 1.3.2.3 La détection d'intrusion basée sur noyau KIPS

Les KIPS (Kernel Intrusion Prevention System) ont pour particularité de s'exécuter dans le noyau d'une machine, pour y bloquer toute activité suspecte. Le KIPS permet de détecter toute tentative d'intrusion et la bloque directement au niveau du noyau, empêchant ainsi toute modification dangereuse pour le système. Il peut reconnaître des motifs caractéristiques du débordement de mémoire et peut ainsi interdire l'exécution du code. Il peut également interdire l'OS d'exécuter un appel système qui ouvrirait un shell de commande. Puisqu'un KIPS analyse les appels systèmes, il ralentit l'exécution. C'est pourquoi ils sont moins utilisés [7].

[13] Permettent la bonne liaison entre votre ordinateur et un périphérique

### 1.3.3 Architecture fonctionnelle d'un IPS

Le fonctionnement d'un IPS est similaire à celui d'un IDS. Il capture le trafic du réseau puis l'analyse. Mais au lieu d'alerter l'utilisateur d'une intrusion ou d'une attaque, l'IPS bloque directement les intrusions en supprimant les paquets illégitimes. Pour informer l'utilisateur, l'IPS peut aussi remplir un fichier de journalisation qui contiendra la liste des paquets supprimés et éventuellement un message indiquant la raison de cette suppression [7].

### 1.3.4 La Différence entre IPS et Firewall

Firewall est un système de contrôle d'accès basé sur des règles statiques autorisant ou refusant certains flux. Il travaille essentiellement au niveau des couches du modèle OSI (de 1 à 4) ; ce qui est insuffisant pour les intrusions [7].

Contrairement à un IPS qui doit être complètement discret. Ceci implique que les interfaces de la sonde ne doivent pas être visibles (pas d'adresse IPS, pas d'adresse MAC), et l'IPS analyse l'intégralité des paquets en transit, depuis les couches réseaux jusqu'au niveau applicatif.

## Conclusion

Dans ce chapitre, nous avons parlé de quelques notions sur la sécurité informatique et parcouru les systèmes de détection et de prévention d'intrusions, leurs architectures, ainsi que leurs fonctionnements. Dans le chapitre 2, nous essayerons de faire un tour d'horizon des IDS disponibles sur le marché afin de choisir celui répondant à nos besoins.

Chapitre 2

# Etude comparative des IDS Open Source et choix technique

## Introduction

Comme vu dans le chapitre précédent, les IDS sont importants dans la démarche de sécurisation des systèmes d'information. Dans ce chapitre, nous vous proposons de découvrir les offres qui existent en matière d'IDS et de faire un étude comparative de ces dernières.

## 2.1 Etude de l'existant

Afin de faire face aux problèmes d'intrusion, les solutions sont moult et diversifiées en termes d'IDS. Il existe tant bien des solutions commerciales que des solutions Open Source[1]. Toutefois les solutions Open Source n'ont rien à envier aux solutions commerciales. Mieux encore, les solutions commerciales se basent même sur les Open Source pour améliorer leur produit. La différence notoire entre ces deux solutions se trouve principalement dans le déploiement, notamment le prix. Cela nécessite beaucoup de prés-requis telles que des utilitaires de base ou encore des connaissances sur le système sur lequel le produit va être déployé. Ainsi en guise d'exemple, comme solution d'IDS open source nous pouvons citer entres autres SNORT, Prelude, OSSEC, Bro, Samhain, Suricata.

### 2.1.1 SNORT

Snort est un système de détection d'intrusion réseau open source capable d'effectuer l'analyse du trafic en temps réel. On l'utilise en général pour détecter une variété d'attaque et de scans tels que des débordements de tampons[2], des scans de ports furtifs, des attaques CGI, des scans SMB, des tentatives d'identification d'OS grâce à l'analyse des signatures et des réponses caractéristiques (fingerprinting) [18]. Il est capable d'analyser le trafic sur le réseau en temps réel et les paquets circulant sur le réseau. Il peut exécuter l'analyse de protocole et peut être employé pour détecter une variété

[1] est un code conçu pour être accessible au public : n'importe qui peut voir, modifier et distribuer le code à sa convenance.
[2] est une anomalie qui se produit lorsque le logiciel qui écrit des données dans une mémoire tampon dépasse la capacité de cette dernière, ce qui entraîne l'écrasement des emplacements mémoire adjacents.

d'attaque, des tentatives comme des débordements, des balayages de port dérobé. Il peut fonctionner selon trois modes principaux : il peut être utilisé comme un simple sniffer de paquets, comme un logger de trafic réseaux (ce qui est fort utile pour déboguer un réseau par exemple puisque ce mode d'enregistrement des activités répertorie toutes les interactions entre les machines d'un même réseau), ou comme un véritable système complet de prévention contre les attaques et les intrusions de tout genre [5]. C'est un très puissant outil, il est connu comme un des meilleurs IDS sur le marché, même quand il est comparé à des IDS commerciaux. De nombreuses personnes dans cette communauté très active, partagent leurs règles de sécurité ; ce qui est très utile si on n'est pas un expert de la sécurité et si on veut des règles à jour.

### 2.1.2 Prelude

Prelude a été conçu dans le but d'être modulaire, souple, et résistant aux attaques. Sa modularité permet notamment de lui rajouter facilement de nouveaux types de détecteurs d'intrusion, d'analyseurs de logs et d'une solution de corrélation, le tout au format et à la norme IDMEF, bien que de nombreux autres formats de logs sont compatibles [19]. Le format IDMEF décrit une alerte[3] de façon objetive et exhaustive. Le but d'IDMEF est de proposer un standard permettant d'avoir une communication hétérogène quel que soit l'environnement ou les capacités d'un analyseur donné. Ces alertes sont définies au format XML, offrant une possibilité de validation de chaque message. En général, les implémentations restent binaires, afin d'éviter les problèmes connus d'ajout d'informations inutiles en dehors d'XML lorsque l'on envoie un message sur le réseau. IDMEF offre aussi un vocabulaire précis, qu'il est courant d'utiliser dans le domaine de la détection d'intrusion.

L'application Prelude est disponible uniquement sous Linux. Bien qu'il ait une offre payante (support, fonctionnalités supplémentaires, etc), le logiciel est gratuit.

### 2.1.3 OSSEC

OSSEC est une application de détection d'intrusion, et plus précisément un HIDS. Il permet de surveiller l'intégrité des fichiers systèmes, aussi bien sur des postes Linux que Windows. De plus, OSSEC détecte également des attaques de pirates comme les rootkits[4], les scans de ports et analyse les logs du système, des applications et des services. Le logiciel propose également un système de réponses actives, c'est-à-dire d'actions à réaliser en cas d'attaque. Tout comme Snort, il dispose de nombreuses règles lui offrant un large panel de détection d'attaques [8]. Lorsqu'il est installé sur des systèmes d'exploitation de type Unix, le moteur de détection du logiciel se concentre principalement sur les fichiers journaux et de configuration. Il crée des totaux de contrôle des fichiers importants et les vérifie périodiquement, en vous alertant ou en déclenchant une action corrective chaque fois que quelque chose d'étrange se produit. Sous Windows, le système garde également un œil sur les modifications de registre non autorisées car elles pourraient être le signe révélateur d'une activité malveillante. Toute détection déclenchera une alerte.

[3] Une alerte est le message qui est émis depuis un analyseur, qui est une sonde en langage IDMEF, vers un collecteur.

[4] est un progiciel conçu pour rester caché sur votre ordinateur tout en permettant l'accès et le contrôle à distance de ce dernier.

### 2.1.4 Bro ou Zeek

Bro est un NIDS gratuit. Il va au-delà de la détection d'intrusion. Il peut également fournir d'autres fonctions de surveillance du réseau. L'analyse de Bro s'opère au niveau de la couche application[5]. Cela donne une visibilité sur les paquets pour obtenir une analyse plus large de l'activité du protocole réseau. Le module d'analyse de Bro comprend deux éléments qui fonctionnent tous deux sur l'analyse de signature et la détection d'anomalies[20]. Le premier de ces outils d'analyse est le Bro event engine. Cela permet de déclencher des événements, tels qu'une nouvelle connexion TCP ou une requête HTTP. Chaque événement est enregistré, donc cette partie du système est neutre sur le plan des politiques. Il fournit simplement une liste d'événements dans lesquels l'analyse peut révéler la répétition d'actions ou une activité étrangement diversifiée générée par le même compte d'utilisateur. Une condition d'alerte provoquera une action d'autodéfense.

### 2.1.5 Samhain

Samhain, produit par Samhain Design Labs en Allemagne, est un système de détection d'intrusion basé sur l'hôte qui est gratuit à utiliser. Il peut être exécuté sur un seul ordinateur ou sur de nombreux hôtes, offrant une collecte centralisée de données sur les événements détectés par les agents exécutés sur chaque machine. Les tâches effectuées par chaque agent incluent la vérification de l'intégrité des fichiers, la surveillance des fichiers journaux et la surveillance des ports. Les processus recherchent les virus rootkit, les SUID (droits d'accès des utilisateurs) et les processus cachés [20]. Le système applique un chiffrement aux communications entre les agents et un contrôleur central dans les implémentations multi-hôtes. Les connexions pour la livraison des données du fichier journal incluent des exigences d'authentification, qui empêchent les intrus de détourner ou de remplacer le processus de surveillance. Les données recueillies par Samhain permettent d'analyser les activités sur le réseau et mettront en évidence des signes d'alerte d'intrusion mais ne bloquera pas l'intrusion, ni n'éliminera les processus frauduleux. Les fichiers journaux centraux et les sauvegardes de configuration sont signés avec une clé PGP pour empêcher la falsification par les intrus. Il a été conçu selon les directives POSIX pour le rendre compatible avec Unix, Linux et Mac OS.

### 2.1.6 Suricata

Suricata est un logiciel open source de détection, de prévention et de supervision de sécurité réseau (NSM) [1]. Suricata permet l'inspection des paquets en profondeur. NIDS qui fonctionne au niveau de la couche application, lui donnant une visibilité multi-paquets. C'est un outil gratuit qui a des capacités très similaires à celles de Bro. Bien que ces systèmes de détection basés sur les signatures travaillées au niveau de l'application, ils ont toujours accès aux détails du paquet, ce qui permet au programme de traitement d'obtenir des informations au niveau du protocole des en-têtes de paquets. Cet IDS emploie également les méthodes de détection basées sur les anomalies. Outre les données de paquets, Suricata est en mesure d'examiner les certificats TLS, les requêtes HTTP et les transactions DNS. L'outil est également capable d'extraire des segments de fichiers au niveau du bit pour la détection de virus. Suricata est l'un des nombreux outils compatibles avec la structure de données Snort [20]. Il est capable d'implémenter des politiques de base Snort.

[5]Couche 7 du modèle OSI qui fournit à un programme d'application des services destinés à lui assurer une communication efficace avec ses homologues.

## 2.2 Etude comparative

Le marché propose plusieurs outils de détection d'intrusion qui ont tous leurs qualités et défauts. Une étude comparative s'annonce être un grand atout pour pouvoir sélectionner la meilleure solution possible pour notre zone de travail. Le tableau 2.1 fait état de cela :

TABLE 2.1 : Etude comparative entre plusieurs outils IDS

| Outils | Avantages | Inconvénients |
|---|---|---|
| Snort | • Une très grande communauté<br>• Mise à jour régulière de la base de signatures.<br>• Nombreux plugins et une documentation riche.<br>• Peut fonctionner sur plusieurs modes<br>• Logiciel libre | • A plus tendance que d'autres IDS à fournir des fausses alertes à cause de certaines signatures qui peuvent être mal jugées par l'outil.<br>• Ne peut pas traiter les flux cryptés.<br>• Nécessite une étude avant le placement des sondes.<br>• Les sondes nids peuvent être vulnérables. |
| Prelude | • Compatible avec plusieurs autres outils.<br>• Il intègre les fonctionnalités NIDS et HIDS.<br>• Scanneur de vulnérabilités.<br>• Centralise les alertes dans sa base de données et les normalise au format IDMEF, puis visualisable dans une interface web.<br>• Logiciel libre. | • La base de données des signatures est moins riche que celle de snort.<br>• Besoin de connaissances en sécurité. |
| OSSEC | • Installation facile<br>• Peu gourmand en ressources et en compétences<br>• Visibilité des activités systèmes (kernel, daemons internes, ...)<br>• N'importe quel logiciel ou système peut être intégré tant qu'il y a des logs<br>• Logiciel libre. | • Peut générer beaucoup de mails -> exceptions sur les règles<br>• Agrégation de mails pas toujours respectée |

| | | |
|---|---|---|
| Bro(Zeek) | • Faux positifs réduits.<br>• Extensible.<br>• Analyse graphique avancée.<br>• Compatible avec d'autres outils.<br>• Compatible avec des règles de snort grâce à snort2 bro. | • Documentation faible.<br>• Absence d'une interface graphique.<br>• Manque d'informations dans les rapports d'alertes. |
| Samhain | • Il peut être exécuté sur un seul ordinateur ou sur de nombreux hôtes.<br>• Les connexions pour la livraison des données du fichier journal incluent des exigences d'authentification, qui empêchent les intrus de détourner ou de remplacer le processus de surveillance. | • Pas de moyens pour empêcher l'intrusion.<br>• Documentation faible. |
| Suricata | • Une grande communauté<br>• Compatible avec d'autres outils<br>• Intègre aussi les fonctionnalités des IPS<br>• Logiciel libre | • Compliqué en opération et nécessite plus de ressources systèmes pour un fonctionnement à part entière.<br>• Moins de supports par rapport aux autres IDS comme Snort. |

## 2.3 Fonctionnalités et résultats attendus de la solution

### 2.3.1 Besoins fonctionnels

L'analyse du sujet nous a permis de dégager les fonctionnalités que doit fournir la solution à retenir. Après mise en place de la solution retenue, cette dernière devra :

- pouvoir détecter toute sorte d'intrusion (interne ou externe) dans le réseau ;
- répertorier toutes les attaques dans un fichier facile d'accès à l'administrateur du réseau ;
- gérer les alertes en temps réel et configurables ;
- pouvoir alerter par mail l'administrateur en cas d'intrusion.

### 2.3.2 Besoins non fonctionnels

Quand les besoins fonctionnels expriment les fonctionnalités concrètes du produit, les besoins non fonctionnels sont des indicateurs de qualité de l'exécution des besoins fonctionnels. Les besoins recensés à ce niveau sont :

- la solution doit être facile à mettre en place : installation et configuration ;

- une solution disponible sur tous les systèmes d'exploitation et sécurisée ;
- les mises à jour doivent être facile à réaliser ;
- un temps de réponse rapide.

### 2.3.3 Résultats attendus de la solution

Après étude et mise en place de l'outil, les changements souhaités au sein de l'entreprise sont les suivants :

- proactivité et réaction en cas d'attaque ;
- un système d'alerte permettant à l'administrateur d'être informé partout où il est en cas d'intrusion ;
- une grande précision sur la nature et l'origine des attaques ;
- un système d'information plus fiable et moins vulnérable.

## 2.4 Motivation du choix de SNORT

Au vu du tableau 2.1 de comparaison qui a été dressé et en fonction de nos attentes et de nos moyens pour la réalisation de notre projet, notre choix s'est porté sur l'IDS Open source SNORT. En effet Snort est l'IDS open source le plus populaire disponible. Sa flexibilité est une des principales raisons de sa popularité. SNORT peut être installé sur les systèmes Unix et Windows, bien qu'il y ait généralement plus de prise en charge tierce pour les versions Unix. Il peut fonctionner selon trois modes principaux :

- il est multiplateforme (il pourra s'adapter à tous les systèmes de l'Entreprise) ;
- il offre plusieurs possibilités pour l'obtention des alertes (les alertes peuvent être lues à l'écran, envoyées dans un fichier, envoyées en messages Popup, etc) ;
- il peut être utilisé sous plusieurs modes(comme un simple sniffer de paquets, comme un logger de trafic réseaux) ;
- il peut être utilisé comme un véritable système complet de prévention contre les attaques et les intrusions de tout genre.

## Conclusion

Ce chapitre nous a permis de mieux comprendre que nous disposons de plusieurs logiciels de détection d'intrusion. Nous avons présenté quelques-uns et avions choisi celle que l'on juge meilleure et adaptée à notre projet. Par la suite, nous allons passer à l'installation et à la configuration de la solution retenue afin d'obtenir les résultats souhaités.

Chapitre 3

# Implémentation, résultats et discussions

## Introduction

Dans ce chapitre, nous ferons une présentation plus détaillée de la solution prise qui est Snort. Nous passerons ensuite à son installation et à sa configuration pour enfin échouer sur les résultats des tests.

## 3.1 Etude détaillée de SNORT

### 3.1.1 Présentation de SNORT

SNORT est un IDS open source du système de détection des intrusions de réseau (NIDS). Il est capable d'analyser le trafic sur le réseau en temps réel et des paquets circulant sur le réseau IP. Il peut exécuter l'analyse de protocole, en cherchant le contenu et peut être employé pour détecter une variété d'attaque [5]. A l'origine, ce fut un sniffer qui, par la suite connut, une telle évolution qu'il fut vite adapté et utilisé dans le monde de détection d'intrusion en s'appuyant sur une base de signatures régulièrement enrichie par « le monde du libre ». SNORT emploie un langage flexible de règles pour décrire le trafic qu'il devrait laisser passer. SNORT a des possibilités en temps réel d'alerter et d'incorporer des mécanismes pour le système d'événement, un dossier indiqué par l'utilisateur, un socket d'Unix, ou des messages de WinPopup aux clients de Windows en utilisant la smbclient. SNORT est constitué de plusieurs composants logiques. Ces composants permettent de détecter une attaque particulière et de générer en sortie, une alerte dans le format désiré. Les composants principaux de SNORT sont : le décodeur de paquet, les préprocesseurs, le moteur de détection, le système de journalisation et d'alerte et les modules de sortie.

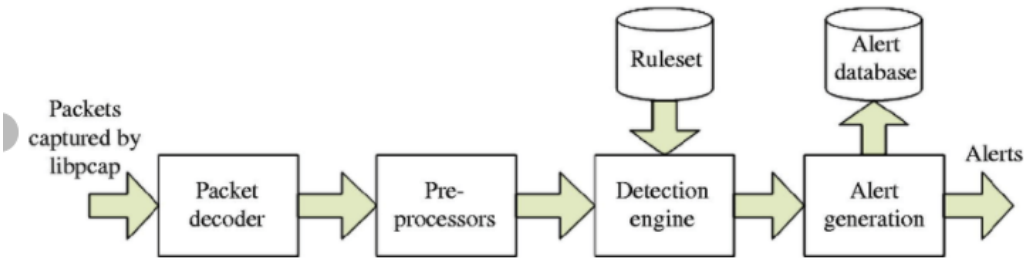



FIGURE 3.1 : Architecture de SNORT [23]

La figure 3.1 nous montre un récapitulatif de l'architecture de snort.

### 3.1.2 Description de quelques composants de SNORT

#### 3.1.2.1 Le décodeur de paquet

Un système de détection d'intrusion active un ou plusieurs interfaces réseau de la machine en mode espion (promiscuous[1] mode), ceci va lui permettre de lire et d'analyser tous les paquets qui passent par le lien de communication. SNORT utilise la bibliothèque libpcap pour faire la capture des trames [5]. Un décodeur de paquets est composé de plusieurs sous décodeurs qui sont organisés par protocole (Ethernet, IP, TCP, etc.). Ces décodeurs transforment les éléments des protocoles en une structure de données internes.

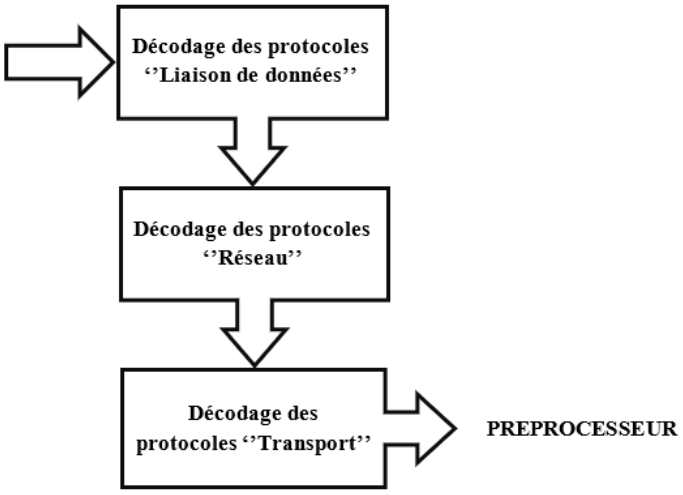



FIGURE 3.2 : Decodeur de paquets [5]

La figure 3.2 nous montre un bref fonctionnement du décodeur de paquets snort.

#### 3.1.2.2 Les préprocesseurs

Les préprocesseurs sont des modules d'extension pour arranger ou modifier les paquets de données avant que le moteur de détection n'intervienne. Certains préprocesseurs détectent aussi des anomalies dans les entêtes des paquets. La préparation des paquets pour le moteur de détection est

[1] une configuration de la carte réseau, qui permet à celle-ci d'accepter tous les paquets qu'elle reçoit, même si ceux-ci ne lui sont pas adressés.

très importante dans le cadre des IDS [21]. Les préprocesseurs sont utilisés pour le réassemblage de fragments IP. Lorsqu'une quantité importante de données est transférée, les paquets sont souvent fragmentés. Le système terminal qui réceptionne les données est capable de réassembler le paquet original. Pour pouvoir détecter des signatures, un IDS positionné sur un équipement intermédiaire doit faire de même afin de déjouer les attaques des attaquants qui utilisent cette technique de fragmentation. En gros, les préprocesseurs permettent de défragmenter les paquets, de décoder les URI, HTTP, de réassembler les flux TCP, etc.

#### 3.1.2.3 Moteur de détection

Le moteur de détection de SNORT constitue le cœur de l'IDS. Il est responsable de la détection de toute activité d'intrusion dans les flux de données. Il utilise des règles qui consistent à une définition d'un ensemble de critères sur les paquets. Si un paquet correspond à une règle, une action est réalisée (journaliser, déclenchement d'une alerte). Selon la puissance de l'équipement où est installé SNORT et le nombre de règles définies, le nombre de paquets analysés peut être plus ou moins important [22].

#### 3.1.2.4 Le système de journalisation et d'alertes

Selon les décisions prises par le moteur de détection, les paquets peuvent être journalisés ou générer une alerte. La journalisation peut s'effectuer par de simples fichiers textes ou selon d'autres formes. Dès que le système devient opérationnel, on pourra consulter les alertes générées directement dans les fichiers textes ou bien utiliser une console de gestion [22].

#### 3.1.2.5 Les modules de sortie

Les modules de sortie peuvent effectuer différentes opérations selon la manière dont on désire sauvegarder les informations générées par le système de journalisation et d'alerte [22] :

- enregistrement simple dans un fichier (comme /log/snort/alerts) ;
- envoyer des notifications d'évènement SNMP ;
- enregistrer dans une base de données comme MySQL ;
- transformer dans un format XML ;
- envoyer des messages SMB (dans le cas de machines Windows, Mail, SMS, etc.

## 3.2 Architecture et outils

### 3.2.1 Architecture déployée pour la simulation

La figure 3.3 montre l'architecture réseau utilisée pour la phase de simulation. Elle est composée d'un routeur, d'un firewall, de deux switchs, de la machine de détection, des serveurs et de quatre machines.

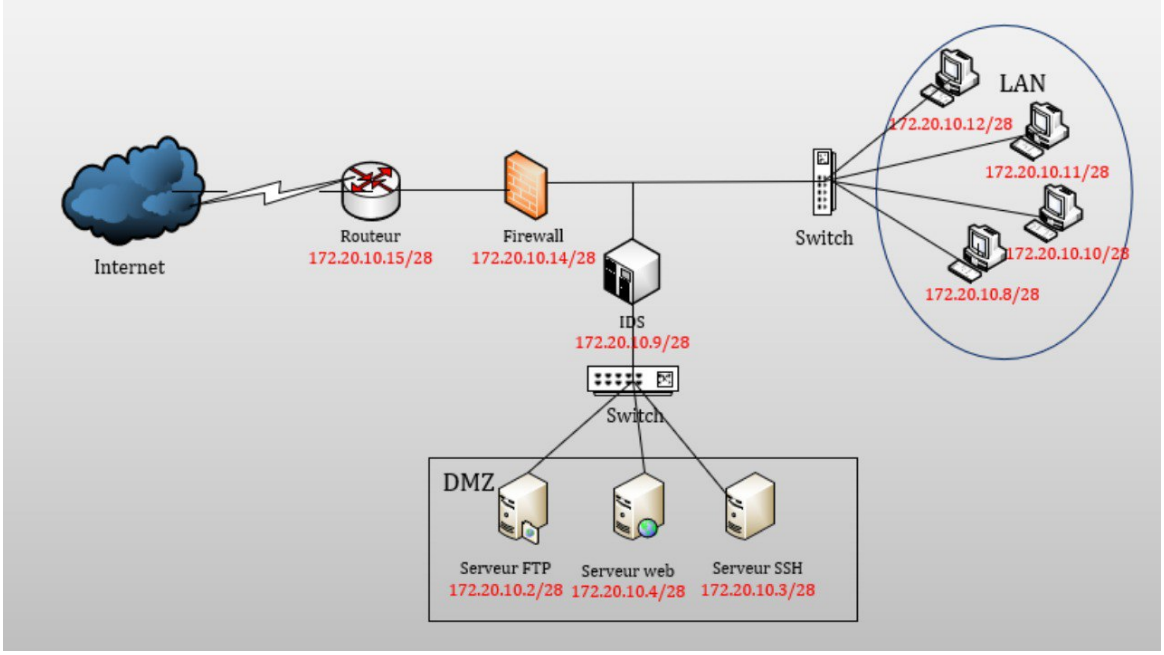



FIGURE 3.3 : Architecture déployée pour la simulation

### 3.2.2 Outils

Dans le cadre de la simulation, les outils utilisés sont :

- l'outil Snort3
- routeur Mikrotik 6.48.1
- l'image du système kali
- deux switchs Cisco SG300-52P 52-Port Gigabit PoE

## 3.3 Installation de SNORT

Avant d'installer SNORT sur la machine, nous nous sommes assurés que certaines dépendances sont préalablement installées sur cette dernière et réalisées avant la mise à jour. Tout ceci s'effectuera en se connectant avec les privilèges « root ».

Les commandes pour la mise à jour :

```
1  apt-get update
2  apt-get upgrade
```

Après la mise à jour, pour une bonne construction et une installation réussie de Snort sur notre machine, il existe un certain nombre d'outils de construction et de dépendance qui doivent être installés avant le processus de construction. La commande qui permet de les installer est la suivante :

- **apt install 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**

Les conditions préalables remplies, pour la prochaine étape, nous avons téléchargé et installé la dernière version de Snort DAQ (bibliothèque d'acquisition de données). DAQ n'est pas disponible sur les référentiels linux par défaut et, par conséquent, nous devons le construire et l'installer à partir de la source. Pour le faire, nous avons tapé les lignes de commandes suivantes [24] :

Listing 3.1 : Téléchargement et installation de DAQ

```
mkdir snort-source-files
cd snort-source-files
git clone https://github.com/snort3/libdaq.git
cd libdaq
./bootstrap
./configure
make
make install
```

Nous avons ensuite téléchargé et installé le malloc de mise en cache des threads de google, Tcmalloc, un répartiteur de mémoire optimisé pour les situations de concurrence élevée qui fournira une meilleure vitesse pour le compromis d'une utilisation plus élevée de la mémoire. Il s'agit d'une dépendance facultative mais fortement recommandée. Pour le faire, nous avons effectué une suite de commande :

Listing 3.2 : Téléchargement et installation de Tcmalloc

```
cd ../
wget https://github.com/gperftools/gperftools/releases/download/gperftools-2.8/gperftools-2.8.tar.gz
tar xzf gperftools-2.8.tar.gz
cd gperftools-2.8/
./configure
make
make install
```

Après la mise en place de toutes les dépendances requises, nous avons procédé au téléchargement et à l'installation de Snort. Nous allons retourner dans le dossier de source de SNORT créé initialement et passer au téléchargement de Snort. Pour le faire, nous avons tapé successivement les commandes :

```
    cd ../
    git clone git://github.com/snortadmin/snort3.git
```

Nous sommes allés au niveau du répertoire source de Snort, nous l'avons compilé et installé :

```
    cd snort3/
    ./configure\_cmake.sh --prefix=/usr/local --enable-tcmalloc
```

Après, nous avons accédé au répertoire de construction, compilé et installé Snort :

```
    cd build
    make
    make install
```

Une fois l'installation terminée, nous avons mis à jour les bibliothèques partagées en tapant la commande :

```
    ldconfig
```

Après, nous avons vérifié l'installation de Snort avec sa version :

```
    snort -V
```

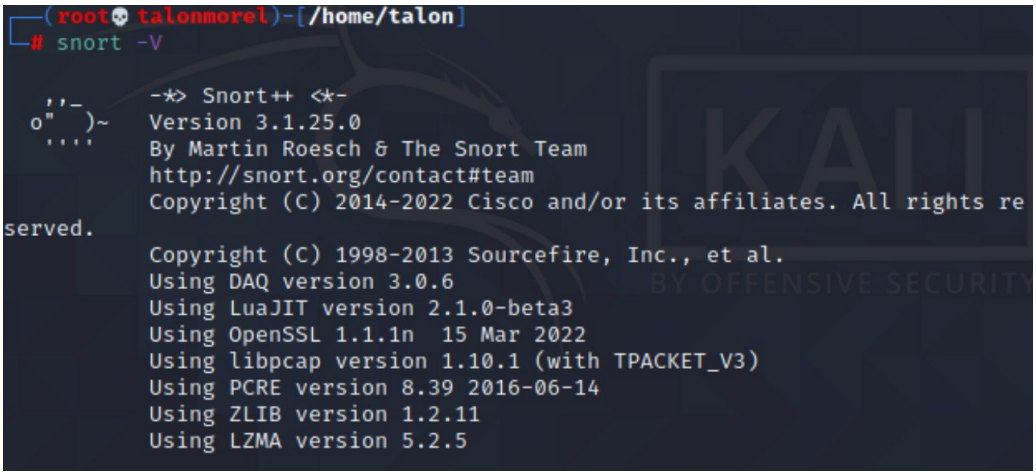



FIGURE 3.4 : Installation de Snort

L'image 3.4 confirme que l'installation de Snort est réussie et fonctionne correctement sur la machine.

## 3.4 Configuration de SNORT

Snort a été installé avec succès, mais il est loin d'être prêt à l'utilisation. Nous avons fait une série de configuration. Cette configuration concerne essentiellement la configuration des cartes d'interface réseau, la configuration des règles de Snort et la configuration de la journalisation de Snort. Pour le faire, nous avons modifié certains fichiers de configuration.

### 3.4.1 Configuration des cartes d'interface réseau

Tout d'abord, nous avons mis l'interface sur laquelle Snort écoute le trafic réseau (wlan0) en promiscuous mode afin qu'elle puisse voir tout le trafic réseau qui lui est envoyé plutôt que de voir uniquement le trafic provenant du serveur Snort seul. Pour le faire, nous avons tapé la commande [24] :

```
1  ip link set dev wlan0 promisc on
```

Vérification :

```
1  ip add sh wlan0
```

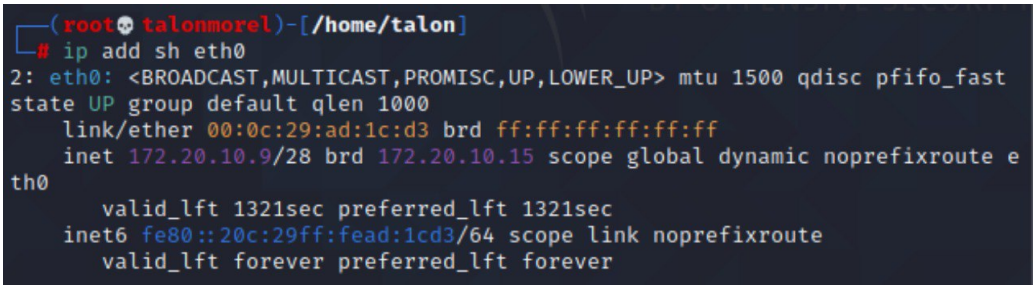



FIGURE 3.5 : Interface wlan0 en mode promiscuous

L'image 3.5 montre l'interface wlan0 en mode promiscuous.

Par la suite, nous avons désactivé le déchargement d'interface pour empêcher Snort de tronquer les gros paquets de plus de 1518 octets. Nous avons d'abord vérifié si cette fonctionnalité est activée :

```
ethtool -k wlan0 | grep receive-offload
```




FIGURE 3.6 : Interface de déchargement

L'image 3.6 montre que GRO[2] est activé alors que LRO[3] est fixe et ne peut donc pas être modifié.
Ensuite, nous avons désactivé GRO avec la commande :

```
ethtool -K wlan0 gro off lro off
```




FIGURE 3.7 : GRO off

L'image 3.7 présente toutes les deux interfaces qui sont désactivées.
Les deux changements de carte réseau sont temporaires. Pour nous assurer que les modifications persistent lors du redémarrage du système, nous avons créé et activé une unité de service systemd[4] pour implémenter les modifications. Pour le faire, nous avons tapé la commande :

```
gedit /etc/systemd/system/snort3-nic.service
```

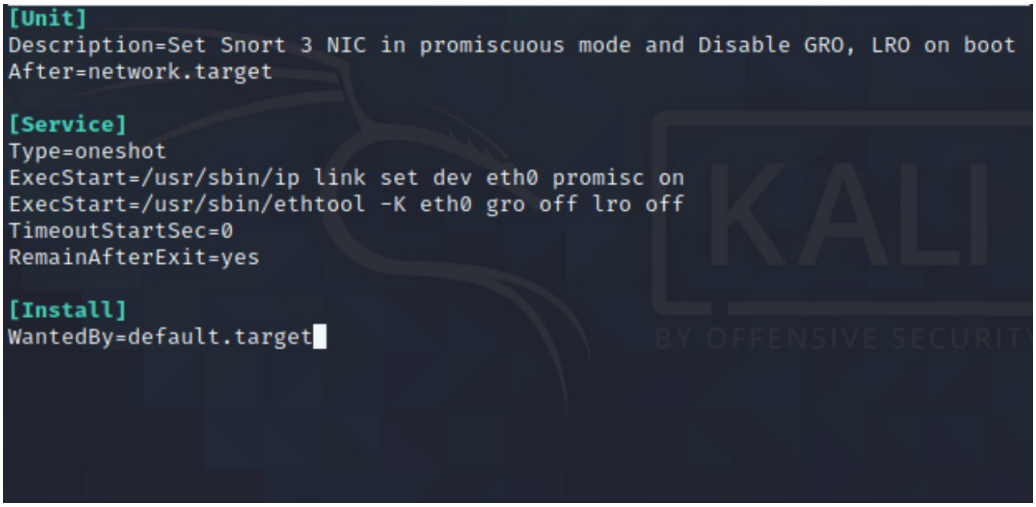



FIGURE 3.8 : Unité de service systemd

[2] generic-receive-offload
[3] large-receive-offload
[4] est une suite logicielle qui fournit une gamme de composants système pour les systèmes d'exploitation Linux. ... Le nom de ce programme vient de « system daemon » : le daemon du système

Nous avons ensuite rechargé les paramètres de configuration ***systemd***, démarré et activé le service au démarrage avec les commandes :

```
systemctl daemon-reload
systemctl enable --now snort3-nic.service
```

### 3.4.2 Installation des règles de Snort

Les règles représentent l'artère principale du moteur de détection d'intrusion Snort. Nous disposons de plusieurs types de règles. Nous avons installé pour notre travail les règles de communauté. Pour le faire, nous avons d'abord créé le répertoire Snort Rules. Dans le /usr/local/etc/snort/snort_defaults.lua fichier de configuration, le chemin des règles par défaut (RULE_PATH) est défini comme /usr/local/etc/rules [24]. Nous l'avons fait avec la commande :

```
mkdir /usr/local/etc/rules
```

Nous avons ensuite téléchargé les règles de la communauté Snort depuis la page de téléchargement de Snort avec la commande :

```
wget https://www.snort.org/downloads/community/snort3-community-rules.tar.gz
```

Après, nous avons extrait les règles et nous les avons stockées dans le répertoire des règles de Snort avec la commande :

```
tar xzf snort3-community-rules.tar.gz -C /usr/local/etc/rules/
```

Une fois les règles en place, nous sommes passés à la configuration principale de Snort. Nous avons ouvert le fichier de configuration pour le modifier avec la commande :

```
gedit /usr/local/etc/snort/snort.lua
```

Ensuite, nous avons défini les réseaux à protéger contre les attaques comme valeur de la HOME_NET variable. Dans notre cas, l'adresse mis n'est rien d'autre que notre adresse réseau qui est : 172.20.10.0/24. Le EXTERNAL_NET est sur option **any**[5] .

```
18 ---------------------------------------------------------------------------
19 -- 1. configure defaults
20 ---------------------------------------------------------------------------
21
22 -- HOME_NET and EXTERNAL_NET must be set now
23 -- setup the network addresses you are protecting
24 HOME_NET = '172.20.10.0/28'
25
26 -- set up the external network addresses.
27 -- (leave as "any" in most situations)
28 EXTERNAL_NET = 'any'
29
30 include 'snort_defaults.lua'
31 include 'file_magic.lua'
32
```

FIGURE 3.9 : Configuration du réseau à protéger

L'image 3.9 nous montre la configuration du réseau sur lequel les intrusions seront détectées.

Sous la section ips, nous avons défini l'emplacement selon nos règles. Nous avons inclus les règles avec la commande :

```
include = '/usr/local/etc/rules/snort3-community-rules/snort3-community.rules'
```

[5] n'importe quel réseau

```
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)

    variables = default_variables,
    include = '/usr/local/etc/rules/snort3-community-rules/snort3-community.rules'
}
```

FIGURE 3.10 : Community-rules inclus

La figure 3.10 montre que nous avons ajouté dans la section ips les règles de communauté.

### 3.4.3 Installation de Snort OpenAppID

OpenAppID est un plug-in de la couche d'application qui permet à Snort de détecter diverses applications, Facebook, Netflix, Twitter et Reddit, utilisées sur le réseau. Nous avons exécuté les commandes ci-dessous à partir de la page de téléchargement de Snort et installé Snort OpenAppID :

Listing 3.3 : Installation de Snort OpenAppID

```
wget https://snort.org/downloads/openappid/12159 -O OpenAppId-12159.tgz
tar -xzvf OpenAppId-12159.tgz
cp -R odp /usr/local/lib/
```

Ensuite, nous avons modifié le fichier de configuration de Snort et défini l'emplacement des bibliothèques OpenAppID. Pour le faire, nous avons d'abord ouvert le fichier de configuration avec la commande : ***gedit /usr/local/etc/snort/snort.lua***. Nous avons inséré dans le fichier ces deux lignes :

```
  app\_detector\_dir = '/usr/local/lib'
  log\_stats = true
```

L'image 3.11 montre l'emplacement des bibliothèques OpenAppID.

```
appid =
{
    -- appid requires this to use appids in rules
    --app_detector_dir = 'directory to load appid detectors from'
    app_detector_dir = '/usr/local/lib',
    log_stats = true,
}
```

FIGURE 3.11 : Emplacement des bibliothèques OpenAppID

Nous avons enregistré et quitté le fichier de configuration.
Nous avons créé après le répertoire Snort Log :

```
  mkdir /var/log/snort
```

### 3.4.4 Création des règles locales

Nous avons créé des règles locales personnalisées dans le but de tester la configuration de notre outil. D'abord, nous avons créé un fichier dans lequel nous avions créé nos règles. Pour le faire, nous avons tapé la commande :

```
gedit /usr/local/etc/rules/local.rules
```

Nous avons ensuite ajouté dans le fichier local.rules nos règles.

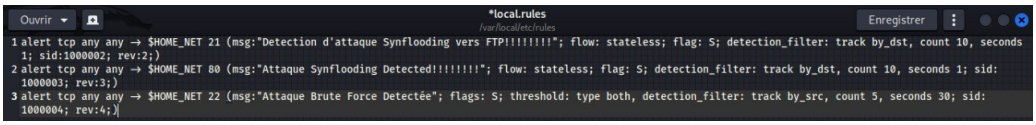



FIGURE 3.12 : Règles locales

La figure 3.12 présente les règles créés pour le test. Pour notre cas, nous avons mis en place trois règles dont l'objectif serait de détecter des tentatives d'attaques Ddos sur les ports 22(SSH), 21(FTP). C'est à dire que lorsqu'il y aura une arrivée de plus de 10 paquets en 1 seconde sur l'un de ces ports, Snort va signaler une intrusion.

Nous avions ensuite inclus nos règles dans le fichier de configuration snort.lua.

```
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)

    variables = default_variables,
    include = '/usr/local/etc/rules/snort3-community-rules/snort3-community.rules',
    include = '/usr/local/etc/rules/local.rules'
}
```

FIGURE 3.13 : Inclusion des règles locales

### 3.4.5 Journalisation de Snort

Pour écrire des événements Snort dans les fichiers journaux, nous avons activé la configuration des paramètres d'alerte. Il existe différentes options de journalisation de Snort. Pour sortir les données d'événement dans un fichier (tel que défini dans la ligne de commande ci-dessus par l'option -A alert_type), nous avons ouvert le fichier de configuration snort.lua avec la commande ***gedit /usr/local/etc/snort/snort.lua*** et nous nous sommes dirigés vers la section des sorties.

```
-- 7. configure outputs

-- event logging
-- you can enable with defaults from the command line with -A <alert_type>
-- uncomment below to set non-default configs
--alert_csv = { }
alert_fast = {
        file = true,
        packet = false,
        limit = 10,
}
--alert_full = { }
--alert_sfsocket = { }
--alert_syslog = { }
--unified2 = { }
```

FIGURE 3.14 : Configuration des sorties

La figure 3.14 présente la partie de configuration de sorties de Snort dans laquelle nous avons ajouté des lignes. Dans notre cas, nous obligeons Snort à enregistrer toutes les alertes dans un fichier alert_fast.txt. Nous avons enregistré et quitté la configuration. Nous avons par la suite, vérifié le répertoire des journaux. Nous avons remarqué que notre fichier alert_fast.txt a été bien créé.

Enfin, nous avons exécuté la vérification de la syntaxe :

```
snort -c /usr/local/etc/snort/snort.lua
```

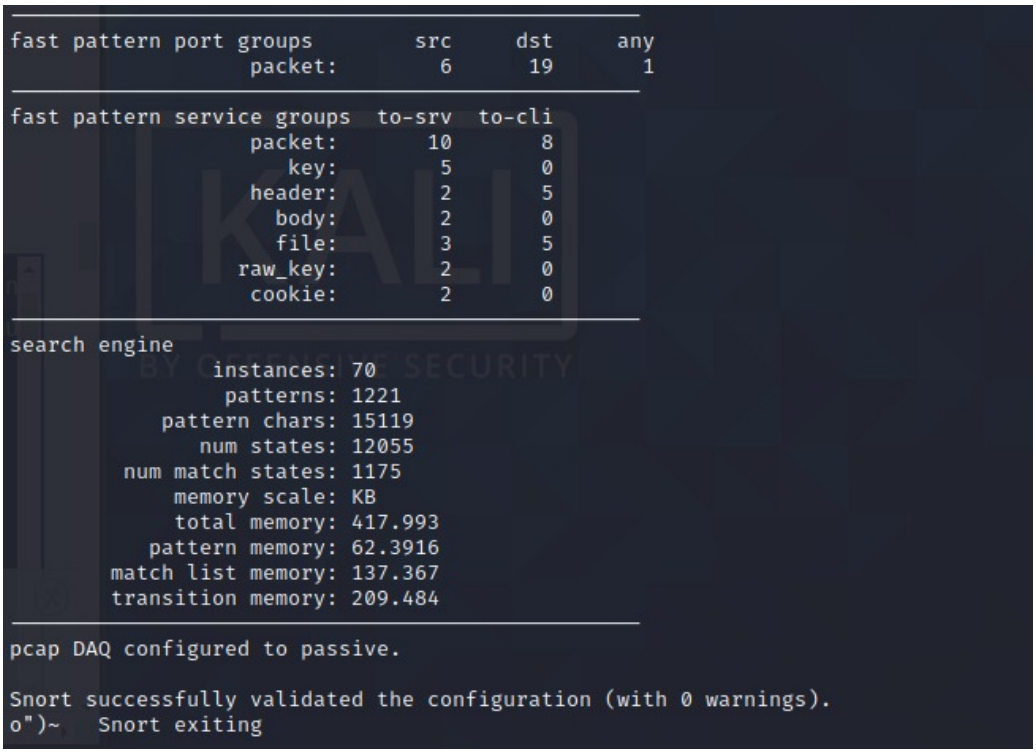



FIGURE 3.15 : Configuration de Snort réussie

L'image prouve que Snort a validé avec succès la configuration (avec 0 avertissement).

### 3.4.6 Configuration de l'alerte par mail

Pour cette partie, nous avons écrit un code python qui va permettre d'envoyer le fichier **alert_fast.txt** par mail. Pour le faire, nous avons utilisé certains modules de python. Déjà, Python est livré avec le **smtplib** module intégré pour envoyer des e-mails à l'aide du protocole SMTP. smtplib utilise le protocole RFC 821 pour SMTP. Nous avons aussi ajouté le module **ssl** pour assurer que la connexion SMTP soit cryptée pour empêcher que le message et les identifiants de connexion ne soient pas facilement accessibles par d'autres, le module **sys** qui nous a permis d'utiliser les commandes systèmes qui sont aussi des modules préétablies et enfin le module **pyinotify** qu'on a installé avec la commande ***pip3 install pyinotify***. Par la suite, dans le code, nous avons précisé l'adresse mail de l'expéditeur(ebenezertalon99@gmail.com) et l'adresse du destinataire (momotalon134@gmail.com).

Listing 3.4 : Code python de configuration d'envoi de mail

```
import smtplib, ssl
import sys
import pyinotify

from email import encoders
from email.mime.base import MIMEBase
```

```
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

subject = "Alert venant de Snort"
body = "Retrouvez ci-joint le fichier d'alerte."
sender_email = "ebenezertalon99@gmail.com"
receiver_email = "momotalon134@gmail.com"
port = 465  # Par SSL
password = "********"

filename = "document.pdf"  # Dans le meme répertoire que le script

def getPartFromFile(filename):# Permet de définir les entetes du fichier qu'on veut envoyer

    with open(filename, "rb") as attachment:
        part = MIMEBase("application", "octet-stream")
        part.set_payload(attachment.read())


    encoders.encode_base64(part)

    part.add_header(
        "Content-Disposition",
        f"attachment; filename= {filename}",
    )
    return part

def forgeMessage(filename): # Permet de créer le message envoyé
    global body, sender_email, receiver_email, subject
    # On a créé un message en plusieurs parties et définir des headers
    message = MIMEMultipart()
    message["From"] = sender_email
    message["To"] = receiver_email
    message["Subject"] = subject
    message["Bcc"] = receiver_email  #Recommandé en cas de plusieurs mails
    # On a ajouté une PJ au message et convertir le message en chaine
    # On a ajouté du corps au mail
    message.attach(MIMEText(body, "plain"))
    message.attach(getPartFromFile(filename))
    return message.as_string()



# Ensuite nous avons créé un contexte SSL sécurisé
context = ssl.create_default_context()

def send_mail(filename): # Permet d'envoyer le message
    with smtplib.SMTP_SSL("smtp.gmail.com", port, context=context) as server:
        server.login(sender_email, password)
        server.sendmail(sender_email, receiver_email, forgeMessage(filename))
    print("Mail sucessfully send!")

def getMessage(path): # Permet d'extraire la nouvelle alert
    try:
        index = open('.trace','r').read()
    except :
        open('.trace','w').write('0')
        return getMessage(path)
    index = int(index)
    print('{ { { {', index)
    file = open(path,'r')
    file.seek(index)
    out = file.read()
    open('.trace','w').write(str(file.tell()))
```

```
71      return out
72
73  class OnWriteHandler(pyinotify.ProcessEvent): # Cette classe est chargé d'enclencher la procédure
74  d'envoie de mail en cas de modification du fichier des alertes
75      def my_init(self, cwd):
76          self.cwd = cwd
77
78      def _run_cmd(self):
79          print('==> Modification detected')
80          #message = getMessage(self.cwd)
81          send_mail(self.cwd)
82          #subprocess.call(self.cmd.split(' '), cwd=self.cwd)
83
84      def process_IN_MODIFY(self, event):
85          self._run_cmd()
86
87  def auto_compile(path): # C'est un écouteur qui est chargé de détecter la modification du fichier
88  d'alertes et d'enclencher la procédure d'envoi de mail
89      wm = pyinotify.WatchManager()
90      handler = OnWriteHandler(cwd=path)
91      notifier = pyinotify.Notifier(wm, default_proc_fun=handler)
92      wm.add_watch(path, pyinotify.ALL_EVENTS, rec=True, auto_add=True)
93      print('==> Start monitoring %s (type c^c to exit)' % path)
94      notifier.loop()
95
96  if __name__ == '__main__':
97
98      if len(sys.argv) != 2:
99          print("Specifié le fichier svp")
100         exit()
101     # Arguments requis
102     path = sys.argv[1]
103
104     # Surveillance des blocs
105     auto_compile(path)
```

Le code présenté ci-dessus, après avoir été lancé, surveille le fichier d'alerte et à chaque fois qu'une ligne va s'ajouter dans le fichier, il va envoyer le fichier à l'administrateur dont l'adresse mail est **momotalon134@gmail.com**.

## 3.5 Résultats

Nous avons d'abord installé les services FTP, SSH pour pouvoir effectuer les tests. Pour l'installation de ces deux services, nous avons tapé les commandes :

```
1  apt-get install openssh-server
2  apt-get install ftpd
```

### 3.5.1 Détection d'attaque

Avant lancement de toute attaque, nous avons lancé le service snort avec la commande :

***- snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/rules/local.rules -i wlan0 -s 65535 -k none -l /var/log/snort/***

#### 3.5.1.1 Attaque par déni de service

Pour cette partie, nous avons utilisé Scapy qui est un utilitaire permettant de forger, de recevoir et d'envoyer des paquets ou des trames de données sur un réseau pour une multitude de protocoles. Scapy va donc nous permettre de créer et d'envoyer les paquets qui vont nous permettre de réaliser notre attaque. Pour le faire, nous avons procédé d'abord au lancement de Scapy.

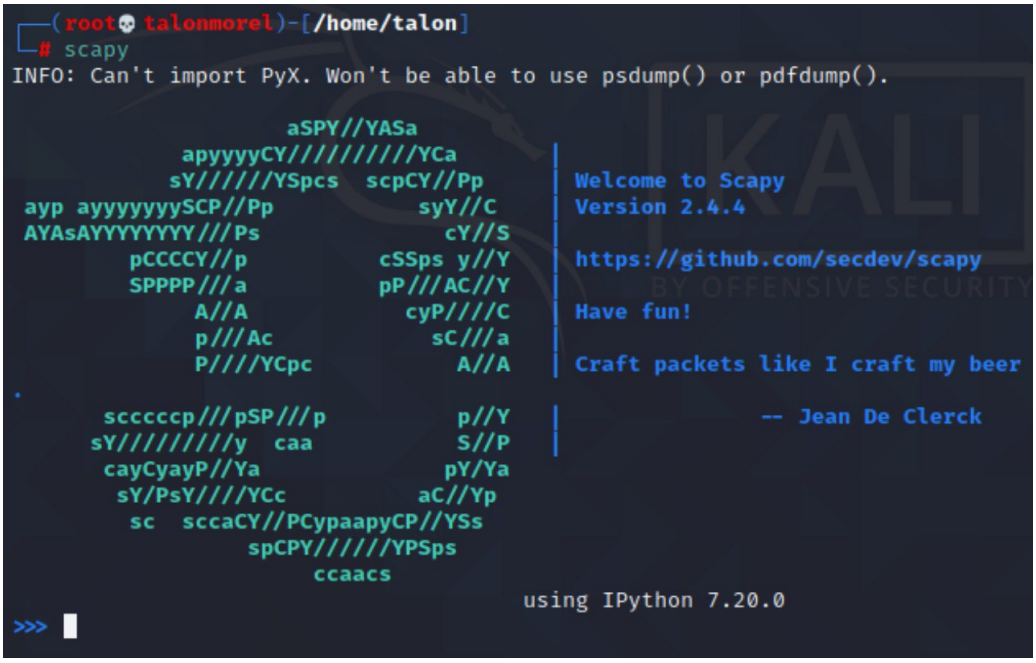



FIGURE 3.16 : Lancement de Scapy

La figure 3.16 présente l'outil Scapy au début du démarrage.
Nous avons ensuite créé nos paquets :




FIGURE 3.17 : Création des paquets

Comme illustré sur la figure 3.17, nous avons créé et envoyé deux paquets. Pour chaque paquet, nous avons précisé l'adresse de destination, l'adresse source, le protocole, le port et le nombre de fois que le paquet sera envoyé. Dans notre cas, Scapy va pemettre à la machine dont l'adresse IP est le 172.20.10.8, d'envoyer simultanément 20 paquets respectivement sur les machines dont les adresses IP sont le 172.20.10.1 et le 172.20.10.2 contenant le service FTP.

Nous avons ouvert le fichier alert_fast.txt créé initialement. Les alertes ont été bien détectées et bien enregistrées dans le fichier comme le montre la figure 3.18.

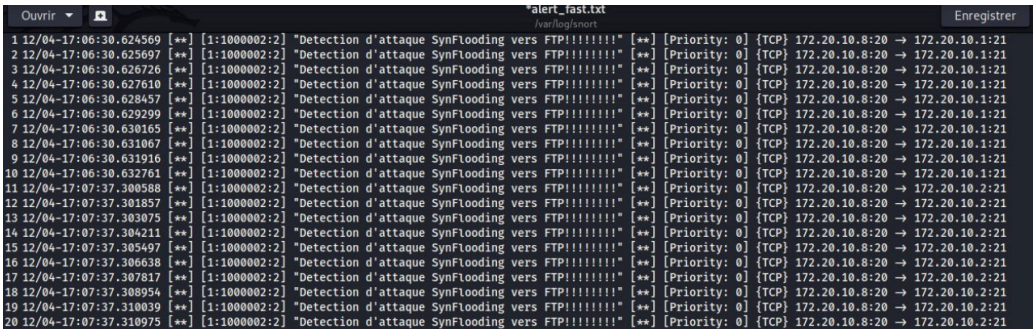



FIGURE 3.18 : Enregistrement des alertes dans le fichier alert_fast.txt

### 3.5.1.2 Attaque par brute force

Dans cette partie, nous avons utilisé l'outil ***Hydra*** qui permet de réaliser une attaque par brute force. Dans notre cas, nous avons essayé de craquer le mot de passe de l'utilisateur user du serveur ssh.

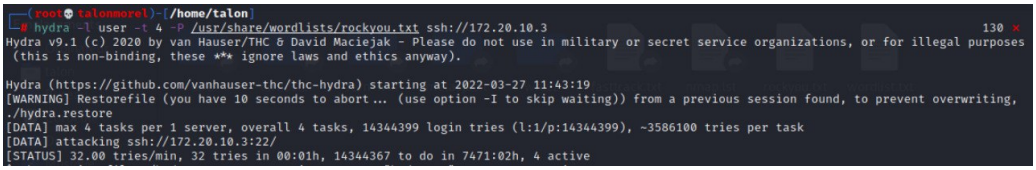



FIGURE 3.19 : Lancemennt de l'attaque par brute force

La figure 3.19, montre notre tentative de brute force. Nous avons utilisé le dictionnaire de mot de passe ***rockyou.txt*** pour le craquage du mot de passe.

Nous avons une fois encore ouvert le fichier d'alerte et nous avons remarqué que des lignes d'alerte ont été ajoutées au fichier.

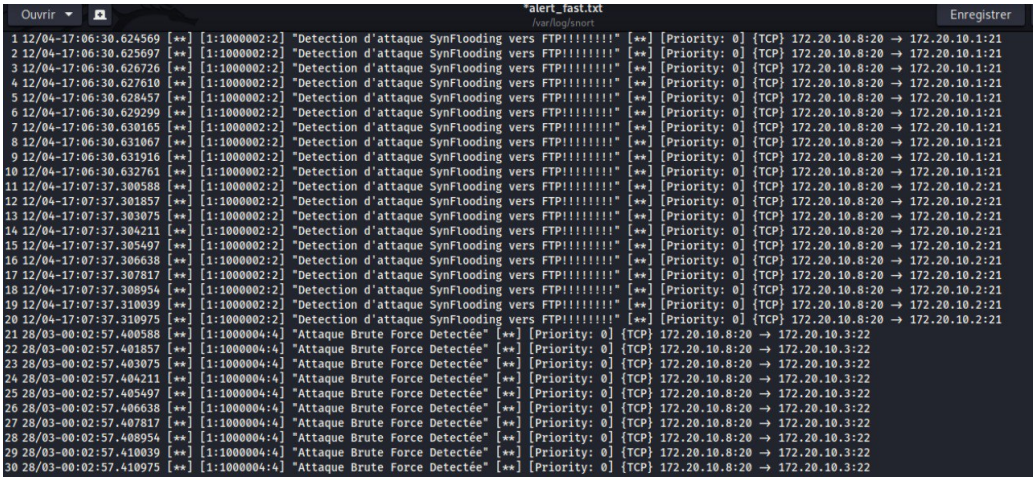



FIGURE 3.20 : Enregistrement de nouveaux alertes au fichier d'alerte

La figure 3.20, montre qu'après le lancement de la nouvelle attaque, d'autres lignes se sont ajoutées auomatiquement au fichier d'alerte.

## 3.5.2 Alerte par mail

Pour cette partie, nous avons lancé le code python avec la commande ***python3 mail.py /var/log/snort/alert_fast.txt***. mail.py représente le nom du fichier contenant le code. Après lancement de la

ligne, le programme fonctionne en arrière-plan.

Ensuite pour vérifier si réellement le fichier alert_fast.txt sera envoyé par mail, nous avons relancé les attaques.

Après la détection des attaques, nous avons reçu une notification sur le mail momotalon134@gmail.com. Cette notification n'est juste que notre fichier avec les nouvelles lignes d'alerte qui ont été enregistrées dans le fichier.

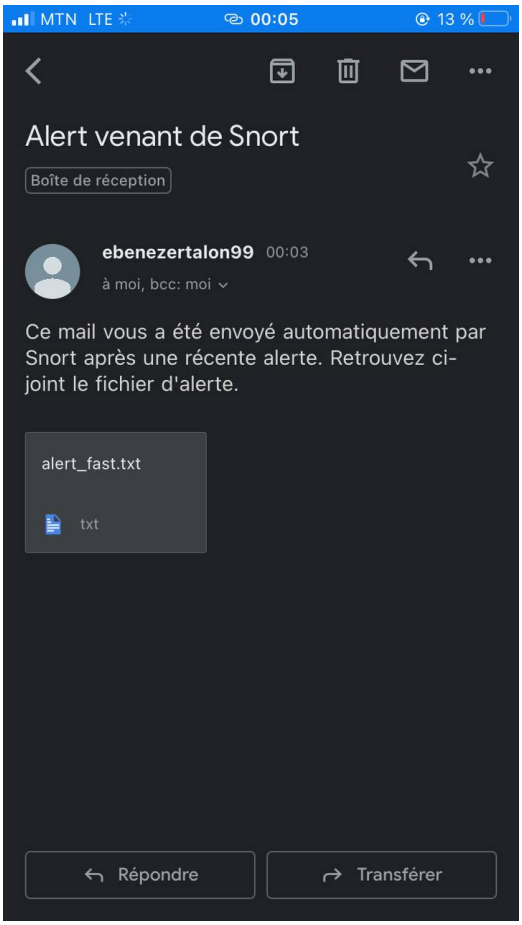



FIGURE 3.21 : Réception du mail

La figure 3.21 montre que le fichier contenant les alertes a été bien reçu sur le mail du destinataire.

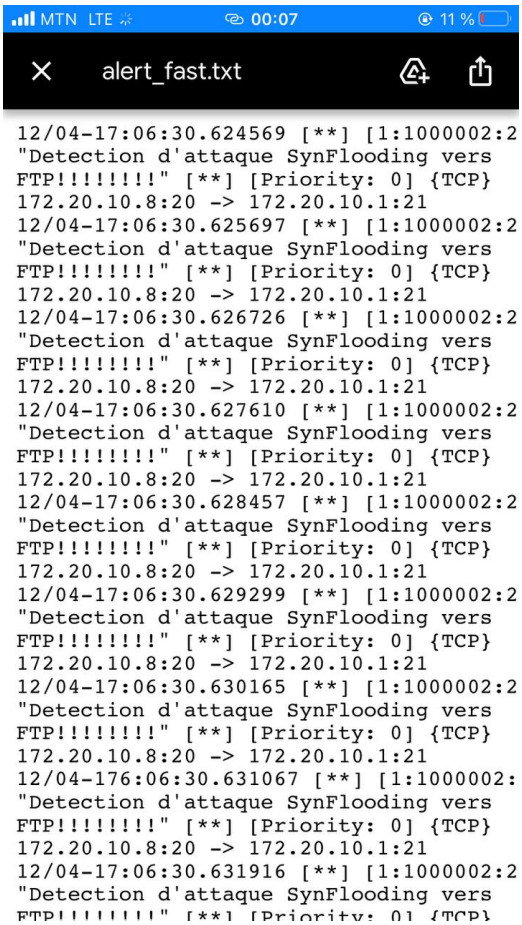



FIGURE 3.22 : Contenu du mail

La figure 3.22 montre le contenu du fichier et nous montre que ce fichier comporte vraiment les alertes. A ce moment, l'administrateur peut prendre connaissance de l'attaque et pourra prendre des mesures.

## 3.6 Discussion

Notre projet est né de plusieurs constats généraux de l'ordre de la sécurité informatique pour la protection de l'information et de la vie privée. Les recherches effectuées nous ont permis de constater l'existence de certaines solutions. Outre la mise en place des pare-feu et des systèmes d'authentification, il est de nos jours nécessaire de mettre en place un système de détection d'intrusion. Nous avons étudié le fonctionnement de ces systèmes, en particulier, nous avons pris comme exemple l'outil Open Source SNORT qui est le plus réputé en terme d'efficacité et présente une souplesse en terme de personnalisation. Pour une meilleure performance de notre solution, notre outil a été suppléé par un système d'envoi de mail.

Ainsi, d'une part, notre solution est simple à utiliser et est accessible à un grand public. Elle permet aux utilisateurs d'être informés de tout ce qui peut être anormal dans le système à leur insu.

D'autre part, notre solution présente également quelques limites :

- Les cartes réseaux sont tous configurées en mode promiscuous. Il se peut alors que la machine répond à certains messages (icmp echo request). Si la machine n'est pas en mode "promiscuous", elle ignorera le paquet sinon, elle répondra ;

- Sur certaines distributions linux, il est plus difficile de faire fonctionner correctement l'outil ;
- Il peut arriver que les alertes prennent du temps avant de venir ;
- L'administrateur doit être constamment connecté à internet.

## Conclusion

Ce chapitre nous a permis de confirmer que SNORT est un outil puissant et efficace qui génère des alertes en temps réel. Par des tests, nous avons vu comment snort a pu détecter les intrusions causées par l'attaque. Nous envisageons dans un futur de l'améliorer pour son utilisation.

# Conclusion générale et perspectives

Une solution de détection d'intrusion permet de détecter à temps réel les intrusions et permet ainsi à l'administrateur d'avoir une bonne vue d'ensemble du réseau. Notre travail consistait à étudier et à mettre en place au sein de l'APDP, un outil open source de détection d'intrusion.

Pour mener à bien cette tâche, nous avons commencé par donner une vue d'ensemble de la sécurité informatique, ensuite nous avons étudié les solutions déjà existantes, faire une comparaison entre elles afin de choisir celle répondant le mieux aux besoins exigés par l'entreprise. Ensuite, nous sommes arrivés sur la mise en place de l'outil retenu(Snort) et nous l'avons ensuite configuré de sorte qu'il puisse détecter différentes sortes d'attaques tout en alertant par mail l'administrateur.

Bien que notre solution réponde à la majorité des besoins énoncés, il faut néanmoins noter certaines insuffisances telles :

- qu'une procédure à suivre lorsqu'une intrusion intervient dans l'entreprise ;
- une base de données pour stocker les alertes ;
- une programmation des actions d'auto-défense du système.

# Bibliographie


[1] Samiou WARAKPE,*Implémentation de SNORT/BASE pour la détection d'intrusion avec alerte par SMS :Cas de la FECECAM-Bénin*,2018-2019.

[2] David Burgermeister et Jonathan Krier.*Anatomie d'une attaque-Les systèmes de détection d'intrusions.64 pages*, Juillet 2006.

[3] Luc SEMASSA,*Système de détection des attaques par webcam et de prévention d'intrusion sous Linux*,2016-2017.

[4] MICKAËL CHOISNARD. *RÉSEAUX ET SÉCURITÉ INFORMATIQUES.30 pages*, Novembre 2015.

[5] Hamzata Gueye,*Mise en place d'un IDS en utilisant Snort*,2010.

[6] David Burgermeister, Jonathan Krier. *Les systèmes de détection d'intrusions*, Juillet 2006.

[7] BELKHATMI Keltouma, BENAMARA Ouarda. *Mise en place d'un système de détection et de prévention d'intusion*, 2015-2016.

[8] BENZIDANE KARIM. *Getion et centralisation des logs avec leurs corrélation*, 2009-2010.

[9] Solange Ghernaouti. *Sécurité informatique et réseaux*, DUNOD 2011.

[10] Christopher Hadnagy. *Social Engineering : The Art of Human Hacking*, Illustrated 2010.

# Webographie


[11] Piloter.org, *Système d'information*, https://www.piloter.org/systeme-information/securite-informatique.htm, consulté le 06 août 2021.

[12] CommentCaMarche.net, *Introduction à la sécurité informatique*, https://web.maths.unsw.edu.au/~lafaye/CCM/secu/secuintro.htm, consulté le 12 août 2021.

[13] reseauwebplus.com, *INTRODUCTION À LA SÉCURITÉ INFORMATIQUE* , https://www.reseauwebplus.com/introduction-a-la-securite-informatique-partie-2/, consulté le 13 août 2021.

[14] CommentCaMarche.net, *Attaques et arnaques*, https://web.maths.unsw.edu.au/~lafaye/CCM/attaques/attaques.htm, consulté le 12 août 2021.

[15] Security Info/, *Le Déni de Service (DoS)*, https://www.securiteinfo.com/attaques/hacking/dos.shtml, consulté le 16 Août 2021.

[16] Ionos.fr, *Un serveur proxy, c'est quoi ?*, https://www.ionos.fr/digitalguide/serveur/know-how/un-serveur-proxy-cest-quoi/, consulté le 20 Juillet 2021.

[17] igm.univ-mlv.fr, *Intrusion Détection Système*, http://igm.univ-mlv.fr/~dr/XPOSE2009/Sonde_de_securite_IDS_IPS/IDS.html, consulté le 26 Août 2021.

[18] commentouvrir.com, *Logiciel gratuit de détection et de prévention des intrusions*, https://commentouvrir.com/blog/logiciel-gratuit-de-detection-et-de-prevention-des-intrusions/, consulté le 2 Septembre 2021.

[19] techno-science.net, *Prelude - Définition et Explications*, https://www.techno-science.net/glossaire-definition/Prelude.html, consulté le 2 Septembre 2021.

[20] dementium2.com, *Explication des systèmes de détection d'intrusion : 12 meilleurs outils IDS examinés*, https://dementium2.com/administrateur-net/explication-des-systemes-de-detection-d-intrusion/, consulté le 2 Septembre 2021

[21] github.com, *TP Network Security Monitoring*, https://github.com/jmanteau/lprims-nsm/blob/master/TP, consulté le 10 Novembrebre 2021

[22] irit.fr, *Prise de contact avec Snort, scapy*, https://www.irit.fr/~Romain.Laborde/du3si/DU3SI-TP1-SNORT.pdf, consulté le 10 Novembrebre 2021.

[23] researchgate.net, *Snort Architecture*, https://www.google.com/imgres, consulté le 12 Décembre 2022.

[24] kifarunix.com, *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 15 Janvier 2022.

# 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** 1

**1 Revue de littérature** 3

Introduction . . . . . 3

1.1 La sécurité informatique . . . . . 3

1.1.1 Définition . . . . . 3

1.1.2 Objectifs de la sécurité informatique . . . . . 3

1.1.3 La mise en place d'une politique de sécurité . . . . . 4

1.1.4 Les attaques informatiques . . . . . 5

1.1.4.1 Anatomie d'une attaque . . . . . 5

1.1.4.2 Quelques attaques informatiques . . . . . 6

1.1.5 Les dispositifs de protection . . . . . 8

1.1.5.1 Un antivirus . . . . . 8

1.1.5.2 Un pare-feu . . . . . 8

1.1.5.3 Serveur proxy : . . . . . 8

1.1.5.4 La DMZ : . . . . . 8

1.2 Système de Détection d'intrusion . . . . . 9

1.2.1 Définition générale des IDS . . . . . 9

1.2.2 Les fonctions principales d'un IDS . . . . . 9

1.2.3 Emplacement de l'IDS dans le réseau . . . . . 9

1.2.4 Les différents types d'IDS . . . . . 9

1.2.5 Comparaison entre les types d'IDS . . . . . 10

1.2.6 Méthodes de détection des intrusions par les IDS . . . . . 11

1.2.6.1 Approche par signature ou scénario . . . 11
1.2.6.2 L'approche comportementale (anomaly detection) . . . 12
1.2.7 Les différents types d'alertes d'un IDS . . . 12
1.3 Système de Prévention d'intrusion . . . 13
1.3.1 Définition . . . 13
1.3.2 Les types d'IPS . . . 13
1.3.2.1 La prévention d'intrusion basée sur l'hôte HIPS . . . 13
1.3.2.2 La prévention d'intrusion basée sur le réseau(NIPS) . . . 13
1.3.2.3 La détection d'intrusion basée sur noyau KIPS . . . 13
1.3.3 Architecture fonctionnelle d'un IPS . . . 14
1.3.4 La Différence entre IPS et Firewall . . . 14
Conclusion . . . 14

**2 Etude comparative des IDS Open Source et choix technique 15**
Introduction . . . 15
2.1 Etude de l'existant . . . 15
2.1.1 SNORT . . . 15
2.1.2 Prelude . . . 16
2.1.3 OSSEC . . . 16
2.1.4 Bro ou Zeek . . . 17
2.1.5 Samhain . . . 17
2.1.6 Suricata . . . 17
2.2 Etude comparative . . . 18
2.3 Fonctionnalités et résultats attendus de la solution . . . 19
2.3.1 Besoins fonctionnels . . . 19
2.3.2 Besoins non fonctionnels . . . 19
2.3.3 Résultats attendus de la solution . . . 20
2.4 Motivation du choix de SNORT . . . 20
Conclusion . . . 20

**3 Implémentation, résultats et discussions 21**
Introduction . . . 21
3.1 Etude détaillée de SNORT . . . 21
3.1.1 Présentation de SNORT . . . 21
3.1.2 Description de quelques composants de SNORT . . . 22
3.1.2.1 Le décodeur de paquet . . . 22
3.1.2.2 Les préprocesseurs . . . 22
3.1.2.3 Moteur de détection . . . 23
3.1.2.4 Le système de journalisation et d'alertes . . . 23
3.1.2.5 Les modules de sortie . . . 23
3.2 Architecture et outils . . . 23
3.2.1 Architecture déployée pour la simulation . . . 23
3.2.2 Outils . . . 24
3.3 Installation de SNORT . . . 24
3.4 Configuration de SNORT . . . 26

3.4.1 Configuration des cartes d'interface réseau . . . 26
3.4.2 Installation des règles de Snort . . . 28
3.4.3 Installation de Snort OpenAppID . . . 29
3.4.4 Création des règles locales . . . 29
3.4.5 Journalisation de Snort . . . 30
3.4.6 Configuration de l'alerte par mail . . . 31
3.5 Résultats . . . 33
3.5.1 Détection d'attaque . . . 33
3.5.1.1 Attaque par déni de service . . . 34
3.5.1.2 Attaque par brute force . . . 35
3.5.2 Alerte par mail . . . 35
3.6 Discussion . . . 37
Conclusion . . . 38

**Conclusion** **39**

**Bibliographie** **40**

**Webographie** **41**

**Table des matières** **43**