N° D'ORDRE : . . . . . .

République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

Université D[r] Moulay Tahar
Faculté Des Sciences Exactes
Département D'Informatique

Mémoire de fin d'étude
Pour l'obtention du diplôme de Master II en informatique
Option : Sécurité Informatique et Cryptographie

Thème

Etude et mise en place d'un Système de Détection d'Intrusion sous Linux

Présentée par :

- Mokeddem Mahammed
- Terras Abdelkader

Encadreur :

- M[me]. Fadia TALEB BENDIMERAD Maître de conférences à l'UMT Saida

Année universitaire 2019/2020

ملخص :

إن الشبكة العنكبوتية انترنيت أحدثت ثورة تكنولوجية كبيرة من حيث تبادل المعلومات والمعارف والعلوم، مما أدى إلى حتمية استعمالها في الحياة الشخصية والمهنية، فأصبح مصير الناس رهينة لهذه الشبكة. هذه الضرورة جعلت البعض يستغلونها بطرق غير شرعية من خلال التجسس، الابتزاز، التخريب و سرقة المعطيات... مما أدى إلى ظهور وسائل حماية مثل أنظمة كشف الفيروسات، الجدار الناري، تقنيات كلمات المرور وأنظمة التشفير...وغيرها من وسائل الحماية.

كل أنظمة الحماية أظهرت قصور وثغرات تركت المجال للقراصنة للولوج. مما أدى بالباحثين إلى اختراع أنظمة كشف التسلل، التي عالجت الكثير من عيوب الحلول السابقة، تجدر الإشارة أنه توجد عدة أنماط لأنظمة كشف التسلل منها ما يرتكز على قاعدة معطيات تحوي على بصمات الهجمات المعروفة والتحليل البروتوكولي مثل سنورت ومنها ما يرتكز على تصرفات المستخدم والتطبيقات أو ما يسمى بنمط التصرفات.

في بحثنا قمنا بتثبيت نظام كشف التسلل سنورت والذي يعتبر من الأنظمة الرائدة في مجال البحث وكشف التسللات وقمنا بتثبيت برنامج إدارة العرض سبلينك لتسهيل قراءة وتحليل الإشعارات والتسجيلات .

رغم ريادته في مجال كشف التسللات إلا أنه يوجد قصور في كشف وتحديد الهجمات الجديدة غير المعرفة في قاعدة بياناته بالإضافة إلى الهجمات الموزعة، كما يعاني من دقة تحديد الحزم العادية و الخبيثة، وعلى هذا الأساس حاولنا إدخال نموذج وفق التعلم العميق باستخدام الخلايا العصبية المتكررة يعمل على نمط كشف التسلل من خلال التصرفات لمعالجة هذه النقائص.

مستقبلا نرغب في تطوير وتحسين النموذج المقترح حيث يمكن من تحديد نوعية الهجمة بالإضافة إلى معالجة حزم الشبكة مباشرة.

الكلمات المفتاحية : أمن المعلوماتية، نظام كشف التسلل، تعلم الآلة، التعلم العميق، نهج السيناريو، النهج السلوكي، الشبكات العصبية المتكررة، ك د د 99

Abstract :

Internet, caused a great technological revolution in terms of the exchange of information, knowledge and science, which led to the inevitability of its use in personal and professional life, and the fate of people became hostage of this network. This necessity has led some to exploit it illegally by espionage, extortion, sabotage, data theft ... which has led to the emergence of protection methods such as Anti-virus systems, firewalls, security technologies, password, and encryption systems ... and other means of protection.

All the security systems had loopholes and tightening loopholes that exploited by hackers. This prompted researchers to invent intrusion detection systems, which corrected many of the shortcomings of previous solutions. It should be noted that there are several types of intrusion detection systems, some of which are based on the scenario approach like IDS Snort, and some of them are based on the behavioral approach of users and applications. .

In our project, we installed & configured the Snort intrusion detection system, which is considered to be one of the leading systems in the intrusion detection field, and we installed the graphical display management software Splunk for easy reading and analysis of the alarms and logs that Snort generates.

Despite its power in the field of intrusion detection, there is a deficiency in the detection of new attacks not identified in its signature base, in addition to distributed attacks. It also suffers from the accuracy of identifying normal and malicious network traffic (false positives & false negatives), and on this basis we tried to introduce a model based on the behavioral approach using deep learning (Deep Learning) in particular recurrent neural networks, to remedy these weaknesses.

In the future, we want to develop and improve the proposed model, so that it can determine the type of attack in addition to directly processing the capture of network flows in real time.

Keywords : Computer Security, Intrusion Detection System, Machine learning, Deep Learning, Scenario Approach, Behavioral Approach , Recurrent Neural Networks, Kddcup'99.

Résumé :

Internet, a provoqué une grande révolution technologique en termes d'échange d'informations, de connaissances et de science, qui a conduit à l'inévitabilité de son utilisation dans la vie personnelle et professionnelle, et le sort des personnes est devenu l'otage de ce réseau. Cette nécessité a conduit certains à l'exploiter illégalement par espionnage, extorsion, sabotage, vol de données ... qui ont conduit à l'émergence de méthodes de protection telles que les systèmes Anti-virus, les pare-feu, les technologies de mot de passe et les systèmes de cryptage ... et d'autres moyens de protection.

Tous les systèmes de sécurité présentaient des lacunes et des failles qui serrent exploiter par les pirates informatiques. Cela a poussé les chercheurs à l'invention de systèmes de détection d'intrusion, qui ont corrigé un bon nombre des défauts des solutions précédentes. Il est à noter qu'il existe plusieurs types de systèmes de détection d'intrusions, dont certains sont basés sur l'approche par scénarios comme IDS Snort, et certains d'entre eux sont basés sur l'approche comportementale des utilisateurs et des applications.

Dans notre projet, nous avons installé & configuré le système de détection d'intrusion Snort, qui est considéré comme l'un des principaux systèmes dans le domaine de la détection d'intrusion, et nous avons installé le logiciel de gestion d'affichage graphique Splunk pour faciliter la lecture et l'analyse des alarmes et des journaux que Snort les générés.

Malgré sa puissance dans le domaine de la détection d'intrusions, il y a une carence dans la détection de nouvelles attaques non identifiées dans sa base de signatures, en plus des attaques distribuées. Il souffre également de la précision de l'identification des trafics réseaux normaux et malveillants (faux positives & faux négatives), et sur cette base, nous avons essayé d'introduire un modèle basé sur l'approche comportementale en utilisant l'apprentissage en profondeur (Deep Learning) en particulier les réseaux de neurones récurrentes, pour remédier à ces faiblesses.

À l'avenir, nous souhaitons développer et améliorer le modèle proposé, afin de lui pouvoir déterminer le type d'attaque en plus de traiter directement la capture des flux réseau en temps réel.

Mots clefs : Sécurité Informatique, Système de Détection d'Intrusion, Apprentissage Automatique, Apprentissage profond, Approche par Scénario, Approche Comportementale, Les réseaux de neurones récurrents, Kdd Cup'99, Snort.

إهداء :

أهدي هذا العمل المتواضع إلى والدي الغاليين

إلى عائلتي الكبيرة، إلى زوجتي، إلى بناتي أسماء، خديجة و فاطيمة الزهراء إلى ولدي محمد ياسين

كما أهدي هذا العمل المتواضع

إلى كل طاقم جامعة سعيدة من أساتذة ، عمال خاصة الأستاذة المشرفة و الأستاذ خاطر معمر

و إلى كل طلبة جامعة سعيدة خاصة المقيمين في الإقامة الجامعية الرياض

و إلى كل من ساهم من قريب أو بعيد في إنجاز هذا العمل المتواضع

الطالب : عبد القادر تراس

إهداء :

أهدي هذا العمل المتواضع إلى والدي الغاليين ، إخوتي وأخواتي

إلى عائلتي الصغيرة

كما أهدي هذا العمل المتواضع

إلى كل طاقم جامعة سعيدة من أساتذة ، عمال خاصة الأستاذة المشرفة و الأستاذ خاطر معمر

و إلى كل طلبة جامعة سعيدة خاصة المقيمين في الإقامة الجامعية الرياض

و إلى كل من ساهم من قريب أو بعيد في إنجاز هذا العمل المتواضع

الطالب : م . مقدم

تشكرات :

أولا و قبل كل شيئ الحمد لله الذي وفقنا و يسر لنا إنجاز هذا العمل ، الشكر موصول للوالدين اللذين لم يدخرا أي جهد منذ ولادتنا و إلى اليوم في سبيل تحصيلنا العلمي، الشكر الموصول إلى كل من ساهم في إنجاز العمل من قريب أو بعيد خاصة الأستاذة المحترمة المشرفة على العمل فايبة طالب بن ديمراد ، و إلى كل طاقم الجامعة من أساتذة و عمال و طلبة خاصة طاقم قسم الإعلام الآلي.

Sommaire

Liste des figures

Table d’abréviations

CPU : Central Processing Unit.

GPU : Graphics Processing Unit.

DDoS : Distributed Denial of Service attack.

DNS : Domain Name System

FN : Faux Négative.

FP : Faux Positive.

HIDS : Hote Intrusion Detection System.

IA : Intelligence Artificielle.

ICMP : Internet Control Message Protocol.

IDS : Intrusion Detection System.

IDWG : Intrusion DetectionWorking Group.

IEC : International Electrotechnical Commission.

IEEE : Institute of Electrical and Electronics Engineers.

IETF : Internet Engineering Task Force.

IP : Internet Protocole

IPS : Intrusion Prevesion System.

IPv4 : Internet Protocole version 4

IPv6 : Internet Protocole version 6

ISO : International Standardization Organization.

KDD : Knowledge Discovery in Databases.

LDAP : Lightweight Directory Access Protocol

NIDS : Network Intrusion Detection System.

RAM: Random Access Memory

SNMP : Sample Network Management Protocol.

SSH : Secure Shell.

SSL : Secure Sockets Layer.

TCP : Transport Control Protocol.

TN : True Negative.

TP : True Positive.

UDP : User Datagram Protocol.

VN : Vrai Négative.

VLBD : Very Large Data Base.

VP : Vrai Positive.

Introduction générale :

L'Internet est devenu la moyenne la plus primordiale et l'une des principales sources d'informations et d'échanges de données dans le monde de nos jours. Internet peut être considéré comme l'une des outils importants dans défirent secteurs, éducatif commercial social militaire scientifique, sanitaire, ... Elle attire de plus en plus d'internautes par les nombreux avantages et la diversité des services rendus accessibles. Ils peuvent ainsi bénéficier de communication rapides avec de minimum coût, partager des ressources de traitement et de stockage de grandes capacité (Cloud Computing), faciliter les échanges commerciaux et financiers (e-Commerce, e-Banking) et, plus généralement, partager et accéder à l'information. En effet, l'Internet doit être un milieu sécurisé et fiable.

La sécurité informatique est l'une des principales préoccupations des entreprises et des individus aujourd'hui. L'accroissement du nombre d'internautes rend l'internet plein des gens de bonnes et de mauvaises intentions. Ils peuvent exploiter les vulnérabilités des réseaux et des systèmes d'informations pour tenter d'accéder à des informations sensibles afin de les consulter, les modifier ou les détruire et donc le disfonctionnement du système. Ces menaces et ces attaques, deviennent un véritable problème pour les entreprises et les organisations. Notre dépendance croissante aux systèmes informatiques dans notre quotidien soulève inévitablement, la problématique est de sécurisé ces systèmes et de sécurisé l'information circulant.

Divers moyens ont été inventés pour interdire quelconques attaques tel que les serveurs d'authentification, l'antivirus, les pare-feu, les Proxy . . . etc., ces moyens ne suffisent pas à bloquer tous les types d'attaques qui violent à la confidentialité, l'intégrité ou la disponibilité. Pour rédimer à ce problème, un nouveau moyen de sécurité appelé systèmes de Détection d'intrusions (IDS), introduit par James Anderson en 1980 a dont l'objectif d'analyser le trafic réseau et la détection des attaques.

Les systèmes de détection d'intrusion sont généralement classés en deux catégories : les IDS qui utilisent l'approche par scénarios ceux qui cherchent à détecter les signatures et les IDS qui utilisent l'approche comportementale ceux qui cherchent à détecter les anomalies, afin d'augmenter le niveau de sécurité il y a des solutions qui combinent les deux approches en même temps.

En pratique, la plus part des IDS souffrent plus ou moins de deux problèmes, le nombre important de faux positifs et faux négatifs.

Notre mémoire s'inscrit dans les domaines de la sécurité informatique, système de détection d'intrusion, le Deep Learning. Nous avons réalisés une étude et mise en place de l'IDS Snort sous Linux, cet IDS repose sur l'approche par scénario, qui nous avons l'améliorer par l'approche comportementale basé sur les réseaux de neurones récurrents.

Chapitre 1 :
Sécurité Informatique

Chapitre 1 : La sécurité informatique

1.1 Introduction

La sécurisation du réseau informatique exige une approche de bout en bout et une prise en compte ferme des vulnérabilités et des mesures de protection associées. Bien que ces mesures ne puissent pas contrecarrer toutes les tentatives d'intrusion de réseau ou d'attaque du système, elles peuvent permettre aux ingénieurs de réseau d'éliminer certains problèmes généraux, réduire considérablement les dommages potentiels, et de détecter rapidement les brèches. Avec le nombre sans cesse croissant et la complexité des attaques, des approches vigilantes à la sécurité sont des préoccupations importantes des utilisateurs et des entreprises afin de préserver les points suivants [9] :

- L'intégrité : garantir que l'information ne sera modifiée que par les personnes autorisées moyennant des actions volontaires et légitimes. Pour assurer l'intégrité des données il faut pouvoir détecter les manipulations de données non autorisées.
- La confidentialité : Ce critère permet de rendre l'information intelligible par les personnes autorisées uniquement.
- La disponibilité : demande que l'information qui se trouve dans le système soit disponible aux personnes autorisées.
- Le non répudiation : permet de garantir qu'une transaction ne puisse être niée par un correspondant.
- L'authentification : permet de garantir à chacun des correspondants que son interlocuteur est bien celui qu'il croit être. Ainsi, ce critère assure l'identité des correspondants.

1.2 La sécurité informatique : [09]

La sécurité informatique c'est l'ensemble des outils et moyens mis en place pour éliminer la vulnérabilité d'un système contre des menaces intentionnelles ou accidentelles.

Une entreprise sécurisée, doit avoir une politique de sécurité complète et efficace. La mise en place d'un dispositif de sécurité requiert un coût monétaire, un temps de mise en place et présente une certaine complexité.

Les problèmes techniques actuels de sécurité informatique peuvent être classés en deux grandes catégories :

• Ceux qui concernent la sécurité de l'équipement proprement dit, serveur ou poste de travail, de son système d'exploitation et des données qu'il héberge ;

• ceux qui découlent directement ou indirectement de l'essor des réseaux (les paquets circulants sur le réseau), ce qui multiplie la quantité et la gravité des menaces.

Tout d'abord il est primordial de comprendre ce qu'on veut protéger. Les entreprises devraient avoir un ensemble de ressources. Chacun de ces derniers doit avoir une valeur relative attribuée dénotant son importance, tels que les routeurs, commutateurs, concentrateurs, serveurs, postes de travail, baies de stockage…etc. A cet effet le degré de sécurité d'une ressource dépend de sa valeur ou de son importance [09].Ensuite, il faut identifier les menaces potentielles pour chacun de ces éléments. Les menaces peuvent provenir de sources internes ou externes, humaines ou automatisées, intentionnelles ou accidentelles. [08]

1.2.1 La sécurité physique, la protection à l'intérieur : [09]

La politique de sécurité d'un système d'information de l'entreprise repose sur la sécurité physique. Il convient donc d'accorder un soin aux points suivants :

• Qualité de l'immeuble qui héberge les données et les traitements et les protège contre d'éventuelles intempéries, inondations, incendies ainsi que de possibles intrusions.

• Contrôles d'accès adéquats :
Un système de contrôle d'accès gère les entrées et les sorties aux locaux. Il se compose des éléments suivants :

– Un logiciel qui permet la gestion des habilitations de chaque employé,

– Un support d'identification de la personne qui accède (badge, empreinte, biométrie, téléphone, etc.),

– Des terminaux d'identification ou des serrures autonomes,

– Des équipements pour la gestion de l'accès physique aux bâtiments (par ex. des obstacles piétons, des barrières, des ventouses, etc.).

• Qualité de l'alimentation électrique redondante et climatisation adéquate.

• Certification adéquate du câblage du réseau local et des accès aux réseaux extérieurs.

• Pour l'utilisation de réseaux sans fil, placement exacte des bornes d'accès, réglage de leur puissance d'émission et contrôle des signaux en provenance et à destination de l'extérieur.

1.2.2 Sécurité au niveau logique : [10]

La sécurité au niveau logique concerne la sécurité des données de l'entreprise, des applications et des systèmes d'exploitation.

Ici, la notion de la sécurité est très sensible : les utilisateurs jouent un rôle principal. En effet, étant donné qu'ils sont connectés directement au système (manipulation des données, utilisation des applications et des services...), ils doivent être vigilants et sensibilisés au sujet des problèmes et des risque liés à la sécurité.

- La complexité des systèmes d'exploitation et leur mauvaise configuration présente une vulnérabilité sur le système.

- Les problèmes applicatives comme : la mauvaise conception des applications, le non-traitement des exceptions, les failles dans les langages de programmation peuvent engendrer beaucoup de problèmes influençant sur la sécurité et le fonctionnement du système.

- Les utilisateurs doivent disposer de profils dont chacun a un niveau d'accès bien défini par l'administrateur réseau : le profil de chaque groupe d'utilisateurs doit être défini par l'administrateur de la sécurité informatique, c'est-à-dire, l'ensemble des services, des applications et des ressources manipulables par le profil.

- La complexité des mots de passe est recommandée : pour protéger les données, chaque utilisateur disposant d'un profil d'accès doit impérativement choisir et utiliser un mot de passe très robustes, difficiles à retrouver à l'aide d'outils automatisés et à deviner par une tierce personne. Autrement, un intrus pourrait accéder aux serveurs suivant le niveau d'accès autorisé au profil piraté.

L'accès au web via le réseau local doit être très restreint : filtrage des sites web autorisés et fichiers téléchargés.

1.2.3 Sécurité au niveau réseau : [09]

La sécurité réseau et l'accès au réseau, relève beaucoup d'interrogations :

Qui a accès au réseau ?, Quel est son niveau d'accès une fois connecté au réseau (droits sur les fichiers et répertoires, privilèges sur les services ...) ? Que faire pour limiter l'accès à certaines ressources du réseau (serveurs par exemple) ?

Il serait très dangereux voire inimaginable d'autoriser l'accès aux ressources internes du réseau à tout le public (quel qu'il soit). Il existe plusieurs mécanismes pour répondre à ces questions:

Mise en place de serveur d'authentification comme par exemple le serveur **RADIUS** (Remote Authentication Dial-In User Service) qui est un protocole d'authentification standard, défini par un certain nombre de RFC.

Le fonctionnement de RADIUS est basé sur un système client/serveur. Il sert à définir les accès des utilisateurs distants à un réseau. Il s'agit du protocole de prédilection des fournisseurs d'accès à internet car il est relativement standard et propose des fonctionnalités de comptabilité permettant aux FAI de facturer précisément leurs clients.

Le protocole RADIUS repose principalement sur un serveur (le serveur RADIUS), relié à une base d'identification (base de données, annuaire LDAP, etc.) et un client RADIUS, appelé NAS (Network Access Server), faisant office d'intermédiaire entre l'utilisateur final et le serveur. L'ensemble des transactions entre le client RADIUS et le serveur RADIUS est chiffrée et authentifiée grâce à un secret partagé.

1.3 Evaluation de la sécurité d'un réseau : [w2]

Pour évaluer la sécurité du réseau nous devons disposer de nombreux outils, parmi ces derniers, il existe des outils payants et d'autres open-source. La difficulté se résume au choix de l'outil idéal pour une bonne évaluation, ainsi qu'au choix du moment opportun de son utilisation, cela avec un taux de confiance acceptable. L'évaluation de la sécurité du réseau repose sur quatre phases fondamentales : reconnaissance, énumération, évaluation et exploitation. La phase de reconnaissance concerne la détection des appareils du réseau par une sonde en direct. Pendant les phases d'énumération et d'évaluation, l'évaluateur de sécurité examine si un service ou une application tourne sur une hôte particulier et analyse

ses éventuelles failles. Dans la phase d'exploitation, il utilise une ou plusieurs vulnérabilités pour obtenir un certain niveau d'accès privilégié à l'hôte et utilise cet accès pour mieux exploiter l'hôte ou pour élever le privilège sur l'hôte ou le réseau.

L'outil Network Mapper (Nmap) peut être utilisé dans la phase de reconnaissance pour effectuer des balayages sur le réseau. Il permet de savoir quels hôtes sont actifs sur le réseau ? Quels systèmes d'exploitation et quels par feu ils utilisent ?...etc. Il est également utile pour la découverte des règles pare-feu ou ACL routeur via le fanion probe scanning ACK (acknowledgement) et autres techniques.

On peut utiliser Nmap dans les phases d'énumération et d'évaluation pour scanner les ports, lister les services et leurs numéros de version. Nmap est excellent pour approfondir mieux des résultats d'outils de scanning automatisés ou pour les vérifier. Nmap a été développé initialement pour l'environnement Unix. Nmap est open-source et disponible gratuitement sur divers sites, dont le principal est http://www.insecure.org/nmap.

1.4 Les failles de sécurité dans un réseau : [09]

Ces dernières années ont vu un fulgurant progrès dans le domaine de la technologie. Nous parlons alors de mode connecté, où les gens, les entreprises, les organismes, les pays et même les objets sont de plus en plus connectés entre eux. Ce qui rend les systèmes informatiques de plus en plus vulnérables.

Une faille peut être considérée comme un trou, un dysfonctionnement ou une insuffisance dans le dispositif de sécurité exploitée par les pirates. Ainsi, les attaques informatiques sont par conséquent de plus en plus fréquentes. Le problème de la sécurité persiste à tous les niveaux et devient un enjeu fondamental.

On distinct deux sorte de failles : passives et actives.

Les failles actives : c'est des failles déclenchées à partir de l'intérieure de l'entreprise (par utilisateurs, applications ou services). Elle peut résulter de plusieurs facteurs, tels que le non-respect de la politique de sécurité définie dans l'entreprise, comme l'installation de programmes non autorisés sur les hôtes, la violation de droits d'accès aux ressources, erreurs de manipulations lors du traitement de certains fichiers, par exemple un fichiers de configuration des services ou un fichier système.

Les failles passives : c'est des failles déclenchées à partir de l'extérieure de l'entreprise. Elles peuvent résulter de plusieurs actions telles que l'envoi de programmes malveillants (Cheval de Troie, …etc) vers les hôtes victimes du réseau d'entreprise. L'objectif est de fournir des informations au pirate afin de passer à l'attaque.

L'exploitation d'une faille de la sécurité informatique détectée, constitue le début de la véritable intrusion réseau.

Pour mieux se protéger, il est primordial de savoir à quoi s'attendre, et donc de connaître les attaques les plus courantes. Voici dans ce qui suit une liste non-exhaustive d'attaques connues.

1.4.1 Les attaques DDoS ou attaques par déni de service : [10]

Le déni de service (ou DoS Denial of Service) est une attaque qui vise à rendre une application ou un service informatique incapable de répondre aux requêtes de ses utilisateurs. Les serveurs de messagerie peuvent être victimes de ces attaques. Le déni de service distribué (DDoS pour Distributed DoS) est une attaque de DoS émise depuis plusieurs sources distinctes. Ce type d'attaque est très difficile à bloquer, car il est impossible de différencier une vraie requête d'une requête de DDos. L'attaque par DDos utilise très souvent une multitude de PC zombies infectés par des backdoors exploités à distance par un pirate afin d'atteindre une cible unique. L'attaque par déni de service peut aussi avoir pour but de lancer un autre type d'attaque.

1.4.2 Le Man-in-the-Middle attaques ou MitM : [09]

Le MitM est un type d'attaque ou le pirate se glisser dans les échanges entre un serveur et un client, il en existe plusieurs :

- **Le détournement de session** : un pirate détourne une session entre un client de confiance et un serveur réseau. Il vole l'identifiant de session valide du client de confiance. Le serveur continue la session, croyant que c'est toujours le client de confiance auquel il a affaire.
- **L'usurpation** (**Spoofing**): consiste à se faire passer pour quelqu'un d'autre. Nous trouvons :

- l'usurpation de l'adresse IP (IP Spoofing) : le pirate peut utiliser l'adresse IP de la victime pour convaincre un système que c'est un client fiable et connu.
- L'usurpation de l'adresse E-mail (Spam) : après la réception d'un E-mail, nous pouvons lire l'adresse de l'expéditeur mais il est possible de changer cette adresse. Dans une entreprise, un pirate peut envoyer à la victime (employé) un e-mail en usurpant l'adresse de son supérieur (exemple adresse de son Directeur).
- L'usurpation Web : c'est le principe du phishing détaillé ci-dessous.

- **Le replay** : le pirate intercepte et enregistre d'anciens messages et tente plus tard de les envoyer, se faisant passer pour quelqu'un de confiance.
- **Interception de trafic :** c'est une attaque qui atteint la confidentialité. Elle a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis. Il peut s'agir d'une personne, d'un programme ou d'un ordinateur, dans le but de capturer des données sur un réseau. [12].
- **Attaque de l'intercepteur :** cette attaque consiste à intercepter les échanges entre deux parties, comme un utilisateur et un serveur Web, à l'insu de la victime qui croit avoir établi une connexion directe et sécurisée avec un site Web. L'attaque de l'intercepteur permet de surveiller les communications, de réacheminer le trafic, de modifier l'information, d'installer des malicieux et d'obtenir des renseignements nominatifs ou de l'information sensible. [12]
- **Attaque de l'enregistreur de frappe :** c'est une attaque qui utilise un keylogger ou enregistreur de frappe (un logiciel espion *spyware* ou un matériel espion), il enregistre les informations frappées au clavier sous certaines conditions, puis les transmet via les réseaux. Le keylogger intercepte le code d'une touche clavier entre le moment où l'utilisateur effectue l'action et l'exécution de cette action par l'application concernée. Les enregistreurs de frappes permettent notamment d'enregistrer les codes secrets et mots de passe des sites visités, d'enregistrer les URLs visitées, les courriers électronique consultés ou envoyés, les fichiers ouverts ou encore de créer une vidéo permettant de retracer toute l'activité de l'ordinateur. [13]

1.4.3 Le drive-by downloads ou téléchargement furtif : [09]

L'attaque par téléchargement frauduleux est une méthode de diffusion des logiciels malveillants. Le pirate insère un virus sur une page d'un site web non sécurisé et infecte les machines de ceux qui le visitent, et qui présentent des failles de sécurité.

1.4.4 Les attaques par mot de passe : [09]

Tenter de trouver un mot de passe est souvent bien plus facile qu'il nous semble. Différentes méthodes existent. Dans certaines méthodes, il suffit de surveiller le trafic d'un réseau, d'une manière active ou non. Un mot de passe intercepté qui est chiffré peut être déchiffré, plus ou moins facilement, suivant le niveau de sécurité de l'algorithme de chiffrement utilisé. D'autres méthodes ont recoure à l'ingénierie sociale, à l'enregistreur de frappe…etc. Toutes ces méthodes sont décrites ci-dessous :

- **Par dictionnaire :** c'est une méthode qui consiste à tester une série de mots de passe potentiels, les unes à la suite des autres afin de trouver un mot de passe utilisé pour le chiffrement qui soit contenu dans le dictionnaire.
- **Par force brute** : Le pirate utilise un processus automatisé qui teste toutes les combinaisons possibles des caractères qu'il a fixé au début (minuscules, majuscules, chiffres...). La longueur maximale du mot de passe constitue aussi une information essentielle au pirate, le craquage du mot de passe prend entre quelque second et plusieurs années en fonction de la langueur et la complexité du mot de passe. Cette opération est bien plus efficace que l'utilisation du dictionnaire.

1.4.5 Injection SQL : [09]

Ce genre d'attaque affectant les sites web exploitant des bases de données : le pirate exécute une requête SQL sur la base de données via les données entrantes du client au serveur. Un morceau de code SQL est inséré directement dans un formulaire. Ensuite, le pirate peut insérer, mettre à jour ou supprimer les données comme bon lui semble, C'est l'un des types d'attaques les plus répandus et menaçants, car il peut potentiellement être utilisé pour nuire à une application Web qui utilise une base de données SQL et même envoyer des commandes au système d'exploitation.

1.4.6 Les logiciels malveillants ou malwares : [09]

Un malware est un logiciel indésirable installé dans un système sans accord. Il en existe différents types, en voici quelques-uns :

- ***Les macro-virus*** : ils infectent des applications comme Microsoft Word ou Excel en s'attachant à la séquence d'initialisation de l'application.
- ***Les infecteurs de fichiers*** : ils s'attachent à des fichiers exécutables.
- ***Les infecteurs de systèmes*** : ils infectent les disques durs.
- ***Les virus polymorphes*** : ils se cachent dans divers cycles de chiffrement.
- ***Les virus furtifs*** : ils prennent le contrôle de certaines fonctions du système pour se dissimuler.
- ***Les chevaux de Troie*** : les chevaux de Troie ou Trojan Horse sont des logiciels qui vont s'installer à l'intérieur d'une machine. Ils vont utiliser un port existant, ou le plus souvent ouvrir une porte de sortie (backdoor) sur cette dernière et qui vont permettre à un pirate de se connecter dessus, ce qui lui donne un accès directe.
- ***Les bombes logiques*** : ces des dispositifs programmés dont le déclenchement s'effectue à un moment déterminé en exploitant la date du système, le lancement d'une commande, ou n'importe quel appel au système.
- ***Les vers :*** les vers sont des fichiers exécutables (des programmes à part entière). La méthode de propagation est différente d'un virus. C'est plus simple qu'un virus, car il n'a pas besoin d'un mécanisme compliqué pour se reproduire. Un ver utilise tous les moyens réseaux pour se propager (Les mails, le web, certains services Internet comme le FTP, …).
- ***Les injecteurs*** : il peut s'agir d'une simple amorce, dont le rôle va être de télécharger un programme plus important qui sera le vrai virus.
- ***Les ransomwares*** : c'est un type de logiciel malveillant qui crypte les données d'une machine et exige une rançon à la victime contre son déchiffrement.
- ***Porte dérobée*** : Une porte dérobée (backdoor) est un logiciel de communication caché, installé par exemple par un virus ou par un cheval de Troie, qui donne à un pirate accès à la machine victime.

1.4.7 Le crypto jacking : [W3]

Les pirates introduisent des programmes malveillants pour altérer les systèmes et les ressources d'une machine et pour exploiter la crypto-monnaie en arrière-plan et gagner de l'argent.

1.4.8 Les intrusions sur les objets connectés : [25]

Un objet non connecté était invulnérable aux attaques informatiques. En le connectant il est exposé à de nouvelle menaces. Les objets connectés exposent principalement les utilisateurs à deux types de risques :

- l'utilisation commerciale des données personnelles et les atteintes à la vie privée : une des conséquences de ce monde de réseau et de communication est que l'utilisateur laisse de traces numériques. Il s'agit de garantir l'anonymat des données collectées par ses appareils;
- le piratage : dès lors que se connecter à internet devient une fonction intégrante d'objets, les concepteurs de ces équipements doivent faire face aux risques des « cybers attaques ».

1.4.9 Les attaques géopolitiques :

Certains pays infligent des attaques informatiques contre d'autres, menées par des organismes gouvernementaux pour les déstabiliser, les intimider ou voler certaines technologies de pointe dans un cadre d'espionnage industriel.

1.4.10 Les cross-site Scripting (XSS) : [09]

L'attaquant à pour cible le client, plutôt que le serveur, son but est de détourner le fonctionnement de son navigateur, moyennant un script écrit dans n'importe quel langage (JavaScript, Java, Flash...).

Plus précisément, le pirate développe un script, il soumet ensuite ce script à un serveur. Si ce dernier présente des vulnérabilités vis-à-vis de ce type d'attaque, il acceptera le script et le déposera dans sa base de données. Si à un moment donné, un client se connecte au serveur et demande une page qui affichera les entrées de la base de données, le script sera automatiquement envoyé et exécuté sur le navigateur du client. Cela n'est pas sans conséquences, puisque le script du pirate pourra accomplir des actions telles que : rediriger

discrètement la navigation vers un site malveillant qui pourra injecter du code dans la page visitée. Il convient de ne pas ignorer les risques induits par ce genre de failles, par exemple si la page détournée comporte des demandes d'authentification avec mot de passe ou des transactions financières.

1.4.11 Les attaques de phishing : [W4]

Le phishing, c'est cette célèbre annonce qui surgit en vous disant que vous avez gagné un million d'euros, ou cet étrange email que vous recevez de votre banque, vous demandant de saisir votre identifiant… ceci vous incite à télécharger par vous-même des malwares qui voleront vos informations personnelles et confidentielles.

1.4.12 **Les attaques cyber-physiques :**

Ce sont des attaques visant surtout des systèmes de transport, de réseaux électriques, d'usines ou d'installations de traitement de l'eau par exemple.

1.4.13 **Les attaques contre les appareils et dossiers médicaux :**

Dans le milieu médical, il existe de nombreuses technologies de pointe, et des données très secrètes. Les cybercriminels s'intéressent de plus en plus à ce domaine, ils peuvent nuire à la vie privée des patients. Ceci est d'autant plus désastreux s'ils ont accès à des appareils médicaux.

1.4.14 Intrusion : [10]

Intrusion correspond à l'action de pénétrer dans un lieu, au sein d'un milieu ou auprès d'un groupe sans y avoir été invité, en informatique c'est l'opération qui consiste à accéder, sans autorisation, aux données d'un système informatique ou d'un réseau, en contournant ou en désamorçant les outils de sécurité existants.

1.5 Comment sécuriser un réseau informatique : [W5].

Nous pensons souvent que les attaques informatiques qui ciblent les entreprises sont très complexes, en réalité, la plus grande menace de cyber sécurité pour ces entreprises vient de leurs propres employés. Pour atténuer ce type d'attaques, il faut :

- Mettre à jour les politiques et procédures de cyber sécurité et éduquer les employés.
- Contrôler les accès internet de l'entreprise.
- Contrôler les accès wi-fi.
- Faire des sauvegardes sur des serveurs extérieurs à l'entreprise.
- Attention aux applications Cloud personnelles.
- Etre prêt à faire face à une attaque informatique.
- Maintenir à jour le parc informatique et l'homogénéiser.
- Responsabiliser et former le personnel de l'entreprise.

1.6 Outils de sécurité informatique :

1.6.1 Partitionnement et protéger les frontières du réseau avec pare-feu : [09]

C'est le contrôle des points de connexion au monde extérieur (Internet). Toutes les entreprises sont connectées à Internet. La séparation entre un réseau d'entreprise « le côté de confiance » et Internet "le côté non fiable" est un élément de sécurité critique.

Un pare-feu (de l'***anglais firewall***) est un logiciel et/ou un matériel qui permet de contrôler le trafic réseau entrant et sortant de l'entreprise. Il autorise ou bloque une partie du trafic suivant des règles de sécurité prédéfinies appelées les listes de contrôle d'accès ACLs (Access Control List). La difficulté pour les pare-feu est la distinction entre le trafic légitime et illégitime.

Le pare-feu n'empêche pas un pirate d'utiliser une connexion autorisée pour attaquer le système. Ne protège pas contre une attaque venant du réseau intérieur (qui ne le traverse pas).

1.6.2 Un serveur proxy (serveur mandataire) : [09]

C'est l'ensemble de processus qui permet d'éliminer la connexion directe entre les employés de l'entreprise et l'extérieur. Les entreprises utilisent les proxys pour permettre à des machines de leur réseau d'utiliser Internet sans risque et sans que les utilisateurs externes ne soient capables d'accéder à ce réseau.

1.6.3 DMZ (Demilitarized zone) : [09]

Vu la nécessité d'accéder au réseau d'entreprise, depuis l'extérieur (serveur web, serveur de messagerie, serveur FTP public, etc.), il est indispensable de créer une nouvelle interface vers un réseau à part, accessible aussi bien du réseau interne que de l'extérieur, sans diminuer la sécurité de l'entreprise. On parle ainsi de « zone démilitarisée » (notée DMZ pour Demilitarized Zone) pour désigner cette zone isolée hébergeant des applications mises à disposition du public. La DMZ fait ainsi office de « zone tampon » entre le réseau à protéger et le réseau publique.

1.6.4 Antivirus : [09]

Les antivirus sont des logiciels de détection et suppression des virus et malwares d'une machine ou du flux analysé. La qualité d'un antivirus dépend de sa capacité à identifier les nouveaux virus et de mettre à jour sa signature virale, mais ils ne protègent pas le système 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.

1.6.5 Système de détection d'intrusion (IDS) : [10]

Outre la mise en œuvre des DMZ, firewalls, scanneurs de failles et antivirus, il est indispensable de nos jours d'installer un Système de détection d'Intrusion (en anglais IDS : **Intrusion detection System**). C'est un outil qui permet de détecter les attaques/intrusions du réseau ou de la machine sur laquelle il est installé. Les IDS alertent l'administrateur en cas de faille de sécurité, de violation de règles ou d'autres problèmes susceptibles de compromettre le système.

Les systèmes de détection d'intrusions contrôlent et analysent les activités d'un réseau (NIDS), ou d'une machine (HIDS), analysent ses vulnérabilités et ses configurations, et examinent l'intégrité des fichiers. Ils peuvent reconnaitre des schémas d'attaque habituelle. Et ce après l'analyse des comportements particuliers et suivent les infractions de règles par les utilisateurs. Les systèmes industriels de détection et de prévention d'intrusions peuvent réagir à des attaques découvertes. Dans le chapitre suivant, nous allons apporter de plus amples détails sur les systèmes de détection d'intrusion.

1.7 Conclusion

Aujourd'hui, avec le degré d'évolutions actuelles, les systèmes informatiques continuent à s'ingérer davantage dans notre quotidien. Le développement de ces systèmes s'accompagne des défis technologiques tels que la portabilité, l'évolutivité, l'autonomie, la réactivité,…etc. Parmi ces défis, la sécurité reste un facteur majeur en raison de la dématérialisation croissante de l'information et des risques liés à la complexité de ces systèmes.

Il existe plusieurs techniques de protection contre les attaques qui gardent les bases de la sécurité : confidentialité, intégrité, authentification, disponibilité. Mais malgré toutes ces techniques utilisées pour empêcher les attaques, un système informatique n'est jamais totalement sûr.

Afin d'améliorer les mécanismes de sécurité dans les réseaux informatiques un IDS est une nécessité impérative pour forcer la sécurité réseau.

Chapitre 2 :
les Systèmes de Détection d'Intrusion IDS

Chapitre 2 : les systèmes de détection d'intrusion IDS

2.1 Introduction :

La méthode idéale de protéger un réseau ou un système informatique est de découvrir les attaques et les menaces et de les empêcher avant même qu'elles ne se produisent et ne puissent causer des dégâts. Les systèmes de détection d'intrusion sont parmi les outils de sécurité les plus récents. Ainsi beaucoup d'experts de la sécurité informatique font appel aux IDS. Ce dernier est l'objet même de notre thématique de recherche. Dans ce chapitre, nous allons développer en détail les systèmes de détection d'intrusion et comment ils fonctionnent.

2.2 Historique :

Les IDS ont évolués rapidement. Le concept de système de détection d'intrusions a été introduit en 1980 par James Anderson afin d'améliorer la sécurité informatique et la capacité de surveillance. Il a ensuite été complété par le premier modèle de détection d'intrusions établit par Denning Doro thyenen 1987. Depuis 1988, il y a eu l'apparition de plusieurs prototypes. [27]

2.3 Définition d'un système de détection d'intrusion IDS :

Un IDS (Intrusion Detection System) est tout équipement, méthode ou ressource qui permet de capturer le trafic réseau et d'examiner les activités d'un réseau ou d'un hôte. Cela afin de détecter toutes activités suspectes ou anormales et de pouvoir les signaler au responsable de sécurité informatique [28].

Les activités suspectes se résument à :

- ✓ Tentatives de compromettre la confidentialité, l'intégrité, la disponibilité…etc.
- ✓ Eviter des mécanismes de sécurité établis dans le réseau ou l'hôte.

Un modèle général qui englobe et standardise la structure d'un système de détection d'intrusion a été proposé par le groupe IDWG (Intrusion Detection Working Group) de l'IETF et a longtemps été leur centre d'intérêt. Ce modèle comprend [29] :

1- un senseur (collecteur),
2- analyseur,
3- manager (administrateur).

Les principales tâches d'un IDS sont : [22]

- Découvrir les tentatives de découvertes du réseau.
- Découvrir dans certains cas, la réussite de l'attaque.
- Découvrir le Dénis de Service.
- Découvrir le degré d'infection du système informatique et les zones réseaux touchées.
- Détecter les machines infectées.
- Avertir d'une façon centrale de toutes les attaques.
- Réagir aux intrusions et corriger les problèmes éventuels.

2.4 Présentation d'un système de détection d'intrusion IDS :

Un IDS est constitué principalement d'un capteur (sniffer) combiné avec un moteur qui analyse le trafic selon des règles. Ces dernières donnent les caractéristiques des trafics réseau à avertir. Un IDS assure les fonctionnalités de contrôle réseau et réagit selon la nature des échanges réseau.

2.4.1 Architecture d'un IDS :

Il existe plusieurs schémas pour décrire les constituants d'un système de détection d'intrusions. Parmi eux, les travaux d'Intrusions Detection exchange format Working Group (IDWG) de l'Internet Engineering Task Force(IETF) comme base de départ, car il résulte d'un large consensus parmi les intervenants du domaine. [01]

Les travaux du groupe IDWG a pour objectif, la détermination d'un standard de communication entre certains composants d'un système de détection d'intrusions. La figure ci-dessous illustre ce modèle. [29]

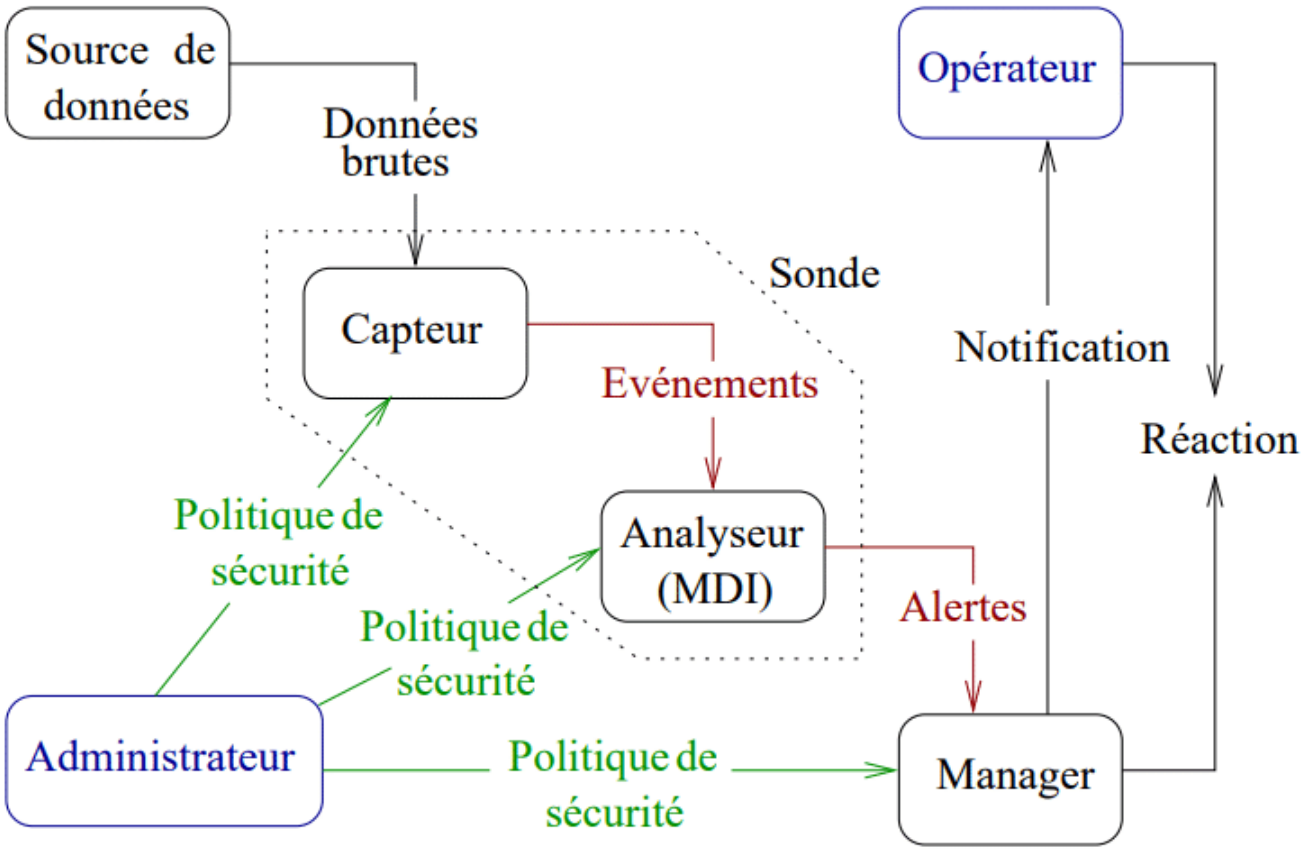


Figure 2.1 : Modèle générique de la détection d'intrusion proposée par l'IDWG [01]

Un IDS, selon L'architecture IDWG se compose d'un nombre de capteurs qui envoient des événements à un analyseur. Les capteurs couplés avec un analyseur forment une sonde, cette dernière envoie des alertes vers un manager qui la notifie à un opérateur humain. [01]

Les différents éléments de cette architecture : [29]

- **Administrateur :** C'est le responsable globale qui définit la politique de sécurité de l'organisation et, par conséquent, des décisions concernant le déploiement et la configuration de l'IDS. Il peut s'agir ou non de la même personne que l'opérateur de l'IDS. Dans certaines organisations, l'administrateur est associé aux groupes d'administration du réseau ou des systèmes. Dans d'autres organisations, c'est une position indépendante.
- **Alerte :** c'est un message envoyé par l'analyseur au gestionnaire, s'il découvre une intrusion dans un flux réseau.
- **Analyseur** : celui qui signale les activités non autorisées ou indésirables ou les événements qui pourraient avoir un intérêt pour l'administrateur de sécurité en

analysant les données recueillies par le capteur. Dans la plupart des IDSs existants, le capteur et l'analyseur font partie d'un même composant.

- **Capteur** : celui qui collecte les données brutes émanant d'une source de données du réseau, selon une fréquence de collecte qui varie suivant la configuration de l'IDS, et transmettre des évènements à l'analyseur.
- **Evénement :** c'est toute occurrence détectée par un capteur dans la source des données et qui peut donner lieu à une alerte. (une attaque).
- **Manager :** Le composant ou processus d'identification à partir duquel l'opérateur gère les différents composants du système d'identification. Les fonctions de gestion incluent généralement (mais sans s'y limiter) :
    1. la configuration du capteur et de l'analyseur.
    2. la gestion de la notification d'événements.
    3. la consolidation des données et la gestion des rapports.
- **Notification :** c'est la méthode par laquelle le manager met au courant l'opérateur de l'occurrence d'alerte.
- **Opérateur :** c'est la personne chargée de l'utilisation du manager associé à l'IDS. Elle propose ou décide de la réaction à apporter en cas d'alerte.
- **Réaction :** mesures actives ou passive prises suite à la détection d'une attaque, pour la bloquer ou pour corriger ses conséquences.
- **Source de données :** Elle représente les informations brutes utilisées par l'IDS pour détecter les activités non autorisées. Elle inclue les paquets bruts du réseau, les journaux d'audit du système d'exploitation, les journaux d'audit d'applications et les données de contrôle générées par le système.
- **Données brutes :** c'est les éléments de la source de données qui sont identifiés par le capteur ou l'analyseur et ayant un intérêt pour l'opérateur. Par exemple, les entrées des fichiers journaux du système d'exploitation montrant un utilisateur essayant d'accéder à des fichiers qui ne lui sont pas autorisés, les fichiers journaux d'application montrant des échecs de connexion persistants...etc.

Dans ce modèle qui représente le processus complet de la détection d'intrusion ainsi que l'acheminement des données au sein d'un IDS. L'administrateur configure les différents composants (capteur(s), analyseurs(s), manager(s)) selon une politique de sécurité bien définie. Les capteurs accèdent aux données brutes, les filtrent et les formatent pour ne renvoyer que les événements intéressants à un analyseur. L'analyseur utilise ces événements

pour décider de la présence ou non d'une intrusion et envoie dans le cas échéant une alerte au manager, qui notifie l'opérateur humain, une réaction éventuelle peut être menée automatiquement par le manager ou manuellement par l'opérateur. [02]

2.4.2 Vocabulaire de la détection d'intrusions :

La détection d'intrusions utilise un lexique bien définis qui ne se trouve pas dans le modèle précédent et qui est comme suit :

- **Attaque ou intrusion** : action de violation de la politique de sécurité.
- **Audit de sécurité** : Selon Joras (1996), l'audit est une démarche spécifique d'investigation et d'évaluation à partir d'un référentiel, incluant un diagnostic et conduisant éventuellement à des recommandations. [30]
- **Détection d'intrusions** : c'est l'analyse des informations collectées par les mécanismes d'audit de sécurité, à la recherche d'éventuelles attaques sur les systèmes informatiques. Les méthodes de détection d'intrusion diffèrent sur la manière d'analyser le journal d'audits. [27]
- **Faux positif** : alerte en l'absence d'une attaque.
- **Faux négatif** : absence d'une alerte en présence d'une attaque.
- **Vulnérabilité** : faille de conception, d'implémentation ou de configuration d'un système d'information ou d'un matériel. [09]
- **Log (trace d'audit)** : Le journal d'évènements est un fichier texte qui enregistre de manière chronologique les évènements exécutés par un hôte ou une application informatique. C'est le fichier à analyser.
- **Scénario** : suite formée des étapes élémentaires d'une attaque.
- **Signature** : suites des étapes observables d'une attaque, utilisée par certains analyseurs pour chercher dans les activités des entités, des traces de scénarios d'attaques connus.
- **Corrélation** : c'est l'interprétation conceptuelle de plusieurs actes (alertes) pointant à leur assigner une meilleure sémantique et à diminuer le nombre des d'alertes.

2.4.3 Caractéristiques d'un système de détection d'intrusions : [10]

Les caractéristiques suivantes sont indispensables dans un IDS :

- Fonctionnement en permanence avec une supervision manuelle minimale, émettre une alerte en cas d'intrusion.
- Etre tolérant aux pannes dans le sens où il doit récupérer après une défaillance ou une réinitialisation de la machine.
- Résister aux tentatives de corruption, c'est-à-dire, il doit pouvoir détecter s'il a subit lui-même une modification indésirable.
- Utiliser un minimum de ressources pour implémenter une politique de sécurité spécifique d'un réseau.
- Etre facilement configurable pour implémenter une politique de sécurité spécifique d'un réseau.
- S'adapter au cours du temps aux changements du système surveillé et du comportement des utilisateurs.
- Etre difficile à tromper.

Comme la taille des réseaux a tendance à croitre, on peut ajouter les caractéristiques suivantes :

- Etre accessible.
- Etre robuste, c'est-à-dire l'arrêt d'un composant ne doit pas entrainer une défaillance totale.

2.4.4 Emplacement d'un IDS :

Le déploiement d'un IDS n'est pas suffisant pour assurer la sécurité. Ainsi il faut toujours réfléchir aux actes ordinaires, particulièrement l'isolation des systèmes utilisant internet (DMZ), la robustesse de la politique des mots de passe et la gestion des mises à jour applicatives.

Il est important de prendre en considération la bonne configuration de l'IDS. Par exemple avec un réseau sous Linux, la configuration destinée aux systèmes Windows s'avèrent inutiles pour la détection des tentatives d'intrusion. Donc il est nécessaire de faire une configuration en tenant compte du système d'exploitation, des applications et du matériel utilisé.

De plus, l'emplacement des sondes lors du déploiement d'un IDS est très important et permet de répondre à certains problèmes, selon la position de la sonde. Le schéma ci-dessous montre les positions possibles.

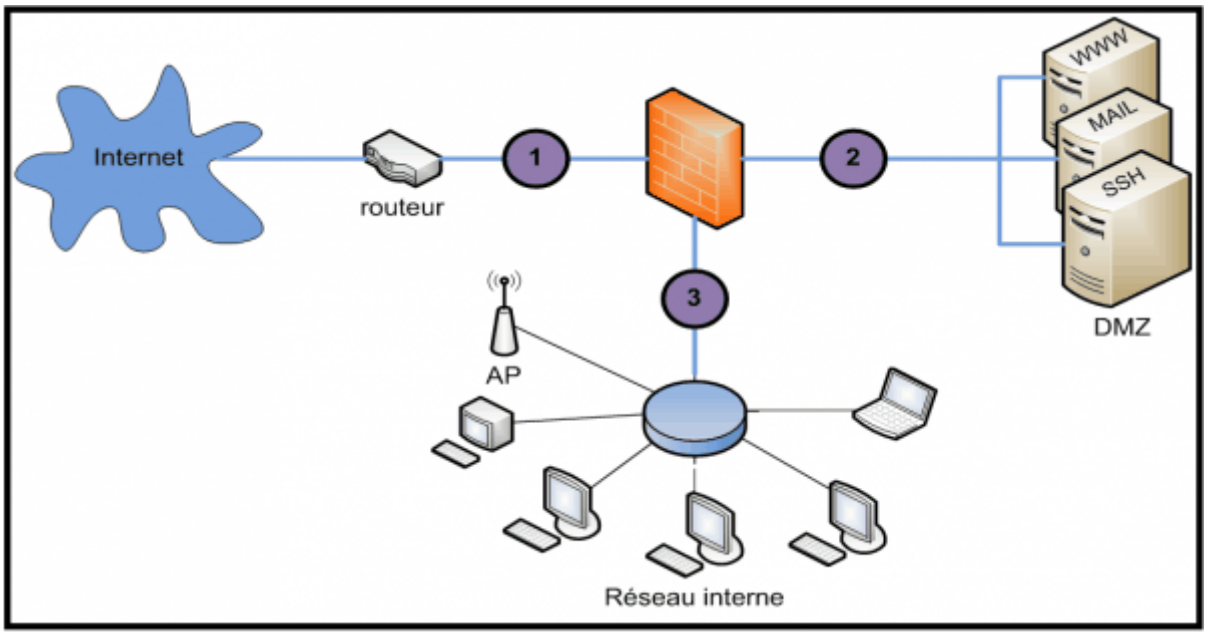


Figure 2.2 : Emplacements possible d'un système de détection d'intrusions [W6].

Position 1 :

Le trafic émanant de l'internet vers et le réseau interne et la DMZ est analysé par l'IDS, ce qui rend le fichier journal trop volumineux, difficilement consultable et par la suite très complexe à analyser. De plus, le trafic entre le réseau interne et la DMZ est invisible pour l'IDS, cette position expose l'IDS au pirate.

Position 2 :

Le flux réseau entre la DMZ et internet ou le réseau interne est analysé. De plus, placer une sonde à cet emplacement nous permet de détecter les attaques non filtré par le pare-feu et donc minimisé le trafic réseau à analyser. Cependant, le trafic entre le réseau interne et l'internet est invisible pour l'IDS.

Position 3 :

Mettre la sonde à cet emplacement nous permet de détecter les tentatives d'intrusion issues du réseau local (émanant de lui) et celles voulant atteindre ce même réseau local. Ce choix nous est apparu judicieux, car la majorité des attaques proviennent de l'intérieur (Troyen, Virus, …etc.).

En effet, il est souvent préférable d'installer la sonde après le firewall du côté interne. Ainsi seuls les trafics permis par le firewall sont analysés et donc nous obtenons une forte diminution en matière de charge des sondes de l'IDS.

De plus, le choix du matériel a également une grande influence car le trafic réseau doit être reçu par les sondes IDS pour l'analyser quel que soit le poste destinataire, donc, la machine qui héberge l'IDS doit être puissante en terme de processeur, de mémoire et d'espace de stockage. Du coup, le choix d'une telle machine doit assurer cette fonctionnalité. Cependant, d'autres problématiques peuvent surgir, selon la nature du réseau à surveiller, particulièrement lorsque les flux sont cryptés, on doit aussi réfléchir comment protéger, les sondes et les alertes qu'elles génèrent contre les attaques.

2.4.5 Classification des systèmes de détection d'intrusions : [02]

La classification des systèmes de détection d'intrusions d'après Hervé Debar, Marc Dacier et Andreas Wespi [31] sont :

- La source des données à analyser.
- Le lieu de l'analyse des données.
- La fréquence de l'analyse.
- Le comportement en cas d'attaque détectée.
- La méthode de détection utilisée.

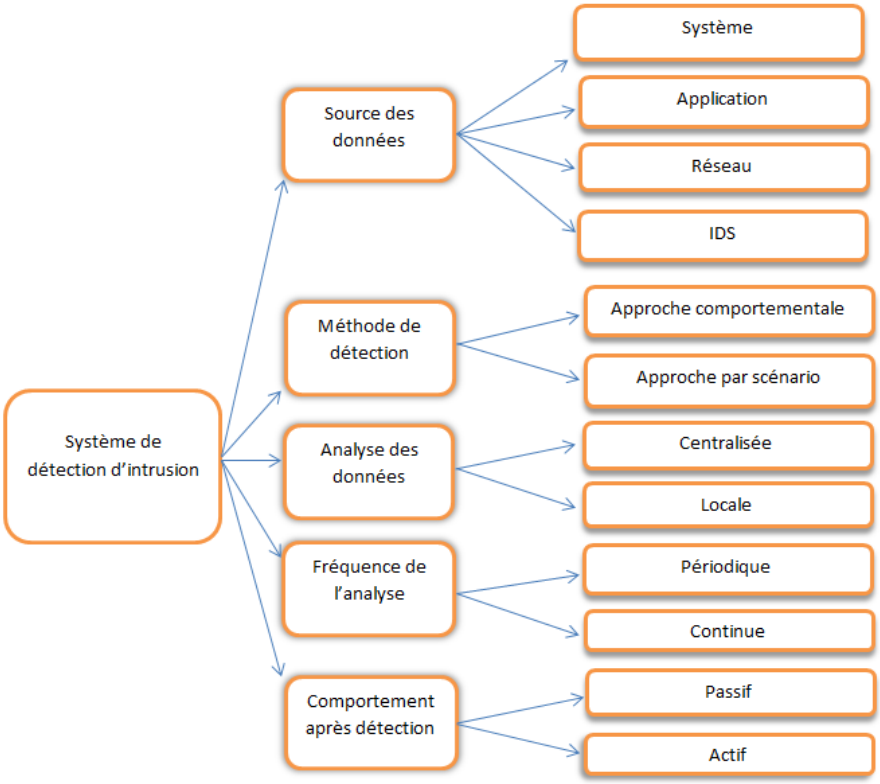


Figure 2.3 : Classification des systèmes de détection d'intrusions. [02]

2.4.5.1 Source des données à analyser :

Les sources possibles de données à analyser sont une caractéristique essentielle des systèmes de détection d'intrusions puisque ces données constituent la matière première du processus de détection. Les données proviennent soit de logs générés par le système d'exploitation, soit de logs applicatifs, soit d'informations provenant du réseau, soit encore d'alertes générées par d'autres IDS. [02]

A- Source d'information système (OS) : [02]

Un système d'exploitation fournit généralement plusieurs sources d'information :

- **Commandes systèmes** : presque tous les systèmes d'exploitation fournissent des commandes pour avoir un « instantané » de ce qui se passe.
- **Accounting** : l'accounting fournit de l'information sur l'usage des ressources partagées par les utilisateurs (temps processeur, mémoire, espace disque, débit réseau, applications lancées, ...).
- **Audit de sécurité** : tous les systèmes d'exploitation modernes proposent ce service pour fournir des événements système, les associer à des utilisateurs et assurer leur collecte dans un fichier d'audit. Nous pouvons donc potentiellement disposer

d'informations sur tout ce que font (ou ont fait) les utilisateurs : accès en lecture à un fichier, exécution d'une application, etc. Par exemple, l'audit BSM [Sun] représente les appels systèmes produits par les programmes qui s'exécutent sur un système.

B- Source d'information applicative : [02]

Les applications peuvent également constituer une source d'information pour les IDS. Les capteurs applicatifs sont de deux natures :

- **Capteur interne** : le filtrage sur les activités de l'application est alors exécuté par le code de l'application.
- **Capteur externe** : le filtrage se fait à l'extérieur de l'application. Plusieurs méthodes sont utilisées : un processus externe peut filtrer les logs produits par l'application ou bien l'exécution de l'application peut être interceptée (au niveau de ses appels de librairies ou d'un proxy applicatif).

Prendre ses informations directement au niveau de l'application présente plusieurs avantages. Premièrement, les données interceptées ont réellement été reçues par l'application.

Il est donc difficile d'introduire une désynchronisation entre ce que voit passer le capteur applicatif et ce que reçoit l'application contrairement à ce qu'il peut se passer avec les capteurs réseau. Ensuite, cette source d'information est généralement de plus haut niveau que les sources système et réseau. Cela permet donc de filtrer des événements qui ont une sémantique plus riche. Finalement, si l'on prend l'exemple d'une connexion web chiffrée par SSL, un capteur réseau ne verra passer que des données pseudo-aléatoires tandis qu'un capteur associé au serveur web pourra analyser le texte en clair de la requête. [02]

C- Source d'information réseau : [02]

Des dispositifs matériels ou logiciels (snifer) permettent de capturer le trafic réseau. Cette source d'information est particulièrement adaptée lorsqu'il s'agit de rechercher les attaques en déni de service qui se passent au niveau réseau ou les tentatives de pénétration à distance. Le processus d'interception des paquets peut être rendu quasiment invisible pour l'attaquant car nous pouvons utiliser une machine dédiée juste reliée à un brin du réseau, configurée

pour ne répondre à aucune sollicitation extérieure et dont personne ne soupçonnera l'existence.

Néanmoins, il est difficile de garantir l'origine réelle de l'attaque détectée car il est facile de masquer son identité en modifiant les paquets réseau.

D- Source d'information basée sur d'autre IDS : [02]

Une autre source d'information, souvent de plus haut niveau que les précédentes, peut être exploitée. Il s'agit des alertes remontées par des analyseurs provenant d'un autre IDS. Chaque alerte synthétise déjà un ou plusieurs événements intéressants du point de vue de la sécurité.

Elles peuvent être utilisées par un IDS pour déclencher une analyse plus fine à la suite d'une indication d'attaque potentielle. De surcroît, en corrélant plusieurs alertes, nous pouvons par fois détecter une intrusion complexe de plus haut niveau. Il y aura alors génération d'une nouvelle alerte plus synthétique que l'on qualifie de méta-alerte.

2.4.5.2 Les méthodes de détection :

La manipulation d'un système de détection d'intrusion 'd'une façon idéale' exige la compréhension de son fonctionnement interne et les méthodes que ce dernier utilise pour détecter les tentatives d'intrusions.

En effet, plusieurs méthodes sont utilisées pour la détection des intrusions aux systèmes d'information. Les IDS disposent principalement de deux approches différentes afin de découvrir les intrusions.

A- Approche comportementale : [2]

Cette approche s'appuie particulièrement sur l'étude du comportement passé des utilisateurs, des services ou des applications. Elle permet d'établir une certaine corrélation entre les différents événements sur un système et ainsi découvrir des anomalies qui mènent à la détection de tentatives intrusives sur ce dernier. En effet, cette approche définie un comportement normal du système et considère toute déviation par rapport à ce comportement comme étant suspecte.

Ainsi, chaque flux et son comportement ordinaire doivent être affirmés. L'IDS se chargera d'envoyer une alerte, si un flux anormal est détecté, sans pouvoir déterminer la gravité de l'attaque potentielle.

Le déploiement de ce type d'IDS nécessite une période d'apprentissage pendant laquelle l'outil va apprendre le comportement "normal" des flux applicatifs présents sur son réseau. Donc il est obligatoire de modéliser des comportements normaux pour le système à protéger en définissant un profil système qualifié comme profil normal. Par ailleurs, cette approche permet de détecter des attaques non connues, contrairement à l'approche par scénario présenté ci-dessous, puisqu'elle utilise des techniques heuristiques, probabilistes ou statistiques pour la détection des tentatives intrusives. Cependant, les IDS, qui adoptent cette approche, sont dotés de fonctionnalités d'adaptation et d'apprentissage afin de définir un comportement normal du système. Du coup la mise en place de cette approche nécessite plus de temps pour arriver à couvrir la totalité des systèmes à protéger.

Nous pouvons distinguer deux catégories de profils : le profil construit par apprentissage et celui spécifiant une politique de sécurité (policy-based)

A.1 Profil construit par apprentissage :

Les méthodes de construire des profils par apprentissage, les plus utilisés sont les suivantes :

- **Méthode statistique** : le profil est calculé à partir de variables considérées comme aléatoires et échantillonnées à intervalles réguliers. Ces variables peuvent être le temps processeur utilisé, la durée et l'heure des connexions, etc. Un modèle statistique est alors utilisé pour construire la distribution de chaque variable et pour mesurer, au travers d'une grandeur synthétique, le taux de déviation entre un comportement courant et le comportement passé.
- **Système expert** : ici, c'est une base de règles qui décrit statistiquement le profil de l'utilisateur au vu de ses précédentes activités. Son comportement courant est comparé aux règles, à la recherche d'une anomalie. La base de règles est rafraîchie régulièrement.
- **Réseaux de neurones** : la technique consiste à apprendre à un réseau de neurones le comportement de l'entité à surveiller. Par la suite, lorsqu'on lui fournira en

entrée les actions courantes effectuées par l'entité, il devra décider de leur normalité.

- **Analyse de signatures** : l'approche proposée par Forrest [32] s'inspire de l'immunologie biologique et met en œuvre des algorithmes de pattern matching. Il s'agit de construire un modèle de comportement normal des services réseaux Unix. Le modèle consiste en un ensemble de courtes séquences d'appels système représentatifs de l'exécution normale du service considéré. Des séquences d'appels étrangères à cet ensemble sont alors considérées comme l'exploitation potentielle d'une faille du service.

Pour toutes ces méthodes, le comportement de référence utilisé pour l'apprentissage étant rarement exhaustif, ce qui augmente les risques de fausses alertes (faux positifs). De plus, lors de cette phase d'apprentissage, les attaques commises seront considérées comme normales (risque de faux négatifs).

A.2 profils spécifiant une politique de sécurité (policy-based) :

Pour ce type d'IDS (policy-based), la phase d'apprentissage n'existe plus. Leur comportement de référence est spécifié par une politique de sécurité, la détection d'une intrusion intervient chaque fois que la politique est violée. Le profil décrit le comportement légal attendu et non un comportement observé empiriquement.

D'après Ko et Redmond [33], le profil est une politique de sécurité qui précise quel groupe d'utilisateurs est autorisé à interférer avec quel groupe de données. Zimmermann, Mé et Bidan [26] déterminent que le profil est une politique de sécurité représentée par un graphe qui définit les flux d'informations autorisés entre objets du système. Dans ces deux cas, toute séquence d'appels systèmes, pour être conforme à la politique de sécurité (profil), doit vérifier un certain prédicat. Par opposition, une suite d'appels systèmes qui ne vérifie pas ce prédicat contient une violation de la politique de sécurité.

L'approche comportementale possède un certain nombre d'avantages et d'inconvénients :

Les avantages :

- Elle n'exige pas des connaissances préalables sur les attaques.
- Elle permet la détection de la mauvaise utilisation des privilèges.
- Elle permet de créer des informations qui peuvent être utilisées pour développer des signatures pour l'analyse basée connaissance.

Les inconvénients :

- Cette approche produit un taux élevé des alertes de type faux positif en raison des comportements imprévisibles des utilisateurs et des réseaux.
- Cette approche nécessite des phases d'apprentissage pour caractériser les profils de comportement normaux à l'exception des IDS dits policy-based.

Les alertes génériques par cette approche ne sont pas significatives.

Les IDS comportementaux sont apparus récemment après les IDS à signature et ne bénéficient pas encore de leur maturité. Ainsi, l'utilisation de tels IDS peut s'avérer délicate dans le sens où les alertes remontées contiendront un nombre important de fausses alertes donc des faux positifs. Cette difficulté peut être résolue en généralisant la déclaration des trafics réseau mais, cet acte peut entrainer une transparence de l'IDS face à la détection de certaines attaques.

En général, les IDS mélangent les différentes méthodes de détection par scénario ou par anomalie en proposant des combinaisons entre la recherche des motifs, l'analyse protocolaire et la détection d'anomalies.

B Approche par scénario : [2]

L'approche par scénario nécessite une connaissance préalable des techniques d'attaques utilisées par les pirates (profil d'une attaque), pour pouvoir en déduire des scénarios typiques. Cette approche consiste à examiner dans un flux réseau les empreintes d'attaques connues (base de signature), à l'instar des anti-virus. Elle ne tient pas compte des activités passées sur le système et s'appuie, sur la base d'un ensemble de caractères permettant d'identifier une activité intrusive ou en étudiant la conformité protocolaire des paquets, pour détecter les tentatives d'intrusions et produire des alertes.

Une signature est habituellement déterminée comme une suite d'événements et d'exigences relatant une tentative d'intrusion. L'inspection est alors fondée sur le concept de "pattern matching" (analyse de chaînes de caractères présente dans le paquet, à la recherche de correspondance au sein d'une base de connaissance). Si une attaque est détectée, une alerte peut être déclenchée.

Ainsi, il existe des méthodes de recherche de motifs statiques ou dynamiques, qui sont à ce jour les suivantes :

B.1 Système expert : le système expert contient une base de règles qui décrit les attaques. Les événements d'audit sont traduits en des faits qui sont interprétables par le système expert. Son moteur d'interférence décide alors si une attaque répertoriée s'est produiteou non.

B.2 Analyse de signatures : c'est la méthode la plus utilisée actuellement. Des signatures d'attaques sont fournies à des niveaux sémantiques variés selon les outils (de la suite d'appels système aux commandes passées par l'utilisateur en passant par les paquets réseau). De nombreux algorithmes ou méthodes sont utilisés pour localiser ces signatures connues dans les traces d'audit. Ces signatures sont toujours exprimées sous une forme proche des traces d'audit. Dans le cas des NIDS, les algorithmes de recherche de motifs utilisés permettent d'obtenir de bonnes performances en vitesse de traitement mais génèrent de nombreuses fausses alertes donc faux positifs.

B.3 Automates à états finis : de nombreux IDS utilisent des automates à états finis pour coder le scénario de reconnaissance de l'attaque. Cela permet d'exprimer des signatures complexes et comportant plusieurs étapes. On passe d'un état initial sûr (confirmé) à un état final attaqué via des états intermédiaires. Chaque transition entre états est déclenchée par des conditions sur les événements remontés par les capteurs.

En général, cette approche est basée particulièrement sur une analyse protocolaire pour examiner la conformité (par rapport aux RFC) des flux qui circulent sur le réseau.

Ainsi, ce procédé présente un inconvénient capital qui s'interprète par le niveau d'exactitude des signatures d'attaques. De plus, le niveau de découverte des attaques dépend principalement de la façon dont elles sont gérées ainsi que des mises à jour de la base des signatures et du nombre des règles présentes pour gérer les alertes. Cette approche permet seulement la détection des attaques qui sont connues au préalable, parmi ses avantages nous trouvons que l'analyse basée sur la connaissance est très efficaces pour la détection des intrusions avec un taux très négligeable de faux négatifs et que ses alertes sont significatives.

2.4.5.3 Localisation de l'analyse des données : [02]

Nous pouvons également faire une distinction entre les IDS en se basant sur la localisation réelle de l'analyse des données :

- **Analyse centralisée** : certains IDS ont une architecture multi-capteurs (ou multisondes). Ils centralisent les événements (ou alertes) pour analyse au sein d'une seule machine. L'intérêt principal de cette architecture est de faciliter la corrélation entre événements. Par contre, la charge des calculs (effectués sur le système central) ainsi que la charge réseau (due à la collecte des événements ou des alertes) peuvent être lourdes et risquent de constituer un goulet d'étranglement.
- **Analyse locale** : l'analyse du flot d'événements est effectuée au plus près de la source de données (généralement en local sur chaque machine disposant d'un capteur). Nous minimisons ainsi le trafic réseau et chaque analyseur séparé dispose de la même puissance de calcul. En contrepartie, il est impossible de croiser des événements qui sont traités séparément et nous risquons de passer à côté de certaines attaques distribuées.

2.4.5.4 Fréquence de l'analyse : [02]

Une autre caractéristique des systèmes de détection d'intrusions est leur fréquence d'utilisation :

- **Périodique** : certains systèmes de détection d'intrusions analysent périodiquement les fichiers d'audit à la recherche d'une éventuelle intrusion ou anomalie passée. Cela peut être suffisant dans des contextes peu sensibles (nous ferons alors une analyse journalière, par exemple).
- **Continue** : la plupart des systèmes de détection d'intrusions récents effectue leur analyse des fichiers d'audit ou des paquets réseau de manière continue afin de proposer une détection en quasi temps-réel. Cela est nécessaire dans des contextes sensibles (confidentialité) et/ou commerciaux (confidentialité, disponibilité). C'est toutefois un processus coûteux en temps de calcul car il faut analyser à la volée tout ce qui se passe sur le système.

2.4.5.5 Comportement après détection : [02]

Une autre façon de classer les systèmes de détection d'intrusions consiste à les classer par type de réaction lorsqu'une attaque est détectée :

- **passif** : la plupart des systèmes de détection d'intrusions réagit passivement lorsqu'une attaque est détectée. En effet, les IDS génèrent une alerte et notifient l'administrateur système à travers divers moyens comme l'envoie d'un e-mail, message dans une console, voire même par alarme sonore, …. c'est lui qui devra décider les mesures nécessaires.
- **actif** : additivement à la notification de l'alerte à l'opérateur, certains systèmes de détection d'intrusions peuvent, en plus réagir automatiquement en prenant des mesures pour stopper l'attaque en cours. (fermer des sessions, bloquer des ports, des connexions suspectes ou même, pour une attaque externe, reconfigurer le pare-feu pour qu'il refuse tout ce qui vient du site incriminé). Cette réaction automatique est potentiellement dangereuse car il peut mener à des dénis de service provoqués par l'IDS. Un attaquant peut tromper l'IDS en usurpant des adresses du réseau local qui seront alors considérées comme la source de l'attaque par l'IDS. Une réaction facultative est préférable en laissant l'opérateur humain prendre la décision finale.

2.4.6 Les différents types d'IDS :

La multitude des attaques minées par les pirates et les failles exploitées par ces attaques sur les différents niveaux des systèmes d'informations justifie l'existence de plusieurs types d'IDS. En effet, pour affirmer le bon fonctionnement du système d'information, l'évaluation des risques encourus par un système d'information et la classification des risques influence le choix du type du système de détection d'intrusion à mettre en place.

Il existe plusieurs types d'IDS qui répondent à des attaques définies. Ils sont classés selon leurs utilisations et les fonctions qu'ils doivent réaliser. Nous allons citer ci-dessous les différents types d'IDS et nous allons ensuite détailler leurs caractéristiques.

2.4.6.1 Les systèmes de détection d'intrusions de type hôte (HIDS) : [4]

Un HIDS est plus attaché à une machine (hôte) qu'à son réseau. Il permet d'analyser, non seulement, le flux réseau, mais aussi l'activité qui se passent sur l'hôte. Le but de cet IDS est de garantir l'intégrité des données et d'analyser le flux de la machine et de ces journaux, nous prenons comme exemple les HIDS Samhain ou Tripwire. Ce type d'IDS présente une limite non négligeable. En effet, il doit être installé sur un système sain pour pouvoir examiner l'intégrité des données. Par conséquent, un HIDS devient faible dans le cas où il est installé sur une machine déjà infecté.

L'avantage de ce type d'IDS est qu'il est possible de constater immédiatement l'impact d'une attaque et donc de mieux réagir. Grâce à la quantité d'informations étudiées, il est possible d'observer les activités qui se déroulent sur la machine avec exactitude et d'optimiser le système en fonction des activités observées.

Voici quelque exemple des systèmes de détection d'intrusion de type HIDS : Tripwire, Watch, DragonSquire, Triger, Security Manager, …

Toutefois il existe plusieurs type d'HIDS tel que :

A- Détection d'Intrusions basée sur le noyau KIDS : [04]

Le KIDS ou Kernel IDS qui appartient à la famille des HIDS. Mais, sa fonctionnalité primordiale est de détecter les intrusions au niveau noyau. L'utilisation d'un KIDS s'avère parfois essentielle, selon le niveau de sécurité à apporter pour une machine. Il permet de reconnaître les caractéristiques des failles présentes sur un système, mais aussi d'interdire le système d'exploitation d'exécuter des appels systèmes qui peuvent s'avérer dangereux ou qui visent à compromettre le système.

B- Détection d'Intrusions basée sur une application [04]

Les IDS basés sur les applications sont un sous-groupe des IDS hôtes. Ils contrôlent l'interaction entre un utilisateur et une application en ajoutant des fichiers log afin de fournir de plus amples informations sur les activités d'une application particulière. Puisque vous opérez entre un utilisateur et un programme, il est facile de filtrer tout comportement notable. Un ABIDS (système de détection d'Intrusion basée sur une application en anglais Application-Based Intrusion Detection System) se situe au niveau de la communication entre un utilisateur et l'application surveillée.

L'avantage de cet IDS est qu'il lui est possible de détecter et d'empêcher des commandes particulières dont l'utilisateur pourrait se servir avec le programme et de surveiller chaque transaction entre l'utilisateur et l'application. De plus, les données sont décodées dans un contexte connu, leur analyse est donc plus fine et précise.

Par contre, du fait que cet IDS n'agit pas au niveau du noyau, la sécurité assurée est plus faible, notamment en ce qui concerne les attaques de type "Cheval de Troie". De plus, les fichiers log générés par ce type d'IDS sont des cibles faciles pour les attaquants et ne sont pas aussi sûrs que les traces d'audit du système.

Ce type d'IDS est utile pour surveiller l'activité d'une application très sensible, mais son utilisation s'effectue en général en association avec un HIDS. Il faudra dans ce cas contrôler le taux d'utilisation CPU des IDS afin de ne pas compromettre les performances de la machine.

2.4.6.2 Les systèmes de détection d'intrusions réseau (NIDS) : [04]

Ce type d'IDS capture avec des sondes situés aux endroits bien déterminés, tout le trafic réseau, analyse de façon passive les flux et découvre en temps réel les intrusions pour produire des alertes. Ces alertes sont envoyées à une console sécurisée, pour une analyse et un traitement éventuel. C'est le type d'IDS le plus utilisé.

Plusieurs endroits peuvent être l'emplacement d'un éventuel capteur, en fonction des nécessités (quel trafic nous désirons capturer). Les capteurs peuvent être installés après ou avant le pare-feu, ou encore, sont parfois installés à l'entrée de zones sensibles (parcs de serveurs, données confidentielles…), de façon à contrôler le trafic en direction de cette zone.

Les avantages des NIDS : les capteurs peuvent être bien sécurisés puisqu'ils filtrent le trafic réseau d'une façon invisible, pour une surveillance discrète du réseau, les attaques de type scans sont simplement détectées.

Les NIDS montrent certains inconvénients. En effet, la probabilité de faux négatifs (attaques non détectées comme telles) est dominante et il est difficile d'examiner le réseau entier. De plus, ils doivent principalement fonctionner d'une façon invisible d'où une difficulté de l'analyse des paquets. Pour finir, contrairement aux IDS basés sur l'hôte, ils ne voient pas l'impact d'une attaque.

Voici quelques exemples de NIDS : Snort, NetRanger, Dragon, NFR, ISSRealSecure.

2.4.6.3 Système de Détection d'Intrusion de Nœud Réseau (NNIDS) : [04]

Le NNIDS fonctionne comme le NIDS classiques. Il analyse les paquets du trafic réseau destinés à un nœud du réseau. Le NNIDS ne fonctionne pas en mode "promiscuous", ce qui n'est pas le cas du NIDS. Il n'analyse que les paquets à destination d'une adresse (hôte) ou d'une plage d'adresse (réseau). Les performances de l'ensemble sont améliorées car tous les paquets ne sont pas analysés.

Cet IDS n'est pas encore très répandu, mais il est de plus en plus utilisé pour étudier le comportement de nœuds sensibles d'un réseau.

2.4.6.4 Les systèmes de détection d'intrusions hybrides :

Ce type d'IDS est utilisé dans un environnement décentralisé. Il centralise les informations en provenance de plusieurs captures (senseurs) sur le réseau. L'objectif de ce type d'IDS est d'avoir une vision globale sur les composants d'un système informatique en permettant un contrôle centralisé en matières d'alertes d'intrusions remontées par les NIDS et les HIDS présents sur l'architecture du réseau surveillé. Nous citons comme exemple d'IDS hybride : Prelude, OSSIM.

2.5 Les limites d'un IDS :

Parmi ces faiblesses nous trouvons : [05]

- Nombreux faux positifs et faux négatifs.
- Configuration complexe et longue.
- Pas de connaissance de la plate-forme.
  1. De ses vulnérabilités.
  2. Du contexte métier.
- Les attaques applicatives sont difficilement détectables.
  1. Injection SQL.
  2. Exploitation de CGI mal conçus.
- Des évènements difficilement détectables.
  1. Scans lents / distribués.
  2. Canaux cachés / tunnels.
- Pollution des IDS.
  1. Consommation des ressources de l'IDS.
  2. Perte de paquets.
  3. Déni de service contre l'IDS / l'opérateur.
  4. Une attaque réelle peut passer inaperçue.
- Attaque contre l'IDS lui-même.
- Ils ne peuvent pas compenser les trous de sécurité dans les protocoles réseaux.
- Ils ne peuvent pas compenser des manques significatifs dans la stratégie de sécurité, la politique de sécurité ou l'architecture de sécurité.

- **Quelques outils :**

Il existe sur le marché plusieurs IDS, nous citons quelques-uns :

Snort : [17]

Snort est un système de détection d'intrusion réseau NIDS, peut être configuré comme NIPS système de prévention des intrusions réseau, basé sur des règles. Dans le chapitre suivant nous allons aborder cet IDS en détail, comment le télécharger, l'installer et le configurer.

ISS RealSecure : [W1]

Internet Security Systems (ISS) fournit ISS RealSecure IDS, c'est un IDS qui permet la détection d'intrusions intégrée. ISS RealSecure IDS utilise une approche appuyée sur des normes pour examiner les entrées de trafic réseau et journaux d'accueil des méthodes connues et probables des attaquants. ISS RealSecure IDS intègre avec de nombreuses applications de réseau et de gestion des systèmes.

RealSecure combine en un seul agent trois fonctionnalités essentielles :

- Un moteur de détection d'intrusions.
- Un firewall personnel.
- Un module de contrôle d'applications et de communications.

Enterasys DRAGON : [06]

Edité par Enterasys Networks, c'est un IDS considérée comme unleader du marché du fait ses capacités d'analyse,ses performances et ses facultés d'adaptation à tout type d'environnement.

Les solutions Dragon sont constituées de sondes NIDS (Network Sensor), d'agents HIDS (Host Sensor) et d'un système de management qui assure les fonctions d'exploitation des événements de la suite Dragon.

Le Network Sensor est un NIDS disponible en version logicielle ou en boitier dédié. Depuis la version 6, Enterasys décline les Appliances et les logiciels en trois versions selon la bande passante à analyser. Les versions matérielles sont appuyées à leur équivalent logiciel.

Le Host Sensor est un agent HIDS qui découvre les attaques contre le système sur lequel il est installé en examinant les journaux systèmes et d'audit ainsi qu'en utilisant des outils d'analyse de signatures.

Dragon détecte les intrusions sur l'ensemble de l'infrastructure informatique ou qu'elles se produisent et permet d'avoir ainsi une visibilité globale sur le système d'information. Cela permet notamment d'optimiser les ressources humaines nécessaires à l'analyse des journaux issus des différents firewalls ou serveur Web en fédérant tous ces journaux au niveau d'une console Dragon unique qui analysera automatiquement les données afférentes.

2.6 Conclusions :

Dans ce chapitre nous avons vu les systèmes de détection d'intrusions (IDS) leurs fonctionnements et leurs capacités. Les IDS sont habituellement fiables, ce qui explique qu'ils sont souvent intégrés dans les solutions de sécurité informatique. Les avantages qu'ils montrent face aux autres outils de sécurités les favorisent, mais d'un autre côté cela n'empêche pas que les meilleurs IDS présentent aussi des lacunes et quelques inconvénients. Nous saisissons donc bien qu'ils sont indispensables mais ne peuvent pas se passer de l'utilisation d'autres outils de sécurité visant à remplir leurs défauts. Nous allons voir dans le chapitre III comment réussir une bonne configuration de ces derniers afin de mieux sécuriser le réseau.

Chapitre 3 :
Mise en place d'un IDS 'Snort'

Chapitre 3 : Mise en place d'un IDS 'Snort'

3.1 Introduction :

Dans ce chapitre nous allons choisir l'environnement de travail (système d'exploitation, IDS, machine physique qui héberge l'IDS), puis nous allons présenter les différentes étapes d'installation et de configuration en détail de l'IDS choisi.

3.2 Environnement de travail :

3.2.1 Machine hébergeant l'IDS : machine LENOVO, processeur Intel Core I3-3110M CPU@ 2.40 Ghz×4, mémoire vive RAM de 8 Go, disque dure 500 Go (5 Go minimum demandé).

3.2.2 Système d'exploitation : parmi les distributions de linux nous avons choisi la distribution Ubuntu 18.04.4 LTS 64 bit (*Long Terme Support*), qui est la dérivée de Debian. C'est un système d'exploitation version serveur en open source et gratuit, universel, complet, et facile à installer.

3.2.3 Le choix de l'IDS :

Dans notre projet de fin d'étude nous avons choisi SNORT comme IDS à mettre en place pour les raisons suivants :

- Snort est un IDS fourni en open source.
- L'un des plus populaires dans le monde de la détection d'intrusion.
- Possède une communauté importante qui a largement contribué à son succès.
- Très grande base de signatures enrichies par la communauté des utilisateurs de Snort.

Snort est un IDS inventé par Martin Roesch en 1998. Il est disponible en open source sous licence GNU (Le projet GNU est une initiative de collaboration pour le développement du logiciel libre, lancée par Richard Stallman). Son code source est accessible et modifiable à partir de l'URL : «http://www.snort.org». Il est donc gratuit et facile à se procurer. En plus de sa gratuité, son avantage est qu'il dispose d'une très grosse base de signatures réalisée par la communauté de ses utilisateurs, sans oublier les développeurs à plein temps qui travaillent sur ce logiciel. Il faut compter un bon millier de programmeurs expérimentés qui revoient et testent le code sans cesse. Ils mettent ainsi à l'épreuve un nombre impressionnant de fonctionnalités, de stratégies, et de jeux de règles. C'est la garantie d'obtenir rapidement des mises à jour de la base dès qu'une nouvelle menace est signalée. Il a été conçu à l'origine pour

le système linux mais il a également été porté sous d'autres systèmes d'exploitation tels que Windows. [07]

Snort est un système de détection d'intrusions réseau, capable d'effectuer l'analyse et la journalisation du trafic en temps réel, l'analyse de protocole, la recherche de contenu, il utilise des capteurs pour détecter les attaques CGI, les scans tels que des débordements de tampons système, des scans de ports furtifs, des scans SMB, des tentatives d'identification d'OS, grâce à l'analyse des signatures qui s'appuie sur une combinaison de règles et de préprocesseurs pour analyser du trafic réseau[7].

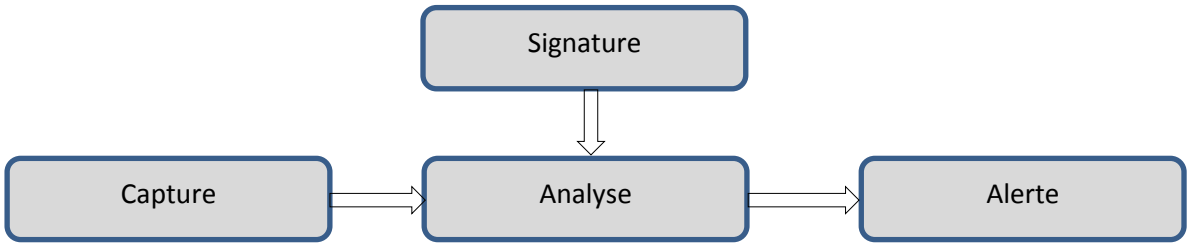


Figure 3.1 : Fonctionnement de Snort.

3.2.4 Emplacement de l'IDS :

Nous avons choisi la position n° 03 déjà détaillée dans le chapitre II section 2.4.4- Emplacement d'un IDS.

3.3 Présentation de Snort : [16]

Snort n'est pas difficile à utiliser, mais il y a beaucoup d'options de ligne de commande avec lesquelles il est possible de jouer et ce n'est pas toujours évident. Snort peut fonctionner en trois modes :

Le mode sniffé : Snort lit tout simplement les paquets du réseau et les affiche à l'administrateur en flux continu sur la console (écran).

Le mode « packet logger » : SNORT, dans ce mode enregistre le trafic réseau dans des fichiers log sur le disque.

Le mode détecteur d'intrusions réseau (NIDS) : SNORT dans ce mode, effectue la détection et l'analyse du trafic réseau. C'est le mode le plus complexe et le plus configurable.

3.4 Architecture de Snort : [24]

Snort est un IDS modulaire. Il est composé de :

A. **Décodeur de paquet** : en anglais (Packet Decoder) au démarrage, ce noyau charge un ensemble de règles, les compile, les optimise et les classe. Les paquets entrants sont décodés sur le fil par le décodeur de paquets, qui détermine quel protocole est utilisé pour un paquet donné et compare les données au comportement autorisé pour les paquets du protocole en question. Le décodeur de paquets peut générer ses propres alertes basées sur des en-têtes de protocole mal formées, des paquets trop longs, des options TCP inhabituelles ou incorrectes définies dans les en-têtes et d'autres comportements de ce type. Nous pouvons activer ou désactiver les alertes plus détaillées pour tous ces champs dans le fichier snort.conf.

B. **Les préprocesseurs** : en anglais (Pre-processor). Ce sont des plug-ins pour Snort qui permettent l'analyse des données entrantes de différentes manières. Si Snort est exécuté sans aucun préprocesseur spécifié dans le fichier de configuration snort.conf, il est probable que certaines attaques passent inaperçues. En effet, de nombreuses attaques modernes dépendent de l'écrasement des données dans des fragments qui se chevauchent par exemple ou des techniques d'évasion IDS délibérées comme placer une partie d'une demande d'application malveillante dans un paquet et le reste dans un autre paquet, et d'autres pratiques similaires.

C. **Moteur de détection** : en anglais (Detection Engine). C'est le composant le plus important de SNORT, celui qui prend les données du décodeur de paquets et des préprocesseurs (le cas échéant) et les compare aux règles du fichier de configuration snort.conf et les compare une à une avec le paquet de données. S'il y a conformité, le détecteur l'enregistre dans le fichier log et/ou génère une alerte.

D. **Système d'alerte et d'enregistrement des logs** : en anglais (The Alerting and Logging Components). Après la comparaison des règles Snort (snort.conf) au trafic réseau, le composant de journalisation archivera les paquets qui ont déclenché une ou plusieurs règles de Snort, tandis que le mécanisme d'alerte est utilisé pour informer l'administrateur de sécurité qu'une règle a été déclenchée. Comme les préprocesseurs,

ces fonctions sont appelées à partir du fichier de configuration snort.conf, où on peut spécifier les composants d'alerte et de journalisation que nous souhaitons activer.

E. **Plug-ins de sortie** : en anglais (Output Plug-Ins). Ces composants d'alerte et de journalisation, comme les préprocesseurs, sont des plug-ins, programmés selon l'API de Snort. La sélection de ces plug-ins de sortie est adaptée suivant les besoins de la société. Nous avons plusieurs choix pour envoyer ces alertes et où et comment enregistrer nos données de paquets. Nous pouvons par exemple envoyer des alertes via des fenêtres contextuelles SMB à un poste de travail Windows, les enregistrer dans un fichier journal, via une connexion réseau via des sockets UNIX ou via des interruptions SNMP. Les alertes peuvent également être stockées dans une base de données SQL. Certains systèmes tiers avertiront un administrateur système d'alertes IDS, ou même les enverront à un téléphone portable via des messages texte SMS.

3.5 Règles Snort : [16]

Snort utilise un langage de description de règles simple et léger, flexible et puissant. Il y a un certain nombre de directives simples à retenir lors du développement de règles Snort.

Les règles Snort sont divisées en deux sections logiques, l'en-tête de règle et les options de règle. L'en-tête de la règle contient l'action, le protocole, les adresses IP et les masques de réseau source et de destination de la règle, ainsi que les informations sur les ports source et de destination. La section d'option de règle contient des messages d'alerte et des informations sur les parties du paquet à inspecter pour déterminer si l'action de la règle doit être prise.

Format d'une règle :

Entête							Options
Action	Protocole	Adresse IP source	Port source	->	Adresse IP destination	Port destination	(msg : 'outside finger attempt' ;)

3.5.1 Entête de la règle (Rules Headers) :

3.5.1.1 Action : L'action de la règle indique à Snort ce qu'il doit faire lorsqu'il trouve un paquet qui correspond aux critères de la règle. Il y a trois actions disponibles dans Snort : alert, log, pass. Si nous exécutons Snort en mode inline, nous disposons d'options supplémentaires qui incluent drop, reject et sdrop.

- **alert** : générer une alerte, puis enregistrer le paquet.
- **log** : enregistre le paquet.

- **pass** : ignorer le paquet.
- **drop** : bloquer et enregistrer le paquet.
- **reject** : bloquer le paquet, l'enregistrer, puis envoyer une reset TCP si le protocole est TCP ou un message de port ICMP inaccessible si le protocole est UDP.
- **sdrop** : bloquer le paquet, mais ne l'enregistre pas.

3.5.1.2 Protocole (Protocols) :

Il existe quatre protocoles que Snort analyse actuellement pour détecter les comportements suspects: TCP, UDP, ICMP et IP. À l'avenir, il pourrait y en avoir plus, comme ARP, IGRP, GRE, OSPF, RIP, IPX, etc.

3.5.1.3 IP Adresse (IP Addresses) :

Cette partie traite de l'adresse IP et des informations de port. Le mot-clé ***any*** peut être utilisé pour définir n'importe quelle adresse. Snort ne peut pas faire la recherche avec le nom d'hôte pour ce champ. Une adresse est composée de : Adresse IP/le masque de réseau.

3.5.1.4 Numéros de port (Port Numbers) :

Le numéro de port peut être spécifié de plusieurs façons, y compris les ports. Les ports statiques sont indiqués par un numéro de port unique, (111 le port mapper, 23 le port Telnet ou 80 pour http, etc.). Nous pouvons utiliser l'opérateur ***ranges*** pour définir une plage de n° de port. Il existe un autre opérateur ***negation*** (!), cet opérateur peut être appliqué à n'importe quel autre type de règles à l'exception d'***any*** *exemple log tcp any any -> 192.168.1.0/24 !6000:6010*.

3.5.1.5 Opérateur de direction (The Direction Operator) :

Cet opérateur indique l'orientation, ou la direction du trafic auquel s'applique la règle. L'adresse IP et les numéros de port sur le côté gauche de l'opérateur de direction sont considérés comme le trafic provenant de l'hôte source, et les informations d'adresse et de port sur le côté droit de l'opérateur sont l'hôte de destination. Il existe également un opérateur bidirectionnel (<>). Cela indique à Snort de considérer les paires adresse/port dans l'orientation source ou destination. Ceci est pratique pour enregistrer / analyser les deux côtés d'une conversation.

3.5.2 Option de la règle (Rule Options) :

Les options de règles constituent le cœur du moteur de détection d'intrusions de Snort, combinant facilité d'utilisation, puissance et flexibilité. Toutes les options de règle Snort sont séparées les unes des autres à l'aide du caractère point-virgule (;). Les mots-clés des options de règle sont séparés de leurs arguments par le caractère deux-points (:).
Il existe quatre grandes catégories d'options de règles.

- ***General :*** cette option fournit des informations sur la règle qui n'ont aucun effet lors de la détection, nous trouvons les mots clés suivant : msg, reference, … voir les détails en annexe 1.
- ***Payload*** : cette option cherche toutes les données utiles à l'intérieur du paquet. Nous trouvons les mots clés suivants : content, protected content, hash, nocase, … voir les détails en annexe 2.
- ***non-payload*** : cette option cherche les données non utiles. Nous trouvons les mots clés suivants : fragoffset, ttl, tos, id, ipopts, … voir les détails en annexe 3.
- ***post-detection*** : cette option est le déclencheur spécifique à une règle qui se produit après qu'une règle s'est «déclenchée». Nous trouvons les mots clés suivants : logto, session, resp, react, tag, replace et detection filter, … voir les détails en annexe 4.

3.6 Installation de Snort 3 : [17]

Dans cette partie nous allons configurer **Snort** 3 avec **Splunk** en tant que système de détection d'intrusion réseau (**NIDS**) et système d'information de sécurité et de gestion des événements (**SIEM**) sur le système d'exploitation **Ubuntu18.04.4 LTS 64 bit**.

Snort 3 : c'est un système de détection d'intrusions réseau basé sur des règles. Il est très flexible et configurable pour s'adapter à des situations différentes. La nouvelle architecture de plugins permet aux développeurs de créer leurs propres plugins à utiliser dans le pipeline de traitement de Snort. Nous nous concentrerons sur l'utilisation la plus courante de Snort : en tant que serveur autonome comparant le trafic réseau à un ensemble de règles (ruleset) pour détecter le trafic suspect 'les éventuelles malveillances' et générer des alertes.

Snort 3 ne doit pas être utilisé dans les systèmes de production (Snort 3 est actuellement en cours de développement), cette version peut avoir subi des modifications qui posent des problèmes de compilation ou de configuration

Splunk : c'est un système de gestion des informations et des événements de sécurité (SIEM) qui collecte, stock et permet d'analyser et de visualiser facilement les données générées par l'hôte, y compris les alertes créées par Snort. Splunk est utilisé pour afficher graphiquement les alertes Snort, fournir des capacités de filtrage et de recherche. Pour démarrer il nécessite 5 Go d'espace libre sur le lecteur sur lequel il est installé (usually /opt/splunk).

OpenAppID : (Snort OpenAppID) permet à Snort d'identifier, de contrôler et de mesurer les applications utilisées sur le réseau. OpenAppID se compose d'un ensemble de packages (signatures) qui correspondent à des types spécifiques de données réseau, y compris des applications de couche 7, telles que Facebook, DNS, netflix, discus, google, ainsi que des applications qui utilisent ces services (chrome, http, https, etc.). Nous pouvons créer des règles basées sur ces signatures OpenAppID, nous permettant de bloquer les trafics qui correspondent à certaines applications (par exemple, bloquer tous les trafics Facebook).

Snort 2 vs Snort 3 : Snort 3 est une nouvelle conception complète du produit Snort pour répondre à un certain nombre de limitations de Snort 2.9.x. Certains points principaux sont le multi-threading, l'architecture de plug-in extensible, les fichiers de configuration basés sur lua, un shell de ligne de commande et bien d'autres idées.

3.6.1 Les étapes d'installation :

Tout d'abord, il faut assurer que le système soit à jour et dispose de la dernière liste de packages via la commande :

sudo apt-get update && sudo apt-get dist-upgrade -y

Ensuite, il faut assurer que l'horloge du système soit bien réglée. Cela sera important plus tard lorsque nous commencerons à traiter les alertes avec Splunk, via la commande :

sudo dpkg-reconfigure tzdata

Ensuite, nous allons créer un dossier snort_src, afin de stocker les téléchargements d'un certain nombre d'archives (.tar) et d'autres fichiers sources :

mkdir ~/snort_src

cd ~/snort_src

Installation des prérequis de Snort 3 :

sudo apt-get install -y build-essential autotools-dev libdumbnet-dev \libluajit-5.1-dev libpcap-dev zlib1g-dev pkg-config libhwloc-dev \cmake.

Les détails de ces packages peuvent être trouvés dans la section des exigences du manuel Snort3.

Ensuite, installons les outils facultatifs (mais fortement recommandé) suivants :

- **Liblzma-dev** : c'est bibliothèque de compression au format xz.
- **Openssl** : est une boîte à outils de chiffrement comportant deux bibliothèques, libcrypto et libssl, fournissant respectivement une implémentation des algorithmes cryptographiques et du protocole de communication SSL/TLS, ainsi qu'une interface en ligne de commande, openssl.
- **CppUTest** est un framework de type xUnit pour les tests unitaires C et C ++. Il est écrit en C ++ et vise la portabilité et la simplicité dans la conception.
- **libsqlite3-dev** : Bibliothèque partagée SQLite 3 qui est une bibliothèque C qui implémente un moteur de base de données SQL.
- **uuid-dev :** La bibliothèque libuuid crée et analyse des UUID 128 bits (UUID – « Universally Unique Identifier »). Consultez la RFC 4122 pour plus de renseignements. Ce paquet fournit l'environnement de développement pour la bibliothèque d'UUID.

sudo apt-get install -y liblzma-dev openssl libssl-dev cpputest \libsqlite3-dev uuid-dev

Pour la création de la dernière documentation à partir de l'arborescence des sources, y compris Snort ++ Developers Guide, installons les packages suivants (purement facultatifs). Ces packages font près de 800 Mo et peuvent être ignorés.

sudo apt-get install -y asciidoc dblatex source-highlight w3m

Puisque nous allons installer Snort à partir du référentiel github, nous avons besoin de quelques outils (pas nécessaire sur Ubuntu 19)

sudo apt-get install -y libtool git autoconf

La bibliothèque Snort DAQ (Data Acquisition Library) a quelques pré-requis qui doivent être installés :

sudo apt-get install -y bison flex

Si nous souhaitons exécuter Snort en mode en ligne à l'aide de NFQ, installons les packages requis (non requis pour le mode IDS ou le mode en ligne avec afpacket). En cas de doute, nous devons installer ce package :

sudo apt-get install -y libnetfilter-queue-dev libmnl-dev

Téléchargeons et installons safec pour les vérifications des limites d'exécution sur certains appels hérités de la bibliothèque C (ceci est facultatif mais recommandé) :

***cd** ~/snort_src*

wget https://github.com/rurban/safeclib/releases/download/v04062019/libsafec-04062019.0-ga99a05.tar.gz

tar -xzvf libsafec-04062019.0-ga99a05.tar.gz

***cd** libsafec-04062019.0-ga99a05/*

./configure

make

sudo make install

Installez PCRE : Perl Compatible Regular Expressions. Nous n'utilisons pas le référentiel Ubuntu car sa version est plus ancienne:

cd ~/snort_src/

wget https://ftp.pcre.org/pub/pcre/pcre-8.43.tar.gz

tar -xzvf pcre-8.43.tar.gz

cd pcre-8.43

./configure

make

sudo make install

Téléchargeons et installons gperftools 2.7, le malloc de mise en cache des threads de Google (utilisé dans Chrome). Tcmalloc est un allocateur de mémoire optimisé pour les situations de forte concurrence, qui offrira une meilleure vitesse pour le compromis d'une utilisation plus élevée de la mémoire. Nous ne voulons pas de la version de tcmalloc du référentiel Ubuntu (version 2.5) car elle n'est pas compatible avec Snort. Tcmalloc est facultatif mais recommandé :

cd ~/snort_src

wget https://github.com/gperftools/gperftools/releases/download/gperftools-2.7/gperftools-2.7.tar.gz

tar xzvf gperftools-2.7.tar.gz

cd gperftools-2.7

./configure

make

sudo make install

Snort 3 utilise Hyperscan pour une correspondance rapide des motifs. Hyperscan nécessite les en-têtes Boost et Ragel :

cd ~/snort_src

wget http://www.colm.net/files/ragel/ragel-6.10.tar.gz

tar -xzvf ragel-6.10.tar.gz

cd ragel-6.10

./configure

make

sudo make install

Hyperscan nécessite les bibliothèques Boost C++. Notons que nous n'utilisons pas la version du référentiel Ubuntu des en-têtes boost (libboost-all-dev) car Hyperscan nécessite des bibliothèques boost au numéro de version 1.58 ou supérieur, et la version du référentiel Ubuntu est trop ancienne. Téléchargeons les bibliothèques Boost 1.71.0, mais n'installons pas :

cd ~/snort_src

wget https://dl.bintray.com/boostorg/release/1.71.0/source/boost_1_71_0.tar.gz

tar -xvzf boost_1_71_0.tar.gz

Installons Hyperscan 5.2 à partir de la source, en référençant l'emplacement du répertoire source des en-têtes Boost :

cd ~/snort_src

wget https://github.com/intel/hyperscan/archive/v5.2.0.tar.gz

tar -xvzf v5.2.0.tar.gz

mkdir ~/snort_src/hyperscan-5.2.0-build

cd hyperscan-5.2.0-build/

cmake -DCMAKE_INSTALL_PREFIX=/usr/local -DBOOST_ROOT=~/snort_src/boost_1_71_0/ ../hyperscan-5.2.0

make

sudo make install

Si nous souhaitons tester si Hyperscan fonctionne, à partir du répertoire de construction, exécutons :

cd ~/snort_src/hyperscan-5.2.0-build/

./bin/unit-hyperscan

Snort a une exigence facultative pour les tampons plats (flatbuffers). Une bibliothèque de sérialisation efficace en mémoire :

cd ~/snort_src

*wget https://github.com/google/flatbuffers/archive/v1.11.0.tar.gz *

-O flatbuffers-v1.11.0.tar.gz

tar -xzvf flatbuffers-v1.11.0.tar.gz

mkdir flatbuffers-build

cd flatbuffers-build

cmake ../flatbuffers-1.11.0

make

sudo make install

Ensuite, téléchargeons et installons la bibliothèque DAQ (Data AcQuisition library) à partir du site Web Snort. Snort 3 utilise un DAQ différent de celui de la série Snort 2.9.x.x :

cd ~/snort_src

git clone https://github.com/snort3/libdaq.git

cd libdaq

./bootstrap

./configure

make

sudo make install

Mettre à jour les bibliothèques partagées :

sudo ldconfig

Nous sommes maintenant prêts à télécharger, compiler et installer Snort 3 à partir du référentiel github. Si nous souhaitons activer des fonctionnalités supplémentaires au moment de la compilation, telles que la possibilité de traiter des fichiers PCAP volumineux (plus de 2 Go) ou le nouveau shell de ligne de commande, nous devons exécuter : ./configure cmake.sh --help pour lister toutes les options possibles :

Téléchargeons et installons, avec les paramètres par défaut comme suit :

***cd** ~/snort_src*

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

***cd** snort3*

*./configure_cmake.sh --prefix=/usr/**local** --**enable**-tcmalloc*

***cd** build*

make

sudo make install

La dernière étape de l'installation consiste à vérifier que Snort est installé et peut fonctionner. Pour se faire, nous passons à l'exécutable Snort le drapeau -V

/usr/local/bin/snort -V

Nous devrions voir une sortie similaire à ce qui suit et que Snort est bien installé :

```
mokeddem@ubuntu_server:~$ sudo /usr/local/bin/snort -V
[sudo] password for mokeddem:

  ,,_     -*> Snort++ <*-
 o"  )~   Version 3.0.1 (Build 3)
  ''''    By Martin Roesch & The Snort Team
          http://snort.org/contact#team
          Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
          Copyright (C) 1998-2013 Sourcefire, Inc., et al.
          Using DAQ version 3.0.0
          Using LuaJIT version 2.1.0-beta3
          Using OpenSSL 1.1.1  11 Sep 2018
          Using libpcap version 1.8.1
          Using PCRE version 8.43 2019-02-23
          Using ZLIB version 1.2.11
          Using FlatBuffers 1.11.0
          Using Hyperscan version 5.2.0 2020-05-20
          Using LZMA version 5.2.2
```

3.7.1 Configuration des variables d'environnement :

Snort 3 nécessite quelques variables d'environnement pour fonctionner correctement. Nous stockons ces variables temporairement dans la session en cours et les sauvegardons de manière permanente dans notre fichier local *.bashrc* (notons que *LUA_PATH* ne peut pas être stocké dans le dossier */etc/profile* car il ne se chargera pas correctement. Nous devrions exécuter ces lignes pour chaque utilisateur qui a besoin d'exécuter Snort sur le système) :

```
export LUA_PATH=/usr/local/include/snort/lua/\?.lua\;\;
export SNORT_LUA_PATH=/usr/local/etc/snort
sh -c "echo 'export LUA_PATH=/usr/local/include/snort/lua/\?.lua\;\;' >> ~/.bashrc"
sh -c "echo 'export SNORT_LUA_PATH=/usr/local/etc/snort' >> ~/.bashrc"
```

Pour rendre ces variables d'environnement disponibles lorsque nous utilisons *sudo*, nous nous assurons que les commandes sont chargées via /etc/sudoers. Nous créons un fichier dans le dossier /etc/sudoers.d avec la commande :

```
sudo visudo -f /etc/sudoers.d/snort-lua
```

Il faut ajouter la ligne suivante à la fin :

```
Defaults env_keep += "LUA_PATH SNORT_LUA_PATH"
```

Utilisons ctrl-x pour quitter, enregistrez avec y après la demande, puis appuyons sur Entrée pour enregistrer dans un fichier temporaire (qui sera automatiquement copié dans /etc/sudoers.d/snort.lua). Testons maintenant Snort avec le fichier de configuration par défaut :

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

```
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
```

3.7.2 Configuration des cartes réseau :

Les cartes réseau modernes utilisent le déchargement (LRO par exemple) pour gérer le réassemblage des paquets réseau dans le matériel, plutôt que dans le logiciel. Dans la plupart des situations, cela est préférable car cela réduit la charge sur le système. Pour un NIDS, nous voulons désactiver LRO et GRO, car cela peut tronquer des paquets plus longs.

Nous devons créer un service systemD pour modifier ces paramètres. Déterminons d'abord le(s) nom(s) de(s) interfaces que nous aurons snort écouter en utilisant ifconfig. Vérifions l'état de grande réception-déchargement (LRO) et générique-réception-déchargement (GRO) pour ces interfaces. Dans l'exemple ci-dessous, notre nom d'interface est wlx0cb6d2f3331e. Nous utilisons ethtool pour vérifier le statut :

```
root@ubuntu_server:~# ethtool -k wlx0cb6d2f3331e | grep receive-offload
generic-receive-offload: on
large-receive-offload: off [fixed]
```

Nous voyions que GRO est activé et que LRO est désactivé (le 'fixe' signifie qu'il ne peut pas être modifié). Nous devons nous assurer que les deux sont définis sur «off» (ou «off [fixed]»). Nous pourrions utiliser la commande ethtool pour désactiver LRO et GRO, mais le paramètre ne persisterait pas lors des redémarrages. La solution est de créer un script systemD pour définir cela à chaque démarrage.

Créons le script systemD :

sudo vi /lib/systemd/system/ethtool.service

wlx0cb6d2f3331e c'est le nom de notre interface réseau.

```
[Unit]
Description=Ethtool Configration for Network Interface

[Service]
Requires=network.target
Type=oneshot
ExecStart=/sbin/ethtool -K wlx0cb6d2f3331e gro off
ExecStart=/sbin/ethtool -K wlx0cb6d2f3331e lro off

[Install]
WantedBy=multi-user.target
```

Une fois le fichier créé, activons le service

sudo systemctl enable ethtool

sudo service ethtool start

Ces paramètres persisteront désormais lors des redémarrages. Nous pouvons vérifier le paramètre en utilisant ethtool et l'interface (off ou off [fixed] sont le paramètre que nous devons voir) :

```
mokeddem@ubuntu_server:~$ sudo ethtool -k wlx0cb6d2f3331e | grep receive-offload
generic-receive-offload: off
large-receive-offload: off [fixed]
```

3.7.3 Installation de l'OpenAppID :

OpenAppID permet l'identification du trafic de la couche application (couche 7). Nous pouvons créer des règles qui fonctionnent sur le trafic de la couche application (par exemple pour bloquer Facebook) et enregistrer les statistiques de trafic pour chaque type de trafic détecté.

L'équipe Snort a mis au point un ensemble de détecteurs avec l'aide de la communauté que nous pouvons télécharger et installer, appelé *Application Detector Package*. Téléchargeons d'abord le package du détecteur OpenAppID et extrayons les fichiers

cd ~/snort_src/

wget https://snort.org/downloads/openappid/11581 -O OpenAppId-11581

tar -xzvf OpenAppId-11581

sudo cp -R odp /usr/local/lib/

Si nous obtenons une erreur indiquant que le fichier n'existe pas, il est possible que l'équipe Snort ait mis à jour l'ensemble de règles.

Accédons à https://snort.org/downloads#openappid et téléchargeons le fichier snort-openappid.tar.gz. Une fois les règles téléchargées et extraites comme ci-dessus, nous devons éditer notre fichier de configuration Snort pour pointer vers ce répertoire odp:

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

À la ligne 89 (le numéro de ligne peut être légèrement différent), nous verrons l'entrée *appid =.* Nous voudrons ajouter l'option app_detector_dir ici, en pointant vers le dossier parent du dossier odp que nous avons extrait ci-dessus. Ça devrait ressembler à ça :

```
appid =
{
    -- appid requires this to use appids in rules
    app_detector_dir = '/usr/local/lib',
}
```

Notons que nous devons avoir quatre espaces (pas une tabulation) pour la ligne en retrait. Maintenant, nous voulons tester le fichier de configuration pour savoir s'il se charge correctement.

snort -c /usr/local/etc/snort/snort.lua --warn-all

Cette commande validera que Snort peut lire correctement le fichier *snort.lua* et qu'il ne contient aucune erreur. Après avoir exécuté cette commande, nous devrions voir la sortie qui se termine par :

```
Snort successfully validated the configuration (with 136 warnings).
o")~   Snort exiting
```

Il est possible de voir un certain nombre d'avertissements (non fatals) relatifs à l'absence d'entrée dans appMapping.data, ceux-ci peuvent être ignorés tant que la sortie se termine par «Snort a validé la configuration avec succès».

Les avertissements ne sont pas fatals et peuvent être ignorés pour le moment (OpenAppID est encore en cours de développement).

Maintenant, nous allons créer une règle simple pour tester que OpenAppID fonctionne correctement.

sudo mkdir /usr/local/etc/snort/rules

sudo touch /usr/local/etc/snort/rules/local.rules

sudo vi /usr/local/etc/snort/rules/local.rules

Nous allons générer deux règles dans le fichier local.rules. La première règle utilise OpenAppID pour rechercher le trafic Facebook, la deuxième règle est bonne pour tester que les alertes sont générées correctement.

alert tcp any any -> any any (msg:"Facebook Detected"; appids:"Facebook";sid:10000001;)

alert icmp any any -> any any (msg:"ICMP Traffic Detected";sid:10000002;)

Maintenant, exécutons Snort et chargeons le fichier local.rules pour nous assurer qu'il charge correctement ses règles (en vérifiant que les règles sont correctement formatées) :

snort -c /usr/local/etc/snort/snort.lua \-R /usr/local/etc/snort/rules/local.rules

La sortie doit se terminer par «Snort a validé la configuration avec succès». Il peut y avoir des avertissements, mais il ne devrait y avoir aucune erreur.

```
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
```

Si nous défilons la sortie vers le haut, nous devrions voir ces deux règles de texte chargées avec succès (sous la section **rule counts**). Encore une fois, nous pouvons ignorer les avertissements concernant les entrées AppID manquantes. Maintenant, exécutons snort en mode détection sur notre interface réseau (wlx0cb6d2f3331e), en affichant des alertes sur la console :

sudo snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/local.rules \-i wlx0cb6d2f3331e -A alert_fast -s 65535 -k none

Le drapeau -k none indique à Snort d'ignorer les mauvais checksums, et le drapeau -s 65535 empêche snort de tronquer les paquets surdimensionnés. Les décodeurs Stream et Frag effaceront les paquets qui ont les mauvais checksums, et les paquets ne seront pas traités par les détecteurs OpenAppID. En incluant ces indicateurs, nous nous assurons qu'un paquet avec un mauvais checksums est toujours traité pour les alertes.

Snort chargera la configuration, puis affichera.

```
pcap DAQ configured to passive.
Commencing packet processing
++ [0] wlx0cb6d2f3331e
```

Cela signifie que snort écoute actuellement tout le trafic sur cette interface et le compare aux deux règles chargées. Lorsque le trafic correspond à une règle, Snort écrira une alerte dans la console. Maintenant, depuis une autre fenêtre sur cet ordinateur (ouvrons une

nouvelle fenêtre de terminal ou une deuxième session ssh), utilisons wget ou autre pour nous connecter à Facebook. Cela déclenchera la première règle.

wget facebook.com

A partir de la première fenêtre de la console, nous verrons une sortie d'alertes similaire à la suivante :

```
05/21-16:40:49.719126 [**] [1:10000001:0] "Facebook Detected" [**] [Priority: 0] [AppID: Facebook] {TCP} 1
57.240.195.35:443 -> 192.168.145.132:46346
05/21-16:40:49.719166 [**] [1:10000001:0] "Facebook Detected" [**] [Priority: 0] [AppID: Facebook] {TCP} 1
92.168.145.132:46346 -> 157.240.195.35:443
```

Utilisons ctrl-c pour arrêter Snort. Nous pouvons également envoyer un ping vers ou depuis cette machine pour générer des alertes lorsque Snort est à l'écoute (déclenchant la deuxième règle dans le fichier local.rules). C'est une bonne règle pour tester Snort, mais cela peut être un peu bruyant lors d'une utilisation réelle en production.

Si nous souhaitons collecter des statistiques OpenAppID (combien de trafic a été détecté par chaque détecteur), nous devrons l'activer dans le fichier snort.lua et exécuter Snort avec l'indicateur -l (répertoire de journal).

Créez d'abord un répertoire de journaux :

sudo mkdir /var/log/snort

Modifiont maintenant /usr/local/etc/snort/snort.lua pour permettre au détecteur appid de journaliser les statistiques (ligne 89) :

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

Maintenant, exécutons Snort en mode écoute sur notre interface réseau, Snort procéder la journalisation dans le dossier /var/log/snort :

sudo snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/local.rules \-i wlx0cb6d2f3331e -A alert_fast -s 65535 -k none -l /var/log/snort

Après avoir collecté des données et arrêté Snort (générons des données en utilisant wget pour nous connecter à facebook.com, puis envoyons un ping, ensuite arrêtez avec Ctrl-C), nous consultons appid_stats.log dans le chemin /var/log/snort. Ce fichier appartient à root, alors rendons le lisible par tous (nous changerons les permissions sur les fichiers écrits automatiquement par Snort dans une section ultérieure) :

sudo chmod a+r /var/log/snort/appid_stats.log

Maintenant nous pouvons regarder les statistiques de protocole que Snort a collecté :

```
mokeddem@ubuntu_server:~$ cat /var/log/snort/appid_stats.log
1594835705,Google,19071,222449
1594835705,Facebook,114078,1898412
1594835705,Firefox,5135,7021
1594835705,HTTP,5135,7021
1594835705,YouTube,1580,4882
1594835705,HTTPS,169183,2183198
1594835705,Mozilla,18927,3574
1594835705,SSL client,157808,2140087
1594835705,Doubleclick,4152,10770
1594835705,IGMP,174,0
1594835705,__unknown,75482,1150766
1594836015,Firefox,16217,20547
1594836015,HTTP,16217,20547
1594836015,HTTPS,4226,8636
1594836015,Mozilla,2008,4307
1594836015,SSL client,2008,4307
1594836015,IGMP,174,0
1594836015,__unknown,73558,1716350
1594836015,Google,2371,22821
1594836015,Facebook,2446,1716
1594836015,YouTube,8604,180894
1594836015,HTTPS,15684,206899
1594836015,SSL client,13421,205431
1594836015,IGMP,174,0
```

Il s'agit d'un fichier séparé par des virgules qui affiche l'heure (unixtime), le détecteur, les octets envoyés (tx) et les octets reçus (rx), dans cet ordre. Si nous ne souhaitons pas que ces données soient collectées, nous pouvons désactiver l'option de statistiques du journal dans le module appid de notre fichier de configuration snort.lua. Notons que ces données sont différentes des alertes générées par les règles de notre fichier local.rules.

Pour plus d'informations sur les détecteurs, veuillez consulter le Guide des détecteurs OpenAppID.

3.7.4 Installation des règles de la communauté Snort :

Les règles Snort 3 ont plus d'options que les règles Snort 2, bien qu'elles sont téléchargées automatiquement avec PulledPork ou manuellement sur snort.org, elles fonctionnent correctement. Pour les tests, nous voudrons télécharger l'ensemble des règles de communauté spécifiquement créées pour Snort 3. Nous allons également créer des dossiers que snort attend une fois que nous commençons à exécuter snort en tant que NIDS :

cd ~/snort_src/

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

tar -xvzf snort3-community-rules.tar.gz

cd snort3-community-rules

sudo mkdir /usr/local/etc/snort/rules

sudo mkdir /usr/local/etc/snort/builtin_rules

sudo mkdir /usr/local/etc/snort/so_rules

10 sudo mkdir /usr/local/etc/snort/lists

sudo cp snort3-community.rules /usr/local/etc/snort/rules/

sudo cp sid-msg.map /usr/local/etc/snort/rules/

Testons maintenant si les règles se chargent correctement :

snort -c /usr/local/etc/snort/snort.lua \-R /usr/local/etc/snort/rules/snort3-community.rules

```
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
```

Remarque : de nombreuses règles de la communauté snort3 sont inactives, leurs activation se fait à la demande et au besoin de l'administrateur. Fondamentalement, la raison est que certaines règles peuvent générer de faux positifs, l'équipe Snort a donc désactiver ces règles, car elles peuvent inonder vos journaux ou entraîner des interruptions de trafic excessives si vous exécutez Snort dans NIPS mode. Pour activer ces règles, veuillez utiliser la commande suivante :

sudo sed -i '17,$s/^# //' /usr/local/etc/snort/rules/snort3-community.rules

Nous pouvons toujours recopier le fichier d'origine des règles à partir de l'archive tar des règles en cas ou Snort génère trop de faux positifs. Le fait d'activer toutes les règles, met plus de 3400 règles en action.

Nous pouvons exécuter Snort avec les drapeaux suivants pour détecter les problèmes pendant les tests et la configuration : les drapeaux ***warn-all*** et ***pedantic***. À partir du manuel Snort 3 :

Les avertissements ne sont émis que si ***–warn***- * est spécifié. ***–Warn-all*** active tous les avertissements, et ***- pedantic*** rend ces avertissements fatals.

Nous ne voudrons pas utiliser le drapeau ***–pedantic*** lors de l'exécution de Snort, car de simples avertissements de débit (flowbits bit de flux définis mais non utilisés dans une règle, un problème courant) généreront des avertissements et entraîneront une erreur de Snort. C'est cependant un bon drapeau pour tester votre configuration.

3.7.5 Activation des règles intégrées :

Pour activer les alertes d'activation du décodeur et de l'inspecteur (celles-ci détectent les trafics malveillantes qui ne peuvent pas être facilement détectées avec les règles normales), nous devons activer cette option dans notre fichier de configuration snort : snort.lua, situé dans le dossier : **/usr/local/etc/snort/**, en utilisant la commande sudo :

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

A la ligne 169, définissons ***enable_builtin_rules*** à ***true***. Les lignes qui commencent par deux traits d'union sont des commentaires (les commandes désactivées sont généralement mises en commentaire) et ne sont pas analysées par Snort lors du chargement. La suppression des deux traits d'union avant ***enable_builtin_rules*** active cette option. N'oublions pas que toutes les lignes indentées de notre snort.lua doivent comporter quatre espaces (pas une tabulation) ou la configuration ne se chargera pas. Le module ips dans snort.lua devrait ressembler à ceci :

```
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
    --include = 'snort3-community.rules'
    include = RULE_PATH .. '/ips.include',
}
```

Maintenant, testez que les modifications que vous avez apportées au fichier de configuration snort.lua ne contiennent aucune erreur :

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

Si nous faites défiler la sortie de la commande précédente, nous devrions voir les règles intégrées chargées (nous n'avons pas chargé les règles de communauté comme avant, présentées sous forme de règles de texte ci-dessus) :

```
--------------------------------------------------
rule counts
       total rules loaded: 476
            builtin rules: 476
            option chains: 476
            chain headers: 1
--------------------------------------------------
```

Maintenant, chargeons les deux types de règles (les règles intégrées spécifiées par ***enable_builtin_rules*** dans le module ***ips*** du fichier ***snort.lua***, et les règles de texte désactivés spécifiées sur la ligne de commande) :

snort -c /usr/local/etc/snort/snort.lua \-R /usr/local/etc/snort/rules/snort3-community.rules

```
-------------------------------------------------
rule counts
       total rules loaded: 4004
               text rules: 3490
             builtin rules: 514
             option chains: 4004
             chain headers: 294
-------------------------------------------------
```

Nous verrons à la fois les règles intégrées et les règles de texte chargées.

Les nombres peuvent être légèrement différents, surtout si les commentaires des lignes dans le fichier de règles de communauté sont supprimés.

3.7.6 Fichiers de configuration de Snort (*snort*.lua et snort_*defaults*.lua) :

Nous voudrons modifier notre fichier de configuration Snort afin de ne pas avoir à spécifier de paramètres sur la ligne de commande, en particulier pour les fichiers de règles. Nous modifions d'abord les chemins d'accès aux fichiers de règles :

sudo vi /usr/local/etc/snort/snort_defaults.lua

À partir de la ligne 25, apportez les modifications suivantes :

```
--------------------------------------------------------------------------------
-- Path to your rules files (this can be a relative path)

RULE_PATH = '/usr/local/etc/snort/rules'
BUILTIN_RULE_PATH = '/usr/local/etc/snort/builtin_rules'
PLUGIN_RULE_PATH = '/usr/local/etc/snort/so_rules'

-- If you are using reputation preprocessor set these
WHITE_LIST_PATH = '/usr/local/etc/snort/lists'
BLACK_LIST_PATH = '/usr/local/etc/snort/lists'

--------------------------------------------------------------------------------
```

Ensuite, nous créons les fichiers de règles (s'ils ne sont pas déjà créés) :

sudo touch /usr/local/etc/snort/rules/ips.include

sudo touch /usr/local/etc/snort/rules/local.rules

Modifions le fichier ips.include, qui est une liste de fichiers de règles que Snort devrait inclure (nous devons le faire pour charger plusieurs fichiers de règles indépendants) :

sudo vi /usr/local/etc/snort/rules/ips.include

Avec le contenu suivant :

```
include rules/snort3-community.rules
include rules/local.rules
```

Editons le fichier sid-msg.map (qui fournit plus d'informations sur les alertes en sortie) :

sudo vi /usr/local/etc/snort/rules/sid-msg.map

Ajoutons maintenant ces informations sur nos deux règles dans le fichier local.rules :

10000001 || ICMP Test detected || url,tools.ietf.org/html/rfc792

10000002 || Facebook Traffic Seen || url,facebook.com

Ensuite, nous éditons le ***snort.lua*** pour permettre le chargement de ces règles :

A la ligne 24, nous définissons notre réseau, qui fait référence au sous-réseau local (les règles utilisent ces informations pour déterminer si une alerte correspond). Définissons ici les paramètres du sous-réseau local pour qu'elles correspondent à notre sous-réseau. Ce dernier a l'adresse IP 192.168.10.0 avec un masque sous-réseau de 24 bits :

```
-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = '192.168.10.0/24'
```

Ensuite, nous nous assurerons que le fichier ips.include est chargé par Snort, et nous vérifierons que les règles intégrées sont activées. A partir de la ligne 169, assurons-nous que notre fichier correspond à ce qui suit (rappelons que les retraits sont de 4 espaces) :

```
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
    --include = 'snort3-community.rules'
    include = RULE_PATH .. '/ips.include',
}
```

Testons maintenant si Snort peut charger correctement le fichier de configuration que nous avons modifié ci-dessus:

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

Assurons-nous que la sortie se termine par ce qui suit (sinon nous avons des problèmes à résoudre). Le nombre d'avertissements peut être différents, mais ces avertissements ne sont pas fatals et peuvent être ignorés.

```
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
```

Si nous défilons la sortie vers le haut, nous devrions voir les règles en cours de chargement

```
Finished /usr/local/etc/snort/snort.lua:
Loading /usr/local/etc/snort/rules/ips.include:
Loading rules/snort3-community.rules:
Finished rules/snort3-community.rules:
Loading rules/local.rules:
Finished rules/local.rules:
Finished /usr/local/etc/snort/rules/ips.include:
--------------------------------------------------
rule counts
       total rules loaded: 4004
               text rules: 3490
            builtin rules: 514
            option chains: 4004
            chain headers: 294
--------------------------------------------------
```

3.7.7 Passer des fichiers PCAP dans Snort et des alertes de sortie vers .csv :

Cette section n'est pas requise pour notre configuration, mais elle nous présentera un certain nombre de drapeaux dont nous aurons besoin plus tard, elle nous donnera quelques fichiers ***pcap*** qui peuvent générer des alertes pour les tests et nous aidera à mieux comprendre Snort 3.

Si nous recherchons des fichiers ***PCAP*** pouvant être utilisés pour générer des alertes à partir de nos règles intégrées et de la communauté à des fins de test, nous trouvons que l'ensemble des données ***MACCDC 2012*** est bon. Commençons par télécharger deux des fichiers ***pcap*** :

```
cd ~
mkdir pcaps
cd pcaps
wget https://download.netresec.com/pcap/maccdc-2012/maccdc2012_00000.pcap.gz
gunzip maccdc2012_00000.pcap.gz
wget https://download.netresec.com/pcap/maccdc-2012/maccdc2012_00001.pcap.gz
gunzip maccdc2012_00001.pcap.gz
```

Maintenant, nous exécutons Snort, en lui demandant :

- De charger le premier fichier pcap que nous avons téléchargé.
- De charger toutes les règles de la communauté.
- D'afficher des alertes sur la console (nous décomposons tous les drapeaux utilisés ici si nous défilons vers le bas).

Notons que cela peut prendre un certain temps et que de nombreuses alertes défileront sur l'écran. Les touches ctrl-c permettent d'accélérer les choses pour les impatients (cela a pris environ 90 secondes pour s'exécuter sur mon système, générant 104861 alertes) :

```
snort -c /usr/local/etc/snort/snort.lua \-r ~/pcaps/maccdc2012_00000.pcap -A alert_fast -s 65535 -k none
```

Une description de ce que font chacun de ces drapeaux est résumée dans le tableau suivant :

Drapeau	Description
sudo snort	C'est le fichier binaire snort que nous appelons.
-c /usr/local/etc/snort/snort.lua	Le fichier de configuration de Snort.
r ~/pcaps/maccdc2012_00000.pcap	Le chemin d'accès au fichier pcap.
-A alert_fast	Sortie vers la console à l'aide du plugin alert_fast.
-s 65535	Réglez le snaplen pour que Snort ne tronque pas et ne supprime pas les paquets surdimensionnés.
-k none	Ignorez les mauvais checksums, sinon Snort laissera tomber les paquets avec des mauvais checksums, et ils ne seront pas évalués.

Pour traiter plusieurs fichiers pcap à la fois, modifiez la dernière commande pour parcourir tous les fichiers pcap dans ce même répertoire comme suit :

*snort -c /usr/local/etc/snort/snort.lua --pcap-filter *.pcap \--pcap-dir ~/pcaps -A alert_fast -s 65535 -k none*

Décomposons cette commande :

Drapeau	Description
sudo snort	C'est le fichier binaire Snort que nous appelons.
-c /usr/local/etc/snort/snort.lua	Le fichier de configuration de Snort.
–pcap-filter *.pcap	Cela indique à Snort comment identifier les fichiers pcap situés dans le répertoire qui contient les fichiers pcap.
–pcap-dir ~/pcaps	Cela indique à Snort le répertoire où se trouvent les fichiers pcap.
-A alert_fast	Sortie vers la console à l'aide du plugin alert_fast.
-s 65535	Réglez le snaplen pour que Snort ne tronque pas et ne supprime pas les paquets surdimensionnés.
-k none	Ignorez les mauvais checksums, sinon Snort laissera tomber les paquets avec des mauvais checksums, et ils ne seront pas évalués.

Lorsque nous avons exécuté cette commande, Snort a généré 137538 alertes (regardons dans la sortie à la fin pour ***Module Statistics : Detection*** : total des alertes) en deux minutes trente-six secondes (regardons dans la sortie pour ***Summary Statistics : timing*** : seconds).

```
-------------------------------------------------
Module Statistics
-------------------------------------------------
detection
                analyzed: 12833954
              hard_evals: 14498334
             raw_searches: 535913
          cooked_searches: 610988
             pkt_searches: 1146901
             alt_searches: 103
             key_searches: 88671
          header_searches: 174504
            body_searches: 1537
            file_searches: 82689
             total_alerts: 137538
                   logged: 137538
-------------------------------------------------

..

-------------------------------------------------
Summary Statistics
-------------------------------------------------
timing
                  runtime: 00:02:36
                  seconds: 156.616996
                  packets: 12833954
                 pkts/sec: 82268
```

Nous pouvons modifier cette dernière commande pour utiliser n'importe quel plugin de sortie (nous avons utilisé le plugin ***alert_fast*** pour afficher les événements sur la console), soit en le spécifiant en ligne de commande comme nous l'avons fait ci-dessus avec l'indicateur -A, soit en activant le plugin dans notre fichier ***snort.lua*** (ligne 232). Par exemple, pour utiliser le plugin ***alert_csv*** afin qu'il enregistre les données d'alerte dans un fichier csv, nous devons modifier notre ***snort.lua*** comme ceci (ligne 232) :

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

Puis exécutons Snort comme suit :

sudo snort -c /usr/local/etc/snort/snort.lua -r ~/pcaps/maccdc2012_00000.pcap \-s 65535 -k none -l /var/log/snort -q -m 0x1b

Tout d'abord, nous remarquerons que rien ne s'affiche à l'écran pendant que Snort traite les fichiers PCAP. En effet, nous ne verrons aucune sortie affichée sur la console lorsque nous exécutons la commande ci-dessus, car nous écrivons des alertes dans le fichier csv et supprimons toutes les autres sorties avec le drapeau -q. Nous remarquerons peut-être quelques erreurs relatives au service odp du détecteur lua, celles-ci peuvent être ignorées. Nous noterons que nous ne spécifions pas le plugin de sortie sur la ligne de commande comme nous l'avons fait auparavant, c'est parce que nous avons activé le plugin ***alert_csv*** dans notre ***snort.lua*** (en spécifier un sur la ligne de commande aurait la priorité sur tous les plugins de sortie configurés dans notre ***snort.lua)***. Les nouveaux drapeaux que nous utilisons sont décrits ci-dessous :

Drapeau	Description
-l /var/log/snort	Le dossier dans lequel nos fichiers de sortie (fichiers csv dans ce cas) doivent être enregistrés.
-q	Mode silencieux - Ne pas afficher la bannière et le rapport d'état.
-m 0x1b	Cela définit l'umask pour les fichiers écrits sur 033.

Umask : Snort utilise par défaut un ***umask*** de 077, qui empêche toute personne de lire les fichiers journaux. Cela pose des problèmes lors de la tentative d'ingestion de journaux avec d'autres outils à moins qu'ils ne s'exécutent sous le même compte utilisateur (il s'agit d'un risque de sécurité). Pour résoudre ce problème, nous utilisons le drapeau -m pour passer à un nouveau ***umask*** de 033 (000 011 011 en binaire, 0x1b en hexadécimal). Ceci signifie que nos

fichiers journaux auront les autorisations: rwxr – r–, permettant à tout le monde de lire ces fichiers. Si nous avons des besoins de sécurité spécifiques, modifions ce paramètre pour qu'il soit plus restrictif. Ces fichiers seront la propriété de l'utilisateur ***root*** lorsque Snort sera exécuté comme ci-dessus (nous utiliserons un utilisateur non root pour exécuter Snort plus tard dans ce chapitre).

Lorsque nous revenons à une invite de commande, cela signifie que Snort a terminé de traiter le fichier pcap. Nous devrions voir ***alert_csv.txt*** dans notre répertoire **/var/log/snort/.**

En utilisant la commande ***wc -l*** (word count), nous pouvons voir combien d'alertes ont été générées à partir de ces fichiers pcap. Ici, nous pouvons voir qu'il y a 21635 lignes dans le fichier, et comme chaque ligne est une alerte individuelle, nous savons que Snort a généré autant d'événements :

wc -l /var/log/snort/alert_csv.txt

```
root@ubuntu_server:/# wc -l /var/log/snort/alert_csv.txt 21635 /var/log/snort/alert_csv.txt
```

Il y a plus d'options que ce plugin (et d'autres plugins). Toutes ces informations et plus encore sont disponibles dans le manuel Snort 3.

3.7.8 Plugin de sortie d'alertes JSON :

Snort 3 a un certain nombre de plugins de sortie. Plus tôt dans ce chapitre, nous avons utilisé le plugin ***alert_fast*** pour envoyer des alertes à la console, et nous avons utilisé le plugin ***alert_csv*** pour écrire des alertes dans un fichier journal au format csv. Il existe un certain nombre de plugins disponibles, chacun avec de nombreuses options.

Afin d'importer facilement les fichiers journaux d'alertes de Snort 3 dans notre interface graphique d'analyse ***Splunk***, nous voudrons utiliser le plugin de sortie ***alert_json*** pour écrire toutes les alertes dans un fichier texte au format ***json***. L'activation du plugin de sortie json est facile, il suffit de modifier notre fichier ***snort.lua*** (autour de la ligne numéro 262) :

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

Tout d'abord, désactivons le plugin ***alert_csv*** en plaçant deux tirets devant chaque ligne dans le plugin, et activons le plugin ***alert_json*** comme indiqué ci-dessous. Nous pourrions

garder les deux plugins activés et nous obtiendrons à la fois des fichiers csv et json représentant les mêmes alertes.

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

Dans le plugin ***alert_json***, nous spécifions trois options :

1. Nous utilisons l'option de fichier pour activer la sortie d'alertes vers le fichier au format json (plutôt que vers la console).
2. Nous spécifions l'option de limite pour indiquer à Snort quand passer à un nouveau fichier. Lorsque le fichier de sortie atteint 10 Mo, un nouveau fichier sera créé, en utilisant l'heure unix actuelle dans le nom de fichier. Nous l'avons défini à 10 Mo pour les tests, mais sur un système de production, il est possible d'augmenter ce nombre à 100 Mo ou plus.
3. Nous spécifions l'option ***fields***, qui identifie les champs spécifiques de l'alerte qui doivent être inclus dans la sortie json. Dans cet exemple, nous avons choisi tous les champs possibles à afficher.

Remarque : après le test, nous pouvons choisir de supprimer certains de ces champs (les champs ***vlan*** et ***mpls*** ne sont souvent pas nécessaires, et le ***b64_data*** contient la totalité de la charge utile du paquet, qui peut être supprimée pour économiser de l'espace). Il ne faut jamais supprimer le champ des secondes et assurons-nous qu'il s'agit toujours du premier champ répertorié. Cela permettra à ***Splunk*** de traiter correctement les événements.

Si nous exécutez Snort comme nous l'avons fait auparavant lors de l'analyse des fichiers pcap et de la sortie dans un fichier csv (mais maintenant avec le plug-in de sortie alert_json activé, nous ne verrons à nouveau rien sortir sur la console.

*sudo snort -c /usr/local/etc/snort/snort.lua --pcap-filter *.pcap \--pcap-dir ~/pcaps -l /var/log/snort -s 65535 -k none -m 0x1b*

Une fois que Snort a terminé de traiter nos fichiers pcap, nous pouvons regarder dans notre dossier de journal (spécifié ci-dessus par **-l /var/log/snort**), nous verrons les fichiers json contenant toutes les alertes :

```
root@ubuntu_server:/# ls -lh /var/log/snort/
total 96M
-rw-r--r-- 1 snort snort 565K août  20 15:51 alert_csv.txt
-rw-r--r-- 1 snort snort  10M juil. 31 21:14 alert_csv.txt.1596226447
-rw-r--r-- 1 snort snort  10M août   4 18:52 alert_csv.txt.1596563562
-rw-r--r-- 1 snort snort  10M août  15 16:15 alert_csv.txt.1597504540
-rw-r--r-- 1 snort snort 5,2M août  20 15:51 alert_json.txt
-rw-r--r-- 1 snort snort  10M juil. 16 20:55 alert_json.txt.1594929330
-rw-r--r-- 1 snort snort  10M juil. 28 20:34 alert_json.txt.1595964896
-rw-r--r-- 1 snort snort  10M juil. 31 21:27 alert_json.txt.1596227263
-rw-r--r-- 1 snort snort  10M août   1 20:25 alert_json.txt.1596309938
-rw-r--r-- 1 snort snort  10M août   4 19:46 alert_json.txt.1596566787
-rw-r--r-- 1 snort snort  10M août   5 19:21 alert_json.txt.1596651684
-rw-r--r-- 1 snort snort  95K août  20 15:45 appid_stats.log
```

Nous remarquerons qu'il existe un certain nombre de fichiers alert_json.txt.nnnnnnnnnn. Les nombres indiquent l'heure unix à laquelle le fichier a été créé, et chaque fichier fait 10 Mo comme nous l'avons spécifié dans le ***snort.lua***.

3.7.9 Script de démarrage Snort :

Nous créons un script systemD pour exécuter Snort automatiquement au démarrage. Nous ferons également exécuter Snort en tant qu'utilisateur régulier (non root) après le démarrage pour des raisons de sécurité. Créons d'abord l'utilisateur et le groupe Snort :

sudo groupadd snort

sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort

Supprimons les anciens fichiers journaux (il est possible de les déplacer pour les conserver) :

*sudo rm /var/log/snort/**

Nous devons accorder les droits d'utilisateur «snort» sur le répertoire des journaux :

sudo chmod -R 5775 /var/log/snort

sudo chown -R snort:snort /var/log/snort

Créons le fichier de service systemD :

sudo vi /lib/systemd/system/snort3.service

```
[Unit]
Description=Snort3 NIDS Daemon
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -s 65535 \
-k none -l /var/log/snort -D -u snort -g snort -i wlx0cb6d2f3331e -m 0x1b

[Install]
WantedBy=multi-user.target
```

Nous devons être familiarisé avec tous les drapeaux que nous transmettons à Snort, car ce sont les mêmes que nous avons utilisé dans les exemples précédents. Nous ajoutons le drapeau -D qui permet à Snort de s'exécuter en tant que démon. Si nous souhaitons exécuter différentes options, nous pouvons les ajouter ici.

Voici une liste de tous les drapeaux que nous utilisons :

Drapeau	Description
/usr/local/bin/snort	C'est le chemin vers le fichier binaire snort. Nous n'utilisons pas sudo ici car le script sera lancé avec des privilèges élevés (root).
-c /usr/local/etc/snort/snort.lua	Le fichier de configuration de Snort.
-l /var/log/snort	Le chemin d'accès au dossier dans lequel Snort stockera tous les fichiers journaux qu'il génère.
-D	Exécutez en tant que démon.
-u snort	Après le démarrage (et après avoir fait quelque chose qui nécessite des privilèges élevés), passez à l'utilisateur «snort».
-g snort	Après le démarrage, exécutez en tant que groupe «snort».
-i wlx0cb6d2f3331e	L'interface (ou la carte réseau) à écouter.
-m 0x1b	Cela définit l'umask pour les fichiers écrits sur 033.

Activons le service Snort systemD et démarrons-le :

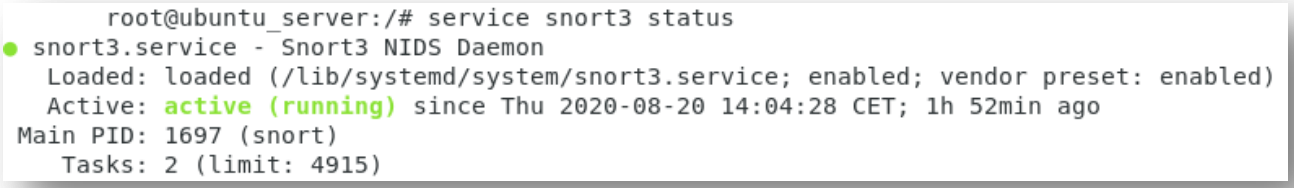
sudo systemctl enable snort3

sudo service snort3 start

Vérifier l'état du service :

service snort3 status

Le résultat doit être similaire à ce qui suit, indiquant "actif (***active running***)" :



Nous pouvons vérifier la sortie complète du service avec la commande suivante en cas de problème :

sudo journalctl -u snort3.service

3.7.10 Splunk :

C'est le logiciel que nous utiliserons comme solution ***SIEM*** (Security information and event management), qui affichera graphiquement (via une interface web) toutes les alertes que Snort a généré. Il existe des outils puissants pour rechercher et comprendre ces alertes et en tirer des informations plus approfondies. ***Splunk*** est un logiciel gratuit (pour les grandes installations Splunk il faut acheter une licence pour des fonctionnalités supplémentaires)

3.7.10.1 Installation de Splunk :

Pour installer correctement Splunk il faut :

- Créer un compte gratuit sur le site Web de ***Splunk***, pour télécharger le logiciel et les modules complémentaires.
- Sous Splunk Free, nous cliquerons sur le lien intitulé Télécharger.
- Cliquons sur l'onglet Linux, puis sur le bouton Télécharger maintenant à côté de .deb (Ubuntu, un système basé sur Debian).
- Acceptons la licence et cliquons sur le bouton Démarrer votre téléchargement maintenant.
- La page de téléchargement ouvrira automatiquement une fenêtre pour enregistrer le téléchargement sur le système local.
- En peut utiliser wget pour télécharger le programme d'installation. Le téléchargement fait environ 290 Mo.

Une fois que nous avons le programme d'installation de Splunk sur le système, nous devons l'installer. Depuis le répertoire dans lequel nous avons enregistré le programme d'installation :

sudo dpkg -i splunk-7..deb*

Cela installera Splunk dans **/opt/splunk**. Notons que le volume sur lequel Splunk est installé doit être au minimum de 5 Go. Dans le cas contraire Splunk ne démarrera pas. Les index dans lesquels Splunk stocke toutes les données de journal collectées résident dans un sous-dossier de l'emplacement d'installation. Il faut s'assurer qu'il y a suffisamment d'espace sur ce volume pour toutes les données que nous prévoyons de collecter.

Maintenant, nous voulons démarrer Splunk pour la première fois (en acceptant la licence et en prenant toutes les options par défaut), puis nous voulons configurer Splunk pour qu'il soit lancé automatiquement au démarrage. Nous serons invité à créer un nouvel utilisateur administrateur et un nouveau mot de passe pour Splunk. Enregistrons ces informations d'identification, car nous les utiliserons plus tard pour nous connecter à l'interface Web.

sudo /opt/splunk/bin/splunk start --answer-yes --accept-license

sudo /opt/splunk/bin/splunk enable boot-start

Le serveur Splunk écoute maintenant sur le port 8000 de ce serveur (http://localhost:8000 si nous nous connectons depuis la machine locale, ou via l'adresse IP du système depuis un autre ordinateur). Le nom d'utilisateur et le mot de passe sont ceux que nous configurons lors de l'installation de Splunk.

Splunk fonctionne actuellement avec la licence d'essai gratuite Enterprise, offrant toutes les fonctionnalités Enterprise pendant 60 jours et nous permettant d'indexer 5 Go de données de journal par jour. La seule fonctionnalité que nous perdrons une fois la licence d'évaluation expirée qui affectera cette installation est la suppression des connexions authentifiées. Une fois que nous somme passé à la licence gratuite, nous ne serons pas invités à nous connecter à l'interface Web Splunk.

Splunk Enterprise offre un certain nombre de fonctionnalités, notamment un serveur de déploiement pour mettre à jour automatiquement les instances Splunk et les applications Splunk qu'elles exécutent automatiquement, plusieurs comptes d'utilisateurs avec des autorisations configurables, l'équilibrage de charge et d'autres fonctionnalités.

3.7.10.2 Configuration de Splunk :

Connectons-nous maintenant à notre instance Splunk, cela en utilisant le nom d'utilisateur et le mot de passe que nous avons créé lors de l'installation. Le serveur Splunk écoute sur le port 8000 (http://localhost:8000).

Nous devons installer un plugin Splunk (appelé ***add-on***) qui nous permettra de collecter facilement les journaux créés par Snort 3 et de les normaliser (il faut assurer que le nom des champs est cohérent avec les données NIDS afin que les applications Splunk puissent afficher nos données facilement).

Pour installer cette application, à partir de la page Web principale de votre instance Splunk, cliquez sur le lien intitulé **+Find More Apps** sur le côté gauche de l'interface Web Splunk :

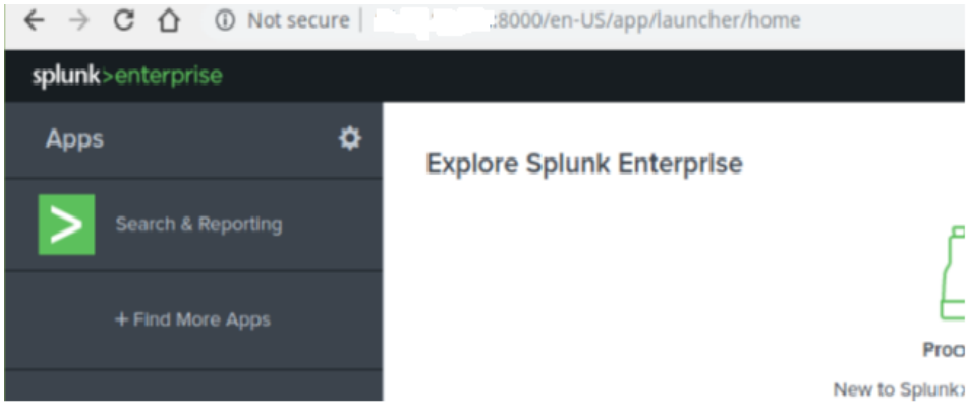
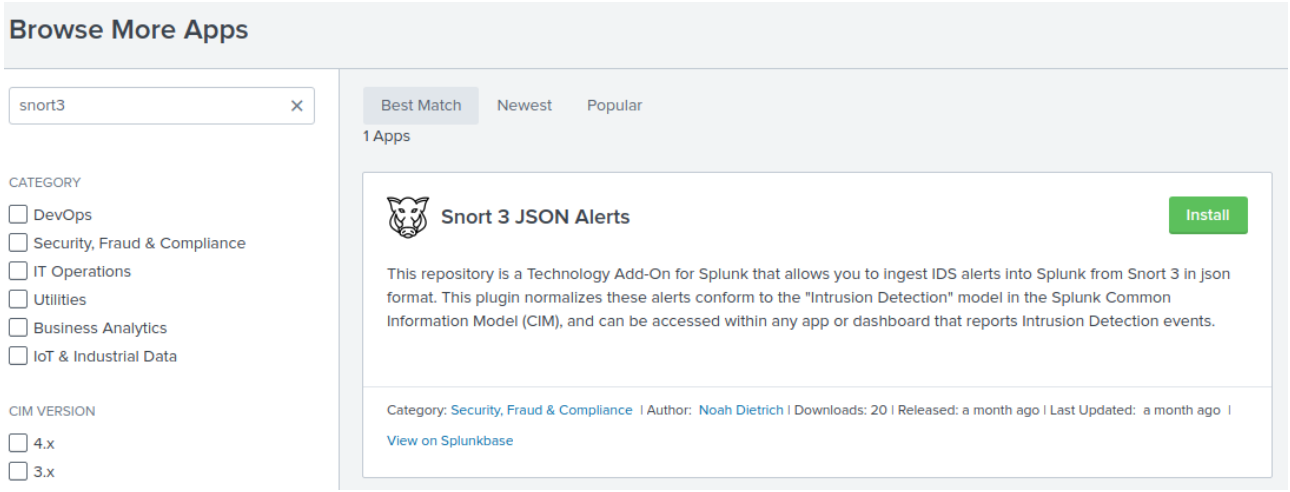


Figure 3.2 –configuration de Splunk.

Cela nous mènera à ***Splunkbase***, un référentiel en ligne pour les modules complémentaires Splunk qui étendent et améliorent les fonctionnalités de votre installation Splunk. Recherchons dans Splunkbase Snort3 et nous obtiendrons un résultat : les alertes JSON Snort 3. Cliquons sur le bouton d'installation vert, à côté de ce module complémentaire :



Entrons le nom d'utilisateur et le mot de passe que nous avons créés avec Splunk lorsque nous nous somme inscrit pour télécharger Splunk (pas le nom d'utilisateur et le mot de passe que nous avons créés pour notre instance de serveur Splunk locale). Acceptons les conditions

générales et cliquons sur Connexion et installation. Cliquons sur Terminé une fois l'installation terminée.

Ce module complémentaire est maintenant installé sur votre serveur Splunk. Ensuite, nous devons configurer cet add-on pour indiquer à Splunk où sont stockés les fichiers journaux générés par Snort 3 afin que Splunk puisse les ingérer. Nous faisons cela à partir de la ligne de commande avec un fichier de configuration :

sudo mkdir /opt/splunk/etc/apps/TA_Snort3_json/local

sudo touch /opt/splunk/etc/apps/TA_Snort3_json/local/inputs.conf

sudo vi /opt/splunk/etc/apps/TA_Snort3_json/local/inputs.conf

Entrons le texte suivant dans ce fichier inputs.conf :

*[monitor:///var/log/snort/*alert_json.txt*]*

sourcetype = snort3:alert:json

Restart Splunk :

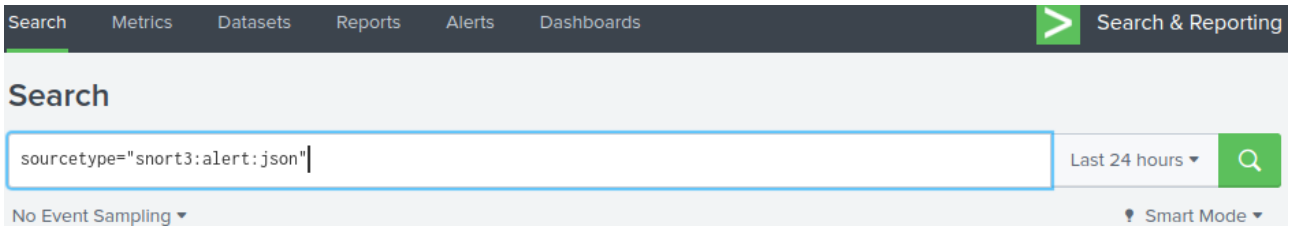
sudo /opt/splunk/bin/splunk restart

Maintenant, quand Splunk démarre, il va scanner le répertoire **/var/log/snort** pour les fichiers json, leur attribuer le type de source **snort3:alert:json**, et les ingérer afin que nous puissions les rechercher.

A partir de notre instance Splunk, connectons-nous (depuis que nous avons redémarré le serveur) et cliquons sur le lien de l'application Recherche et création de rapports sur le côté gauche. Dans le champ de recherche, saisissez le texte suivant :

sourcetype="snort3:alert:json"

Il faut ensuite cliquer sur l'icône en forme de loupe verte pour lancer la recherche.



Cela montrera tous les événements que notre serveur collecte. Nous ne verrons peut-être pas beaucoup d'événements, surtout si nous avons supprimé les anciens fichiers json que nous avons créés à partir de nos fichiers pcap. Nous pouvons créer quelques nouvelles alertes

en utilisant ping et wget facebook.com si nous le souhaitons (nous avons créé ces règles auparavant) et si aucune alerte ne s'affiche. Il y a un léger décalage entre un événement généré et affiché dans Splunk. Si nous ne voyons toujours aucune alerte, modifions la plage horaire (la liste déroulante définie sur les dernières 24 heures à côté de l'icône de recherche) sur toute l'heure et relançons la recherche. Si nous ne voyons toujours aucun événement, vérifions qu'il y a des fichiers json dans notre dossier **/var/log/snort**.

3.7.10.3 Utilisation de Splunk :

Dans notre étude nous n'abordons pas en détail l'utilisation de Splunk. Il existe d'excellentes ressources gratuites disponibles sur Splunk que nous mentionnons ci-dessous.

Nous trouverons ci-dessous quelques recherches simples qui pourraient nous être utiles pour démarrer. Pour afficher tous les événements d'une table avec l'heure, la source, la destination et le message, exécutez la recherche suivante :

sourcetype="snort3:alert:json" | table _time src_ap dst_ap msg

Pour afficher le nombre de tous les événements par destination :

sourcetype="snort3:alert:json" | stats count by dest

Pour afficher toutes les sources d'événements sur une carte :

sourcetype="snort3:alert:json" | iplocation src_addr | stats count by Country | geom geo_countries featureIdField="Country"

Voici quelques excellentes ressources gratuites pour utiliser Splunk : **EBook :** Splunk.com.

3.8 Conclusion :

Dans ce chapitre nous avons abordé l'IDS **SNORT** en détail : téléchargement, installation et configuration. Ensuite, nous avons expliqué en détail comment, télécharger, installer et configurer **Splunk,** ce dernier permet d'afficher graphiquement (via une interface web) toutes les alertes que Snort pourra générer.

Par ailleurs, puisque la base des signatures est étendue, elle nécessite un travail supplémentaire et constant de l'administrateur qui doit veiller à la recherche et le téléchargement des nouvelles mises à jour d'une façon manuelle. Car il n'y a pas de procédure de mise à jour automatiques. [07]

Malheureusement, malgré leur utilité, en pratique l'IDS **SNORT** souffre plus ou moins de deux problèmes : le nombre important de faux positifs et de faux négatifs. Les faux positifs (c'est-à-dire les fausses alertes) sont générés lorsque l'IDS identifie des activités normales comme des intrusions, alors que les faux négatifs correspondent aux attaques ou intrusions qui ne sont pas détectées (aucune alerte n'est générée). Afin de remédier à cette situation, nous proposons une amélioration. Celle-ci sera basée sur l'approche comportementale et le principe du Deep Learning. Nous verrons ça en détails dans le chapitre suivant.

Chapitre 4 :
Amélioration de l'IDS 'Snort' avec le Deep_Learing

Chapitre 4 : Amélioration de l'IDS 'Snort' avec le Deep Learning

4.1 Introduction :

L'IDS basé sur les signatures (comme Snort) est efficace dans la détection des attaques connues et se traduit par une précision de détection élevée et des taux de fausses alertes moins élevés. Cependant, ses performances souffrent lors de la détection d'attaques inconnues ou nouvelles en raison de la limitation des règles qui peuvent être préalablement installées dans un IDS. D'un autre côté, l'IDS basé sur les anomalies est bien adapté pour la détection d'attaques inconnues et nouvelles. [23]

A cet effet nous avons utilisés, dans notre travail, l'approche comportementale par l'application du Deep Learning et en particulier les réseaux de neurones récurrents RNN, pour améliorer l'IDS Snort.

L'analyse comportementale a généralement deux phases : une phase d'apprentissage et une phase de détection. Au cours de la phase d'apprentissage, le système apprend à reconnaitre ce qu'est un comportement normal, et au cours de la phase de détection, il identifie les comportements anormaux. L'analyse des scenarios n'a pas de phase d'apprentissage. Ils viennent avec une base de scénarios d'attaques prédéfinis qu'ils doivent reconnaitre (signature). [02].

4.2 Machine Learning :

Le Machine Learning est une technologie d'intelligence artificielle, qui permet aux ordinateurs d'apprendre sans avoir été programmés explicitement à cet effet. Pour apprendre et se développer, les ordinateurs ont besoin de données à analyser et sur lesquelles s'entraîner. De fait, le Big Data est l'essence du Machine Learning, et c'est la technologie permettant l'exploitation potentiellement du Big Data. [21]

4.3 Deep Learning :

Le deep learning ou l'apprentissage profond est un type d'intelligence artificielle dérivé de la machine learning (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées. [23]

4.4 Les réseaux de neurones :

Ce sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle et reliés par des poids. Ces poids de connexion gouvernent le fonctionnement du réseau. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Les réseaux de neurones ont plusieurs avantages dans la mise en œuvre d'un système de détection d'intrusion. Ils sont très efficaces et rapides dans la tâche de classification [19]. Ils sont capables d'apprendre et d'identifier facilement les nouvelles menaces qui leur sont soumises. Les réseaux de neurones sont capables de traiter les données incomplètes, imprécises et provenant de sources multiples. La rapidité naturelle des réseaux de neurones permet de réduire les dommages lorsque la menace est détectée [18]. L'utilisation des réseaux de neurones permet d'extraire les relations non linéaires qui existent entre les différents champs d'un paquet et de détecter en temps réel les attaques complexes [15]. Les réseaux de neurones, après avoir appris correctement, ont une bonne capacité de généralisation, c'est-à-dire qu'ils sont capables de calculer avec précision les sorties correspondantes même pour des données qui n'ont pas été apprises. La flexibilité qu'offrent les réseaux de neurones est également l'un des atouts pour la détection d'intrusion [14].

4.5 Les réseaux de neurones récurrents :

Les réseaux de neurones récurrents (RNN pour Recurrent Neural Networks) sont des réseaux de neurones dans lesquels l'information peut se propager dans les deux sens, y compris des couches profondes aux premières couches. En cela, ils sont plus proches du vrai fonctionnement du système nerveux humain, qui n'est pas à sens unique. Ces réseaux possèdent des connexions récurrentes au sens où elles conservent des informations en mémoire : ils peuvent prendre en compte à un instant t un certain nombre d'états passés.

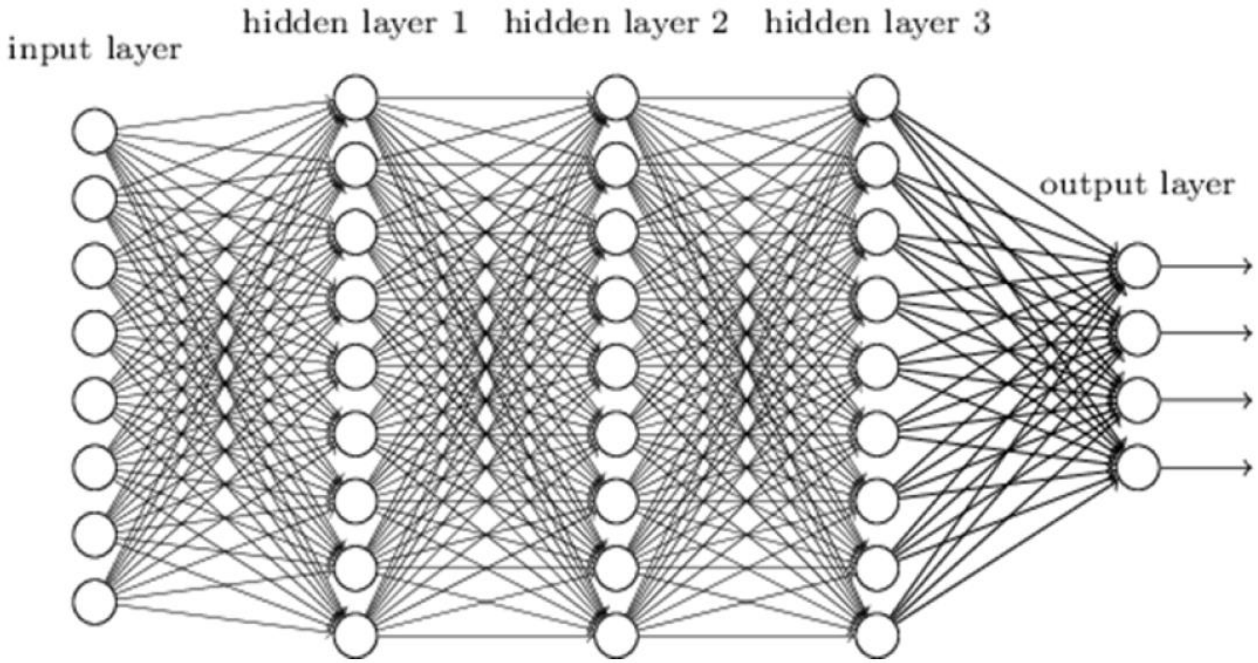


Figure 4.1 : Réseaux de neurones à cinq couches [23]

Dans la figure 4.1 nous avons une couche d'entrée de huit neurones, avec trois couches cachés de neuf neurones pour chaque couche et une couche de sortie de quatre résultats (neurones), notre modèle possède une couche d'entrée de 41 neurones représente les colonnes de la base de données (Data Set) KDD'99 qui nous voulons l'utilisée, deux couches cachés et une couche de sortie d'un seul neurone (normale ou intrusion).

4.6 RNN LSTM Long Short-Term Memory :

C'est un type de RNN est très utilisé sur défirent domaines, L'idée est de diviser le signal entre ce qui est important à court terme à travers le ***hidden state*** (analogue à la sortie d'une cellule de RNN simple), et ce qui l'est à long terme, à travers le ***cell state***. Ainsi, le fonctionnement global d'un LSTM peut se résumer en 3 étapes :

1. Détecter les informations pertinentes venant du passé, piochées dans le ***cell state*** à travers la ***forget gate*** ;
2. Choisir, à partir de l'entrée courante, celles qui seront pertinentes *à* ***long terme***, via *l'****input gate***. Celles-ci seront ajoutées au ***cell state*** qui fait office de mémoire longue ;
3. Piocher dans le nouveau ***cell state*** les informations importantes *à court terme* pour générer le ***hidden state*** suivant à travers ***l'output gate***.

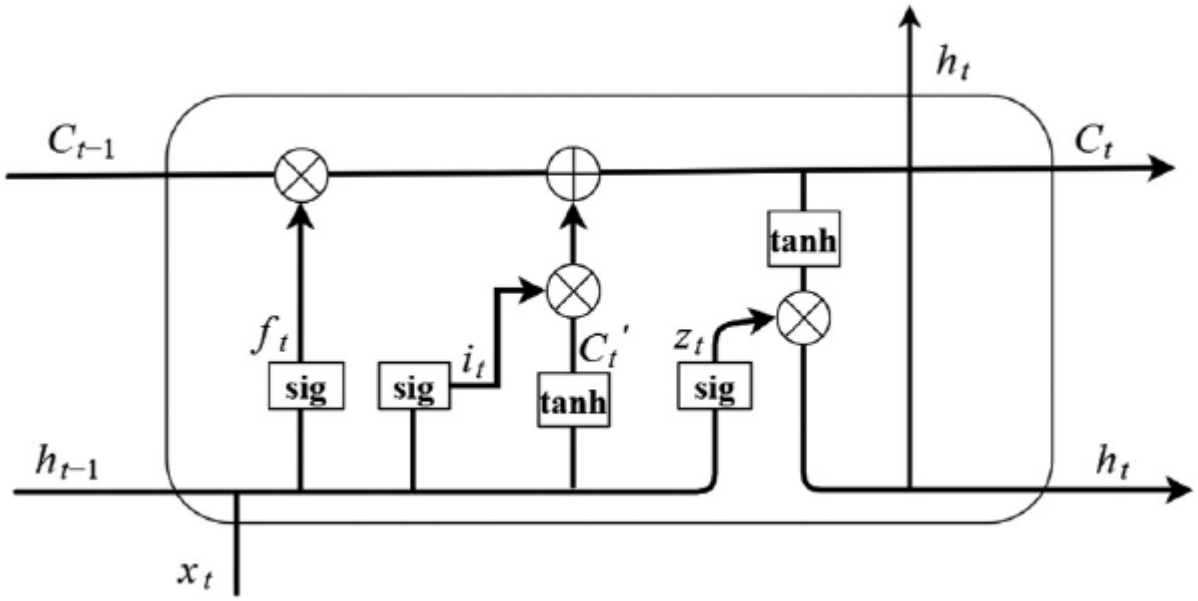
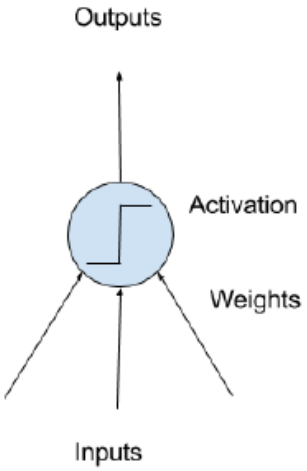


Figure 4.2 : Structure d'une cellule LSTM [34]

4.7 Vue d'ensemble des réseaux de neurones profonds : [23]

4.7.1 Neurone : La pierre angulaire des réseaux de neurones sont les neurones artificiels. Ce sont de simples unités de calcul qui ont des signaux d'entrée pondérés et produisent un signal de sortie à l'aide d'une fonction d'activation.

Figure 4.3 : Modèle d'un simple neurone. [23]



4.7.2 Poids des neurones (Neuron Weights) :

Chaque neurone a un biais (***bias***) qui peut être considéré comme une entrée qui a toujours la valeur 1.0 et qui doit également être pondérée. Par exemple, un neurone peut avoir deux entrées, auquel cas il nécessite trois poids. Un pour chaque entrée et un pour le biais. Les pondérations sont souvent initialisées à de petites valeurs aléatoires, telles que des valeurs comprises entre 0 et 0,3, bien que des schémas d'initialisation plus complexes puissent être utilisés. Comme la régression linéaire, des poids plus élevés indiquent une complexité et une fragilité accrues du modèle. Il est souhaitable de garder les pondérations dans le réseau petites et des techniques de régularisation peuvent être utilisées.

4.7.3 Activation *:*

Les entrées pondérées sont additionnées et passées par une fonction d'activation, parfois appelée fonction de transfert. Une fonction d'activation est un simple mappage de l'entrée pondérée additionnée à la sortie du neurone. C'est ce qu'on appelle une fonction d'activation car elle régit le seuil auquel le neurone est activé et la force du signal de sortie. Des fonctions d'activation par étapes historiquement simples étaient utilisées ou si l'entrée additionnée était supérieure à un seuil, par exemple 0,5, alors le neurone produirait une valeur de 1, si non il produirait un 0.

De nombreuses fonctions d'activation existent, nous avons utilisés dans notre modèle la fonction ***relu*** pour les couches cachés et sigmoïde (***sogmoid***) pour la couche de sortie : [W7]

La fonction sigmoïde : $\sigma(z) = \frac{1}{1+e^{-z}}$ ou z= $b + \sum_i w_i x_i$ ou :

- $\mathbf{x_i}$ valeurs d'entrée.
- $\mathbf{w_i}$ les poids.
- **b** c'est le biais.
- **e** est la constante exponentielle, à peu près égale à 2.71828.

La fonction ***relu*** laisse toutes les valeurs positives passer inchangées et attribue un zéro au valeur négatives : R(z) = max (0,z).

4.8 Détail du réseau de neurones :

Deep Learning implique de créer des réseaux de neurones très grands et profonds (c'est-à-dire en plus de la couche d'entrée et seul de la sortie il existe de nombreuses couches cachées de neurones) pour résoudre des problèmes spécifiques. Ainsi, de la même manière que les neurones sont organisés en couches dans les cellules cérébrales humaines, les neurones des réseaux neuronaux sont souvent organisés en couches également. Ainsi, un algorithme est profond si l'entrée est passée par plusieurs couches non-linéarités avant d'être sortie.

4.8.1 Couche d'entrée (*Input Layer*) :

La première couche qui prend l'entrée d'un ensemble de données est appelée la couche d'entrée ou visible, car c'est la partie exposée du réseau neuronal. Souvent, un réseau de neurones est caractérisé par une couche d'entrée avec un neurone pour chaque valeur d'entrée dans l'ensemble de données, dans notre cas nous avons 41 entrées.

4.8.2 Couches cachées (Hidden Layer) :

Ces couches effectuent les calculs et transmettent les informations à la couche en sortie. Elles sont appelées cachées car elles ne sont pas directement exposées à l'entrée. L'exemple le plus simple d'un réseau de neurones est d'avoir un seul neurone dans la couche cachée qui produit directement une valeur. Avec l'augmentation de la puissance de calcul et des bibliothèques très efficaces, des réseaux de neurones très profonds peuvent être construits. Le réseau neuronal peut contenir de nombreuses couches cachées, dans notre modèle nous avons deux couches cachées.

4.8.3 Couche de sortie (*Output Layer*) :

C'est la dernière couche qui est appelée la couche de sortie et elle est chargée d'exporter la valeur ou le vecteur de valeurs qui correspondent au format requis pour le problème, dans notre cas c'est le résultat normal ou intrusion, et ce, après la formation de notre modèle, qui est responsable de la production des variables de sortie.

4.9 Comment construire le modèle d'apprentissage approfondi :

4.9.1 Le jeu de données KDD'99 : [20]

C'est un ensemble de données construit à partir de sept semaines de trafic réseau capturé avec TCPdump (Garcia, 2017), il est sans doute le plus cité et utilisé (University of California, Irvine, 1999) et ceci avec une existence de 20 années, KDD Cup 99 est utilisé comme ensemble de données exemple dans les systèmes de détection d'intrusion comportementale [11]. Chaque paquet de l'ensemble des données de KDD Cup 99 est constitué de 41 champs et est labellisés comme paquet normal ou paquet anormal avec les types d'attaques. Parmi ces champs, 37 sont des champs de type numérique et 4 sont des champs de type non numérique. KDD'99 regroupe 37 types d'attaque. Ces attaques sont divisées en quatre grandes classes :

- DOS : déni de service ;
- R2L : accès non autorisé depuis une machine distante ;
- U2R : accès non autorisé aux privilèges du super-utilisateur local (root) ;
- Probe (sonde) : surveillance et autres sondages comme le balayage des ports (port scanning).

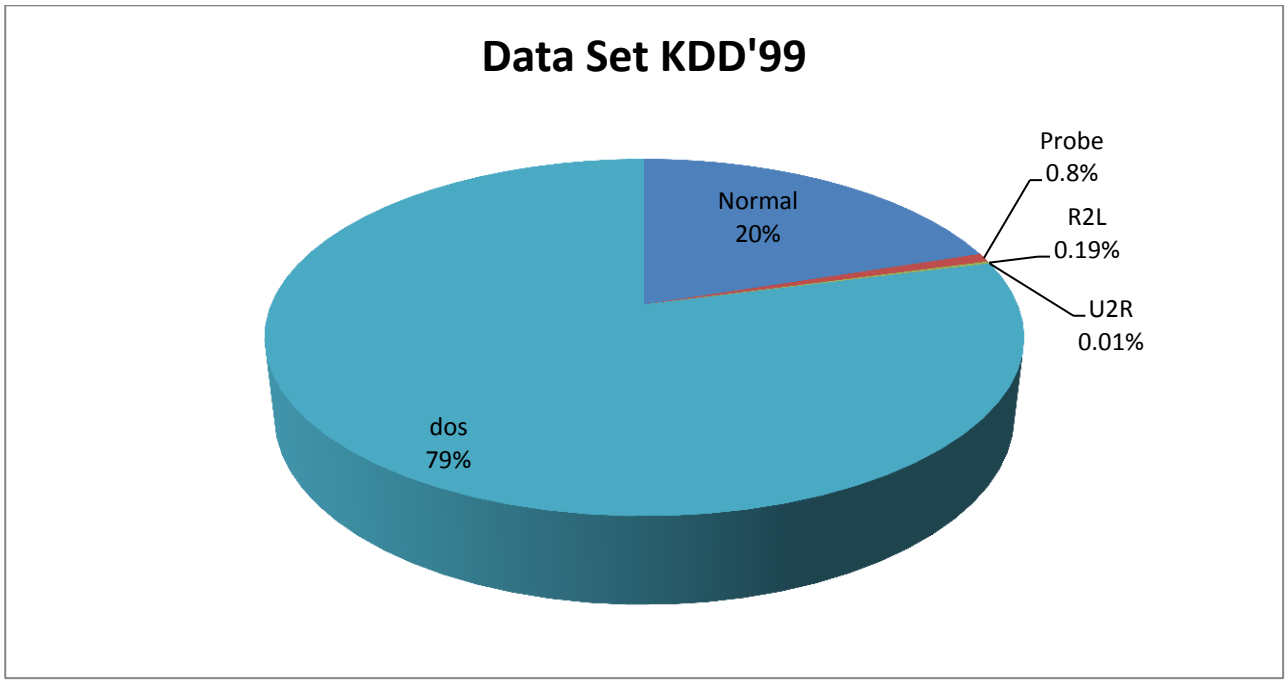


Figure 4.4 : Le contenue du KDD'99 [03]

Il est nécessaire de présenter les étapes de construction de notre modèle : [21]

- Le chargement des données.
- Le prétraitement et la normalisation des données.

- Phase d'apprentissage et de validation.
- Phase de prédiction.

Dans notre application nous avons utilisés la base de données KDD'99 au format csv.

4.9.2 Le prétraitement et la normalisation des données :

Le prétraitement :

Le prétraitement est appliqué sur les champs non numériques. Les champs non numériques sont : le type de protocole (TCP, UDP, ICMP), le type de service (aol, auth, bgp, … Z39_50), le drapeau (OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, SH) et la classe du paquet (Normal 0 ou Intrusion 1). Pour le type de protocole, nous affectons les valeurs numériques suivantes : TCP=1, UDP=2 et ICMP=3. Nous affectons 0 aux paquets normaux et 1 aux paquets intrusions. Pour les champs de types service et drapeau, nous avons leur attribuer les valeurs numériques par ordre croissant ou décroissant de leur nombre total.

Normalisation :

Il s'agit de transformer les données pour qu'elles varient entre 0 et 1 afin de rendre les données plus homogènes et ainsi simplifier l'apprentissage des réseaux. Nous allons dans notre application utiliser la normalisation Min-Max scaler. Soit min_x et max_x le minimum et le maximum respectifs des valeurs de l'attribut X, la valeur normalisée X_i**'= (**x_i-min_x)/(max_x-min_x)**.** Pour chaque attribut du vecteur de données, calculer sa valeur normalisée et remplacer cette dernière par la valeur normalisée.

4.9.3 Séparation de nos données d'entrainement et de teste :

Nous avons tout d'abord divisé notre Data Set KDD'99 en deux sous-ensembles base de d'apprentissage (E_train représente les 41 premiers colonnes et S_train c'est le résultat) et base de teste (E_test représente les 41 premiers colonnes et S_test c'est le résultat), cela est nécessaire pour pouvoir utiliser une partie des données soit de 75% (370 515 ligne) pour entrainer notre modèle et une partie soit de 25% (123 505 ligne) pour tester ses performances, cette division garantit que notre modèle apprend uniquement à partir des données d'apprentissage et que nous pouvons ensuite tester ces performances avec les données de teste.

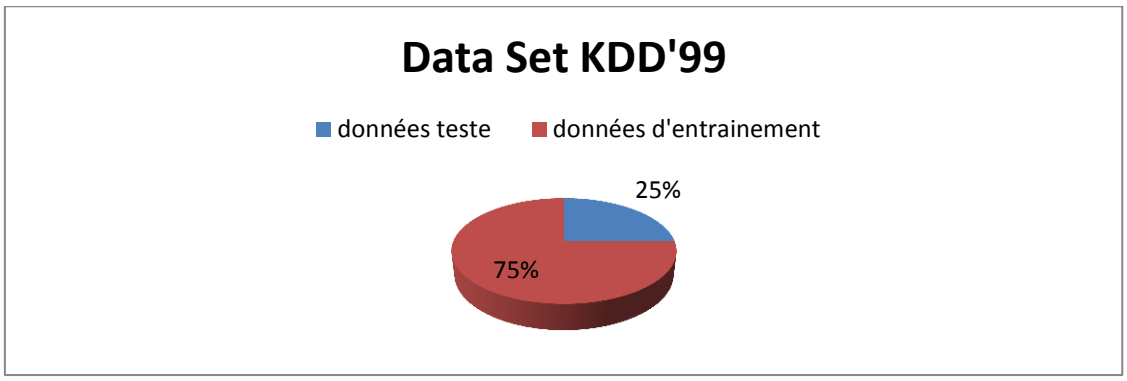


Figure 4.5 : La division de notre Data Set KDD'99

4.9.4 Réalisation du réseau de neurones (modèle) :

Avant de commencer la construction de notre modèle nous aurons utilisés l'environnement de développement *Anaconda* sur notre machine, *anaconda* est une distribution libre et open source des langages de programmation Python (langage de programmation utilisé) et R appliqué au développement d'application dédiées à la science des données et à l'apprentissage automatique.

Nous utilisons l'API *keras* pour construire le modèle d'apprentissage en profondeur. Pour ce faire. A partir de keras, nous importerons ensuite le module *Sequential* pour initialiser notre modèle. Nous importerons également les deux modules *LSTM* pour construire les couches cachées, le module *Dense* pour la couche de sortie, ce qui ajoutera des couches à notre modèle d'apprentissage approfondi.

Keras : est une *API* de réseau de neurones écrite en Python. Il fonctionne au-dessus de *TensorFlow*. C'est une abstraction de haut niveau de ces cadres d'apprentissage en profondeur et rend donc l'expérimentation plus rapide et plus facile. *Keras* est modulaire, ce qui signifie que la mise en œuvre est transparente, les développeurs pouvant rapidement étendre des modèles en ajoutant des modules.

TensorFlow : est une bibliothèque de logiciels open-source pour l'apprentissage automatique. Cela fonctionne efficacement avec le calcul impliquant des tableaux; c'est donc un excellent choix pour le modèle que nous allons construire dans notre application. De plus, TensorFlow permet l'exécution de code sur une CPU ou un GPU (processeur de la carte graphique), ce qui est une fonctionnalité utile, en particulier lorsque nous travaillons avec un jeu de données volumineux.

Lors de la réalisation d'un modèle d'apprentissage en profondeur, nous spécifions généralement trois types de couche : *Input layer, Hidden Layer, Output layer* vue ci-dessus.

Puisqu'il s'agit d'un problème de classification, nous allons créer un variable de classificateur. Ou nous avons étiqueté des données et souhaitons effectuer des prévisions basées sur les données étiquetées *model = Sequential().*

Nous pouvons maintenant commencer à ajouter des couches à notre modèle. Avec le module *LSTM* et la fonction *.add()* en spécifiant un certain paramètre :

1. Le nombre d'unités que notre réseau devrait avoir. La connexion entre différents nœuds est ce qui forme le réseau de neurones.
2. kernel_initializer. : durant l'adaptation de notre modèle d'apprentissage en profondeur, les poids seront initialisés à des nombres proches de zéro, mais pas de zéro. Pour ce faire, utilisons l'initialiseur de distribution uniforme. kernel_initializer.
3. la fonction activation : Notre modèle d'apprentissage en profondeur apprendra à travers cette fonction. Il existe généralement des fonctions d'activation linéaires et non linéaires. Nous utilisons la fonction d'activation [relu] car elle généralise bien nos données.
4. input_dim : qui représente le nombre d'entités dans notre jeu de données soit 41 entrées.

En fin, nous allons ajouter la couche de sortie qui nous donnera les prédictions, en utilisant le module ***Dense*** et la fonction *.add(),* la couche de sortie prend les paramètres suivants :

- Le nombre de nœuds de sortie. nous nous attendons à obtenir un résultat : si un paquet est normal ou intrusion. Par conséquent, nous spécifions un nœud de sortie.
- Nous utilisons la fonction d'activation sigmoïde, puisque nous avons deux possibilité de sortie soit normal ou intrusion.

Ensuite, il faut appliquer un gradient descente au réseau de neurones. C'est la stratégie d'optimisation qui réduit les erreurs pendant le processus de formation. La descente de gradient est la manière dont les poids attribués de manière aléatoire dans un réseau neuronal sont ajustés en réduisant la fonction de coûts (cost function), qui est une mesure de la performance d'un réseau neuronal sur la base des résultats attendus.

L'objectif d'une descente de gradient est d'obtenir le point où l'erreur est au minimum. Ceci est fait en recherchant où la fonction de coût est à son minimum, ce qui est appelé un minimum local. En descente, nous faites la différence pour trouver la pente en un point spécifique et déterminer si la pente est négative ou positive : nous descendons dans le minimum de la fonction de coût. Il existe plusieurs types de stratégies d'optimisation, mais nous en utiliserons une connue sous le nom ***adam*** dans notre application.

L'application de la descente de gradient se fait via la fonction ***compile*** qui prend les paramètres suivants:

- optimizer= 'adam' est la descente du gradient.
- loss est une fonction que nous utiliserons dans la descente de gradient. Comme il s'agit d'un problème de classification binaire, nous utilisons la fonction loss = 'binary_crossentropy`.
- métrique que nous utiliserons pour évaluer notre modèle. Dans ce cas, nous souhaitons l'évaluer en fonction de sa précision lors de la prévision, nous utilisons metrics = ["accuracy"].

Nous sommes prêts à adapter notre modèle à notre ensemble de données. Keras rend cela possible via la méthode .fit (), en précisons :

- Les données d'entrainement d'entrée (E_train) et les données d'entrainement de sorties (S_train).
- Les données de validation d'entrée (E_test) et les données de validation de sorties (S_test).
- Les critères d'arrêt et de contrôle (val loss) après le nombre de patience de dix.
- Le nombre d'epochs (1000).

4.9.5 Exécution de prévisions sur le test :

Pour commencer à faire des prédictions, nous utiliserons le jeu de données de test dans le modèle que nous avons créé. Keras nous permet de faire des prédictions en utilisant la fonction .predict ().

Vérification de la matrice de confusion :

Dans cette étape, nous utiliserons une matrice de confusion pour vérifier le nombre de prédictions correctes et incorrectes. Une matrice de confusion (matrice d'erreur), est une matrice carrée indiquant le nombre de vrais positifs (tp), de faux positifs (fp), de vrais négatifs (tn) et de faux négatifs (fn) d'un classificateur.

- Un * vrai positif * est un résultat dans lequel le modèle prédit correctement la classe positive (également appelée sensibilité ou rappel).
- Un * vrai négatif * est un résultat dans lequel le modèle prédit correctement la classe négative.
- Un * faux positif * est un résultat dans lequel le modèle prédit de manière incorrecte la classe positive.
- Un * faux négatif * est un résultat dans lequel le modèle prédit de manière incorrecte la classe négative.

Pour ce faire, nous utiliserons une matrice de confusion fournie par scikit-learn.

4.10 Discussion des résultats :

Dans notre application et après plusieurs tentatives d'amélioration du modèle, on statuer sur le modèle définie qui comme suit :

Une couche d'entrée de 41 neurones (les colonnes du KDD'99 sans la dernière celle de la classe).

Deux couches cachées :

- La première couche cachée de 128 neurones.
- La deuxième couche cachée de 64 neurones.

La couche de sortie constituée d'un seul neurone qui est le résultat normal ou intrusion.

On a fixé à 1000 le nombre d'epoch (cycle complet d'apprentissage).

```
Epoch 16/1000
345814/345814 [==============================] - 3656s 11ms/sample - loss: 0.0022 - acc: 0.9993 - val_loss: 0.0020 - val_acc:
0.9994
Epoch 17/1000
345814/345814 [==============================] - 2098s 6ms/sample - loss: 0.0020 - acc: 0.9994 - val_loss: 0.0019 - val_acc: 0.
9994
Epoch 18/1000
345814/345814 [==============================] - 2086s 6ms/sample - loss: 0.0020 - acc: 0.9994 - val_loss: 0.0016 - val_acc: 0.
9995
Epoch 19/1000
345814/345814 [==============================] - 2237s 6ms/sample - loss: 0.0020 - acc: 0.9994 - val_loss: 0.0020 - val_acc: 0.
9994
Epoch 20/1000
345814/345814 [==============================] - 2179s 6ms/sample - loss: 0.0019 - acc: 0.9994 - val_loss: 0.0017 - val_acc: 0.
9995
Epoch 21/1000
345814/345814 [==============================] - 2197s 6ms/sample - loss: 0.0018 - acc: 0.9995 - val_loss: 0.0023 - val_acc: 0.
9993
Epoch 22/1000
345814/345814 [==============================] - 2181s 6ms/sample - loss: 0.0018 - acc: 0.9994 - val_loss: 0.0024 - val_acc: 0.
9993
Epoch 23/1000
345814/345814 [==============================] - 2163s 6ms/sample - loss: 0.0016 - acc: 0.9995 - val_loss: 0.0016 - val_acc: 0.
9995
Epoch 24/1000
345814/345814 [==============================] - 2129s 6ms/sample - loss: 0.0017 - acc: 0.9994 - val_loss: 0.0024 - val_acc: 0.
9993
Epoch 00024: early stopping
```

Figure 4.6 : la phase d'apprentissage.

La matrice de confusion :

vrai positif : 24 243	faux positif : 39	`confusion_matrix [[24243  39]` `[  42 99181]]`
faux négatif : 42	vrai négatif : 99 181	

Les paramètres loss et accuracy :

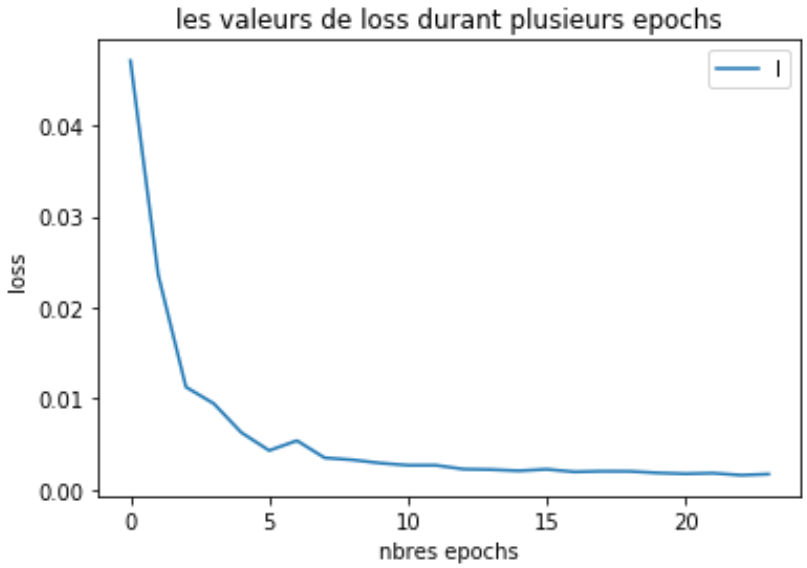


Figure 4.7 : Les valeurs de loss par rapport au nombre d'epochs.

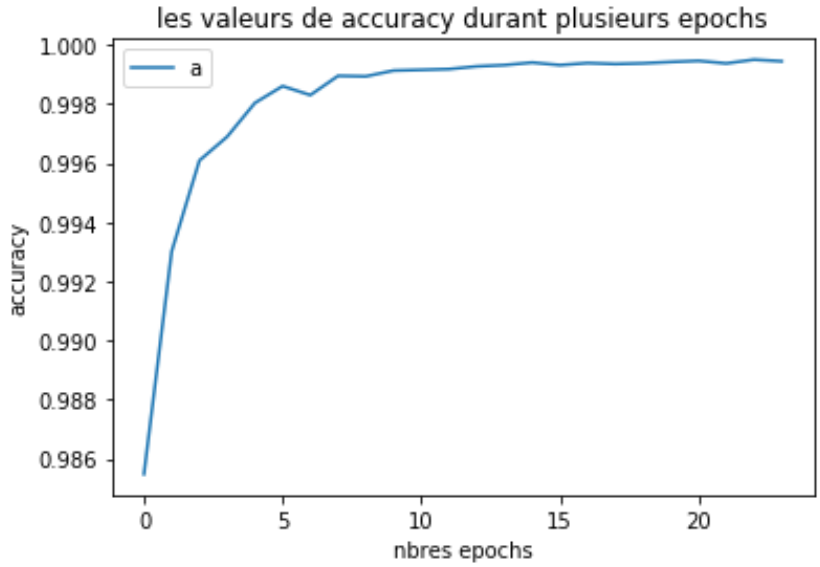


Figure 4.8 : Les valeurs d'accuracy par rapport au nombre d'epochs.

Voici une comparaison du résultat de 150 première ligne de la base de teste et 140 ligne de la prédiction de notre modèle nous voyons que ces identique la couleur bleu représente les valeurs prédites et la couleur rouge représente les valeurs réelles de la base de teste.

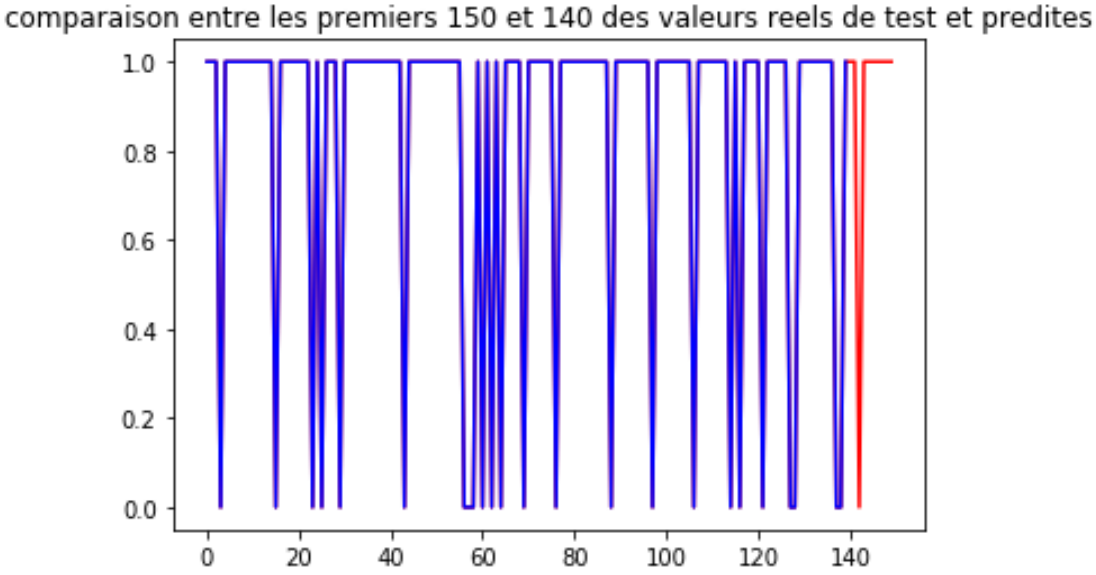


Figure 4.9 : Comparaison de résultat de 150 lignes de base de teste par apport à 140 celui de la prédiction de notre modèle.

Comparaison de notre travail avec d'autres travaux :

Reference	Approach	Accuracy	Dataset
Reddy et al. [35]	SVM	99.95 %	KDD99
B. Inger et al. [36]	ANN	99.67 %	KDD99
Sabhnani et al. [37]	MLP	97.00 %	KDD99
Ying Chung et al. [38]	SSO	93.00 %	KDD99
Notre modèle	LSTM	99.95 %	KDD99

4.11 Conclusion

Dans ce chapitre nous avons développé notre modèle de prédiction sur la base KDD'99, dans le future nous espérons améliorer notre modèle en terme de minimiser les pertes (loss) et d'augmenter le taux de la précision (accuracy), et intégrer un autre modèle pour classifier les type d'intrusion. Et appliquer ce modèle directement sur le trafic réseaux en temps réel.

Conclusion générale :

Aujourd'hui, les systèmes informatiques vont de plus en plus continuer à s'accroitre immiscer d'avantage dans notre vie quotidien. L'évolution de ces systèmes implique de nombreux défis technologiques tels que la mobilité, l'évolutivité et l'autonomie, et la réactivité, etc. parmi ces défis, la question fondamentale de la sécurité demeure un enjeu majeur en raison de la dématérialisation croissante de l'information et des risques de plus en plus importants liés à la complexité de ces systèmes.

Cette mémoire traite la sécurité des systèmes informatiques. La contribution que nous avons réalisée se concentre principalement sur la protection de ces systèmes par la détection d'attaques informatiques, qui provoque aujourd'hui un intérêt croissant autant dans le monde académique que professionnel et surtout industriel en raison de la diffusion de ces menaces (intrusions) informatiques, et qui se font actuellement de plus en plus agressif. La mise en place d'une bonne stratégie de sécurité fasse aux intrusions passe par l'amélioration des techniques de détections des IDS. A cet effet, il faut améliorer les techniques de détection, pour traiter les données de manière efficace et fiable.

Nous avons entamé notre contribution en proposant un modèle artificiel calqué des réseaux des neurones humain, par l'utilisation des techniques du Deep Learning, sur lesquelles nous nous sommes basés pour identifier les intrusions. Pour cerner cette problématique complexe, nous avons proposé une idée qui a dominé notre contribution en se basant sur le fait que l'approche comportementale renforce l'approche par scénario de l'IDS Snort afin de détecter les nouvelles menaces et remédier aux insuffisances du Snort aux terme de faux positifs et faux négatifs, ce qui augmente la sécurité et la robustesse du système informatique.

Annexes

Annexe 1 : les mots clés de l'option ***general.*** [16]

Mot clé	Description
msg	msg indique au moteur de journalisation et d'alerte le message à imprimer avec le vidage de paquets ou l'alerte.
reference	reference permet aux règles d'inclure des références à des systèmes d'identification d'attaque externes.
gid	gid est utilisé pour identifier quelle partie de Snort génère l'événement lorsqu'une règle particulière se déclenche.
sid	Il est utilisé pour identifier de manière unique les règles Snort.
rev	Il est utilisé pour identifier de manière unique les révisions des règles Snort.
classtype	Il est utilisé pour catégoriser une règle comme détectant une attaque faisant partie d'un type plus général de classe d'attaque.
priority	Il attribue un niveau de gravité aux règles.
metadata	Il permet à un rédacteur de règles d'incorporer des informations supplémentaires sur la règle, généralement dans un format de valeur-clé.

Annexe 2 : les mots clés de l'option ***payload***. [16]

Mot clé	Description
content	Il permet à l'utilisateur de définir des règles qui recherchent un contenu spécifique dans la charge utile du paquet et déclenchent une réponse en fonction de ces données.
protected content	Il fournit une grande partie des fonctionnalités du mot-clé de contenu, mais il fonctionne et est utilisé d'une manière très différente.
hash	Il est utilisé pour spécifier l'algorithme de hachage à utiliser lors de la mise en correspondance d'une règle de contenu protégé.
length	Il est utilisé pour spécifier la longueur d'origine du contenu spécifié dans un résumé de règle de contenu protégé. La valeur fournie doit être supérieure à 0 et inférieure à 65536.
nocase	Il permet à l'auteur de la règle de spécifier que Snort doit rechercher le modèle spécifique, en ignorant la case. nocase modifie le mot-clé de contenu précédent dans la règle.
rawbytes	Il permet aux règles d'examiner les données brutes des paquets, en ignorant tout décodage effectué par les préprocesseurs.
depth	Il permet à l'auteur de la règle de spécifier jusqu'où dans un paquet Snort doit rechercher le modèle spécifié. profondeur modifie le mot-clé "content" précédent dans la règle, La depth de 5 indiquerait à Snort de ne rechercher que le modèle spécifié dans les 5 premiers octets de la charge utile.
offset	Il permet au rédacteur de règles de spécifier où commencer la recherche d'un modèle dans un paquet. offset modifie le mot-clé "content" précédent dans la règle.

Pour plus de mots clés voir notre référence [16]

Annexe 3 : les mots clés de l'option ***non-payload.*** [16]

Mot clé	Description
fragoffset	Il permet de comparer le champ de décalage de fragment IP à une valeur décimale. Pour capturer tous les premiers fragments d'une session IP.
ttl	Il est utilisé pour vérifier la valeur de durée de vie IP. Ce mot-clé d'option était destiné à être utilisé dans la détection des tentatives de traceroute.
tos	Le mot-clé tos est utilisé pour vérifier le champ IP TOS pour une valeur spécifique
id	Il est utilisé pour vérifier le champ ID IP pour une valeur spécifique.
ipopts	Il est utilisé pour vérifier si une option IP spécifique est présente. Les options suivantes peuvent être cochées : rr - Record Route, eol - End of list, …
fragbits	Il est utilisé pour vérifier si la fragmentation et les bits réservés sont définis dans l'en-tête IP. les bits suivants peuvent être vérifiés : M - More Fragments, D - Don't Fragment, R - Reserved Bit
dsize	Il est utilisé pour tester la taille de la charge utile du paquet. Cela peut être utilisé pour rechercher des paquets de taille anormale qui pourraient provoquer des débordements de tampon.
flags	Il est utilisé pour vérifier si des bits drapeau TCP spécifiques sont présents.

Pour plus de mots clés voir notre référence [17]

Annexe 4 : les mots clés de l'option ***post-detection***. [16]

Mot clé	Description
logto	Il indique à Snort de journaliser tous les paquets qui déclenchent la règle dans un fichier journal de sortie spécial.
session	Le mot-clé session est conçu pour extraire les données utilisateur des sessions TCP.
resp	Le mot clé resp est utilisé pour tenter de fermer les sessions lorsqu'une alerte est déclenchée.
react	Il implémente une capacité pour les utilisateurs de réagir au trafic qui correspond à une règle Snort en fermant la connexion et en envoyant un avis.
tag	Le mot-clé tag permet aux règles d'enregistrer plus que le seul paquet qui a déclenché la règle.
replace	Remplacez le contenu correspondant antérieur par la chaîne donnée de même longueur. Disponible en mode en ligne uniquement.
detection filter	Effectuez le suivi par adresse IP source ou de destination et si la règle correspond autrement au débit configuré, elle se déclenchera.

Pour plus de mots clés voir notre référence [16]

Annexe 5 : Installation d'un plugin. [17]

Si nous souhaitons développer des plugins Snort, il faut compiler et installer le paquet snort-extras.

Les snort3_extras sont dans leur propre référentiel git. Clonons ce référentiel, compilons et installons :

cd ~/snort_src/

git clone https://github.com/snort3/snort3_extra.git

cd ./snort3_extra/

export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig/./configure_cmake.sh --prefix=/usr/local

cd build

make

sudo make install

Pour que Snort utilise les nouveaux plugins, nous devons lui passer les options **–plugin**path et/ou **–script-path** (les plugins peuvent être écrits en C++ ou en lua).

Par exemple, pour charger le plugin alert_ex :

snort --plugin-path /usr/local/lib/snort_extra -A alert_ex --warn-all

(Le drapeau warn-all est là pour détecter toute erreur). Nous voyons souvent des avertissements, mais il n'y a pas de quoi s'inquiéter.

Si en veut tester un plugin de script lua :

snort --script-path /usr/local/lib/snort_extra -A lualert --warn-all

Pour voir tous les plugins disponibles (à l'exclusion des plugins du package snort-extras), utilisons la commande suivante :

snort --list-plugins

Maintenant, pour avoir snort lister tous les nouveaux plugins, y compris ceux qu'il peut voir dans le répertoire extras :

snort --script-path /usr/local/lib/snort_extra \ --plugin-path /usr/local/lib/snort_extra \ --list-plugins

Le fichier Lisez-moi des extras de Snort 3 et les fichiers source inclus pour plus d'informations.

Pour voir le nombre de nouveaux plugins disponibles dans le dossier extras, montrons tous les modules de journalisation par défaut, puis avec les extras activés :

snort --list-plugins | grep logger

Bibliographie :

[01] Détection d'intrusions : corrélation d'alertes. Article de synthèse, Hervé Debar, Benjamin Morin, Frédéric Cuppens, Fabien Autrel, Ludovic Mé, Bernard Vivinis Salem Benferhat, Mireille Ducassé, Rodolphe Ortalo, Caen, France, 2004.

[02] Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène, Cédric Michel, thèse de doctorat de l'Université de Rennes1,16 Décembre 2003.

[03] Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detction, Jihyn Kim, Jaehyun Kim, Huong Le Thi, Howon Kim, Conference Paper Février 2016.

[04] NT Reseau, IDS et IPS, Baudoin, Karle, 2004

[05] Détection d'intrusions et analyse forensique, Yann Berthier, Jean-Baptiste Marchand.

[06] Les IDS Les systèmes de détection d'intrusions informatiques, Thierry Evangilista, édition DUNOD, Paris 2004.

[07] sécuriser l'informatique de l'entreprise, Enjeux, menaces, prévention et parades, Jean-Marc ROYER, Edition ENI

[08] Système de détection d'Intrusion adaptatif et distribué, Ahmed AHMIM, thèse de doctorat. Université Badji Mokhtar Annaba, 2014.

[09] Sécurité informatique Principes et méthode a l'usage des DSI, RSSI et administrateurs Laurent Bloch & Christophe Wolfhugel.

[10] Le problème de sécurité par le Data Mining thèse de doctorat en science présentée par : Ahmed Chaouki LOKBANI.

[11] A Detailed Analysis of the KDD CUP 99 Data Set, Mahbod Tavallaee & all Proceding of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Application (CISDA 2009).

[12] Introduction à la sécurité informatique Laurent Poinsot UMR 7030 - Université Paris 13 - Institut Galilée Cours " Sécrypt ".

[13] Cryptographie Les Keyloggers CARPENTIER Julien, CHATELAIN Arnaud.

[14] Flow-Based Anomaly Intrusion Detection System Using Two Neural Network Stage, Yousef Abuadlla & all "Computer Science and Information systems 11(2): 601-622: 2012

[15] Intruision recognition using neural networks Vladimir Golovko, PavelKochurko International Scientific Journal of computing, 2005, vol. 4, Issue3, 37-42

[16] SNORT User's Manual The Snort Project (https://www.snort.org/).

[17] Snort3 on Ubuntu 18 & 19Noah Dietrich (https://www.snort.org/).

[18] Artificial Neural Network s for Misuse Detection, Cannady,J, Proceedings, National Information Systems Security Conference (NISSC '98), October, Arlington, VA, pp .443 -456. 1 998.

[19] les réseaux de neurones Mécanique Industriel et Matériaux, G. DREYFUS, septembre 1998

[20] Vers une nouvelle architecture de détection d'intrusion réseaux à base de réseaux neuronaux, Berlin Hervé Djionang Lekagning, Gilbert Tindo,

[21] Les systèmes de détection d'intrusion basés sur du machine Learning, Liran LERMAN, Université libre de Bruxelles

[22] Détection d'intrusions dans les réseaux LAN : Installation et configuration de l'IDS-SNORTTOUATI Azeddine Université Abderrahmane Mira de Béjaïa.

[23] A Deep Learning Approach for Intrusion Detection System in Industry Network, Ahmad HIJAZI, EL Abed EL SAFADI, Jean-Marie FLAUS, Univ. Grenoble Alpes, G-SCOP

[24] Snort 2.1 Intrusion Detection second editions Jay Beale and Snort Development Team Andrew R.Baker Brian Caswell Mike Poor

[25] détection d'intrusion sur les objets connectés par analyse comportementale robin gassais université de montréal.

[26] Introducing reference flow control for intrusion detection at the os level, Zimmermann (Jacob), Mé (Ludovic) et Bidan (Christophe). In : Proceedings of the 5th International Symposium on the Recent Advances in Intrusion Detection (RAID'2002).

[27] Architecture expérimentale pour la détection d'intrusions dans un système informatique, Philippe Biondi, 2001.

[28] Intrusion detection & prevention. Carl F ENDORF, Eugene SCHULTZ et Jim MELLANDER. McGraw-Hill Osborne Media, 2004.

[29] Intrusion detection message exchange requirements. Mark WOOD et Michael A ERLINGER Rapp.tech. 2007.

[30] Intérêts et limites de l'audit de sécurité C. Rousseau Centre de Lorraine, INRS, Vandoeuvre.

[31] A revised taxonomy for intrusion-detection systems, Hervé DEBAR, Marc DACIER et Andreas WESPI. In : Annales des telecommunications. T. 55. 7-8. Springer. 2000, p. 361–378.

[32] Computer immunology Communications of the ACM, Forrest (S.), Hofmeyr (S.A.) et Somayaji (A.) vol. 40, n_ 10, October 1997, pp.88_96.

[33] Non-interference and intrusion detection. Ko (Calvin) et Redmond (Timothy). In : Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 177_187.

[34] A Deep Long Short-Term Memory based classifier for Wireless Intrusion Detection System Sydney Mambwe Kasongo, Yanxia Sun*

[35] Discriminant function for intrusion detection using SVM. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), Reddy, R.R.; Ramadevi, Y.; Sunitha, K.V.N. E_ective Jaipur, India, 21–24 Septembert 2016; pp. 1148–1153.

[36] Performance analysis of NSL-KDD dataset using ANN. Ingre, B.; Yadav, A. In Proceedings of the IEEE International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96.

[37] Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. Sabhnani, M.; Serpen, G. In Proceedings of the International Conference on Machine Learning : Models, Technologies, and Applications (MLMTA), Las Vegas, NV, USA, 23–26 June 2003; pp. 209–215.

[38] A hybrid network intrusion detection system using simplified swarm optimization (SSO). Chung, Y.Y.; Wahid, N. Appl. Soft Comput. **2012**, 12, 3014–3022. [CrossRef]

[W1] http://publib.boulder.ibm.com/tividd/td/TRM/SC23482300/en_US/HTML/adapter_guide06.html.

[W2] https://www.itpro.fr/10-outils-d-evaluation-de-la-securite-du-reseau-a-posseder-absolument/

[W3] https://www.eset.com/fr/cryptojacking/

[W4] http://www.vienne.gouv.fr/

[W5] https://www.pandasecurity.com/

[W6] https://www.nbs-system.com/blog/howto-idsips/

[W7] https://ml4a.github.io/ml4a/fr/neural_networks/