MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Réseaux et Systèmes**

Présenté par:

SADOUN Habib

THEME :

PATTERN MATCHING

POUR LA SÉCURISATION DU TRAFIC RÉSEAU

Soutenu le : 03/07/2022

Devant le jury composé de :

Dr. LAREDJ A.	MCA Université de Mostaganem	Président
Ma. BAHNES N.	MAA Université de Mostaganem	Examinatrice
Dr. BENTAOUZA C.	MCB Université de Mostaganem	Encadrant

Année Universitaire 2021-2022

Résumé

Ce projet traite la détection d'intrusion dans un système informatique pour concevoir une application basée sur le pattern matching et les bases de données afin d'assurer la sécurité de trafic réseau. Cette recherche à nous éclairer les concepts de la sécurité et le mécanisme et les outils pour mieux protégé le système informatique, parmi ces outils se trouve les IDS qui jouent un rôle essentiel pour détecter les intrusions par les différentes méthodes. Les algorithmes de détections des patterns constitue le cœur des IDS, ils sont nombreux chacun d'eux a des avantages et des lacunes. Notre application implémente l'algorithme Aho-Corasick qui est basé sur la recherche des motifs multiples et considérés comme un choix adéquat pour créer un nouvel IDS basé sur les signatures. Nous avons intégré les bases de données pour stocker les patterns et consulter les alertes.

Mots-clés: Base de données, Détection d'intrusion, Pattern matching, Sécurité.

Abstract

This project deals with intrusion detection in a computer system to design an application based on pattern matching and databases to ensure network traffic security. This research to enlighten us the concepts of security and the mechanism and tools to better protect the computer system, among these tools is the IDS quit play a vital role in detecting intrusions by the different methods. Pattern detection algorithms are the core of IDS and there are many of them, each of which has advantages and shortcomings. Our application implements the AhoCorasick algorithm which is based on the search for multiple patterns and considered an adequate choice to create a new IDS based on signatures. We have integrated databases to store patterns and view alerts.

Keywords : Database, Intrusion detection, Pattern matching, Security

تلخيص

يتناول هذا المشروع كشف التسلل في الأنظمة المعلوماتية لتصميم تطبيق يعتمد على مطابقة الأنماط وقواعد البيانات لضمان أمن حركة مرور الشبكة. يسلط هذا البحث الضوء على مفاهيم الأمن وآلية وأدوات حماية النظام ألمعلوماتي بشكل أفضل ، ومن بين هذه الأدوات نظامً IDS الذي يلعب دورا أساسيًا في اكتشاف الاختراقات بطرق مختلفة. تعد خوارزميات اكتشاف الأنماط هي قلب أنظمة اكتشاف التسلل، وهناك العديد منها، ولكل منها مزايا وعيوب. يطبق تطبيقنا خوارزميةً Aho-Corasick التي تعتمد على البحث عن أنماط متعددة وتعتبر خيارا مناسبًا لإنشاء IDSجديد قائم على التوقيع. كما أنه يعتمد قواعد بيانات متكاملة لتخزين الأنماط وعرض التنبيهات.

كلمات مفتاحية : قواعد بيانات ، كشف التسلل ، مطابقة الأنماط ، الأمن.

Dédicace

Á ma chère mère,

Á mon cher père,

Á mes frères,

Á mes sœurs,

Á ma femme et mes enfants,

Á tous ceux qui m'ont encouragétout au long de ce travail.

Remerciements

Je remercie profondément notre professeur Madame Bentaouza Chahinez Meriem a su me diriger tout au long de ce projet de fin d'étude avec patience, rigueur, enthousiasme et bonne humeur.

Je remercie tous mes professeurs du département de l'informatique de l'université Abdelahamid Ibn Badis Mostaganem, que j'ai appris le savoir, le respect. Je n'ai pas cité les noms parce qu'ils tous sons très chers à mon cœur. Je remercie mon ami MENAD Mohamed Bekkai pour les encouragements et le soutien dans mon parcourt de formation.

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

Abréviation	Expression Complète	Page
DMZ	Dense Multicast Zone	22
HIDS	Host Intrusion Detection System	25
NIDS	Network Intrusion Detection System	27

Table des matières

Introduction générale

La découverte de l'internet et l'implémentation des réseaux informatiques dans les différentes infrastructures administratives, industrielles et les opérateurs de télécommunications imposent l'échange énorme des informations via ces réseaux.

La sensibilité et l'importance de ces données soient personnelles soit organisationnels risque d'être divulguer, ce qui rendre la sécurité un facteur majeur qui doit être respecté.

Face à ce défi, les organisations ont pris des mécanismes et des outils nécessaire sur différents niveaux (organisationnels, matériels et logiciels) pour se protéger contre les menaces et les attaques informatiques.

L'objectif de la sécurité est de garantir que les critères de la sécurité sont respecter, soit sur la machine hôte ou sur le trafic réseau.

Ce projet vient d'éclairer les concepts de la sécurité au niveau des réseaux informatiques en exploitant le pattern matching. Le pattern matching est un sujet ancien traite les correspondances des sous-chaînes dans les blocs de textes, exploités par divers outils de recherche des occurrences des mots-clés tel que qrep sous Linus ou les logiciels de traitement de texte etc.

La problématique ces comment profiter de ce concept afin de sécuriser le trafic réseau ? Pour atteindre ces objectives nous proposons la démarche suivante :

Dans le chapitre 1 nous éclairons les concepts de base de la sécurité informatique, ces critères, ces mécanismes et ces outils d'assurer la protection des données, dans le chapitre 2 nous expliquons ces quoi les systèmes de détection des intrusions ? ces méthodes de détection avec ces avantages et ces inconvénients. Comme les paquets capturés ont des données qui peuvent être analysées pour trouver des patterns suspects qui indiquent qu'une intrusion, nous sommes face de les détecter pour éviter une menace réelle. La détection des patterns fait par l'exploitation des algorithmes de

correspondances des motifs. Dans le chapitre3 nous expliquant ces algorithmes, son concept, son fonctionnement et une comparaison de performance afin de trouver un algorithme adaptable a notre projet.

Tout ce travaille doit être réalisé, testé et analysé les résultats qui doivent représenter dans le chapitre 4, où nous présentons notre application de détection des patterns, ces processus et ces différentes interfaces.

Comme nous avons testé notre application sur trois algorithmes Aho-Corasick (pyahocorasick, ahocorapy, py-aho-corasick) implémenter de manière différente, sur des machines virtuelles afin d'avoir de détecter des patterns en temps réel.

Nous finissons avec une conclusion générale de notre travail et les futurs perspectifs qui peuvent améliorer notre application.

Chapitre 1 Sécurisation du trafic réseau

1.1 Introduction

Le trafic réseau désigne les communications transportées entre hôtes sur le réseau informatique soit de réseaux câblés ou sans-fil [1]. Ces communications doivent transporter des données sous différents formats suivant la couche de réseaux, dans cette transmission le facteur majeur qui doit garantie c'est la sécurité. Pour cela, le chapitre est compose par une introduction des concepts et des objectifs de la sécurité informatiques dans la *section 1.1* et les différentes attaques et les mécanismes et les mesures de protection dans la *section 1.2*.

1.2 Concepts de la sécurité informatique

1.2.1 Définitions

2. LA DIFFERENCE ENTRE LA SECURITE ET LA SURETE

   La sureté (Safety) correspond à la démarche, les méthodes et les dispositions, visant a limiter les risques de nature accidentelle (sans malveillance), étant susceptibles d'avoir des répercussions sur l'environnement du système, et la sécurité (Security) correspond aussi à la démarche, méthodes et dispositions, mais pour limiter les risques de nature malveillante (provenant d'une entité voulant nuire)[2].

3. SECURITE DE SYSTEME D ’INFORMATION

   La sécurité des systèmes d’information (System information Security) est l’ensemble des mesures techniques et non techniques de protection permettant à un système d’information de résister à des événements susceptibles de compromettre des données stockées, traitées ou transmises et des services connexes que ces systèmes offrent ou qu’ils rendent accessibles.

4. SECURITE INFORMATIQUE

   La sécurité informatique (Computer Security) consiste à assurer l’accès aux ressources informatiques est garantie que seules les personnes ou autres systèmes autorises.

5. SECURITE D ’INFORMATION

   La sécurité de l’information (Information Security) se focalise sur la protection de l’information, comprend les mesures qui permettent d’implémenter et d’assurer la sécurité des systèmes d’information incluant le système informatique.

1.2.2 Objectives de la sécurité

La sécurité à plusieurs objectifs peut être prise en compte, on trouve les objectifs suivants :

1. CONFIDENTIALITE (CONFIDENTIALITY)

   La confidentialité est le maintien du secret des informations avec une protection des données contre une divulgation non autorisée par un mécanisme de contrôle d’accès et des procédures de chiffrement.

2 INTEGRITE (INTEGRITY)

   L’intégrité permet de certifier que les données, les traitements ou les services n’ont pas été modifiés, altérés ou détruit tant de façon intentionnelle ou accidentelle.

3 DISPONIBILITE (AVAILABILITY)

   La disponibilité permet qu’une ressource doive être accessible, avec un temps de réponse acceptable.

4 AUTHENTIFICATION (AUTHENTICATION)

L'authentification CONSISTE en l'identification de la source d'une donnée et la garantie de l'authenticité de la source.

5 NON-REPUDIATION (NON-REPUDIATION)

La non-répudiation est le fait de ne pouvoir nier ou rejeter qu'un événement a eu lieu. A cette notion sont associées.

1.2.3 Les attaques des réseaux informatiques

Les données sensibles peuvent avoir des nuire par des attaquants chacun a sa motivation (vols des données, espionnage,...), voir la figure 1.

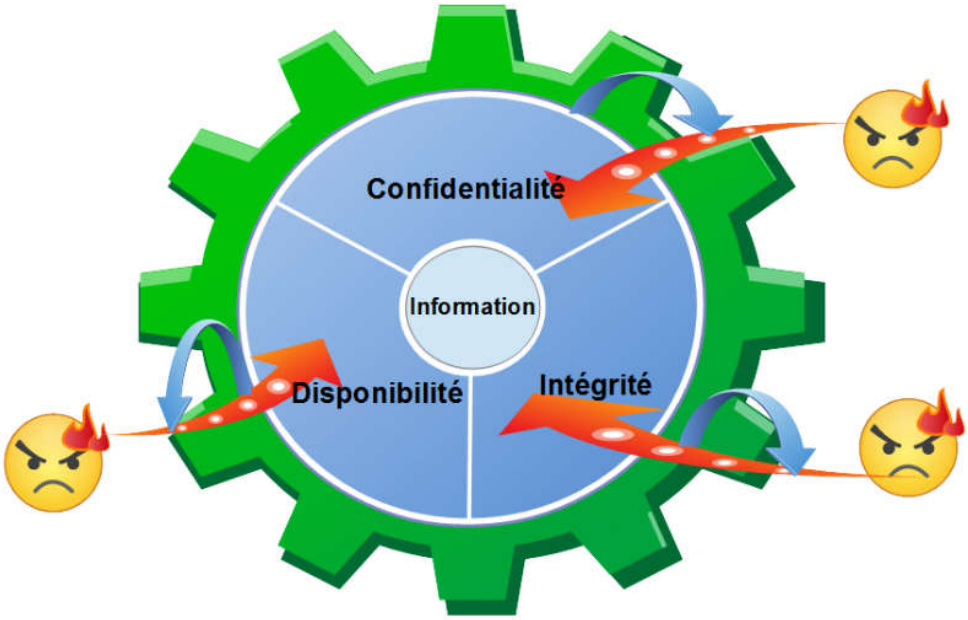


Figure 1 - Motivation d'une attaque

A fin de comprendre les attaques informatiques on doit éclairer les notions suivantes :

1. VULNERABILITES

Une vulnérabilité représente une faille, brèche, ou faiblesse inhérente à une entité (logicielle ou matérielle). C'est une faiblesse dans la sécurité d'un système informatique qui permet a un utilisateur de faire une action malveillante. Elle peut être exploitée pour différentes raisons afin d'affecter plusieurs objets.

Exemples des vulnérabilités :

• vulnérabilités logicielles ou matérielles;

• vulnérabilités de spécification, architecture, codage, ...;

• vulnérabilités d'utilisation (configuration, administration, déploiement, ...);

• vulnérabilités liée à l'utilisateur (mot de passe faible, non-respect des bonnes pratiques, ...).

2. MENACES

Une menace est un ensemble de circonstances qui pourraient causer des dommages. Elle peut être un danger interne ou externe. Une menace peut aussi être définie comme une violation d'un ou plusieurs critères de sécurité.

Exemples de menaces :

• acte volontaire : écoute de trafic, usurpation d'identité, ...;

• accident : panne électrique, bug,

3. ATTAQUES INFORMATIQUE

3.1.1. *Définition d'une* attaque

Une attaque est l'exploitation d'une vulnerabilités d'un système informatique a des fins non connues [3], par la divulgation non autorisé des ressources informatiques.

3.1.2. *Développement d'une attaque*

Une attaquant recherche des vulnerabilités dans les systèmes informatiques et qui se représente une menace a exploité montrer par la figure 2.

Figure 2 - Développement d'une attaque

3.1.3. *Types d'attaques*

Ces attaques sont pour la plupart lancées automatiquement a partir des machines infectées (par des virus, chevaux de Troie, vers, etc.) montrer dans la figure 3 [4].

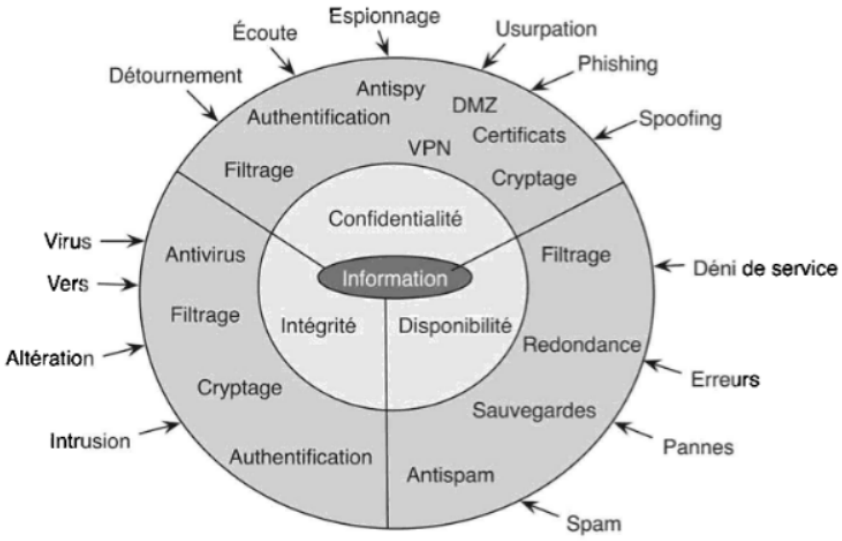


Figure 3 - Types d'attaques

Selon le critère de l'action de l'attaquant il existe deux types d'attaque active et passive.

i. **Attaques passives**

Les attaquants passifs sont principalement intéressées par le vol d'informations sensibles à l'insu de la victime voir la figure 4.

Les attaques passives sont difficiles à détecter.

Exemples des attaques passives qui existent :

- analyse du trafic : l'attaquant détecte le chemin de communication entre l'expéditeur et le récepteur;
- interception : l'attaquant peut lire les données confidentielles, mais il ne peut pas les éditer ou les modifier.

Figure 4 – Interception

ii. **Attaques actives**

Les attaques actives consistent a manipuler le trafic pour injecter, altérer ou supprimer des données [5]. En conséquence, la victime interrompt la communication avec l'autre fête. Certaines des attaques actives sont les suivantes.

A. Modification:

Il s'agit d'une attaque contre l'intégrité, comme le montre la figure 5.
Exemple : certaines altérations de la route de routage sont effectuées par le nœud. Cela oblige l'expéditeur à envoyer des messages par le long chemin, ce qui provoque un retard de communication.

b)

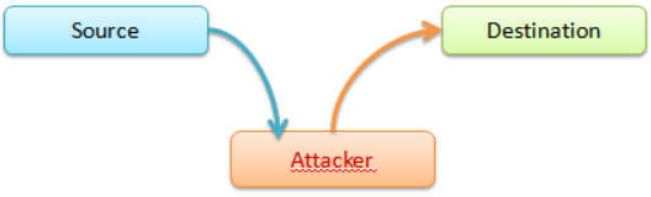


Figure 5 - Modification

B. Fabrication

Un nœud malveillant génère un faux message de routage qui provoque le génération d'informations incorrectes sur l'itinéraire entre les nœuds. C'est une attaque contre l'authenticité comme le montre la figure 6.

Figure 6 -Fabrication

C. Spoofing

Un nœud malveillant présente mal son identité afin que l'expéditeur change sa topologie comme le montre la figure 7 [6].

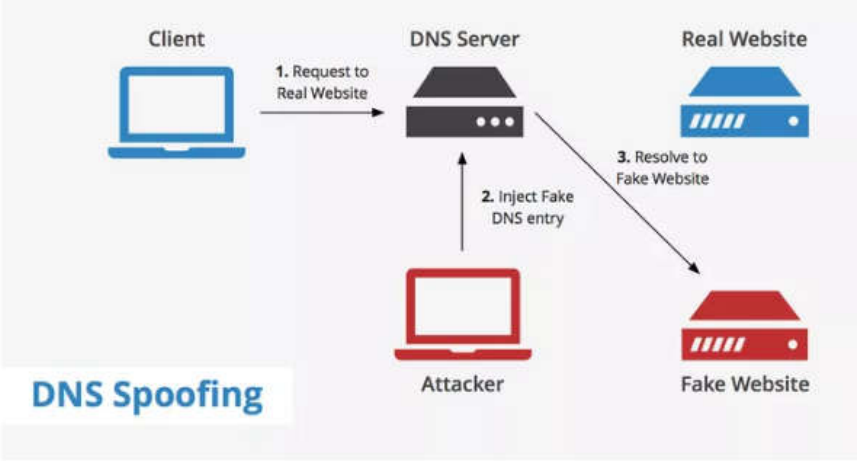


Figure 7 - Spoofing

D. Dénis de service

Un nœud malveillant envoie un message au nœud et consomme la bande passante du réseau comme indiqué dans la figure 8 [7].

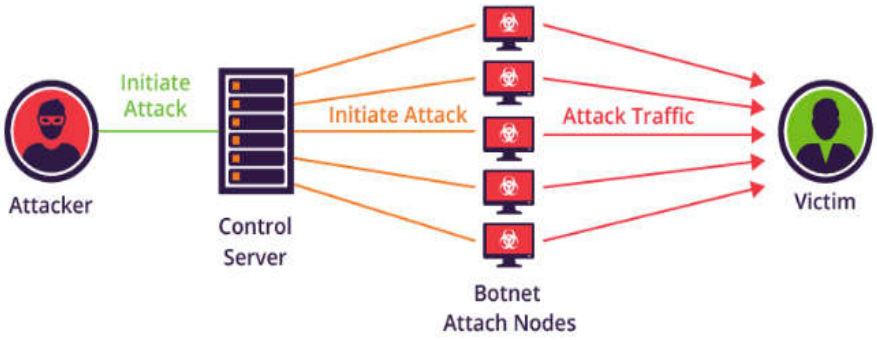


Figure 8 - Dénis de Service

1.3 Mécanismes et mesures d'assurer la sécurité informatiques

1.3.1 Mécanismes de la sécurité

Un mécanisme de sécurité conçut pour détecter, prévenir et lutter contre une attaque de sécurité. Les mécanismes de sécurité sont divisés en trois grandes catégories : les mécanismes de prévention, les mécanismes de détection, et les mécanismes de recouvrement.

1. PREVENTION

   La prévention consiste à prévenir l'arrivée des attaques. Son objectif était de faire échouer les attaques. Les mécanismes de prévention peuvent être :

   — la sureté de fonctionnement : renforcer la qualité du logiciel (méthodes

formelles, analyse statique, ...);

— le chiffrement et la cryptographie.

2. DETECTION

   La détection consiste à détecter l'arrivée des attaques ou des intrusions. Les mécanismes de détection peuvent être :

   — la journalisation et audit;

   — la détection d'intrusion;

   — la supervision de sécurité;

   — le contrôle d'intégrité.

3. RECOUVREMENT

   Le recouvrement ou récupération consiste à récupérer après l'arrivée d'une attaque. Les mécanismes de recouvrement peuvent être :

   — l'antivirus;

   — la détection d'intrusion avec sauvegarde.

1.3.2 Mesures d'assurer la sécurité dans les réseaux informatiques

La figure 9 montre les niveaux de sécurité qui nous devons les assurés. Elles sont détaillées comme suit :

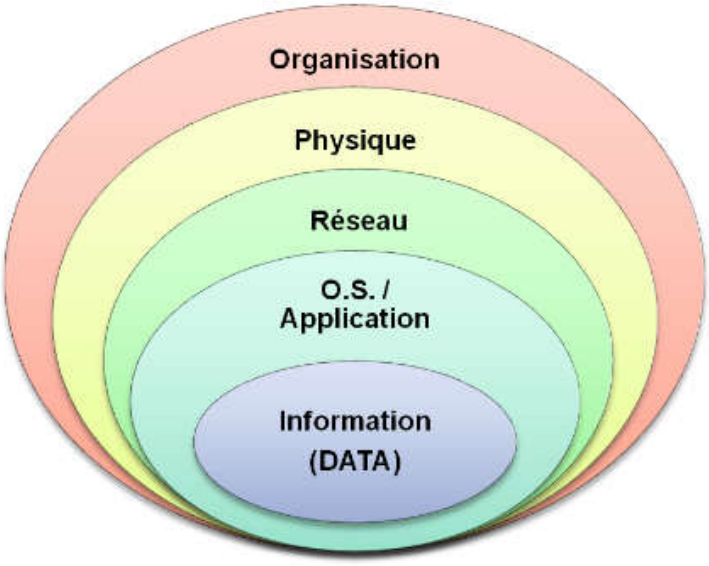


Figure 9 - Niveaux de sécurité informatique

1. SECURITE AU NIVEAUX D 'ORGANISATION

   Les étapes de mise en œuvre de la sécurité d'entreprise suivent généralement les phases suivantes :

   (a) **Analyse de risques :** consiste a identifier les ressources a protéger, les menaces qui présentent sur ces ressources et évaluer l'occurrence de ces menaces.

   (b) **définir une politique de sécurité qui doit comprend de :**

   — définir qui est autoriser et les privilégies d'utilisation des ressources.

   — sensibiliser les utilisateurs de l'importance de la sécurité

2. SECURITE AU NIVEAUX PHYSIQUE

   — Mettre les mécanismes de sécurité permettant de protéger les ressources matériels de la manière la plus efficace (protection, verrou)

3. SECURITE AU NIVEAUX DE RESEAU

   La sécurité de réseau consiste à mis en place des:

   (a) pare-feu,

   (b) système de détection d'intrusion,

   (c) DMZ,

   (d) VPN...

4. SECURITE AU NIVEAUX DES LOGICIELLES (OS/APPLICATIONS)

   La sécurité logicielle consiste à utiliser des mécanismes de sécurité logicielles,

tels que :

(a) installer des Anti-virus

(b) faire des mises ajoures (updates) des systèmes d'exploitation .

5. SECURITE AU NIVEAU DES DONNEES

La sécurité de données consiste pour protéger les données. On y retrouve les mécanismes de sécurité suivants :

(a) **Chiffrement** : algorithmes généralement basés sur des clefs et transformant les données. Sa sécurité est dépendante du niveau de sécurité des clefs.

(b) **Signature numérique** : données ajoutées pour vérifier l'intégrité ou l'origine des données.

1.4 Conclusion

Dans ce chapitre on a survole sur les concepts de sécurité et ses applications, comme on a détaillé les différentes attaques et les mesures de protection, ces mesures sont différent et chacun apporte un potentiel majeur dans la sécurité du trafic de réseaux. Parmi eux les systèmes de détection d'intrusion qui est notre point pour le développer dans le chapitre 2.

Chapitre 2 Système de détection d'intrusion (IDS)

2.1 Introduction

Les premières recherches sur les IDS apparues en 1980, par P. Anderson «Computer security threat monitoring and surveillance » a mené des recherches sur la surveillance des menaces et ont propose que les pistes d'audit soient utilisées pour les surveiller [8]. L'importance de ces données n'était pas comprise à l'époque et toutes les procédures de sécurité du système disponibles visaient a refuser l'accès aux données sensible provenant d'une source non autorisée. Plus tard, en 1987 Dorothy[9] a proposé le concept de détection d'intrusion comme solution au problème de fournir un sentiment de sécurité dans les systèmes informatiques. Dans ce chapitre, nous présentons les systèmes de détection d'intrusion dans la section 1 et les outils de mise en œuvre des IDS dans la section 2.

2.2 Présentation de Système de détection d'intrusion

2.2.1 Définitions

1. DETECTION D'INTRUSION (INTRUSION DETECTION)

Il s'agit des tentatives actives incessantes de découverte ou de détection de la présence des activités intrusives. Ce faite référence à tous les processus utilises pour découvrir les utilisations non autorisées du réseau ou des périphériques informatiques. Ceci est atteint grâce à un logiciel spécialement conçu dans le seul but de détecter une activité inhabituelle ou anormale [10].

2. SYSTÈME DE DETECTION D'INTRUSION (IDS)

Les systèmes de détection d'intrusion font référence aux systèmes conçus pour surveiller l'activité d'un agent afin de déterminer si l'agent présente un comportement inattendu. « L'objectif général d'un IDS est d'informer le personnel informatique qu'une intrusion réseau peut avoir lieu. Les informations d'alerte comprendront généralement des informations sur l'adresse source de l'intrusion, l'adresse cible/victime et le type d'attaque suspectée» [11].

3. FAUX POSITIF ET FAUX NEGATIF

On parle de **faux positif** lorsque un paquet ne consister aucune menace réelle est détecté comme tentative d'intrusion, et le **faux négatif** lorsqu'une tentative d'intrusion n'est pas détectée.

La détection d'intrusion est le processus de surveillance des ordinateurs ou des réseaux pour détecter toute entrée, activité ou modification de fichier non autorisée. Un système de détection d'intrusion est un système qui tente d'identifier les intrusions, cette l'utilisation non autorisée, des utilisations abusives ou des abus de systèmes informatiques par l'un ou l'autre.

2.2.2 Principaux types d'IDS

La façon la plus courante de classer les IDS est de les regrouper par sources d'information. Certains IDS analysent les paquets de réseau capturés pour trouver des attaquants. D'autres IDS analysent sources d'information générées par le système d'exploitation ou l'application logicielle de signalisation d'intrusion.

1. IDS BASES SUR LE RESEAU (NIDS)

Les NIDS détectent les attaques en capturant et en analysant les paquets de réseau par un ensemble de capteurs à usage unique ou les hôtes placés en divers points d'un réseau . Le NIDS renifle l'interface interne du pare-feu en mode lecture seule et envoie des alertes a un serveur de gestion NIDS Management

via une interface réseau différente montrer par la figure 10[12] .

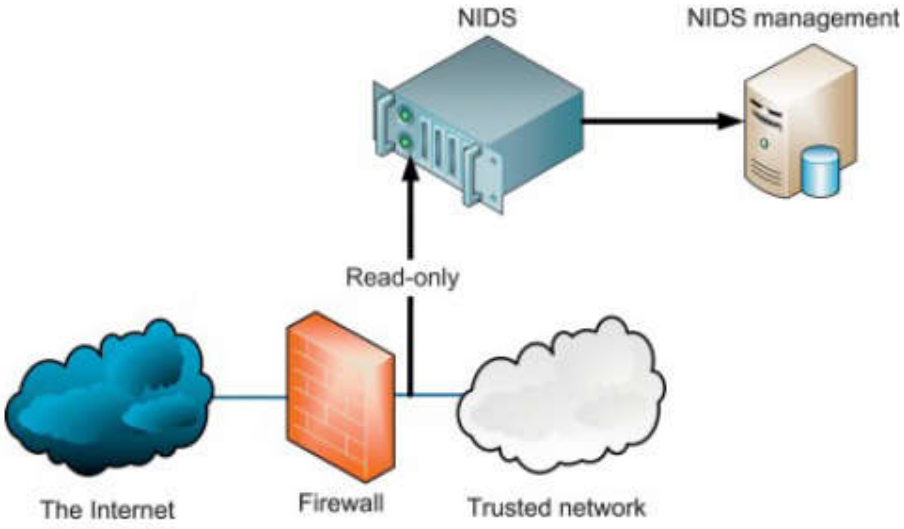


Figure 10 - – Implémentation de NIDS

Avantages :

— les NIDS bien places basés sur le réseau peuvent surveiller un grand réseau.

— Les NIDS sont des appareils passifs (écoute le trafic) sans interférer avec le fonctionnement normal d'un réseau.

— Les NIDS peuvent être très sécurisés contre les attaques.

— Le déploiement a peu d'impact sur un réseau existant.

Inconvénients :

— Les NIDS peuvent avoir des difficultés à traiter tous les paquets dans un réseau étendu ou occupé et, par conséquent, peuvent ne pas reconnaître une attaque lancée pendant les périodes de fort trafic.

— La plupart des commutateurs ne fournissent pas de ports de surveillance universels, ce qui limite la plage de surveillance d'un capteur NIDS à un seul hôte. Même quand les commutateurs fournissent de tels ports de surveillance, souvent le port unique ne peut pas refléter tout le trafic traversant le commutateur.

— Les NIDS ne peuvent pas analyser les informations cryptées.

— Certains NIDS ont des problèmes face aux attaques basées sur le réseau qui impliquent la fragmentation des paquets. Ces les paquets mal formes rendent

les IDS instables et se bloquent.

a. IDS BASES SUR L'HOTE (HIDS)

Les HIDS fonctionnent sur des informations collectées à partir d'un système informatique individuel. Ils utilisent des sources d'information de deux types, pistes d'audit du système d'exploitation (g généralement générées au niveau le plus interne (noyau) du système d'exploitation) et journaux système. La figure 11[13] représente l'implémentation d' HIDS dans un réseau informatique.

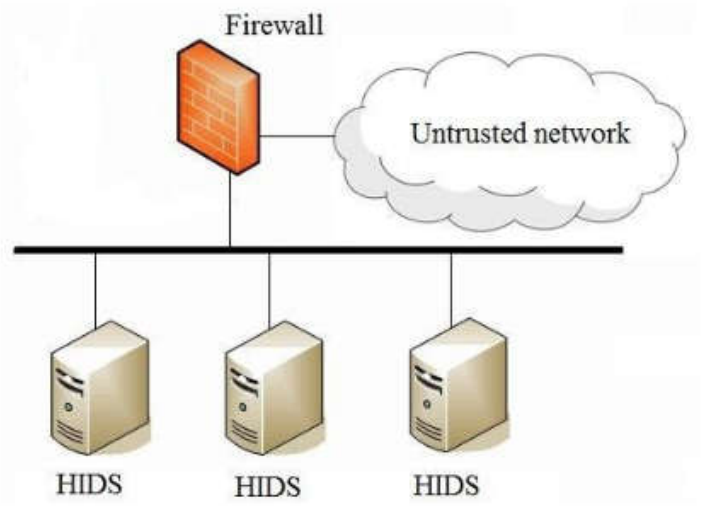


Figure 11 – Implémentation des HIDS

Avantages :

— Les HIDS, avec leur capacité a surveiller les événements locaux â un hôte, peuvent détecter des attaques qui ne peuvent pas être vues par un NIDS.

— Les HIDS peuvent souvent fonctionner dans un environnement dans lequel le trafic réseau est chiffré, lorsque les sources d'information basées sur l'hôte sont générées avant que les données ne soient chiffrées et/ ou après que les données soient déchiffrées sur l'hôte de destination

— Les HIDS ne sont pas affectes par les réseaux commutés.

— Lorsque les HIDS fonctionnent sur des pistes d'audit du système d'exploitation, ils peuvent aider a détecter les chevaux de Troie ou d'autres attaques

impliquant des violations de l'intégrité des logiciels.

Inconvénients :

— Les HIDS sont plus difficiles a gérer, car les informations doivent être configures et gérées pour chaque hôte surveillé.

— ·Les HIDS ne sont pas bien adaptes pour détecter les balayages de réseau ou toute autre surveillance de ce type qui cible un réseau entier, car l'IDS ne voit que les paquets réseau reçus par son hôte. ˆ

— Les HIDS peuvent être désactivés par certaines attaques déni de service.

— Lorsque les HIDS utilisent des pistes d'audit du système d'exploitation comme source d'information, la quantité d'informations peut être immense, nécessitant un stockage local supplémentaire sur le système.

— Les HIDS utilisent les ressources informatiques des hôtes qu'ils surveillent, infligeant ainsi un cout de performance aux systèmes surveillés.

2.2.3 Fonctionnement des IDS

Il existe plusieurs types d'IDS, caractérisés par différentes approches de surveillance et d'analyse. Chaque approche présente des avantages distincts et des inconvénients. Toutes les approches peuvent être décrites en matière d'un générique modèle de processus pour les IDS qui peuvent être décrits en matière de trois fonctions composantes fondamentales : la collection des données, l'analyse des éventements et la réaction attendue.

1. COLLECTION DES DONNEES (DATA COLLECTION)

   Tout d'abord, les données utilisées pour déterminer les intrusions sont collectées. Les données de trafic réseau peuvent également être collectées â l'aide d'un logiciel comme TCPDUMP. Comme la quantité de données collectées est énorme, elles ne seront utiles que lorsqu'elles seront filtrées pour que de petits groupes ne contiennent que les données utiles pour détecter l'intrusion.

   Les NIDS peuvent inclure des en-têtes de paquets IP, qui contiennent les adresses source et de destination, la longueur des paquets et le type de protocole de la couche transport. Pour les HIDS, il peut inclure des informations sur

l'utilisateur telles que l'heure de connexion, la durée de la session et les fichiers utilisés.

2. ANALYSE LES EVENEMENTS

Il existe deux approches principales montrées par la figure 12 [14] pour analyser les événements afin de détecter les attaques : la méthode de détection d'abus et la méthode de détection d'anomalies.

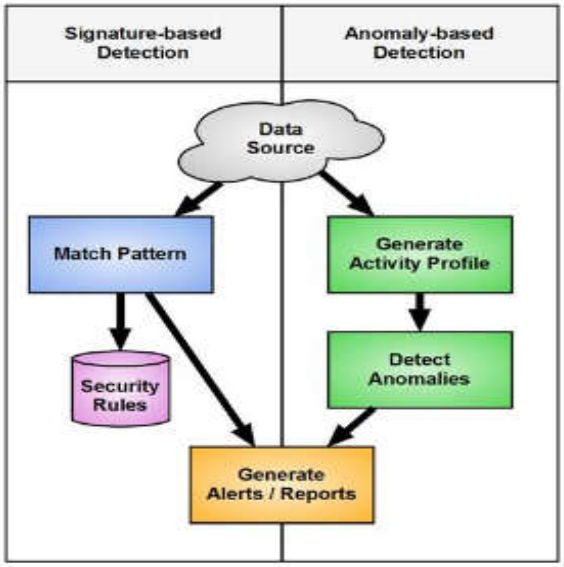


Figure 12 - Méthodes de détections

(a) Méthode de détection d'abus

Les détecteurs d'abus analysent l'activité du système, recherchant des événements ou des ensembles d'événements qui correspondent à un modèle prédéfini d'événements décrivant une attaque connue. Comme les modèles correspondant aux attaques connues sont appelés des signatures, la détection des abus est parfois appelée la détection basée sur les signatures. La forme la plus courante de détection d'abus spécifie chaque modèle d'événements correspondant à une attaque sous la forme d'une signature distincte. Cependant, il existe des approches plus sophistiquées de détection des abus (appelées techniques d'analyse basées sur l'état) qui peuvent exploiter une seule signature pour détecter des groupes d'attaques.

Avantages :

— Les détecteurs d'abus sont très efficaces pour détecter les attaques sans générer de fausses alarmes.

— Les détecteurs d'abus peuvent diagnostiquer rapidement et de manière fiable l'utilisation d'un outil ou technique d'attaque spécifique. Cela peut aider les responsables de la sécurité à hiérarchiser les mesures correctives.

— Les détecteurs d'abus peuvent permettre aux responsables systèmes, quel que soit leur niveau d'expertise en matière de sécurité, de suivre les problèmes de sécurité sur leurs systèmes, en lançant des procédures de traitement des incidents.

Inconvénients :

— Les détecteurs d'abus ne peuvent détecter que les attaques dont ils ont connaissance. Ils doivent donc être constamment mis â jour avec les signatures des nouvelles attaques.

— De nombreux détecteurs d'abus sont conçus pour utiliser des signatures qui les empêchent de détecter des variantes d'attaques courantes.

(b) Méthode de détection d'une anomalie

Les détecteurs d'anomalies identifient les comportements inhabituels anormaux (anomalies) sur un hôte ou un réseau. Ils fonctionnent sur l'hypothèse que les attaques sont différentes de l'activité « normale» « légitime» et peuvent donc être détectées par des systèmes qui identifient ces différences. Les détecteurs d'anomalies construisent des profils représentant le comportement normal des utilisateurs, des hôtes ou des connexions réseau. Ces profils sont construits a partir de données historiques collectées sur une période de fonctionnement normal.

Les détecteurs collectent ensuite des données d'événement et utilisent une variété de mesures pour déterminer quand l'activité surveillée s'écarte de la norme.

Avantages :

— Les IDS bases sur la détection d'anomalies détectent les comportements inhabituels et ont ainsi la capacité de détecter les symptômes d'attaques sans connaissances spécifiques des détails.

— Les détecteurs d'anomalies peuvent produire des informations qui peuvent a leur tour être utilisées pour définir des signatures pour les détecteurs d'abus.

Inconvénients :

— Les approches de détection d'anomalies produisent généralement un grand nombre de fausses alarmes en raison des comportements imprévisibles des utilisateurs et des réseaux

— Les approches de détection d'anomalies nécessitent souvent des « training sets» étendus d'enregistrements d'événements systèmes afin de caractériser les modèles de comportement normaux.

Il y a des forces et des faiblesses associées â chaque approche, et il semble que les IDS le plus efficaces utilisent principalement des méthodes de détection d'abus avec une poignée de composants de détection d'anomalies.

3. LA REACTION DES IDS

Une fois que les IDS ont obtenu des informations sur les événements et les ont analysées pour trouver des symptômes d'attaques, ils génèrent des réponses.
Les réponses sont classées comme réponses actives, réponses passives ou un mélange des deux.

a. **Réponses actives**

Les réponses IDS actives sont des actions automatisées prises lorsque certains types d'intrusions sont détectés. Il existe trois catégories de réponses actives :

i. Collecter des informations supplémentaires sur une attaque suspectée:

consiste a collecter des informations supplémentaires sur une attaque

suspectée. Cela peut impliquer d'augmenter le niveau de sensibilité des sources d'information (par exemple, augmenter le nombre d'événements consignes par une piste d'audit du système d'exploitation, ou augmenter la sensibilité d'un moniteur réseau pour capturer tous les paquets, pas seulement ceux ciblant un port ou un système cible particulier.). Les informations supplémentaires collectées peuvent aider à résoudre la détection de l'attaque. (Aider le système a diagnostiqué si une attaque a eu lieu ou non.).

ii. *Modifier l'environnement*

consiste a stopper une attaque en cours, puis a bloquer l'accès ultérieur de l'attaquant. En règle générale, les IDS n'ont pas la capacité de bloquer l'accès d'une personne spécifique, mais bloquent à la place les adresses

IP (Internet Protocole) d'ou l'attaquant semble provenir en prenant les mesures suivantes :

— injecter des paquets de réinitialisation TCP dans la connexion de l'attaquant au système victime, mettant ainsi fin a la connexion;

— reconfigurer les routeurs et les pare-feu pour bloquer les ports réseau,

les protocoles ou les services utilises par un attaquant, Injecter des paquets de réinitialisation TCP dans la connexion de l'attaquant au système victime, mettant ainsi fin a la connexion;

— reconfiguration des routeurs et des pare-feu pour bloquer les paquets

provenant de l'emplacement apparent de l'attaquant (adresse IP ou site),

iii. *Agir contre l'intrus*

Certains IDS sont conçus pour générer des alarmes et des alertes, en les signalant a un système de gestion de réseau. Ceux-ci utilisent des déroutements et des messages SNMP pour envoyer des alarmes et des alertes aux consoles centrales de gestion du réseau, où ils peuvent être traités par le personnel d'exploitation du réseau.

b. **Réponses passives**

Les réponses IDS passives fournissent des informations aux utilisateurs du système, en s'appuyant sur les humains pour prendre des mesures ultérieures en fonction de ces informations, les réponses passives sont :

i. Alarmes et notifications

Des alarmes et des notifications sont générées par les IDS pour informer les utilisateurs lorsque des attaques sont détectées. La forme d'alarme la plus courante est une alerte à l'écran ou une fenêtre contextuelle. Ceci est affiche sur la console IDS ou sur d'autres systèmes comme spécifiés par l'utilisateur lors de la configuration de l'IDS. Un autres ensemble d'options qui sont utiles aux grandes organisations ou aux organisations distribuées sont celles qui impliquent la notification à distance des alarmes ou des alertes.

ii. Capacités de production de rapports et d'archivage

De nombreux IDS offrent des capacités de génération de rapports de routine et d'autres documents d'information détaillées. Certains d'entre eux peuvent produire des rapports d'événements système et d'intrusions détectés au cours d'une période de rapport particulier.

2.3 Outils utilises pour la mise en œuvre des IDS

Il existe plusieurs outils de mettre en œuvre un system de détection d'intrusion par choix de notre sujet «Pattern matching »; nous choisissant l'application SNORT qui est plus adéquate pour mieux comprendre de la méthode de détection basé sur la signature.

2.3.1 Outil SNORT

SNORT est une application open source de détection d'intrusion réseau et système de prévention [15]. Ce peut l'analyse du trafic en temps réel et le flux de données dans le

réseau. Il est capable de vérifier l'analyse du protocole et peut détecter différents types d'attaque. SNORT vérifie essentiellement le paquet contre la règle écrite par l'utilisateur. Les règles de SNORT peuvent être facilement lu et les peuvent également être modifiées. SNORT analyse le trafic en temps réel, chaque fois qu'un paquet arrive sur le réseau puis il vérifie le comportement du réseau si les performances des dégradations du réseau puis SNORT arrêtent le traitement du paquet, rejette le paquet et stockent ses détails dans la base de données des signatures.

2.3.2 Les Composants de SNORT

SNORT est essentiellement la combinaison de plusieurs composants. Tous les composants fonctionnent en ensemble pour trouver une attaque particulière puis prendre l'action correspondante. Fondamentalement, il consiste a suivre les principaux composants comme indique sur la figure 13 [16] :

1. Décodeur de paquets
2. Préprocesseurs
3. Moteur de détection
4. Système de journalisation et d'alerte
5. Modules de sortie

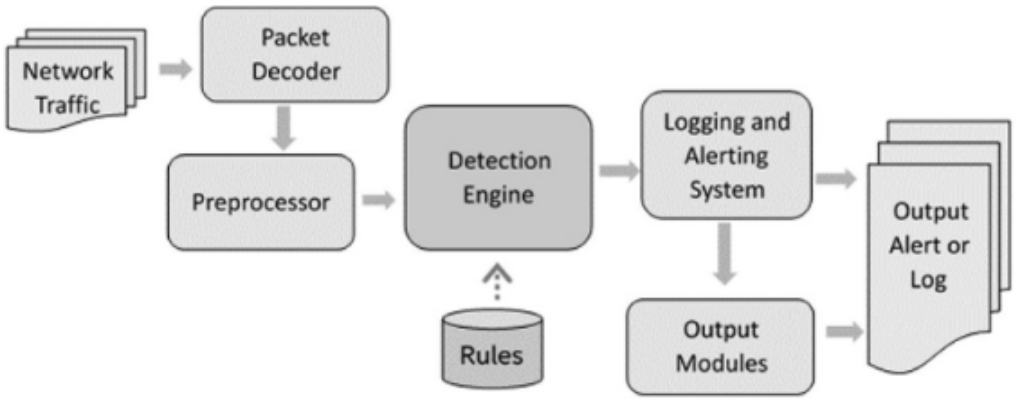


Figure 13 – Composants de SNORT

1. DECODEUR DE PAQUETS

Le décodeur de paquets collecte les paquets provenant de différentes interfaces réseau, puis les envoie au préprocesseur ou les envoie au moteur de détection. L'interface réseau peut être Ethernet, SLIP, ppp, etc. ˆ

2. PREPROCESSEURS

Cela fonctionne avec SNORT pour modifier ou organiser le paquet avant que le moteur de détection n'applique une opération sur le paquet si le paquet est corrompu. Parfois, ils génèrent également une alerte en cas d'anomalies détectées dans le paquet. Le préprocesseur effectue une tâche très importante, à savoir la défragmentation. Lorsque l'intrus divise la signature en deux parties et les envoie en deux paquets, avant de versifier la signature, les deux paquets doivent être défragmentés et alors seulement la signature peut être trouvée.

3. LE MOTEUR DE DÉTECTION

Son travail principal est de découvrir les sorties d'activités d'intrusion dans le paquet a l'aide des règles SNORT et, si elles sont trouvées, d'appliquer la règle appropriée, sinon il abandonne le paquet.

4. SYSTÈME DE JOURNALISATION ET D 'ALERTE

Quel que soit le moteur de détection trouvé dans le paquet, il peut générer une alerte ou être utilisé pour enregistrer l'activité. Tous les fichiers journaux sont conserves. Il contrôle principalement les différentes sorties dues au système de journalisation et d'alerte.

5. LES MODULES DE SORTIE

peuvent effectuer les opérations suivantes en fonction de la configuration :

— se connecter simplement au fichier /var/log/SNORT/alertes ou a un autre fichier

— Envoi d'interruptions SNMP

— Envoi de messages a la fonction syslog

— Peut générer des messages SMB de sortie XML vers des machines Microsoft Windows

2.3.3 Modélisation des signatures pour SNORT

SNORT permet de modéliser des signatures en une seule étape en règles. En SNORT, les signatures sont également désignées comme des règles voir la figure14 [10].

.

.

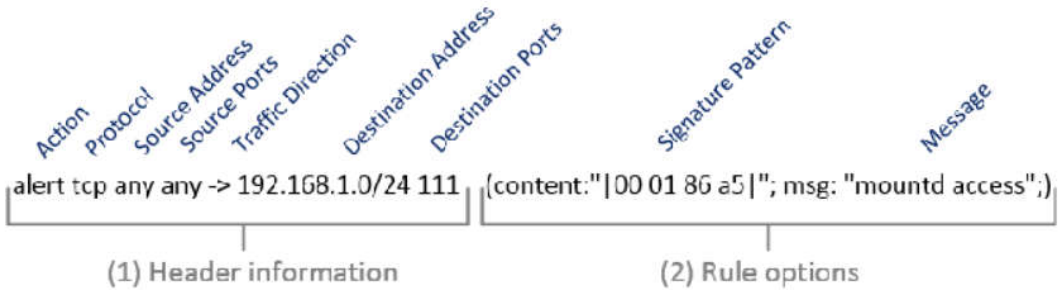


Figure 14 – Syntaxe d'une règle Sort

Ces règles sont divisées en deux parties en-tête et option comme suit :

1. L'EN-TETE DE REGLE

   L'en-tête de la règle contient les informations qui définissent le "qui, où et quoi" d'un paquet, ainsi que ce qu'il faut faire dans le cas ou un paquet avec tous les attributs indiques dans la règle devrait apparaître :

a. l'action de la règle.

L'action de la règle indique à SNORT quoi faire lorsqu'il trouve un paquet qui correspond aux critères de la règle. Il existe cinq types d'actions de la règle voir le tableau 2.1.

Action	Description
alert	génère une alerte a l'aide de la méthode d'alerte sélectionnée, puis enregistre le paquet
log	enregistre le paquet
pass	ignorer le paquet
drop	bloquer et enregistrer le paquet
rejeter	bloquer le paquet, le consigner, puis envoyer une réinitialisation TCP si le protocole est TCP ou un port ICMP inaccessible message si le protocole est UDP.
sdrop	bloque le paquet mais ne connecte pas

TABLE 2.1 – Types d'action de la règle

°

b. le protocole qui est utilise pour la transmission des données (SNORT en considère trois : TCP, UDP et ICMP);

c. les **adresses IP source** et **destination** et leur masque;

d. les **ports source** et **destination** sur lesquels il faudra versifier les paquets.

2. L'OPTION DE LA REGLE

Les options de règle constituent le cœur du moteur de détection d'intrusion de SNORT, alliant la facilite d'utilisation, la puissance et la flexibilité.

Il existe quatre grandes catégories d'options de règle voire le tableau 2.2 .

Catégories	Description	Mots-clés
general	options fournissent des informations sur la règle mais n'ont aucun effet lors de la détection	msg, reference, gid, sid, rev, classtype, within, priority
payload	recherchent toutes des données à l'intérieur de la charge utile du paquet et peuvent être interdépendantes.	content, rawbytes, depth, offset, distance, within, ...
non-payload	recherchent des données non-payload	fragoffset, ttl, tos, ipopts, fragbits, flags, flowbits, seq, ack, window, ...
post-detection	sont des déclencheurs spécifiques à une règle qui se produisent après le déclenchement d'une règle.	logto, session, resp,react, tag, replace, detection filter

TABLE 2.2 – Description d'option de règle

2.4 Conclusion

Apres avoir connu les systèmes de détection d'intrusion et les méthodes de détection des intrusions puis abordées la mise en œuvre d'outil Snort dont le produit est exploités par des milliers d'utilisateurs nous construisons une idée générale pour aboutir le pattern matching dans le chapitre suivant.

Chapitre 3 Pattern matching

3.1 Introduction

L'un des processus importants de l'IDS est l'inspection des paquets d'individus dans le trafic réseau et la décision de savoir si ces paquets sont infectés par des activités malveillantes. Ce processus est effectue via des méthodes de détection de comportement, des signatures ou hybride vues dans le chapitre précédent. La méthode de détection des signatures est basée sur la recherche des patterns qui est exploite les algorithmes de correspondance de chaînes ou la correspondance de contenu. Cette section mette en évidence les algorithmes de correspondance de chaînes qui sont appliques dans l'IDS[17].

3.2 Définition de Pattern Matching

3.2.1 Motifs ou Pattern

Dans la littérature les motifs sont qualifiés en termes anglo-saxons de "pattern" ou de "motif". « Un motif est généralement un segment court, continu et non ambigu d'une séquence alors qu'un pattern a une structure plus complexe. Il est souvent composé de différents motifs qui peuvent être plus ou moins éloignés les uns des autres et sa définition peut comporter des exclusions ou des associations de motifs»[18].

3.2.2 Que signifie Pattern Matching?

Pattern matching consiste a vérifier et a localiser des séquences spécifiques de données d'un motif parmi des données brutes ou une séquence de jetons voire la figure15 [19]. Pattern Matching est l'un des paradigmes les plus fondamentaux et les plus importants dans plusieurs langages de programmation. De nombreuses

applications utilisent la correspondance de motif comme une partie importante de leurs taches [17].

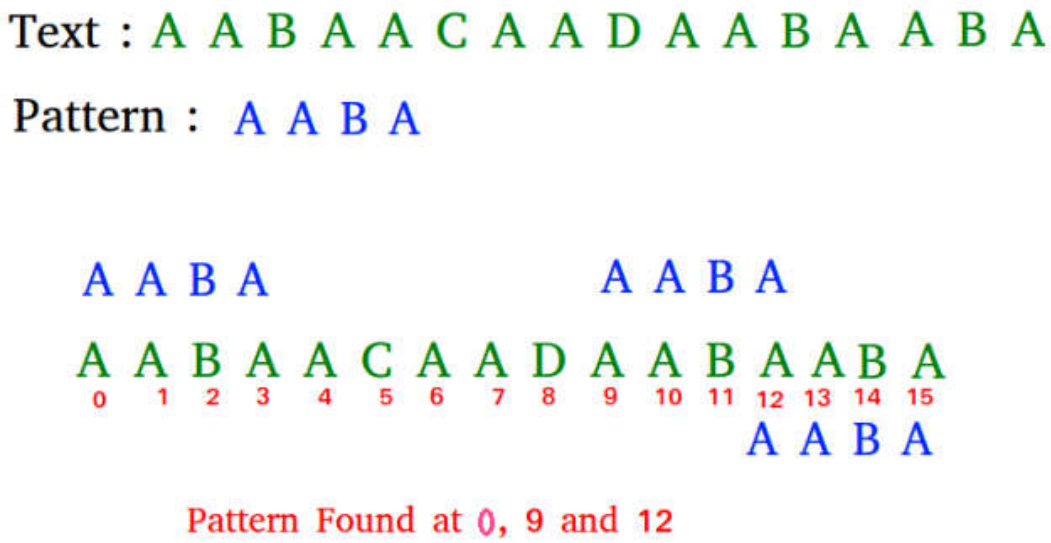


Figure 15 - Pattern Matching

3.3 Algorithmes Pattern Matching

La recherche des motifs du pattern dans le paquet analyse a besoin des algorithmes spécifiques désignés â cette tache. Ces algorithmes sont exploités dans différents logiciels pour la recherche des occurrences des mots dans le traitement des textes, comme ils sont implantes dans les outils des systèmes d'exploitation pour manipuler la recherche dans les différents fichiers. Il existe deux types d'algorithmes pour détecter la correspondance des motifs soit par la recherche d'un mot-clé unique ou par plusieurs mots-clés.

3.3.1 Algorithmes Pattern Matching a mot-clé unique

L'algorithme de correspondance de modèle de mot cl é unique est utilisé pour trouver toutes les occurrences d'un mot-clé dans une entrée donné, parmi ces algorithmes :

1. ALGORITHME BRUTE FORCE (NA¨IF)(BF)

L'algorithme commence par comparer le motif au texte, en balayant de gauche a droite, un caractère à la fois, jusqu' a ce qu'il n'y ait plus de caractères correspondants.

Algorithm 3.1 Brute Force

Input: T (Text to search in them)
Input: P (Pattern)
Output: j (Position pattern found)

```
1:  function BRUTEFORCESEARCH(T,P)
2:      n ← length(T)
3:      m ← length(P)
4:      for i ← 0 to n − m do
5:          j ← 0
6:          while j < m and S[i, j] = P[j] do
7:              j ← j + 1
8:          if j = m then
9:              return j                    ▷ Pattern found at position j
10:     return −1                           ▷ Pattern not found
```

La complexité temporelle de la force brute est O(mn), qui s'écrit parfois O(n*m) , nous devions rechercher une chaine motif (P) de "n" caractères dans une chaîne texte (T) de "m" caractères en utilisant la force brute, cela nous prendrait n * m essai[20].

2. ALGORITHME DE KARP- RABIN (KR)

L'algorithme de Rabin-Karp ou Karp-Rabin est un algorithme de recherche de sous-chaîne par fonction de hachage crée par Richard M. Karp et Michael O. Rabin (1987). Contrairement a l'algorithme naïf qui compare directement le motif a toutes les sous-chaînes du texte caractère par caractère, l'algorithme va comparer des empreintes du motif aux empreintes des sous-chaînes du texte, de manière a faire des comparaisons et des sauts de sous-chaîne plutôt que de simples caractères.

Algorithm 3.2 Karp Rabin

Input: T (Text to search in them)
Input: P (Pattern)
Output: i (Position pattern found)

```
1: function KARPRABINSEARCH(T,P)
2:     n ← length(T)
3:     m ← length(P)
4:     htext ← hash(T[0..m − 1])
5:     hpattern ← hash(P)
6:     for i ← 0 to n do
7:         if htext = hpattern then
8:             if hash(T[i..i + m − 1]) = P then
9:                 return i                          ▷ Pattern found at position i
10:    return −1                                     ▷ Pattern not found
```

Par rapport a la façon naïve de rechercher m motifs en répétant une recherche mono-motif en O(n), ce qui aboutit a une recherche en O(n m), la variante de l'algorithme ci-dessus peut trouver tous les m motifs en O(n+m) attendus.[21].

2. ALGORITHME DE BOYER- MOORE

L'algorithme de Boyer-Moore (BM) est un algorithme de recherche de sous-chaîne particulièrement efficace. Il a été développé par Robert S. Boyer et J Strother Moore1 en 1977. Il compare le motif P a certains facteurs du texte T, en faisant glisser P de gauche a droite le long du texte T.

Algorithm 3.3 Boyer Moore

Input: T (Text to search in them)
Input: P (Pattern)
Output: i (Position pattern found)

```
1:  function BADCHARHEURISTIC(P)
2:      m ← length(P)
3:      for i ← 0 to AlphabetSize do
4:          badChar[i] ← −1                    ▷ Initialize all occurrence as -1
5:      for j ← 0 to m do
6:          a ← P[j]
7:          badChar[a] ← j                     ▷ Initialize all occurrence as -1
8:      return badChar

9:  function BOYERMOORESEARCH(T,P)
10:     n ← length(T)
11:     m ← length(P)
12:     badChar ← badCharHeuristic(P)
13:     j ← m − 1
14:     j ← m − 1
15:     while i > n − 1 do
16:         if T[i] = P[j] then
17:             if j = 0 then
18:                 return i                   ▷ Pattern found at position i
19:             else
20:                 i ← i − 1
21:                 j ← j − 1
22:         else
23:             jumpc ← badChar[T[i]]
24:             i ← i + m − min(j, 1 + jumpc + 1)
25:             j ← m − 1
26:     return −1                              ▷ Pattern not found
```

3. La complexité de l'ordre de O (m/ n) le meilleur cas, seul un caractère sur P doit être vérifié. Ainsi, plus la sous-chaîne est longue, et plus l'algorithme est efficace pour la trouver; de l'ordre de O(m + n) dans le pire cas (en utilisant la variante améliorée de l'algorithme avec la seconde table de vérification des occurrences). BM est l'un des premiers algorithmes utilises auparavant dans Snort IDS dans son moteur de détection. Le problème avec l'algorithme BM est qu'il n'est pas efficace dans le cas ou de petits alphabets sont appliqués. Cela fonctionne rapidement avec peu d'ensemble de règles, mais il ralentit avec une grande taille d'ensemble des règles [22].

La remarque que ces types d'algorithmes seront très limitée dans les systèmes de détection d'intrusion, car il est impossible de construire une seule signature qui détecte tous les modèles malveillants connus [23].

3.3.2 Algorithmes Pattern Matching a plusieurs mots-clés

Le système de détection d'intrusion réseau, prend en charge différentes signatures contenant des définitions d'attaque qui doivent être vérifiées. Le Pattern Matching à mot- clé unique ne pas idéal pour convaincre cette opportunité. De nombreuses solutions ont été présentées, testées et mises en œuvre. Nous examinerons certains des algorithmes fondamentaux qui sont actuellement utilise dans la détection d'intrusion contemporaine.

1. AHO-CORASICK

L'algorithme Aho- Corasick (AC) a été proposé par Alfred Aho et Margaret Corasick en 1975. C'est un algorithme de correspondance des chaînes multiples qui construit une machine a états finis à partir d'un motif (liste de mots-clés), puis utilise la machine pour localiser toutes les occurrences des mots-clés dans un corps de texte [24].

Algorithm 3.4 Aho Corasick

Input: T Text String T=$a_1a_2...a_n$
Input: g goto function for pattern P=$p_1, p_2..p_m$
Input: f faillure function
Input: $output$ output function
Output: Locations keywords(P) occur in T

```
1:  function AHOCORASICKSEARCH(T,P)
2:      state ← 0
3:      n ← length(T)
4:      for i ← 0 to n do
5:          while g(state, a_i) = fail do
6:              state ← f(state)
7:          state ← g(state, a_i)
8:          if output(stat) ≠ empty then
9:              Print i
10:             Print output(state)
```

L'algorithme fonctionne bien en général et n'est pas affecté par la taille du motif (m), ce qui le rend plus stable. L'algorithme AC a moins de temps d'exécution et de complexité de O (n), car il fonctionne linéairement dans la taille de l'entrée.
A côté de cela, il a les mêmes performances dans les cas moyens et dans les pires cas, il peut donc être empêché d'être attaqué[25].
C'est l'un des algorithmes de correspondance de chaînes largement utilises dans Snort IDS, le seul problème avec cet algorithme est lorsqu'un grand nombre de paquets entrants et un grand ensemble de règles sont présents; l'algorithme ralentit les performances de Snort IDS.

2. ALGORITHME DE WU- MANBER

L'algorithme WM fait progresser les algorithmes de correspondance de chaînes multiples en introduisant un nouveau concept de décalage (saut) des caractères correspondants inutiles dans la phase de recherche basée sur un bloc de caractères. Ce concept de saut améliore la vitesse de recherche et est plus rapide que les algorithmes précédents basés sur des automates. L'algorithme peut être divisé en deux phases; 1) Phase de prétraitement, 2) Phase de numérisation [26].

Algorithm 3.5 Wu Manber

Input: T : Text String T=$a_1a_2...a_n$
Input: P : Pattern P=$p_1, p_2..p_m$
Input: $SHIFT$: SHIFT table
Input: $HASH$: HASH table
Input: $PREFIX$: PREFIX table
Input: B : suffix block length
Input: B' : prefix block length
Output: Locations keywords(P) occur in T

```
1:  procedure WuManberSearch(T,P)
2:      m ← min(length(p_1),..,length(p_m))
3:      i ← m − 1
4:      while i ≤ n do
5:          h ← HASH(T[i − B + 1],..,T[i])
6:          shift ← SHIFT[h]
7:          if shift = 0 then                          ▷ Shift block matches
8:              textPrefix ← hash(T[i − m + 1],..,T[i − m + 1 + B'])
9:              pFirst ← HASH[h]
10:             pEnd ← HASH[h + 1]
11:             while pFirst < pEnd do
12:                 if textPrefix = PREFIX[pFirst] then   ▷ Prefix block matches
13:                     px ← P[pFirst]                    ▷ Current Keyword
14:                     pxLen ← length(px)
15:                     j ← 0                             ▷ Count catarctere matched
16:                     while j < pxlen and T[i − pxlen + 1 + j] = px[j] do
17:                         j ← j + 1
18:                     if j ≥ pxlen then
19:                         Output i − pxlen + 1
20:                 pFirst ← PFirst + 1
21:             i ← i + 1                                 ▷ Shift only one place
22:         else
23:             i ← i + shift                             ▷ Shift part of text
```

L'algorithme fonctionne rapidement lorsque la taille des règles Snort est importante (supérieure â 3 octets). Mais lorsqu'une petite taille de règles est appliquée, telle que 1 octet, 2 octets et 3 octets, les performances du l'algorithme ralenti [27]. WM a un temps d'exécution quadratique dans le pire des cas, mais il peut fonctionner très rapidement. Par conséquent, cette variante dans le cas moyen et dans le pire des cas rend l'algorithme susceptible de tomber sur une attaque de complexité algorithmique.

3. AUTRE ALGORITHMES

Le Wu-Manber modifie (MWM) est une version améliorée de l'algorithme WM; il est développé pour résoudre les problèmes hérités de l'algorithme original de WM. Dans la base de données de l'ensemble de règles Snort, il existe un certain nombre de règles de taille quatre octets et moins, MWM utilisé en particulier pour les tailles de motif inferieures à quatre octets. Le problèmes de l'algorithme MWM est l'incapacité de travailler correctement avec la taille de pattern d'un octet. A coté de cela, MWM a un comportement quadratique dans le pire des cas et, par conséquent, il peut tomber dans la complexité algorithmique exploité par des attaques [28]. Un autre algorithme, Piranha est conçu directement pour fonctionner avec IDS, base sur l'idée à rechercher la sous-chaîne de quatre octets la plus rare du motif dans le contenu des paquets. Pour toutes les instances de la sous-chaîne rare, Snort est charge de vérifier la règle correspondante. Piranha lui-même ne peut gérer que des modèles avec une longueur supérieure ou égale à 4octets [29]. L'une des principales faiblesses de Piranha est son incapacité à travailler correctement avec une longueur de motif plus petite plus de quatre octets. A coté de cela, il a le problème de collision (le faux positif), qui peut surviennent pendant la phase de recherche de paquets. Les attaquants tentent de faire plus de collisions dans un paquet et de maximiser le temps nécessaire à l'algorithme pour reconnaitre chaque correspondance de sous-chaîne faussement positive.

3.4 Pattern Matching et trafic réseau

3.4.1 Problématique

La plupart des algorithmes de correspondance de chaînes sont destines de rechercher des occurrences des motifs dans des textes. Notre objectif est de **comment exploiter ces algorithmes pour détecter les patterns dans les paquets capturés afin de détecter les intrus est offrir la sécurité du trafic réseau?**

3.4.2 Contraintes d'implémentation

Les algorithmes de correspondance de chaînes souffrent de nombreuses lacunes et limitations qui les empêchent d'effectuer correctement leur travail dans IDS parmi eux [30] :

1. VARIANTE DE LA LONGUEUR DE PATTERN

Les patterns ont des longueurs différentes (1 octet, 2 octets et 3 octets, ou plus), ces algorithmes existe des correspondances très rapides dans des uns mais se panique dans d'autres.

2. RISQUE D'ATTAQUES PAR LA COMPLEXITE ALGORITHMIQUE

L'attaquant tente de forcer l'algorithme de correspondance de chaîne IDS à atteindre son pire comportement de cas en surchargeant l'entrée dans laquelle l'algorithme n'est pas entièrement conçu pour le prendre en charge. Comme résultat, l'attaque de complexité algorithmique réduit les performances de l'IDS et le paquet qui contient de l'activité de signature réussiront à l'empêcher de détection.

3.4.3 Proposition de la solution

1. PROPOSITION DES FONCTIONNALITES :

a. *Sensibilité aux majuscules et minuscules*

Il est préférable d'avoir une fonction qui convertit tous les cas de sensible aux majuscules et minuscules pour éviter cette faiblesse tout en comparant les patterns aux règles (signatures). L'algorithme doit prendre en charge les deux cas pour augmenter la fiabilité de la recherche sur les activités d'intrusion.

b. *Amélioration de la performance par le traitement parallèle*

Dans le moteur de détection IDS, la correspondance de contenu compare le contenu du paquet avec les signatures des règles. Cependant, ce processus ralentit 70 % des performances d'IDS. Pour réduire la charge de travail dans

la correspondance de contenu consiste en le parallélisant le traitement, où la correspondance de pattern d'entrée est répartie sur plusieurs processeurs.

c. *Face aux attaques par la complexité algorithmique*

L'algorithme de correspondance doit avoir une stabilité de performance entre les pires cas et les moyens cas.

2. ALGORITHME ADEQUAT :

Le meilleur choix pour les algorithmes vue dans cette étude est d'implémenter l'algorithme Aho-Corasick suite a ces avantages de la recherche multi mots-clés et son efficacité de traitements des patterns de taille variante, comme il devienne un standard dans des IDS bases sur les signatures comme SNORT.

3. ORGANIGRAMME INITIAL DE LA SOLUTION

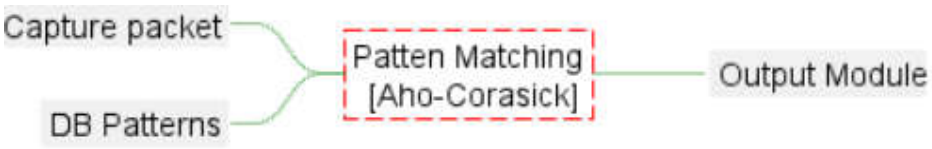


Figure 16 - Organigramme Initial

3.5 Conclusion

Dans ce chapitre nous avons aperçu les différents algorithmes conçus pour la recherche des motifs sur des chaines de caractère et ces applications sur les patterns, comme nous ont bien perçus quels algorithmes adapter pour les IDS, quel sont les recommandations pour choisir.

Apres les trois chapitres l'idée est bien éclairée de quoi faire pour réaliser l'application qui doit voir dans le chapitre 4.

Chapitre 4 Conception et réalisation

4.1 Introduction

Apres avoir survol é la littérature des systèmes de détection des intrusions et les algorithmes de pattern matching, il s'agit maintenant d'implémenter l'algorithme choisi pour obtenir une application de « pattern matching pour la sécurité de trafic réseau». Ce chapitre porte sur la construction d'une application qui permet de capturer les paquets circuler dans le trafic réseau et les détectés, s'il porte des patterns considérés comme suspects. Par la suite, on présente les résultats de l'ensemble des tests effectués.

4.2 Conception

4.2.1 Description

L'application pattern matching « IDSPattern » est réalisée pour surveiller le trafic réseau par capturer les différents paquets propagé sur une interface réseau déterminé (Ethernet, Wlan, Virtual interface...), afin de déterminer s'il existe des patterns identifié comme suspects. La détection est basée sur l'algorithme Aho-Corasick qui a prouvé sa efficacité sur différents logiciels de détection réalisé tel que SNORT. L'application génère des alertes et des notifications après chaque identifications des l'intrus.

4.2.2 Organigramme de l'application

L'application est organisée comme le schéma indiquer voir la figure 17.

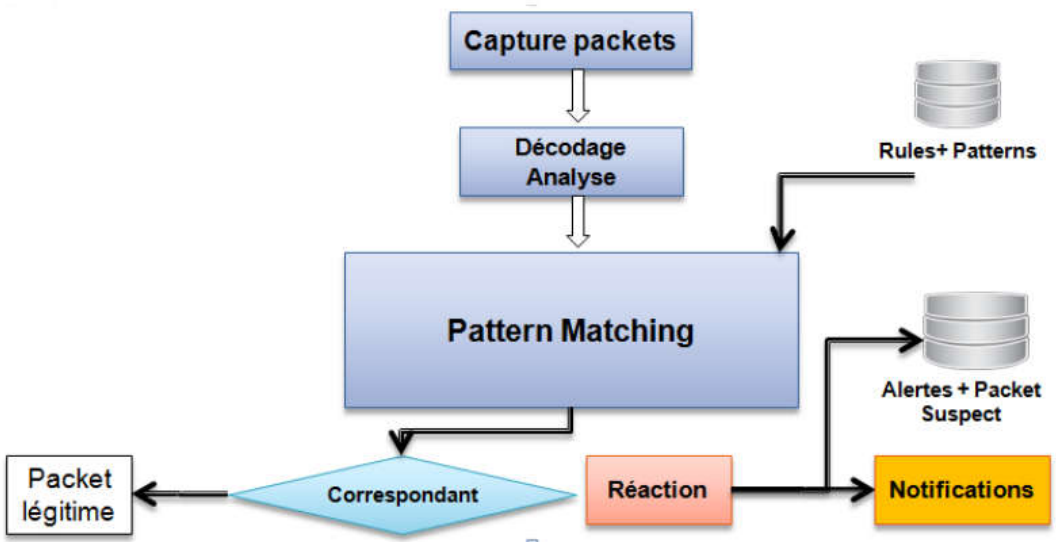


Figure 17 - Organigramme de l'application

4.2.3 Processus Pattern Matching

1. CAPTURE DES PAQUETS

   La capture des paquets (Sniffing) c'est l'étape initial qui permet de capturer les paquets depuis les flux des interfaces réseaux ou d'un fichier « .pcap », pour faire des analyses montrées par la figure 18.

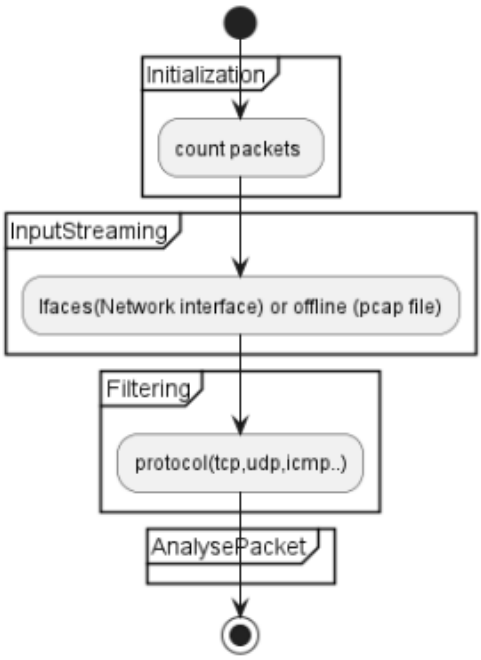


Figure 18 – Capture des paquets

2. ANALYSE DES PAQUETS

Apres capturer un paquet doit le décapsuler afin d'obtenir la partie payload (data) qui forme l'objet de la détection; voir la figure 19.

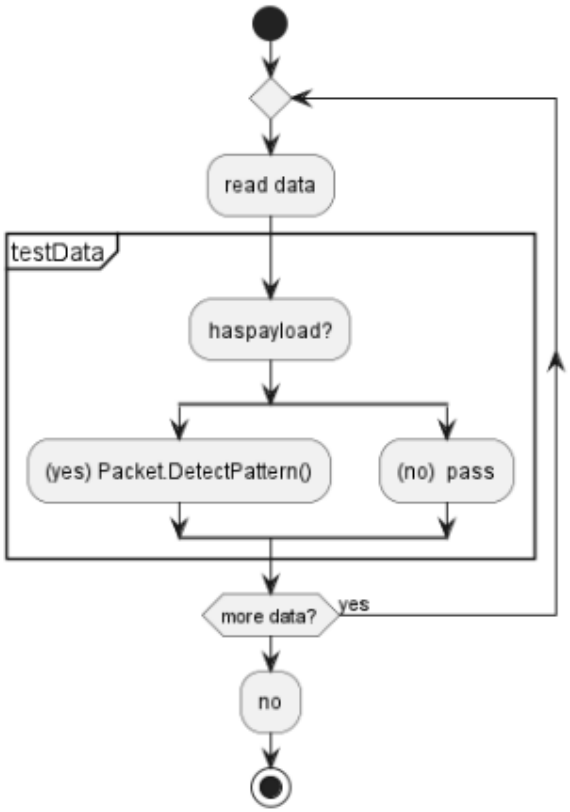


Figure 19 – Analyse des paquets

3. DETECTION D'INTRUSION

La détection constitue le cœur de l'IDS notre approche est d'utiliser l'algorithme Aho-Corasick pour la détection multiple des patterns. La figure illustre
les étapes de détection comme suit :

(a) Récupérer la liste des règles et des patterns (fillListPaterrn()) a partir de la base de données;
(b) Appliquer la méthode de recherche (Aho-Corasick) sur chaque payload entré;
(c) S'il existe des patterns suspects on récupère la règle correspondante, puis générer le module de sortie.

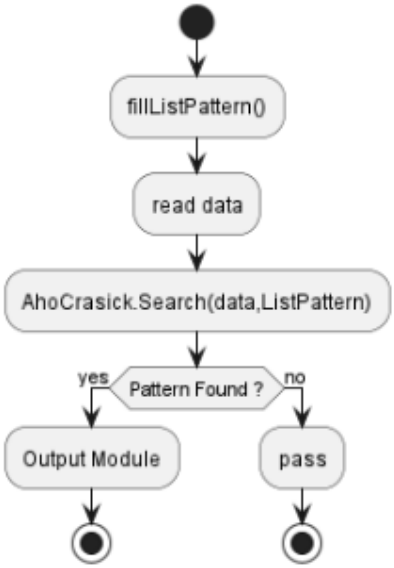


Figure 20 – Détection d'Intrusion

4. MODULES DE SORTIE

Le module de sortie OUTPUT MODULE constitue l'étape finale; après la détection des patterns il faut avertir l'administrateur réseau ou système qu'il y a un intrus comme illustre la figure 21 par :

(a) générer des alertes plus un fichier PCAP pour le paquet suspect ;
(b) sauvegarder les alertes dans la base de données pour l'analyse et le audit ;
(c) envoyer des notifications à l'administrateur réseau.

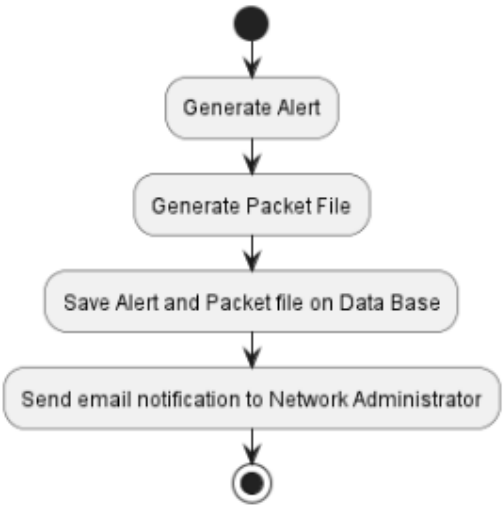


Figure 21 – Modules de sortie

5. MONITORING DES ALERTS

Apres les détections des intrus il faut revoir la liste des alertes de manière plus détaillé sur les informations récoltées concernant le paquet suspect, afin de prendre la décision convenable, comme montré par la figure 22.

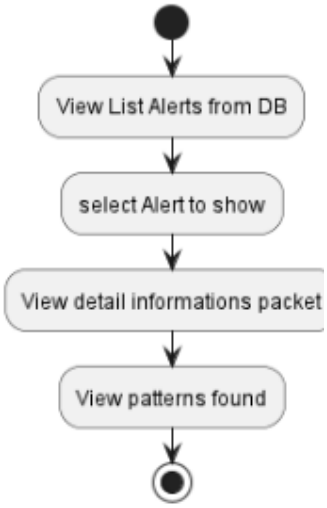


Figure 22 - Monitoring Alertes

4.3 Réalisation

4.3.1 Environnement matériel et logiciel

1. RESSOURCES UTILISEES

(a) Les ressources physiques utilisées sont :

- processeur Intel(R) Core(TM) i3 CPU M 380 @ 2.53GHz 2.53Ghz
- mémoire vive d'une capacité de 8 GB

(b) Et pour ce qui est coté logiciel (Soft) :

- système d'exploitation Windows 7 Professionnel
- machines virtuels Linux Q4OS4.8 basée sur Debian

2. APPLICATIONS DE DEVELOPPEMENT

(a) Python:

Le langage de programmation choisi dans notre travail est le langage Python qui reste l'un des langages les plus utilises dans le domaine de la sécurité informatique. Le langage Python possède un interpréteur très performant, orienter objet, et possède des milliers de bibliothèques et des Framework qui traitent les différents aspects informatiques.

(b) Scapy :

Pour la manipulation des paquets il existe des bibliothèques spécialisées telles que Scapy. Scapy est un puissant programme interactif de manipulation de paquets écrit en Python par Philippe Biondi. Il permet de décoder des paquets d'un grand nombre de protocoles, de les envoyer, de les capturer et de faire correspondre les demandes et les réponses, et bien plus encore. Il peut facilement gérer la plupart des tâches classiques comme le scan, le trace routing, les tests unitaires, les attaques ou la découverte des réseaux [31].

(c) SGBD :

Afin de manipuler les règles et les patterns il est préféré d'utiliser un SGBD local SQLite. SQLite est une bibliothèque écrite en langage C qui implémente un petit moteur de base de données SQL rapide, autonome, hautement fiable et complet [32].

(d) PySide :

L'application sera présentée sur des interfaces graphiques, après la recherche sur les GUIs qui collaborent avec Python, le Framework idéal c'est PySide. PySide combine les avantages de Qt et Python. Un programmeur PySide a toute la puissance de Qt, mais il est capable de l'exploiter avec la simplicité de Python. PySide est sous licence LGPL [33].

(e) VMware :

Pour tester l'application il faut créer un réseau informatique virtuel entre différents machines virtuels. Nous utilisons le logiciel VMware. VMware est un logiciel propriétaire à VMware, Inc., Palo Alto, CA (www.vmware.com), permet a plusieurs copies du même système d'exploitation ou a plusieurs systèmes d'exploitation différents de s'exécuter sur la même machine.

(f) PyCharm:

Un éditeur de texte intelligent pour langage python de la société JetBrains.com «https ://www.jetbrains.com/ »

(g) PlantUml:

PlantUml est une plugins pour PyCharm permet de dessiner tous les diagrammes et les schémas UML «https ://plantuml.com/ »

4.3.2 Implémentation

1. L'ARCHITECTURE UTILISEE

L'architecture idéale pour les applications de détection des intrusions c'est le client serveur, vue la figure 23.

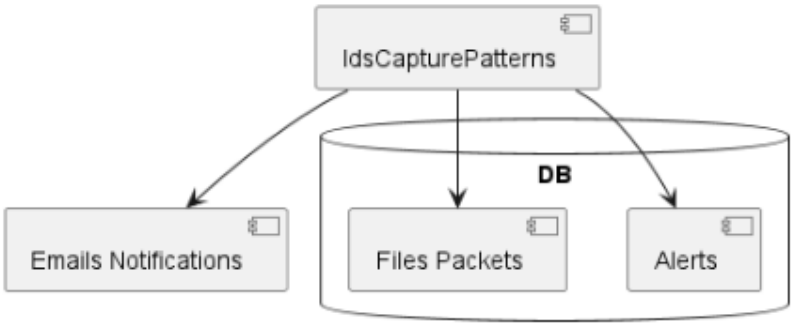


Figure 23 – Architecture utilisée

2. LES INTERFACES D'APPLICATION

Les interfaces de l'application sont représentées comme montre la figure 24 sont :

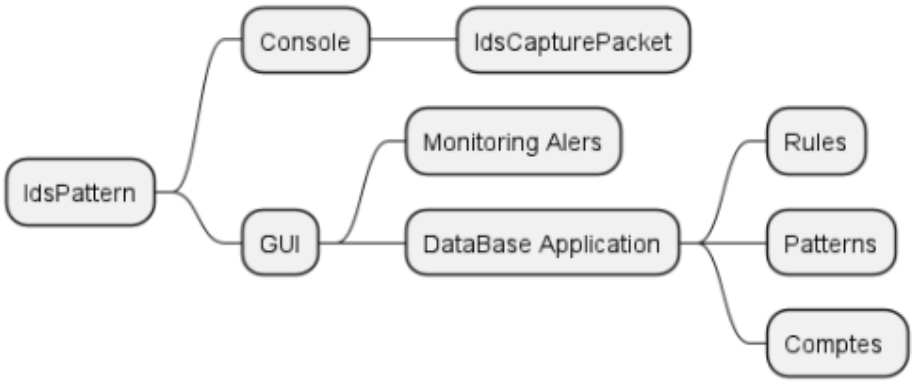


Figure 24 – Interfaces d'application

(a) Interface Capture Paquet

Cette interface représentée dans la figure 25 permet de lancer le sniffing en temps réel soit sur l'interface réseau ou par un fichier pcap déjà capturé.

Figure 25 – Interface Capture paquet

(b) Interface Rule

l'Interface Rule représente les attributs essentiels constituant une règle qui doit être définie et sa manipulation. Les attributs de la règle sont présentés par la figure26.

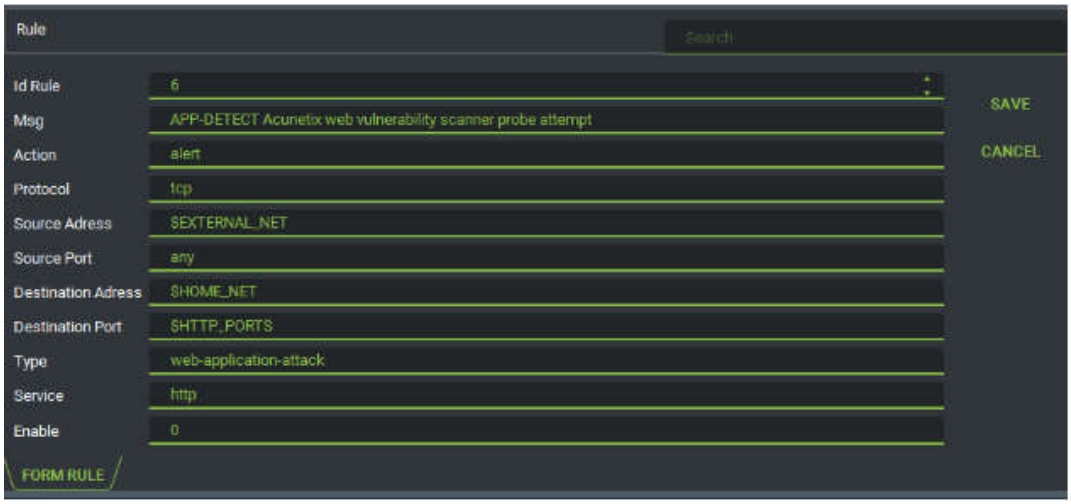


Figure 26 – Interface Rule

(c) Interface Pattern

L'interface Pattern constitue les attributs d'un pattern qui doit être manipulé est représenté dans la figure 27.

Figure 27 – Interface Pattern

(d) Interface Alert

Consiste a contrôler les alertes générées par le temps de capture, le message d'alerte, les informations sur le paquet suspect et les patterns trouves voire la figure 28.

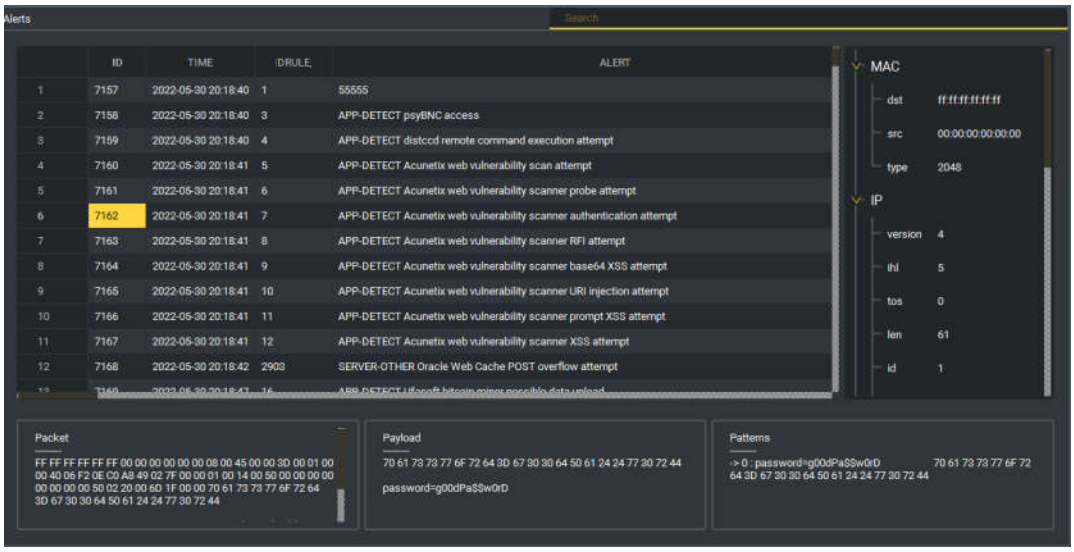


Figure 28 – Interface Alert

(a) Interface Email

Cette interface montre l'envoi des emails par l'application IdsPattern vers Email montre par la figure 29.

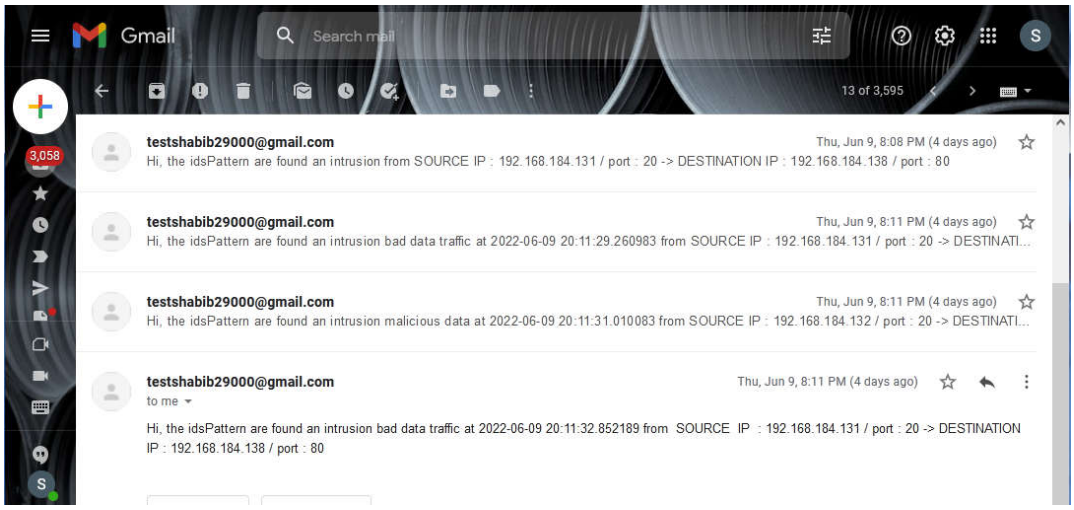


Figure 29 – Interface Email (Gmail.com)

4.3.3 Test de performance

Le test de performance de l'application est base sur l'expérience de tester la performance des algorithmes Aho-Corasick appliques sur l'IDS SNORT cité dans l'article [34]. Notre expérience vient de tester les algorithmes Aho-Corasick implémentés en python qui peur intégrer dans notre application.

1. ALGORITHMES UTILISES

Les algorithmes qui son adaptes pour l'application IdsPattern sont :

(a) Pyahocorasick

Pyahocorasick est implémenté en C et testé sur python 3.6 et plus. L'auteur initial et le responsable est Wojciech Muła ,il est maintenu par un groupe de dizaines développeurs[35].

(b) Ahocrapy

Ahocorasick est entièrement écrite en python. Supporte Unicode. Assez optimise, crée le 5 janvier 2016 par Frederik Petersen[36].

(c) Py aho corasick

Un autre algorithme Ahocorasick entièrement écrite en python, écrit par Jan-Fan, pour ces exigences de travail[37].

2. PLAN D'EXPERIENCE

L'expérience exige d'avoir trois machines virtuelles Linux (deux attaquants (utilisent Scapy) et une machine victime) et une quatrième machine avec un Windows 7 et le déploiement de l'application IdsPattern montrer par la figure 30.

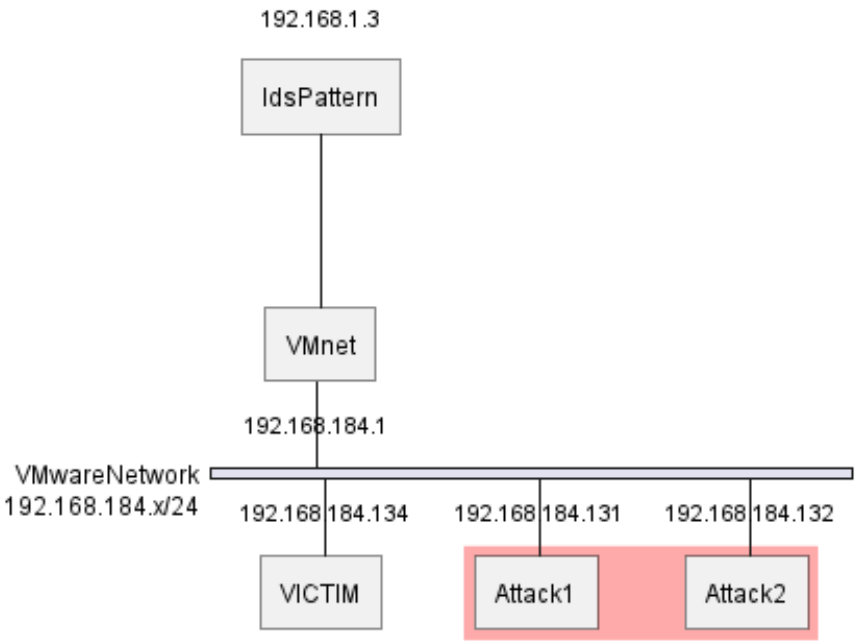


Figure 30 – Network test topologie

3.PARAMETRES D'EVALUATIONS

Pour mesurer la performance des différents algorithmes, soit les paramètres suivants :

(a) **Débit** : C'est le nombre total de paquets analysés dans un laps de temps de capture. Il est mesure en Kpps (kilo paquets par seconde)
(b) Utilisation du processeur (**CPU**) : L'utilisation du processeur fait référence à l'utilisation du processeur par l'IDS
(c) Utilisation de la mémoire **(MC**): L'utilisation de la mémoire fait référence à la quantiste totale de mémoire utilisée par processus IDS.

4. MESURES DE LA PERFORMANCE

(a) Test1

Pour comparer les performances des différents modèles des algorithmes de l'IDSPattern sous trafic lourd, deux nouvelles règles ont été ajoutés aux jeux de règles. La règle vérifie chaque paquet TCP entrant pour une charge utile « payload » contenant les chaînes « malicious » et « bad traffic ». Lorsqu'une correspondance se produit, un message «malicious data detected» et «bad data detected » généré dans le fichier d'alerte, des paquets de taille 1024 octets a 1 Gbps ont été envoyés au VM VICTIM pour 120 secondes.

Les deux attaquants utilisent les commandes suivantes dans Scapy pour générer les paquets envoyés vers la machine VM VICTIM

- send(IP(dst="192.168.184.134")/TCP()/"bad traffic", loop=1)
- send(IP(dst="192.168.184.134")/TCP()/"malicious", loop=1)

i. Résultats des tests

Apres la génération des paquets suspects, on lance le script suivant :

python idscapturet.py a= algorithmes
// a = 1 : payahocorasick ,
2 : ahocorapy,
3 : pay-aho-corasick)

Résultats obtenues voir figure 31:

```
F:\projet>python idscapturet.py a=1     // test pyahocorasick
Rules    :  2
Patterns :  2
Packets analyse : 69333
Packets patterns: 17014
Times durring  : 0:02:00.015865
Total Memory  : 326639616
  Cpu   used  : 81.8
  Memory used : 62.7
```

```
F:\projet>python idscapturet.py a=2     // test ahocorapy
Rules    :  2
Patterns :  2
Packets analyse : 57207
Packets patterns: 14110
Times durring  : 0:02:00.019865
Total Memory  : 286089216
  Cpu   used  : 83.9
  Memory used : 62.9
```

```
F:\projet>python idscapturet.py a=3 // test py_aho_corasick
Rules    :  2
Patterns :  2
Packets analyse : 55220
Packets patterns: 13517
Times durring  : 0:02:00.017864
Total Memory  : 279474176
  Cpu   used  : 82.1
  Memory used : 62.8
```

Figure 31 – Résultats Test1

ii. Analyse des résultats du test1

Afin d'analyser les résultats obtenus il faut faire des comparaisons avec des tests déjà effectués sur des IDS semblable, suite au article [34] cite au-dessus et ces tests comme suit :

Algorithms/ Parameters	Memory	CPU	Throughput
ac-split	25.9	57.825	40.01804
ac-q	42	73.09	42.22628
ac-bnfa	7.1	69.38	41.56937
ac-banded	50.8	74.95	40.46659
ac-sparsebands	40.3	90.4	41.42213

Tableau 1 Résultats des tests de l'article 34

Ces Algorithmes cites dans l'article précédent sont implémenter au SNORT, ont les exploites par calculer la moyenne de ces paramètres et les rendrez comparables avec les algorithmes implémentés dans l'IdsPatterns.

Apres la combinaison des résultats on construit le tableau suivant :

Algorithme	Mémoire (%)	CPU (%)	Débit (kpps)
avg_ac_Snort	33,20	73,13	41,14
pyahocorasick	62,70	81,80	56,42
ahocorapy	62,90	83,90	46,56
py_aho_corasick	62,80	82,10	44,94

Tableau 2 - Résultats de test1

iii. Comparatives des algorithmes

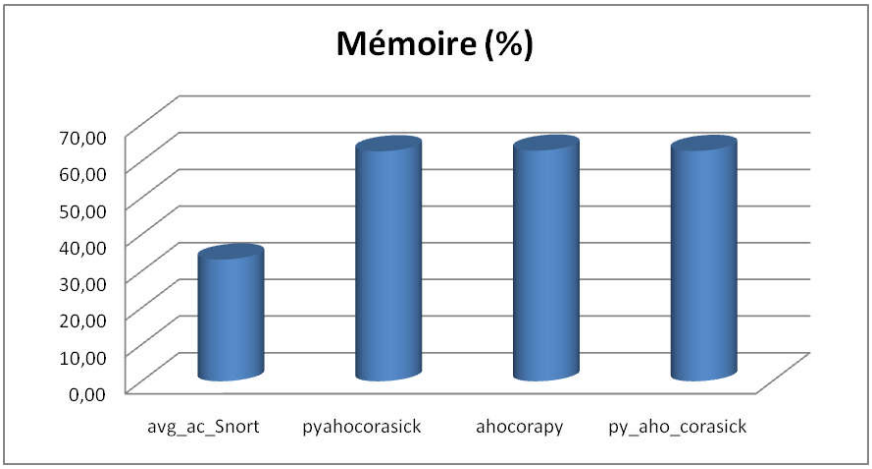


Figure 32 – Comparative des algorithmes en terme de Mémoire / test1

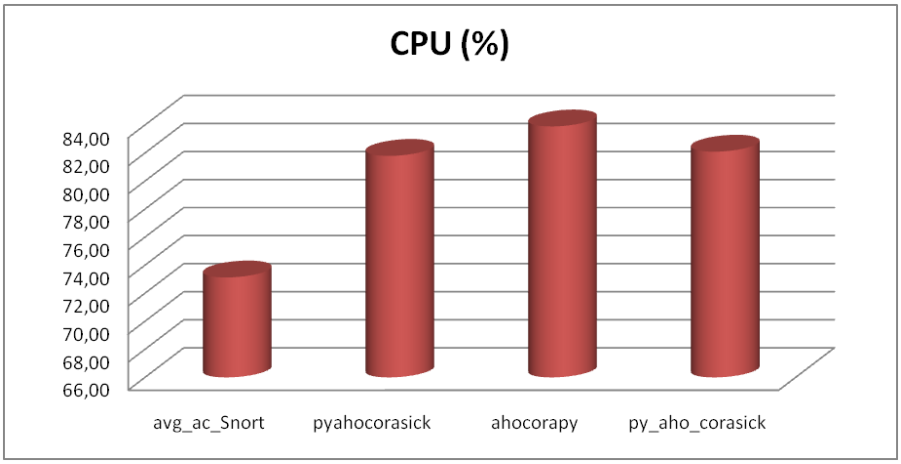


Figure 33 – Comparative des algorithmes en terme de CPU / test1

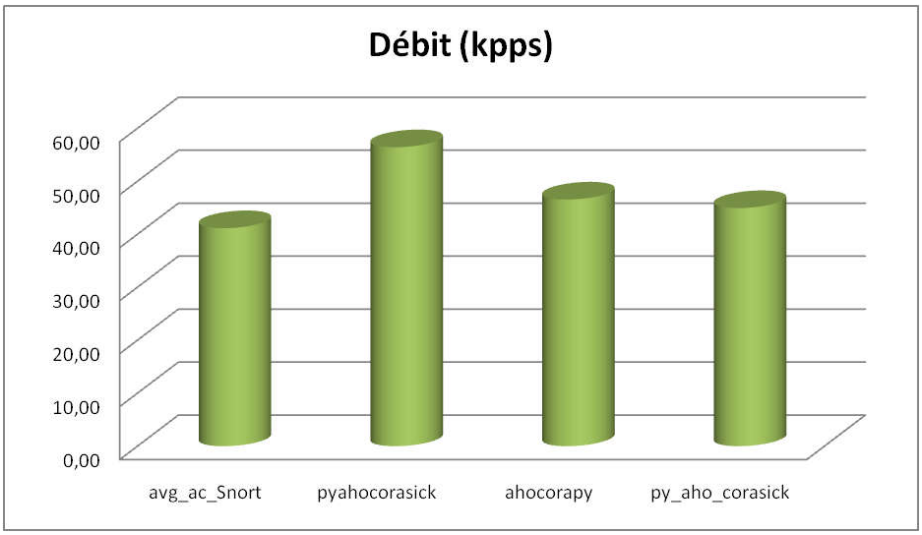


Figure 34 – Comparative des algorithmes en terme de Débit / test1

(b) Test2

Un deuxième test sera appliqué sur IdsPattern par augmentation des nombres de patterns, suite de manques de dataset nous avons convertir le fichier «snort3-community.rules »[38] réalisé par la communauté SNORT et remplir la base de données par les informations des règles et des patterns.

Apres avoir lancé le test2 nous constatons les résultats suivantes voir figure 35:

```
F:\projet>python idscapturet.py a=1 p=-1        // test pyahocorasick
Rules     :  3665
Patterns  :  4390
Packets analyse : 40960
Packets patterns: 9967
Times durring   : 0:02:00.016864
Total Memory    : 235110400
  Cpu    used   : 80.7
  Memory used   : 62.2
```

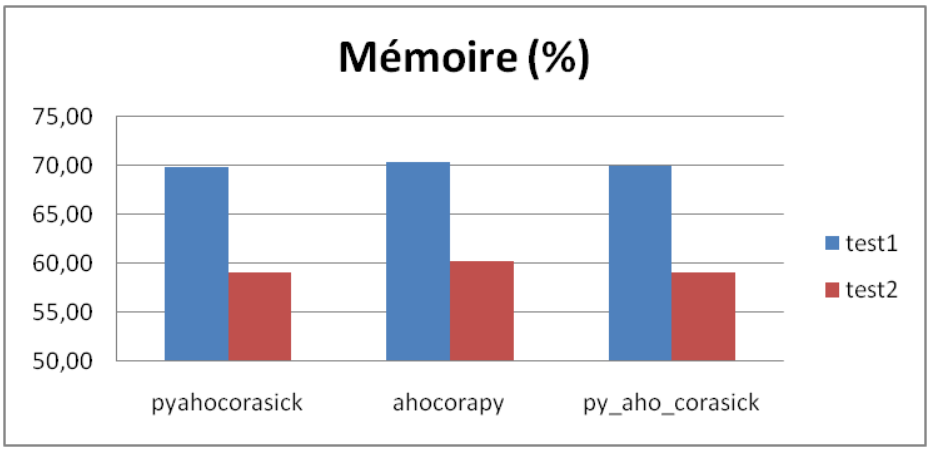
```
F:\projet>python idscapturet.py a=2 p=-1        // test ahocorapy
Rules     :  3665
Patterns  :  4390
Packets analyse : 38655
Packets patterns: 9408
Times durring   : 0:02:00.018864
Total Memory    : 244649984
  Cpu    used   : 82.1
  Memory used   : 62.3
```

```
F:\projet>python idscapturet.py a=3 p=-1        // test py_aho_corasick
Rules     :  3665
Patterns  :  4390
Packets analyse : 29650
Packets patterns: 14780
Times durring   : 0:02:00.046866
Total Memory    : 206905344
  Cpu    used   : 82.8
  Memory used   : 61.9
```

Figure 35 - Résultats Test2

Comparatives des algorithmes des deux tests:

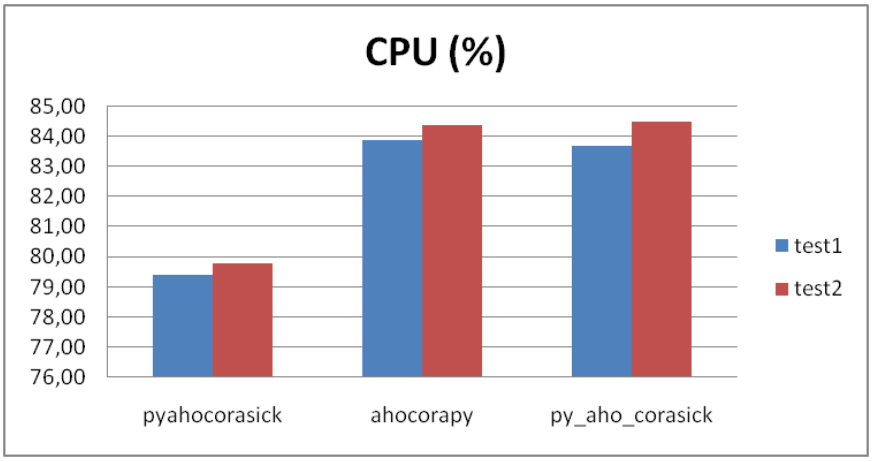
A. Mémoire



Algorithme	test1	test2
pyahocorasick	69,90	59,00
ahocorapy	70,40	60,20
py_aho_corasick	70,00	59,00

Tableau 3 - Tableau comparative (test1-test2) en terme de mémoire

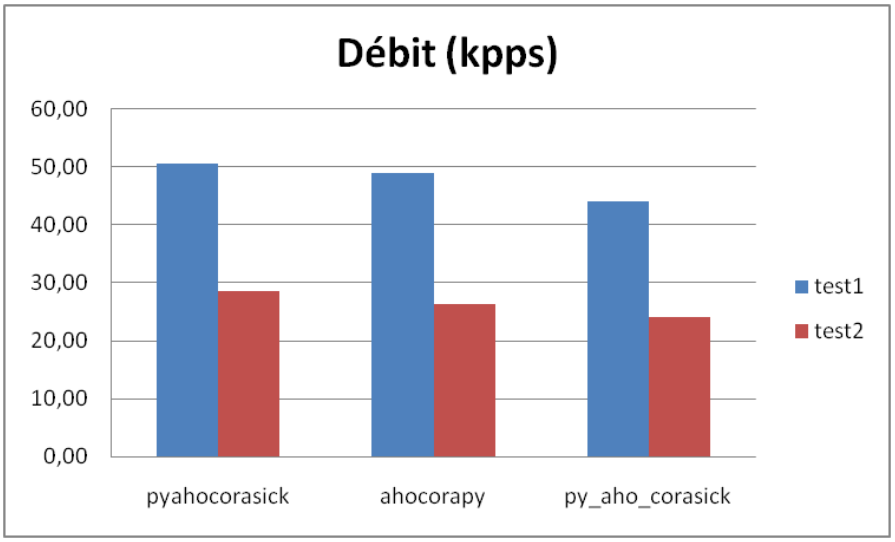
B. CPU :



Algorithme	test1	test2
pyahocorasick	79,40	79,80
ahocorapy	83,90	84,40
py_aho_corasick	83,70	84,50

Tableau 4 – Tableau comparative (test1-test2) en terme de CPU

C. Débit :



Algorithme	test1	test2
pyahocorasick	50,53	28,62
ahocorapy	48,91	26,40
py_aho_corasick	44,05	24,03

Tableau 5 – Tableau comparative (test1-test2) en terme de Débit

4.3.4 Discussion

Les résultats obtenus par les deux tests prouvent l'augmentation de consommation de la mémoire et l'utilisation exhaustive de CPU cela fut à l'expérience faite par d'utilisation des trois machines virtuelles plus la machine local Windows 7, tous exécutent simultanément, cela rendre une charge massive des ressources.

La deuxième remarque c'est le débit où nous constatons que les valeurs obtenues par le test1 voir le **Tableau 2** sont raisonnables, mais l'IDS dégrade lorsqu'il y a un nombre important des patterns à rechercher.

L'application IdsPattern est basée sur la méthode de détection par signature, cela rendre tous les avantages et les inconvénients à un impact sur l'application. L'application est moins de faux positifs mais peut se produire par un paquet contenant des patterns multiplies dans des positions différentes dans la même payload. Pour le faux négatif, l'application ne détecte pas les intrusions non enregistrées dans la base des signatures pour cela, Il est donc nécessaire de les mettre à jour régulièrement.

4.4 Conclusion

Dans ce chapitre nous avons vu tous les étapes nécessaires pour réaliser l'application IdsPattern, ces interfaces principales et les tests de performance, cela rendre l'application a être déployer pour travailler sur un NIDS ou HIDS.

Conclusion générale

Notre projet traite la détection d'intrusion dans un système informatique et l'effet sur la sécurité de trafic réseau. Cette recherche à nous éclairer les concepts de la sécurité et le mécanisme et les outils pour mieux protégé le système informatique, parmi ces outils se trouve les IDS qui jouent un rôle essentiel pour détecter les intrusions par les différentes méthodes.

Les algorithmes de détections des patterns constitue le cœur des IDS, ils sont nombreux chacun d'eux a des avantages et des lacunes. Notre application implémente l'algorithme Aho-Corasick qui est base sur la recherche des motifs multiples et considérés comme un choix adéquat pour créer un nouvel IDS basé sur les signatures. Nous avons intégré les bases de données pour stocker les patterns et consulter les alertes.

Ce projet a nous permet d'expérimenter les compétences de notre formation en master RéSys, presque tous les modules ont ces retouches. Ce projet nous a éclairé les concepts de créations des systèmes de détection des intrusions personnalisées adaptatives à chaque organisme quoi qu'existe des IDS standard. Nous avons essayé de voir toutes les étapes de mettre en œuvre un IDS basé sur les patterns matching afin de renforcer la sécurité du trafic réseau.

La sécurité informatique reste un vaste domaine de recherche, toujours où il y a un potentiel de mettre les données sensibles à l' égard des personnes non désirer, paresse des attaquants pour faire son profit. Malgré tous les infrastructures et les diapositives de protection restent insuffisantes pour garantir le zéro risque.

Future perspective

Notre projet est une étape initiale pour future améliorations, par ajout des modules de vérification et de contrôle du comportement des intrus, et d'exploité d'une manière exhaustive les paramètres récoltés sur les paquets suspects afin de mieux renforcer la détection des intrusions.

Bibliographie

[1] Karen Kent, Suzanne Chevalier, and Tim Grance. Guide to integrating forensic techniques into incident. *Tech. Rep. 800-86*, 2006.

[2] Ludovic Pietre-Cambacedes. *Des relations entre sureté et sécurité*. PhD thesis, Telécom ParisTech, 2010.

[3] Jean-Philippe Bay Jean-Franc¸ois Pillou. *Tout sur la securité informatique* , page 4. Dunod, 2016.

[4] Stephane Lohier and Dominique Present. *Reseaux et Transmissions* , page 281. January 2016.

[5] Jonathan Tournier, Franc¸ois Lesueur, Frédéric Le Mouél, Laurent Guyon, and Hi- ¨ cham Ben-Hassine. Audit d'un systeme iot par test d'intrusion. In *RESSI 2018-Rendes-Vous de la Recherche et de l'Enseignement de la Securité des Systémes d'Information*, pages 1–3, 2018.

[6] What is dns spoofing? https ://www.keycdn.com/support/dns-spoofing. Consulter le 01/02/2022.

[7] How to identify a mirai-style ddos attack. https ://www.imperva.com/blog/how-toidentify-a-mirai-style-ddos-attack/dos. Consulter le01/02/2022.

[8] James P Anderson. Computer security threat monitoring and surveillance. *TechnicalReport, James P. Anderson Company*, 1980.

[9] Dorothy E Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, (2) :222–232, 1987.

[10] Winn Schwartau. The history and evolution of intrusion detection. 2001.

[11] Mary K. Pratt. What is an intrusion detection system? how an ids spots threats. https ://www.csoonline.com/article/3255632/what-is-an-intrusion /detection-system-how-an-ids-spots-threats.html. Consulter le09/02/2022.

[12] Eric Conrad, Seth Misenar, and Joshua Feldman. Chapter 7 - domain 7 : Security operations. In Eric Conrad, Seth Misenar, and Joshua Feldman, editors, *Eleventh Hour CISSP® (Third Edition)*, pages 145–183. Syngress, third edition edition, 2017.

[13] Mohammed Almansor and Kok Beng Gan. Intrusion detection systems : Principlesand perspectives. pages 2458–925, 11 2018.

[14] Roman Fekolkin. Intrusion detection and prevention systems : Overview of snort and suricata. 01 2015.

[15] SNORT R Users Manual. Snort r users manual. In *SNORT R Users Manual 2.9.17.1*. The Snort Project, 2021.

[16] Noor Farjana Firoz, Md. Taslim Arefin, and Md. Raihan Uddin. Performance optimization of layered signature based intrusion detection system using snort. In Touhid Bhuiyan, Md. Mostafijur Rahman, and Md. Asraf Ali, editors, *Cyber Security and Computer Science*, pages 14–27, Cham, 2020. Springer International Publishing.

[17] Pattern matching. https ://www.techopedia.com/definition/8801/pattern-matching. Consulter le28/02/2022.

[18] Recherche de motifs. http ://www.dsi.univ-paris5.fr/bio2/autof2/cha3-5.html. Last Access 28/02/2022.

[19] Boyer moore algorithm for pattern searching. https ://www.geeksforgeeks.org/. Consulter le13/04/2022.

[20] Brute force algorithms explained. https ://www.freecodecamp.org/news/brute-forcealgorithms-explained/. Consulter le02/03/2022.

[21] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L. et Stein, Clifford, *Introduction to Algorithms*, Cambridge, Massachusetts, MIT Press, 2001, 2nd éd. (1re éd. 1990), 1180 p. (ISBN 978-0-262-03293-3), « The Rabin–Karp algorithm », p. 911–916. Consulté le 02/03/2022.

[22] Daniele Beauquier, Jean Berstel, and Philippe Chr étienne.é*léments d'algorith-mique*. Masson, 1992.

[23] Stanimir Bogomilov Belchev. *Pattern matching algorithms for intrusion detection systems*. PhD thesis, California State University, Northridge, 2012.

[24] Vadim Okun. Aho-corasick. https ://xlinux.nist.gov/dads/HTML/ahoCorasick.html. Consulter le 03/03/2022.

[25] Sami Mohamed Halawani and Zailani Mohamed Sidek. Visualization of aho corasick algorithm using z-eves and petri net. *European Journal of Scientific Research*,45(1), 2010.

[26] Mahaveer Prasad Malav and Akhtar Rasool. Variations of wu-manber string matching algorithm. *International Journal of Engineering Research & Technology (IJERT)(3)*, (4), 2014.

[27] G Tyler. Information assurance tools report intrusion detection systems. *Information Assurance Technology Analysis Center (IATAC)*, 2009.

[28] Dan S Wallach and Scott Crosby. Denial of service via algorithmic complexity attacks. 2003.

[29] S Antonatos, M Polychronakis, P Akritidis, KG Anagnostakis, and EP Markatos. Piranha : A fast lookup pattern matching algorithm for intrusion detection. In *IFIP International Information Security Conference*, page 22. Citeseer, 2005.

[30] Awsan Hasan, Nur'Aini Abdul Rashid, Atheer Akram Abdulrazzaq, and Muhannad Abu-Hashem. String matching algorithms for intrusion detection system a survey and taxonomy. *International Journal of Advancements in Computing Technology*, 5 :317–333, 04 2013.

[31] Philippe Biondi and the Scapy community. • 2021 • scapy.net. What is scapy? https ://scapy.net/. Consulter le 10/05/2022.

[32] sqlite.org. What is sqlite? https ://www.sqlite.org/index.html. Consulter le 10/05/2022.

[33] Gopinath Jaganmohan and Venkateshwaran Loganathan. *PySide GUI Application Development*. Packt Publishing Ltd, 2016.

[34] Abhigya Mahajan and Alka Gupta. Performance evaluation of different pattern matching algorithms of snort. 10 :3776–3781, 01 2018.

[35] Wojciech Muła. https ://github.com/WojciechMula/pyahocorasick/issues/145. Last commit Mai 4,2022.

[36] Frederik Petersen. https ://github.com/abusix/ahocorapy. last commit Sep3,2021.

[37] JanFan. https ://github.com/Guangyi-Z/py-aho-corasick. Last commit jul,10 2018.

[38] COMMUNITY SNORT. https ://www.snort.org/downloads/community/snort3-community-rules.tar. Consulter le07/04/2022.