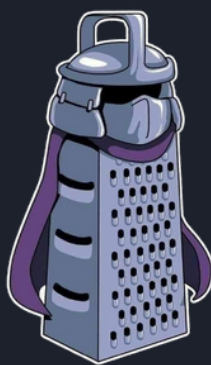


JUIN 2021

MISE EN PLACE D'UN SYSTÈME DE DÉTECTION DU TRAFIC DOUTEUX

NETWORK SHREDDER IDS



Réalisé par : AOUAJ Omar & RAHALI Oussama

Encadré par : M. EL MOSTAFA BELMEKKI



Résumé

Dans le cadre du Blue Teaming, les IDS jouent un rôle très important dans la détection du trafic malicieux. La structure de l'IDS le permet de détecter, voire filtrer les paquets suspects selon des règles personnalisées par l'utilisateur.

Le progrès technologique dans le domaine de sécurité informatique offensive implique la présence de systèmes de détection d'intrusions avec des stratégies avancées de filtrage des paquets en jouant sur les en-têtes protocolaires ainsi que le contenu des datagrammes.

Dans ce rapport, nous avons discuté de l'historique de ces IDS à partir de 1980, leur évolution ainsi que leur fonctionnement selon leur type et leur façon d'implémentation dans le réseau d'une entreprise. En faisant une sorte de Benchmark des IDS répandus dans le monde tels que Snort et Firestorm, nous allons aboutir à construire un propre IDS en s'inspirant de ces fameux systèmes: Le **Network Shredder**.

Nous allons donc illustrer la structure de cet IDS, les modes de son fonctionnement et la méthode avec laquelle il traite les paquets ainsi que les détails des options des règles que l'utilisateur peut personnaliser. Finalement, nous allons faire une sorte de simulation d'attaques et de scan afin de mettre l'IDS en test et en évaluation.



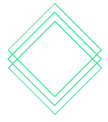
ABSTRACT

As part of Blue Teaming, IDSs play a very important role in the detection of malicious traffic. The structure of the IDS allows it to detect and even filter suspicious packets according to rules personalized by the user.

Technological progress in the field of offensive computer security involves the presence of intrusion detection systems with advanced packet filtering strategies by integrating protocol headers as well as the content of datagrams in detection criterias.

In this report, we discussed the history of these IDS since their apparition in the 1980s, their evolution as well as their functioning according to their type and their way of implementation in the network of a company. By doing a sort of a Benchmark of IDS widely used in the world such as Snort and Firestorm, we ended up building our own IDS, inspired from these famous systems: **The Network Shredder**.

We will therefore illustrate the structure of this IDS, its functioning mods and the method with which it processes the packets as well as the details of the options of the rules that the user can customize. Finally, we are going to do a kind of attacks and scans simulations to put the IDS to the test.



Liste des abréviations

<i>Abréviation</i>	<i>Désignation</i>
IDS	Système de Détection d'Intrusion
NIDS	Système de Détection d'Intrusion basé sur le Réseau
HIDS	Système de Détection d'Intrusion basé sur le Hôte
DIDS	Système de Détection d'intrusion Distribué
IDES	Système Expert de Détection d'Intrusion
NSM	Moniteur de la Sécurité du Réseau
CIDF	Cadre Commun de Détection d'Intrusion
IETF	Internet Engineering Task Force
RFC	Request For Comments



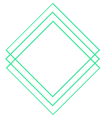
Table des matières

Résumé	2
ABSTRACT	3
Liste des abréviations	4
Table des figures	6
Introduction générale	7
I- Système de Détection d'Intrusion	8
A- Historique	8
B- Fonctionnement du NIDS	11
C- Benchmarking des outils IDS dans le marché	16
1- Snort IDS :.....	16
2- Firestorm IDS	19
II- Recherche et Développement	21
A- Étude profonde du Snort	21
B- Développement: Network-Shredder	24
1- Structure	25
2- Règles	30
C- Preuve du Concept - Simulation et Détection	33
1- Ping Sweep Scan	33
2- DOS Attack: Syn Flood	35
3- SMTP User Enumeration	36
Conclusion	38
Bibliographie	39



Table des figures

Figure 1 : bref historique du développement de la détection d'intrusion	08
Figure 2: Intégration centralisée d'un NIDS dans un réseau	12
Figure 3: Intégration décentralisée d'un NIDS dans un réseau	13
Figure 4: Les Composants d'IDS [Kazienko 2003]	14
Figure 5: Les Composants de Snort IDS [Caswell 2003]	16
Figure 6: Interface de Snort IDS	17
Figure 7: Interface de Firestorm IDS	20
Figure 8: Packets capturées par Snort IDS	21
Figure 9: Format de règle du Snort IDS	22
Figure 10: Règles de détection d'anomalies	23
Figure 11: Network-Shredder IDS - Interface -	24
Figure 12: Network-Shredder IDS - Structure des fichiers -	25
Figure 13: Network-Shredder IDS - règle erronée [Protocol] -	26
Figure 14: Network-Shredder IDS - règle erronée [Adresse IP] -	27
Figure 15: Structure du logique de fonctionnement du Network-Shredder IDS	29
Figure 16: Format de fichier de règles du Network-Shredder IDS	30
Figure 17: Ping sweep scan avec Nmap	33
Figure 18: Network-Shredder IDS - Détection du Ping sweep [Console] -	33
Figure 19: Network-Shredder IDS - Détection du Ping sweep [Web] -	34
Figure 20: Network-Shredder IDS - Détection du Ping sweep [Web - détails] -	34
Figure 21: DOS Attack - Syn Flood avec hping3	35
Figure 22: Network-Shredder IDS - Détection du DOS Attack [Console] -	35
Figure 23: Network-Shredder IDS - Détection du DOS Attack [Web] -	36
Figure 24: Enumeration automatisée des utilisateurs de SMTP avec Nmap	37
Figure 25: Network-Shredder IDS - Détection du SMTP user Enumeration [Console] -	37
Figure 26: Network-Shredder IDS - Détection du SMTP user Enumeration [Web] -	37



Introduction générale

Le **blue Teaming** est une approche de sécurité qui consiste à tester la sécurité des systèmes d'une entreprise, elle est continuellement impliquée dans l'analyse d'activité suspecte. Une Blue Team analyse d'une manière périodique l'efficacité que ça soit des politiques, des termes et des mesures de sécurité adoptées par l'organisation afin d'assurer que les menaces et les risques potentiels soient **analysés, identifiés** et **réduits**. Pour ce faire, elle mettra en place des pratiques d'audits de sécurité, de l'analyse de logs, du reverse engineering ...

Dans cette optique, notre mission est une sorte d'exercice Blue Team, qui consistera à développer et mettre en place un **système de détection du trafic douteux** et ainsi élaborer une preuve de concept d'une simulation d'efficacité de ce système face à quelques attaques (DOS attack SYN Flood, Nmap NULL scan, SMTP User Enumeration ...)

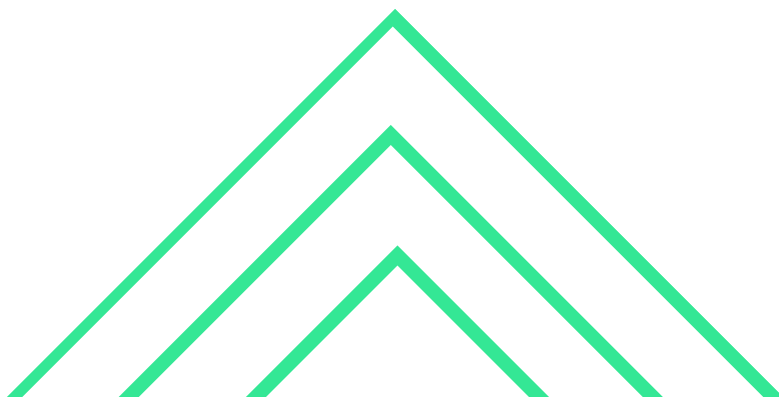
Le présent rapport illustre les phases de déroulement de ce projet. Il comporte deux chapitres organisés comme suit :

- **Chapitre 1 : "Contexte général du projet"**

Présente la problématique ainsi que les objectifs et les missions effectuées dans le cadre de ce projet.

- **Chapitre 2 : "Recherche et Développement"**

Le développement d'un IDS basé sur le réseau (Network-Shredder IDS).



I- Système de Détection d'Intrusion

A- Historique

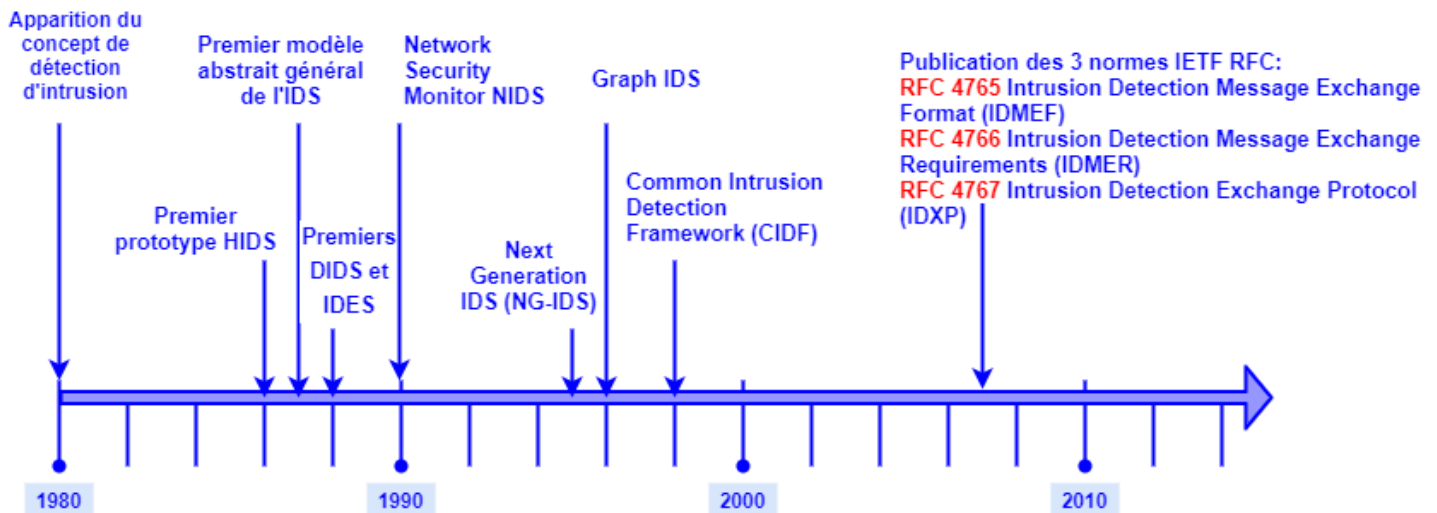


Figure 1: bref historique du développement de la détection d'intrusion

En 1980, James P. Anderson a proposé pour la première fois le concept de détection d'intrusion dans un rapport technique intitulé "Computer Security Threat Monitoring and Surveillance". La tentative ou la menace d'intrusion était définie comme un accès potentiel et prémédité et non autorisé à l'information, et une tentative de rendre le système peu fiable ou inutilisable. En outre, James P. Anderson a proposé une méthode de classification correspondante pour les risques et menaces du système informatique, divisant les menaces en pénétration externe, pénétration interne et comportement illégal, et a également proposé l'idée d'utiliser les données de piste d'audit pour surveiller les activités d'intrusion. Ce rapport est considéré comme le pionnier de la détection d'intrusion.

En 1986, afin de détecter l'accès anormal des utilisateurs à la base de données, W.T.Tener a développé le système Discovery avec COBOL sur le mainframe IBM, qui était le premier prototype de système de détection d'intrusion basé sur l'hôte (HIDS). En 1987, Dorothy E. Denning a proposé un modèle abstrait général de système de détection d'intrusion et a proposé pour la première fois le concept de détection d'intrusion comme mesure des problèmes de défense de sécurité du système informatique. Dans ce modèle, la structure du système se compose de sujets, d'objets, d'enregistrements d'audit, de profils, d'enregistrements d'anomalies et de règles d'activité. Il est indépendant des plates-formes système spécifiques, des environnements d'application, des faiblesses du système et des types d'intrusion, et fournit un cadre général pour les systèmes de détection d'intrusion dans les bâtiments.

I- Système de Détection d'Intrusion

6

A- Historique

Après l'incident du ver Morris en 1988, la sécurité du réseau a vraiment attiré l'attention de l'armée, des universités et des entreprises. L'U.S. Air Force, la National Security Agency et le ministère de l'Énergie ont financé conjointement l'Air Force Cryptographic Support Center, Lawrence Livermore National Laboratory, University of California, Davis, Haystack Laboratory pour développer le système de détection d'intrusion distribué (DIDS). La recherche intègre des méthodes de détection basées sur l'hôte et sur le réseau. DIDS est un produit marquant dans l'histoire des systèmes de détection d'intrusion distribués. Son modèle de détection adopte une structure hiérarchique, comprenant 6 couches de données, d'événements, de sujets, de contexte, de menaces et d'état de sécurité. La même année, Teresa Lunt de SRI/CSL et d'autres ont amélioré le modèle de détection d'intrusion de Denning et ont en fait développé un système expert de détection d'intrusion (IDES). Le système est utilisé pour détecter les tentatives d'intrusion d'un seul hôte, et une idée de détection en temps réel indépendante de la plate-forme du système est proposée, comprenant un détecteur d'anomalies et un système expert, qui sont respectivement utilisés pour l'établissement de modèles statistiques d'anomalies, analyse et détection de fonctionnalités basées sur des règles.

1990 a été un tournant dans l'histoire du développement des systèmes de détection d'intrusion. Cette année, LT Heberlein de l'Université de Californie, Davis et d'autres ont développé un système de détection d'intrusion basé sur le réseau (Network Intrusion Detection System, NIDS) : Network Security Monitor (NSM). Pour la première fois, le système utilise directement les flux réseau comme source de données d'audit, de sorte qu'il peut surveiller des hôtes hétérogènes sans convertir les données d'audit dans un format unifié. Depuis lors, l'histoire du développement des systèmes de détection d'intrusion a tourné une nouvelle page et deux camps se sont formellement formés : l'IDS basé sur le réseau et l'IDS basé sur l'hôte.

I- Système de Détection d'Intrusion

10

A- Historique

Des années 1990 à nos jours, la recherche et le développement de systèmes de détection d'intrusions ont connu un essor fulgurant. En 1995, le Next Generation IDES (NG-IDEs) a été produit, capable de détecter les intrusions sur plusieurs hôtes. En 1996, le système de détection d'intrusion basé sur des graphes (GrIDS) est apparu, principalement utilisé pour la détection collaborative et automatisée à grande échelle.

En 1998, S. Staniford et ses collaborateurs ont proposé un Common Intrusion Detection Framework (CIDF), qui a divisé le système de détection d'intrusion en 4 composants : générateur d'événements, analyseur d'événements, unité de réponse et base de données d'événements. En fait, les données dont le système de détection a besoin d'analyser sont collectivement appelées un événement. Il peut s'agir d'un paquet de données dans le réseau ou d'informations obtenues à partir d'autres canaux tels que les journaux système. Il définit le langage standard pour l'IDS pour exprimer les informations de détection et le protocole de communication entre les composants IDS. Il peut intégrer différents IDS pour les faire fonctionner ensemble et réaliser la réutilisation des composants entre les IDS. Par conséquent, CIDF est également la base de la construction d'IDS distribués. En 2007, le groupe de travail CIDF a de nouveau publié trois projets de normes IETF RFC, à savoir RFC 4765: Intrusion Detection Message Exchange Format (IDMEF), RFC 4766: Intrusion Detection Message Exchange Requirements (IDMER) et RFC 4767: Intrusion Detection Exchange Protocol (IDXP).

I- Système de Détection d'Intrusion

11

B- Fonctionnement du NIDS

“Les systèmes de détection d'intrusion sont un élément important des mesures défensives protégeant les réseaux informatiques contre les abus.”

Dr. John McHugh (2000)



NIDS effectue une surveillance sur les paquets qui entrent dans le réseau et puis juge si quelque chose d'anormal s'introduit dans un système selon des critères, comme sur un système surveillant un grand nombre de demandes de connexion TCP entrant dans divers ports sur un système de destination. Afin de découvrir si quelqu'un essaie un scan de port TCP, le NIDS peut être placé soit sur le système cible, qui surveille son trafic, soit sur une machine distincte du réseau (Routeur par exemple), qui surveille de près l'ensemble du réseau.

I- Système de Détection d'Intrusion

12

B- Fonctionnement du NIDS

En ce qui concerne le NIDS, il peut être intégré dans le réseau d'une entreprise de deux façons:

- **Intégration centralisée:**

Le NIDS est localisé entre le routeur lié à l'internet et le pare-feu qui assure la sécurité des zones DMZ ainsi que le LAN interne.

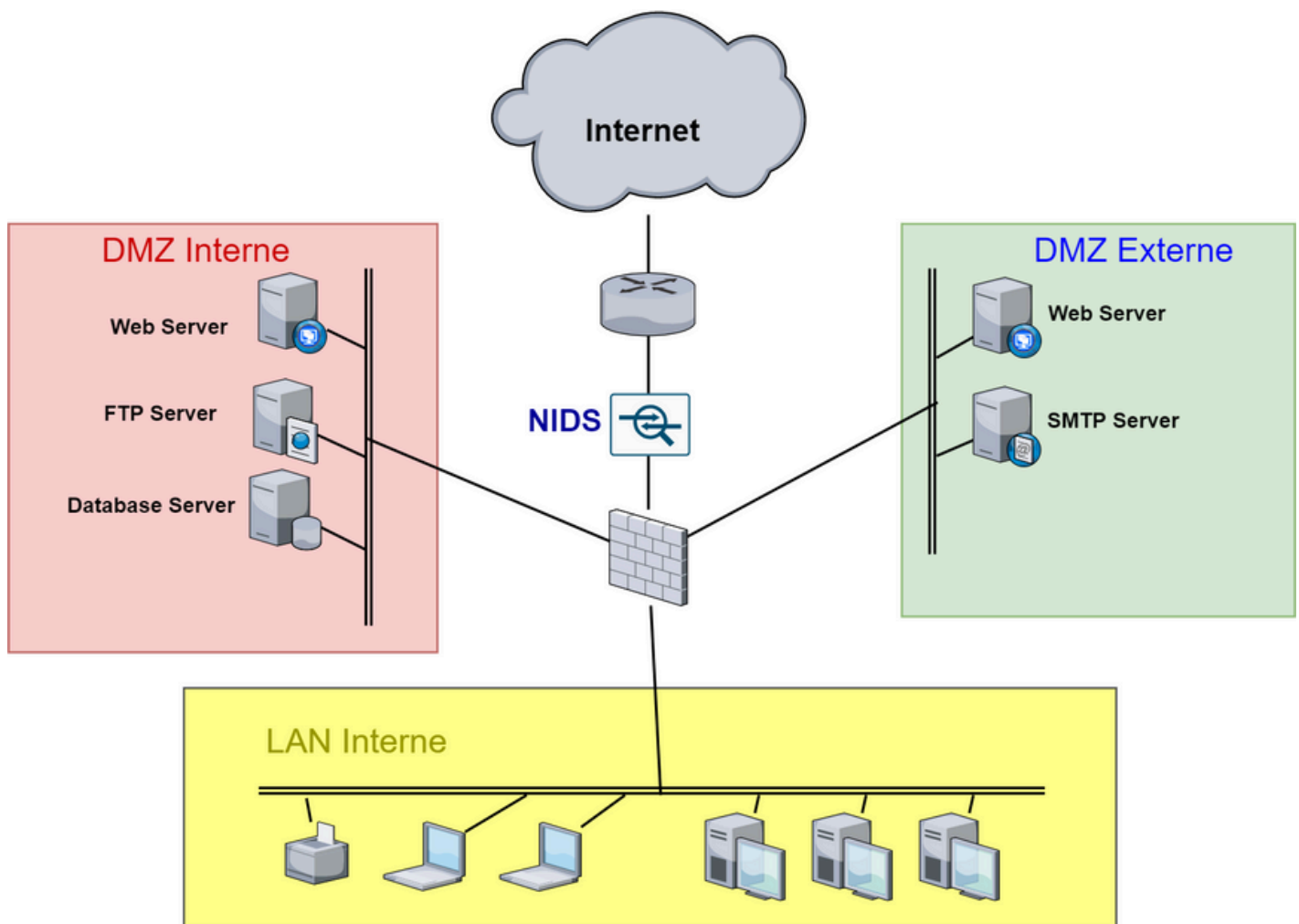


Figure 2: Intégration centralisée d'un NIDS dans un réseau

I- Système de Détection d'Intrusion

13

B- Fonctionnement du NIDS

- **Intégration décentralisée:**

Le NIDS est plutôt intégré dans chaque segment de réseau de l'entreprise, à savoir les zones DMZ externe et interne ainsi que dans le réseau LAN interne.

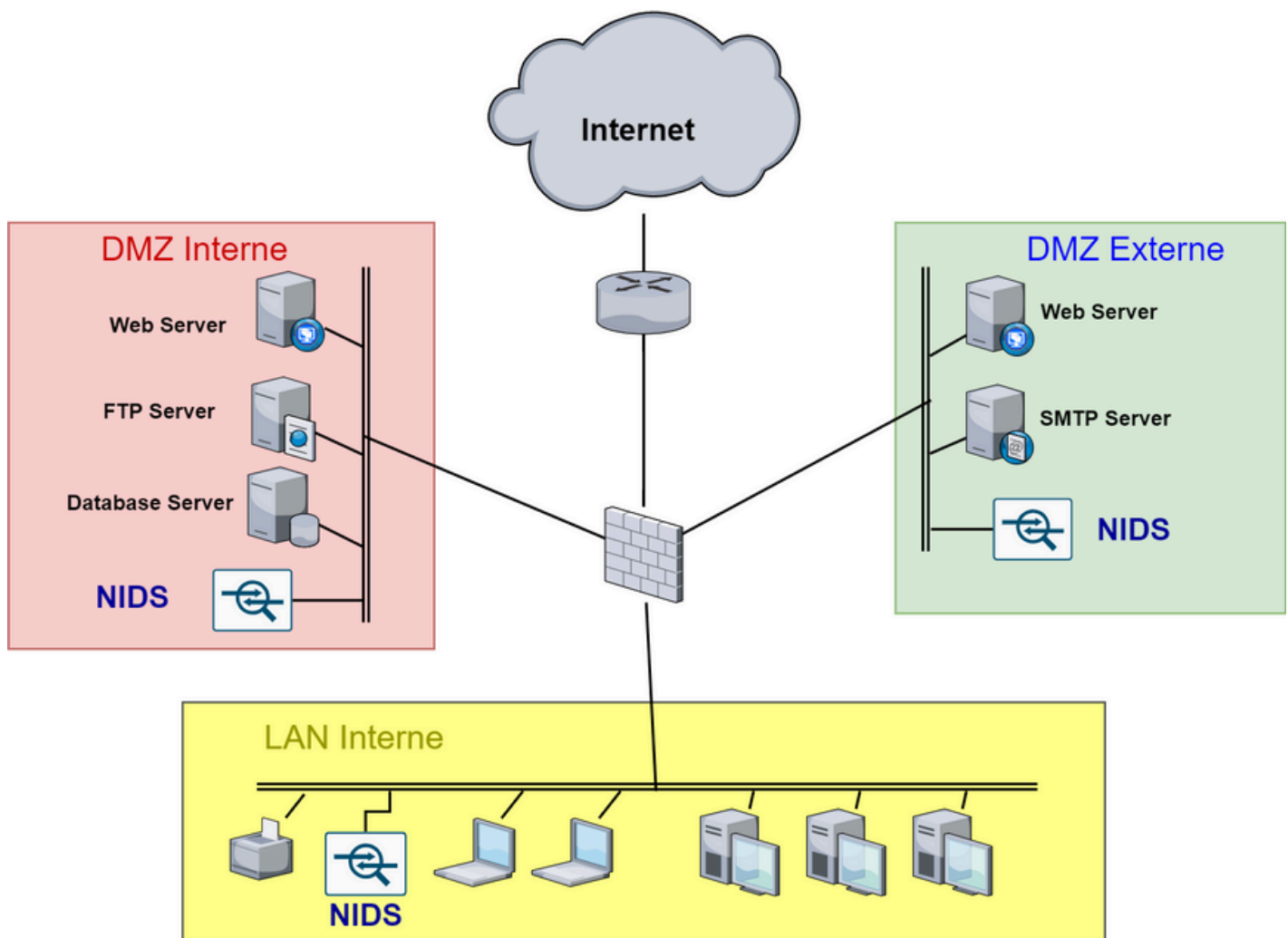


Figure 3: Intégration décentralisée d'un NIDS dans un réseau

Un système de détection d'intrusion est décortiqué en plusieurs composants placés entre le réseau interne et externe. Le réseau interne est effectivement le côté administratif de l'IDS, tandis que le réseau externe ou public est le côté non-administratif de l'IDS.

Les IDS sont en fait constitués de 3 composants principaux:

- **Des capteurs pour détecter les événements ou les activités.**
- **Outils de surveillance des événements et de contrôle des capteurs.**
- **Un dispositif central qui classe les événements enregistrés par les capteurs dans une base de données, ou dans un fichier de log, puis applique un système de règles pour générer des alertes à partir des événements de sécurité reçus.**

I- Système de Détection d'Intrusion

14

B- Fonctionnement du NIDS

Le capteur est le noyau de tout IDS, puisque sa tâche est la détection des intrusions. Il contient des mécanismes de prise de décision selon la nature de l'activité. Les données utilisées par ces mécanismes sont celles générées par exemple par **syslog**. En effet, syslog inclut la configuration des systèmes de fichiers, les autorisations des utilisateurs, etc... Ces données créent ainsi la base du processus de prise de décision.

Le capteur est intégré à un autre composant responsable de la collecte de données, appelé générateur d'événements. Celui-ci crée une politique pour un ensemble d'événements qui peut être un journal ou un audit d'événements système.

Le capteur filtre aussi les données, ignorant toute donnée non pertinente obtenue, pour détecter les activités suspectes. Pour ce faire, l'analyseur utilise la base de données de la politique de détection. Le capteur maintient sa propre base de données qui contient l'historique dynamique des intrusions possibles.

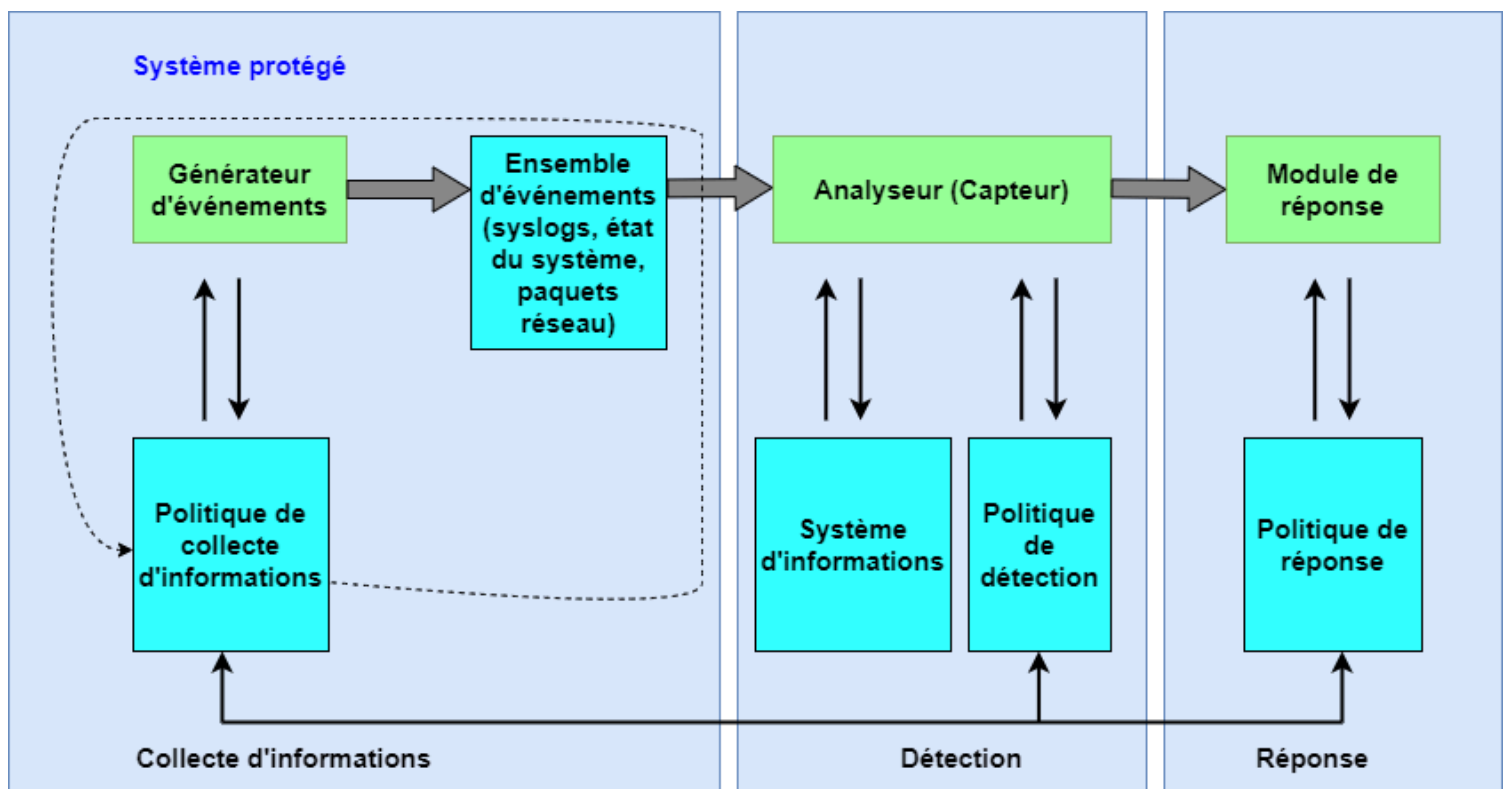


Figure 4: Les Composants d'IDS [Kazienko 2003]

I- Système de Détection d'Intrusion

15

B- Fonctionnement du NIDS

Voici les méthodes principales utilisées par le NIDS pour signaler et bloquer les intrusions:

- **Reconfiguration des appareils tiers (pare-feu ou listes de contrôle d'accès sur les routeurs):**

NIDS envoie une commande à un périphérique tiers, tel qu'un filtre de paquets ou un pare-feu, pour se reconfigurer immédiatement afin de bloquer une intrusion.

- **Envoi d'un e-mail à un ou plusieurs utilisateurs:**

Cela peut être réalisé en envoyant un e-mail à une ou plusieurs boîtes de réception pour signaler une intrusion grave.

- **Envoi d'un trap SNMP à un hyperviseur tiers:**

Ceci est réalisé en envoyant une alerte avec des détails sur les données impliquées sous la forme d'un datagramme SNMP à un hyperviseur tel que HP OpenView ou Tivoli.

- **Log de l'attaque:**

L'IDS enregistre les détails de l'alerte dans une base de données centrale, y compris des informations telles que le timestamp, les adresses IP de l'attaquant et de la cible, le protocole utilisé et le payload.

- **Sauvegarde des paquets suspects:**

NIDS enregistre tous les paquets réseau suspects capturés.

- **Ouverture d'une application:**

NIDS lance un programme externe pour effectuer une action spécifique. Les actions comprennent l'envoi d'un SMS ou le lancement d'un son pour indiquer une alerte.

- **Notification visuelle de l'alerte:**

NIDS affiche une alerte sur une ou plusieurs interfaces de gestion de l'IDS ou du réseau interne en général.

I- Système de Détection d'Intrusion

16

C- Benchmarking des outils IDS dans le marché

1- Snort IDS:

Snort est un système de détection d'intrusion réseau open source, léger et complet, développé par Marty Rosech en 1998. C'est un système léger et qui peut facilement être déployé sur la plupart des nœuds d'un réseau, avec une interruption minimale des opérations. Snort est un langage basé sur des règles, combinant les avantages de la détection des signatures et des anomalies.



De nombreux chercheurs s'accordent à dire que Snort est le meilleur IDS disponible. Avec plus de 5 millions de téléchargements à ce jour, Snort est le système de détection d'intrusion le plus largement déployé au monde et est devenu le standard de facto de l'industrie. De nombreux IDS y utilisent les règles de Snort et agissent comme des fronts avec d'autres fonctionnalités.

Architecture de Snort:

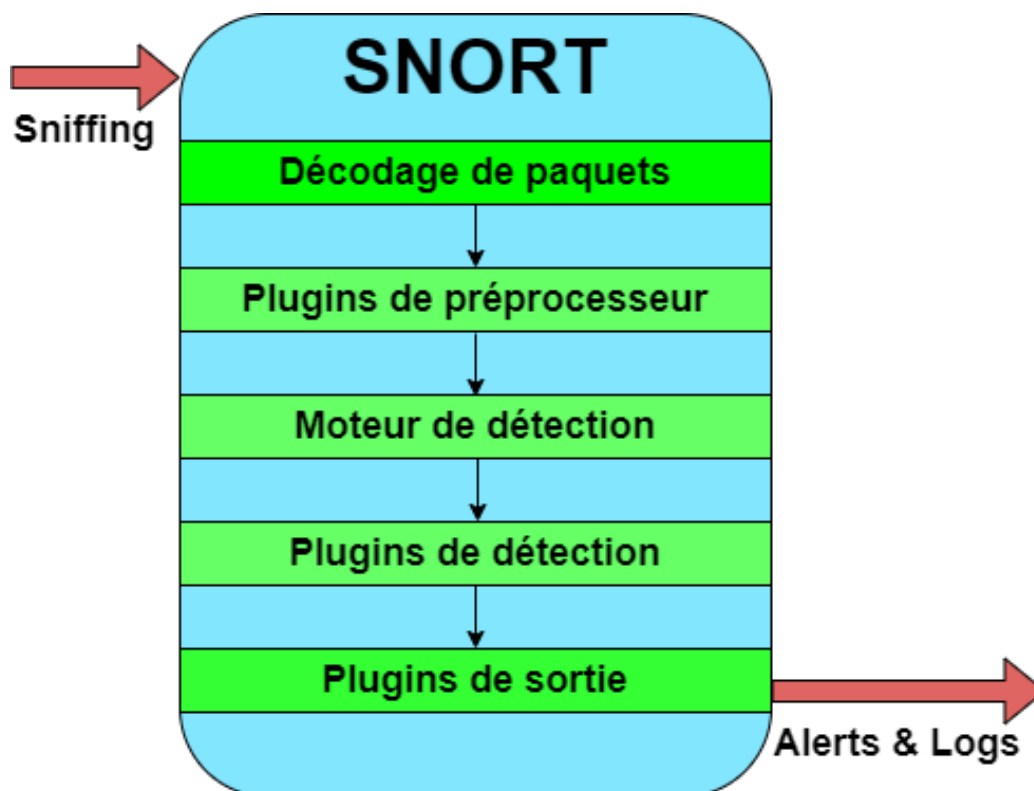


Figure 5: Les Composants de Snort IDS [Caswell 2003]

I- Système de Détection d'Intrusion

17

C- Benchmarking des outils IDS dans le marché

1- Snort IDS:

Interface de Snort IDS:

```
C:\Snort\bin>snort
Running in packet dump mode

--== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{FF41371F-2996-493E-AAF1-F8D22DE17A10}".
Decoding Ethernet

--== Initialization Complete ==--

_*> Snort! <*-
Version 2.9.18-WIN64 GRE (Build 169)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2021 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.11

Commencing packet processing (pid=83764)
```

Figure 6: Interface de Snort IDS

I- Système de Détection d'Intrusion

18

C- Benchmarking des outils IDS dans le marché

1- Snort IDS:

Avantages et Inconvénients:

Avantages

- Snort est un outil IDS gratuit, open source, portable et rapide.
- Snort est un outil léger (déploiement facile sur un système) et fonctionne sur tous les principaux systèmes d'exploitation.
- Snort fournit une détection et un reporting extrêmement flexibles. Son affichage de sortie décodé est plus compréhensible que d'autres outils, comme tcpdump.
- L'utilisateur peut personnaliser les règles pour une meilleure sécurité.

Inconvénients

- Snort laisse tomber les paquets sous charge.
- Snort n'a pas une bonne interface de gestion et de configuration compréhensible par l'utilisateur.

I- Système de Détection d'Intrusion

19

C- Benchmarking des outils IDS dans le marché

2- Firestorm IDS:

Firestorm est un outil sous licence **GPL** (General Public License) basé sur Unix avec des performances NIDS exceptionnelles. Il s'agit d'un capteur offrant un réel support d'analyse, de reporting et de console distante. Il est plus flexible car il est entièrement pluggable.

Firestorm NIDS comprend quatre éléments architecturaux, à savoir:

- **Capteur (Firestorm-NIDS):**

Sniffer le trafic réseau, l'analyser, puis écrire les alertes dans un journal étendu dans un format elog spécifique. Firestorm utilise les signatures Snort pour analyser le trafic réseau.

- **Journaux étendus (fichiers elog):**

Extended Logs est une nouvelle présentation des informations d'alerte. Ce fichier journal contient des informations sur les paquets, les alertes, le décodage et le suivi d'état et d'autres métadonnées. Elogs est un format avantageux car il conserve toutes les données dans un seul fichier.

- **Stormwall:**

Son objectif est de surveiller les spools d'alerte ainsi que d'effectuer des actions lorsque de nouveaux fichiers elog apparaissent. Le capteur est chargé d'informer Stormwall en cas de modification du spool.

- **Console:**

Elle permet à l'utilisateur de rechercher, trier, filtrer, corrélérer et extraire les données des capteurs.

I- Système de Détection d'Intrusion

20

C- Benchmarking des outils IDS dans le marché

2- Firestorm IDS:

Fonctionnalités

- Capable de détecter les anomalies de protocole et d'effectuer des décodages complets de la couche d'application.
- Avec un seul fichier de configuration, il est facile à configurer.
- Fournit une prise en charge complète des règles Snort.
- Inspection du stateful TCP.

Interface de Firestorm:

```
→ src git:(master) X firestorm ~/Downloads/Network-Shredder/source/challenge1.pcap
info: Firestorm NIDS v0.6.0
info: Copyright (c) 2002-2010 Gianni Tedesco
info: This program is free software; released under the GNU GPL v3 (see: COPYING)
info: memchunk: 16384K requested (4096 chunks), 16544K total
info: memchunk: 160K metadata 40 chunks: 0.96% of total
info: objcache: new: _global/_objcache (96 byte)
info: objcache: new: _global/_mempool (56 byte)
info: decode: decode.dot: dumped protocol graph
info: decode: 7 decoders, 18 protocols, max dcb = 56 bytes
info: packet = 80 + 448 bytes
info: tcpdump: /root/Downloads/Network-Shredder/source/challenge1.pcap: standard: snaplen=262144
info: ipdefrag: minttl=1 timeout=60s
info: objcache: new: ipdefrag/ipq (64 byte)
info: objcache: new: ipdefrag/ipfrag (40 byte)
info: objcache: new: tcpflow/tcp_session (120 byte)
info: objcache: new: tcpflow/tcp_state (40 byte)
info: objcache: new: tcpflow/tcp_sbuf (208 byte)
info: objcache: new: tcpflow/tcp_rbuf (32 byte)
info: objcache: new: tcpflow/tcp_data (256 byte)
info: objcache: new: tcpflow/tcp_gap (8 byte)
info: pipeline: starting: tcpdump[/root/Downloads/Network-Shredder/source/challenge1.pcap]
debug: [*S*****] 10.0.2.15:41902 10.0.2.5:80 s:a013c863 w:29200 l:0
debug: bad checksum

debug: [*S**A***] 10.0.2.5:80 10.0.2.15:41902 s:5ecb40b8 a:a013c864 w:28960 l:0
debug: not a valid syn packet
debug: [****A***] 10.0.2.15:41902 10.0.2.5:80 s:a013c864 a:5ecb40b9 w:229 l:0
debug: bad checksum

debug: [***PA***] 10.0.2.15:41902 10.0.2.5:80 s:a013c864 a:5ecb40b9 w:229 l:351
debug: bad checksum
00000 : GET / HTTP/1.1.. 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a
00010 : Host: 10.0.2.5.. 48 6f 73 74 3a 20 31 30 2e 30 2e 32 2e 35 0d 0a
00020 : User-Agent: Mozi 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69
00030 : lla/5.0 (X11; Li 6c 6c 61 2f 35 2e 30 20 28 58 31 31 3b 20 4c 69
00040 : nux x86_64; rv:6 6e 75 78 20 78 38 36 5f 36 34 3b 20 72 76 3a 36
00050 : 0.0) Gecko/20100 30 2e 30 29 20 47 65 63 6b 6f 2f 32 30 31 30 30
00060 : 101 Firefox/60.0 31 30 31 20 46 69 72 65 66 6f 78 2f 36 30 2e 30
00070 : ..Accept: text/h 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 74 2f 68
00080 : tml,application/ 74 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f
00090 : xhtml+xml,applic 78 68 74 6d 6c 2b 78 6d 6c 2c 61 70 70 6c 69 63
```

Figure 7: Interface de Firestorm IDS

Il existe de nombreux autres systèmes de détection d'intrusion basés sur le réseau dans le marché tels que : **Suricata, Bro/Zeek, Strata Guard, Sax2** ...

27

La figure suivante est une capture d'écran des données capturées par Snort:

Figure 8: Packets capturées par Snort IDS

II- Recherche et Développement

A- Étude profonde du Snort

22

Règles de Sniffing

La plupart des **règles SNORT** sont écrites sur une seule ligne ou séparées par "/" à la fin des lignes entre plusieurs lignes. Les règles Snort sont divisées en deux parties logiques : les en-têtes de règle et les options de règle. L'en-tête de règle contient l'action de la règle, le protocole, l'adresse IP et le masque de réseau source et destination, ainsi que les informations sur le port source et de destination ; la partie option de règle contient le contenu du message d'alerte et la partie spécifique du paquet à vérifier.

snort.conf définit l'emplacement du fichier de règles par défaut : **var RULE_PATH /etc/snort/rules**

La majorité des règles snort sont des règles de détection d'abus, c'est-à-dire pour identifier des intrusions connues, un très petit nombre de règles de snort sont des règles de détection d'anomalies, c'est-à-dire une détection basée sur des anomalies statistiques.

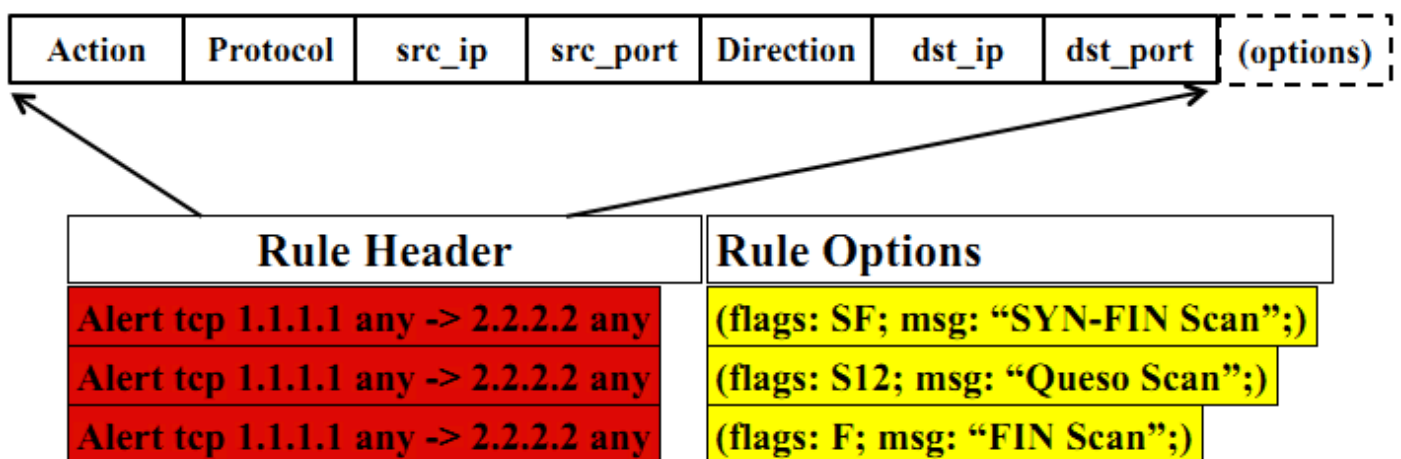


Figure 9: Format de règle du Snort IDS

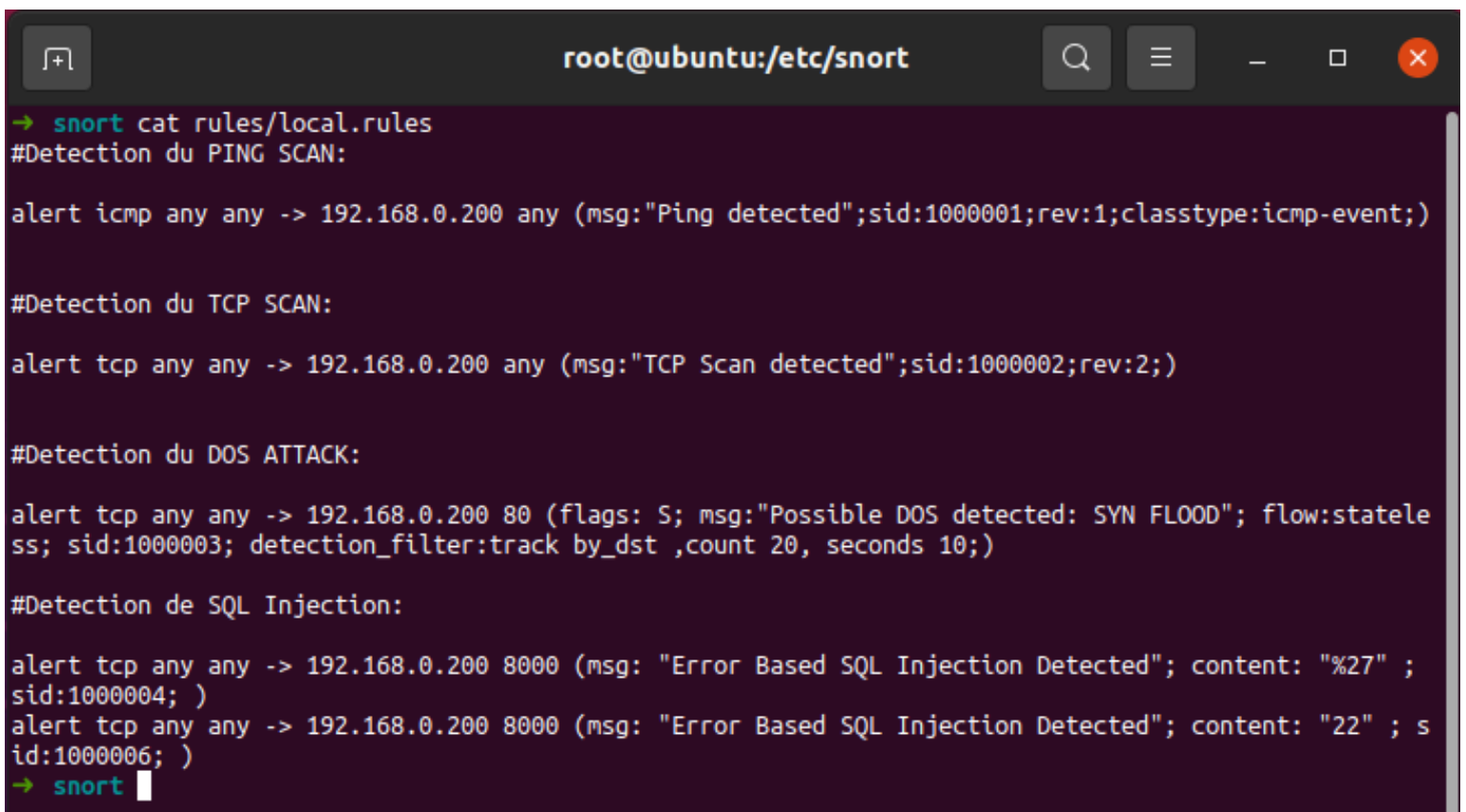
II- Recherche et Développement

A- Étude profonde du Snort

23

L'ordre d'exécution des règles n'est pas en fonction de l'ordre d'apparition des règles, mais en fonction de la priorité du type de règle, et la priorité est décroissante comme suit :

Pass > Drop > Alert > Log



```
root@ubuntu:/etc/snort
→ snort cat rules/local.rules
#Detection du PING SCAN:
alert icmp any any -> 192.168.0.200 any (msg:"Ping detected";sid:1000001;rev:1;classtype:icmp-event;)

#Detection du TCP SCAN:
alert tcp any any -> 192.168.0.200 any (msg:"TCP Scan detected";sid:1000002;rev:2;)

#Detection du DOS ATTACK:
alert tcp any any -> 192.168.0.200 80 (flags: S; msg:"Possible DOS detected: SYN FLOOD"; flow:stateless; sid:1000003; detection_filter:track by_dst ,count 20, seconds 10;)

#Detection de SQL Injection:
alert tcp any any -> 192.168.0.200 8000 (msg: "Error Based SQL Injection Detected"; content: "%27" ; sid:1000004; )
alert tcp any any -> 192.168.0.200 8000 (msg: "Error Based SQL Injection Detected"; content: "22" ; sid:1000006; )
→ snort
```

Figure 10: Règles de détection d'anomalies

II- Recherche et Développement

B- Développement: Network-Shredder

Network-Shredder est un IDS basé sur le réseau, il utilise un fichier de règles afin de décider de la nature malicieuse des paquets détectés selon les options déclarés dans ses lignes. L'idée générale de traitement des règles est issue du concept de Snort tandis que d'autres nouvelles fonctionnalités sont mises en place durant notre développement.

```

(volcker@kali) - [~/Desktop/Network-Shredder/source]
$ python3 Network-Shredder.py -h
NETWORK SHREDDER

      -+++++====-:::
      +++++,      ,=-,-
      *++==      -=-:-
      ,*++=,:::----=-:=
      ,*+++++++*=-:++:::
      -=:,:::--=++*=-=-:-=-=-
      =+*=-:-:,,,:--=+*=-=++++=-:
      =*+++++++====-:::--=-=
      =+*****+**#+
      ==+++++++-@@#@@#@@#@@#@@*+=,
      -=*****+-@#@#@#@#+===-=+*,
      =+++++*#+#@#++++++--+-:*=
      :+++++++=:#++++++--+-:=-#*-:
      =+++++++=-+*+++++--+-:=-#*=
      *#%#%#*+++=-#++++++==--++###,
      :*##*#%#@%#%#%#*#+*#%#*+=,
      +**++++++###%*#*#####*--:
      *#+++++++--=+*****+=-----
      :*#+++++++-----+*+=#-#*=@=-
      +**#%#*+++++++--#*-+*-=---=-=-
      -**#*#@%#%*+=-----=-+*+=-
      :***#=-=-=+=+-----#+@=#-#=-+-=-
      ,*****#++=-----++-----=-+==
      +***#++++++++--=*-=-#+@+#+*+=
      +***#%#*+++++++--+*-=+-=-=-=-=-
      -***#%#*+++++++-----=-*#+-#%+-#-
      ,***#%#*#@%#*+++++---%#+-+=-----##+:
      *****%#+-=*##%##+-----=-=-+=#-##-#%#*=
      =***%#%##++=-----++-=-+#+@+*+=-##%#%#*,
      ,***%#%##+++++++=====++*+=-----++*+=-#%#%#*
      -***%#%##++++++++-----=-*=-%#+#+-+-%#%#%*:
      =***%#%#*#++++++++---*+=-----=-#+%#%#%*:
      +***#%#%#+%#*+++++++-----=-*-%#-#*+=+-=#%#*+
      :*****#=:++#@%#%#++++-=-#+%-=-=-=+-=-,  +*+=,
      :+*****+=-----+*****+=-----++=-%#+#+-+-,  :
      ,:+=*#*+=-----+++-++-#+-+*+=-----*=-:
      +%#*+++++++--+++-=-=-*=-##-#=-
      =+++++++-----=++%-+=-----
      -+++++++--##-+*+=-----=-:::,
      :-+++++++--=-=-=-=-:::,
      :-+++++-----:::,
      :-=-:::

|_ Version : 1.0#beta
|_ Authors : AOUAJ & RAHALI
|_ Usage : python3 Network-Shredder.py rules.txt

usage: Network-Shredder.py [-h] [--pcap PCAP] [--logdir LOGDIR] [--interface INTERFACE] [--web] [--quiet] file
positional arguments:
  file                  Rules file

optional arguments:
  -h, --help            show this help message and exit
  --pcap PCAP           PCAP file (Exclusive for PCAP Mode)
  --logdir LOGDIR       Log Directory (FULL PATH) e.g: /path/to/log/
  --interface INTERFACE Sniff Interface (e.g: tun0)
  --web                Show Logs In Web Interface
  --quiet              Quiet Mode

```

Figure 11: Network-Shredder IDS - Interface -

II- Recherche et Développement

B- Développement: Network-Shredder

Network-Shredder se dispose de deux modes généraux:

- Mode de capture du trafic réseau en temps réel (Live IDS).
- Mode basé sur le traitement d'un fichier PCAP donné en argument.

Cet IDS comporte d'autres options, à savoir:

- -- **logdir**: L'utilisateur peut spécifier le répertoire où le log est enregistré.
- -- **interface**: L'utilisateur peut choisir quel interface l'IDS doit sniffer.
- -- **web**: L'utilisateur, en spécifiant cette option, aura l'accès à une interface web graphique afin de bien visualiser les paquets capturés.
- -- **quiet**: En spécifiant cette option, l'utilisateur ne sera pas dérangé par un terminal rempli de logs et de messages d'alertes.

Dans ce qui suit, nous allons aborder en détail la structure et le fonctionnement du Network-Shredder IDS.

1- Structure:

```
→ Network-Shredder git:(main) ✗ tree
.
├── README.md
├── requirements.txt
├── source
│   ├── banner.txt
│   ├── functions.py
│   ├── Network-Shredder.py
│   ├── ReadRules.py
│   ├── rules.txt
│   ├── Sniffer.py
│   ├── static
│   │   └── logo.png
│   ├── templates
│   │   ├── logs.html
│   │   └── packet.html
│   └── web.py
```

Figure 12: Network-Shredder IDS - Structure des fichiers -

II- Recherche et Développement

B- Développement: Network-Shredder

La mise en place du Network-Shredder IDS est très simple. En effet, il suffit de cloner le repository git :

```
root@projet:~# git clone https://github.com/volck3r/Network-Shredder.git
```

Et installer les pré-requis déclarés dans le fichier **requirements.txt**, en exécutant la commande suivante:

```
root@projet:~# pip3 install -r requirements.txt
```

Une fois que les prérequis sont installés, il suffit de lancer la commande:

```
root@projet:~# python3 Network-Shredder.py -h
```

afin de visualiser la syntaxe des options.

a/ Fichier : Network-Shredder.py

Le rôle du fichier **Network-Shredder.py** est de prendre en entrée les arguments donnés par l'utilisateur, à savoir le fichier des règles, le répertoire des logs, le fichier PCAP ou l'interface à sniffer, ainsi que la présence ou l'absence du mode silencieux.

Ce fichier vérifie la validité des options, puis génère le fichier du log en fonction du temps de début de la capture. Cela fait, il commence à lire les règles à l'aide de la fonction **readrules()** qui existe dans le fichier **ReadRules.py** (Nous allons l'aborder dans ce qui suit).

Network-Shredder.py, après avoir lu toutes les règles existantes dans le fichier de l'utilisateur, va lancer le thread **Sniffer** qui est déclaré dans le fichier **Sniffer.py** (Voir ce qui suit). Enfin, si l'utilisateur spécifier l'option **web**, il va lancer la partie interface web accessible via **http://127.0.0.1:5000/logs**.

b/ Fichier : ReadRules.py

Dans ce fichier, on définit la fonction **readrules(file)** qui prend en argument le fichier des règles, vérifie la validité de chaque ligne (donc de chaque règle). Au cas où une erreur est survenue, le programme affiche l'option de la règle erronée et demande à l'utilisateur de la changer (Voir figures ci-dessous).

```
[+] Starting Network-Shredding...
[~] Reading Rules File rules.txt...
ValueError: [!] Invalid Rule : Incorrect protocol : 'wrong_protocol'.
```

Figure 13: Network-Shredder IDS - règle erronée [Protocol] -

II- Recherche et Développement

B- Développement: Network-Shredder

27

```
[+] Starting Network-Shredding...
[~] Reading Rules File rules.txt...
ValueError: '192.168.0.256/32' does not appear to be an IPv4 or IPv6 network

During handling of the above exception, another exception occurred:

ValueError: [!] Invalid Rule : Incorrect source IP : '192.168.0.256'
```

Figure 14: Network-Shredder IDS - règle erronée [Adresse IP]-

Après vérification de toutes les options d'une règle, la fonction **readrules()** convertit la ligne de cette règle en un dictionnaire qui effectue un mapping du nom de l'option à sa valeur, et à chaque itération, ce dictionnaire est ajouté à une liste. Donc à la fin, la fonction retourne une liste de dictionnaires dont chacun représente une règle.

c/ Fichier: Sniffer.py

Ce fichier définit la classe **Sniffer** qui hérite de la classe **threading.Thread**, en fait, un objet de cette classe lance la fonction **sniff()** importée de **scapy**. Cette fonction a comme argument une autre fonction qui va être appelée à chaque fois qu'un paquet est détecté. Dans notre cas, c'est la fonction **incomingPacket()**.

La fonction **incomingPacket()** prend comme argument le paquet entrant, parcourt l'ensemble des règles et teste si le paquet vérifie chacune d'elles à l'aide de la fonction **match()** (Abordée dans la suite).

Dans ce contexte, nous avons intégré un algorithme qui répond au besoin de l'utilisateur suivant:

"Recevoir une alerte uniquement si le nombre *count* de paquets est détecté dans une période de temps *time*."

Pour cela, si le paquet vérifie une règle, la fonction **incomingPacket()** va tester la condition:

"Nombre de paquets vérifiant la règle égal au nombre *count* spécifié par l'utilisateur et la durée entre le paquet actuel et le premier est inférieure à celle spécifiée par l'utilisateur à l'aide de l'option *time*."

Si cette condition est vérifiée, la fonction va enregistrer le paquet avec le message spécifié dans le fichier de log en appelant la fonction **log()**, et si l'utilisateur n'a pas spécifié l'option **quiet**, va aussi afficher l'alerte dans la console à l'aide de la fonction **console()**.

II- Recherche et Développement

B- Développement: Network-Shredder

d/ Fichier : **functions.py**

Ce fichier contient plusieurs fonctions, comprenant celles appelées par la fichier **Sniffer.py**. Dans ce qui suit, nous allons expliquer le fonctionnement de chacune de ces fonctions:

- La fonction **match(rule, packet)**:

Cette fonction vérifie si le paquet donné en argument vérifie la règle spécifiée, elle fait appel à 4 fonctions comme suit:

- **Fonction checkProtocol(rule, packet)**: Elle vérifie si le protocole du paquet correspond à celui spécifié dans la règle, si oui, la fonction retourne **True**.

- **Fonction checkIPs(rule, packet)**: Elle vérifie si les IPs source et destination du paquet correspondent à celles spécifiées dans la règle, si oui, la fonction retourne **True**.

- **Fonction checkPorts(rule, packet)**: Elle vérifie si les ports source et destination du paquet correspondent à ceux spécifiées dans la règle, si oui, la fonction retourne **True**.

- **Fonction checkOptions(rule, packet)**: Elle vérifie si les autres champs du paquet correspondent à ceux spécifiées dans la règle, si oui, la fonction retourne **True**.

- La fonction **log(rule, packet)**:

Cette fonction enregistre la règle ainsi que le paquet dans le fichier de log en utilisant la méthode **packet.show(dump=True)**, la syntaxe est ainsi la même pour tous les paquets détectés.

- La fonction **console(rule, packet)**:

Cette fonction affiche juste la règle vérifiée et le message d'alerte puisque l'affichage du paquet va remplir le terminal.

II- Recherche et Développement

B- Développement: Network-Shredder

e/ Fichier : web.py

Ce fichier représente la partie web de l'application en utilisant **flask**, en fait, il définit deux routes, à savoir:

- La route **"/logs"**:

En naviguant sur cette route, la fonction **show_tables()** est appelée, celle-ci ouvre le fichier des logs, collecte les informations des paquets telles que le timestamp, le message spécifié par l'utilisateur, les IP source et destination, les ports source et destination, ainsi que l'utilisateur de la session actuelle. Cette fonction va retourner une liste de listes (liste de règles) et va l'envoyer au fichier **logs.html** qui va afficher ces informations en tant que tableau. Une option **Show Details** est disponible, cette dernière va appeler la route **"/logs/id_du_paquet"**.

- La route **"/logs/id_du_paquet"**:

Cette route fait appel à la fonction **show_details()** selon l'id du paquet, celui-ci est issu du timestamp puisqu'il est unique. Cette fonction affiche tous les détails du paquet en les envoyant au fichier **packet.html** qui va les traduire en un tableau qui change de couleur selon les couches du paquet.

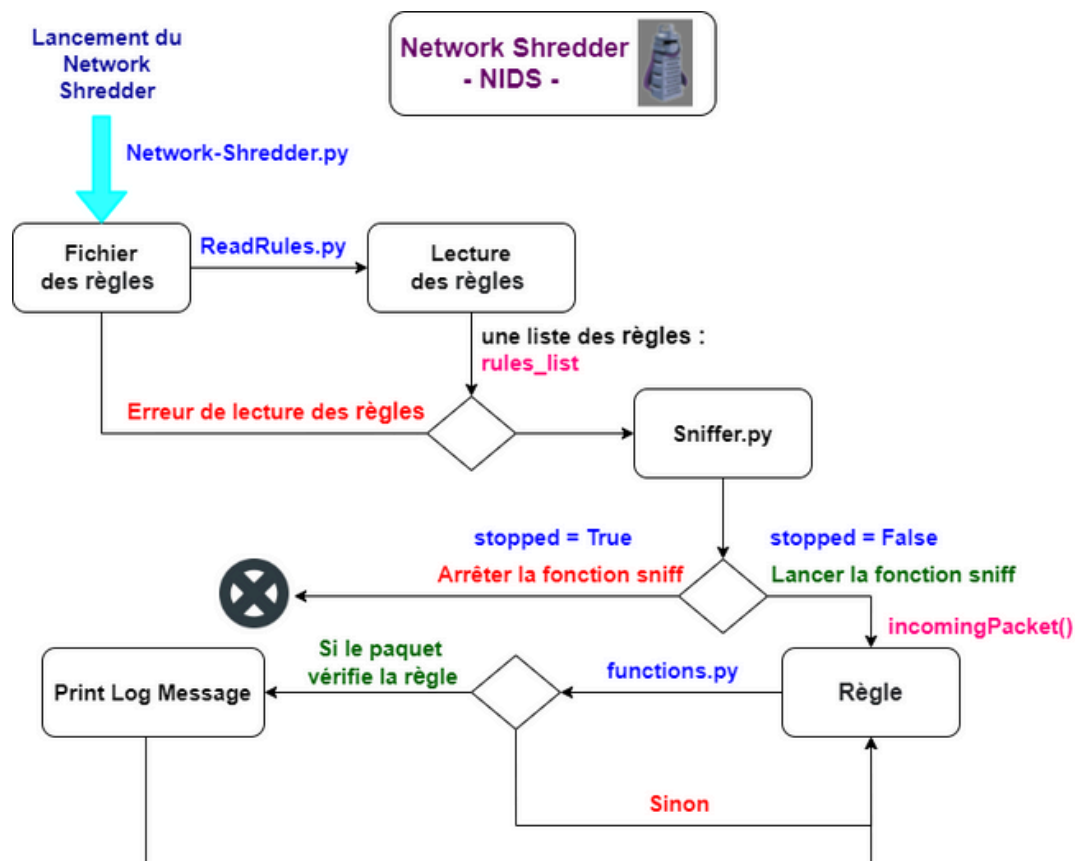


Figure 15: Structure du logique de fonctionnement du Network-Shredder IDS

II- Recherche et Développement

30

B- Développement: Network-Shredder

2- Règles:

Network Shredder - NIDS -



Rules File format

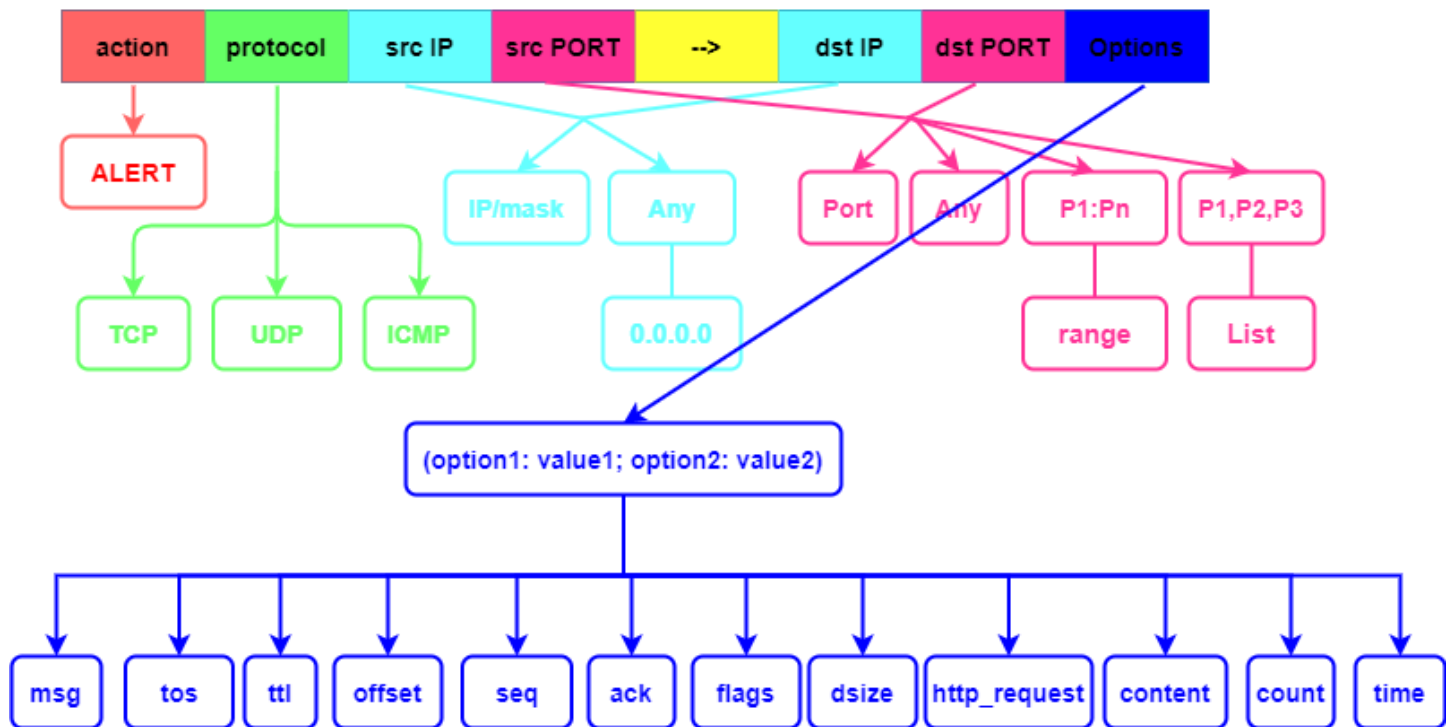


Figure 16: Format de fichier de règles du Network-Shredder IDS

Le fichier des règles est spécifié comme option du programme, à titre d'exemple:

```
root@projet:~# python3 Network-Shredder.py rules.txt
```

Ce fichier doit suivre la syntaxe suivante:

action protocol ipSource portSource -> ipDestination portDestination (option1: valeur1; option2: valeur2)

Dans ce qui suit, nous allons expliquer chaque champ d'une ligne de règle déclarée:

a/ Le champ action:

Ce champ spécifie l'action que l'IDS doit faire si la règle est vérifiée par un paquet. En ce qui concerne notre IDS, nous nous sommes limités initialement à l'action **alert**.

II- Recherche et Développement

B- Développement: Network-Shredder

b/ Le champ protocol:

Ce champ spécifie le protocole de la couche de transport (ou si le protocole de la couche réseau est **ICMP**), en ce qui concerne la couche de transport, les deux protocoles supportés sont **TCP** et **UDP**.

c/ Les champs ipSource et ipDestination:

Ce champ spécifie l'IP source et l'IP destination du paquet détecté, les cas possibles supportés sont:

- **Toutes les adresses IP:** Valeur **any** dans le fichier des règles.
- **Un intervalle d'adresses IP:** Valeur **IP/masque** (Le cas particulier d'un masque 32 pour une seule adresse est aussi pris en compte).

d/ Les champs portSource et portDestination:

Ce champ spécifie le port source et le port destination du paquet détecté (Bien sûr si le protocole de la couche transport est TCP ou UDP). Les cas possibles supportés sont:

- **Tous les ports:** Valeur **any** dans le fichier des règles.
- **Un intervalle de ports:** Valeur **PremierPort:DernierPort** (Pour un intervalle limité d'un seul côté, il suffit de mettre la valeur **PremierPort:** ou **:DernierPort**).
- **Plusieurs ports:** Valeur **Port1,Port2,Port3**.
- **Un seul port:** Valeur **Port**.

e/ Le champ des options:

Network-Shredder est compatible avec plusieurs options des champs du paquet, en effet la syntaxe utilisée pour les options est la suivante:

(option1:valeur1; option2:valeur2....)

Les options supportées par Network-Shredder sont les suivantes:

- **msg:** Le message à afficher si la règle est vérifiée par le paquet.
- **tos:** Type Of Service (Doit être un entier).
- **ttl:** Time To Leave (Doit être un entier).
- **offset:** Fragmentation Offset (Doit être un entier).
- **seq:** Sequence Number (Doit être un entier).
- **ack:** Acknowledgment Number (Doit être un entier).

II- Recherche et Développement

B- Développement: Network-Shredder

- **flags:** TCP Flags (Valeur **0** si aucun flag ou bien une chaîne de combinaison des flags suivants: **F,S,R,P,A,U**).
 - **dsiz**e: La taille de la charge du paquet (Doit être un entier).
 - **http_request:** La méthode de la requête HTTP, à savoir: **POST, GET, PUT, PATCH, DELETE**.
 - **content:** Motif à chercher dans le contenu de la charge du paquet.
 - **count:** Nombre de paquets vérifiant la règle pour déclencher l'alerte.
 - **time:** Durée dans laquelle le nombre **count** ne doit pas être dépassé.
- Notons que la présence de **count** nécessite la présence de **time**, et vice-versa.

Exemple de règles:

```
alert tcp 192.168.0.0/24 any -> 192.168.0.24 80 (msg: "Possible Local File Inclusion Detected"; http_request: "GET"; content: "../etc/passwd")
alert tcp 192.168.0.0/24 any -> 192.168.0.24 any (msg: "DOS Attack: Syn Flood Detected"; flags: S; count: 70; time: 10)
alert icmp 192.168.0.0/24 any -> 192.168.0.24 any (msg: "NMAP ping sweep Scan Detected"; dsiz
```

II- Recherche et Développement

C- Preuve du Concept - Simulation et Détection:

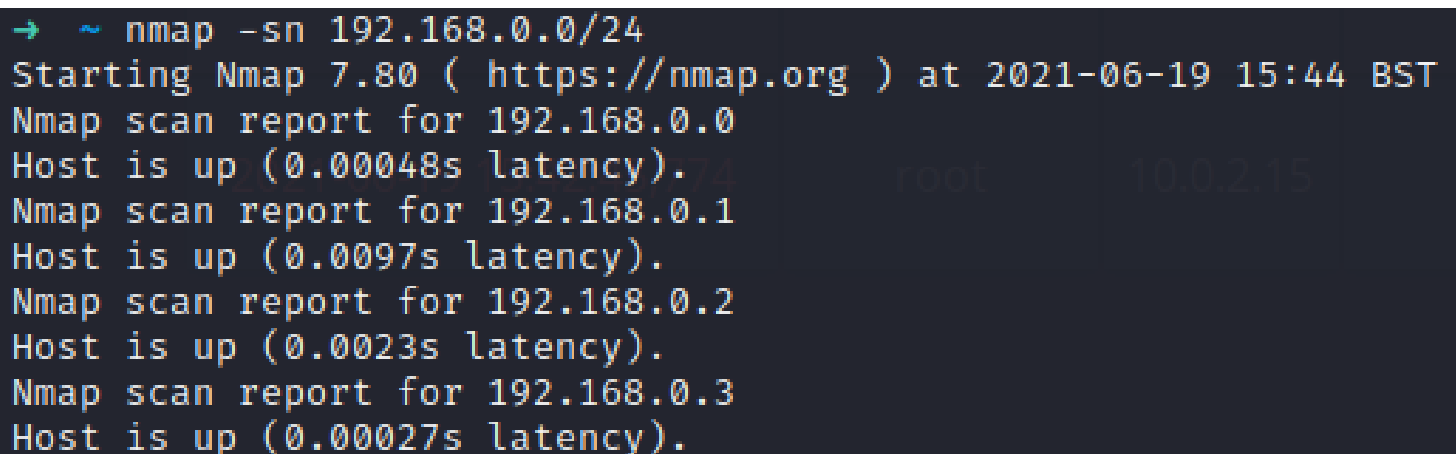
1- Ping Sweep Scan:

Avant de commencer toute attaque, l'attaquant doit scanner le réseau afin d'identifier l'état des hôtes. Pour cela, il suffit d'envoyer des paquets ICMP avec une charge vide sur l'ensemble des hôtes situés dans le réseau. Cette démarche est appelée Ping Sweep Scan.

Nous allons intégrer une règle qui détecte ce genre de scan de la façon suivante:

alert icmp any any -> any any (msg: "NMAP ping sweep Scan Detected"; dsize: 0)

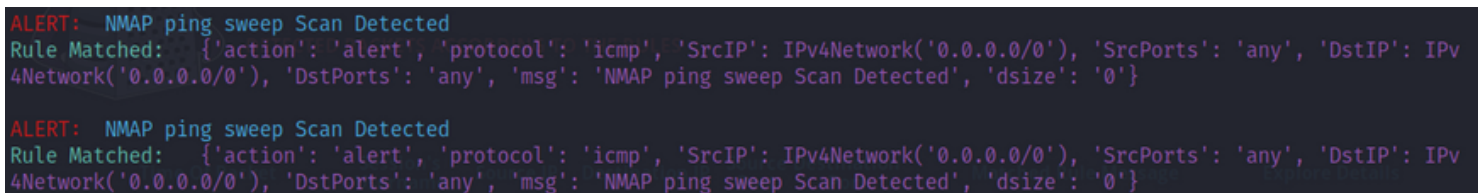
a/ Nmap ping sweep scan :



```
→ ~ nmap -sn 192.168.0.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-19 15:44 BST
Nmap scan report for 192.168.0.0
Host is up (0.00048s latency).
Nmap scan report for 192.168.0.1
Host is up (0.0097s latency).
Nmap scan report for 192.168.0.2
Host is up (0.0023s latency).
Nmap scan report for 192.168.0.3
Host is up (0.00027s latency).
```

Figure 17: Ping sweep scan avec Nmap

b/ Détection par Network-Shredder : résultats dans la Console



```
ALERT: NMAP ping sweep Scan Detected
Rule Matched: {'action': 'alert', 'protocol': 'icmp', 'SrcIP': IPv4Network('0.0.0.0/0'), 'SrcPorts': 'any', 'DstIP': IPv4Network('0.0.0.0/0'), 'DstPorts': 'any', 'msg': 'NMAP ping sweep Scan Detected', 'dsize': '0'}

ALERT: NMAP ping sweep Scan Detected
Rule Matched: {'action': 'alert', 'protocol': 'icmp', 'SrcIP': IPv4Network('0.0.0.0/0'), 'SrcPorts': 'any', 'DstIP': IPv4Network('0.0.0.0/0'), 'DstPorts': 'any', 'msg': 'NMAP ping sweep Scan Detected', 'dsize': '0'}
```


Figure 18: Network-Shredder IDS - Détection du Ping sweep [Console] -

II- Recherche et Développement

C- Preuve du Concept - Simulation et Détection:

1- Ping Sweep Scan:

c/ Détection par Network-Shredder : résultats dans l'Interface Web



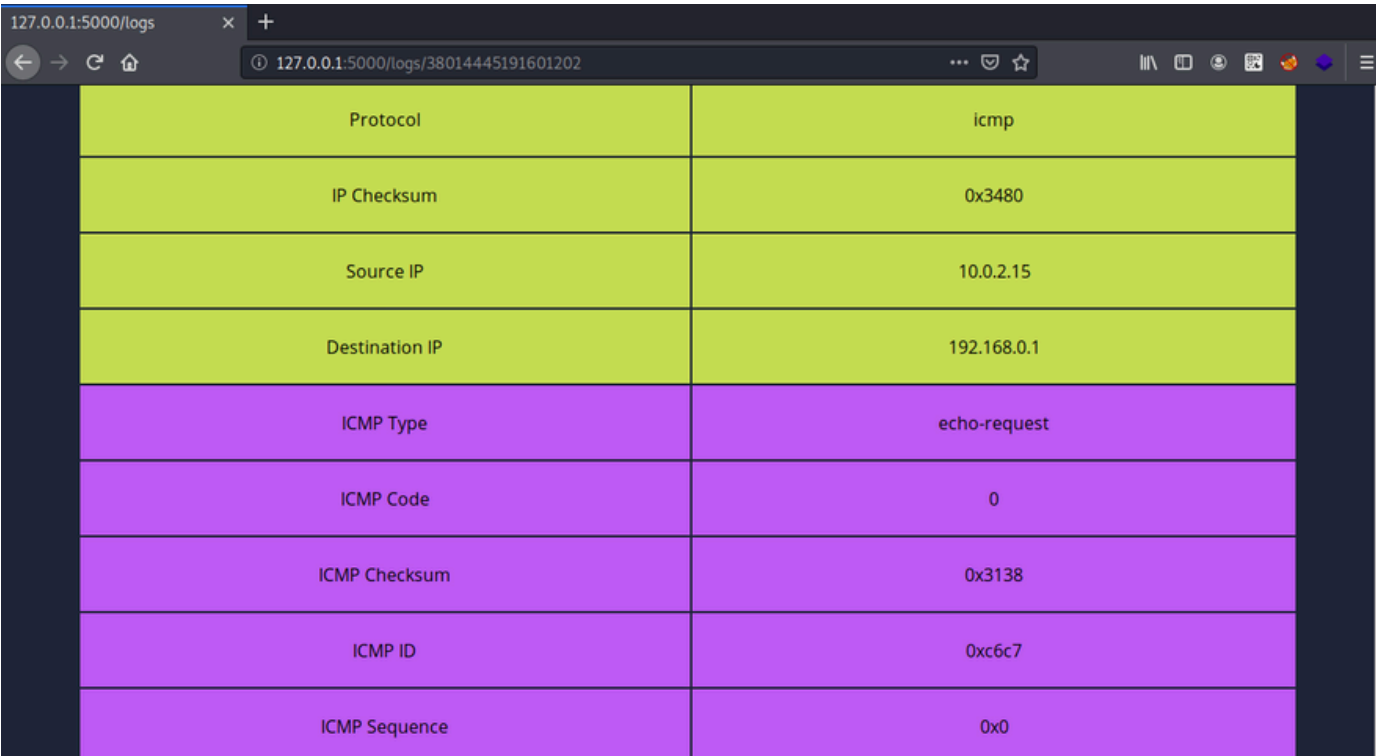
Network Shredder

DETECTED PACKETS ACCORDING TO THE RULES

Time Of Packet	Session's Username	Source IP	Destination IP	Source Port	Destination Port	Matched Rule Message	Explore Details
2021-06-19 15:44:41,083	root	10.0.2.15	192.168.0.1	None	None	NMAP ping sweep Scan Detected	Show Details
2021-06-19 15:44:41,085	root	10.0.2.15	192.168.0.2	None	None	NMAP ping sweep Scan Detected	Show Details

Figure 19: Network-Shredder IDS - Détection du Ping sweep [Web] -

d/ Détection par Network-Shredder : résultats dans l'Interface Web - détails



Protocol	icmp
IP Checksum	0x3480
Source IP	10.0.2.15
Destination IP	192.168.0.1
ICMP Type	echo-request
ICMP Code	0
ICMP Checksum	0x3138
ICMP ID	0xc6c7
ICMP Sequence	0x0

Figure 20: Network-Shredder IDS - Détection du Ping sweep [Web - détails] -

II- Recherche et Développement

C- Preuve du Concept - Simulation et Détection:

2- DOS Attack: Syn Flood:

Cette attaque utilise une faiblesse du protocole TCP en se basant sur l'envoi massif de demande d'ouverture de session SYN. L'objectif étant de saturer le nombre maximum de sessions TCP en cours. Ainsi, lorsque cette limite est atteinte, la cible ne pourra plus établir aucune session TCP causant une indisponibilité de toutes ces applications en écoute de port TCP.

Nous allons intégrer une règle qui détecte ce genre d'attaque de la façon suivante:

alert tcp any any -> any any (msg: "DOS Attack: Syn Flood Detected"; flags: S; count: 10; time: 10)

a/ DOS Attack - Syn Flood :

```
→ ~ hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood 10.0.0.1
HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.1 hping statistic ---
3349 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure 21: DOS Attack - Syn Flood avec hping3

b/ Détection par Network-Shredder : résultats dans la Console

```
ALERT: DOS Attack
Rule Matched: {'action': 'alert', 'protocol': 'tcp', 'SrcIP': IPv4Network('0.0.0.0/0'), 'SrcPorts': 'any', 'DstIP': IPv4Network('0.0.0.0/0'), 'DstPorts': 'any', 'msg': 'DOS Attack', 'flags': 'S', 'count': '10', 'time': '10'}

ALERT: DOS Attack
Rule Matched: {'action': 'alert', 'protocol': 'tcp', 'SrcIP': IPv4Network('0.0.0.0/0'), 'SrcPorts': 'any', 'DstIP': IPv4Network('0.0.0.0/0'), 'DstPorts': 'any', 'msg': 'DOS Attack', 'flags': 'S', 'count': '10', 'time': '10'}
```


Figure 22: Network-Shredder IDS - Détection du DOS Attack [Console] -

II- Recherche et Développement

C- Preuve du Concept - Simulation et Détection:

2- DOS Attack: Syn Flood:

c/ Détection par Network-Shredder : résultats dans l'Interface Web



Network Shredder

DETECTED PACKETS ACCORDING TO THE RULES

Time Of Packet	Session's Username	Source IP	Destination IP	Source Port	Destination Port	Matched Rule Message	Explore Details
2021-06-19 15:58:12,177	root	10.0.2.15	10.0.0.1	2804	http	DOS Attack	Show Details
2021-06-19 15:58:12,243	root	10.0.2.15	10.0.0.1	2814	http	DOS Attack	Show Details

Figure 23: Network-Shredder IDS - Détection du DOS Attack [Web] -

3- SMTP User Enumeration:

SMTP est un service qui peut être trouvé dans la plupart des tests de pénétration d'infrastructure. Ce service peut aider le testeur de pénétration à effectuer l'énumération du nom d'utilisateur via les commandes EXPN et VRFY si ces commandes n'ont pas été désactivées par l'administrateur système.

Le rôle de la commande EXPN est de révéler l'adresse réelle des alias des utilisateurs et des listes de courrier électronique et VRFY va ainsi confirmer l'existence de noms d'utilisateurs valides.

Nous allons intégrer une règle qui détecte ce genre d'énumération de la façon suivante:

alert tcp any any -> any 25 (msg: "Automated SMTP User Enumeration Detected"; content: "RCPT"; count: 5; time: 5)

II- Recherche et Développement

C- Preuve du Concept - Simulation et Détection:

37

3- SMTP User Enumeration::

a/ SMTP User Enumeration:

```
→ ~ nmap --script smtp-enum-users.nse -p 25 127.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-19 16:08 BST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000082s latency).

PORT      STATE SERVICE
25/tcp    open  smtp
smtp-enum-users:
  root
  admin
  administrator
  webadmin
  sysadmin
  netadmin
  guest
  user
  web
  _ test

Nmap done: 1 IP address (1 host up) scanned in 0.60 seconds
→ ~
```

```
→ ~ python3 -m smtpd -c DebuggingServer -n 127.0.0.1:25
```

Figure 24: Enumeration automatisée des utilisateurs de SMTP avec Nmap


b/ Détection par Network-Shredder : résultats dans la Console

```
ALERT: Automated SMTP User Enumeration Detected
Rule Matched: {'action': 'alert', 'protocol': 'tcp', 'SrcIP': IPv4Network('0.0.0.0/0'), 'SrcPorts': 'any', 'DstIP': IPv4Network('0.0.0.0/0'), 'DstPorts': 25, 'msg': 'Automated SMTP User Enumeration Detected', 'content': 'RCPT', 'count': '5', 'time': '5'}
```

```
ALERT: Automated SMTP User Enumeration Detected
Rule Matched: {'action': 'alert', 'protocol': 'tcp', 'SrcIP': IPv4Network('0.0.0.0/0'), 'SrcPorts': 'any', 'DstIP': IPv4Network('0.0.0.0/0'), 'DstPorts': 25, 'msg': 'Automated SMTP User Enumeration Detected', 'content': 'RCPT', 'count': '5', 'time': '5'}
```

Figure 25: Network-Shredder IDS - Détection du SMTP user Enumeration [Console] -

c/ Détection par Network-Shredder : résultats dans l'Interface Web

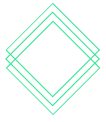


Network Shredder

DETECTED PACKETS ACCORDING TO THE RULES

Time Of Packet	Session's Username	Source IP	Destination IP	Source Port	Destination Port	Matched Rule Message	Explore Details
2021-06-19 16:07:53,966	root	127.0.0.1	127.0.0.1	39192	smtp	Automated SMTP User Enumeration Detected	Show Details
2021-06-19 16:07:53,978	root	127.0.0.1	127.0.0.1	39192	smtp	Automated SMTP User Enumeration Detected	Show Details

Figure 26: Network-Shredder IDS - Détection du SMTP user Enumeration [Web] -

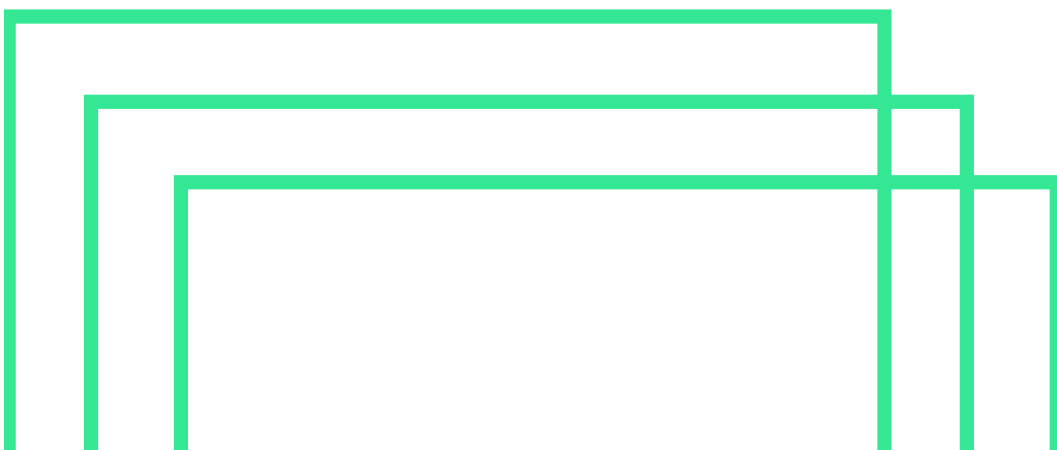


Conclusion

A fur et à mesure que les technologies de sécurité offensive évoluent vers plus de complexité en devenant plus sophistiquées, les technologies de sécurité défensive, en particulier les IDS, doivent suivre le même chemin et conjurer cet aspect qui engendra plus de risques et de menaces envers la disponibilité, l'intégrité et la confidentialité des données et des ressources.

Cela dit, les spécialistes du domaine de la détection d'intrusion ont abouti à un résultat très important: "Avec l'évolution des modèles de comportement du réseau, il est nécessaire de passer à une approche dynamique pour détecter et prévenir de telles intrusions."

Ce constat a poussé les recherches au domaine du machine learning et son application dans les système de détection d'intrusions en y intégrant les méthodes de clustering supervisé ou non-supervisé (**K-Means, Random Forest Classifier..**) en réduisant la zone d'erreur en optimisant le nombre des **false positives** et des **false negatives**.



BIBLIOGRAPHIE

- [1] **Managing Security With Snort and IDS Tools - Christopher Gerg et Kerry J. Cox - 2004**

 - [2] **Intrusion Detection Systems - P. Skrobanek - 2011**

 - [3] **Intrusion Detection Networks: A Key to Collaborative Security - Carol Fung, Raouf Boutaba - 2013**

 - [4] **Intrusion Detection: A Machine Learning Approach - Zhenwei Yu, Jeffrey J P Tsai - 2011**

 - [5] **SCADA Security: Machine Learning Concepts for Intrusion Detection and Prevention - Abdulmohsen Almalawi, Zahir Tari, Adil Fahad et Xun Yi - 2021**

 - [6] **Python Penetration Testing Cookbook: Practical recipes on implementing information gathering, network security, intrusion detection, and post-exploitation - Rehim Rejah - 2017**

 - [7] **Practical intrusion analysis: prevention and detection for the twenty-first century - Trost Ryan - 2009/2010**

 - [8] **Intrusion Detection Systems - Luigi V. Mancini et Roberto Pietro (eds.) - 2008**
- 