République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

ESI ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
Ecole nationale Supérieure d' Informatique
ex. INI (Institut National de formation en Informatique)

# RAPPORT DE STAGE D'ÉTÉ

## Thème

Revue et mise en place d'un système de détection d'intrusion (avec Snort comme exemple)

**Réalisé par :**

- AZIZ Rezak – 2SQ3
- MECHOUEK Lounes – 2SL1

**Proposé par :**

- Hôpital Chahids Mahmoudi
- Wilaya : Tizi-Ouzou

**Encadrés par :**

- Amazigh AIT-OUAKLI

***Promotion 2CS 2020/2021***

# Remerciements

Tout d'abord, nous remercions Dieu de nous avoir donné la lucidité et la patience nécessaires afin de mener ce stage à bien.

Nos plus profonds sentiments de reconnaissance et de gratitude vont vers Mr. Amazigh AIT-OUAKLI qui a su nous encadrer de la meilleure des manières. Grâce à ses interventions, ses remarques et sa grande bienveillance nous avons énormément appris durant les semaines qui viennent de s'écouler.

Nous tenons ensuite à remercier l'ensemble des employés du service Informatique de l'hôpital qui ont fait en sorte que notre stage de se déroule dans des conditions optimales malgré la situation sanitaire particulièrement difficile, sans oublier monsieur MECHOUK Mustapha sans qui rien de tout cela n'aurait été possible.

Bien évidemment, nos remerciements les plus chaleureux vont vers nos parents pour leur innombrable soutien. Nous ne saurions guère exprimer à travers des mots la gratitude et l'amour que nous leurs portons.

Enfin, nous saisissons cette opportunité pour remercier le service des stages de l'école, et plus particulièrement sa responsable Mme AIT ALI YAHIA Dahbia pour cette expérience très enrichissante. Nous lui souhaitons une agréable retraite.

# Table des matières

## Table des abréviations

| | |
|---|---|
| LAN | Local Area Network |
| SI | Système d'Information |
| IDS | Intrusion Detection System |
| NIDS | Network-based Intrusion Detection System |
| HIDS | Host-based Intrusion Detection System |
| IPS | Intrusion Prevention System |
| IPDS | Intrusion Detection and Prevention System |
| DMZ | Demilitarized Zone |

## Table des termes

| | |
|---|---|
| Switch | Équipement qui relie plusieurs segments dans un réseau informatique. |
| Routeur | Équipement réseau informatique dont le rôle est de faire transiter des paquets d'une interface réseau vers une autre. |
| Pare-feu | Logiciel et/ou un matériel qui surveille et contrôle les applications et les flux de données dans le réseau. |
| Antivirus | Logiciels conçus pour identifier, neutraliser et éliminer des logiciels malveillants. |
| Serveur | Dispositif informatique matériel et logiciel qui offre des services à un ou plusieurs clients. |
| Client | Désigne généralement un ordinateur personnel ordinaire utilisant des logiciels qui envoient des requêtes à un ou des serveur(s). |
| Fichier exécutable | Fichier contenant un programme et identifié par le système d'exploitation en tant que tel. |
| Signature numérique | Mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur. |
| Entropie | D'après Claude Shannon, il s'agit d'une fonction mathématique qui correspond à la quantité d'information contenue dans un fichier. |
| Protocole réseau | Il s'agit de la spécification de plusieurs règles permettant un type de communication particulier. |

| | |
|---|---|
| Faux Positif | Il s’agit d’un programme ou d’une communication légitime (valide) qui a été détectée comme étant nuisible. |
| Faux négatif | A contrario, il s’agit d’un programme/communication dangereuse n’ayant pas été filtrée car considérée comme légitime. |
| Routage | Il s’agit de l’opération d’acheminement de paquets réseau d’une interface A à une interface B. |
| Brute Force | Méthode visant à trouver un mot de passe ou une clé en testant, une à une, toutes les combinaisons possibles. |
| SOCKS | Protocole réseau qui permet à des applications client-serveur d'employer d'une manière transparente les services d'un pare-feu. |
| SMB | Protocole permettant le partage de ressources sur des réseaux locaux avec des PC sous Windows. |
| SSH | Protocole de communication sécurisé permettant de se connecter à une machine distante. |
| Debugger | Outil permettant d’exécuter des programmes informatiques pas-à-pas (ligne par ligne) pour en desceller les bugs. |

# Liste des figures

# Introduction générale

La popularité exponentiellement croissante qu'ont connu Internet et les systèmes informatiques durant les dernières décennies a soulevé d'innombrables problèmes liés à la sécurité des données et des infrastructures utilisées.

En effet, le nombre d'infections liées à des programmes malveillants est en augmentation sur les dix dernières années pour atteindre environ 812 millions d'intrusions en 2018 (PurpleSec, 2020) et cela met évidemment en péril la sécurité et la pérennité des individus et des organisations. Afin de contrer ce danger permanent, de nombreux mécanismes ont été développés dont les antivirus, les pare-feux et les systèmes de détection et de prévention d'intrusions.

Ce stage a été l'opportunité de mettre en pratique des connaissances théoriques que nous avons développées durant les trois dernières années. En parallèle, nous avons aussi été plongé dans un environnement de travail professionnel qui nous a permis de mieux cerner certains rôles d'un ingénieur comme l'administration réseaux et systèmes, l'administration de bases de données…etc.

Notre principal rôle a été d'implémenter un système permettant d'analyser en temps réel les communications au niveau du réseau de l'hôpital afin de détecter tout comportement suspect (susceptible d'être une intrusion) et de le signaler à l'administrateur du réseau.

Dans un souci d'organisation, nous avons décidé de segmenter ce document en quatre chapitres. Premièrement, nous présenterons l'organisme d'accueil à travers la structure de son réseau informatique ainsi que la problématique ayant poussé à la réalisation de ce projet. La deuxième partie sera consacrée à une présentation générale de la sécurité informatique ainsi que les types de menaces existantes. Le troisième chapitre expliquera quant à lui le fonctionnement d'un système de détection d'intrusion. Finalement, nous exposerons la solution que nous avons proposé à travers les différentes étapes de sa conception et de sa réalisation.

# Chapitre 1 : Présentation de l'organisme d'accueil

## 1. Vue d'ensemble

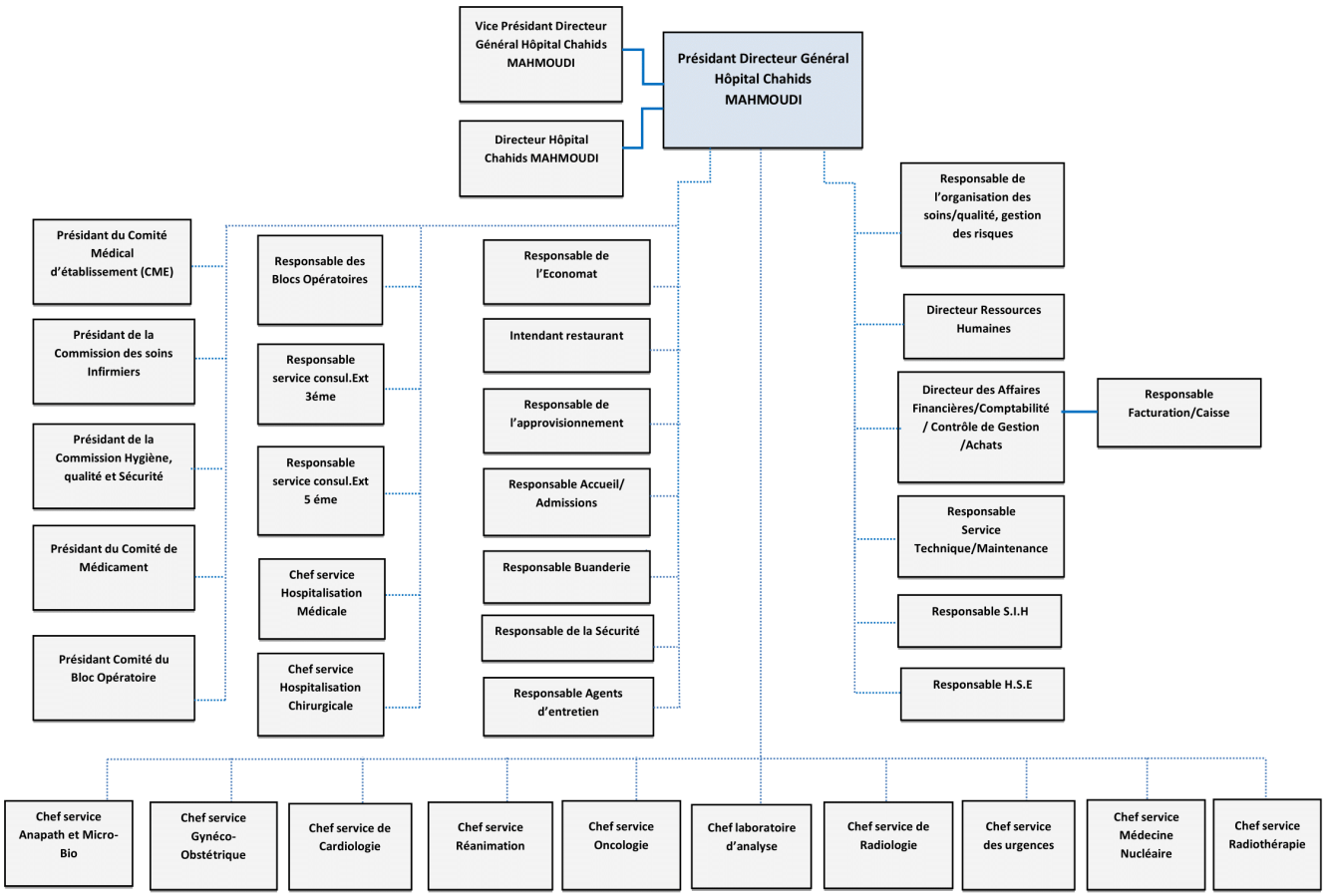
L'Hôpital Chahids Mahmoudi (HCM), conçu et réalisé par le Docteur Said MAHMOUDI (Radiologue et Vice-président de la Société Algérienne de Radiologie et d'Imagerie Médicale, « SARIM »), et membre du comité national du Plan Cancer, est un Etablissement Hospitalier Privé offrant des prises en charge dans toutes les spécialités médico-chirurgicales, y compris un pavillon des urgences fonctionnel 24 heures sur 24, 7 jours sur 7.

Les offres de soins reposent sur un plateau technique performant, comprenant essentiellement : un département de radiologie, d'un laboratoire d'analyses médicales complètes, d'anatomo-pathologie et cytologie, de microbiologie et de médecine moléculaire.

Par ailleurs, l'HCM est équipé pour aspirer à être un centre d'excellence pour le diagnostic et le traitement des pathologies cancéreuses. Dans ce contexte, il en est même honoré d'être le premier hôpital algérien à mettre à la disposition des concitoyens la technique du PET Scan, utilisé entre autres pour le diagnostic et le suivi des traitements des pathologies cancéreuses. Et afin de compléter ce segment, nous disposons aussi d'une unité de médecine nucléaire et d'imagerie moléculaire.[1]

[1] Site web de l'hôpital hcm-dz.com

## 2. Organigramme




Notre stage s'est effectué sous l'encadrement du responsable du système d'information hospitalier. Cela nous a permis de découvrir différentes tâches au sein de ce service. Parmi ces tâches (liste non exhaustive) :

- Améliorer, maintenir et administrer le système d'information hospitalier
- Définition des orientations technologiques de l'hôpital
- Pilotage des projets informatiques
- Conseil et assistance des instances de l'établissement

## 3. Situation informatique

L'hôpital dispose d'un réseau local reliant les différents départements à travers différents switchs. Ils sont tous reliés à un switch fédérateur qui est connecté au Datacenter. Ce dernier regroupe l'ensemble des serveurs hébergeant les différentes bases de données ainsi que les applications d'imagerie médicale et de traitement d'images.

Le trafic transitant vers l'extérieur passe par un routeur doté d'un pare-feu assurant le blocage de certains échanges sur des ports définis au préalable. Il assure également la fonction de VPN

afin de permettre l'accès à distance au réseau LAN de l'hôpital pour les médecins et les administrateurs du système.

## 4. Problématique

Dans une période de pandémie où les attaques informatiques pullulent, notamment par l'intermédiaire de logiciels d'extorsion[2], les structures de santé sont particulièrement vulnérables.

Disposant de données sensibles et cruciales pour leurs patients, les hôpitaux ne peuvent pas se permettre que ces dernières courent le moindre de risque de corruption ; et les malfaiteurs le savent.

En effet, d'après un rapport de la société informatique **Forescout** c'est environ 400 hôpitaux qui ont été victimes de cyberattaques au cours des dernières semaines aux États-Unis et en Grande Bretagne.[3]

L'HCM dispose d'un pare-feu limitant les interactions avec l'extérieur et permettant de bloquer un grand nombre de menaces. Néanmoins, certains services de base tels que la messagerie ou l'accès à des sites web par des machines clientes ne peuvent être bloqués même s'ils peuvent éventuellement représenter une menace de sécurité. Il est donc primordial de disposer d'un système fiable étant capable d'intercepter tout trafic suspect et de le signaler.

[2] Un logiciel d'extorsion est un logiciel malveillant qui prend en otage des données personnelles. Pour ce faire, un rançongiciel chiffre des données personnelles puis demande à leur propriétaire d'envoyer de l'argent en échange de la clé qui permettra de les déchiffrer.

[3] Forescout Research Labs, *Connected Medical Device Security : A Deep Dive into Healthcare Networks,* 2020

# Chapitre 2 : Généralités sur la sécurité Informatique

## 1. Définition de la sécurité Informatique

La sécurité informatique est un domaine qui englobe l'ensemble des dispositifs physiques, logiciels et humains permettant de garantir l'intégrité, la disponibilité et la confidentialité des ressources des systèmes informatiques (Stalling, 2017).

Elle assure donc trois principaux services : la confidentialité, l'intégrité et la disponibilité (appelés **CIA triad**) ; auxquels peuvent s'ajouter deux autres concepts complémentaires : l'authenticité et la responsabilité (Stalling, 2017).

- **La confidentialité** : Protection contre la consultation abusive des données par des entités tierces indésirables.
- **L'intégrité** : Faire en sorte que l'information ne soit pas altérée ou détruite en assurant la non-répudiation et l'authenticité de cette dernière.
- **La disponibilité** : Garantir un accès fiable à l'information et son utilisation à tout instant.
- **L'authenticité** : Assurer qu'un message, une transmission ou un interlocuteur est bien ce qu'il prétend être (A l'aide notamment des signatures électroniques).
- **La responsabilité** : Les actions d'une entité doivent être reliées à cette dernière. Le risque zéro pour un système informatique étant inexistant à l'heure actuelle, il est primordial de tracer les brèches de sécurité et de les attribuer aux entités responsables à l'aide notamment de : la non-répudiation, la dissuasion, la détection et la prévention d'intrusion, la récupération suite à une attaque et l'entreprise d'actions judiciaires.

## 2. Menaces sur les systèmes d'information

Les SI peuvent être menacés de plusieurs manières. Dans un premier temps, les composantes matérielles peuvent être sujettes à corruption de manière intentionnelle ou non à travers leur sabotage, des pannes ou erreurs, etc.

Les menaces peuvent aussi être liées à l'utilisation des réseaux. En effet, à travers des actions malveillantes une partie tierce peut exploiter certaines failles de sécurité (comme celles des protocoles SMB, SSH, etc.) et infiltrer le réseau. (Mouna Jouinia, 2014)

## 3. Types de malwares

Un malware, ou « logiciel malveillant » est un terme générique qui décrit tous les programmes malveillants qui peuvent être nocifs pour les systèmes (Malwarebytes, s.d.).

Afin d'effectuer une analyse complète et confirmer ses hypothèses lors d'une éventuelle intrusion, il est primordial de connaître les différents malwares existants :

- **Virus :** Un virus est un logiciel qui s'attache à tout type de document électronique « hôtes », et dont le but est d'infecter ceux-ci et de se propager sur d'autres documents et d'autres ordinateurs. Un virus a besoin d'une intervention humaine pour se propager (AMADEUS, 2016).
- **Vers :** Un ver se reproduit en s'envoyant à travers un réseau (e-mail, Bluetooth,…). Le ver n'a pas besoin de l'interaction humaine pour pouvoir se proliférer (AMADEUS, 2016).
- **Virus réticulaire (Botnet)** : Il s'agit d'un malware qui s'installe sur un grand nombre de machines sans y commettre le moindre dégât. Elles sont alors commandées par un serveur « command-and-control » qui peut décider de les utiliser pour mener des actions malveillantes (Bloch & Wolfhugel, 2009).
- **Backdoor (Porte dérobée)** : Une porte dérobée (backdoor) est un logiciel de communication caché, installé par exemple par un virus ou par un cheval de Troie, qui donne à un agresseur extérieur accès à l'ordinateur victime, par le réseau (Bloch & Wolfhugel, 2009).
- **Un logiciel espion** : Comme son nom l'indique, il collecte à l'insu de l'utilisateur légitime des informations au sein du système où il est installé, et les communique à un agent extérieur, par exemple au moyen d'une porte dérobée (Bloch & Wolfhugel, 2009).
- **Rançongiciel (Ransomware)** : Programme qui crypte les données de la machine infectée et qui propose de les restituer en échange d'une rançon (Sikorski & Honig, 2012).
- **Scareware** : Il s'agit d'un malware dont le principe est de faire peur à l'utilisateur en lui faisant croire que son système est en danger. Le scareware se fait généralement passer pour un antivirus légitime, le but étant de pousser l'utilisateur à payer pour que son système soit « nettoyé » (Sikorski & Honig, 2012).
- **Rootkit** : Programmes dont le rôle premier est de cacher l'existence d'autres malwares comme les backdoors (Sikorski & Honig, 2012).

## 4. Sécurisation d'un SI

Un système d'information peut être sécurisé sur plusieurs niveaux :

- **Sécurité Physique** : Protéger les entités physiques sensibles et les endroits les contenant en y limitant l'accès.
- **Sécurité du Personnel** : Délimiter clairement les droits d'accès de chacun pour éviter toute probable erreur de manipulation ou fuite de données.
- **Sécurité du Réseau** : Protéger les composantes du réseau, ses connexions et son contenu.
- **Sécurité des Informations** : Assurer l'intégrité et la disponibilité de l'information lors de son stockage, traitement et transmission.
- **Sécurité des communications** : Protéger les supports de communication et leur contenu.
- **Sécurité des opérations** : Protection des échanges de données ainsi que les systèmes informatiques.

Au niveau de l'organisation, cette sécurisation se fait en général en se basant sur un ensemble de règles formalisées basées sur une analyse de risques que l'on appelle Politique de Sécurité. (Introduction à la Sécurité Informatique (1CS - ESI), 2020)

## 5. Analyse de malwares

## A. Cas théorique : Méthodes d'analyse d'un malware quelconque

### Définition

L'analyse des malwares est l'art de disséquer ces derniers pour comprendre leur fonctionnement ainsi que les différentes façons de les identifier, les vaincre ou les éliminer (Sikorski & Honig, 2012).

L'analyse de malwares se structure en deux grandes parties :

- L'analyse statique qui consiste à extraire des informations depuis le fichier exécutable sans le lancer.
- L'analyse dynamique dont le but est d'obtenir des informations sur le comportement du malware après avoir lancé le fichier exécutable.

### Étapes et outils de l'analyse

**Étape 1 : Analyse statique**

**Utiliser des antivirus**

La première étape de l'analyse statique consiste à effectuer un scan du fichier suspect à l'aide d'un ou plusieurs antivirus.

Bien que cette méthode ne soit pas sans failles, notamment car les antivirus utilisent une base de données de signatures que les créateurs de malwares peuvent contourner, elle permet néanmoins d'offrir une première impression sur l'état du fichier (Sikorski & Honig, 2012).

Durant notre étude, nous nous appuierons sur l'outil en ligne VirusTotal.com qui permet d'inspecter des fichiers et des URL à l'aide de plus de 70 scanners antivirus et des listes noires d'URL et de domaines (VirusTotal, s.d.).

**Analyser les chaînes de caractères**

L'étape suivante consiste à inspecter les chaînes de caractères du fichier afin d'y déceler les fonctions ou les chaines de caractères auxquelles ce dernier fait appel. Ces fonctions peuvent en dire long sur le fonctionnement du programme ; par exemple s'il fait appel à une fonction de connexion nous pourrions en déduire qu'il communique éventuellement avec un serveur distant.

Nous utiliserons l'outil **PeStudio** qui permet d'exécuter l'analyse d'un fichier exécutable ou DLL sous Windows et d'afficher sa structure sans le lancer.

**Détecter le Packing (emballage)**

Le packing de programmes est un procédé très courant qui consiste à « brouiller » le code source et le rendre difficile à analyser. Il est notamment utilisé pour éviter que des programmes ne soient crackés.

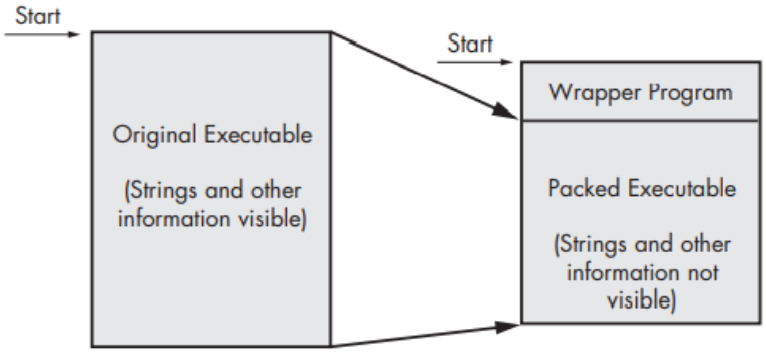



*Figure 2: Différences entre un fichier emballé et non emballé (Sikorski & Honig, 2012)*

Néanmoins, il arrive que les concepteurs de malwares utilisent ce procédé afin de cacher certaines chaînes de caractères et tenter de faire passer leurs programmes malveillants pour des programmes légitimes (Sikorski & Honig, 2012).

Durant notre analyse, nous utiliserons l'outil Detect it easy (**DIE**) qui permettra dans un premier temps de savoir si le fichier étudié est emballé ou non grâce à une analyse par entropie. Cette dernière est par définition une mesure du niveau de chaos et de désordre. Un fichier .exe comporte une entropie moyenne. un fichier .exe possède des zones qui ont une entropie forte, et d'autres une entropie faible (securiteinfo, s.d.).

Dans le cas où ce dernier est bien emballé, nous essaierons d'obtenir le type de packer qui a été utilisé dans ce but grâce à l'outil **PeiD**.

Enfin, le dernier outil dont nous aurons besoin dans cette étape est Unpac.me qui est un service automatisé de décompression de logiciels malveillants.

**Désassembler le fichier**

Une fois le package levé, nous pouvons avoir accès au code source du programme en langage assembleur. Grâce à cela, nous aurons une idée précise des fonctions appelées, leur contexte et l'endroit du code où l'appel se fait.

Le désassembleur IDA Pro nous permettra d'effectuer cette opération.

## Étape 2 : Analyse dynamique

**Utiliser un environnement sécurisé**

Afin d'éviter d'infecter sa machine en lançant le malware, il est primordial d'utiliser un « laboratoire de test », à savoir une machine virtuelle.

Cependant, certains malwares peuvent reconnaître lorsqu'ils s'exécutent sur une machine virtuelle (grâce à des instructions machine spécifiques comme cpuid) et changer leur comportement en conséquence (Sikorski & Honig, 2012).

**Observer le comportement du programme**

Dans le but de connaitre le comportement d'un malware, il est intéressant de capturer l'état des registres Windows avant et après le lancement de malware. Cela peut se faire grâce à l'outil Regshot. En effet, les malwares utilisent généralement ce registre afin d'altérer le fonctionnement de la machine infectée. De plus, ils peuvent aussi modifier certaines clés de

registres dans un but malveillant. Par exemple : définir une action récurrente à effectuer à chaque allumage de la machine (Sikorski & Honig, 2012).

Une fois le programme suspect lancé, on peut commencer à le surveiller grâce à la série d'outils Windows « Sysinternals Suite » développée par **Mark Russinovich,** CTO de Microsoft Azure. Parmi ces outils figurent :

- **Process Explorer :** affiche une arborescence détaillée de tous les processus en exécution sur le système ainsi que la validité de leur signature.
- **Autoruns** : affiche les autoruns de tous les programmes du système.
- **TCPView :** permet de montrer la liste détaillée de toutes les connexions TCP et UDP du système ainsi que leur état.

De plus, on utilisera l'outil **Wireshark** qui est l'analyseur de protocoles réseaux le plus utilisé au monde pour capturer les paquets envoyés par le programme et les analyser.

**Débugger le programme**

En dernier lieu, nous utiliserons IDA Pro en mode debugger afin d'exécuter pas-à-pas le malware et ainsi connaitre son fonctionnement au point d'appel des fonction suspecte.

Il est à noter par ailleurs que certains malwares peuvent détecter l'utilisation d'un debugger ( en utilisant la fonction « IsDebuggerPresent » de l'API Windows par exemple) et ainsi changer de comportement ou éviter de s'exécuter complètement (Sikorski & Honig, 2012).

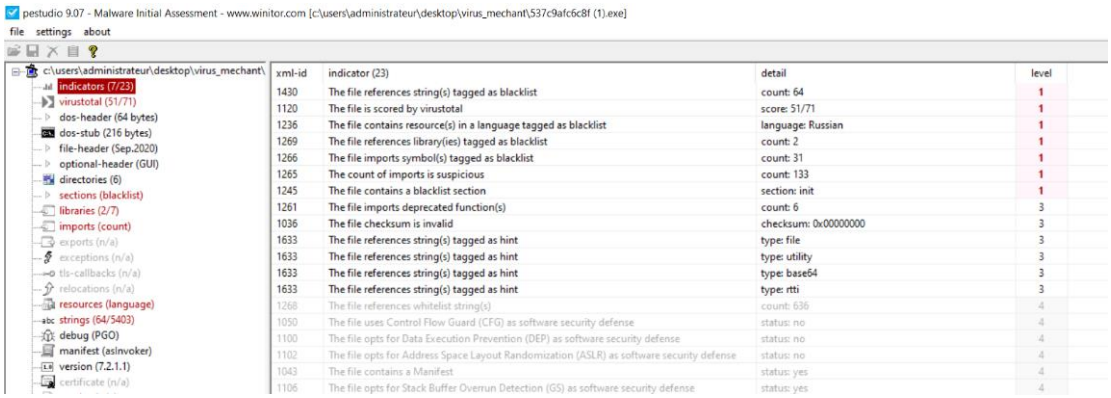
## B. Cas pratique : Analyse du malware « cloudnet.exe »

Afin d'illustrer un cas concret d'analyse de malware, un fichier **éventuellement corrompu** nous a été fourni. On ne connait pas son code source et on procède à son analyse.

### Analyse statique

On effectue une première analyse via **VirusTotal**


On remarque que le fichier a été détecté comme étant une menace par 51 antivirus.




Utilisons à présent l'outil **PeStudio** pour obtenir plus de détails.

Dans un premier temps, on remarque qu'il y a 7 indicateurs suspects (2 librairies, des chaînes de caractères, etc.)

Poussons la recherche plus loin pour les chaînes de caractères récupérées et les fonctions importées.

| | name (133) |
|---|---|
| Unknown error (%d) | InternetCheckConnectio... |
| init | SleepEx |
| InternetCheckConnection | QueueUserAPC |
| MoveFileEx | TerminateThread |
| SleepEx | VerSetConditionMask |
| QueueUserAPC | GetTempFileNameW |
| TerminateThread | DeleteFileW |
| VerSetConditionMask | RemoveDirectoryW |
| GetTempFileName | ReadConsoleInputW |
| DeleteFile | VirtualProtect |
| RemoveDirectory | GetModuleFileNameW |
| ReadConsoleInput | CreateProcessW |
| VirtualProtect | SetEnvironmentVariableW |
| GetModuleFileName | MoveFileExW |
| CreateProcess | GetEnvironmentStringsW |
| OpenProcessToken | GetTimeZoneInformation |
| RegSetValueEx | |
| ConvertSidToStringSid | |
| RegDeleteValue | |
| WSASend | |

*Figure : Fonctions utilisées par le programme*

A partir de là, nous pouvons émettre des hypothèses quant au rôle du malware :

- *InternetCheckConnection* : le malware vérifie s'il est connecté à internet.
- *DeleteFile* et *DeleteDirectory* : le malware supprime certains fichiers et certains dossiers.
- *VirtualProtect* : le malware manipule la mémoire virtuelle.
- *CreateProcess* : Ça nous laisse penser que le malware lance d'autre programmes.
- *RegSetValueEx* et *RegDeleteValue* : le malware manipule des valeurs dans le registre.
- *WSASend* : le malware envoie des données sur internet

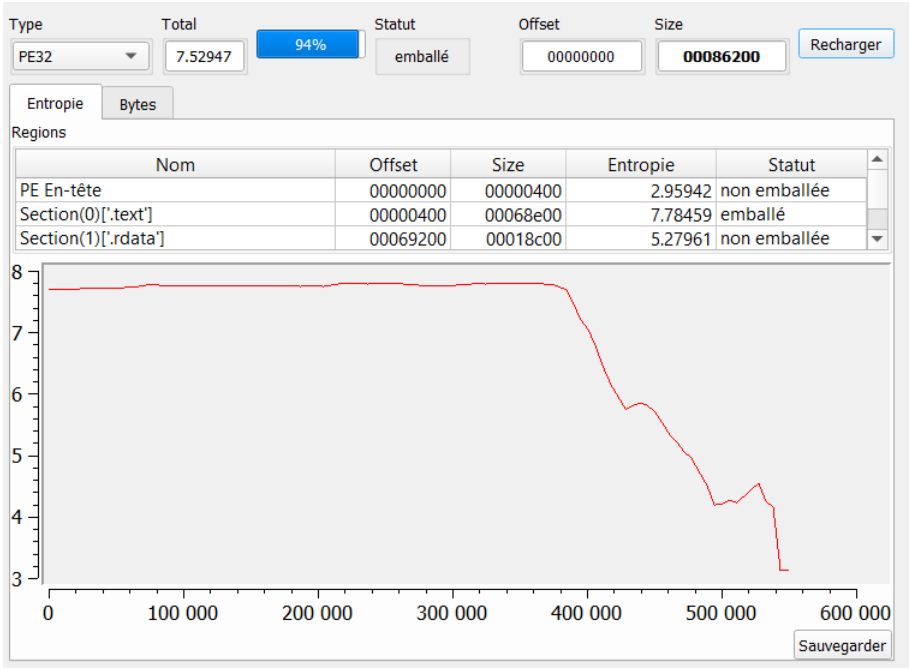
Pour résumer, il semble très probable que le processus récupère des données et les envoie sur Internet.

Afin de vérifier notre hypothèse, on va effectuer un désassemblage du fichier exécutable avec l'outil **IDA Pro**.

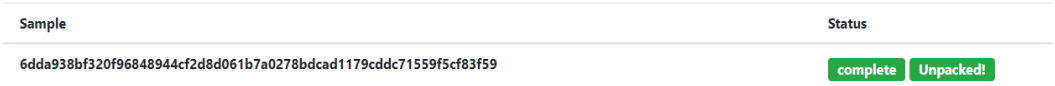
```
  extrn WSACleanup:dword
l WSASend(SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount, LPDWORD lpNumberOfBytesSent, DWORD dwFlags, LPW
  extrn WSASend:dword
l select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, const struct timeval *timeout)
  extrn select:dword
l connect(SOCKET s, const struct sockaddr *name, int namelen)
  extrn connect:dword
```

Une fois fait, on voit que ces fonctions ne sont pas référencées. Néanmoins, pour s'en assurer on vérifie tout de même si le fichier n'est pas **packé**.

L'outil DIE.exe montre bien que le segment « .text » a une **entropie** bien plus élevée que la normale et donc le fichier est très probablement **packé** (emballé).



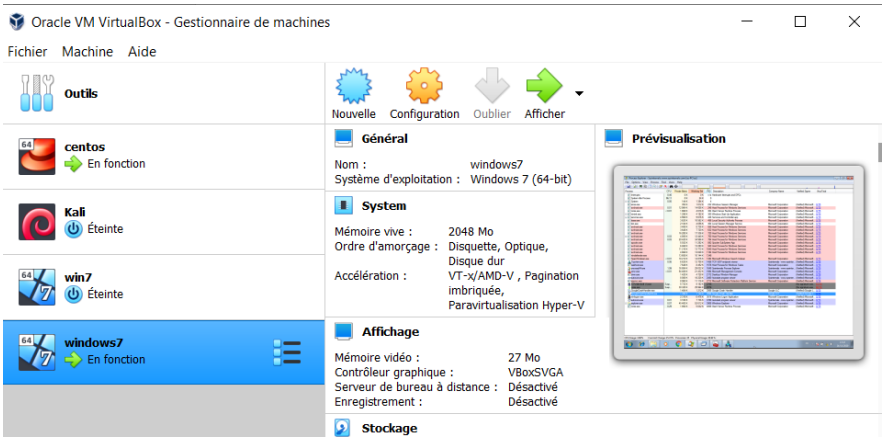

Grâce à l'outil en ligne **unpac.me**, nous réussissons à unpacker le fichier




Lorsqu'on rouvre le fichier obtenu une nouvelle fois avec **IDA**, on remarque que les fonctions de connexion sont bien référencées.

```
                                        ; sub_40A650+60↑p ...
; int __stdcall WSASend(SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount, LPDWORD
                extrn WSASend:dword     ; CODE XREF: sub_410610+2A↑p
                                        ; DATA XREF: sub_410610+2A↑r
; int __stdcall select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *except
                extrn select:dword      ; CODE XREF: sub_40E1B0+1F1↑p
                                        ; sub_40E4F0+109↑p ...
; int __stdcall connect(SOCKET s, const struct sockaddr *name, int namelen)
                extrn connect:dword     ; CODE XREF: sub_4101B0+4A↑p
                                        ; DATA XREF: sub_4101B0+4A↑r
```

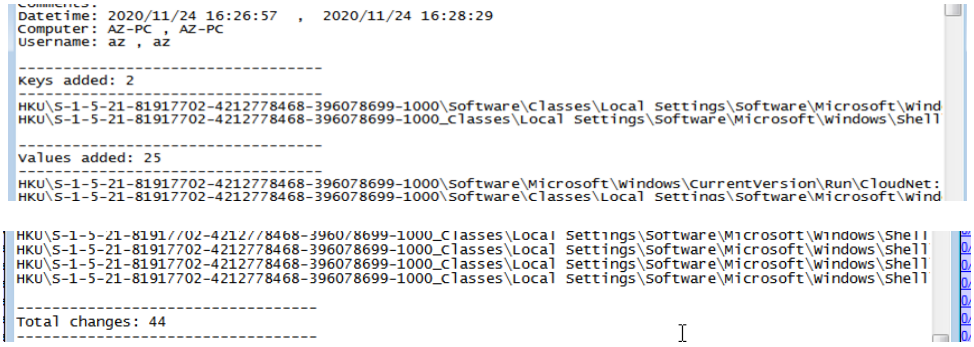
## Analyse dynamique




Dans cette partie, nous allons effectuer un laboratoire de test sous **VirtualBox**. Nous créons une machine virtuelle sous Windows 7, nous l'infectons et observons comment le malware se comporte.

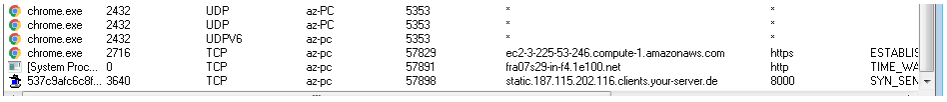
Nous vérifions via **regshot** si des clés de registres ont été modifiées (Pour éventuellement se lancer au démarrage).

On remarque qu'au total 44 clés de registres ont été modifiées.

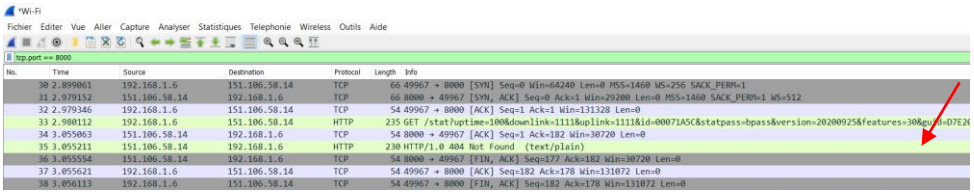



Essayons à présent de comprendre ce que le malware envoie sur Internet en procédant à une capture du trafic réseau. On utilise l'un des outils **SysInternals** qui est **TcpView** ainsi que l'outil **Wireshark**.

**TcpView** permet d'obtenir des informations sur les connexions que le malware établit tandis que **Wireshark** nous permet d'analyser ce qui est échangé.




**TcpView** montre que le malware utilise le port **8000** pour les connexions et qu'il communique avec plusieurs **adresses IP**. Grâce à **Wireshark**, on filtre les données selon le filtre **tcp.port==8000** :




Il en découle que le malware envoie une requête http inhabituelle.

Grâce à une recherche sur google de cette syntaxe, nous trouvons que ce malware est déjà connu. D'après un article[4] paru en Juin 2020, ce malware appartiendrait à un botnet lié à une opération massive qui aurait touché près de **500.000** utilisateurs web.

On y apprend que ce malware fonctionne comme suit :

- La machine infectée envoie une requête sur le port 8000 vers son serveur « **Command and control** » puis elle doit recevoir un numéro de session sur lequel se connecter comme suit :

```
GET /stat?
uptime=100&downlink=1111&uplink=1111&id=01BA8E7E&statpass=bpass&version=20200414&features=30&guid=1AF76704-0612-45BC-
AAE1-DB1A3A635E9E&comment=20200414&p=0&s= HTTP/1.0

HTTP/1.1 200 OK
Content-Length: 11

session:444
```

*Figure 13: An example of the bot (red text) communicating with its server (blue text)*

- A partir de là, le serveur peut donner des instructions à ce bot et ce dernier envoie des réponses. Cependant, dans notre cas nous obtenons une réponse 404 au lieu de 200.

Selon un autre article[5] de *Jaromir Horejsi*, *Joseph C. Chen* paru dans Trendmicro : La requête http est censée retourner une erreur « http 404 » jusqu'à ce que l'administrateur du malware envoie une commande (Horejsi & C Chen,, 2019).

Ce dernier retourne alors un fichier **RSC** qui est le format d'une configuration de RouterOS qui est un système d'exploitation du routeur Microtik. Suite à cela, il pourra configurer le routeur de la victime comme un SOCKS proxy grâce à d'autres composants de ce virus qui sont téléchargés au fur et à mesure (Horejsi & C Chen,, 2019).

[4] Luca Nagy, *Glupteba: Hidden Malware Delivery in Plain Sight,* Juin 2020

[5] Jaromir Horejsi, Joseph C Chen, *Glupteba Hits Routers and Updates C&C Servers,* Avril 2019

## Suppression du malware

Pour supprimer le malware, on se basera sur la suite **Sysinternals** en suivant les étapes décrites par Mark Russinovich (CTO de Microsoft Azure et co créateur des outils Sysinternals) dans l'une de ses vidéos sur YouTube[6].

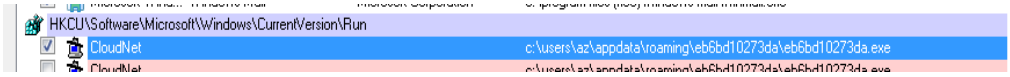
- On se déconnecte d'abord du réseau pour éviter une réinfection par le serveur.
- On suspend le processus principal du malware puis on l'arrête (kill) avec **Process Explorer** pour éviter qu'il se réactive.


- La prochaine étape consiste à nettoyer toute trace d'autostarts du malware, c'est-à-dire les chemins qui lui permettent de s'exécuter de manière automatique.

Pour cela, on utilise l'outil **Autoruns** faisant bien attention à vérifier tous les chemins suspects (car il peut avoir d'autres dépendances qui ont des noms de processus système usuels).

Dans un premier temps on décoche la case des processus qui n'ont pas de **signature valide** et qui sont reconnus par **VirusTotal** comme étant corrompus, cela les empêchera de se lancer automatiquement.

Dans un second temps, on supprimera ces entrées-là (après vérification pour éviter de supprimer de véritables processus système).




Il suffit maintenant de supprimer les fichiers liés au malware et de redémarrer sa machine. On pourra éventuellement lancer une réparation des registres.

[6] Mark Russinovich, *Malware Hunting with Mark Russinovich and the Sysinternals Tools*, YouTube Juillet 2020

# Chapitre 3 : Les systèmes de détection d'intrusion (IDS)

## 1. Définition d'une intrusion

Toute activité non autorisée et nuisible lancée par un utilisateur tiers de manière locale ou distante est appelée intrusion informatique (Sitesh Sinha, 2017).

## 2. Définition d'un IDS

Le système de détection d'intrusion est tout dispositif matériel, logiciel ou combinaison des deux, dont le rôle est de détecter toute activités d'intrusion (Ur Rehman, 2003) et de les signaler afin que des dispositions soient prises pour les contrer avant que le système ne soit compromis.

## 3. Types d'IDS

On distingue deux types de systèmes de détection d'intrusion, HIDS et NIDS.

- **HIDS (Host-Based IDS)** : Les systèmes de détection d'intrusion hôte. Ils sont déployés au niveau de machines hôtes. Ils peuvent consulter les fichiers du système et des applications, collecter et analyser les activités de système pour détecter les activités malveillantes (Ur Rehman, 2003). Ce type d'IDS est souvent comparé à un antivirus, néanmoins une différence les distingue : il est plus poussé, un antivirus détecte que les actions malveillantes alors qu'un HIDS peut détecter un dépassement de tampon par exemple.
- **NIDS (Network Based IDS) :** Ce sont des systèmes de détection d'intrusion réseaux. Ils sont déployés à des niveaux stratégiques du réseau et surveillent le trafic vers d'autres hôtes (Koziol, 2003). Ils analysent le trafic qu'ils reçoivent pour détecter les actions malveillantes.

Il est à noter que l'on peut combiner les deux méthodes. Une méthode n'exclue pas l'autre et chacune est utilisée selon des objectifs à atteindre.

## 4. Méthodes de détection

Les IDS utilisent différentes méthodes pour détecter les intrusions. Parmi ces techniques on distingue principalement la détection par signature et la détection par anomalie.

1- **Détection par signature :** (Ali, Lu, & Tavalaee, 2010)

Cette méthode est dite aussi approche par scénario. Elle repose sur la création a priori d'une base de motifs représentant des scénarios d'attaque connus au préalable. Les intrusions sont détectées par correspondance de la signature d'une action aux motifs déjà présent dans la base.

Pour implémenter cette méthode on utilise principalement les techniques suivantes :

- **Correspondance de signature (Pattern matching)** : elle consiste à comparer des données analysées à des signatures d'attaques bien connus.
- **Système expert basé sur des règles (Rule based expert system)** : elle garde une base de données de règles pour tous les scénarios possibles d'intrusion.
- **Technique basée sur les états de transitions** : (state-based techniques) : les scénarios sont définis sous forme de diagrammes de transition d'états. Les activités contribuant à des scénarios d'intrusion sont définies comme des transitions entre les états du système.

**2- Détection par anomalie** (Khraisat , Gondal, Vamplew, & Kamruzzaman, 2019)

Cette méthode est dite aussi approche comportementale. Elle repose sur l'hypothèse que toutes les activités intrusives sont anormales. Cela revient à établir des profils d'activités normaux. On retrouve trois techniques de détection d'anomalies :

a- **Modèles statistiques** : Définir des modèles statistiques pour détecter des comportements anormaux

b- **Approches basées sur la connaissance** : Elles nécessitent la création d'une base de comportements normaux comme par exemple le trafic légitime. Dès qu'une action n'est pas dans cette base, une intrusion est signalée.

c- **Approches basées sur le Machine Learning** : Elles consistent à extraire des connaissances à partir de grandes quantités de données et améliorer la précision de la détection en éliminant les faux positifs et les faux négatifs.

**3- Détection hybride :**
Elle consiste à combiner les deux méthodes précédentes.

## 5. Architecture d'un IDS

Selon les travaux de (Hiet, 2008), un IDS est composé classiquement de trois composants : Un capteur chargé de collecter les informations sur l'évolution de l'état de système. Un analyseur qui détermine si un ensemble d'évènements produits par le capteur est

caractéristique d'une activité malveillante. Un manager qui collecte les alertes produites par le capteur, les met en forme et les présente à l'opérateur.

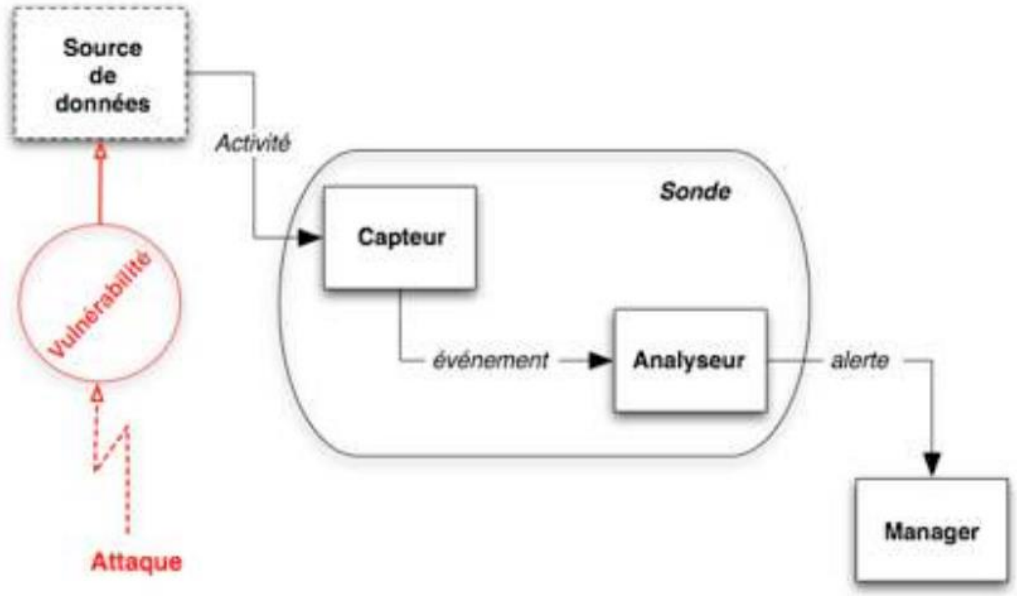



*Figure 17: Architecture classique d'un IDS (Hiet, 2008)*

**Le capteur :**

Le capteur collecte des informations sur l'activité du système puis les fournit à l'analyseur. Le capteur peut transmettre directement ces données, mais en général il effectue un prétraitement dans le but de filtrer des données considérées comme non pertinentes afin de limiter la quantité d'information à analyser. Ce capteur réalise généralement une mise en forme des données pour les présenter à l'analyseur sous un format précis (Hiet, 2008).

**L'analyseur**

L'objectif de l'analyseur est de déterminer si les données fournies par le capteur contiennent des éléments caractéristiques d'une activité malveillante. Pour cela on utilise l'une des approches déjà cités (approche par scénario ou approche comportementale) (Hiet, 2008).

**Le manager**

Le manager est responsable de la présentation des alertes à l'administrateur. Il peut aussi assurer le traitement de l'incident mais actuellement les réactions sont rarement utilisées car elles peuvent se traduire par un déni de service en cas de réaction a un faux positif (Hiet, 2008).

# Chapitre 4 : Mise en place d'un IDS (exemple de Snort)

## 1. Introduction à Snort

Snort est un système de détection et de prévention d'intrusions[7] open-source initié par Martin Roesch en 1998. Il est actuellement développé par Cisco[8].

Snort offre actuellement de hautes performances ainsi qu'une large variété de fonctionnalités, certains vont même jusqu'à le définir comme étant l'un des « meilleurs logiciels open source de tous les temps »[9].

Bien qu'il ait été pensé à l'origine pour fonctionner en tant qu'analyseur de paquets réseaux, il a depuis bien évolué pour devenir un incontournable de la détection d'intrusions aussi bien pour les réseaux domestiques que pour les entreprises. En effet, la vitesse, la stabilité et la flexibilité au déploiement font partie des nombreux atouts dont dispose cet outil en plus de l'aspect communautaire très fort qui lui est associé. (Christopher Gerg, 2009)

Snort peut fonctionner selon 3 modes (Senders & Smith, 2014):

1- Mode Sniffer : lit simplement les paquets entrant vers ou sortant du réseau et les affiche dans un flux continu sur la console.
2- Mode Logger : cette option lit les paquets et les enregistre sur le disque pour les analyser ultérieurement.
3- Mode NIDS : il s'agit de sujet de ce projet. Dans ce mode, Snort effectue l'analyse du trafic réseau afin de détecter des intrusions selon des règles prédéfinies.

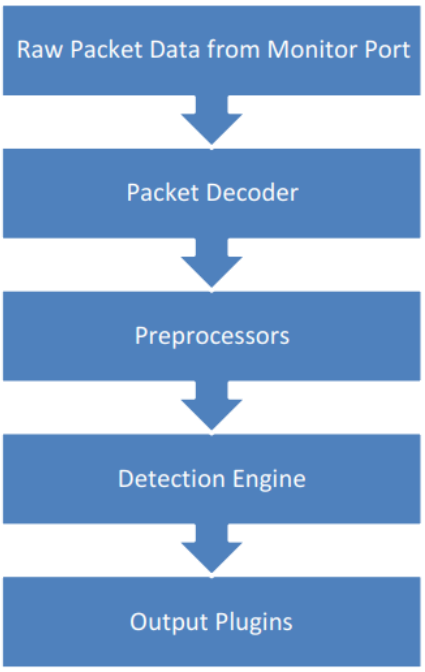
Dans un souci d'évolutivité, nous avons opté pour l'implémentation de la version 3.0 de Snort ayant subi plusieurs améliorations en comparaison avec les anciennes versions telles que :

- Des changements au niveau de la syntaxe utilisée pour l'écriture de règles.
- L'ajout de traitements multiples de paquets.
- De meilleures performances.

[7] Snort.org, *What is Snort?*
[8] Cisco Systems, *« Cisco Completes Acquisition of Sourcefire »,* 2013
[9] Doug Dineley, High Mobley, *"The greatest open source software of all time",* 2009

## 2. Architecture de Snort :

Selon le mode utilisé snort possède une architecture bien définie. Nous intéressons spécialement au mode NIDS.

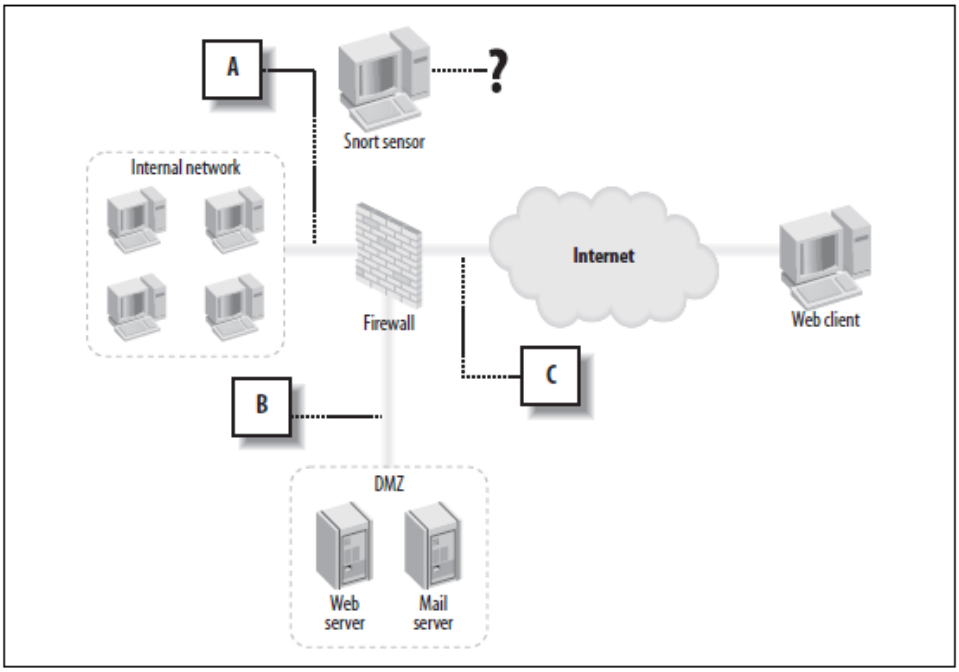



- **Packet Decoder :** Il s'agit d'une série de décodeurs qui analysent les paquets de données avant de les enregistrer dans un format approprié afin qu'ils soient traités par le préprocesseur et le Detection Engine (Senders & Smith, 2014).
- **Preprocessors :** Snort utilise deux types de préprocesseurs. Le premier est utilisé à des fins de détection. Le second lui sert à réordonner les données des paquets afin de faciliter le travail du Detection Engine (Senders & Smith, 2014)
- **Detection Engine :** Ou « moteur de détection » en français. C'est la partie chargée d'analyser les règles et de déterminer si les conditions qui y sont spécifiées correspondent au trafic analysé (Senders & Smith, 2014).
- **Output Plugins :** Quand le Detetion Engine détecte qu'une règle correspond au trafic analysé il envoie une alerte à l'administrateur (Senders & Smith, 2014)

## 3. Mise en place du NIDS sur le réseau hospitalier

## A. Mise en place de l'environnement

En sachant que le capteur (snort) ne détecte que les paquets réseau transitant par la machine sur laquelle il est actif, le placement de cette dernière a été l'un des enjeux majeurs du projet




- **Emplacement A (avant le pare-feu)** : Snort est capable de détecter toute communication entre le réseau interne et l'extérieur. Néanmoins, il y a un risque d'attaque de l'extérieur vers les serveurs de la DMZ. (Christopher Gerg, 2009)
- **Emplacement B (avant la DMZ) :** Dans ce cas on détecte les communications entre le réseau interne et la DMZ ainsi qu'entre la DMZ et Internet. Cependant, les communications entre le LAN et Internet ne sont pas détectées. (Christopher Gerg, 2009)
- **Emplacement C (après le pare-feu) :** Perte de la détection du trafic entre la DMZ et le réseau Interne. (Christopher Gerg, 2009)
- **Emplacement alternatif :** Placer plusieurs capteurs Snort. Bien que théoriquement cette solution est envisageable, dans la pratique le temps que nécessiterait le traitement des paquets à plusieurs niveaux pourrait ralentir le trafic réseau.

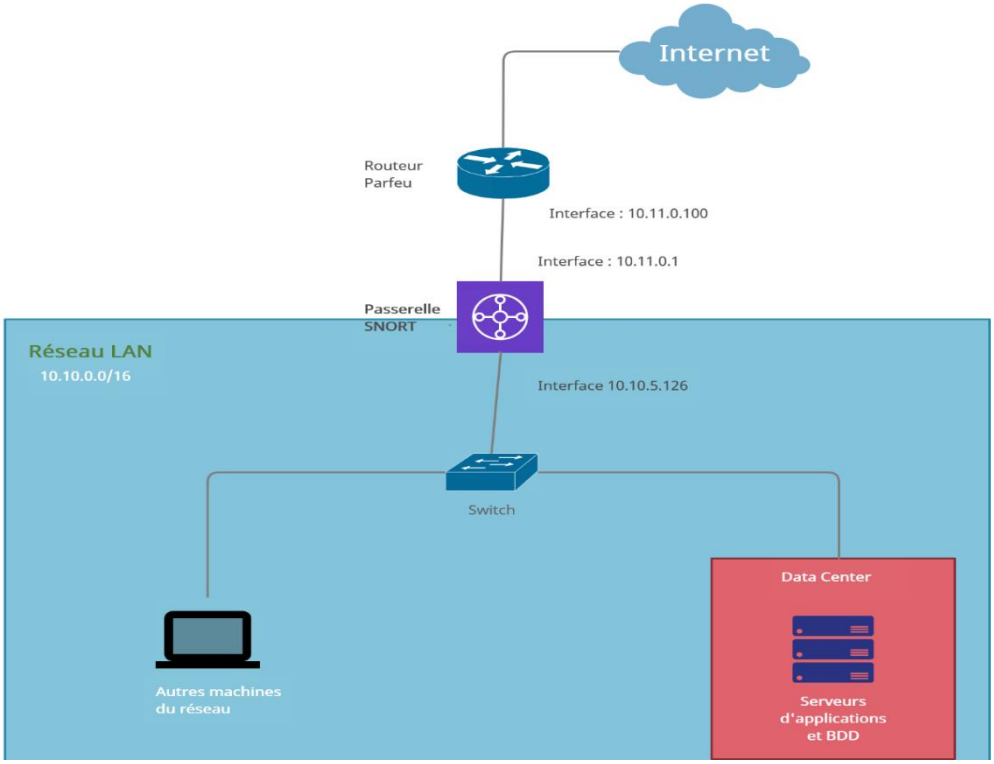
Dans notre cas le réseau ne dispose pas de DMZ ce qui réduit le choix aux emplacements A et C. Cependant, après réflexion l'emplacement A est le plus approprié et cela pour plusieurs raisons :

- Ne pas surcharger l'IDS en analysant seulement les menaces qui n'ont pas été détectées par le firewall.
- Détecter les attaques en interne et leurs source.

Une fois ce choix effectué, trois configurations s'offrent à nous pour placer la sonde :

- **Utiliser une TAP réseau** : cette solution consiste à utiliser un dispositif matériel (TAP) qui aura comme mission de copier le trafic passant de et vers internet sans que cela n'affecte les performances du réseau. Cette solution nécessite du matériel en plus ce qui induit des coûts supplémentaires.
- **Utiliser le port mirroring** : il s'agit de reproduire le fonctionnement de la TAP réseau grâce à un switch, cela en copiant le trafic de certains ports sur un port dit port miroir. Cette option peut affecter les performances (à cause des traitements supplémentaires pour le switch). De plus le port miroir pourrait être surchargé, ce qui engendrerait la perte de certains paquets
- **Utiliser Snort comme passerelle** : c'est la solution que l'on a privilégiée. Ce choix a été fait en se basant sur les points suivants :
  - Simple à mettre en œuvre.
  - Moins couteuse (pas de matériels en plus).
  - Pas de perte de paquets.
  - Évolutive (dans le cas où on veut utiliser Snort comme un IPS).

Snort en mode NIDS sera utilisé en complément au pare-feu de l'hôpital afin de permettre la détection d'éventuelles intrusions comme illustré dans la figure qui suit.




## Fonctionnement

La machine hôte de l'IDS tournant sous la version 8 de CentOS est représentée sur le schéma sous l'appellation « Passerelle SNORT ». Elle dispose de 2 interfaces réseau dont l'une est connectée au réseau LAN (10.10.5.126/16) et l'autre à l'interface d'entrée du routeur (10.11.0.1/16).

Les machines du réseau se voient attribuer la passerelle par défaut « 10.10.5.126 » par le serveur DHCP du Data Center. Cela veut dire que chaque paquet émanant du réseau LAN et étant à destination d'Internet (et réciproquement) transitera par la machine sur laquelle l'IDS sera configuré.

## Configuration de la Passerelle

Afin de permettre le passage des paquets du réseau interne vers le routeur, la machine doit être configurée en tant que passerelle (elle doit router les paquets).

- Dans un premier temps on modifie les fichiers de configuration des interfaces enp0s41 (sortie) et enp0s46 (entrée) afin de leur attribuer des adresses IP fixes :

```
# nano /etc/sysconfig/network-scripts/ifcfg-enp0s41
```

```
DEVICE=enp0s41
ONBOOT=yes
IPADDR=10.11.0.1
PREFIX=16
GATEWAY=10.11.0.100
```

```
# nano /etc/sysconfig/network-scripts/ifcfg-enp0s46
```

```
DEVICE=enp0s46
ONBOOT=yes
IPADDR=10.10.5.126
PREFIX=16
GATEWAY=10.11.0.1
```

- Afin de permettre l'envoi de paquets d'une interface à l'autre de la machine, on active le routage au niveau de la machine snort comme suit :

```
# sysctl net.ipv4.ip_forward
```

- On désactive ensuite le pare-feu de la machine pour éviter de bloquer les paquets :

```
# systemctl stop firewalld
```

- On ajoute les routes nécessaires nécessaire au niveau de routeur externe pour assurer le bon acheminement des paquets :

```
# route add 10.10.0.0/16 gw 10.11.0.1 enp0s41
```

- On redémarre les services réseau ainsi que les interfaces :

```
# service network restart
# ifdown enp0s46
# ifup enp0s46
# ifdown enp0s41
# ifup enp0s41
```

## B. Installation et configuration de l'IDS Snort

Une fois la machine correctement configurée, on passe à l'installation de Snort afin de permettre l'analyse du trafic. Pour cette installation nous nous sommes basés sur le document « Snort 3 on CentOS 8 »[10]

- Installation des dépendances et des librairies nécessaires :

[10] Yaser Mansour, *Snort 3 on CentOS 8*, 2020 – Voir : *snort.org/documents/snort-3-0-3-on-centos8*

```
# dnf config-manager --add-repo /etc/yum.repos.d/CentOS-PowerTools.repo
# dnf config-manager --set-enabled PowerTools
# dnf install epel-release
# dnf upgrade
# reboot now
# mkdir sources && cd sources
# nano /etc/ld.so.conf.d/local.conf
```

Ajout de 2 lignes : `'/usr/local/lib' et '/usr/local/lib64'`

```
# ldconfig
# dnf install git flex bison gcc gcc-c++ make cmake automake autoconf
libtool libpcap-devel pcre-devel libdnet-devel hwloc-devel openssl-devel
zlib-devel luajit-devel pkgconf libmnl-devel libunwind-devel
libnfnetlink-devel libnetfilter_queue-devel
# git clone https://github.com/snort3/libdaq.git
# cd libdaq
# ./bootstrap
# ./configure
# make && make install
# ldconfig
# cd ../
```

- Installation de Snort

```
# git clone https://github.com/snort3/snort3.git
# cd snort3
# export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH
# export PKG_CONFIG_PATH=/usr/local/lib64/pkgconfig:$PKG_CONFIG_PATH
# cd build/
# make -j$(nproc)&& make -j$(nproc) install
```

- Vérification de l'installation

```
# /usr/local/snort/bin/snort –V
```

Le résultat est le suivant :

```
   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.0.3 (Build 5)
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reser
ved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 3.0.0
           Using LuaJIT version 2.1.0-beta3
           Using OpenSSL 1.1.1c FIPS  28 May 2019
           Using libpcap version 1.9.0-PRE-GIT (with TPACKET_V3)
           Using PCRE version 8.42 2018-03-20
           Using ZLIB version 1.2.11
```

*Figure 4.3 Installation de Snort avec succès*

## Ajout de règles pour la détection

Snort est un IDS basé sur les signatures, cela signifie qu'il se base sur les signatures présentes dans les différents fichiers de règles pour détecter d'éventuelles intrusions. C'est pour cela que ces règles-ci doivent continuellement être tenues à jour afin de contrer les malwares récents. (Christopher Gerg, 2009).

## Structure d'une règle Snort

La structure d'une règle Snort est la suivante (Christopher Gerg, 2009)

```
Action Protocole IP_Source Port_Source -> IP_Dest Port_Dest (option_1 ;… ;
option_n ;)
```

**Action**

- alert : affiche un message à l'écran et ajoute le packet à l'historique.
- log : ajoute le paquet à l'historique.
- pass : ignore le paquet.

**Protocole**

- nom_protocole : IP, TCP, UDP, ICMP, ARP, GRE, RIP…etc.

**IP Source/Destination**

- any : tout réseau.
- $EXTERNAL_NET ou $HOME_NET : variables spécifiées dans le fichier de configuration.
- Liste d'IP : [64.147.128.0/19, 198.60.72.0/23,…]

**Port Source/Destination**

- any : tout port
- numéro_port : spécifier le numéro du port (ex. port 22)
- port_min : port_max : spécifier une plage de ports (ex. 20 :23)

**Quelques options**

- msg("Message à afficher") : spécifie le message à afficher.
- flags : spécifie les flags activés dans le paquet TCP (SYN, ACK…etc.).
- content : spécifie une chaîne de caractères à chercher dans le paquet.
- dsize : spécifie la taille limite acceptée pour un paquet.
- reference : référence vers un lien pour plus d'infos sur la menace.

**Ajout des règles préexistantes**

- Community rules : Ce sont des règles écrites et maintenues à jour par la communauté.

```
# tar -xvzf community-rules.tar.gz -C /etc/snort/rules
```

- Registred rules : Ce sont des règles destinées aux membres inscrits.

```
# wget https://snort.org/rules/snortrules-snapshot-
3000.tar.gz?oinkcode=<confidentiel> -O snortrules-snapshot-
3000.tar.gz
# tar -xvzf snortrules-snapshot-3000.tar.gz -C /etc/snort/rules
```

Les « registred rules » se déclinent selon des catégories, voici l'ensemble des fichiers disponibles :

```
snort3-browser-chrome.rules          snort3-policy-social.rules
snort3-browser-firefox.rules         snort3-policy-spam.rules
snort3-browser-ie.rules              snort3-protocol-dns.rules
snort3-browser-other.rules           snort3-protocol-finger.rules
snort3-browser-plugins.rules         snort3-protocol-ftp.rules
snort3-browser-webkit.rules          snort3-protocol-icmp.rules
snort3-content-replace.rules         snort3-protocol-imap.rules
snort3-deleted.rules                 snort3-protocol-nntp.rules
snort3-exploit-kit.rules             snort3-protocol-other.rules
snort3-file-executable.rules         snort3-protocol-pop.rules
snort3-file-flash.rules              snort3-protocol-rpc.rules
snort3-file-identify.rules           snort3-protocol-scada.rules
snort3-file-image.rules              snort3-protocol-services.rules
snort3-file-java.rules               snort3-protocol-snmp.rules
snort3-file-multimedia.rules         snort3-protocol-telnet.rules
snort3-file-office.rules             snort3-protocol-tftp.rules
snort3-file-other.rules              snort3-protocol-voip.rules
snort3-file-pdf.rules                snort3-pua-adware.rules
snort3-indicator-compromise.rules    snort3-pua-other.rules
snort3-indicator-obfuscation.rules   snort3-pua-p2p.rules
snort3-indicator-scan.rules          snort3-pua-toolbars.rules
snort3-indicator-shellcode.rules     snort3-server-apache.rules
snort3-malware-backdoor.rules        snort3-server-iis.rules
snort3-malware-cnc.rules             snort3-server-mail.rules
snort3-malware-other.rules           snort3-server-mssql.rules
snort3-malware-tools.rules           snort3-server-mysql.rules
snort3-native.rules                  snort3-server-oracle.rules
snort3-netbios.rules                 snort3-server-other.rules
snort3-os-linux.rules                snort3-server-samba.rules
snort3-os-mobile.rules               snort3-server-webapp.rules
snort3-os-other.rules                snort3-sql.rules
snort3-os-solaris.rules              snort3-x11.rules
snort3-os-windows.rules              VRT-License.txt
```

Afin de finir la configuration de Snort, nous modifions la variable $HOME_NET et le tableau rules au niveau du fichier de configuration « snort.lua » pour y spécifier les adresses du réseau ainsi que les chemins vers les règles que nous venons de télécharger.

```
-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = [[ 10.10.0.0/16 10.11.0.0/16 ]]
```

*Figure 22:Modification de la variable globale HOME_NET*

```
-- use include for rules files; be sure to set your path
-- note that rules files can include other rules files
rules = [[
    include ./snort3-community.rules
    include ./rules/snort3-malware-backdoor.rules
    include ./rules/snort3-malware-cnc.rules
    include ./rules/snort3-malware-other.rules
    include ./rules/snort3-malware-tools.rules
    include ./rules/snort3-exploit-kit.rules
    include ./rules/snort3-file-pdf.rules
    include ./rules/snort3-indicator-scan.rules

]],
```

*Figure 23: Ajout des chemins vers les fichiers de règles*

```
--alert_csv = { }
alert_fast = {  file = true  }
--alert_full = { }
```

*Figure 24: Activation de  alert_fast pour journaliser les alertes*

### Règle spécifique à « cloudnet.exe »

Le but de l'analyse de malware effectuée précédemment était de permettre sa compréhension et de définir une règle Snort permettant de détecter son activité. Néanmoins, nous avons constaté que celle-ci existait déjà au niveau du fichier « snort3-malware-cnc.rules» sous la forme :

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 1024:(
msg:"MALWARE-CNC Win.Trojan.Glupteba.M"; flow:to_server, established;
content:"/stat?"; content:"uptime="; content:"&downlink=", distance 0; content:"&uplink=", distance 0;
content:"&id=",distance 0; content:"&statpass=bpass",distance 0,fast_pattern; content:"&version=", distance
0; content:"&features=", distance 0; content:"&guid=", distance 0; content:"&comment=", distance 0;
content!"&p=",distance 0; content:"&s=",distance 0;
metadata:ruleset community;
service:http ;
reference:url, www.welivesecurity.com/wp-content/uploads/2014/03/operation_windigo.pdf;
classtype:trojan-activity;
sid: 30288; rev: 3;)
```

En analysant cette règle, on remarque qu'elle opère sur les ports supérieurs à 1024 et que le champ "content" spécifie exactement les éléments de la requête http que nous avons identifiée lors de l'analyse du malware.

## C. Tests

Afin de nous assurer que Snort fonctionne correctement, nous allons le lancer et tester quelques règles.

Dans ce qui suit nous nous limiterons à la documentation de deux tests. Le premier consiste à effectuer une attaque SSH par brute force et le second à lancer l'exécution du malware étudié au cours du chapitre 2. Le but étant de voir si Snort détecte chacune des deux activités.

Pour cela, nous activons les règles spécifiques à la détection de ces deux types d'intrusions depuis les fichiers « snort3-scan-indicator.rules » et « snort3-malware-cnc.rules » puis on lance l'exécution de Snort avec la commande :

# snort –c /usr/local/snort/etc/snort/snort.lua –i enp0s46 :enp0s41 –A alert_fast –l /var/log/snort &

## Attaque par dictionnaire en SSH avec Kali Linux

Dans un premier temps, nous utilisons l'outil metasploit disponible sur le système Kali Linux

```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   BLANK_PASSWORDS  false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED 5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS     false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS      false            no        Add all passwords in the current database to the list
   DB_ALL_USERS     false            no        Add all users in the current database to the list
   PASSWORD                          no        A specific password to authenticate with
   PASS_FILE                         no        File containing passwords, one per line
   RHOSTS                            yes       The target address range or CIDR identifier
   RPORT            22               yes       The target port
   STOP_ON_SUCCESS  false            yes       Stop guessing when a credential works for a host
   THREADS          1                yes       The number of concurrent threads
   USERNAME                          no        A specific username to authenticate as
   USERPASS_FILE                     no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS     false            no        Try the username as the password for all users
   USER_FILE                         no        File containing usernames, one per line
   VERBOSE          false            yes       Whether to print output for all attempts

msf auxiliary(scanner/ssh/ssh_login) > set rhosts 10.10.5.126
rhosts => 10.10.5.126
msf auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
USERPASS_FILE => /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
msf auxiliary(scanner/ssh/ssh_login) > set verbose true
verbose => true
msf auxiliary(scanner/ssh/ssh_login) > run

[-] 10.10.5.126:22 - Failed: 'root:'
[!] No active DB -- Credential data will not be saved!
[-] 10.10.5.126:22 - Failed: 'root:!root'
[-] 10.10.5.126:22 - Failed: 'root:Cisco'
[-] 10.10.5.126:22 - Failed: 'root:NeXT'
[-] 10.10.5.126:22 - Failed: 'root:ONX'
```

Une fois l'attaque lancée, voici ce que Snort affiche :

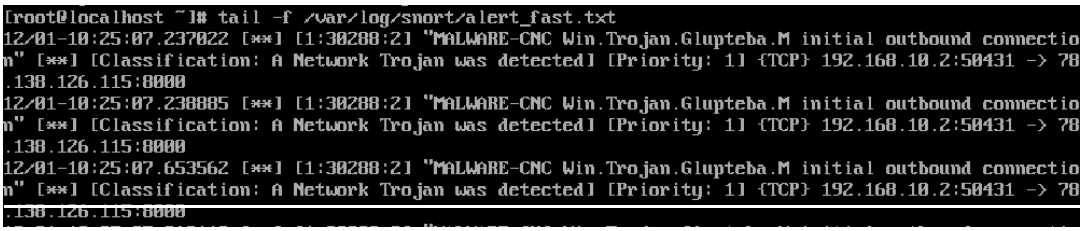
```
[root@localhost ~]# tail -f /var/log/snort/alert_fast.txt

12/03-04:54:26.140064 [**] [1:19559:13] "INDICATOR-SCAN SSH brute force login at
tempt" [**] [Classification: Misc activity] [Priority: 3] {TCP} 10.10.3.114:4108
5 -> 10.10.5.126:22
12/03-04:54:26.140063 [**] [1:19559:13] "INDICATOR-SCAN SSH brute force login at
tempt" [**] [Classification: Misc activity] [Priority: 3] {TCP} 10.10.3.114:4108
5 -> 10.10.5.126:22
12/03-04:54:30.524376 [**] [1:19559:13] "INDICATOR-SCAN SSH brute force login at
tempt" [**] [Classification: Misc activity] [Priority: 3] {TCP} 10.10.3.114:3948
9 -> 10.10.5.126:22
12/03-04:54:30.524375 [**] [1:19559:13] "INDICATOR-SCAN SSH brute force login at
tempt" [**] [Classification: Misc activity] [Priority: 3] {TCP} 10.10.3.114:3948
9 -> 10.10.5.126:22
```

*Figure 26: Résultats attaque par bruteforce*

## Détection de l’activité du malware :

Premièrement, on lance notre malware sur la machine virtuelle déjà mise en place puis on connecte cette machine sur le réseau.
Néanmoins, pour des soucis de sécurité on lance ce malware dans un réseau autre que celui de l’hôpital. On aura les résultats suivants :




On déduit que la règle écrite fonctionne parfaitement puisqu’au bout de 6 tentatives erronées des messages s’affichent indiquant que quelqu’un essaie de forcer l’accès via SSH.

# Conclusion

L'évolution rapide que connait l'informatique tant sur le plan physique que logiciel au fil des années permet un développement considérable des services auxquels elle donne accès afin de toujours permettre de meilleures expériences pour l'utilisateur. Toutefois, ce progrès s'accompagne inéluctablement d'un lot de failles et de brèches informatiques qu'il faut toujours s'efforcer de colmater.

Dans des milieux aussi sensibles que les hôpitaux, la sécurité informatique est un pilier car tout écart peut donner lieu à des dégâts irréversibles. Cela dit, durant les quatre dernières semaines nous avons pu prendre pleinement conscience de l'importance qu'il faut accorder à l'instauration d'une politique de sécurité solide au sein d'une organisation afin d'assurer le bon fonctionnement de son système d'information.

Nous avons notamment pu, au terme de ce stage contribuer au renforcement de la sécurité du réseau hospitalier. Et bien que le système mis en place ne soit pas parfait, il offre des perspectives d'évolution intéressantes sur plusieurs axes :

- Dans un premier temps, il serait envisageable d'intégrer la détection au niveau du réseau local. Dans la version actuelle de l'installation, les communications entre les machines du réseau ainsi qu'avec le data center ne sont pas scannées par Snort.

  Un moyen de concrétiser cela serait d'effectuer une opération de « Mirroring » en utilisant le Switch. Cela reviendrait à copier tout paquet passant par ce dernier et envoyer cette copie vers l'IDS. Cette opération serait complémentaire au système actuel, toutefois il est nécessaire d'étudier le risque de surcharge du commutateur.

- Ensuite, nous pouvons citer les règles de détection qui peuvent être aussi bien une force qu'une faiblesse.

  D'un côté, elles permettent d'être exhaustif et flexible en termes de menaces à contrer mais de l'autre elles nécessitent une forte implication pour toujours être à jour. Elles pourraient se montrer obsolètes dans le cas où un malware nouveau ou non répertorié venait à apparaître dans le réseau, et c'est donc pour cela qu'il serait pertinent d'envisager l'ajout futur d'une fonctionnalité de détection automatique grâce, par exemple, à des outils d'intelligence artificielle.

## Bibliographie


Ali, A., Lu, W., & Tavalaee, M. (2010). *Network Intrusion Detection and Prevention: Concepts and Techniques.* Springer.

AMADEUS. (2016). La sécurité des systèmes d'information. *Séminaire de sensibilisation*, (pp. 20-22).

Anane, M. (2020). Introduction à la Sécurité Informatique (1CS - ESI). Alger.

Bloch, L., & Wolfhugel, C. (2009). *Sécurité informatique: Principes et méthodes à l'usage des DSI, RSSI et administrateurs.* EYROLLES.

Christopher Gerg, K. J. (2009). *Managing Security with Snort & IDS Tools.* O'Reilly Media, Inc.

Cisco. (s.d.). Récupéré sur snort.org: https://www.snort.org/

Forescout Research Labs. (2020). *Connected Medical Device Security : A Deep Dive into Healthcare Networks.*

Hiet, G. (2008). *Détection d'intrusions paramétrée par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications : mise en œuvre sous Linux pour les programmes Java.*

Horejsi, J., & C Chen,, J. (2019). *Glupteba Hits Routers and Updates C&C Servers, Avril 2019.*

Khraisat , A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). *Survey of intrusion detection systems: techniques, datasets and challenges.* Récupéré sur https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7

Koziol, J. (2003). *Intrusion Detection with Snort.*

Malwarebytes. (s.d.). *malware*. Récupéré sur fr.malwarebytes.com/: http://fr.malwarebytes.com/malware

Mouna Jouinia, L. B. (2014). Classification of security threats in information systems. *ScienceDirect*.

Nagy, L. (2020). *Glupteba: Hidden Malware Delivery in Plain Sight.*

PurpleSec. (2020). *Cyber Security Statistics.* Washington DC. Récupéré sur https://purplesec.us/resources/cyber-security-statistics/

securiteinfo. (s.d.). *techniques-detection-malware.shtml*. Récupéré sur www.securiteinfo.com: https://www.securiteinfo.com/attaques/malwares-virus-spam-logiciels-indesirables/techniques-detection-malware.shtml

Senders, C., & Smith, J. (2014). *Applied Network Security Monitoring.* Elsevier.

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to dissecting Malicious Software.*

Sitesh Sinha, P. S. (2017). General Study of Intrusion Detection System and Survey of Agent Based Intrusion Detection System. *International Conference on Computing, Communication and Automation (ICCCA2017).*

Stalling, W. (2017). *Cryptography and Network Security : Principles and Practice* (éd. 7e). Pearson.

Ur Rehman, R. (2003). *Intrusion Detection Systems with Snort.* Pearson Educational, Inc.

VirusTotal. (s.d.). *How-it-works*. Récupéré sur virustotal.com: https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works