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

BP 526 Cotonou Tel : +229 21 14 19 88
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 :
Axel Kévin Kadoukpè **BANKOLE**

Mise en place d'un serveur sécurisé pour l'hébergement d'une application web : Cas de l'entreprise NOVEDIX

Sous la supervision de :
Mr Mawuenan Hervé Guy **AKAKPO**
Directeur de l'Informatique à la Caisse Autonome d'Amortissement

Membres du jury :

AHOUANDJINOU Arnaud	Docteur	IFRI	Président
BIBI Honorat	Ingénieur	IFRI	Examinateur
AKAKPO Mawuenan Hervé	Ingénieur	IFRI	Rapporteur

Année Académique : **2020-2021**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Table des figures vi

Liste des tableaux vii

Liste des acronymes 1

Introduction 2

1 Revue de littérature 4

2 Analyse et choix techniques 10

3 Mise en place, tests et résultats 16

Conclusion 28

Webographie 29

Table des matières 30

Dédicace

A
Mon père Edgard BANKOLE
Ma mère Jacqueline Da CRUZ
Mes soeurs

Remerciements

Mes remerciements s'adressent à toutes les personnes qui de près ou de loin ont contribué à la réalisation de ce mémoire .

Je tiens à remercier particulièrement :

- Le Professeur Eugène C. EZIN, Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI) ;
- Mr Mawuenan Hervé Guy AKAKPO, mon Maitre de mémoire pour sa disponibilité et son soutien ;
- Tous les enseignants de notre institut pour avoir accepté de partager avec dévouement une partie de leurs connaissances avec nous ;
- Ma famille pour tout le soutien qu'elle m'a apportée durant ces trois années de formation ;
- Mes proches et amis

Résumé

Les serveurs web sont massivement ciblés par des attaques de pirates informatiques, qui convoitent les précieuses données stockées sur ces serveurs afin d'en faire un usage commercial ou frauduleux. Dans le souci de produire une solution d'hébergement web sécurisé, nous optons pour plusieurs techniques et méthodes de sécurisation des serveurs web et de détection d'intrusion. Notre travail consiste tout d'abord à mettre en place un serveur web principal prenant en charge le stockage des applications web puis un autre servant de proxy inverse. Nous intégrons ensuite un certificat SSL à chaque serveur pour sécuriser les communications. Enfin, nous mettons en place un système de détection d'intrusion couplé à une interface graphique et un système d'alerte par SMS. Après les différents tests effectués, nous avons d'abord pu constater l'effectivité du chiffrement des communications client-serveur. Ensuite, nous remarquons l'efficacité de notre système de détection d'intrusion qui, après identification d'un comportement suspect, notifie systématiquement l'administrateur système via le système d'alerte par SMS mis en place.

Mots clés : Serveur, détection d'intrusion, proxy inverse, certificat, SSL.

Abstract

Web servers are massively targeted by attacks from hackers, who covet the valuable data stored on these servers in order to make commercial or fraudulent use of it. In order to produce a secure web hosting solution, we opt for several techniques and methods for securing web servers and intrusion detection. Our work consists first of all in setting up a main web server supporting the storage of web applications and then another serving as a reverse proxy. We then integrate an SSL certificate into each server to secure communications. Finally, we are setting up an intrusion detection system coupled with a graphical interface and an SMS alert system. After the various tests carried out, we were first able to see the effectiveness of the encryption of client-server communications. Then, we note the effectiveness of our intrusion detection system which, after identification of suspicious behavior, systematically notifies the system administrator via the SMS alert system set up.

Key words: Servers, intrusion detection, reverse proxy, certificate, SSL

Table des figures

2.1 Architecture du réseau 13

3.1 Test de configuration 17
3.2 Configuration 18
3.3 Installation de snort réussie 18
3.4 Validation de configuration de snort 19
3.5 Validation de configuration de snort 19
3.6 Installation de Barnyard2 réussie 20
3.7 Configuration de PulledPork 20
3.8 Interface Web BASE 21
3.9 Configuration réussie 22
3.10 Avant SSL 23
3.11 Après SSL 23
3.12 Identifiants interceptés sans SSL 24
3.13 Identifiants interceptés avec SSL 24
3.14 Lancement de l'attaque 25
3.15 Lancement de l'attaque 25
3.16 Message d'alerte 26
3.17 Détails sur la menace 26

Liste des tableaux

2.1 Comparatif entre Apache et NGINX 11
2.2 Comparatif des IDS : Snort, Suricata, Zeek 12

Liste des acronymes

BASE : Basic Analysis and Security Engine

DMZ : Demilitarized Zone
DNS : Domain Name System
DOS : Denial Of Service

HTML : HyperText Markup Language
HTTP : HyperText Transfer Protocol
HTTPS : HyperText Transfer Protocol Secure

IDS : Intrusion Detection System
IP : Internet Protocol
IPS : Intrusion Prevention System

MMS : Multimedia Messaging Service

NIDS : Network Intrusion Detection System

SMS : Short Message Service
SQL : Structured Query Language
SSL : Secure Socket Layer

TCP : Transmission Control Protocol
TLS : Transport Layer Security

VPN : Virtual Private Network

Introduction Générale

Les plateformes web proposant des services au grand public sont monnaie courante et accueillent pour les plus utilisées une grande quantité d'informations sensibles. Pour cela, elles subissent de nombreuses attaques de pirates qui convoitent les données ou simplement veulent empêcher l'accès aux ressources par des utilisateurs légitimes. Plusieurs systèmes sont mis en place pour assurer la sécurité, mais celle-ci n'est pas toujours garantie.

Contexte et justification

Dans les entreprises, la sécurité des réseaux est améliorée en continu. Plusieurs technologies (DMZ, pare-feu, proxy...) sont utilisées de nos jours pour sécuriser les accès depuis Internet. Cependant, bon nombre d'entreprises subissent quotidiennement des vols ou l'altération des données échangées dans le réseau et tout cela sans en avoir conscience. Il est donc possible pour un individu mal intentionné de forcer les accès aux serveurs ou encore d'incruster des logiciels malveillants et créer des portes dérobées sans que personne ne soit au courant du fait de l'absence d'un suivi des attaques subies. Cela constitue une menace permanente aussi bien pour l'infrastructure que pour les finances de l'entreprise. Quand nous prenons l'exemple d'une entreprise dont une grande partie des revenus dépend du bon fonctionnement de ses applications web, nous pouvons clairement voir que l'enjeu est de taille. Il s'agit donc pour nous de proposer une solution permettant aux entreprises de pallier à ces menaces, et ainsi, protéger leur système d'information. C'est dans cette perspective que s'inscrit cette étude intitulée : "Mise en place d'un serveur sécurisé pour l'hébergement d'une application web : cas de l'entreprise NOVEDIX".

Notre travail consiste donc à mettre en place pour le compte de NOVEDIX une solution d'hébergement sécurisée.

Problématique

Les données représentent un pilier essentiel pour toute entreprise, constituant une source précieuse de ressources pour les utilisateurs, tout en étant des cibles privilégiées pour les cyberattaques. Cette problématique est encore plus critique dans le contexte des ressources à accès public, telles que les sites et applications web. Nous observons néanmoins un déficit de mesures de sécurité efficaces, notamment en matière d'hébergement, où la disponibilité et la confidentialité des données doivent

être garanties. De plus, l'absence de solutions permettant au personnel de sécurité de détecter rapidement les intrusions constitue une faiblesse critique dans la protection des systèmes. Dès lors, il devient crucial de se pencher sur des solutions permettant d'assurer une défense efficace contre les menaces, ainsi que la détection de celles-ci.

Objectifs

L'objectif principal de notre travail est de mettre en place et de sécuriser un serveur web. De manière spécifique, il s'agit entre autres de :

- Découvrir les attaques qui peuvent viser un serveur web et comprendre leur fonctionnement
- Présenter les solutions proposées face à ces attaques tout en justifiant leur utilisation
- Analyser et ressortir les meilleures offres de solutions sur le marché
- Mettre en place le serveur principal et un proxy inverse en intégrant à chacun d'eux un certificat SSL
- Mettre en place une solution de détection d'intrusion couplé à un système d'alerte par SMS.
- Tester la performance de notre solution.

Organisation du mémoire

Notre travail est structuré en trois chapitres. Le premier met en lumière des concepts clés à travers une revue de littérature. Dans le second, nous faisons des choix de solutions parmi les propositions après analyse. Le dernier chapitre présente la mise en place des solutions, les tests effectués ainsi que les différentes perspectives d'amélioration de notre travail.

Chapitre 1

Revue de littérature

Introduction

Dans ce chapitre, nous survolons rapidement le monde de la sécurité informatique puis nous plongeons au cœur des serveurs et des attaques informatiques qui y sont liées . Nous présentons ensuite le serveur proxy inverse, le certificat SSL et les systèmes de détection d'intrusion comme solutions en justifiant nos choix.

1.1 Généralités sur la sécurité informatique

1.1.1 Définition

La sécurité informatique est l'ensemble des moyens mis en œuvre pour réduire la vulnérabilité d'un système face aux menaces physiques comme logiques qui pèsent sur celui-ci.

1.1.2 Objectifs de la sécurité informatique

La sécurité informatique, d'une manière générale, vise à s'assurer que les ressources soient utilisées par des personnes légitimes, aux heures prescrites et dans le cadre prévu. Ainsi, les objectifs principaux de la sécurité informatique se déclinent comme suit :

- L'intégrité : garantir que les données sont bien celles que l'on croit être ;
- La disponibilité : maintenir le bon fonctionnement du système d'information ;
- La confidentialité : rendre l'information inintelligible aux personnes autres que les acteurs d'une transaction ;
- La non répudiation : garantir qu'une transaction ne peut être niée ;
- L 'authentification : assurer que seules les personnes autorisées aient accès aux ressources.

1.2 Les serveurs

1.2.1 Définition et Fonctionnement d'un serveur

Le terme de « serveur » possède deux significations en informatique. On désigne par serveur aussi bien une machine physique qu'un logiciel hébergeant un service et le mettant à la disposition d'utilisateurs.

Pour faire office de serveur, une machine ou un logiciel doit être configuré pour écouter et traiter les requêtes qui lui sont transmises via une connexion réseau. Quand un client sollicite des données ou une fonctionnalité auprès d'un serveur, il envoie une requête sur le réseau. Le serveur la reçoit et y répond en communiquant l'information qui lui a été demandée. C'est le mode requête-réponse du modèle client-serveur. Un serveur peut aussi effectuer d'autres tâches dans le cadre d'une requête-réponse à savoir : vérifier l'identité de l'auteur de la requête, s'assurer que le client autorise l'accès aux données ou ressources qu'il sollicite, et formater correctement ou renvoyer la réponse requise selon la méthode qui est attendue.

1.2.2 Rôle d'un serveur web

Un serveur web doit stocker les fichiers nécessaires au fonctionnement du service web qu'il fournit : tous les documents HTML et les ressources liées, dont les images, les fichiers JavaScript, les feuilles de styles, les fichiers de polices, les vidéos, etc.

1.3 Vue globale de différentes attaques sur les serveurs web

[2] En entreprise, mettre en place un serveur sécurisé est une tâche qui nécessite de prendre en compte les menaces physiques, les menaces logicielles, les menaces internes, les menaces externes, les risques liés à son utilisation etc. Pour notre étude, nous avons décidé de nous concentrer sur les menaces liées à des pirates informatiques positionnés en externe. Ici encore, nous allons réduire le champ de recherche aux attaques de pirates ciblant directement le serveur mis en place, mettant donc de côté celles liées à l'exploitation de failles au niveau des applications hébergées.

- Attaque de déni de service

  On parle de déni de service lorsqu'un attaquant peut surcharger de requêtes le serveur Web, entraînant ainsi un plantage et une indisponibilité pour les utilisateurs légitimes. Nous avons par exemple : attaque par débordement de tampon, inondation SYN, HTTP get Request Flooding, ping de la mort. Cette attaque exploite l'absence de mesures de traitement de gros flux de requêtes.

- Phishing

  Avec ce type d'attaque, l'attaquant usurpe l'identité des sites Web pour appâter ses victimes. Les utilisateurs ainsi mis en confiance sont amenés à soumettre des données sensibles telles que les informations de connexion, les numéros de carte de crédit, etc. Cette attaque exploite le manque d'informations des clients et l'absence d'un référentiel de sécurité pouvant les guider sur la fiabilité de certains services web.

- Attaque de l'homme du milieu

  On parle d'attaque de l'homme du milieu lorsqu'un pirate informatique se positionne entre un client et un serveur en interceptant les informations échangées et les manipulant à sa guise. La réussite de cette attaque ne réside pas dans le fait que les données échangées soient interceptées mais plutôt dans le fait que celles-ci soient compréhensibles et modifiables par l'attaquant. Cette attaque exploite simplement l'absence de mesures assurant la confidentialité des données échangées.

- Collecte d'informations

  Il ne s'agit pas d'une attaque à proprement parler mais plutôt d'une phase précédant une attaque. La dangerosité de la collecte réside dans les informations qu'elle peut fournir de manière légale :

  – adresse IP du serveur
  – nom du serveur utilisé et sa version
  – emplacement géographique du serveur
  – protocoles de sécurité utilisés et leurs versions
  – nom de l'entreprise en charge du serveur
  – etc.

  Cette attaque exploite l'absence de mesures de camouflage des données relatives aux serveurs sur internet.

- Empoisonnement du cache

  Un cache Web est une technologie d'information permettant de stocker temporairement des documents Web tels que des pages Web, des mots de passe et des images. L'empoisonnement du cache Web est une technique dans laquelle l'attaquant envoie de fausses demandes d'entrée au serveur, efface tous les caches réels du serveur et redirige l'utilisateur vers le site Web malveillant. Cette attaque exploite généralement un défaut de configuration au niveau des serveurs web.

- Attaques de mots de passe

  Dans cette attaque, l'attaquant utilise de nombreux outils de craquage de mots de passe afin de déchiffrer les mots de passe utilisés sur le serveur. Les outils courants sont Hydra, John the Ripper, Hashcat, Aircrack, etc. Cette attaque exploite le plus souvent la simplicité des mots de passe et l'absence d'une politique de gestion des échecs d'authentification en continu.

  Chacune de ces attaques exploite une ou plusieurs vulnérabilités du serveur web et met en péril les objectifs de sécurité énumérés plus haut.

1.4 Répercussions d'attaques réussies

– La réputation d'une organisation peut être ruinée si l'attaquant modifie le contenu du site Web et inclut des informations malveillantes ou des liens vers du contenu illégal.

  - Le serveur Web peut être utilisé pour installer des logiciels malveillants sur les terminaux des utilisateurs qui visitent le site Web compromis. Le logiciel malveillant téléchargé sur l'ordinateur du visiteur peut être un virus, un ransomware, un cheval de Troie, un botnet, etc.
  - Les données utilisateurs étant compromises, elles peuvent être utilisées pour des activités frauduleuses susceptibles d'entraîner des pertes commerciales ou des poursuites judiciaires de la part des utilisateurs qui ont confié leurs coordonnées à l'organisation.

Pour notre étude, nous nous sommes particulièrement intéressés à mettre en place des éléments de sécurité permettant de pallier aux vulnérabilités liées aux attaques suivantes :

- Déni de Service
- Homme du milieu
- Phishing
- Collecte d'informations.

1.5 Présentation des solutions et justification des choix

Notre choix se porte sur deux solutions à savoir le Proxy inverse et le certificat SSL.

1.5.1 Proxy inverse

1.5.1.1 Présentation

Un proxy inverse désigne un serveur placé devant un ou plusieurs serveurs web et qui intercepte les requêtes provenant des clients. Ce dernier envoie alors des requêtes au serveur d'origine et reçoit des réponses de ce dernier.

1.5.1.2 justification du choix

- Le proxy inverse protège un serveur Web des attaques provenant de l'extérieur. En effet, avec un proxy inverse en place, les informations relatives au serveur d'origine sont inaccessibles puisque le proxy inverse se comporte comme le serveur d'origine. La **collecte d'informations** sur le serveur d'origine est ainsi déjouée.
- Aussi, il est donc beaucoup plus difficile pour les pirates de lancer une attaque ciblée contre le serveur d'origine, comme une attaque de **déni de service**. Les pirates ne peuvent alors viser que le proxy inverse.
- La fonction de cache du proxy inverse, en plus d'améliorer les performances en matière de traitement des requêtes des utilisateurs, contribue aussi à la protection du serveur d'origine. En effet, en cas d'attaque de **Déni de Service**, de nombreuses requêtes à une seule ressource n'entraînent qu'une seule requête au serveur d'origine. Le serveur d'origine ne subit donc pas l'impact de l'attaque.

1.5.2 Certificat SSL

[3]

1.5.2.1 Présentation

Essentiellement, SSL sécurise les informations qui sont échangées entre un serveur web et un client.

- Définition :

  Un certificat est un document électronique qui fait correspondre une clé publique avec une entité (entreprise ou organisation), il est validé par une autre autorité de certification. Il contient un ensemble de données comme le nom du certificat, son utilisation, des informations identifiant le propriétaire, la clé publique, la date d'expiration et le nom de l'organisme certificateur. Le passage par une autorité de certification générant le certificat est obligatoire, incontournable pour mise place du système sécurisé d'un commerce électronique.

- Type de certificat suivant la signature :

  Il existe deux signatures de certificats : le certificat signé par le fournisseur de certificat et le certificat autographe ou communément appelé « sef-signed certificate » en anglais.

- Différence entre les deux signatures de certificat :

  La principale différence entre ces deux types de signatures de certificats est que le certificat signé par un fournisseur est reconnu automatiquement par les navigateurs et permet de garantir l'identité de l'organisation qui fournit les pages web au navigateur ce qui n'est pas le cas du certificat autographe. Celui-ci est généralement utilisé pour faire des tests en interne en vue d'une solution commerciale.

Le déploiement d'un certificat SSL présente un certain nombre d'avantages.

1.5.2.2 Justification du choix

- L'un des principaux avantages de SSL/TLS est le cryptage. Chaque fois que des utilisateurs entrent des informations sur un site, ces données passent par plusieurs points de contact avant d'atteindre leur destination finale. Sans SSL, ces données sont envoyées en texte clair et des acteurs malveillants peuvent écouter ou modifier ces données. SSL/TLS offre une protection point à point pour assurer la sécurité des données pendant le transport. Les attaques de **l'homme du milieu** ont ainsi moins de chance d'aboutir.

- Un autre avantage clé est l'authentification. Une connexion SSL/TLS en bon état de fonctionnement garantit que les données sont envoyées et reçues par le bon serveur, plutôt que par un individu malveillant. Tout cela est possible grâce à l'unicité des certificats SSL. En d'autres termes, il aide à empêcher les acteurs malveillants de se faire passer pour un site à tort puisque le certificat sera différent. C'est encore un autre point de sécurité contre les attaques de **Phishing et homme du milieu**

1.6 Les systèmes de détection d'intrusion

La sécurité de notre serveur Proxy inverse n'étant pas garantie puisqu'il sert de protection à notre serveur d'origine, il serait judicieux de disposer d'un système pouvant alerter le personnel de sécurité dans le cas où il serait visé par une attaque de pirate. Cette pratique peut contribuer à limiter les dégâts mais aussi à récolter de précieuses informations sur le mode opératoire des pirates. Ces informations servent à améliorer la sécurité de nos installations. C'est dans cette optique que nous avons décidé de mettre en place un système de détection d'intrusion.

1.6.1 Définition

Un système de détection d'intrusion est un dispositif matériel et ou logiciel utilisé pour surveiller et protéger les réseaux et systèmes informatiques contre de potentielles activités malveillantes. Il est essentiellement chargé de passer au peigne fin toutes les activités du réseau ou du système surveillé puis alerter le personnel de sécurité en cas de détection d'une intrusion.

1.6.2 Fonctions principales d'un IDS

Les principales fonctionnalités d'un IDS peuvent se résumer comme suit :

- Recueillir et analyser les informations à l'intérieur d'un ordinateur ou d'un réseau, afin de détecter les potentielles violations de la politique de sécurité, des accès non autorisés etc
- Se comporter en sniffeur de paquets pour intercepter les paquets qui circulent à travers les médias et protocoles de communication.
- Analyser les paquets une fois qu'ils ont été capturés et produire une alarme après identification d'une activité suspecte.

Dans ce chapitre, nous avons d'abord présenté les serveurs web ainsi que les différents types d'attaques visant les serveurs web. Nous avons ensuite présenté le Proxy inverse, le certificat SSL et les IDS comme solutions face à ces attaques en argumentant nos choix . Dans le chapitre suivant, nous présentons nos choix concernant lesdites solutions.

Chapitre 2

Analyse et choix techniques

Introduction

Dans ce chapitre, nous vous proposons de découvrir les meilleures offres en matière de serveur web et de système de détection d'intrusion tout en faisant une étude comparative dans chaque cas afin d'en ressortir les meilleures solutions en matière de sécurité mais aussi de performances.

2.1 Mesures de sécurité côté serveurs web

2.1.1 Choix des serveurs à utiliser

Il existe une multitude de serveurs web, mais seuls deux d'entre eux requièrent notre attention puisqu'ils sont de loin les plus utilisés sur le marché.

2.1.1.1 Apache

Serveur HTTP Apache, familièrement connu comme Apache ou httpd sur les distributions Redhat est un serveur Web gratuit et développé par la fondation Apache Software. Publié en 1995, Apache a grandi à pas de géant pour devenir l'un des serveurs Web les plus populaires et les plus utilisés : 37% de tous les sites Web. Apache est écrit en C et est un serveur Web hautement personnalisable grâce à ses tonnes de modules qui étendent les fonctionnalités du serveur Web.

2.1.1.2 Nginx

Nginx est apparu en 2004 quand il a été publié pour la première fois par le développeur russe Igor Sysoev et a gagné en popularité pour devancer ses concurrents et devenir l'un des serveurs Web les plus stables et les plus fiables. Comme l'a dit Owen Garrett, le chef de projet de Nginx :« Nginx a été écrit spécifiquement pour répondre aux limitations de performance des serveurs web Apache. » Après sa sortie, Nginx a été utilisé principalement pour servir des fichiers statiques et servir d'équilibreur de charges ou de proxy inverse devant les installations d'Apache.

2.1.1.3 Analyse comparative des deux serveurs

Le tableau suivant présente une brève comparaison entre Apache et Nginx :

TABLE 2.1 – Comparatif entre Apache et NGINX

Critères	Apache	NGINX
Performances	Faibles	Excellentes
Gestion des connexions	Moyenne	Excellente
Utilisation des ressources	Plus élevée	Moins élevée
Facilité de configuration	Très bonne	Moyenne
Flexibilité	Excellente	Bonne
Modules et extensions	Très nombreux	Moins nombreux
Communauté et support	Très bonne	Bonne

Au vu de ces résultats, nous avons décidé de faire ce qui se fait le mieux actuellement, c'est-à-dire utiliser d'abord Apache comme serveur principal au vu de sa flexibilité et de ses nombreux modules disponibles, puis Nginx comme serveur proxy inverse au vu de ses meilleures performances en gestion de contenu et de flux de requêtes [1].

2.1.2 Sécurité des communications sur le serveur web

Dans l'objectif de sécuriser les échanges client-serveur, nous utilisons des certificats fournis par Open SSL au niveau de chaque serveur à mettre en place.

2.1.3 Environnement

L'environnement de déploiement de nos serveurs est CentOS, une distribution linux, fréquemment utilisée dans le déploiement des serveurs web. Aussi, ce choix constitue un défi personnel puisqu'il s'agit de notre premier contact avec cet environnement.

2.2 Choix de la solution de détection d'intrusion

2.2.1 Etude comparative de quelques IDS les plus populaires sur le marché

Pour faire face aux menaces d'intrusion, les solutions en matière d'IDS sont multiples et diversifiées. Nous pouvons présenter les solutions de détection d'intrusion suivantes :

- Zeek(Ancien Bro)

  Un système de détection d'intrusion réseau gratuit, Bro peut faire plus que simplement détecter une intrusion. Il peut également effectuer une analyse de signature. En d'autres termes, il existe deux étapes de détection d'intrusion dans Bro, à savoir la journalisation et l'analyse du trafic.

- Suricata

  Moteur de détection des menaces réseau robuste, Suricata effectue la collecte de données au niveau de la couche application. De plus, cet IDS peut effectuer la détection d'intrusion, la surveillance de la sécurité du réseau et la prévention d'intrusion en ligne en temps réel.

- Snort

  Le principal outil NIDS, Snort est gratuit et c'est l'un des rares systèmes de détection d'intrusion pouvant être installés sur Windows. Snort n'est pas seulement un détecteur d'intrusion, mais c'est aussi un enregistreur de paquets et un renifleur de paquets. Cependant, la caractéristique la plus importante de cet outil est la détection d'intrusion. Snort effectue une détection d'intrusion à l'aide de méthodes basées sur les anomalies et sur les signatures.

TABLE 2.2 – Comparatif des IDS : Snort, Suricata, Zeek

Critères	**Snort**	**Suricata**	**Zeek**
Performances	Bonne	Excellente	Moyenne
Simplicité d'installation	Moyenne	Moyenne	Facile
Capacités de détection	Bonne	Excellente	Excellente
Scalabilité	Faible	Excellente	Moyenne
Personnalisation	Excellente	Bonne	Excellente
Documentation et communauté	Très active	Bonne	Bonne

Plusieurs critères sont pris en compte pour le choix de notre IDS :

- **Performance** : Snort est reconnu pour sa performance exceptionnelle, capable de traiter rapidement les paquets de données à grande échelle, avec une latence minimale. Suricata, bien qu'ayant des performances solides, peut être plus gourmand en ressources en fonction du trafic, tandis que Zeek, tout en étant performant, peut avoir une légère surcharge par rapport aux deux autres.

- **Prise en charge des protocole**s : Snort et Suricata offrent un support exceptionnel pour une large gamme de protocoles, permettant une détection fine et précise des attaques. Zeek est également performant, mais son focus est davantage sur l'analyse réseau et le comportement, ce qui peut restreindre sa couverture des protocoles.

- **Facilité de configuration** : Snort présente une courbe d'apprentissage modérée en raison de sa configuration détaillée. Suricata offre une configuration plus accessible, tandis que Zeek est plus complexe à mettre en place et nécessite souvent des connaissances avancées en réseau et en script.

- **Communauté/Support** : Snort dispose d'une communauté large et active, ce qui garantit un soutien constant et une mise à jour régulière des signatures. Suricata et Zeek bénéficient aussi d'un bon support communautaire, mais Snort reste en tête grâce à sa longévité et son adoption généralisée.

- **Coût** : Les trois solutions sont gratuites, ce qui permet à Snort, Suricata et Zeek de se positionner comme des solutions attractives pour les entreprises souhaitant des IDS performants sans coûts liés aux licences.

Suite à cette étude, notre choix s'est porté sur Snort. En effet, chacune de ces solutions propose de décentes performances proportionnellement au travail demandé. Les principales raisons de notre choix se résument comme suit :

- **Performance éprouvée** : Snort est reconnu pour sa rapidité et son efficacité dans la détection des intrusions, même sur des réseaux à fort trafic.
- **Large support communautaire** : Sa communauté vaste et active garantit un accès à des mises à jour régulières et à des conseils d'experts.
- **Flexibilité** : Grâce à une personnalisation poussée, Snort peut être configuré pour répondre à des besoins spécifiques et s'adapter à différents environnements.
- **Support de protocoles multiples** : Il offre une couverture exhaustive de différents protocoles, augmentant sa capacité à détecter des attaques complexes.
- **Maturité et fiabilité** : Ayant une longue histoire et de nombreuses installations en production, Snort est une solution mature et fiable, idéale pour des environnements nécessitant une détection robuste.

2.2.2 Emplacement de Snort choisi

Après étude du sujet, nous avons décidé de positionner notre IDS comme l'indique l'architecture suivante :

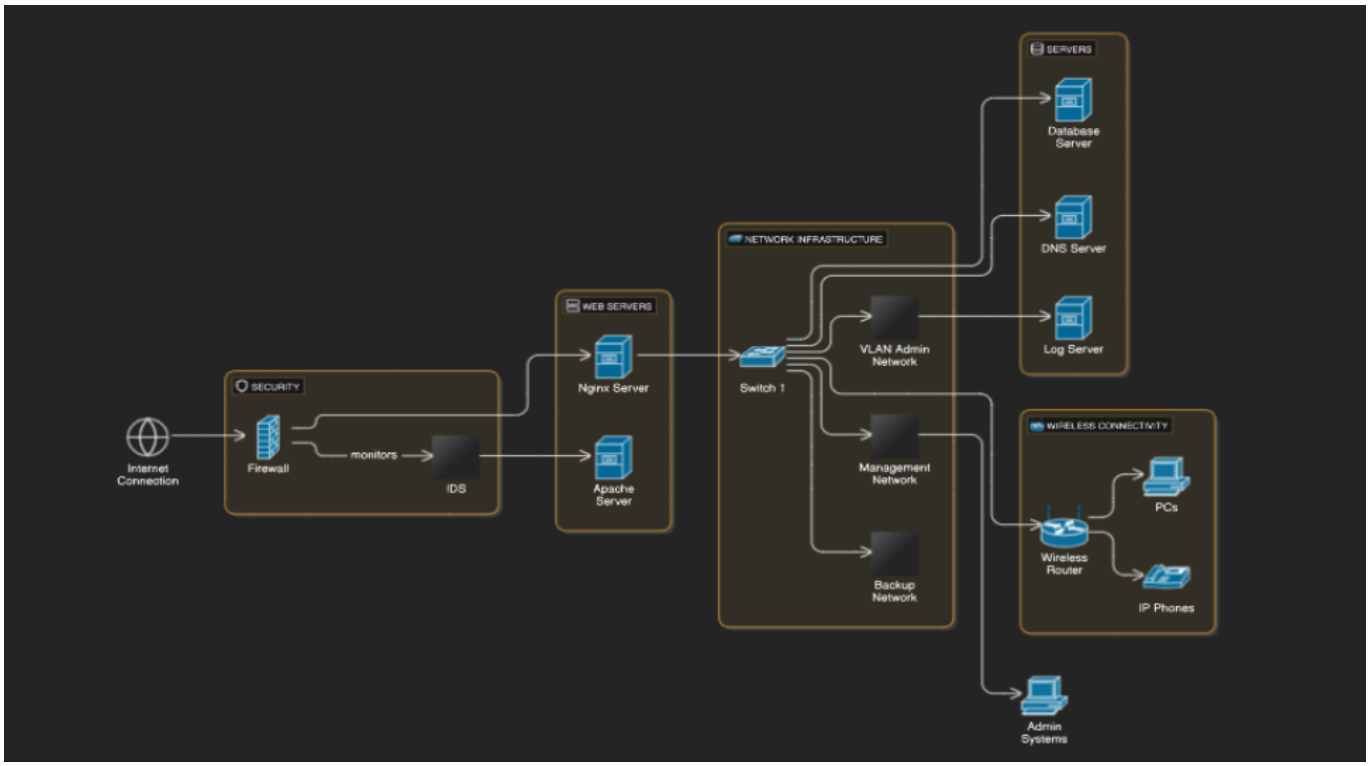


FIGURE 2.1 – Architecture du réseau

2.2.2.1 Avantages et justification

- **Surveillance centralisée** : Le positionnement de l'IDS permet de surveiller à la fois le trafic interne entre les VLAN, le réseau de management, et le réseau principal.
- **Détection avancée** : Il est en mesure d'identifier des attaques ciblées sur les serveurs critiques (Apache, Nginx) ainsi que sur les ressources réseau sensibles (DNS, logs).

- **Protection multi-couches** : Il complète le pare-feu en surveillant le trafic interne, filtrant ainsi les menaces qui auraient franchi la première ligne de défense.
- **Surveillance des VLANs** : Avec un VLAN dédié pour les administrateurs réseau, l'IDS peut contrôler les accès et garantir qu'aucun trafic anormal n'affecte la gestion du réseau.
- **Monitoring des sauvegardes** : En surveillant le réseau de sauvegarde, il garantit qu'aucune attaque ne cible les copies de données essentielles en transit ou en stockage.

2.2.3 Environnement

Ici notre environnement de déploiement est Ubuntu 14 au vu de la complexité de configuration sur les distributions plus récentes et l'incompatibilité de la solution graphique avec celles-ci.

2.2.4 Barnyard

Barnyard prend en charge l'inscription dans la base de données des évènements, allégeant ainsi la charge de snort qui focalise ses ressources sur la détection d'intrusion. Snort inscrit les évènements au format unifié et ces derniers seront exploités par barnyard afin d'être inscrits dans la base de données.

2.2.4.1 Le plugin unified2

Le plugin de sortie unifié est conçu pour être la méthode la plus rapide pour la journalisation des événements de snort. Il enregistre les événements sous un format binaire, ce qui permet d'alléger la charge des mécanismes d'alertes face aux événements.

Le plugin propose deux fichiers de sortie différents : un fichier d'alerte et un fichier journal.

- Le fichier d'alerte contient les détails de haut niveau d'un évènement (IP, Protocole, Port, identifiant du message, etc)
- Le fichier journal contient des informations détaillées sur les paquets (un dump paquet avec l'id de l'évènement associé)

2.2.5 La console base

Par défaut, les alertes de snort sont stockées dans un fichier texte. L'analyse de ce fichier n'est pas aisée même en utilisant des outils de tri. Pour cela, il est recommandé d'utiliser des outils de monitoring. Ici, nous nous intéressons à BASE, qui est largement utilisé sur le marché et surtout est simple de configuration et d'utilisation par rapport aux services fournis.

Pour son fonctionnement, BASE a besoin d'un certain nombre de dépendances :

- Un SGBD installé
- Snort compilé avec le support de ce SGBD
- Php5 : module PHP
- Php-mysql : interface PHP/MYSQL
- la bibliothèque ADODB (Active Data Object Data Base), destinée à communiquer avec le reste du SGBD

2.2.6 PulledPork

Pulled pork est un outil permettant de mettre à jour automatiquement les règles de snort au fur et à mesure qu'elles sont publiées sur le site officiel.

2.2.7 Twilio

Twilio est une API (Application Programming Interface) de communication basée sur le cloud permettant à ses utilisateurs de disposer d'un numéro de téléphone virtuel et de pouvoir émettre et recevoir des appels, MMS et SMS. Il s'agit de notre centre d'envoi d'alertes de Snort par SMS.

2.2.8 Swatch

Swatch est un outil capable de lire les logs de Snort et d'exécuter une action quand certaines conditions spécifiques sont réunies. Il s'agit de notre pont entre Snort et Twilio.

Conclusion

A la fin de ce chapitre, nous avons pu identifier toutes les ressources dont nous avons besoin pour mener à bien notre projet.

Chapitre 3

Mise en place, tests et résultats

Introduction

Dans ce chapitre, nous abordons la partie pratique de notre travail. Nous présentons les différentes installations et configurations effectuées du côté des serveurs, puis la mise en place du système de détection d'intrusion. Enfin, nous testons l'efficacité des différentes solutions proposées.

3.1 Implémentation de la sécurité côté serveur

3.1.1 Installation et configuration du serveur Apache

- Commande d'installation : *sudo yum install httpd*.
- Configuration pour exécution au démarrage : *sudo systemctl enable httpd.service*.

3.1.2 Installation et configuration du certificat SSL

- Installation de Mod SSL : *sudo yum install mod_ssl*.
- Création du certificat avec OpenSSL :

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.
    key -out /etc/ssl/certs/apache-selfsigned.crt
```

- Configuration Apache pour SSL : modifiez */etc/httpd/conf.d/ssl.conf*.
- Redirection HTTP vers HTTPS via *ssl-redirect.conf* :

```
<VirtualHost *:80>
    ServerName www.example.com
    Redirect "/" "https://www.example.com/"
</VirtualHost>
```

Nous avons la sortie suivante :

```
[root@localhost server]# sudo apachectl configtest
AH00558: httpd: Could not reliably determine the server's fully qualified domai
 name, using localhost.localdomain. Set the 'ServerName' directive globally to
uppress this message
Syntax OK
[root@localhost server]#
```

FIGURE 3.1 – Test de configuration

Enfin, on redémarre le serveur :

```
sudo systemctl restart httpd.service
```

3.1.3 Installation et configuration du serveur Nginx

- Installation : *sudo yum install epel-release*, puis *sudo yum install nginx*.
- Démarrage : *sudo systemctl start nginx*.
- Configuration Nginx comme proxy inverse : modifiez */etc/nginx/conf.d/proxy.conf*.

3.1.4 Installation et configuration du certificat SSL pour Nginx

- Création du certificat :

```

    sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/nginx-
        selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

.

- Configuration Nginx pour SSL : modifiez */etc/nginx/conf.d/ssl.conf*.
- Redirection HTTP vers HTTPS : ajoutez dans *default.d/ssl-redirect.conf* :

```
return 301 https://\$host\$request\_uri/;
```

3.1.5 Test de configuration et redémarrage des serveurs

Avant de redémarrer, testons la configuration d'Apache et Nginx :

- Apache : *sudo apachectl configtest*.
- Nginx : *sudo nginx -t*.

Redémarrez ensuite les services :

```
sudo systemctl restart httpd.service
sudo systemctl restart nginx
```

Nous avons la sortie suivante :

```
root@localhost proxy]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@localhost proxy]#
```

FIGURE 3.2 – Configuration

```
sudo systemctl restart nginx
```

3.2 Mise en place de l'IDS

3.2.1 Installation et Configuration de Snort

- Installation des prérequis : *sudo apt-get install flex bison ....*
- Téléchargement et installation de Snort :

```
wget https://snort.org/downloads/snort/snort-2.9.9.0.tar.gz
tar -xvzf snort-2.9.9.0.tar.gz
cd snort-2.9.9.0
./configure --enable-sourcefire
make
sudo make install
```

- Exécution de Snort pour vérifier l'installation : *sudo snort -V*.

```
ubuntu: ~
snort@ubuntu:~$ /usr/sbin/snort -V

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.16 GRE (Build 118)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reser
ved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.5.3
           Using PCRE version: 8.31 2012-07-06
           Using ZLIB version: 1.2.8

snort@ubuntu:~$
```

FIGURE 3.3 – Installation de snort réussie

- Configuration de Snort : création d'un utilisateur non privilégié et configuration des répertoires nécessaires.

```
/etc/snort
├── attribute_table.dtd
├── classification.config
├── file_magic.conf
├── gen-msg.map
├── preproc_rules
├── reference.config
├── rules
│   ├── local.rules
│   ├── iplists
│   │   ├── black_list.rules
│   │   └── white_list.rules
├── sid-msg.map
├── snort.conf
├── so_rules
├── threshold.conf
└── unicode.map
```

FIGURE 3.4 – Validation de configuration de snort

```
      --== Initialization Complete ==--

         -*> Snort! <*-
  ,,_    Version 2.9.16 GRE (Build 118)
 o"  )~  By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
  ''''   Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
         Copyright (C) 1998-2013 Sourcefire, Inc., et al.
         Using libpcap version 1.5.3
         Using PCRE version: 8.31 2012-07-06
         Using ZLIB version: 1.2.8

         Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
         Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
         Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
         Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
         Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
         Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
         Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
         Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
         Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
         Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
         Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
         Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
         Preprocessor Object: SF_POP  Version 1.0  <Build 1>
         Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
         Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>

Snort successfully validated the configuration!
Snort exiting
```

FIGURE 3.5 – Validation de configuration de snort

3.2.2 Installation et configuration de Barnyard2

- Installation des prérequis : *sudo apt-get install -y mysql-server ....*
- Installation de Barnyard2 :

```
wget https://github.com/firnsy/barnyard2/archive/master.tar.gz
tar zxvf barnyard2-Master.tar.gz
cd barnyard2-master
./configure --with-mysql
make
sudo make install
```

- Configuration de Barnyard2 pour la base de données MySQL et validation de l'installation.



FIGURE 3.6 – Installation de Barnyard2 réussie

3.2.3 Installation et Configuration de PulledPork

- Installation des prérequis : *sudo apt-get install -y libcrypt-ssleay-perl ....*
- Installation de PulledPork :

```
wget https://github.com/shirkdog/pulledpork/archive/master.tar.gz
tar xzvf pulledpork-master.tar.gz
cd pulledpork-master/
sudo cp pulledpork.pl /usr/local/bin
```

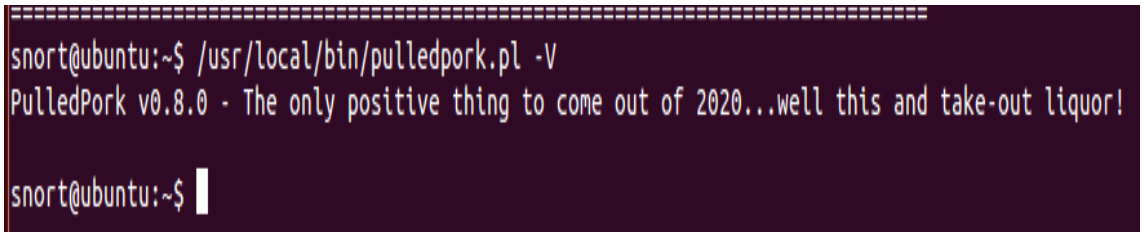


FIGURE 3.7 – Configuration de PulledPork

3.2.4 Création des scripts Upstart

Snort Upstart :

Créez le script Snort Upstart :

```
sudo vi /etc/init/snort.conf
```

Ajoutez le contenu suivant :

```
description "Snort NIDS service"
stop on runlevel [!2345]
start on runlevel [2345]
script
    exec /usr/sbin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0 -D
end script
```

Rendez le script exécutable :

```
sudo chmod +x /etc/init/snort.conf
initctl list | grep snort
```

Barnyard2 Upstart :

Créez le script Barnyard2 Upstart :

```
sudo vi /etc/init/barnyard2.conf
```

Ajoutez le contenu suivant :

```
description "barnyard2 service"
stop on runlevel [!2345]
start on runlevel [2345]
script
    exec /usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var
        /log/snort/barnyard2.waldo -g snort -u snort -D -a /var/log/snort/archived_logs
end script
```

Rendez le script exécutable :

```
sudo chmod +x /etc/init/barnyard2.conf
initctl list | grep barnyard
```

Redémarrez l'ordinateur et vérifiez les services :

```
service snort status
service barnyard2 status
```

3.2.5 Installation de l'interface web BASE

Installation des prérequis et de BASE :

Installez les prérequis :

```
sudo apt-get install -y apache2 libapache2-mod-php5 php5 php5-mysql php5-common php5-gd php5-cli php
    -pear wget tar xzvf base-1.4.5.tar.gz
sudo mv base-1.4.5 /var/www/html/base/
```

Configuration de BASE :

Modifiez le fichier de configuration :

```
sudo vi /var/www/html/base/base_conf.php
```

Ensuite, redémarrez Apache :

```
sudo service apache2 restart
```

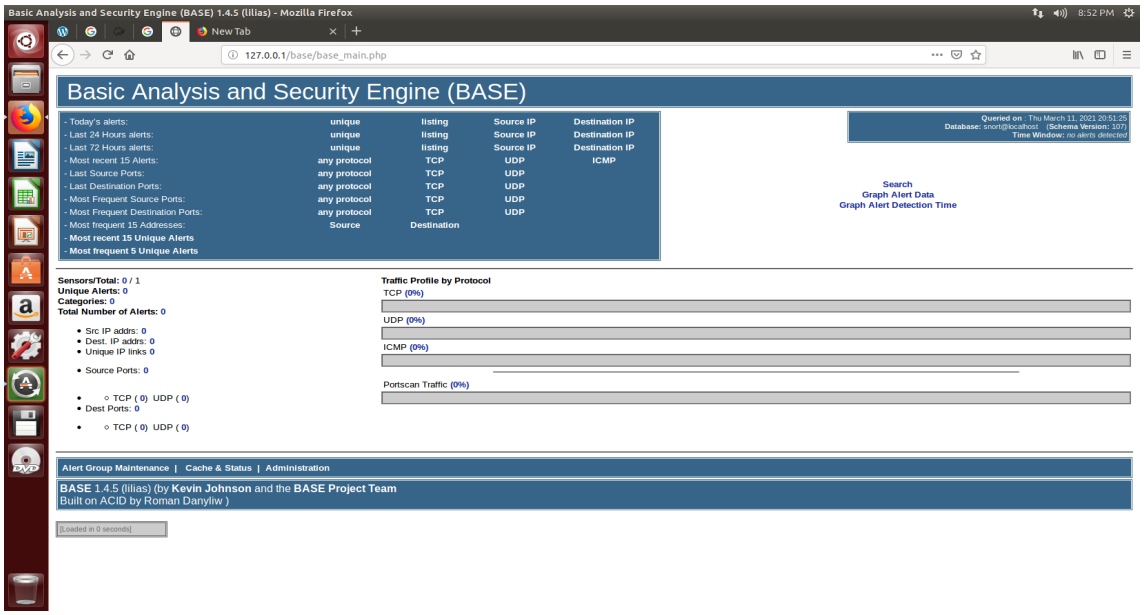


FIGURE 3.8 – Interface Web BASE

Validation :

Allez à l'URL http ://localhost/base et vérifiez si l'interface web est fonctionnelle.

3.3 Tests de la solution mise en place et résultats

3.3.1 Explication de la méthode de tests

Dans un premier temps, nous testons l'activité de notre proxy inverse et la configuration SSL au niveau des serveurs, puis dans un second temps, la capacité de détection de notre IDS

- Serveurs

  Pour se faire nous nous connectons au serveur proxy inverse via un navigateur pour verifier si le contenu du serveur d'origine est disponible. Ensuite, nous vérifions l'activité du certificat SSL au niveau des deux serveurs via le navigateur. Enfin, nous effectuons un test d'interception de données avec l'outil Burpsuite pour vérifier le chiffrement des données avec SSL.

- IDS

  Pour se faire nous allons simuler une attaque visant le serveur proxy à partir d'une machine avec pour OS Parrot Security. C'est alors au tour de notre système d'alerte par SMS de prouver son efficacité

3.3.2 Mise en application et résultats obtenus

- Lorsqu'on se connecte au serveur proxy inverse (192.168.200.131) via le navigateur on obtient la sortie suivante :



FIGURE 3.9 – Configuration réussie

  On remarque que les pages disponibles sur le serveur d'origine (192.168.250.132) s'affichent sans qu'elles ne soient stockées sur le serveur proxy inverse. Nous pouvons donc dire que notre configuration est un succès.

- Nous pouvons ensuite noter une différence dans l'affichage de nos pages web avec l'ajout du certificat SSL. Les résultats se présentent comme suit :

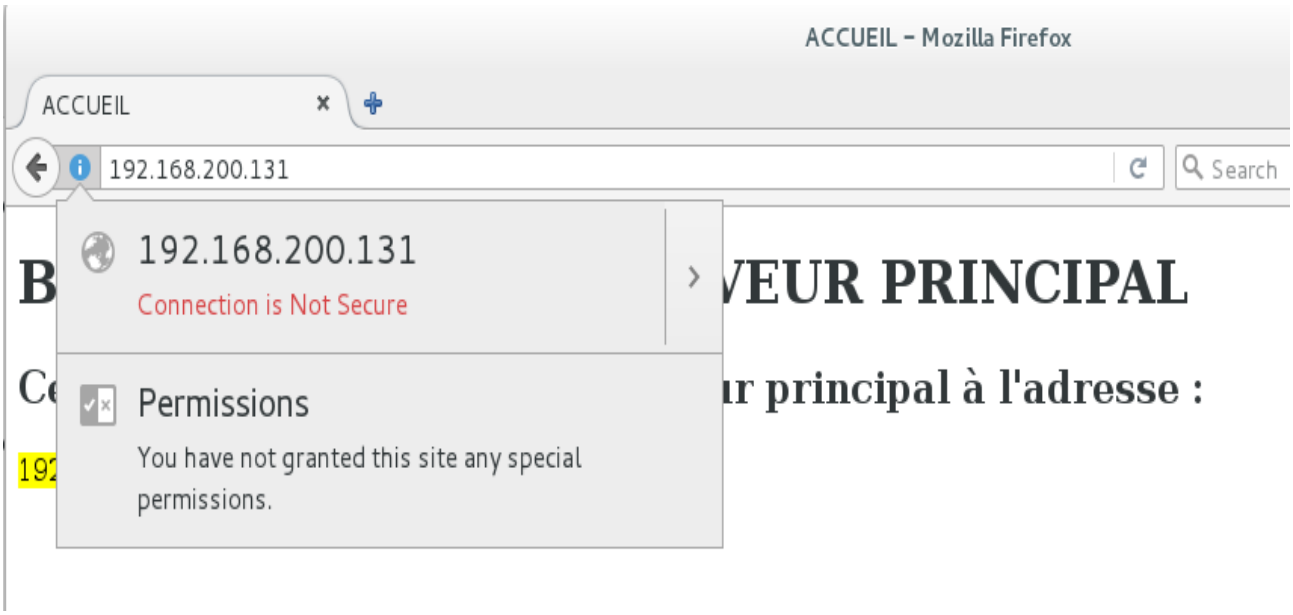


FIGURE 3.10 – Avant SSL



FIGURE 3.11 – Après SSL

- Aussi pour s'assurer de son efficacité nous interceptons une requête d'authentification sur la page d'accueil du serveur web grâce à l'outil Burpsuite. Avant l'application de SSL à l'environnement nous avons les résultats suivants :

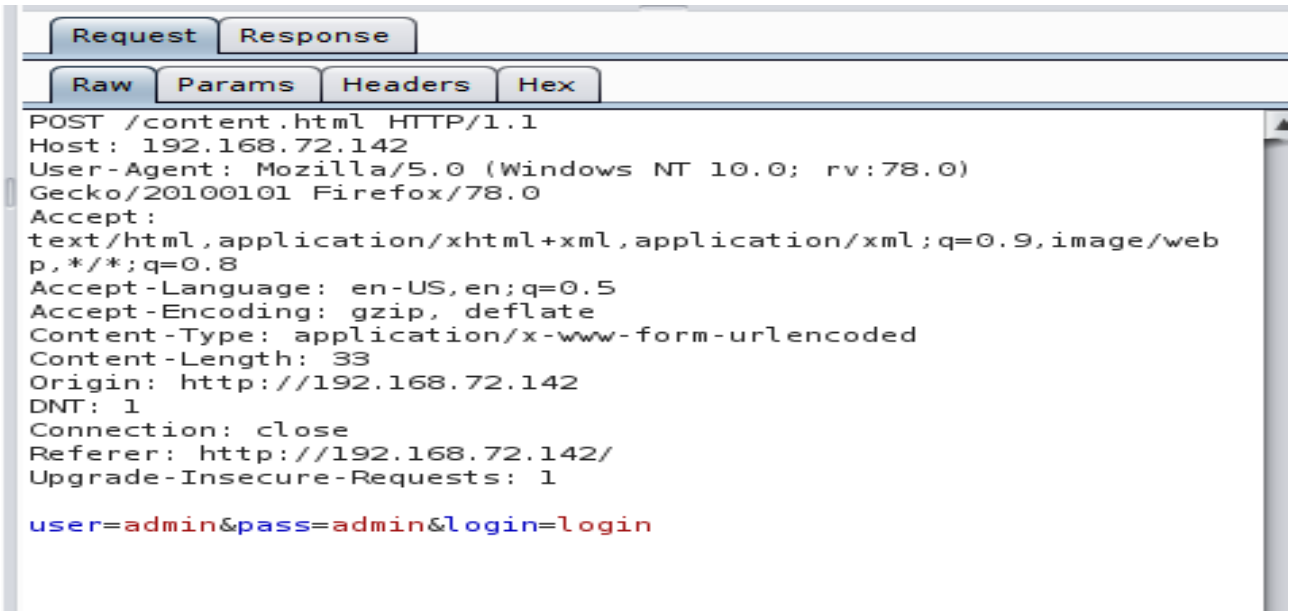


FIGURE 3.12 – Identifiants interceptés sans SSL

On remarque que l'identifiant et le mot de passe sont interceptés en clair. Un attaquant n'aura donc aucun mal à s'en servir de manière frauduleuse.

Après ajout du certificat SSL, on dispose d'une sortie qui se présente comme suit :

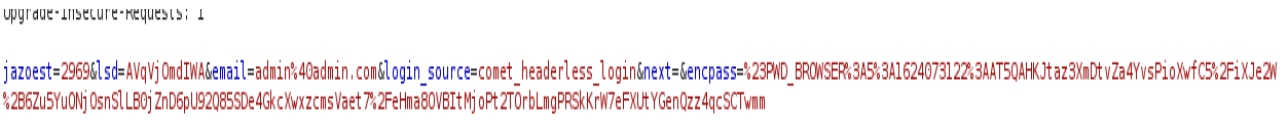


FIGURE 3.13 – Identifiants interceptés avec SSL

On remarque que les identifiants inscrits sont chiffrés, et ne peuvent donc pas être détournés.

- Test de la solution de détection d'intrusion implémentée

Ici nous effectuons une attaque de type **DOS** contre le serveur. Pour cela nous utilisons un outil appelé Xerosploit.

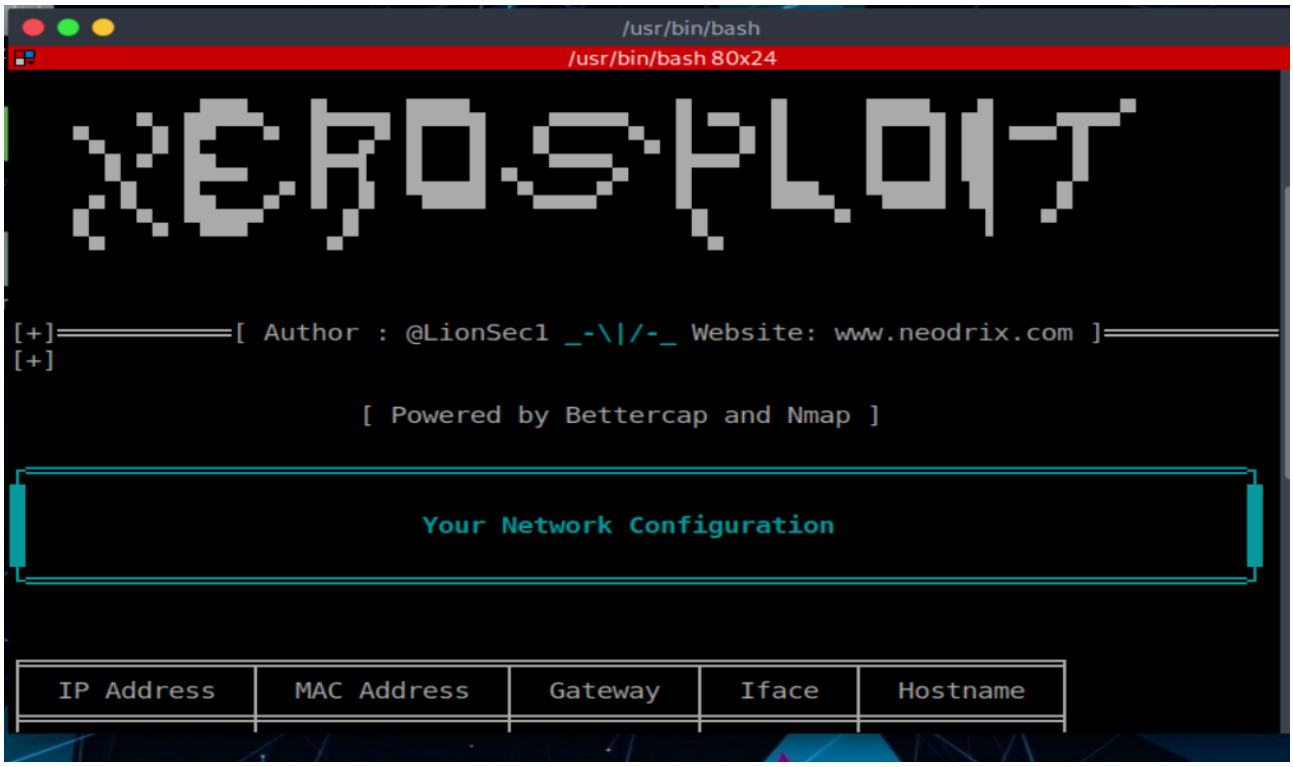


FIGURE 3.14 – Lancement de l'attaque

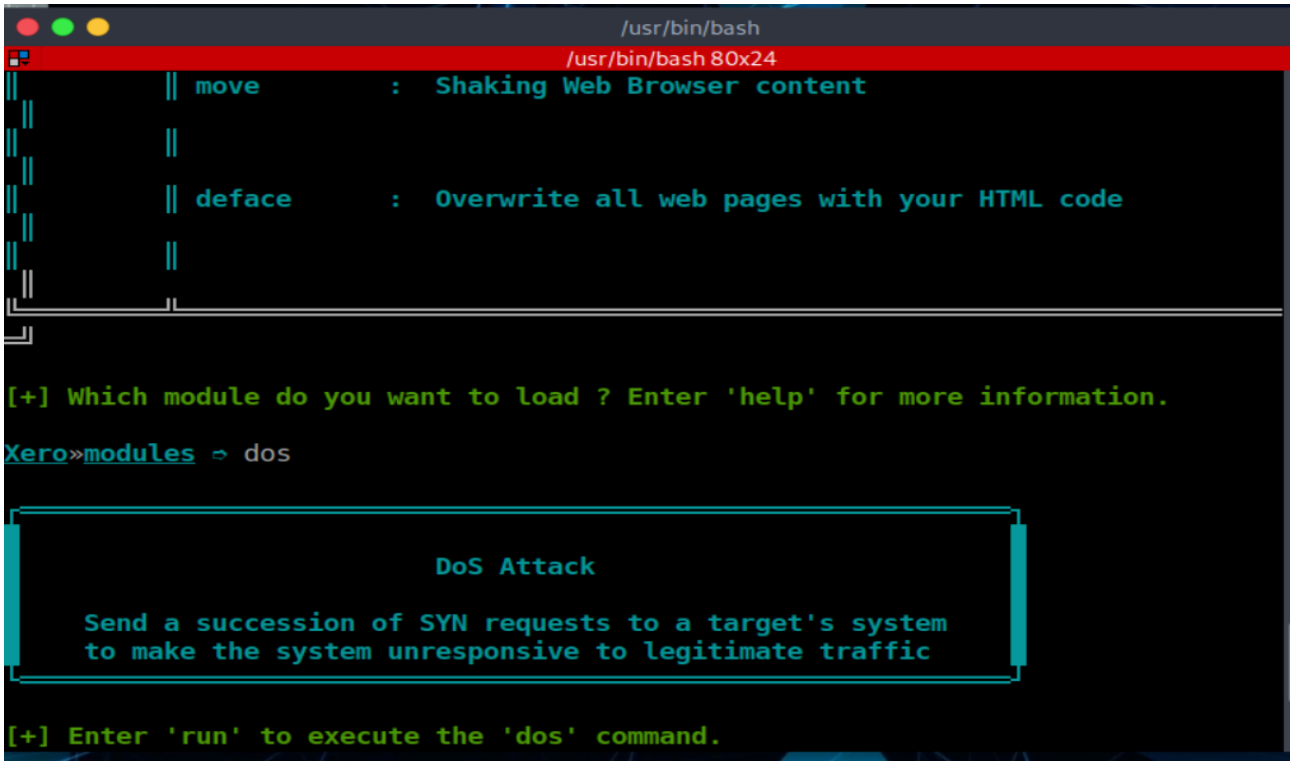


FIGURE 3.15 – Lancement de l'attaque

Quelques secondes après, nous recevons une alerte par SMS de notre système :

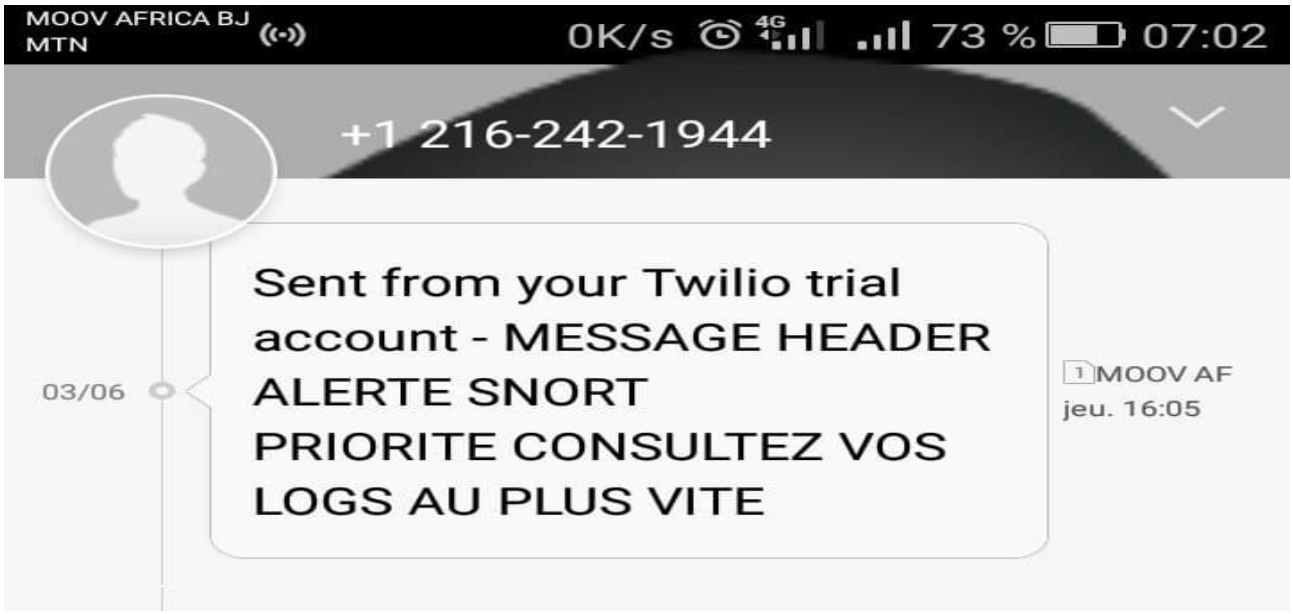


FIGURE 3.16 – Message d'alerte

L'analyse détaillée peut être consultée sur l'interface web et se présente comme suit :

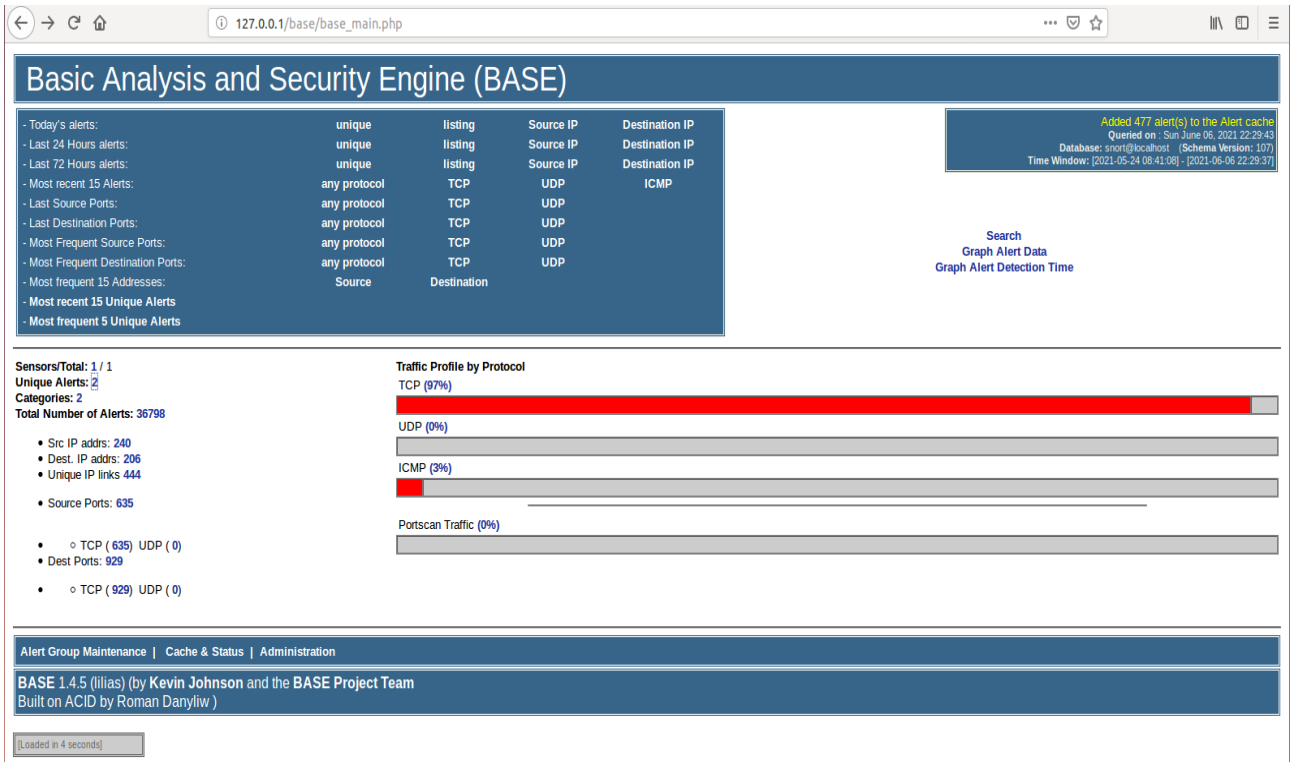


FIGURE 3.17 – Détails sur la menace

3.3.3 Difficultés rencontrées

Les principales difficultés rencontrées se rassemblent autour des points suivants :

- Trouver le système adéquat pouvant accueillir à la fois Snort et Base (Nous avons tenté l'installation sur trois autres systèmes). Le problème majeur ici était l'installation des dépendances.
- Se familiariser avec les commandes importantes du système CentOS

- Trouver une documentation adéquate sur l'utilisation de Nginx comme proxy inverse pour Apache.

Conclusion

A la fin de ce chapitre, nous avons pu achever toutes les configurations nécessaires et tester l'efficacité des solutions mises en place.

Conclusion générale et perspectives

Dans une entreprise, il est vital de disposer d'un système de sécurité pouvant d'une certaine manière assurer la protection de ses actifs. L'objectif principal de notre travail est de mettre en place, pour le compte de NOVEDIX une solution d'hébergement web sécurisée.
Dans ce mémoire, nous avons d'abord présenté différentes attaques pouvant nuire au serveur et justifié le choix des solutions dans une revue de littérature. Ensuite, notre choix s'est porté sur Snort comme IDS et sur les serveurs Apache et Nginx après des analyses et comparaisons. Enfin, nous avons effectué toutes les configurations nécessaires ainsi que quelques tests et présenté nos résultats.
À la fin de cette étude, nous retenons que notre solution, bien qu'efficace après les tests effectués, doit être optimisée pour pouvoir faire face à l'évolution permanante de l'environnement.
Dans ce souci, nous proposons de nouveaux sujets de recherche pour améliorer ce travail. Il s'agit de :

- mettre en place une politique de gestion des intrusions détectées ;
- Mettre en place un pare-feu applicatif ;
- faire une étude des différentes solutions d'hébergement Cloud et les offres se rapportant à notre projet.

Webographie

[1] Sindhuja Hari. Nginx vs apache :head to head comparison. Publié le 7 Janvier 2021, https://hackr.io/blog/nginx-vs-apache. Consulté le 20/02/2021.

[2] KnowledgeHut. Web server hacking and its types. Publié le 9 Novembre 2020, https://www.knowledgehut.com/blog/security/hacking-web-server. Consulté le 28/02/2021.

[3] RAZAFINDRAKOTO Arnaud Ludovic. Mise en place d'un système d'information sécurisé sous linux. Consulté le 04/08/2020.

Table des matières

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Table des figures vi

Liste des tableaux vii

Liste des acronymes 1

Introduction 2

1 Revue de littérature 4
Introduction . . . 4
1.1 Généralités sur la sécurité informatique . . . 4
1.1.1 Définition . . . 4
1.1.2 Objectifs de la sécurité informatique . . . 4
1.2 Les serveurs . . . 5
1.2.1 Définition et Fonctionnement d'un serveur . . . 5
1.2.2 Rôle d'un serveur web . . . 5
1.3 Vue globale de différentes attaques sur les serveurs web . . . 5
1.4 Répercussions d'attaques réussies . . . 6
1.5 Présentation des solutions et justification des choix . . . 7
1.5.1 Proxy inverse . . . 7
1.5.1.1 Présentation . . . 7
1.5.1.2 justification du choix . . . 7
1.5.2 Certificat SSL . . . 8
1.5.2.1 Présentation . . . 8
1.5.2.2 Justification du choix . . . 8
1.6 Les systèmes de détection d'intrusion . . . 9
1.6.1 Définition . . . 9
1.6.2 Fonctions principales d'un IDS . . . 9

2 Analyse et choix techniques 10
Introduction . . . 10

2.1 Mesures de sécurité côté serveurs web . . . 10
2.1.1 Choix des serveurs à utiliser . . . 10
2.1.1.1 Apache . . . 10
2.1.1.2 Nginx . . . 10
2.1.1.3 Analyse comparative des deux serveurs . . . 11
2.1.2 Sécurité des communications sur le serveur web . . . 11
2.1.3 Environnement . . . 11
2.2 Choix de la solution de détection d'intrusion . . . 11
2.2.1 Etude comparative de quelques IDS les plus populaires sur le marché . . . 11
2.2.2 Emplacement de Snort choisi . . . 13
2.2.2.1 Avantages et justification . . . 13
2.2.3 Environnement . . . 14
2.2.4 Barnyard . . . 14
2.2.4.1 Le plugin unified2 . . . 14
2.2.5 La console base . . . 14
2.2.6 PulledPork . . . 15
2.2.7 Twilio . . . 15
2.2.8 Swatch . . . 15
Conclusion . . . 15

3 Mise en place, tests et résultats **16**
Introduction . . . 16
3.1 Implémentation de la sécurité côté serveur . . . 16
3.1.1 Installation et configuration du serveur Apache . . . 16
3.1.2 Installation et configuration du certificat SSL . . . 16
3.1.3 Installation et configuration du serveur Nginx . . . 17
3.1.4 Installation et configuration du certificat SSL pour Nginx . . . 17
3.1.5 Test de configuration et redémarrage des serveurs . . . 17
3.2 Mise en place de l'IDS . . . 18
3.2.1 Installation et Configuration de Snort . . . 18
3.2.2 Installation et configuration de Barnyard2 . . . 19
3.2.3 Installation et Configuration de PulledPork . . . 20
3.2.4 Création des scripts Upstart . . . 20
3.2.5 Installation de l'interface web BASE . . . 21
3.3 Tests de la solution mise en place et résultats . . . 22
3.3.1 Explication de la méthode de tests . . . 22
3.3.2 Mise en application et résultats obtenus . . . 22
3.3.3 Difficultés rencontrées . . . 26
Conclusion . . . 27

Conclusion **28**

Webographie **29**

Table des matières **30**