Ministère de l'Enseignement Supérieure et de la Recherche Scientifique Université Abderrahmane Mira

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

ATE Department

Faculté de Technologie

Département d'Automatique, Télécommunications et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Télécommunications

Spécialité :Réseaux et Télécommunications

Thème

Implémentation d'un serveur web sécurisé : approche pratique sous linux

Préparé par:

BENSTITI nesrine
BOUCHETA sofia

Dirigé par :

M.DIBOUNE Abdelhani

M.MANSOUR Abdelaziz

Examiné par:

Mme MAMMERI Karima

Mme GHERBI Meriem

Année universitaire: 2024/2025

REMERCIEMENTS

*Nous tenons tout d'abord à exprimer notre profonde gratitude envers **ALLAH**, le Grand, l'Infini et le Tout-Puissant, de nous avoir illuminées et d'avoir ouvert les portes du savoir.*

Nous sommes reconnaissantes d'avoir bénéficié de Sa volonté, de la santé et du courage nécessaires pour mener à bien ce travail.

*Nous souhaitons exprimer nos sincères remerciements à notre promoteur, **M. DIBOUNE Abdelhani**, pour son soutien, ses encouragements et ses précieux conseils tout au long de ce travail. Son expertise et ses orientations nous ont été d'une grande aide pour mener à bien notre recherche.*

*Nous tenons également à exprimer notre reconnaissance à l'ensemble des membres du jury, en partiulier à **Mme Mammeri Karima** et **Mme GHERBI Meriem** pour l'intérêt qu'ils ont porté à notre travail.*

Nous sommes reconnaissantes de leur disponibilité et de leur expertise, ainsi que de leur acceptation d'examiner notre travail et de l'enrichir par leurs propositions constructives.

Nous tenons à remercier tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail. Leur soutien, leurs encouragements et leur confiance ont été essentiels dans notre cheminement.

Nous exprimons notre profonde gratitude envers tous ceux qui ont participé à cette aventure et nous ont permis de mener à bien ce travail.

Merci à tous.

Dédicaces

Je tiens tout d'abord à exprimer ma gratitude envers Allah, Le Tout-Puissant, pour m'avoir accordé la force, la patience et le courage nécessaires pour réaliser ce modeste travail.

C'est avec un immense plaisir, et une joie profonde que je dédie ce travail :

À mes très chers parents, Khireddine et Karima, que j'aime énormément, merci pour votre patience, votre amour, vos encouragements constants et vos sacrifices tout au long de mon parcours.Que Dieu vous accorde une longue vie en bonne santé.

À mes précieuses sœurs, Salma et Celia, que le bon Dieu vous protège et vous guide dans votre chemin.

À mes adorables neveux, Mehdi et Amir, que Dieu veille toujours sur vous.

À mes meilleures amies, Yasmine et Lyna, merci pour votre présence, amour et votre soutien sans faille.

Que cette dédicace soit le témoignage sincère de ma reconnaissance envers toutes les personnes qui ont su m'entourer d'amour, de bienveillance et de soutien.
Votre présence a été un véritable pilier dans la réalisation de ce travail.

Merci à tous.

Sofia

Dédicaces

Je tiens tout d'abord à exprimer ma gratitude envers Allah, Le Tout-Puissant, pour m'avoir accordé la force, la patience et la volonté nécessaires à l'accomplissement de ce travail.

C'est avec une immense émotion que je dédie ce travail :

À ma chère mère, Mounia,
pour son amour infini, sa patience, son soutien sans relâche et sa présence constante à mes côtés.
Merci pour tes prières, tes sacrifices et ta force. Que Dieu te protège et t'accorde santé et bonheur.

À la mémoire de mon père, Kamel,
parti trop tôt mais qui reste à jamais vivant dans mon cœur.
Que Dieu lui accorde Sa miséricorde et l'accueille dans Son vaste paradis.

Nesrine

Résumé

La sécurité des serveurs web vise à protéger les données échangées entre un site web et ses utilisateurs contre les accès non autorisés, les intrusions et les attaques malveillantes. Elle est devenue indispensable face à l'augmentation des menaces informatiques et à la sensibilité croissante des informations hébergées en ligne.

Ce mémoire a pour objectif de déployer et sécuriser un serveur web sous Linux, en mettant en œuvre une série de bonnes pratiques en cybersécurité. L'ensemble du travail repose sur l'utilisation d'une machine virtuelle CentOS configurée via VMware.

La solution mise en place comprend l'installation d'un serveur Apache,la sécurisation de la base de données MariaDB, l'installation et la configuration de WordPress, la désactivation de l'accès SSH root, la mise en place d'un pare-feu avec iptables, l'ajout d'ACL, ainsi que le déploiement du protocole HTTPS à l'aide d'un certificat SSL auto-signé. Toutes ces étapes ont permis d'héberger un site web dynamique dans un environnement sécurisé.

Mots clés : sécurité web, Apache, CentOS , WordPress, HTTPS, iptables, VMware.

Abstract

Web server security aims to protect the data exchanged between a website and its users from unauthorized access, intrusions, and malicious attacks. It has become essential due to the growing number of cyber threats and the increasing sensitivity of the information hosted online.

This thesis aims to deploy and secure a web server on a Linux system by applying a set of cybersecurity best practices. The entire work is based on a CentOS virtual machine configured through VMware.

The implemented solution includes the installation of an Apache web server, the securing of the MariaDB database, the installation and configuration of WordPress, the deactivation of SSH root access, the setup of a firewall using iptables, the integration of ACLs, and the deployment of the HTTPS protocol with a self-signed SSL certificate. All these steps made it possible to host a dynamic website in a secure environment.

Keywords*:* web security, Apache, CentOS, WordPress, HTTPS, iptables, VMware.

Table des matières

Liste des Figures

Chapitre I

Chapitre II

ChapitreIII

Chapitre IV

Liste des Tableaux

ChapitreII

Liste des abréviations

A:

ARP: Address Resolution Protocol

AES: AdvancedEncryptionStandard

ACL: Access Control List

C:

CentOS: Community enterprise Operating System

CLI: Command Line Interface

CA: Certification Authority

CMS: Content Management System

CMD : Command

D:

DMZ: Demilitarized Zone

DES: Data encryption standard

DH : Diffie-Hellman

DDos: Distributed Denial of Service

DoS : Denial of Service

DHCP: Dynamic HostConfiguration Protocol Operating System

DNS: Domain Name System

E:

ECDSA: Elliptic Curve Digital Signature Algorithm

EDDSA: Edwards-curve Digital Signature Algorithm

ECDHE: Elliptic Curve Diffie-Hellman Ephemeral

Eth: Ethernet

EPEL: Extra Packages for Enterprise Linux

F:

FTP: File Transfer Protocol

G:

GCM: Galois/Counter Mode

GCC :GNU Compiler Collection

GNU:GNU's Not Unix

H:

HTTPS: Hyper Text Transfer Protocol Secure

HTTP: Hyper Text Transfer Protocol

HTML: Hyper Text Markup Language

I:

IP : Internet Protocol

ID: Identificateur

IPS: Intrusion Prevention System

IDS: Intrusion Detetion System

ICMP: Internet Control Message Protocol

L:

LAN: Local Area Network

LAMP: Linux–Apache–MySQL–PHP

M:

MAN: Metropolitan Area Network

MAC :Message Authentication Code

MitM : Man-in-the-Middle

MTA: Mail Transfer Agent

MySQL: My Structured Query Language

N:

Nmap: Network Mapper

NT: New Technology

NTP : Network Time Protocol

O:

OSI: Open Systems Interconnection

P:

PAN: Personal Area Network

P2P: PeerTo Peer

R:

RSA: Rivest,Shamir,Adleman

RC4 : Rivest Cipher 4

RLE:réseau local d'entreprise

RAS: Remote Access Service

RHEL : Red Hat Entreprise Linux

S:

SSH: Secure Shell

SSL: Secure sockets Layer

SNI: Server Name Indication

SHA : Secure Hash Algorithm

SMTP: Simple Mail Transfer protocol

SYN: Synchronize

SSHFS:SSH File System

T:

TCP: Transmission Control Protocol

TLS: Transport Layer Security

TFN: Tribe Flood Network

U:

URL: Uniform Resource Locator

UDP: User Datagram Protocol

UNIX: UNiplexed Information and Computing System.

W:

WAN: Wide Area Network

Introduction Générale

De nos jours, les systèmes d'informations et les réseaux occupent une place centrale dans le fonctionnement des entreprises, notamment dans les secteurs sensibles. Certaines d'entre elles disposent des réseaux étendus, essentiels à leur activité commerciale, ce qui rend leur surveillance et sécurisation d'autant plus cruciales.

En effet, à mesure que le volume des données stockées dans des systèmes informatiques ne cesse de croître, le nombre d'utilisateurs(identifiés ou non), susceptibles d'adopter des comportements malveillants sur les réseaux, augmente également. Ces individus peuvent tenter d'accéder à des informations sensibles afin de les lire, les altérer, voire de les détruire, compromettant ainsi le fonctionnement normal du système et accentuant sa vulnérabilité.

Dans ce contexte, les serveurs web, en tant que composants centraux du système informatique, représentent des cibles privilégiées pour les cyberattaques. Hébergeant souvent des données sensibles et assurant l'accès à des services critiques, ils doivent faire l'objet d'une sécurisation rigoureuse. En effet, la moindre faille de sécurité peut être exploitée pour compromettre la confidentialité, l'intégrité ou la disponibilité des ressources hébergées. Il est donc primordial de mettre en place des mécanismes de protection adaptés, tels que les pare-feux, le renforcement des configurations de sécurité, ainsi qu'une politique de mise à jour régulières, afin de réduire les risques et renforcer la résilience du serveur face aux menaces.

L'objectif principal de ce projet de fin d'études est d'étudier, de déployer et de sécuriser un serveur web sous Linux, en mettant en œuvre les meilleures pratiques en matière de cybersécurité et en utilisant des outils et services couramment utilisés dans le milieu professionnel. Il s'agit de mettre en place un environnement LAMP (Linux, Apache, MariaDB, PHP) fonctionnel, capable d'héberger une application web dynamique (en l'occurrence WordPress), tout en garantissant un haut niveau de sécurité à la fois au niveau du système et du réseau.

Afin d'atteindre les objectifs fixés, ce mémoire a été structuré en quatre chapitres :

Le premier chapitre introduit les concepts fondamentaux des réseaux informatiques. Il aborde leur classification, leurs différentes architectures, ainsi que le modèle TCP/IP. Ce chapitre se conclut par une introduction aux notions de sécurité réseau,en mettant l'accent sur les vulnérabilités, les menaces et attaques, ainsi que sur les principaux mécanismes de protection.

Le deuxième chapitre expose les bases techniques de la sécurité des serveurs web en présentant les architectures, protocoles et outils essentiels tels que HTTPS, TLS, SSH, pare-feu et ACL. Il explique comment ces mécanismes garantissent la confidentialité, l'intégrité et la protection contre les attaques réseau.

Le troisième chapitre explique la mise en place d'un serveur web Apache sur CentOS,incluant la configuration des utilisateurs, des pare-feux, de SSH et du protocole HTTPS. Il détaille également le déploiement de WordPress ainsi que l'installation des paquets nécessaires.

Le quatrième chapitre présente les tests réalisés sur le serveur web Apache installé sous CentOS. Ces tests permettent de vérifier le bon fonctionnement de l'ensemble des services ainsi que les différents niveaux de sécurisation du serveur.

Chapitre I:Généralités sur les réseaux informatiques

I.1 Introduction

Les réseaux informatiques ont émergé pour répondre au besoin de communication entre terminaux distants. Ils jouent un rôle essentiel dans les entreprises et dans la société, facilitant l'interconnexion des ordinateurs et permettant l'échange de données. Aujourd'hui, ces réseaux sont indispensables à la vie numérique.

Dans ce chapitre, nous aborderons les concepts fondamentaux des réseaux informatiques, en explorant leur classification ainsi que leur architecture. Nous détaillerons notamment le modèle TCP/IP , qui sert de cadres de référence pour la conception et la gestion des réseaux. Enfin, nous terminerons par les généralités sur la sécurité réseau ainsi que ses mécanismes.

I.2 Définition d'un réseau informatique

Un réseau informatique est un ensemble interconnecté d'appareils comme des ordinateurs, des serveurs, des routeurs, des commutateurs, des imprimantes et des périphériques, qui sont conçus pour partager des ressources, communiquer entre eux et échanger des données. Ces appareils sont reliés par des câbles physiques ou des connexions sans fil.

Dans notre société actuelle, les réseaux informatiques font partie intégrante des entreprises et autres entités et ont totalement changé notre manière de travailler et de communiquer, devenant aujourd'hui indispensables.

I.3 Classification des réseaux informatiques

Les réseaux informatiques peuvent être classés selon plusieurs critères tels que leur architecture, leur mode de transmission ou encore leur finalité. Toutefois, l'un des critères les plus couramment utilisés reste l'étendue géographique du réseau. Cette approche permet de regrouper les réseaux en fonction de leur portée physique et de leur zone de couverture. On distingue généralement trois grandes catégories : les réseaux locaux (LAN),les réseaux personnels (PAN), les réseaux métropolitains (MAN) et les réseaux étendus (WAN). Cette classification, bien que simplifiée, reste largement utilisée dans le domaine des réseaux, car elle permet de structurer les solutions techniques et les choix d'infrastructure en fonction des besoins spécifiques liés à la distance, à l'échelle du déploiement, et aux ressources disponibles. Ainsi, en fonction de l'environnement (domestique, professionnel, urbain ou international), un type de réseau sera plus adapté qu'un autre,nous allons présenter une classification de ces réseaux informatiques selon leur étendue**.**

I.3.1 Réseau local d'entreprise (RLE) ou (Local Area Network, LAN)

Lorsque deux stations sont séparées de quelques kilomètres, le réseau sera dit local. Généralement, le réseau local est la propriété du même organisme. C'est le type de réseau que l'on rencontre le plus souvent dans les organismes. Étant donné la proximité des stations, le taux de transmission des données est relativement élevé. On peut utiliser les trois types de support de transmission : la paire torsadée, le câble coaxial ou la fibre optique ; en général, la paire torsadée est la plus fréquemment utilisée [1].

I.3.2 Réseau métropolitain (Metropolitan Area Network, MAN)

Le réseau métropolitain se situe à mi-chemin entre le réseau local et le réseau étendu. Il couvre habituellement les stations d'une même ville. Le support de transmission utilisé est le câble coaxial ou la fibre optique. [1].

I.3.3 Réseau étendu (Wide Area Network,WAN)

Lorsque la distance entre deux stations situées dans des lieux différents atteint quelques centaines de kilomètres, le réseau est dit étendu. Les compagnies disposant de plusieurs sites éloignés géographiquement, tels que ceux gérant les systèmes de réservation de places d'avion et les systèmes bancaires, utilisent ce genre de réseau. Les vitesses de transmission d'un réseau étendu sont généralement moins grandes que celles d'un réseau local [1].

I.3.4 Réseau personnel(Personal Area Network ,PAN)

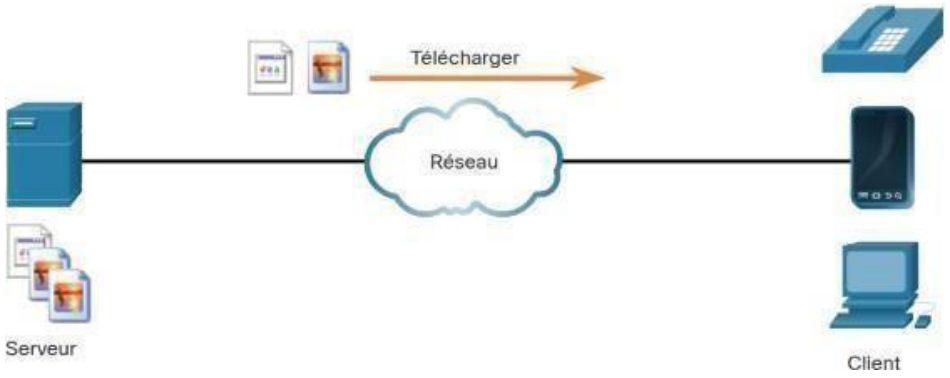
Désigne des réseaux conçus pour une utilisation personnelle ; les plus courants sont l'USB, les technologies sans fil telles que Bluetooth ou IR (infra rouge) ou le wifi[1].

I.4 Architectures des réseaux informatiques

L'architecture des réseaux informatiques désigne la manière dont les différents équipements et systèmes sont organisés pour permettre la communication et l'échange de données. Elle comprend l'ensemble des composants matériels et logiciels qui assurent la transmission, la réception et le traitement des informations entre les utilisateurs et les services disponibles sur le réseau,tels que [2].

I.4.1 Architecture Client/Serveur

Dans le modèle client/serveur, le client envoie une requête d'information et le serveur y répond. Le client est un ensemble matériel/logiciel utilisé pour accéder aux ressources du serveur.Les échanges entre client et serveur se font dans la couche application. Le client initie la demande, et le serveur y répond en envoyant un ou plusieurs flux de données. Des protocoles comme TCP/IP définissent le format des requêtes et des réponses[2].
la figure I.1 illustre comment les fichiers sont téléchargés du serveur vers le client.

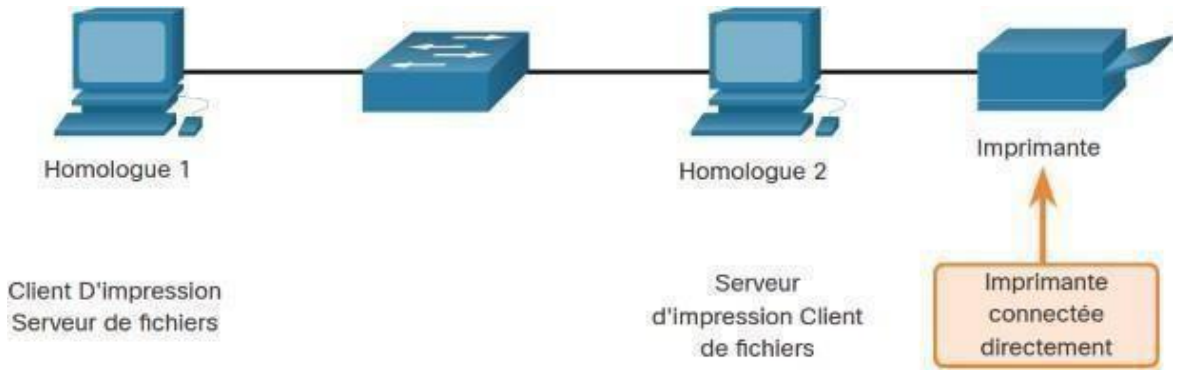


FigureI.1:architecture client/serveur**[2]**

I.4.2 Architecture Pair à Pair(Peer to peer)

Dans le modèle de réseau peer-to-peer (P2P), les données sont accessibles à partir d'un périphérique homologue (Peer) sans l'intervention d'un serveur dédié. Dans un réseau Peer to Peer, deux ordinateurs ou plus sont connectés via un réseau et peuvent partager des ressources (par exemple, des imprimantes et des fichiers) sans disposer de serveur dédié. Chaque périphérique final connecté (ou homologue) peut opérer à la fois en tant que serveur et client. Un ordinateur peut jouer le rôle de serveur pour une transaction et servir simultanément de client pour une autre. Les rôles de client et de serveur sont définis en fonction de chaque requête.Outre le partage de fichiers, un réseau comme celui-ci peut autoriser par exemple le partage de connexion internet.[2].

La figure I.2 illustre que dans un réseau Peer-to-Peer, les deux périphériques sont égaux : chacun peut partager ses ressources. Par exemple, Peer 1 partage des fichiers avec Peer 2, qui en retour partage son imprimante.



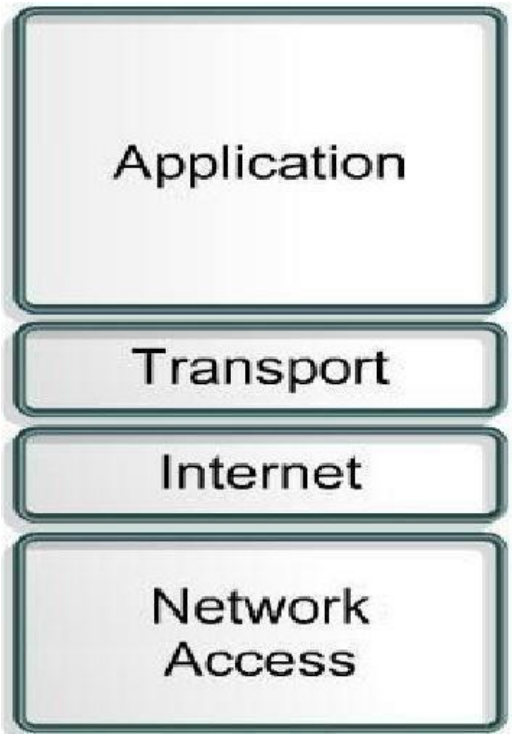
FigureI.2: ArchitectureP2P[2].

I.5 Pile protocolaire TCP/IP

Le modèle de protocole TCP/IP pour les communications sur l'internet a été créé au début des années 1970 et est parfois appelé le modèle internet. Ce type de modèle correspond étroitement à la structure d'une suite de protocoles particulière. Le modèle TCP/IP est un modèle de protocole, car il décrit les fonctions qui interviennent à chaque couche de protocoles au sein de la suite TCP/IP. Il est également utilisé comme modèle de référence [2].

- **Couche Accès Réseau (Network Interface)**:contrôle les périphériques du matériel et les supports qui constituent le réseau [2].
- **Couche Internet**: détermine le meilleur chemin a travers le réseau[2].
- **Couche Transport**: prend en charge la communication entre plusieurs périphériques à travers divers réseaux[2].
- **Couche Application:** représente des données pour l'utilisateur ainsi que du codage et un contrôle du dialogue [2].

Figure I.3 illustre les différentes couches protocolaires du modèle TCP/IP



FigureI.3:modèleTCP/IP

I.6 Généralités sur la sécurité réseau

La sécurité informatique vise a protéger les données et à garantir leur confidentialité, intégrité et disponibilité face aux vulnérabilités, menaces et attaques.

I.6.1 Les vulnérabilités informatiques

Faiblesse affectant un système (au niveau de la conception, de la réalisation,de l'installation, de la configuration ou de l'utilisation)[3].

I.6.2 Exemples de vulnérabilité dans les différentes couches TCP/IP

Les exemples de vulnérabilités suivants sont issus de classifications (ou *taxonomies*) proposées dans deux références majeures [4] et [5] qui offre une taxonomie détaillée des attaques sur les systèmes informatiques et réseaux.

A. Couche Accès réseau

- **Vulnérabilités protocolaires**:bugs ou failles dans des protocoles (TCP, ICMP, ARP, HTTP, etc) ex :exploitation d'un bug dans le protocole TCP avec l'attaque SYN flood.

B. Couche Internet

- **Mauvaise gestion des fragments ICMP**: certaines piles réseau, notamment sur Windows 95/NT ne traitent pas correctement la fragmentation des paquets ICMP. Cette absence de contrôle adéquat permet à un attaquant d'envoyer des fragments ICMP mal formés provoquant un déni de service ou un plantage du système.

C. Couche Transport

- **Ports ouverts non surveillés:** services écoutant sur des ports TCP/UDP vulnérables.

D. Couche Application

- **Données non chiffrées**: faille logicielle liée à la gestion des données (pas de chiffrement, ex : Facebook).

I.6.3 Les menaces informatique

Cause potentielle d'un incident, qui pourrait entraîner des dommages sur un système si cette menace se concrétisait)[3].

I.6.4 Exemples de menaces dans les différentes couches TCP/IP

Les menaces présentées ci-dessous sont issues de deux sources complémentaires [4] et [6], ces dernières elles apportent des exemples concrets et récents de menaces observées en pratique.

A. Couche Accès réseau

- **Espionnage réseau (sniffing):** interception des paquets sur le réseau local ou via un routeur compromis.

B. Couche Internet

- **Employé négligeant ou malveillant:** employé qui sort des données confidentielles (même sans intention malveillante**)**.

C. Couche Transport

- **Manipulation de paquets:**outils de forging pour créer de faux paquets(ex :*Nemesis*, *Packit* ,*ARP-SK*) permettant l'usurpation d'identité.

D. Couche Application

- **Botnets:**réseaux de machines compromises (zombies) contrôlées à distance pour lancer des attaques **.**

I.6.5 Les Attaques informatiques

Une action qui vise a compromettre la sécurité d'un système[3] notamment la confidentialité, l'intégrité ou la disponibilité de ses donnée. Il existe deux types d'attaques :

- **Attaque active**: tente de modifier les ressources du système ou d'affecter leur fonctionnement[7].

- **Attaque passive**:tente d'apprendre ou d'utiliser des informations du système mais n'affecte pas les ressources du système [7].

I.6.6 Exemple d'attaques dans les différentes couches TCP/IP:

Dans le but de mieux comprendre les risques pesant sur les systèmes informatiques, il est utile d'analyser les types d'attaques pouvant survenir à chaque couche du modèle TCP/IP. Les attaques peuvent être passives, lorsqu'elles consistent à intercepter ou surveiller des données sans les modifier, ou actives, lorsqu'elles visent à perturber, altérer ou détourner les communications.
Les exemples présentés ci-dessous sont tirés de différentes sources de référence, notamment [5], [6] et [8] sont classés par couche du modèle TCP/IP afin d'offrir une vision structurée des menaces possibles à chaque niveau.

A. Couche Accès réseau

Attaques passives:

- **Sniffing (Wireshark):** capture de paquets sur le réseau.

Attaques actives :

- **DoS (LandAttack, TCP/UDP/ICMP Flooding):**inondation ou abus de protocoles réseau pour rendre le service indisponible.

B. Couche Internet

Attaques passives:

- **ICMP Sniffing :** Un attaquant intercepte passivement des messages ICMP (comme les requêtes "ping") pour observer le trafic, identifier les hôtes actifs et cartographier le réseau, sans interférer directement avec les communications.

Attaques actives:

- **DHCP Spoofing:** Un attaquant fournit de fausses adresses IP via un faux serveur DHCP, détournant ainsi le trafic réseau pour l'intercepter ou lancer une attaque de type man-in-the-middle.

C. CoucheTransport

Attaques passives:

- **Analyse de trafic(Traffic Analysis):** l'attaquant observe les flux de données TCP/UDP sans les modifier, afin d'en déduire des informations sensibles comme les ports utilisés, la fréquence des connexions ou les services actifs, facilitant ainsi de futures attaques ciblées.

Attaques actives:

- **Buffer Over flow(Slammer, CodeRed):**injection de plus de données qu'un programme peut gérer (exécution de code malveillant).

D. Couche Application

Attaques passives:

- **Keylogger:** logiciel malveillant qui surveille en silence les frappes clavier de l'utilisateur, permettant ainsi à un attaquant de voler des informations sensibles comme des mots de passe ou des données personnelles.

Attaques actives:

- **Trojans (PKZIP 3 Trojan,FTPD Trojan):** logiciels malveillants cachés dans des programmes légitimes – espionnage ou contrôle.

I.7 Les critères de sécurité

Les critères de sécurité sont les bases sur lesquelles repose la protection des systèmes informatiques, afin de prévenir les accès non autorisés, les modifications illégitimes et les interruptions de service.

- **Confidentialité** : seuls les utilisateurs ayant droit ont accès aux données[3].
- **Intégrité** : seuls les utilisateurs ayant droit peuvent modifier les données[3].
- **Disponibilité** : garantir l'accessibilité des données aux utilisateurs ayant-droit[3].
- **Authenticité** : garantir d'avoir l'identité exacte de la source d'une action[3].
- **Non-répudiation** : garantir que le destinataire (resp. émetteur) d'un message ne peut pas prétendre ne pas l'avoir reçu (resp. envoyé) [3].
- **Traçabilité** : consiste à suivre et identifier les accès aux données sensibles ainsi que les modifications qui leur sont apportées, ce qui permet de retracer l'origine, les mouvements et les transformations des données tout au long de leur cycle de vie[9].

I.8 Les mécanismes de sécurité

Les mécanismes de sécurité sont des mesures mises en place pour protéger les systèmes informatiques contre les menaces, en garantissant la sécurité des données et en contrôlant l'accès aux ressources.

- **Pare-feu :** un pare-feu est un dispositif placé entre deux réseaux qui constitue un passage obligatoire entre le réseau de l'entreprise et le reste des réseaux. Il permet de limiter l'accès de l'interne vers l'externe et inversement, en filtrant les entrées et sorties selon des critères tels que l'adresse IP et le port[10].

- **La DMZ (Zone démilitarisée)** : une zone démilitarisée (DMZ) est un sous-réseau se trouvant entre le réseau local et le réseau extérieur. Les connexions à la DMZ sont autorisées, mais elle sert à isoler le réseau interne pour le protéger contre les accès non autorisés[3].
- **IDS (Intrusion Detection System)**: un équipement, logiciel ou matériel, qui automatise le processus d'analyse des évènements d'un ordinateurréseau pour y détecter des signes de problèmes de sécurité[3].

- **IPS (Intrusion Prevention System)**: est une technologie de sécurité conçue pour détecter et bloquer activement ou atténuer les accès non autorisés, les activités malveillantes et les menaces potentielles au sein d'un réseau ou d'un système informatique[11].

- **Chiffrement :**est un mécanisme de sécurité qui transforme des données lisibles (texte clair) en données illisibles (texte chiffré) afin de protéger leur confidentialité contre les accès non autorisés[12**].**

- **Certificat :** un certificat numérique est un fichier ou un mot de passe électronique qui prouve l'authenticité d'un dispositif, d'un serveur ou d'un utilisateur grâce à l'utilisation de la cryptographie et de l'infrastructure de clé publique[13].

- **Les protocoles SSL(Secure Sockets Layer)/TLS (Transport Layer Security)** : sont essentiels à la protection des connexions et des transactions sur Internet. Ils veillent à ce que votre site web inspire confiance et à ce que l'identité du propriétaire du domaine soit vérifiée. Le TLS/SSL est la technologie de sécurité standard qui opère en coulisses pour sécuriser vos transactions et vos processus d'identification en ligne[14].

Ce projet a pour objectif de sécuriser un serveur web en appliquant des mécanismes de sécurité adaptés pour protéger les données et prévenir les attaques. L'utilisation d'un pare-feu permettra de contrôler les accès et filtrer les communications, tandis qu'une DMZ isolera le serveur web du réseau interne pour limiter les risques. De plus, l**'**implémentation de SSL/TLS et des certificats numériques assurera la confidentialité et l'intégrité des échanges de données entre le serveur et ses utilisateurs, en chiffrant les communications. Ces mesures combinées offriront un niveau de sécurité optimal pour le serveur web.

I.9 Conclusion

Les réseaux informatiques, en tant que fondation de l'interconnexion moderne, sont au cœur des échanges de données dans les entreprises et au sein de la société. Leur architecture, qu'elle soit locale, métropolitaine ou étendue, repose sur des protocoles de communication standards comme TCP/IP, qui assurent un acheminement fiable et sécurisé des informations. Toutefois, cette interconnexion expose les réseaux à des vulnérabilités et des menaces diverses, qui peuvent

compromettre la sécurité des systèmes. C’est pourquoi la mise en place de mécanismes de sécurité tels que les pare-feu, les zones démilitarisées (DMZ), les systèmes IDS/IPS, et des techniques de chiffrement devient indispensable. En combinant ces outils et en adoptant des politiques de sécurité rigoureuses, il est possible de créer un environnement sécurisé, garantissant la protection des données et la résilience des réseaux face aux attaques.

ChapitreII:Généralités sur la sécurité des serveurs web: SSH/TLS,pare-feu

II.1 introduction

La sécurité des serveurs web constitue une composante essentielle de la protection des systèmes informatiques. Elle a pour but de protéger les services hébergés sur des serveurs contre les attaques, les intrusions ou bien les fuites d'informations. Les technologies telles que SSH (Secure Shell), TLS (Transport Layer Security) ainsi que les pare-feu jouent un rôle important dans la sécurisation des échanges entre les clients et les serveurs.

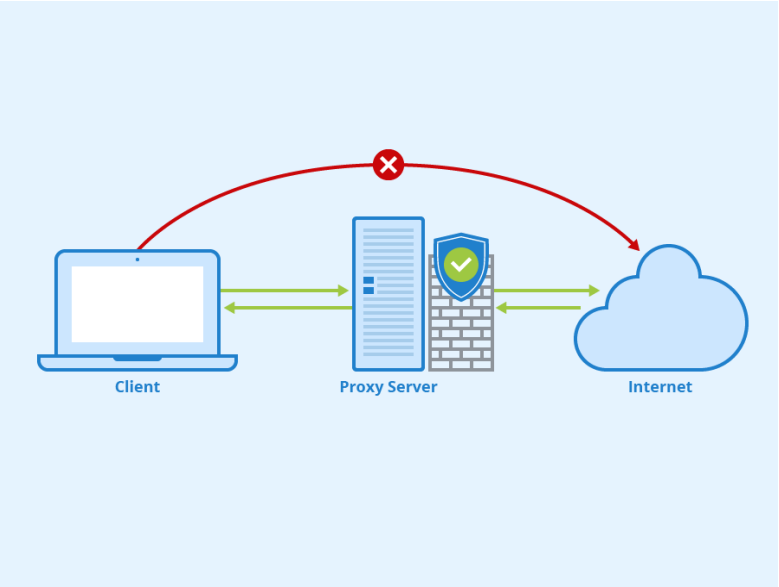
II.2 Architecture client/serveur web

L'architecture client/serveur web est un modèle informatique fondamental qui régit l'interaction entre différents dispositifs informatiques connectés en réseau, tels que des ordinateurs, des serveurs et des périphériques. Elle implique deux acteurs principaux : le client web et le serveur web[15].

II.2.1 Client web :le client web désigne un utilisateur ou une machine qui initie une requête vers un serveur, cette dernière est souvent déclenchée par l'accès à une URL (Uniform Resource Locator) via un navigateur ensuite le navigateur traduit cette URL en adresse IP grâce au service DNS (Domain Name System). Ce mécanismepermetd'associerunnomdedomainelisible(ex.:www.google.com)à une adresse IP compréhensible par les machines, Cela rend les recherches plus faciles à retrouver.

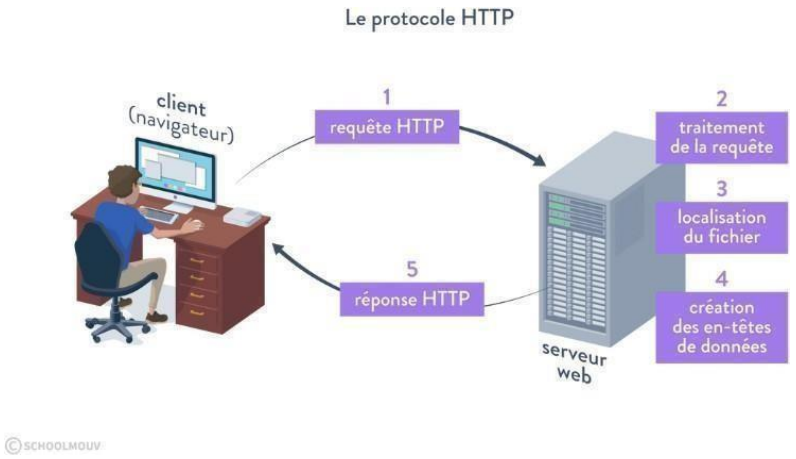
II.2.2 Serveur web: le serveur web est responsable de la réception des requêtes et de la fourniture des ressources demandées comme des pages HTML, des images ou des données. Il peut être final ou proxy.

- **Serveur final:**Il s'agit du serveur d'origine qui héberge les ressources, contenus ou applications demandés par le client, sans passer par un intermédiaire.
- **Serveur proxy:** c'est un intermédiaire entre un client et le serveur final. Il relaie les requêtes tout en offrant plusieurs fonctions telles que :
    - ✓ Il peut anonymiser les connexions en masquant l'adresse IP de l'utilisateur.
    - ✓ Filtrer les accès selon des règles définies (blocage de sites, plages horaires).
    - ✓ Mettre en cache des contenus pour améliorer les performances.
    - ✓ Surveiller l'activité réseau à des fins de contrôle ou d'audit.Figure II.1 illustre la position d'un serveur proxy au sein d'un réseau.



FigureII.1:Illustration du serveur proxy

II.2.3 Échange Client/Serveur : l'échange entre client et serveur repose généralement sur le protocole HTTP ou sa version sécurisée, HTTPS. Lorsqu'un client envoie une requête, le serveur la traite et renvoie par la suite une réponse appropriée.Cet échange entre Client et Serveur est schématisé par Figure II.2. Pour sécuriser ces échanges, le protocole TLS intervient en chiffrant les données, empêchant leur interception ou modification par des tiers. Cela garantit la confidentialité et l'authenticité du serveur.



FigureII.2: architecture client/serveurs web [16]

II. 3Protocole HTTP/HTTPS

Le protocole HTTPS est une extension sécurisée du protocole HTTP. Le « S » pour « Secured » (sécurisé) signifie que les données échangées entre le navigateur de l'internaute et le site web sont chiffrées, et ne peuvent en aucun cas être espionnées (confidentialité) ou modifiées (intégrité) [17].

II.3.1 Fonctionnement du HTTP

HTTP est un protocole qui permet de récupérer des ressources telles que des documents HTML. Il est à la base de tout échange de données sur le Web. C'est un protocole de type client-serveur, ce qui signifie que les requêtes sont initiées par le destinataire (qui est généralement un navigateur web). Un document complet est construit à partir de différents sous-documents qui sont récupérés, par exemple du texte, des descriptions de mise en page, des images, des vidéos, des scripts et bien plus. Les clients et serveurs communiquent par l'échange de messages individuels (en opposition à un flux de données). Les messages envoyés par le client, généralement un navigateur web, sont appelés des requêtes, et les messages renvoyés par le serveur sont appelés des réponses. [18].

II.3.2 Fonctionnement du HTTPS

Le protocole HTTPS assure une communication sécurisée entre le navigateur et le serveur web grâce au chiffrement SSL/TLS. Lorsqu'un utilisateur accède à un site via HTTPS, le navigateur initie une connexion sécurisée. Le serveur répond en envoyant son certificat SSL/TLS, contenant sa clé publique et son identité. Le navigateur vérifie la validité de ce certificat, notamment s'il a été émis par une autorité de certification reconnue. Une fois cette vérification effectuée, le navigateur génère une clé de session symétrique, la chiffre avec la clé publique du serveur, puis l'envoie à ce dernier. Après déchiffrement, les deux parties utilisent cette clé pour échanger des données de manière chiffrée, garantissant la confidentialité et l'intégrité des informations transmises[19].

II.3.3 Comparaison de HTTP/HTTPS

Critères	http	https
Sécurité	Faible: données transmises en texte clair, vulnérables à l'interception,modification,et attaques de type Man-in-the-Middle(MITM)	Élevée:données chiffrées via SSL/TLS, assurant confidentialité,intégrité et authentification du serveur
Fiabilité	Moins fiable, car aucune garantie d'authenticité du site, Susceptible aux attaques et modifications des données	Plus fiable grâce à la validation via certificat SSL/TLS, Garantissant l'identité du site et la sécurité des échanges
URL	Commence par `http://`	Commence par `https://`
Port	Port 80 par défaut	Port 443 par défaut

Attaques	Très exposé: risque élevé d'interception, de modification ou de vol de données	Protection contre les interceptions,falsifications, et usurpations

Tableau II.1:comparaison du HTTP/HTTPS[20]

II.4 Protocoles et chiffrement sécurisés

Les protocoles SSH et TLS jouent un rôle important dans la sécurisation des communications en assurant la confidentialité, l'intégrité et l'authenticité des échanges de données sur un réseau. Cela permet d'établir un canal de communication sûr entre deux parties.Cette sécurité repose principalement sur l'utilisation de techniques de chiffrement, qui assurent que seules les parties autorisées peuvent accéder aux données échangées.

II.4.1 Cryptage asymétrique

Le chiffrement asymétrique utilise un ensemble de deux clés : une clé publique pour le chiffrement et une clé privée pour le déchiffrement (cette dernière étant connue uniquement du destinataire). La clé privée doit être gardée secrète par le destinataire, car toute partie ayant accès à une clé privée est en mesure d'accéder aux données confidentielles.

Dans les communications sécurisées, notamment lors du processus de *handshaking* (échange initial) dans des protocoles comme TLS, le chiffrement asymétrique est utilisé pour échanger en toute sécurité une clé symétrique. Cette clé servira ensuite à chiffrer l'ensemble de la session à l'aide d'un algorithme symétrique, plus rapide et adapté au traitement de grandes quantités de données.

Parmi les algorithmes de chiffrement asymétrique les plus couramment utilisés dans ce contexte, on peut citer :

1. **RSA(Rivest–Shamir-Adleman)**

Le sytème RSA est un algorithme de chiffrement à clé publique utilisé pour l'établissement de clés ainsi que pour la génération et la vérification de signatures numériques [21].

Le RSA est un algorithme de chiffrement asymétrique basé sur des opérations mathématiques complexes liées à la factorisation de grands nombres premiers. Il utilise une paire de clés : une clé publique, utilisée pour chiffrer les données, et une clé privée, utilisée pour les déchiffrer. Cet algorithme est principalement utilisé au début de la communication pour échanger une clé symétrique de manière sécurisée. Cette clé sera ensuite utilisée avec un algorithme symétrique pour chiffrer les données échangées.

Grâce à sa capacité à sécuriser l'échange initial, RSA joue un rôle crucial dans la mise en place d'une communication confidentielle, bien qu'il ne soit pas utilisé pour chiffrer de grandes quantités de données en raison de sa lenteur.

II.4.2 Cryptage symétrique

Le chiffrement symétrique est un procédé de chiffrement où la même clé secrète est utilisée à la fois pour chiffrer et déchiffrer les données. Cette méthode est rapide et efficace pour protéger les informations, mais nécessite que la clé soit partagée de manière sécurisée entre les parties avant tout échange, ce qui peut représenter une faiblesse si cette clé est interceptée. [22].

Bien que le chiffrement symétrique repose généralement sur des algorithmes tels que AES, 3DES, ChaCha20 ou RC4 (désormais obsolète), il est souvent utilisé en combinaison avec des mécanismes asymétriques, notamment dans les systèmes de sécurité hybrides comme TLS/HTTPS.

1. **RC4(RivestCipher4):** algorithme de chiffrement symétrique par flot,connu pour sa simplicité et sa rapidité. Il est largement utilisé dans des protocoles comme SSL, mais il est aujourd'hui considéré comme vulnérable et obsolète en raison de failles de sa génération de flux pseudo-aléatoire.

2. **DES (Triple Data Encryption Standard):** c'est une extension du DES standard. dans cet algorithme, le chiffrement se fait en trois étapes : la première étape utilise une clé de chiffrement pour chiffrer les données. La deuxième étape utilise une clé de déchiffrement pour déchiffrer les données chiffrées. La dernière étape utilise une autre clé de chiffrement pour chiffrer à nouveau les données. Cette technique est connue sous le nom de "chiffrement en cascade". De nos jours, 3DES ne répond plus aux exigences de sécurité modernes, d'où sa mise en retrait progressive.

3. **ChaCha20 :**algorithme de chiffrement symétrique par flot conçu comme une alternative plus sécurisée à RC4.Il est basé sur le chiffrement de blocs de 64 bits mais avec une construction de génération de flux pseudo-aléatoire plus robuste.Il est utilisé dans des protocoles comme TLS et dans des applications de sécurité modernes et est souvent préféré pour sa sécurité accrue par rapport à des algorithmes plus anciens comme RC4 et 3DES.

4. **AES(Advanced Encryption Standard):**l'algorithme symétrique le plus utilisé de nos jours. Il combine rapidité, efficacité et un haut niveau de sécurité. Il prend en charge plusieurs longueurs de clé (128, 192 et 256 bits) et est largement déployé notamment dans des secteurs sensibles tels que les transactions financières et la protection des données personnelles.

II.5 Certificat SSL/TLS

II.5.1 Structure du certificat SSL/TLS

Lorsqu'un client souhaite établir une connexion sécurisée avec un serveur via SSL/TLS, il envoie une requête de connexion. En réponse, le serveur transmet son certificat X.509, qui contient sa clé publique. Le client vérifie ce certificat auprès d'une autorité de certification (CA) pour s'assurer de son authenticité. Une fois validé, le client génère une clé symétrique (appelée aussi clé de session) et la chiffre avec la clé publique du serveur. Le serveur peut alors déchiffrer cette clé grâce à sa clé privée. À partir de là, les deux parties partagent une clé symétrique permettant de chiffrer et déchiffrer les données échangées en toute sécurité. Une fois la session terminée, cette clé est supprimée, garantissant qu'elle ne puisse pas être réutilisée ou interceptée. Ce mécanisme est illustré dans la Figure III.3.

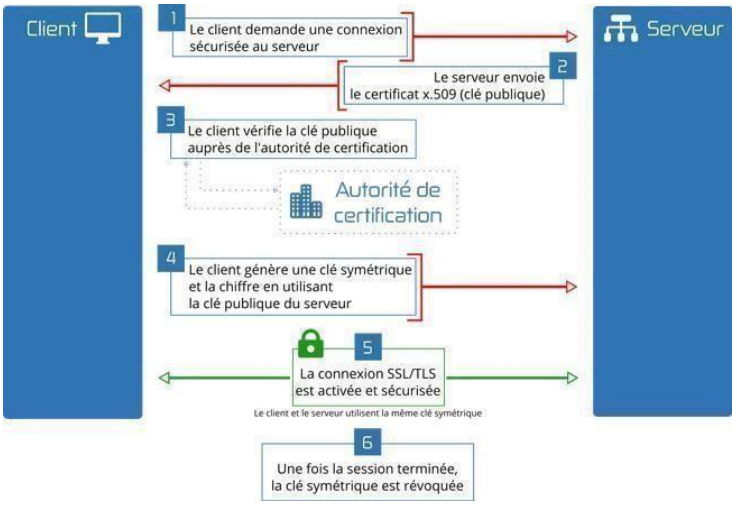


Figure II.3 : Structure certificat SSL/TLS[23]

II.5.2 Principe du hachage et de la signature

Les certificats numériques reposent sur deux mécanismes cryptographiques fondamentaux : le hachage et la signature numérique. Ensemble, ils garantissent l'authenticité du certificat, c'est-à-dire qu'il provient bien de l'émetteur annoncé, et l'intégrité, c'est-à-dire qu'il n'a pas été altéré depuis sa génération.

1. Hachage: le hachage consiste à appliquer une fonction algorithmique (appelée fonction de hachage cryptographique) sur un ensemble de données afin de produire une empreinte unique, appelée *digest*. Cette empreinte est de taille fixe, peu importe la taille du message d'origine. Elle sert à détecter toute modification, car même un petit changement dans le contenu entraînera une empreinte complètement différente.

2. Signature:la signature numérique repose sur la cryptographie asymétrique. Elle utilise une paire de clés :

Une clé privée :conservée secrète par l'émetteur (ici, l'autorité de certification – CA).

Une clé publique: accessible à tous.

II.5.3 Algorithmes utilisés

1. RSA-2048

- **Type:** algorithme de cryptographie asymétrique à clé publique.

- **Taille de clé:** 2048bits.

- **Principe:** RSA repose sur la difficulté de factoriser un grand nombre entier, produit de deux grands nombres premiers choisis aléatoirement[24].

- **Fonctionnement**[25]:

  - Génération de deux grands nombres premiers pp et qq.
  - Calcul du modulen=p×qn=p×q.
  - Calcul de la fonction totient ϕ(n)=(p−1)(q−1)ϕ(n)=(p−1)(q−1).
  - Choix d'un exposant public (souvent65537) sans facteur commun avec ϕ(n)ϕ(n).
  - Calcul de l'exposant privé dd, inverse modulaire de ee modulo ϕ(n)ϕ(n).
  - La clé publique est (e,n)(e,n) et la clé privée (d,n)(d,n)

- **Utilisation:** chiffrement, signatures numériques, sécurisation des échanges (TLS, SSH, email) [26].

- **Avantages et inconvénients:** RSA-2048 est sécurisé, mais plus lent et gourmand en ressources que les algorithmes à courbes elliptiques.

2. SSH-256

- **Type:** fonction de hachage cryptographique, partie de la famille SHA-2.

- **Sortie:** 256 bits(32octets).

- **Fonction:** transforme des données d'entrée de taille arbitraire en une empreinte numérique fixe, unique et irréversible [25].

- **Utilisation:** intégrité des données, signatures numériques, empreintes de clés SSH, blockchain[27].

- **Caractéristiques:** résistance aux collisions, sensibilité à la moindre modification des données, largement adopté dans la cyber sécurité moderne[25].

3. ECDSA(Elliptic Curve Digital Signature Algorithm)

- **Type:** Algorithme de signature numérique basés sur la cryptographie à courbe selliptiques (ECC).

- **Principe:** utilise les propriétés mathématiques des courbes elliptiques pour générer des signatures numériques sécurisées, avec des clés plus courtes que RSA pour un niveau de sécurité équivalent [28].

- **Fonctionnement:** génération d'une paire clé privée/publique, création et vérification de signatures numériques basées sur des opérations sur la courbe elliptique [28].

- **Avantages:** clés plus courtes,calcul plus rapide, adapté aux environnements contraints, résistance accrue aux attaques quantiques potentielles comparé à RSA[28].

- **Utilisation:**TLS,blockchain (Bitcoin),communications sécurisées.

4. EdDSA(Edwards-curve Digital Signature Algorithm)

- **Type:** algorithme de signature numérique utilisant des courbes elliptiques de type Edwards, souvent Curve25519 (Ed25519) [29].

- **Caractéristiques:** signatures déterministes (évite les failles liées à la génération aléatoire), plus rapide et plus simple à implémenter que ECDSA, résistant aux attaques par canaux auxiliaires [29].

- **Fonctionnement:**basé sur des opérations sur la courbe elliptique, produit des signatures compactes et sécurisées [29].

- **Utilisation:** applications modernes nécessitant des signatures efficaces et sûres, adoption croissante dans les protocoles récents.

II.6 Principe du fonctionnement du SSH/TLS

Le SSH et le TLS sont des protocoles utilisés pour sécuriser les communications sur des réseaux non sécurisés, ils suivent un processus appelé handshaking qui inclut des étapes comme l'échange de certificats, la génération de clés secrètes et dans certains cas l'authentification du client, afin d'assurer la confidentialité et l'intégrité des données échangées.

II.6.1 Hello Client

Lorsqu'un client souhaite établir une connexion sécurisée avec un serveur, la première étape consiste en l'envoi d'un message d'initialisation de type "Hello". Le format de ce message varie selon le protocole utilisé (TLS pour le web / SSH pour les accès distants) et permet d'initier la négociation des paramètres de sécurité.

Contenu du message Client Hello

Dans le cadre du protocole TLS, le message *Client Hello* inclut notamment [30] :

- **La version du protocole**:indique la version de TLS supportée par le client (ex: TLS 1.2).
- **Client Random**:32octets générés aléatoirement par le client composé de :
  - ✓ **4octets**: times tamp(date/heure)
  - ✓ **28octets**:valeur aléatoire(cryptographique)
  - ✓ **Utilité**:contribue à la génération des clés de session et empêche les attaques par rejeu (replay attacks)
- **Suites cryptographiques**: liste des algorithmes supportés.
- **SessionID:**identifiant de session, pour éventuellement reprendre une session existante.
- **Méthode de compression**:sert à indiquer si les données doivent être compressées avant d'être chiffrées.Ce champ est toujours présent dans les messages Client Hello mais il n'est plus utilisé pour des raisons de sécurité. La seule valeur utilisée aujourd'hui est null(0) ce qui signifie pas de compression.
- **Des extensions supplémentaires**:
  - ✓ Les noms d'hôte(SNI)
  - ✓ Les groupes de Diffie-Hellman
  - ✓ Méthodes de compression

II.6.2 Hello server

Lorsqu'un serveur reçoit le message *Client Hello*, il répond à son tour par un message *Server Hello*. Ce dernier contient les paramètres de sécurité choisis parmi ceux proposés par le client, tels que la version du protocole, la suite cryptographique (*cipher suite*) retenue et un identifiant de session. Le serveur peut également y inclure son certificat numérique afin de permettre au client de vérifier son identité. Cela marque le début de la phase de négociation sécurisée entre le client et le serveur.

Contenu du message Server Hello

Dans le cadre du protocole TLS, le message Server Hello inclut notamment[30].

- **Version du protocole:** Indique la version TLS retenue par le serveur (généralement la version la plus élevée compatible avec le Client Hello).
- **Server Random***(*équivalent du ClientRandom).

32octets générés aléatoirement par le serveur :

- ✓ **4 octets**:times tamp (optionnel enTLS1.3)

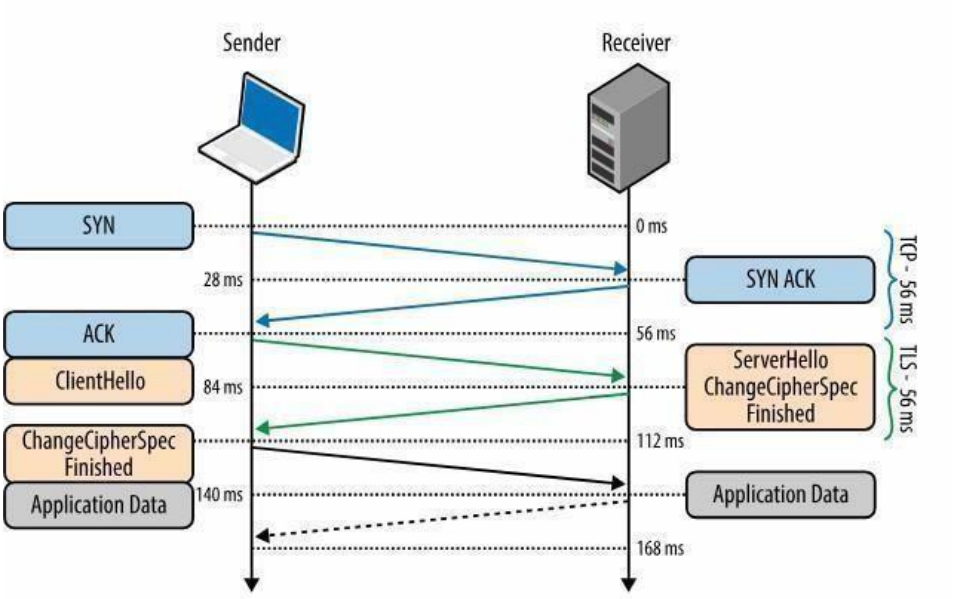
- ✓ **28 octets**: données aléatoires cryptographique.
- ✓ **Utilité:**Participe à la génération du Pre-Master Secretet du Master Secret et garantit l'imprévisibilité des clés de session.

- **Suite cryptographique sélectionnée:** le serveur choisit une seule suite parmi celles proposées par le client, contient :
  - ✓ **Échange de clés**(exemple:ECDHE-RSA)
  - ✓ **Chiffrement** (exemple :AES-256-GCM)
  - ✓ **Intégrité** (exemple :SHA384)

- **SessionID**(optionnel): si le client a fourni une SessionID valide:
  - ✓ Le serveur le reprend pour réutiliser la session existante.
  - ✓ Si non un nouvel ID est généré pour une future reprise.

  Méthode de compression: sert à indiquer si les données doivent être compressées avant d'être chiffrées. Ce champ est toujours présent dans les messages *ServerHello*, mais il n'est plus utilisé pour des raisons de sécurité. La seule valeur utilisée aujourd'hui est *null* (0), ce qui signifie : pas de compression.

- **Extensions** (réponse aux extensions du Client Hello)
  - ✓ **Groupes Diffie-Hellman**:Spécifie le groupe retenu(exemple:*secp256r1*).
  - ✓ **Signature Algorithms**: algorithme de signature utilisé (exemple:*RSA- PSS*).



FigureII.4: Protocole d'établissement de session TLS[31]

Cette étape permet au serveur de prendre connaissance des préférences du client et de décider de la manière dont la communication sécurisée sera établie.Ces différentes étapes sont résumées dans Figure II.4.

II.6.3 Échange de certificat dans SSL/TLS

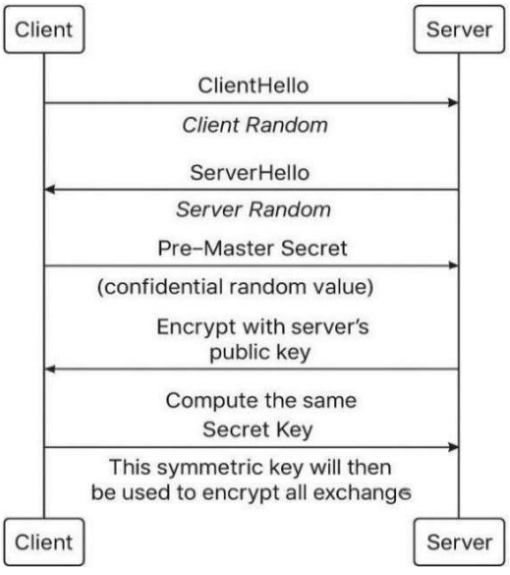
Lors de l'établissement d'une connexion sécurisée via SSL/TLS le serveur envoie un certificat numérique au client pendant la phase du 3 handshaking. Ce certificat joue un rôle essentiel car il permet au client de vérifier l'identité du serveur.

1. Principe de fonctionnement

Le serveur envoie au client un certificat numérique dans le but de prouver son identité, ce certificat joue un rôle important dans la mise en place d'une relation de confiance car il contient plusieurs informations importantes telles que le nom du serveur (par exemple :www.monsite.com), la clé publique du serveur et les détails de l'autorité de certification (CA) qui a délivré le certificat, ainsi que sa période de validité. Le certificat inclut également une signature numérique apposée par l'autorité de certification afin de garantir son authenticité.
À la réception de ce certificat le client procède à une vérification rigoureuse,il s'assure que le certificat est bien signé par une autorité reconnue et de confiance et assure qu'il n'est pas expiré et que le nom du serveur correspond bien à celui qu'il souhaite contacter. Si toutes ces vérifications sont réussies le client peut alors avoir confiance en la clé publique contenue dans le certificat qui sera ensuite utilisée pour échanger la clé secrète ou procéder à un échange de clés sécurisé.
L'échange de certificat constitue une étape fondamentale pour garantir l'authenticité du serveur et la sécurité des échanges futurs.

II.6.4 Génération de la clé secrète

Après validation du certificat du serveur,le client génère un pre-master secret,un nombre aléatoire confidentiel. Ce secret est ensuite chiffré à l'aide de la clé publique du serveur(extraite du certificat)et envoyé. Le serveur de son côté,le déchiffre à l'aide de sa clé privée. À partir de ce pre-master secret ainsi que des deux valeurs aléatoires échangées précédemment (client random et server random), le client et le serveur calculent indépendamment la même clé secrète,qui servira à sécuriser la suite de la communication[32].Ces étapes sont dépeintes dans Figure II.5



FigureII.5: Schéma de la génération clé secrète

Cette clé symétrique servira ensuite à chiffrer tous les échanges de manière sécurisée.

En plus de la méthode classique basée sur RSA, il existe une autre approche plus sécurisée pour générer la clé secrète : c'est la méthode Diffie-Hellman, utilisée dans certaines versions de TLS.

Pour la version TLS 1.2, la méthode Diffie-Hellman est optionnelle : on peut choisir RSA pour l'échange de clés. En revanche, dans TLS 1.3, elle est obligatoire, car seul Diffie-Hellman (notamment ECDHE) est utilisé.

II.6.5 Génération de la clé secrète avec Diffie-Hellman

Dans certains protocoles comme TLS (notamment à partir de la version 1.2) et SSH, l'algorithme Diffie-Hellman est utilisé pour établir une clé secrète sans l'envoyer directement. À la place, le client et le serveur échangent des clés publiques temporaires, puis chacun utilise sa clé privée pour calculer localement la même clé secrète. Ce procédé permet de renforcer la sécurité, car même si un attaquant intercepte les clés publiques échangées il ne pourra pas retrouver la clé finale sans connaître les clés privées. Cette méthode est aujourd'hui privilégiée car elle garantit une propriété importante qui est la confidentialité persistante.

II.7 Échange de données

Après l'étape du handshaking, le client et le serveur partagent une clé de session. Cette clé est utilisée pour chiffrer les données échangées entre eux, afin que personne d'autre ne puisse les lire.

Chaque message est protégé non seulement par le chiffrement (pour la confidentialité), mais aussi par un code d'authentification comme un MAC, basé sur des fonctions de hachage telles que SHA afin de garantir qu'il n'a pas été modifié pendant la transmission.

Dans certains cas, on ajoute aussi une signature numérique pour vérifier que le message vient bien de l'expéditeur légitime.

II.8 Destruction de la clé

La destruction de la clé de session est une étape cruciale après une communication sécurisée via des protocoles comme SSH ou TLS. Une fois les données échangées, la clé symétrique utilisée pour chiffrer et déchiffrer ces informations doit être supprimée afin d'empêcher toute récupération ou utilisation non autorisée ultérieure. En effet, si la clé n'est pas détruite, un attaquant ayant accès au système pourrait potentiellement la retrouver et compromettre la confidentialité des échanges.

II.9 Sécurisation des serveurs par pare-feu

Dans un contexte où les cybermenaces sont de plus en plus fréquentes, la protection des réseaux informatiques est devenue une priorité et l'un des mécanismes essentiels de cette sécurité est le pare-feu**.**

II.9.1 Définition du pare-feu

Un pare-feu (ou *firewall* en anglais) est un dispositif de sécurité informatique, matériel et/ou logiciel, conçu pour surveiller, filtrer et contrôler le trafic réseau entre différentes zones de confiance.Il applique des règles prédéfinies pour autoriser ou bloquer les communications, protégeant ainsi les systèmes contre les accès non autorisés, les intrusions et les attaques malveillantes. [33].

II.9.2 Classification des pare-feux

La classification des pare-feux peut se faire selon plusieurs critères, notamment leur emplacement dans l'architecture réseau et leur mode d'implémentation. Cette distinction permet de mieux comprendre leur rôle et leur champ d'action dans la protection des systèmes informatiques, Voici la classification suivante, regroupée selon ces deux axes [34]:

1. **Selon l'emplacement du pare-feu**

- **Pare-feu basé sur l'hôte(Host-based firewall)**: logiciel installé directement sur une machine (poste de travail ou serveur) qui filtre tout le trafic entrant et sortant de ce seul hôte offrant une protection granulaire au niveau local.

- **Pare-feu réseau(Network-based firewall)**: dispositif (matériel ou virtuel) placé à la frontière entre deux réseaux (typiquement réseau interne versus Internet) et chargé de filtrer tout le trafic qui transite entre ces deux domaines, protégeant ainsi l'ensemble du réseau interne.

2. **Selon l'implémentation du pare-feu**

- **Pare-feu logiciel(Software firewall)**: implémentation logicielle du pare-feu, s'exécutant sur du matériel standard (serveur, VM ou poste) facilement déployable et configurable pour filtrer le trafic réseau selon des règles définies.

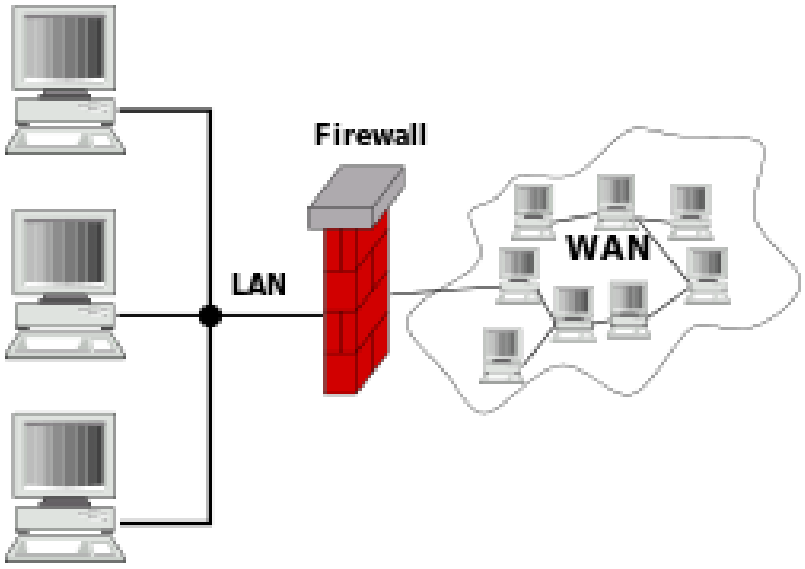
- **Pare-feu matériel (Hardware firewall)** : appliance dédiée, dotée de processeurs et de systèmes optimisés pour l'inspection et le filtrage du trafic réseau à haut débit, résistante aux contournements logiciels et généralement déployée en périphérie du réseau.

II.9.3 Fonctionnement général

Le pare-feu agit comme une barrière entre un réseau interne sécurisé (par exemple, un réseau d'entreprise ou domestique) et des réseaux externes non fiables, tels qu'Internet.(voir Figure II.6) Il filtre les flux de données en fonction de critères telles que [33]:

- L'adresse IP source et destination
- Les ports utilisés(TCP/UDP)
- Le protocole de communication
- Le contenu des paquets
- L'identité de l'utilisateur ou de l'application

En appliquant ces règles, le pare-feu empêche les communications non autorisées tout en permettant les échanges légitimes.



FigureII.6: Illustration de l'emplacement du pare-feu entre un réseau interne et Internet

II.9.4 Les trois zones de sécurités

Dans un pare-feu les zones de sécurité sont des interfaces logiques classées selon des niveaux de sécurité définis par l'utilisateur,plus une zone est considérée comme sécurisée plus les règles de trafic par défaut seront strictes pour les connexions entrantes depuis des zones externes (moins sécurisées) donc cette classification permet de contrôler le trafic autorisé entre les différentes zones du réseau.Comme illustré dans Figure II.7,on peut distiguer trois principales zones de sécurités.

1. LAN(Local Area Network): il s'agit de la zone interne représentant le réseau local de l'entreprise (ordinateurs, imprimantes, serveurs internes…). Elle est considérée comme totalement fiable d'où son niveau de sécurité élevé (dépend toujours du choix de l'utilisateur).

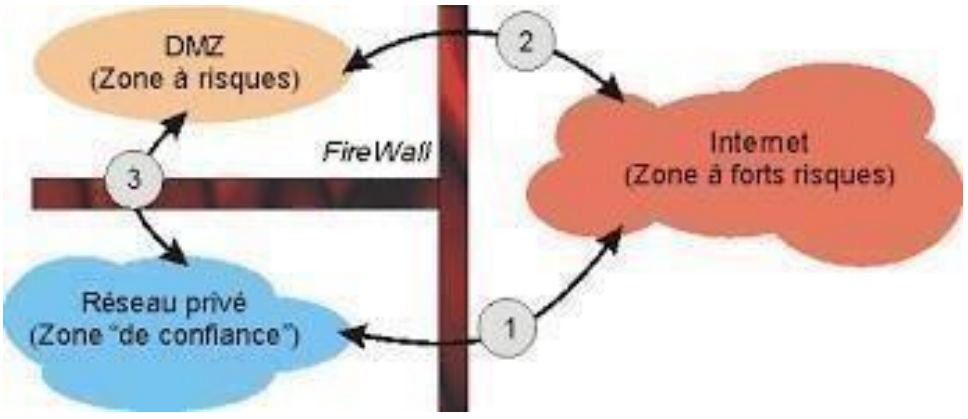
- ✓ Remarque:le trafic du LAN vers les autres zones est autorisé par défaut car le flux part d'une zone ayant un niveau de sécurité plus élevé vers une zone ayant un niveau de sécurité moins élevé.

2. WAN (WideArea Network): le WAN représente l'Internet ou tout réseau externe non fiable, C'est aussi la zone la moins sécurisée avec un niveau de sécurité très bas (dépend du choix de l'utilisateur).

- ✓ Remarque: le trafic du WAN vers les autres zones est bloqué par défaut sauf s'il est explicitement autorisé.

3. DMZ (Demilitarized Zone) : la DMZ est une zone tampon placée entre le LAN et le WAN, elle héberge généralement des serveurs accessibles depuis Internet (comme un serveur web ou mail) mais isolés du LAN pour limiter les risques en cas de piratage.

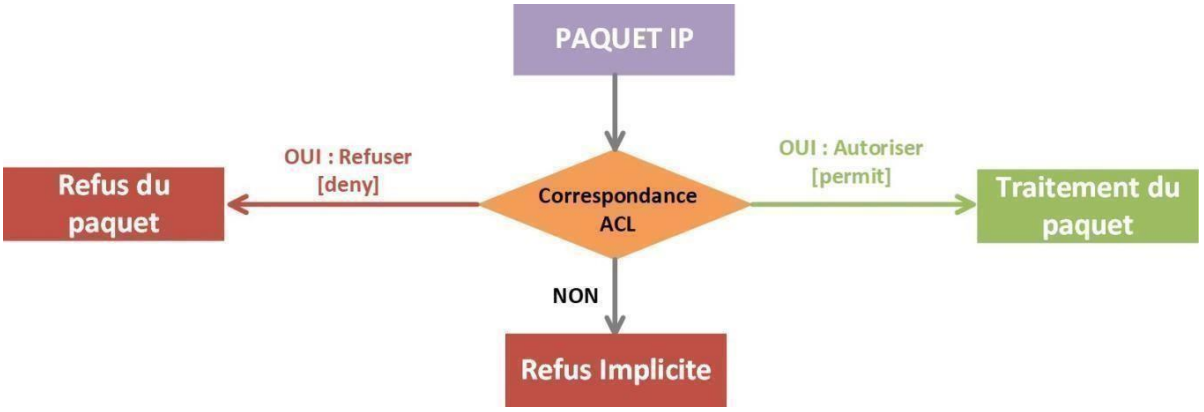
- ✓ Remarque: elle a un niveau de sécurité intermédiaire,souvent fixé à 50 (dépend du choix del'utilisateur),ce qui permet au trafic du LAN vers la DMZ, mais pas l'inverse sans règle explicite, comme ceci[35] :
- Trafic du réseau externe vers la DMZ régulé,
- Trafic du réseau externe vers le réseau interne interdit,
- Trafic du réseau interne vers la DMZ peut être régulé,
- Trafic du réseau interne vers le réseau externe autorisé(mais aucune réponse ne sera reçu, néanmoins).



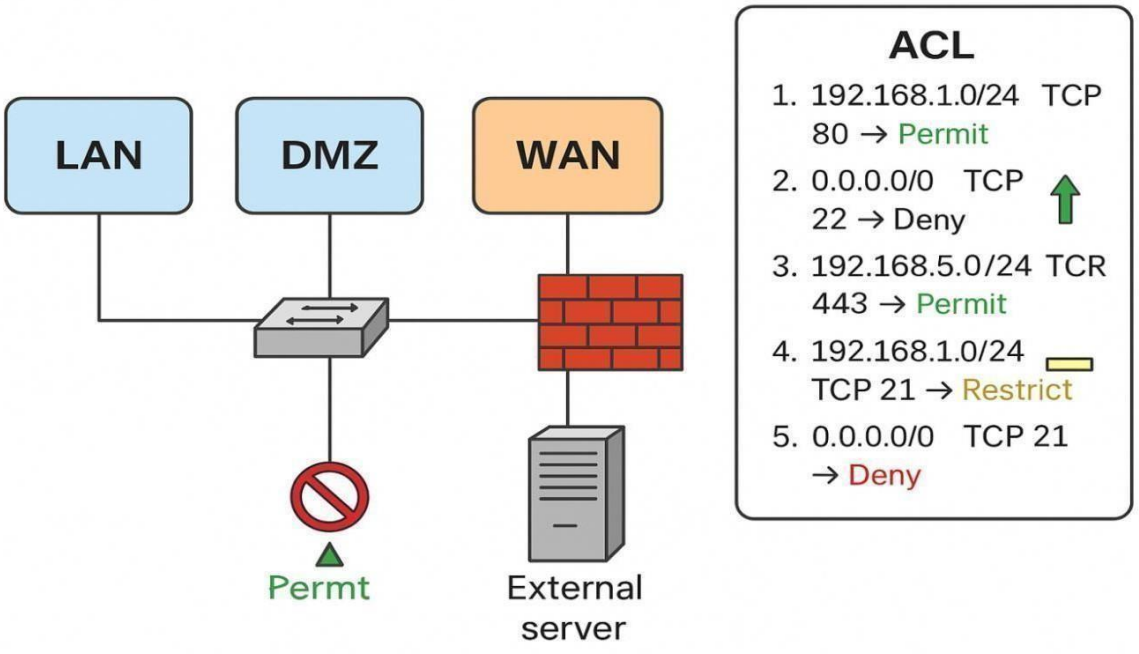
FigureII.7:Classification des zones réseau selon le niveau de confiance et de sécurité

II.10 Définition des ACL

Les ACL sont des mécanismes de sécurité utilisés pour filtrer le trafic réseau en fonction de critères définis. Elle permet (permit), restreint (restrict) ou refuse (deny) l'accès à certaines ressources du serveur web.Une ACL se base sur plusieurs critères tels que Adresse IP source et destination,Protocoles utilisés (ex. TCP, UDP,ICMP),Zone réseau(LAN,WAN,DMZ)pour classifier le trafic,Ports spécifiques(par exemple HTTP : port 80, HTTPS : port 443) .Le schéma général du fonctionnement d'une ACL est décrit par Figure II.8. Figure II.9 et Table II.2 illustrent des exemples d'application des ACL sur un Pare-feu dans un réseau.



FigureII.8: schéma ACL



FigureII.9: Contrôle d'accès réseau avec des ACL entre LAN,DMZ et WAN

II.10.1 Exemple de Liste de Contrôle d'Accès (ACL)

N°	SourceIP	Destination IP	Protocole	Port	Zone Source	Zone Destination	Action	Description
1	192.168.1.100	192.168.10.10	TCP	80	WAN	DMZ	Permit	Autoriser le trafic HTTP vers le serveur web
2	0.0.0.0/0	192.168.1.1	TCP	22	WAN	LAN	Deny	Bloquer les connexions SSH externes
3	192.168.5.0/24	192.168.10.20	TCP	443	LAN	DMZ	Permit	Autoriser HTTPS entre LAN et DMZ
4	192.168.2.50	192.168.1.1	ICMP	-	WAN	LAN	Restrict	Autoriser uniquement le ping, bloquerle reste
5	192.168.3.10	192.168.10.30	TCP	21	WAN	DMZ	Deny	Bloquer les connexions FTP externes

TableauII.2: Liste de Contrôle d'Accès(ACL)

II.11 Conclusion

Ce chapitre a permis de poser les bases théoriques essentielles à la compréhension des mécanismes de sécurisation des serveurs web en explorant les concepts clés liés à la sécurité informatique, ainsi que les principales techniques et protocoles de protection.Nous avons également mis en évidence l'importance cruciale de la prévention et de la détection des attaques. Ces éléments théoriques constituent un pilier fondamental pour la mise en œuvre pratique présentée dans le chapitre suivant, où nous aborderons le déploiement et la sécurisation d'un serveur web sous CentOS Linux.

ChapitreIII:déploiement et sécurisation d'un serveur web Apache sur Centos

III.1 Introduction

Ce chapitre est consacré à la mise en place des mécanismes nécessaires à la sécurisation d'un serveur web Apache ,déployé sur une machine virtuelle CentOS. Il expose de manière structurée les étapes techniques suivies pour garantir la protection du serveur, en s'appuyant sur des outils de sécurité spécifiques tels que le pare-feu.

L'ensemble des procédures présentées vise à renforcer la résilience du système face aux menaces potentielles, tout en garantissant sa stabilité et sa disponibilité.

III.2 Environnement de développement

Dans cette section nous présentons l'environnement logiciel utilisé pour le déploiement et la sécurisation de notre serveur web, le choix du système d'exploitation constitue une étape cruciale pour garantir la stabilité, la compatibilité et la sécurité de l'infrastructure.

III.2.1 Système d'exploitation Unix, distribution CentOS

CentOS(Community ENTerprise Operating System) est une distribution Linux open source, basée sur le code source de Red Hat Enterprise Linux (RHEL), elle est connue pour sa stabilité, sa sécurité et son compatibilité avec les environnements serveurs. Elle est largement utilisée dans le monde professionnel pour héberger des services critiques, tels que les serveurs web (Apache, Nginx), les bases de données (MariaDB, PostgreSQL) ou encore des services réseau.

Dans ce projet, nous avons utilisé CentOS Stream 9, qui est une version en développement continu servant de passerelle entre RHEL et les futures versions stables de CentOS. Elle offre les dernières mises à jour tout en conservant une base fiable,adaptée à un usage en environnement de test ou de pré-production.

Grâce à son excellent support des outils d'administration, à la longévité de ses versions et à ses dépôts de paquets fiables, CentOS constitue un choix idéal pour la gestion des services réseau et d'infrastructure (serveur web, serveur de messagerie, centre de données, etc.).Figure III.1 représente le logo de centOS.

FigureIII.1: Logo de centOS

III.2.2 Serveur web open source Apache HTTP Server

Dans le cadre de ce projet, le serveur web Apache a été utilisé comme composant principal pour la mise en place de l'environnement de travail. Figure III.2 ci-dessous illustre le logo de APACHE .

FigureIII.2: Logo de APACHE

Il s'agit d'un serveur HTTP libre et largement utilisé dans le domaine de l'hébergement web réputé pour sa robustesse et sa large adoption dans les systèmes basés sur Linux. Son rôle principal est de recevoir les requêtes des clients via le protocole HTTP ou HTTPS, et de leur fournir les ressources demandées, telles que les pages HTML ou les scripts dynamiques (PHP). Ce choix s'est imposé naturellement en raison de sa fiabilité et sa facilité d'intégration dans des environnements de développement sécurisés. Sa configuration est hautement personnalisable grâce à des fichiers de configuration spécifiques, offrant un contrôle précis sur le comportement du serveur.

De plus, Apache intègre des modules de sécurité permettant de sécuriser les échanges de données, notamment par la mise en œuvre du protocole SSL/TLS pour les connexions HTTPS.

III.2.3 Paquets de base pour les services web Apache

Nous allons installer plusieurs paquets essentiels qui permettront de configurer et de gérer efficacement notre serveur web Apache, à savoir:

a. Outils de base et utilitaires système

- **wget**:outil en ligne de commande permettant de télécharger des fichiers depuis Internet via http et HTTPS.
- **net-tools**: fournit des outils réseau classiques, tels queifconfig.
- **gcc**: compilateur C/C++ du projet GNU, nécessaire pour compiler des logiciels à partir du code source.

- **fuse-sshfs**: permet de monter un répertoire distant via SSH comme s'il était local, ce qui estpratique pour accéder à un serveur distant en utilisant le protocole SSH.

b. Paquets de gestion du temps et des mises à jour

- **chrony**: service de synchronisation de l'heure avec des serveurs NTP.
- **dnf-automatic**: permet d'automatiser les mises à jour logicielles viale gestionnaire de paquets DNF.

c. Paquets de sécurité et de gestion du pare-feu

- **iptables-services**: permet de gérer et de sauvegarder les règles du pare-feu iptables en tant que service.

d. Paquets de dépôts et sources de logiciels

- **epel-release**: ajoute le dépôt EPEL(Extra Packages for Enterprise Linux), qui fournit des paquets supplémentaires non inclus dans les dépôts officiels de CentOS.

e. Services système complémentaires

- **postfix** : agent de transfert de courrier (MTA) utilisé pour envoyer, recevoir et relayer des e-mails entre serveurs de messagerie. Il sert également à générer des notifications système par e-mail, ce qui est utile pour l'administration et la surveillance du serveur.

f. Langage serveur et modules PHP

- **php** : PHP est un langage de script côté serveur, principalement utilisé pour le développement web.
- **php-devel**: contient les fichiers de développement nécessaires pour compiler des extensions PHP ou pour développer des modules PHP personnalisés.
- **php-pear**: un système de gestion de paquets pour ajouter des fonctionnalités PHP, appelé PEAR (PHP Extension and Application Repository).
- **php-gd** : ajoute la prise en charge du traitement d'images dans les scripts PHP.
- **php-mbstring**: une extension PHP qui gère les chaînes de caractères multioctets (multibyte strings).

III.2.4 Les paquets de base pour la sécurisation du serveur web Apache

Afin de garantir la sécurité du serveur web Apache, il est essentiel d'installer des paquets supplémentaires permettant d'assurer principalement la confidentialité, l'authentification et l'intégrité des communications.
Dans ce cadre, cette section présente les paquets indispensables mis en place pour renforcer la résilience du serveur face aux menaces potentielles.

- **openssl**: fournit des outils et des bibliothèques puissantes pour la cryptographie

(RSA,EC, etc.) et la gestion des certificats.

- **mod_ssl**: module Apache(HTTPD)qui permet à notre serveur web d'utiliser SSL/TLS pour sécuriser les connexions HTTPS.
- **libssh2**: bibliothèque C permettant de réaliser des connexions SSH au sein d'applications.
- **libssh2-devel** : fichiers de développement nécessaires pour compiler ou intégrer libssh2 dans des projets.
- **libssh2-docs**: contient la documentation de la bibliothèque libssh2.

III.3 Présentation des niveaux de sécurité implémentés dans le serveur web Apache

La sécurisation de notre serveur web Apache repose sur la mise en place de plusieurs niveaux de protection complémentaires, couvrant à la fois l'accès système, la gestion des services, et la sécurisation des communications. Cette section détaille les principaux mécanismes de sécurité implémentés dans notre environnement, incluant :

- Gestion des utilisateurs et de leurs privilèges.
- Configuration d'un pare-feu avec iptables.
- Configuration sécurisée d'un environnement de développement web sous Apache.
- Mise en place d'une sécurité avancée du service SSH.
- Sécurisation par certificat SSL.

Nous détaillerons chaque niveau de sécurité dans les cinq prochaines sections (Section III.4 à Section III.8).

III.4 Gestion des utilisateurs et de leurs privilèges

III.4.1 Objectifs

- Empêcher les accès non autorisés aux ressources critiques du système.
- Favoriser une gestion optimale des droits d'accès grâce à une attribution précise des permissions.
- Appliquer le principe du moindre privilège en attribuant uniquement les droits nécessaires à chaque utilisateur.
- Permettre une supervision efficace des activités des utilisateurs à des fins de sécurité et d'audit.

III.4.2 Utilisateurs créés

Deux comptes utilisateurs ont été configurés sur le serveur afin de sécuriser l'accès. Tout d'abord, le compte root, présent par défaut sur les systèmes Linux, dispose des privilèges administratifs nécessaires pour effectuer les configurations critiques du système. Ensuite, un compte utilisateur standard nommé telecom*s* a été créé ; ce dernier possède des droits limités, ce qui permet de réduire les risques liés aux erreurs humaines. Deux autres comptes de test,testuser et testuser1, ont également été ajoutés pour valider les restrictions SSH : le premier a permis de vérifier que l'accès sans clé est refusé et le second a servi à tester le blocage automatique après plusieurs tentatives échouées.

III.4.3 Les privilèges attribués pour chaque utilisateurs

L'attribution des privilèges est un mécanisme fondamental pour garantir la sécurité et la stabilité de l'environnement, elle permet de contrôler précisément les actions qu'un utilisateur est autorisé à effectuer en fonction de son rôle en évitant de donner un accès total à tous les utilisateurs, on réduit considérablement les risques liés aux erreurs de manipulation ou aux actions malveillantes.

- ✓ **Pour root:**

  - Accès complet au système y compris aux fichiers sensibles et aux configurations critiques.
  - Création, suppression et gestion des utilisateurs, des groupes et des droits d'accès.
  - Installation, suppression et mise à jour des paquets et services système.

- ✓ **Pour telecoms:**

  - Exécution de commandes administratives spécifiques via *sudo*.
  - Accès limité aux fichiers système, uniquement si nécessaire à ses tâches.
  - Réalisation des configurations et manipulations courantes sans compromettre l'ensemble du système.

III.4.4 Désactivation de l'accès SSH direct au compte root

Pour renforcer la sécurité du serveur, nous avons désactivé l'accès SSH direct pour le compte root. Ce dernier possède tous les privilèges sur le système et représente une cible de privilégiée pour les attaques automatisées, notamment les tentatives de connexion par force brute. En interdisant son accès distant, on ajoute une barrière de sécurité supplémentaire, obligeant ainsi les utilisateurs distants à se connecter d'abord via un compte standard, puis à utiliser la commande sudo pour effectuer des actions administratives.

Cette mesure a été appliquée en modifiant le fichier de configuration SSH /etc/ssh/sshd_config, en y ajoutant ou en ajustant la directive PermitRootLogin no. Ainsi, toute connexion SSH directe avec le compte root est bloquée réduisant considérablement les risques d'intrusion et contribuant à une meilleure protection du système.

III.5 Configuration d'un pare-feu avec iptables

Dans le cadre de notre projet, nous avons utilisé iptables, un outil puissant et flexible intégré au système Linux, qui permet de définir, gérer et appliquer des règles précises de filtrage du trafic réseau. Cet outil joue un rôle central dans la mise en place des politiques de sécurité du pare-feu, en contrôlant les connexions entrantes et sortantes pour protéger le serveur contre les accès non autorisés et les attaques potentielles.

III.5.1 Objectifs

- Contrôler le trafic réseau en autorisant uniquement les connexions nécessaires au bon fonctionnement du serveur.

- Réduire la surface d'attaque du serveur en bloquant par défaut tout le trafic entrant non explicitement autorisé.

III.5.2 Configuration des politiques de sécurités sur pare-feu IPtables

Avec iptables, nous avons implémenté des listes de contrôle d'accès (ACL) permettant de:

- Configurer des règles pour autoriser uniquement les ports nécessaires (HTTP, HTTPS, DNS, NTP, SMTP), en incluant le trafic loopback pour les communications internes, et bloquer le reste dans le but de sécuriser l'accès au serveur.

- Restreindre les connexions SSH à un maximum de 4 tentatives par minute pour limiter les risques de brute force.

- Permettre un filtrage intelligent du trafic en autorisant uniquement les connexions nouvelles, établies ou liées, renforçant la sécurité sans bloquer les échanges légitimes.

III.5.3 Implémentation des ACL sur IPtables

Cette sous-section présente en détail les différentes règles de contrôle d'accès (ACL) mises en œuvre à l'aide d'iptables, dans le but de renforcer la sécurité du serveur en filtrant précisément le trafic réseau autorisé.Ces règles ont été appliquées sur l'interface ens160, qui correspond à l'interface réseau du serveur reliée au WAN, c'est-à-dire le réseau externe ou Internet.

ACL1 :autorisation des connexions entrantes HTTP (port 80) et HTTPS (port 443) sur l'interface réseau ens160, mais uniquement si elles sont : nouvelles (NEW), i.e. un client essaie de se connecter, ou déjà établies (ESTABLISHED), une communication est déjà en cours.

Les commandes permettant l'implémentation de cette ACL :

```
iptables -A INPUT -i ens160 -p tcp --dport 80 -m state --state
NEW,ESTABLISHED -j ACCEPT

iptables -A INPUT -i ens160 -p tcp --dport 443 -m state --state
NEW,ESTABLISHED -j ACCEPT
```

ACL2: autorisation des paquets sortants via l'interface réseau (ens160 dans notre cas), pour le protocole TCP, sur le port de destination 80 (HTTP) et le port de destination 443 (HTTPS), mais uniquement si la connexion est déjà établie (--state ESTABLISHED).

Les commandes permettant l'implémentation de cette ACL :

```
iptables -A OUTPUT -o ens160 -p tcp --dport 80 -m state --state
ESTABLISHED -j ACCEPT

Iptables -A OUTPUT -o ens160 -p tcp --dport 443 -m state --state
ESTABLISHED -j ACCEPT
```

ACL 3 : autorisation des paquets UDP entrants sur l'interface ens160, destinés au port 53 (DNS), uniquement s'ils font partie d'une connexion déjà établie.

La commande permettant l'implémentation de cette ACL :

```
iptables -A INPUT -i ens160 -p udp --dport 53 -m state --state
ESTABLISHED -j ACCEPT
```

ACL 4 :autorisation du serveur à envoyer des requêtes DNS UDP sortantes vers les serveurs DNS (port 53) sur l'interface ens160, ainsi que les paquets faisant partie de connexions DNS déjà établies.

La commande permettant l'implémentation de cette ACL :

```
iptables -A OUTPUT -o ens160 -p udp --dport 53 -m state --state
NEW,ESTABLISHED -j ACCEPT
```

ACL 5 :autorisation des paquets UDP entrants arrivant sur l'interface ens160 et destinés au port 123, mais uniquement s'ils font partie d'une connexion déjà établie.

La commande permettant l'implémentation de cette ACL :

```
iptables -A INPUT -i ens160 -p udp --dport 123 -m state --state
ESTABLISHED -j ACCEPT
```

ACL 6 :autorisation du serveur à envoyer des paquets UDP vers le port 123 (NTP) sur l'interface ens160, pour synchroniser l'heure via le protocole NTP.

La commande permettant l'implémentation de cette ACL :

```
Iptables -A OUTPUT -o ens160 -p udp --dport 123 -m state --
state NEW,ESTABLISHED -j ACCEPT
```

ACL 7 : autorisation de tout le trafic entrant provenant de l'interface locale (lo), c'est-à-dire toutes les communications internes au serveur (exemple : un programme qui communique avec un autre via localhost).

La commande permettant l'implémentation de cette ACL :

```
Iptables -A INPUT -i lo- j ACCEPT
```

ACL 8 :autorisation de tout le trafic sortant sur l'interface loopback, c'est-à-dire toutes les communications internes au serveur vers lui-même(localhost).

La commande permettant l'implémentation de cette ACL :

```
Iptables -A OUTPUT -o lo -j ACCEPT
```

ACL 9 : autorisation des paquets TCP entrants sur le port 80 (HTTP) qui font partie d'une connexion déjà établie ou sont liés à une connexion existante.

La commande permettant l'implémentation de cette ACL :

```
iptables -A INPUT -i ens160 -p tcp --dport 80 -m state --state
```

```
ESTABLISHED,RELATED -j ACCEPT
```

ACL 10 :autorisation des paquets TCP entrants sur le port 443 (HTTPS) qui font partie d'une connexion déjà établie ou sont liés à une connexion existante.

La commande permettant l'implémentation de cette ACL :

```
iptables -A INPUT -i ens160 -p tcp --dport 443 -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

ACL 11 :autorisation de l'envoi des paquets TCP sortants vers le port 443 (HTTPS), pour : démarrer de nouvelles connexions HTTPS, continuer des connexions déjà établies, ou gérer des paquets liés à ces connexions.

La commande permettant l'implémentation de cette ACL :

```
Iptables -A OUTPUT -o ens160 -p tcp --dport 443 -mstate --state
NEW,ESTABLISHED,RELATED
```

ACL 12 :autorisation de l'envoi des paquets TCP sortants vers le port 80 (HTTP), pour : démarrer de nouvelles connexions HTTP, continuer des connexions déjà établies, ou gérer des paquets liés à ces connexions.

La commande permettant l'implémentation de cette ACL :

```
iptables -A OUTPUT -o ens160 -p tcp --dport 80 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
```

ACL 13 :autorisation des paquets TCP entrants sur le port 25 (SMTP) qui font partie d'une connexion déjà établie, en passant par l'interface réseau ens160.

La commande permettant l'implémentation de cette ACL :

```
iptables -A INPUT -i ens160 -p tcp --dport 25 -m state --state
ESTABLISHED -j ACCEPT
```

ACL 14 : autorisation du serveur à envoyer des e-mails via SMTP (port 25) sur l'interface réseau ens160, en établissant de nouvelles connexions ou en continuant celles déjà établies.

La commande permettant l'implémentation de cette ACL :

```
iptables -A OUTPUT -o ens160 -p tcp --dport 25 -m state --state
NEW,ESTABLISHED -j ACCEPT
```

III.6 Configuration sécurisée d'un environnement de développement web sous Apache

La configuration sécurisée d'un environnement de développement web constitue une étape essentielle pour garantir la fiabilité, la confidentialité et l'intégrité des données échangées entre le serveur et ses utilisateurs. Dans cette sous-section, nous détaillons les principales configurations mises en œuvre pour préparer un environnement de développement web sécurisé pour hébergement des applications et site web.

III.6.1 Création d'une base de données et des utilisateurs

La création et la sécurisation d'une base de données ont pour objectif de garantir la confidentialité, l'intégrité et la disponibilité des informations stockées.
Dans le cadre de ce projet une base de données MariaDB a été mise en place pour stocker les données du site WordPress une fois celui-ci créé. Un utilisateur dédié a été configuré avec un mot de passe sécurisé. Seuls les privilèges nécessaires lui ont été accordés. Cette séparation des droits permet de limiter l'impact d'une éventuelle faille ou erreur.
Cette démarche permet d'éviter l'exposition du compte administrateur de la base. Elle renforce également le contrôle des accès et des opérations autorisées, ce qui est essentiel pour garantir la sécurité des données manipulées par les applications web.

- Créé une base de données wordpress via MariaDB/MySQL en suivant les étapes ci-dessous :
  - ✓ Connexion à MariaDB en tant qu'administrateur, avec la commande suivante: mysql -u root –p
  - ✓ Création de la base de données nommée wordpress, avec la commande suivante: CREATE DATABASE wordpress;
  - ✓ Sortie de l'environnement MySQL, avec la commande suivante: EXIT;
- Création d'un utilisateur dédié wpuser avec un mot de passe telecoms en suivant les étapes ci-dessous:
  - ✓ Connexion à MariaDB, avec la commande suivante: mysql -u root –p
  - ✓ Création de l'utilisateur wpuser avec le mot de passe telecoms, avec la commande suivante: CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'telecoms';
  - ✓ Attribution de tous les privilèges à wpuser sur la base wordpress, avec la commande suivante: GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'localhost';
  - ✓ Rechargement des privilèges pour prise en compte, avec la commande suivante: FLUSH PRIVILEGES;
  - ✓ Quitter MariaDB, avec la commande suivante: EXIT;

III.6.2 Sécurisation de la Base de Données (Maria DB)

La sécurité de la base de données MariaDB a été renforcée à l'aide de l'utilitaire mysql_secure_installation, qui permet de renforcer la configuration initiale de sécurité du serveur Apache. Dans ce cadre, les configurations suivantes ont été attribuées :

- **Suppression des utilisateurs anonymes :** cette mesure empêche toute connexion non authentifiée à la base de données, réduisant ainsi les risques d'accès non autorisé :

- ✓ Lors de l'exécution de l'utilitaire, à la question « Remove anonymous users? », la réponse « y » a été choisie.

- **Désactivation des connexions root à distance :** en limitant l'accès au compte administrateur root à la machine locale, on empêche toute tentative de prise de contrôle à distance :
  - ✓ Lors de l'exécution, à la question « Disallow root login remotely? », la réponse « y » a été fournie.
- **Suppression de la base de données de test :** cette base, créée par défaut, est accessible à tous les utilisateurs. Sa suppression évite une utilisation non sécurisée dans un environnement de production :
  - ✓ À la question « Remove test database and access to it? », la réponse « y » a été validée.

III.6.3 Configuration d'Apache pour WordPress

Afin de permettre l'hébergement et le bon fonctionnement du site WordPress, le serveur web Apache doit être correctement configuré. Ceci est réalisé par l'installation des paquets nécessaires, incluant WordPress lui-même ainsi que ses dépendances. Ces composants assurent la compatibilité entre Apache, PHP et la base de données, tout en préparant l'environnement à accueillir le site.

1. Installation des paquets WordPress

Dans cette sous-section, nous décrivons les étapes d'installation de Wordpress.

- ➢ D'abord, WordPress doit être téléchargé, puis extrait son contenu à l'aide des commandes suivantes :
  - ✓ *curl -O https://wordpress.org/latest.tar.gz* (téléchargement)
  - ✓ *tar -xzf latest.tar.gz*extraction

Afin de permettre la lecture et la modification des fichiers WordPress :

- Les fichiers doivent être déplacés dans le répertoire /var/www/html, qui correspond à la racine web par défaut d'Apache.
- Les permissions nécessaires ont été attribuées afin de garantir le bon fonctionnement de WordPress tout en maintenant un niveau de sécurité adapté.

  Chmod 755 pour les dossiers (lecture/exécution),avec la commande suivante :*sudo find /var/www/html -type d -exec chmod 755 {} \.*

  Chmod 644 pour les fichiers (lecture seule),avec la commande suivante :*sudo find /var/www/html -type f -exec chmod 644 {} \ .*

2. Hébergement du site sur Apache

Le serveur a été configuré pour écouter toutes les adresses IP disponibles sur le port 80, ce qui permet d'héberger le site WordPress en accès local. Ensuite, un nom d'hôte a été défini pour le site à l'aide de la directive ServerName, avec la valeur localhost,afin de faciliter sa résolution en local. Enfin, WordPress a été autorisé à utiliser un fichier .htaccess pour gérer les liens personnalisés et activer des règles supplémentaires, cela a été rendu possible en définissant la directive AllowOverride All dans la configuration d'Apache, garantissant ainsi un bon fonctionnement du site.

III.7 Mise en place d'une sécurité avancée du service SSH

La sécurisation du service SSH est une étape essentielle pour protéger l'accès distant au serveur Apache, notamment contre les tentatives d'intrusion ou les attaques par force brute. En renforçant la sécurisation SSH, on limite les possibilité d'attaque tout en garantissant un accès sécurisé aux administrateurs du système.

Cette sous-section décrit les différentes mesures mises en œuvre pour renforcer la sécurité du service SSH, telles que la modification du port par défaut, la désactivation de l'authentification par mot de passe, la restriction des utilisateurs autorisés et la mise en place d'une authentification par clés.

III.7.1 Objectifs

- Sécuriser l'accès distant au serveur en utilisant un canal chiffré.
- Remplacer les mots de passe par une authentification plus fiable via clé publique/privée.
- Réduire les risques de connexion non autorisée en limitant l'accès aux utilisateurs possédant une clé privée valide.

III.7.2 Modification du port SSH par défaut

Changer le port SSH par défaut est une mesure de sécurité simple mais efficace,qui permet de réduire les tentatives de connexion non autorisées. En remplaçant le port 22 par un port moins courant, comme le 9922, on évite une grande partie des scans automatisés effectués par des outils de reconnaissance ou des scripts malveillants. Cette configuration s'effectue dans le fichier de configuration SSH, puis nécessite un redémarrage du service SSH pour être prise en compte.

III.7.3 Principe de la clé privée et de la clé publique

- **Clé publique :** c'est la clé que l'on copie sur le serveur et qui peut être diffusée sans risque. Lorsqu'un utilisateur tente de se connecter, le serveur l'utilise pour vérifier que l'utilisateur possède bien la clé privée correspondante.
- **Clé privée :** il s'agit de la clé confidentielle conservée sur la machine cliente, elle ne doit jamais être partagée. Elle permet de prouver l'identité de l'utilisateur lors de la connexion SSH. Si elle correspond à la clé publique présente sur le serveur, la connexion peut être autorisée sans mot de passe.

III.7.4 Génération de la paire de clé privée/publique

Pour sécuriser les connexions SSH, nous avons mis en place un système d'authentification basé sur des paires de clés cryptographiques. Deux paires de clés ont été générées l'une depuis le terminal Linux du serveur et l'autre depuis l'invite de commande Windows du client. Les deux clés publiques ont ensuite été copiées sur le serveur, dans le fichier authorized_keys de l'utilisateur telecoms, ce qui permet à cet utilisateur de se connecter depuis les deux machines. Lorsqu'une connexion SSH est initiée, le serveur utilise la clé publique correspondante pour vérifier la signature envoyée par le client. Cette signature est générée à partir de la clé privée conservée localement sur la machine d'origine. Ce mécanisme permet une authentification sécurisée sans mot de passe, reposant sur un échange cryptographique fiable.

III.7.5 Configuration d'authentification

La configuration de l'authentification SSH repose sur le renforcement de la sécurité des connexions au serveur. Une méthode efficace consiste à activer l'authentification par clé publique tout en désactivant l'authentification par mot de passe ainsi que le mécanisme de "ChallengeResponse".

L'activation de l'authentification par clé publique permet au serveur de n'accepter que les connexions provenant de clients possédant une clé privée correspondant à une clé publique autorisée. Cela repose sur un système de cryptographie asymétrique garantissant une connexion sécurisée et difficile à compromettre.

En parallèle la désactivation de l'authentification par mot de passe empêche tout accès par simple saisie, ce qui élimine les risques liés aux attaques par force brute ou au vol de mot de passe.

Enfin, la désactivation de la méthode ChallengeResponseAuthentication permet d'éviter que le serveur SSH pose des questions interactives à chaque tentative de connexion. Cela simplifie l'accès pour les utilisateurs configurés avec une clé SSH et renforce la sécurité en supprimant une méthode d'authentification inutile dans notre contexte.

Afin de permettre ces paramétrages, les lignes suivantes ont été ajoutées ou modifiées dans le fichier de configuration /etc/ssh/sshd_config :

- ✓ PubkeyAuthentication yes
- ✓ AuthorizedKeysFile .ssh/authorized_keys
- ✓ PasswordAuthentication no
- ✓ ChallengeResponseAuthentication no

Ces mesures combinées garantissent que seuls les utilisateurs explicitement autorisés et possédant la clé privée adéquate peuvent accéder au serveur, offrant ainsi un niveau de sécurité bien supérieur.

III.8 Sécurisation par certificat SSL

III.8.1 Objectifs

- ➢ Chiffrer les données échangées entre le client et le serveur pour éviter toute interception.
- ➢ Garantir l'identité du serveur grâce à un certificat numérique validé.

- Renforcer la confiance des utilisateurs en affichant le protocole HTTPS dans le navigateur.

III.8.2 Création de certificat (autorité)

Pour sécuriser les connexions entre le serveur web et les clients, un certificat SSL a été mis en place. Celui-ci permet de chiffrer les échanges et d'afficher le protocole HTTPS dans le navigateur, garantissant ainsi la confidentialité et l'authenticité des communications.

III.8.3 Déploiement d'un certificat SSL auto-signé et activation de HTTPS

Dans le cadre de notre projet, nous avons sécurisé l'accès au site web en mettant en place le protocole HTTPS à l'aide d'un certificat SSL auto-signé généré avec la commande openssl. Cette opération a permis de créer deux éléments essentiels :

- Une clé privée stockée dans le fichier sécurisé /etc/ssl/private/apache-selfsigned.key. Cette clé joue un rôle central dans la sécurité de la connexion HTTPS car elle permet au serveur de déchiffrer les données envoyées par le client et de prouver son identité.
- Un certificat auto-signé enregistré dans /etc/ssl/certs/apache-selfsigned.crt. Ce certificat contient des informations comme la clé publique ainsi que des données d'identification (pays, région, nom d'hôte, etc.).

Ensuite, nous avons configuré le fichier /etc/httpd/conf.d/ssl.conf afin d'activer le support du protocole HTTPS dans Apache.Ce fichier contient les directives suivantes :

- SSLEngine on : permet d'activer la couche SSL/TLS dans Apache.
- SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt : indique le chemin du certificat SSL.
- SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key : indique le chemin de la clé privée.
- Le serveur est configuré pour écouter les connexions sécurisées sur le port 443 à travers le bloc <VirtualHost _default_:443>.

Nous avons également modifié le nom d'hôte du serveur en webserver local dans le fichier de configuration /etc/httpd/conf.d/ssl.conf en spécifiant cette valeur dans la directive Server Name cette modification permet une identification plus claire et cohérente du serveur lors des connexions HTTPS. Ensuite afin d'autoriser les connexions entrantes via HTTPS le port 443 a été ouvert dans le pare-feu avec la commande :

- *firewall-cmd --add-port=443/tcp –permanent*

suivie du rechargement du pare-feu :

- *firewall-cmd –reload*

Grâce à cette configuration le site web est désormais accessible de manière sécurisée via HTTPS ce qui garantit la confidentialité et l'intégrité des échanges entre le client et le serveur.

III.8.4 Configuration de la redirection HTTPS sous Apache

Cette redirection a été configurée dans le fichier /etc/httpd/conf.d/wordpress.conf à l'aide de la directive Redirect permanent.

Figure III.3 : Redirection de HTTP vers HTTPS

III.9 Conclusion

La sécurisation d'un serveur web ne se limite pas à son installation ; elle exige une série de mesures cohérentes et complémentaires pour garantir sa robustesse face aux menaces. À travers ce chapitre, l'objectif principal était de limiter les risques d'intrusion tout en garantissant un accès fiable et contrôlé aux services essentiels d'un serveur web Apache. En combinant des mécanismes tels que l'authentification par clé SSH, le filtrage réseau avec iptables, la gestion des privilèges et la mise en place du protocole HTTPS, nous avons réussi à construire un environnement sécurisé et fonctionnel.

Cette démarche proactive permet d'assurer la confidentialité, l'intégrité et la disponibilité des services web tout en respectant les standards de sécurité actuels.

ChapitreIV:Test et validation de la sécurisation du serveur web Apache

IV.1 Introduction

Dans le cadre de ce projet, la sécurisation d'un serveur web repose non seulement sur la configuration initiale, mais également sur la vérification systématique de l'efficacité des mesures mises en œuvre. Ce chapitre a pour objectif de détailler la mise en place d'une série de tests visant à valider la sécurité du serveur web Apache installé sous CentOS.

Ces tests ont pour objectif de s'assurer que les différentes couches de protection mises en place fonctionnent comme prévu. Ils visent plus précisément à vérifier que le pare-feu (iptables) est correctement configuré pour bloquer les connexions non autorisées tout en autorisant uniquement le trafic légitime ; que l'accès SSH est restreint et sécurisé afin de limiter les risques d'intrusion ; et que le protocole SSL (via HTTPS) est bien activé, avec un certificat auto-signé valide garantissant la confidentialité et l'intégrité des échanges entre le client et le serveur.

À travers cette démarche nous nous assurerons que le serveur respecte les bonnes pratiques en matière de sécurité et qu'il est capable de résister aux menaces les plus courantes.

IV.2 Environnement de test

Un environnement de test contrôlé a été mis en place afin de procéder à la vérification des différentes mesures de sécurité mises en œuvre sur le serveur web Apache. Cet environnement permet de simuler des interactions entre un client et le serveur, afin d'observer le comportement du système face aux diverses tentatives d'accès, qu'elles soient autorisées ou non.

1. **Coté serveur :** le serveur utilisé pour cette phase de test est une machine virtuelle exécutant le système d'exploitation CentOS Stream 9. Cette machine héberge le service Apache HTTP Server, un serveur web open source développé par la Apache Software Foundation. Apache HTTP Serverpermet de diffuser des contenus web via les protocoles HTTP et HTTPS en assurant la gestion des requêtes et des réponses entre les clients (navigateurs) et le serveur. Ce serveur web est largement utilisé dans le domaine de l'hébergement web en raison de sa stabilité, sa flexibilité et sa compatibilité avec les systèmes Unix/Linux. Il permet notamment la gestion des serveurs virtuels (virtual hosts), le traitement de scripts via des modules tels que mod_php (pour PHP), ainsi qu'une configuration fine à travers des fichiers tels que httpd.conf et .htaccess.

2. **Coté client :** du côté client, les tests ont été réalisés à partir d'une machine physique exécutant le système d'exploitation Windows. Celle-ci est équipée de divers outils d'analyse et la vérification des services réseau notamment Nmap, un utilitaire open source permettant l'exploration du réseau et l'audit de sécurité.

IV.3 Mise en œuvre des tests de validation de la sécurité

IV.3.1 Test du pare feu IPTables

L'objectif de ce test est de vérifier que seules les connexions à destination des ports explicitement autorisés sont acceptées par le pare-feu. En l'occurrence, les ports concernés sont :9922(SSH, port personnalisé), 80(HTTP), 443(HTTPS) et 53(DNS).

Depuis une machine cliente (machine physique sous Windows), la commande *nmap -Pn adresse_Ip_ens160* a été lancée afin d'afficher les ports autorisés. Le résultat de la commande est illustré dans Figure IV.1.

```
C:\Users\user>nmap -Pn 192.168.1.131
Starting Nmap 7.80 ( https://nmap.org ) at 2025-06-17 16:44 Afr. centrale Ouest
Nmap scan report for 192.168.1.131
Host is up (0.0049s latency).
Not shown: 995 filtered ports
PORT     STATE  SERVICE
22/tcp   closed ssh
80/tcp   open   http
443/tcp  open   https
3306/tcp open   mysql
9090/tcp open   zeus-admin
MAC Address: 00:0C:29:BF:88:DB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 8.42 seconds

C:\Users\user>
```

Figure IV.1 : Résultat du scan Nmap.

IV.3.2 Scan explicite du port DNS en TCP

Figure IV.2 ci-dessous présente le résultat du scan explicite du port 53 en TCP à l'aide de l'outil Nmap.

```
Administrateur : Invite de commandes
Microsoft Windows [version 10.0.19045.5965]
(c) Microsoft Corporation. Tous droits réservés.

C:\WINDOWS\system32>nmap -Pn -p 53 192.168.1.131
Starting Nmap 7.80 ( https://nmap.org ) at 2025-06-28 18:25 Afr. centrale Ouest
Nmap scan report for webserver.local (192.168.1.131)
Host is up (0.00088s latency).

PORT   STATE SERVICE
53/tcp open  domain
MAC Address: 00:0C:29:BF:88:DB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.80 seconds

C:\WINDOWS\system32>
```

Figure IV.2 : Résultat du scan Nmap – Port 53/TCP (DNS) en état open

IV.3.3 Scan explicite du port DNS en UDP

Figure IV.3 ci-dessous présente le résultat du scan explicite du port 53 en UDP à l'aide de l'outil Nmap.

```
C:\WINDOWS\system32>nmap -sU -p 53 192.168.1.131
Starting Nmap 7.80 ( https://nmap.org ) at 2025-06-28 18:27 Afr. centrale Ouest
Nmap scan report for webserver.local (192.168.1.131)
Host is up (0.0020s latency).

PORT   STATE SERVICE
53/udp open  domain
MAC Address: 00:0C:29:BF:88:DB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.79 seconds

C:\WINDOWS\system32>
```

Figure IV.3:Résultat du scan Nmap – Port 53/UDP (DNS) en état open

Remarque :Comme nous avons modifié le port SSH par défaut (22) en un port personnalisé (9922), il est nécessaire de spécifier ce port explicitement lors de la connexion. En effet, par défaut, l'invite de commende Windows (CMD) utilise automatiquement le port 22 pour SSH. C'est pourquoi il faut écrire la commande complète :
ssh telecoms@IP_du_serveur -p 9922
afin que le système sache qu'il doit utiliser le port 9922 au lieu du port par défaut.

Le résultat renvoyé par cette commande est illustré par Figure IV.4.

```
C:\Users\user>nmap -Pn -p 9922 192.168.1.131
Starting Nmap 7.80 ( https://nmap.org ) at 2025-06-17 16:28 Afr. centrale Ouest
Nmap scan report for 192.168.1.131
Host is up (0.0010s latency).

PORT     STATE SERVICE
9922/tcp open  unknown
MAC Address: 00:0C:29:BF:88:DB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.99 seconds
```

Figure IV.4 : Connexion SSH avec un port personnalisé.

IV.3.4 Comparaison des résultats avec la configuration du pare-feu sur le serveur

1. Avec la commande iptables -L on peut consulter les règles de filtrage actuellement appliquées par le pare-feu sur le serveur. Le résultat de la commande est illustré dans la Figure IV.5.

```
[root@localhost ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http state NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https state NEW,ESTABLISHED
ACCEPT     udp  --  anywhere             anywhere             udp dpt:domain state ESTABLISHED
ACCEPT     udp  --  anywhere             anywhere             udp dpt:ntp state ESTABLISHED
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:smtp state ESTABLISHED
           tcp  --  anywhere             anywhere             tcp dpt:9922 state NEW recent: SET name: SSH side: source ma
sk: 255.255.255.255
DROP       tcp  --  anywhere             anywhere             tcp dpt:9922 state NEW recent: UPDATE seconds: 60 hit_count:
 4 TTL-Match name: SSH side: source mask: 255.255.255.255
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:9922 state NEW,ESTABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http state ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https state ESTABLISHED
```

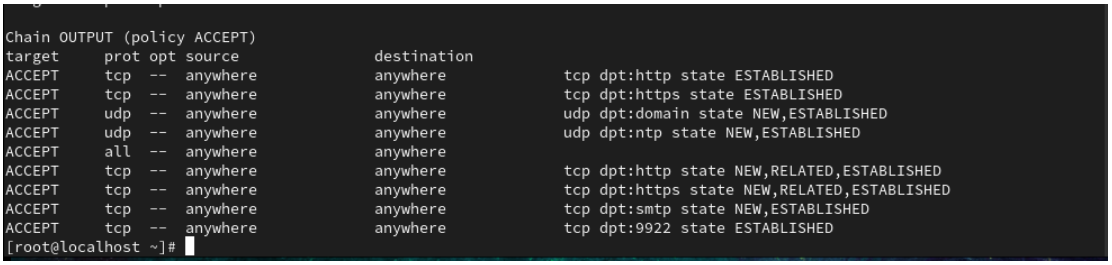


Figure IV.5 : Résultats des règles IPTables.

La comparaison des résultats du scan nmap avec la sortie de la commande iptables -L permet de confirmer que le pare-feu fonctionne comme prévu et qu'aucun port non autorisé n'est accessible depuis l'extérieur. Cela indique que les règles de filtrage ont bien été appliquées et que la configuration du pare-feu assure une restriction efficace du trafic réseau.

IV.4 Vérification des restrictions d'accès SSH

L'objectif de ce test est d'assurer que l'accès SSH au serveur est correctement restreint afin de prévenir les tentatives d'intrusion. Deux principales mesures de sécurité doivent être vérifiées : l'authentification par clé SSH et la désactivation de la connexion directe avec l'utilisateur root. Depuis la machine cliente, plusieurs tentatives de connexion SSH on étaient effectuée afin de tester ces restrictions.

1. **Connexion SSH sans clé depuis l'invite de commande:**cette tentative vise à vérifier que l'accès SSH est refusé pour tout utilisateur ne disposant pas d'une clé d'authentification valide. La commande utilisée est :
ssh -p 9922 testuser@adresse_ip_du_serveur
Comme attendu, la connexion a été refusée, confirmant que seule l'authentification par clé est autorisée. Le résultat de la commande est illustré dans la Figure IV.6.

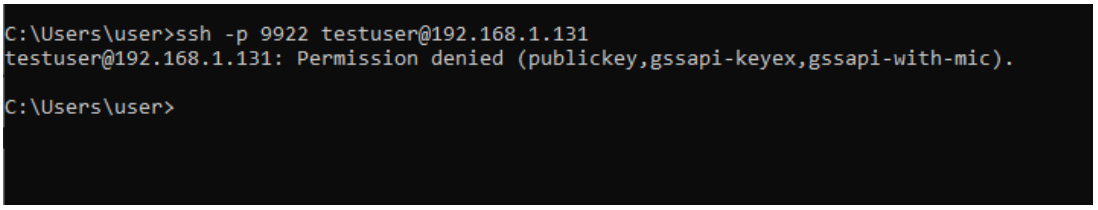


Figure IV.6 : Connexion SSH échouée par un utilisateur sans clé d'authentification.

2. **Connexion en tant que root :**une tentative de connexion SSH avec l'utilisateur root a été effectuée afin de vérifier si l'accès direct à ce compte est autorisé. Pour cela, la commande suivante a été utilisée :*ssh -p 9922 root@adresse_ip_du_serveur.*

le serveur a refusé l'accès cela qui confirme que la connexion directe en tant que root est bien désactivée pour renforcer la sécurité. Le résultat de cette tentative est illustré dans la Figure IV.7.

Figure IV.7 : Connexion SSH refusée pour l'utilisateur root (accès direct désactivé).

3. **Connexion avec un utilisateur autorisé :**La connexion SSH avec un utilisateur autorisé qui dispose une clé privée a été testée, la commande utilisée est : *ssh -p 9922 telecoms@adresse_ip_du_serveur.*

   On voit bien que l'accès a été autorisé ce qui confirme que le serveur accepte bien les connexions sécurisées uniquement pour les utilisateurs configurés avec authentification par clé.Le résultat est illustré dans Figure IV.8.

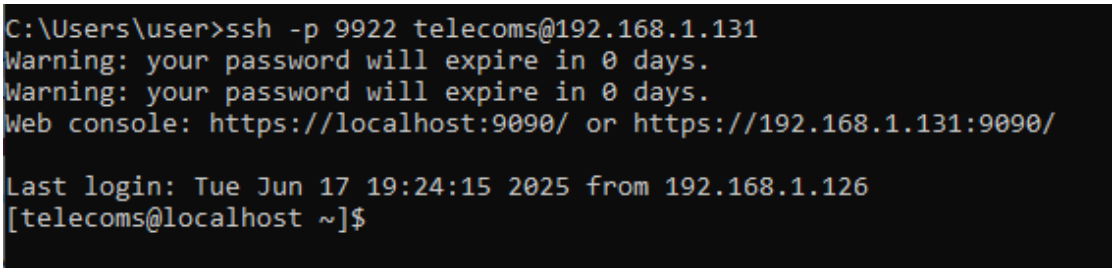


Figure IV.8 : Connexion SSH réussie avec un utilisateur autorisé.

Les résultats observés confirment que la sécurité SSH est bien mise place et l'accès est refusé pour tout utilisateur sans clé SSH ainsi que pour le compte root. La connexion est uniquement autorisée aux utilisateurs disposant des droits nécessaires et d'une clé valide. Cette configuration permet de réduire significativement les risques d'accès non autorisés au serveur.

IV.4.1 Blocage d'un utilisateur après plusieurs tentatives SSH

Dans le cadre de la sécurisation du service SSH, une règle iptables a été mise en place afin de bloquer automatiquement toute adresse IP tentant de se connecter à plusieurs reprises sans succès sur le port 9922.

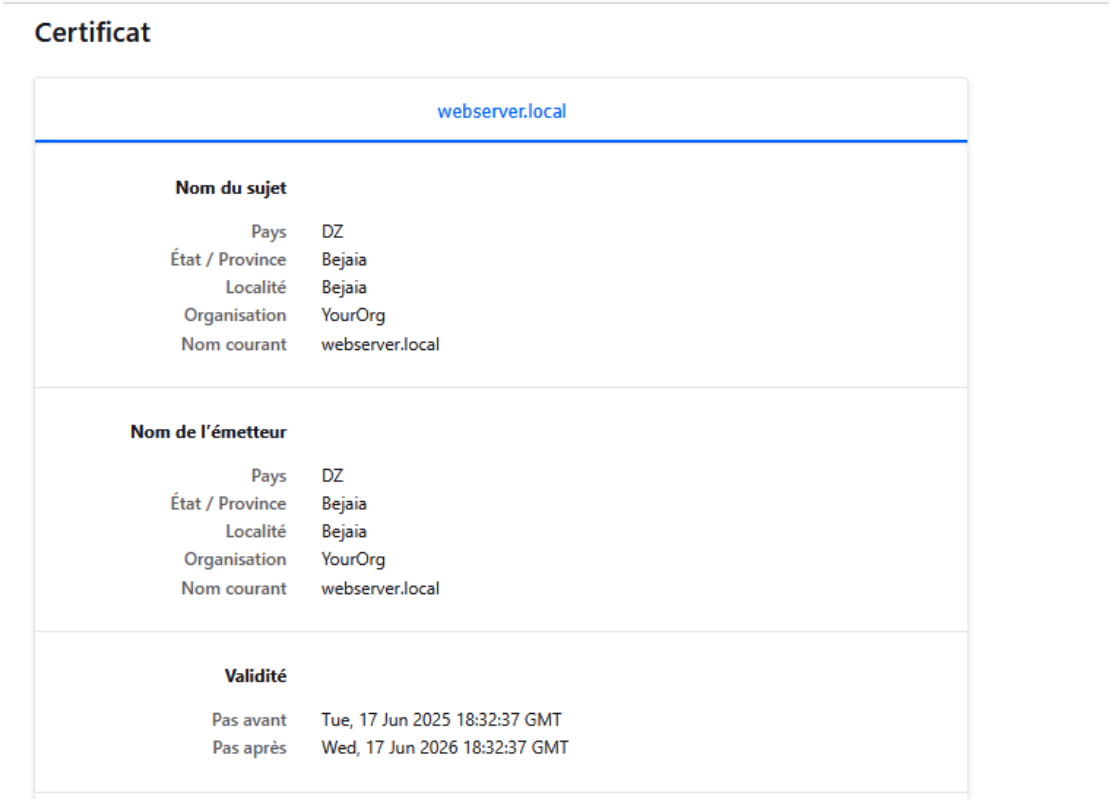
Figure IV.9 montre un exemple concret dans lequel l'utilisateur testuser1 tente de se connecter trois fois avec un mot de passe incorrect. À la quatrième tentative, la connexion est refusée.

Figure IV.9 : Blocage automatique d'une IP après plusieurs tentatives SSH échouées.

Cette mesure permet de protéger le serveur contre les attaques par force brute, en limitant le nombre de tentatives de connexion autorisées sur une courte période.

IV.5 Test de la sécurisation HTTPS du serveur Apache

L'objectif de ce test est de s'assurer que le protocole de chiffrement SSL/TLS est bien activé sur le serveur Apache et que le certificat auto-signé est fonctionnel. Cela garantit la sécurisation des échanges entre le client et le serveur web.
Pour effectuer cette vérification, on accède au serveur Apache via un navigateur web en utilisant le protocole HTTPS.
Le certificat étant correctement configuré (voir Figure IV.10), la connexion s'établit de manière sécurisée, avec affichage d'un avertissement. Cette étape est essentielle pour confirmer que le serveur prend bien en charge les connexions chiffrées via TLS.



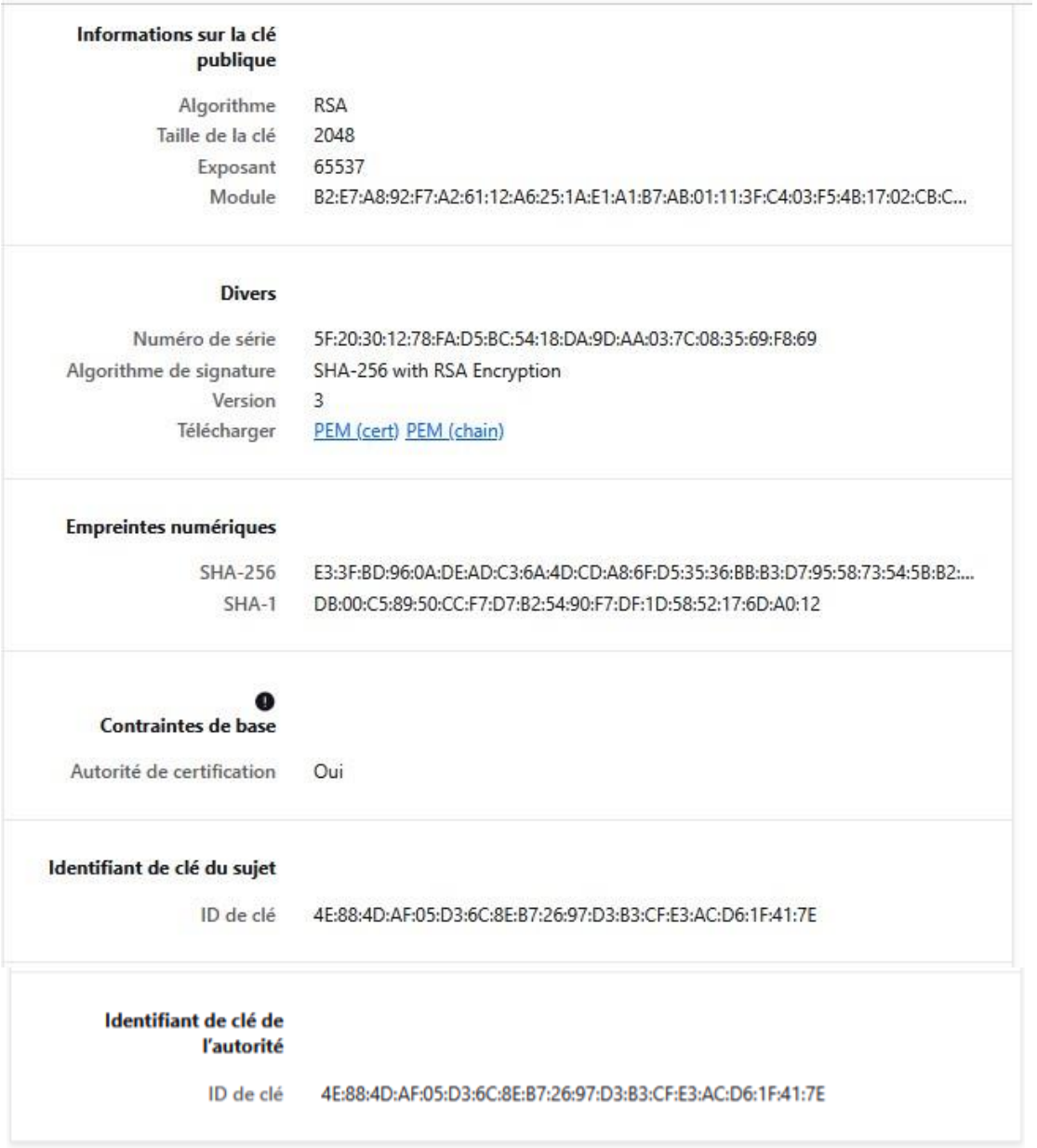


Figure IV.10 : Certificat SSL auto-signé

IV.5.1 Vérification manuelle du certificat SSL avec OpenSSL

Une vérification du certificat SSL a été effectuée en ligne de commande à l'aide de la commande openssl s_client -connect 192.168.1.131:443, cette commande a été lancée par l'utilisateur telecoms, qui se comporte comme une machine cliente et qui a permis d'afficher les détails suivants:

- Le serveur écoute bien sur le port 443.
- Le certificat SSL est fonctionnel.
- Le chiffrement TLS est bien activé.
- Le message d'avertissement self-signed certificate est normale dans ce cas, car le certificat que le serveur nous a envoyé est auto-signé (et non délivré par une autorité de certification reconnue).

IV.5.2. Test de redirection HTTPS sous Apache

La redirection de HTTP vers HTTPS a été testée avec succès en accédant à l'adresse http://adresse_serveur/ depuis un navigateur, la requête a été automatiquement redirigée vers https://adresse_serveur/,confirmant que la redirection est bien active.

IV.6 Conclusion

En conclusion les différents tests réalisés ont permis de confirmer la robustesse des mécanismes de sécurité mis en place sur le serveur. L'environnement est désormais en mesure de garantir un accès restreint, des communications sécurisées et une résistance face aux tentatives d'intrusion. Ces validations assurent que le serveur peut fonctionner de manière fiable tout en respectant les exigences essentielles en matière de sécurité.

Conclusion Générale

Conclusion générale

Dans un contexte numérique en perpétuelle évolution, la sécurisation des serveurs web représente un enjeu stratégique pour les entreprises soucieuses de protéger leurs données sensibles et de garantir la fiabilité de leurs services. Ce projet de fin d'études s'inscrit dans cette perspective en adoptant une approche globale,combinant cadre théorique, mise en œuvre technique et validation pratique.Il nous a permis d'explorer les fondamentaux des réseaux informatiques ainsi que les principaux mécanismes de sécurité, tels que les protocoles SSH et TLS, les pare-feux,et les certificats numériques.

Pour mettre en place notre solution, nous avons utilisé une machine virtuelle crée avec VMware, sur laquelle nous avons installé CentOS stream 9 afin de simuler un environnement serveur réaliste. Notre démarche s'est articulée autour du déploiement et de la sécurisation d'un serveur web Apache, en y intégrant plusieurs outils et protocoles essentiels à la protection du système. Cela inclut la configuration des utilisateurs avec gestion des privilèges, la désactivation de l'accès root via SSH, la mise en place d'un pare-feu avec iptables, l'utilisation des ACL pour le filtrage réseau, la sécurisation d'une base de données MariaDB, ainsi que l'installation de WordPress pour la gestion du site web. L'ajout du protocole HTTPS àl'aide d'un certificat SSL auto-signé a permis de renforcer la confidentialité des échanges.

Toutes ces implémentations ont été documentées et testées, illustrant la manière dont les concepts théoriques ont été traduits en solutions concrètes dans notre environnement. À l'issue de la configuration, plusieurs vérifications ont été menées afin de valider la robustesse du serveur face aux menaces courantes, tout en garantissant son bon fonctionnement et la continuité des services web qu'il fournit.

En conclusion, ce projet met en exergue l'importance d'une démarche structurée pour garantir la sécurité d'un serveur web, offrant une base solide à toute personne souhaitant déployer une infrastructure web fiable et sécurisée.

Bibliographie

[1] YAZID,M.Cours et Travaux Pratiques, Administration des Réseaux, page 4,5, Bejaia.

[4] Hoque, N., Bhuyan, M. H., Baishya, R. C., Bhattacharyya, D. K., & Kalita, J. K. (2014). Attaques réseau : Taxonomie, outils et systèmes**.** *Revue des Applications Réseau et Informatique*,40,307–324.

[5] Hansman, S., &Hunt, R. (2005). Une taxonomie des attaques réseau et informatique. *Informatique & Sécurité*,24(1),31–43. Elsevier.

[6] Sénétaire,V., Lepotier,N., & Soulard,T.(2024).*La cybersécurité de zéro*. Éditions Eyrolles.

[8] Alotaibi, A. M., Alshamrani, S.S., &Alghamdi, M. A. (2017). Problèmes de sécurité dans les protocoles du modèle TCP/IP au niveau des couches**.** *Revue internationale des réseaux informatiques et de la sécurité des communications*,5(5), 96–104.

[10] Khernane, N. (2021). *Les systèmes de filtrage de paquets (Pare-feu)*.In cyber sécurité 1. Université Batna 2, Faculté des mathématiques et de l'informatique, Département d'informatique.

[22] Menezes,A., vanOorschot,P.,&Vanstone,S.(1996). Overview Of Cryptography. *Handbook of applied cryptography*(pp.15-20).CRC Press.

[34] Stallings,W.(2017).*Cryptographie et sécurité des réseaux:Principes et pratiques*(7e éd.).Pearson.

[35] Pillou,J.-F.,&Bay,J.-Ph.(2013).*Sécurité informatique* (3eéd.).Dunod.

Webographie

[2] Cisco Networking Academy.(2020).Introduction aux réseaux–Guide d'accompagnement(CCNAv7).*Cisco Press.*[Enligne] Available : Available:https://unidel.edu.ng/focelibrary/books/121Introduction%20to%20Networks%2 0Companion%20Guide%20(CCNAv7)%20by%20Cisco%20Networking %20Academy%20(z-lib.org).pdf.[Accès le 20 Mars 2025].

[3] Kanawati,R.(2020). *Introduction à la cybersécurité* [Cours]. LIPN, Université Paris 13.[En ligne] Available: https://www-lipn.univ-paris13.fr/~kanawati/pdf/cybersec- CYU.pdf [Accés le 20 Mars 2025]

[7] Ferrag,M.A.(2018). *Sécurité Informatique* –[Cours].[En ligne]. Available : https://www.researchgate.net/profile/Mohamed-Amine-Ferrag/publication/350495879_Securite_Informatique_-_Cours_et_TD/links/60634353a6fdccbfea1a28ba/Securite-Informatique-Cours-et- TD.pdf [Accès le 12 avril 2025].

[9] CapCertification.(s.d).Norme ISO 27001:critères de sécurité.[Enligne]: https://capcertification.com/norme-iso-27001-criteres-de-securite/[Accès le 2 Avril 2025].

[11] Proofpoint.(s.d.).Qu'est e qu'un intrusion prevention system(IPS)[En ligne].Available: https://www.proofpoint.com/fr/threat-reference/intrusion-prevention-system-ips[Accès le 5 avril 2025].

[12] IBM.Qu'est ce que le chiffrement asymétrique?. 8 août 2024 [En ligne] . Available: https://www.ibm.com/fr fr/think/topics/asymmetric-encryption [Accès le 5 avril 2025].

[13] Fortinet. (s.d). Qu'est ce qu'un certificats numériques. [En ligne]. Available : https://www.fortinet.com/fr/resources/cyberglossary/digital-certificates [Accès le 5 avril 2025].

[14] GeoTrust.(s.d.).Comment fonctionnent un certificat TLS/SSL.[En ligne]. Available: https://www.geotrust.com/fr/how-does-tls-ssl-work [Accès le 5 avril 2025].

[15] Baie Brassage.(2024).Qu'est ce qu'un client serveur?.[En ligne]. Available: https://www.baiebrassage.fr/blog/client-serveur-definiton-role-fonctionnement.html [Accès le 10 avril 2025].

[16] SchoolMouv.(s.d.).*La page web (HTTP et langages HTML et CSS)*[En ligne]. Available:https://www.schoolmouv.fr/cours/la-page-web-http-et-langages-html-et-css-/fiche-de-cours[Accès le 10 avril 2025]

[17] Nameshield.(s.d.).*Protocole HTTPS*. [En ligne]. Available:https://www.nameshield.com/ressources/lexique/protocole-https/ [Accès le 10 avril 2025]

[18] MDNWebDocs.(2025). *Un aperçu du protocole HTTP*.[En ligne].Available: https://developer.mozilla.org/fr/docs/Web/HTTP/Guides/Overview[Accès le 10 Avril 2025]

[19] AlexHostSRL.(s.d).*FAQ–Explication du protocole HTTPS*.[En ligne]. Available : https://alexhost.com/fr/faq/ [Accès le 10 avril 2025].

[20] Guru99.(2024). *HTTP vs HTTPS:Différence entre eux* .[En ligne].Available :https://www.guru99.com/fr/difference-http-vs-https.html[Accès le 10 avril 2025].

[21] Pité,J.,& Zhong,Y. (2024).*The RSA cryptosystem*. Conférence PRIMES, Massachusetts Institute of Technology.[En ligne]. Available : https://math.mit.edu/research/highschool/primes/circle/documents/2024/Honglin.pdf [Accèsle12avril2025].

[23] Icodia. (s.d). *Les certificats SSL*. [En ligne]. Available : https://www.icodia.com/fr/solutions/certificats-ssl/en-savoir-plus-certificats-ssl.html[Accès le 17 avril 2025].

[24] LeMagIT. (s.d). *RSA (algorithme)*.02 Août 2016 [En ligne]. Available : https://www.lemagit.fr/definition/RSA-algorithme [Accès le 17 avril 2025].

[25] SSL Dragon.(2025).*Chiffrement RSA:Comment ça marche et pourquoi c'est important*. [En ligne]. Available:https://www.ssldragon.com/fr/blog/chiffrement-rsa/[Accès le 17 avril 2025]

[26] Wikipédia contributors.(2025).*Chiffrement RSA*.Wikipédia.[Enligne]. Available : https://fr.wikipedia.org/wiki/Chiffrement_RSA [Accès le 17 avril 2025]

[27] TBS Certificats.(2020).*Tout sur les algorithmes de hachage SHA1, SHA2 et le SHA256.* [En ligne]. Available : https://www.tbs-certificats.com/FAQ/fr/sha256.html[Accès le 18 avril 2025]

[28] VPN Unlimited.(s.d).*Algorithme de Signature Numérique à Courbe Elliptique (ECDSA).* [En ligne]. Available : https://www.vpnunlimited.com/fr/help/cybersecurity/elliptic-curve-digital-signature-algorithm[Accèsle 18 avril 2025]

[29] w3r.one. (s.d). *Curve 25519 et EdDSA : le passage aux standards de signature et de chiffrement modernes*.[En ligne]. Available : https://w3r.one/fr/blog/blockchain-web3/cryptographie/normes-evolutions-cryptographiques/curve25519-et-eddsa-le-passage-aux-standards-de-signature-et-de-chiffrement-modernes[Accès le 18 avril 2025]

[30] Dierks, T., & Rescorla, E. (2008). *RFC 5246 : The Transport Layer Security (TLS)Protocol Version1.2*.Internet Engineering Task Force.[En ligne]. Available : https://datatracker.ietf.org/doc/html/rfc5246[Accès le 15 Mai 2025]

[31] Grigorik,I.(s.d).*Transport layer security(TLS)*.O'Reilly Media.[En ligne]. Available:https://hpbn.co/transport-layer-security-tls/[Accès le 15 Mai 2025]

[32] IETF.(2008).LeprotocoleTLS(TransportLayerSecurity),version1.2(RFC5246). Internet Engineering Task Force.[En ligne]. Available : https://datatracker.ietf.org/doc/html/rfc5246 [Accès le 20 Mai 2025]

[33] Wikipédia contributors.(2025). Pare-feu (informatique) .[En ligne]. Available:https://fr.wikipedia.org/wiki/Pare-feu_(informatique)[Accès le 20 Mai 2025]

ANNEXE 1 : Installation de CentOS Stream 9 sur VMware

Dans cette annexe, nous détaillons les étapes suivies pour installer et configurer le système d'exploitation CentOS Stream 9, utilisé comme base de notre serveur web sécurisé.

1. Téléchargement de l'image ISO

La première étape a consisté à se rendre sur le site officiel de CentOS :
https://www.centos.org/download/
Nous avons choisi la version CentOS Stream 9, plus précisément l'image "x86_64 architecture ISOs", compatible avec notre machine hôte.

2. Création de la machine virtuelle

Après le téléchargement de l'ISO, nous avons lancé VMware Workstation puis cliqué sur "Create a New Virtual Machine". Nous avons sélectionné le fichier ISO comme image d'installation et configuré la machine avec les paramètres recommandés :

Type : Linux.

Version :CentOS 9 64-bit.

Mémoire RAM : selon les ressources disponibles (2 Go ou plus).

Disque dur virtuel : 20 Go minimum.

3. Sélection des paquets logiciels

Lors de l'installation de CentOS, l'étape "Software Selection" nous a permis de choisir les paquets nécessaires à notre projet. Nous avons coché les composants suivants :

Basic Web Server : pour disposer d'Apache et des outils de base liés au service web.

System Administration Tools : pour gérer et maintenir le système.

Security Tools : pour intégrer des outils liés à la sécurité (pare-feu, audit, etc.).

Development Tools : pour compiler, éditer et développer si nécessaire.

Performance Tools : pour surveiller l'état du système et ses performances.

MariaDB Database Client : pour interagir avec les bases de données MariaDB.

Python : utilisé par plusieurs scripts et outils système.

PHP Support : indispensable pour le fonctionnement de WordPress.

Cette sélection nous a permis d'avoir un environnement complet prêt à accueillir notre serveur web sécurisé.

4. Configuration de l'interface réseau

L'interface réseau utilisée dans notre machine virtuelle est nommée ens160. Nous avons choisi de la configurer en mode "Bridged", ce qui signifie que la machine virtuelle est connectée directement au même réseau que la machine physique hôte.
Ce mode permet à la machine virtuelle d'obtenir une adresse IP réelle sur le réseau local, comme s'il s'agissait d'un véritable serveur physique. Cela facilite :

- l'accès distant via SSH ou navigateur depuis d'autres machines du réseau .
- les tests en condition réelle de connectivité .
- la simulation d'un environnement de production plus réaliste.

ANNEXE 2 : Les commandes d'installation des différents paquets

Cette annexe présente les commandes utilisées pour l'installation des paquets nécessaires à la mise en place du serveur web sécurisé sous CentOS Stream 9 qui sont :

1 . Installation des paquets de base :

yum install -y wget chrony gcc net-tools iptables-services dnf automatic epel-release

2 . Installation du système de fichiers distant SSHFS :

dnf install fuse-sshfs

3. Installation de l'agent de transfert de courrier (Postfix) :

yum install -y postfix

4. Installation du serveur web Apache :

dnf install httpd

5. Installation du système de base de données MariaDB :

dnf install mariadb_server mariadb

6. Installation des bibliothèques de sécurité (HTTPS & cryptographie) :

dnf install openssl mod_ssl

7. Installation de bibliothèques système complémentaires :

dnf install libstdc++ libssh2 libssh2-dev libssh2-docs

8. Installation de PHP et des modules associés :

dnf install php gcc php-devel php-pear php-gd php-mbstring