Projet : Sécurisation d’un serveur Ubuntu (SSH, Apache, HTTPS, Fail2Ban)

1. Présentation du projet

Ce projet a pour objectif de mettre en place un serveur Ubuntu sécurisé dans un environnement de laboratoire, afin de comprendre concrètement les mécanismes de configuration, durcissement et protection d’un serveur Linux.

Les travaux ont été réalisés dans un environnement virtualisé, à des fins d’apprentissage et de tests contrôlés.

2. Objectifs pédagogiques

Comprendre l’installation et l’administration d’Ubuntu Server

Sécuriser l’accès distant via SSH

Mettre en place un serveur web Apache

Configurer HTTPS avec certificats SSL

Protéger les services contre les attaques par force brute avec Fail2Ban

Comprendre l’importance des rôles, des privilèges et des permissions

Documenter proprement un projet technique

3. Environnement de travail

Système serveur : Ubuntu Server 22.04 LTS

Virtualisation : VirtualBox / VMware

Machine cliente : Kali Linux

Réseau : Bridged

Accès distant : SSH

Pare-feu : UFW

Services sécurisés : Apache, SSH

4. Installation du serveur Ubuntu

Installation d’Ubuntu Server depuis l’image ISO officielle

Mise à jour du système dès l’installation

Création d’un utilisateur non-root

Activation de l’accès SSH

sudo apt update && sudo apt upgrade -y

5. Sécurisation de SSH
5.1 Principes appliqués

Désactivation de la connexion root

Restriction des utilisateurs autorisés

Limitation des tentatives de connexion

Choix du mode d’authentification (clé SSH et/ou mot de passe)

Réduction des privilèges utilisateurs

5.2 Extraits de configuration SSH

Fichier : /etc/ssh/sshd_config

PermitRootLogin no
AllowUsers test
MaxAuthTries 3
MaxSessions 2
PasswordAuthentication yes
PubkeyAuthentication yes

Après modification :

sudo systemctl restart ssh

6. Tests de sécurité SSH

Des tests d’attaque par force brute ont été réalisés dans un laboratoire personnel à l’aide

d’outils comme Hydra.

Mots de passe volontairement faibles

Observation du comportement du serveur

Ajustement progressif du durcissement

Analyse du moment où les attaques sont bloquées

👉 Conclusion :
Une bonne sécurité commence par une configuration stricte, la réduction des privilèges et des mots de passe solides.

7. Installation et configuration d’Apache
7.1 Installation
sudo apt install apache2 -y

7.2 Vérification
systemctl status apache2

Ports utilisés :

HTTP : 80

HTTPS : 443

8. Configuration HTTPS (SSL)
8.1 Activation du module SSL
sudo a2enmod ssl

8.2 Certificats SSL

Dans ce laboratoire, l’utilisation de certificats (Let’s Encrypt ou auto-signés) a permis de comprendre :

Le rôle des fichiers .crt et .key

L’importance des chemins corrects

Les permissions strictes sur la clé privée

Emplacements utilisés :

Certificat : /etc/ssl/certs/
Clé privée : /etc/ssl/private/

Permissions appliquées :

sudo chmod 600 /etc/ssl/private/serveur.key
sudo chown root:root /etc/ssl/private/serveur.key

9. Protection avec Fail2Ban

Fail2Ban a été configuré pour protéger :

SSH

Apache

Objectifs :

Bloquer automatiquement les adresses IP malveillantes

Réduire l’impact des attaques par force brute

Journaliser les tentatives suspectes

10. Bonnes pratiques appliquées

Principe du moindre privilège

Séparation des rôles utilisateurs

Surveillance des logs

Pare-feu restrictif

Tests de sécurité contrôlés

Documentation systématique

11. Avertissement

⚠️
Toutes les expérimentations ont été réalisées dans un environnement de test personnel.
Aucune attaque n’a été menée sur des systèmes tiers ou en production.

12. Conclusion

Ce projet m’a permis de renforcer ma compréhension :

de l’administration système Linux

de la sécurité des services réseau

de la logique défensive face aux attaques

de l’importance d’une configuration propre et documentée