Département Informatique
Promotion 2022-2025

SAE 4.01
Déployer et sécuriser des services dans un réseau

Rapport intermédiaire collectif

- Groupe 2 -

Alexis DEBRA
[redacted]

Projet effectué au département informatique de l'IUT de Metz

Introduction

Dans le cadre de la SAE 4.01 de notre 4ème semestre de BUT Informatique à l'IUT de Metz, il nous a été demandé de sécuriser un système Linux Debian 11 hébergeant un serveur et une application Web. Une machine virtuelle nous a été attribuée et notre but ces dernières semaines a été de sécuriser au maximum cette dernière afin que les autres groupes n'aient pas la capacité de porter atteinte de quelque manière que ce soit à son intégrité ou à celle du serveur Web qu'elle héberge.

Pour ce faire, nous avons tout d'abord dû mener une phase de recherches, nécessaire à la compréhension des points sensibles à surveiller dans le cadre d'un tel projet de sécurisation. Nous avons ensuite procédé à une phase d'installations et de paramétrages des différents outils installés ainsi que de la machine virtuelle et de ses composants hébergés.

Ce rapport sert de bilan en ce sens, nous reviendrons donc sur nos recherches, sur ce que nous avons identifié comme des points à sécuriser et ainsi, sur ce que nous avons installé et configuré jusqu'à présent sur notre machine. Nous conclurons ce rapport sur ce qui nous reste encore à configurer ainsi que sur les sources que nous avons consultées et qui nous ont aidés durant notre phase de recherches.

Recherches

Lors de nos recherches, nous avons exploré une variété de ressources en ligne pour identifier les vulnérabilités potentielles et les techniques d'attaques courantes.

Parmi les sources importantes que nous avons consultées figurent les guides de bonnes pratiques en matière de sécurité informatique publiés par le gouvernement français, tels que ceux disponibles sur les sites du CERT-FR et de l'ANSSI. De plus, nous avons étudié les recommandations de l'OWASP (Open Web Application Security Project) pour mieux comprendre les principales menaces pesant sur les systèmes informatiques.

Nous avons également examiné des documents techniques spécifiques à Debian, tels que le manuel Debian FAQ, pour obtenir des informations sur les meilleures pratiques de sécurisation des systèmes Linux Debian 11.

En complément, nous avons consulté des blogs spécialisés, des forums de sécurité en ligne et des ressources telles que RaspberryTips et Root-Me pour approfondir nos connaissances et trouver des conseils pratiques pour renforcer la sécurité de notre infrastructure.

En synthétisant les informations recueillies à partir de ces diférentes sources, voici une liste de ce que nous allons réaliser sur notre infrastructure afin de la sécuriser au maximum :

- Effectuer des mises à jour régulières du système et des dépendances.
- Surveiller activement les fichiers de logs pour détecter les activités suspectes.
- Être vigilant face aux techniques d'ingénierie sociale et sensibiliser les utilisateurs.
- Mettre en place un système de sauvegarde géré par l'IUT de Metz.
- Changer les mots de passe par défaut (debian1, root, phpmyadmin).
- Générer des clés SSH et désactiver la connexion par mot de passe.
- Bloquer la connexion root en SSH et changer le port de connexion par défaut.
- Mettre en place un certificat SSL auto-signé pour chiffrer les échanges client-serveur.
- Forcer la redirection de HTTP vers HTTPS pour sécuriser les communications web.
- Utiliser Fail2ban pour limiter le nombre de tentatives de connexion infructueuses et bloquer les adresses IP suspectes.
- Configurer UFW pour bloquer les ports inutilisés et limiter l'accès réseau.
- Appliquer une politique de mots de passe solides pour les utilisateurs de l'application.
- Examiner régulièrement le code de l'application pour détecter et corriger les éventuelles vulnérabilités.
- Mettre en place une Content Security Policy pour sécuriser les entrées utilisateur et prévenir les attaques XSS.
- Appliquer le principe du moindre privilège sur les dossiers de la machine virtuelle et les groupes d'utilisateurs.
- Réaliser des tests d'intrusion pour identifier et corriger les failles de sécurité potentielles.

Installations et configurations

Tout d'abord, il est nécessaire de prendre connaissance des caractéristiques du serveur (CPU, RAM, Espace disque).

CPU :

```
root@sujet12:~# lscpu
Architecture :                       x86_64
Mode(s) opératoire(s) des processeurs : 32-bit, 64-bit
Boutisme :                           Little Endian
Tailles des adresses:                43 bits physical, 48 bits virtual
Processeur(s) :                      1
Liste de processeur(s) en ligne :    0
Thread(s) par cœur :                 1
Cœur(s) par socket :                 1
Socket(s) :                          1
Nœud(s) NUMA :                       1
Identifiant constructeur :           GenuineIntel
Famille de processeur :              6
Modèle :                             85
Nom de modèle :                      Intel(R) Xeon(R) Bronze 3106 CPU @ 1.70GHz
Révision :                           4
Vitesse du processeur en MHz :       1696.015
BogoMIPS :                           3392.03
Constructeur d'hyperviseur :         VMware
Type de virtualisation :             complet
Cache L1d :                          32 KiB
Cache L1i :                          32 KiB
Cache L2 :                           1 MiB
Cache L3 :                           11 MiB
Nœud NUMA 0 de processeur(s) :       0
```

Nous avons un processeur Intel Xeon Bronze 3106 avec une vitesse de 1.70GHz. Son architecture est en x86-64 et le système fonctionne sur la base du Little Endian.

RAM :

```
root@sujet12:~# free -h
               total        utilisé      libre     partagé tamp/cache   disponible
Mem:           1,9Gi       271Mi       166Mi       3,0Mi       1,5Gi       1,5Gi
Partition d'échange:      974Mi       0,0Ki       974Mi
```

Nous possédons 1.9Gi de mémoire vive / RAM et en utilisons 271Mi.

Espace disque :

```
root@sujet12:~# df -h
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
udev                976M       0  976M   0% /dev
tmpfs               199M    660K  198M   1% /run
/dev/sda1            15G    3,2G   11G  23% /
tmpfs               992M       0  992M   0% /dev/shm
tmpfs               5,0M       0  5,0M   0% /run/lock
tmpfs               199M       0  199M   0% /run/user/1000
root@sujet12:~#
```

Notre disque se situe dans /dev/sda1 et avons un espace total de 15Go et en utilisons 3.2Go.

Ensuite, nous mettons régulièrement à jour le système, comme conseillé par la documentation Debian :

```
root@sujet12:~# apt update
Atteint :1 http://security.debian.org/debian-security bullseye-security InRelease
Atteint :2 http://deb.debian.org/debian bullseye InRelease
Atteint :3 http://deb.debian.org/debian bullseye-updates InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Tous les paquets sont à jour.
root@sujet12:~#
```

```
root@sujet12:~# apt upgrade
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Calcul de la mise à jour... Fait
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
root@sujet12:~#
```

La mise à jour du système est importante car lorsqu'une faille est découverte elle commencera à être exploitée et ce sont les mises à jour qui la corrigerons dès que possible.
Aussi, nous avons pris conscience qu'il était important de surveiller les fichiers de logs de notre machine. Les principaux logs sont situés sous /var/log, les principaux services à surveiller sont évidemment dans notre cas : ssh (auth.log), apache (apache2), fail2ban (fail2ban.log), ufw (ufw.log) et mysql (mysql).

```
root@sujet12:/var/log# ls
alternatives.log       daemon.log       dpkg.log.4.gz   mail.err       runit                vmware-network.3.log
alternatives.log.1     daemon.log.1     dpkg.log.5.gz   mail.err.1     syslog               vmware-network.4.log
alternatives.log.2.gz  daemon.log.2.gz  exim4           mail.info      syslog.1             vmware-network.5.log
alternatives.log.3.gz  daemon.log.3.gz  fail2ban.log    mail.info.1    syslog.2.gz          vmware-network.6.log
alternatives.log.4.gz  daemon.log.4.gz  fail2ban.log.1  mail.log       syslog.3.gz          vmware-network.7.log
alternatives.log.5.gz  dbconfig-common  faillog         mail.log.1     syslog.4.gz          vmware-network.log
apache2                debug            fontconfig.log  mail.warn      ufw.log              vmware-vmsvc-root.1.log
apt                    debug.1          installer       mail.warn.1    ufw.log.1            vmware-vmsvc-root.2.log
auth.log               debug.2.gz       journal         messages       user.log             vmware-vmsvc-root.3.log
auth.log.1             debug.3.gz       kern.log        messages.1     user.log.1           vmware-vmsvc-root.log
auth.log.2.gz          debug.4.gz       kern.log.1      messages.2.gz  user.log.2.gz        vmware-vmtoolsd-root.log
auth.log.3.gz          dpkg.log         kern.log.2.gz   messages.3.gz  user.log.3.gz        wtmp
auth.log.4.gz          dpkg.log.1       kern.log.3.gz   messages.4.gz  user.log.4.gz
btmp                   dpkg.log.2.gz    kern.log.4.gz   mysql          vmware-network.1.log
btmp.1                 dpkg.log.3.gz    lastlog         private        vmware-network.2.log
root@sujet12:/var/log#
```

Nous avons par la suite relevé le fait que nous allions devoir rester vigilant vis à vis des techniques d'ingénierie sociale qui auraient pour but de nous subtiliser nos identifiants ou des informations sur notre configuration.

Enfin, un système de sauvegarde est primordial, tout d'abord dans le cas d'un « bête » accident physique (incendie ou inondation du serveur hôte, panne matérielle, vol, …) de manière que nous puissions assurer la disponibilité et l'intégrité des données des « utilisateurs ». Mais aussi dans le cas où un hacker parviendrait à installer sur notre machine un ransomware cryptant nos données et nous obligeant à réinitialiser notre machine. Ce système de sauvegarde est géré dans le cadre de notre SAE par l'IUT de Metz, mais il est tout de même important de le rappeler ici.

Après ces points simples, mais primordiaux, nous avons décidé d'aller plus loin dans notre sécurisation.

L'une des actions recommandées dans nos recherches était de changer les mots de passe root et utilisateur par défaut de notre machine virtuelle. D'autant plus que dans notre cas, les mots de passe par défaut de ces utilisateurs ont été communiqués aux autres groupes censés attaquer notre infrastructure plus tard.
Pour ce faire, nous nous sommes connectés en tant qu'utilisateur 'root' et avons simplement utilisé la commande 'passwd debian1' et 'passwd'. La première nous permit de changer le mot de passe de debian1 et la seconde celui de root.

L'un des principaux vecteurs de risques, notamment top 1 de l'OWASP dans son classement des principales failles de sécurité informatique, est l'authentification. En l'occurrence, nous avons jugé notre mécanisme d'authentification comme étant toujours trop faible, et avons décidé passer à une authentification par clé. Nous avons donc chacun généré une paire de clé nous permettant de nous connecter au serveur, en suivant le TP de sécurité que nous avons dû réaliser en cours portant sur les connexions SSH. Nous avons de plus changé le port de connexion par défaut à SSH par le port 5500 afin de compliquer la recherche/compromission du service lors des scans ou outils utilisant le port 22 par défaut et activé le niveau de log de ssh 'LogLovel' à VERBOSE.

Pour forcer cette authentification par clés, nous avons désactivé l'authentification à l'utilisateur root par ssh, désactivé l'authentification par mot de passe et forcé la connexion par clés dans le fichier de configuration de ssh '/etc/ssh/sshd_config' :

```
Include /etc/ssh/sshd_config.d/*.conf

Port 5500
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
LogLevel VERBOSE

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes
```

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no
```

La commande 'systemctl restart sshd.service' fut nécessaire pour que le service ssh prenne en compte les modifications.

Afin de pouvoir tous générer nos clés sans encombre, un programme bash a été créé afin de générer nos clés automatiquement.

$ key_add.sh

```
#!/bin/bash

# Programme de génération de clé pour serveurr debian1 - sujet12

SSH_SERVER="debian1@100.74.7.96 -p 5500";
USER="$1";
DIR_SSH=".ssh";

if [ "$#" -eq 0 ]; then
    echo "[-] Usage => ./key_add.sh <user> (check with whoami before running ^^)"
    exit 1
fi

printf "[+] MacOS / Linux [1 - 2]\n";
read -r VERSION;

if [ "$VERSION" = 1 ]; then
    GO_PATH="/Users/$USER";
elif [ "$VERSION" = 2 ]; then
    GO_PATH="/home/\$USER";
else
    echo "[-] Error...";
    exit 1
fi


if [ ! -d "$GO_PATH/$DIR_SSH" ]; then
    printf "[-] Errorrr...";
    exit 1;
else
    cd "$GO_PATH/$DIR_SSH" || exit 1;
    printf "[+] Current directory : %s" "$GO_PATH/$DIR_SSH";
    ssh-keygen -t rsa -b 4096
    ssh-add id_rsa
    ssh-copy-id "$SSH_SERVER";

    printf "Connection to %s with key authentification...\n" "$SSH_SERVER";
    CONNECT_SSH="ssh $SSH_SERVER";
    $CONNECT_SSH;
fi
```

Nous avons aussi jugé important que l'application en elle-même force les utilisateurs à choisir un mot de passe solide. En effet, les logiciels de brute force actuels sont bien trop puissants et il faut absolument que les comptes utilisateurs et surtout administrateurs ne soient pas accessibles à de potentiels attaquants. Pour ce faire nous avons examiné le code de l'application à la recherche de l'endroit où la création de compte avait lieu. Heureusement, les développeurs étaient forts compétents et avaient pris en compte cela. En effet, le code php de l'application force l'utilisateur à choisir un mot de passe solide à l'aide d'expressions régulières dans le fichier 'controleur/inscription.php'.

```
// Vérification des exigences du mot de passe
if (strlen($mdp) >= 8 && preg_match('/[A-Za-z]/', $mdp) && preg_match('/\d/', $mdp) && preg_match('/[^A-Za-z\d]/', $mdp)) {
    // Mise à jour des données de l'utilisateur avec une requête SQL
    $updateUserQuery = "UPDATE utilisateur SET nom = :nom, prenom = :prenom, mdp = :mdp WHERE id_user = :userId";
    $conn->execSQL($updateUserQuery, [':nom' => $nom, ':prenom' => $prenom, ':mdp' => password_hash($mdp, PASSWORD_DEFAULT), ':userId' => $userId]);

    // Ajout de la question de sécurité et de la réponse correspondante
    $utilisateurDAO->insertSecurityQuestion($userId, $questionSecurite, $reponseSecurite);

    // Redirection vers la page de confirmation ou autre
    header('Location: login.php');
    exit;
} else {
    echo "Le mot de passe doit contenir au moins 8 caractères, y compris des lettres, des chiffres et un caractère spécial ( # $ % & ' ( ) * + , - . / :
}
```

Nous venons de traiter des risques liés aux attaques par force brute et menées par des bots. Et c'est pour nous prémunir de ce genre d'attaques que nous avons par la suite installé et configuré fail2ban. Cet outil surveille les fichiers de logs que nous lui indiquons et repère les patterns de logs suspects provoqués par les tentatives d'attaques par brute force et DdoS menées par des bots sur

notre machine. Fail2ban peut en fonction de sa configuration bloquer les IP suspectes définitivement ou temporairement.
Nous avons installé l'outil avec 'apt-install fail2ban' et avons édité un fichier situé sous '/etc/fail2ban/jail.d/ ' afin de configurer nos jails, les services que fail2ban va par la suite surveiller. Ici nous avons décidés de surveiller ssh (classique et les attaques par Ddos), apache, les IP qui récidivent dans leurs tentatives infructueuses de connexion, et un filtre personnalisé que nous avons créé, censé repérer les logs liés à un échec de connexion par clés sur notre VM dont nous allons parler par la suite.

```
root@sujet12:~# cat /etc/fail2ban/jail.d/defaults-debian.conf
[DEFAULT]
maxretry = 10
findtime = 120
bantime = 1200

[sshd]
port = 5500
enabled = true
maxretry = 10
findtime = 120
bantime = 1200
logpath = /var/log/auth.log

[sshd-ddos]
port = 5500
enabled = true
filter = sshd
maxretry = 10
findtime = 120
bantime = 1200
logpath = /var/log/auth.log

[recidive]
enabled = true

[apache]
enabled = true
port     = http,https
filter   = apache-auth
logpath  = /var/log/apache*/*error.log
maxretry = 10
findtime = 120
bantime = 1200

[key-filter]
enabled = true
filter = key-filter
maxretry = 10
findtime = 120
bantime = 1200
logpath = /var/log/auth.log
port = 5500
root@sujet12:~#
```

```
root@sujet12:~# fail2ban-client status
Status
|- Number of jail:      5
`- Jail list:   apache, key-filter, recidive, sshd, sshd-ddos
root@sujet12:~#
```

Nous avons par défaut activé cette protection : Si une IP échoue à s'authentifier 10 fois en l'espace de 2 minutes, alors cette IP est bannie pour 20 minutes de serveur. Grâce à cela, le risque d'attaques par DdoS et par brute force sera réduit.

D'autant plus que nous avons créé un filtre personnalisé qui détecte dans les logs ssh les logs relatifs aux échecs d'authentification par clé à notre machine. Voilà ce filtre, basé sur une expression régulière et situé sous '/etc/fail2ban/filter.d/key-filter' :

```
root@sujet12:/etc/fail2ban/filter.d# cat ./key-filter.conf
[Definition]
failregex = ^.*sshd\[\d+\]: Connection from <HOST> port \d+.*$
            ^.*sshd\[\d+\]: Connection reset by authenticating user .* <HOST> port \d+ \[preauth\].*$
root@sujet12:/etc/fail2ban/filter.d#
```

Voici l'état du filtre, alors qu'il n'a bloqué aucune IP à gauche, et à droite, lorsqu'il a bloqué une IP d'un client qui a échoué à s'authentifier 10 fois en 2 minutes car il n'avait pas la bonne clé d'authentification :

```
Status for the jail: key-filter
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- Journal matches:
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
root@sujet12:/etc/fail2ban/filter.d#
```

```
Status for the jail: key-filter
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     10
|  `- Journal matches:
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   172.19.170.216
root@sujet12:/etc/fail2ban/filter.d#
```

Une fois fail2ban configuré, le mot de passe root de la base de donnée à été changé à l'aide de mysql_secure_installation :

```
root@sujet12:/etc/apache2/sites-available# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!
```

Pour renforcer davantage notre sécurité, nous avons déployé et configuré UFW (Uncomplicated Firewall). Cet outil simplifié nous permet de gérer et de filtrer le trafic réseau entrant et sortant sur notre machine.
En configurant UFW, nous avons établi des règles spécifiques pour autoriser ou refuser le trafic en fonction de ses origines et de ses destinations. Par exemple, nous avons autorisé le trafic entrant sur les ports SSH et HTTP/HTTPS, tout en bloquant le trafic sur les ports non essentiels. Cette approche nous permet de limiter les tentatives d'accès non autorisées à notre système, réduisant ainsi les risques de compromission de la sécurité.

```
root@sujet12:~# apt-get update
Atteint :1 http://security.debian.org/debian-security bullseye-security InRelea
e
Atteint :2 http://deb.debian.org/debian bullseye InRelease
Atteint :3 http://deb.debian.org/debian bullseye-updates InRelease
Lecture des listes de paquets... Fait
root@sujet12:~#
```

```
root@sujet12:~# apt install ufw
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
ufw est déjà la version la plus récente (0.36-7.1).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 1 non mis à jour.
root@sujet12:~# ufw status
Status: inactive
root@sujet12:~#
```

Dans un premier temps nous mettons à jour le système et installons les paquets du pare-feu.

```
root@sujet12:~# ufw reset
Resetting all rules to installed defaults. This may disrupt existing ssh
connections. Proceed with operation (y|n)? Y
Backing up 'user.rules' to '/etc/ufw/user.rules.20240303_140315'
Backing up 'before.rules' to '/etc/ufw/before.rules.20240303_140315'
Backing up 'after.rules' to '/etc/ufw/after.rules.20240303_140315'
Backing up 'user6.rules' to '/etc/ufw/user6.rules.20240303_140315'
Backing up 'before6.rules' to '/etc/ufw/before6.rules.20240303_140315'
Backing up 'after6.rules' to '/etc/ufw/after6.rules.20240303_140315'
```

On a décidé de remettre à 0 les anciennes règles pour éviter tout potentiel conflit avec les anciennes configurations.

```
root@sujet12:~# ufw allow 80/tcp
Rules updated
Rules updated (v6)
root@sujet12:~# ufw allow OpenSSH
Rules updated
Rules updated (v6)
root@sujet12:~#
```

Cette configuration ci-dessus fut la première mise en place.

Par la suite, afin de sécuriser le plus possible nous avons seulement ouvert les ports 443 et 5500 respectivement utilisés pour le protocole https et les connexions SSH. Ainsi cela réduit les chances de tentatives de connexion via SSH directement.

```
root@sujet12:~# ufw status
Status: active

To                         Action      From
--                         ------      ----
443/tcp                    ALLOW       Anywhere
5500/tcp                   ALLOW       Anywhere
443/tcp (v6)               ALLOW       Anywhere (v6)
5500/tcp (v6)              ALLOW       Anywhere (v6)
```

```
root@sujet12:~# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@sujet12:~#
```

Il ne reste plus qu'à rendre le service disponible sur Debian.

Un autre service tiers dans le style de Fail2ban est ModSecurity. C'est un pare-feu d'application web open source qui agit en tant que module Apache ou Nginx et protège contre plusieurs types d'attaques, y compris les attaques par injection SQL, les attaques XSS, les tentatives d'accès non autorisées, etc…

Nous avons décidé de l'installer sur le serveur mais de ne pas l'activer pour le moment, puisque nous avons jugé que de maintenir Fail2ban ou UFW était plus simple pour nous en cas de soucis. Si on jugeait par la suite une nécessité de l'activer nous le ferions.

La procédure fut suivie sur des documentations internet :

```
root@sujet12:~# apt install libapache2-mod-security2
```

Dans un premier temps nous avons installé les paquets nécessaires.

```
root@sujet12:~# git clone https://github.com/coreruleset/coreruleset /etc/modsecurity/coreruleset
Clonage dans '/etc/modsecurity/coreruleset'...
remote: Enumerating objects: 29732, done.
remote: Counting objects: 100% (741/741), done.
remote: Compressing objects: 100% (278/278), done.
remote: Total 29732 (delta 517), reused 645 (delta 463), pack-reused 28991
Réception d'objets: 100% (29732/29732), 7.87 Mio | 13.04 Mio/s, fait.
Résolution des deltas: 100% (23198/23198), fait.
root@sujet12:~#
```

Nous avons ensuite téléchargé le CRS depuis le référentiel GitHub officiel. Le CRS est un fichier qui permet de configurer par la suite ModSecurity.

```
root@sujet12:~# cp /etc/modsecurity/coreruleset/crs-setup.conf.example /etc/modsecurity/coreruleset/crs-setup.conf
root@sujet12:~#
```

On copie ensuite le fichier de configuration par défaut.

```
root@sujet12:~# a2enmod security2
Considering dependency unique_id for security2:
Module unique_id already enabled
Module security2 already enabled
```

On active par la suite ModSecurity, mais cela ne fonctionnera pas si le CRS n'est pas configuré correctement.

Le fichier tel que modifié ci-dessous permet d'activer les règles sur le serveur afin que l'outil soit utilisable. Nous ne l'avons pas fait mais voici les commandes qui le permettent en cas de besoin.

```
#
# Include des règles ModSecurity
Include "/etc/modsecurity/coreruleset/crs-setup.conf"
Include "/etc/modsecurity/coreruleset/rules/*.conf"

AccessFileName .htaccess

#
```

On ajoute les lignes suivantes à la fin du fichier de configuration par défaut pour inclure le module ModSecurity et activer les règles du CRS.

```
root@sujet12:/# chmod 755 /etc/modsecurity/coreruleset
root@sujet12:/# chmod 755 /etc/modsecurity/coreruleset/rules
root@sujet12:/# chmod 644 /etc/modsecurity/coreruleset/*.conf
root@sujet12:/# chmod 644 /etc/modsecurity/coreruleset/rules/*.conf
root@sujet12:/#
```

Afin de ne pas avoir d'erreur, il est conseillé de mettre des droits spécifiques (non root bien entendu) mais nécessaire pour ne pas louper la configuration.

```
root@sujet12:/etc/modsecurity/coreruleset# apache2ctl -M | grep security2_module
 security2_module (shared)
root@sujet12:/etc/modsecurity/coreruleset#
```

On voit que tout est bien installé.

Afin de faire prendre en compte les modifications on redémarre le service Apache et tout fonctionne.

En testant la page Web du projet réalisé par le groupe précédent, nous nous sommes rendu compte d'un oubli assez important, la mise en place du protocole HTTPS.

HTTPS est essentiel pour sécuriser les échanges de données en ligne. Il protège la confidentialité, l'intégrité et l'authenticité des informations échangées entre les navigateurs web et notre serveur. Voici une configuration que nous avons suivie.

Création d'un répertoire pour stocker les certificats + génération clé privé RSA de longueur 2048:

```
root@sujet12:/etc/apache2# mkdir Certificat
root@sujet12:/etc/apache2# cd Certificat/
root@sujet12:/etc/apache2/Certificat# openssl genrsa -out private.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
..................+++++
.........................................+++++
e is 65537 (0x010001)
root@sujet12:/etc/apache2/Certificat# ls -la private.key
-rw------- 1 root root 1679 16 mars  14:05 private.key
root@sujet12:/etc/apache2/Certificat#
```

Génération d'une demande de certificat (CSR) en utilisant la clé privée :

```
root@sujet12:/etc/apache2/Certificat# openssl req -new -key private.key -out certificat.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

Signature du CSR avec la clé privée pour obtenir un certificat auto-signé (valable 365 jours) :

```
root@sujet12:/etc/apache2/Certificat# openssl x509 -req -days 365 -in certificat.csr -signkey private.key -out certificat.crt
Signature ok
subject=C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
Getting Private key
root@sujet12:/etc/apache2/Certificat# ls
certificat.crt  certificat.csr  private.key
root@sujet12:/etc/apache2/Certificat#
```

Gestion des droits sur le répertoire Certificat et son arborescence :

```
root@sujet12:/etc/apache2/Certificat# chmod a+x -R ../Certificat/
root@sujet12:/etc/apache2/Certificat# chmod u+rw -R ../Certificat/
root@sujet12:/etc/apache2/Certificat# ls -la ../Certificat/
total 20
drwx--x--x 2 root root 4096 16 mars  14:18 .
drwxr-xr-x 9 root root 4096 16 mars  14:13 ..
-rwx--x--x 1 root root 1123 16 mars  14:09 certificat.crt
-rwx--x--x 1 root root  956 16 mars  14:06 certificat.csr
-rwx--x--x 1 root root 1679 16 mars  14:05 private.key
root@sujet12:/etc/apache2/Certificat#
```

On ouvre le fichier de configuration d'Apache et on y inclus :

```
# Certicat SSL/TLS for https
SSLEngine on
SSLCertificateFile "/etc/apache2/Certificat/certificat.crt"
SSLCertificateKeyFile "/etc/apache2/Certificat/private.key"
```

Afin d'éviter les attaques XSS / CSRF, nous avons décidé de mettre en place des règles CSP (Content-Security-Policy) et activer le mod_rewrite afin d'interdire l'utilisation de certains mots pouvant être utilisés afin de compromettre la sécurité de notre serveur.

```
# sudo a2enmod headers - CSP (Content-Security-Policy)
<IfModule mod_headers.c>
    Header set Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://100.74.7.96/phpmyadmin/"
</IfModule>

# sudo a2enmod rewrite
RewriteEngine On
# Pour le repertoire controleur
RewriteCond %{REQUEST_URI} ^/controleur/ [NC]
# Interdiction des mots script / SCRIPT / document / cookie dans la requete
RewriteCond %{QUERY_STRING} (script|SCRIPT|document|cookie) [NC]
# Idem pour la méthode POST
RewriteCond %{REQUEST_METHOD} POST
RewriteCond %{REQUEST_BODY} (script|SCRIPT|document|cookie) [NC]
# Renvoie un acces forbidden en cas de violation des regles
RewriteRule .* - [F]
```

Activation des modules headers et rewrite de apache2 :

```
root@sujet12:/etc/apache2/sites-available# a2enmod headers
Module headers already enabled
root@sujet12:/etc/apache2/sites-available# a2enmod rewrite
Module rewrite already enabled
root@sujet12:/etc/apache2/sites-available#
```

Voici le fichier final de configuration apache2 :

```
<VirtualHost *:443>
    DocumentRoot "/var/www/html"

    # Certicat SSL/TLS for https
    SSLEngine on
    SSLCertificateFile "/etc/apache2/Certificat/certificat.crt"
    SSLCertificateKeyFile "/etc/apache2/Certificat/private.key"

    # Web directory
    <Directory "/var/www/html">
        Options +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    # sudo a2enmod headers - CSP (Content-Security-Policy)
    <IfModule mod_headers.c>
        Header set Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://100.74.7.96/phpmyadmin/"
    </IfModule>

    # sudo a2enmod rewrite
    RewriteEngine On
    # Pour le repertoire controleur
    RewriteCond %{REQUEST_URI} ^/controleur/ [NC]
    # Interdiction des mots script / SCRIPT / document / cookie dans la requete
    RewriteCond %{QUERY_STRING} (script|SCRIPT|document|cookie) [NC]
    # Idem pour la méthode POST
    RewriteCond %{REQUEST_METHOD} POST
    RewriteCond %{REQUEST_BODY} (script|SCRIPT|document|cookie) [NC]
    # Renvoie un acces forbidden en cas de violation des regles
    RewriteRule .* - [F]

    # Fichier de journalisation
    ErrorLog /var/log/apache2/error.log
    CustomLog /var/log/apache2/access.log combined
</VirtualHost>
```

Toujours dans le cadre de nos recherches de failles dans le système nous avons remarqué un problème de droits sur fichiers sensibles.

```
root@sujet12:/var/www/html# ls
60552B73958D2A31F05304A558895996.txt  controleur  __MACOSX  s301.zip    vue
Certificat                            index.php   modele    sae301.zip
root@sujet12:/var/www/html# rm s301.zip
root@sujet12:/var/www/html# rm sae301.zip
root@sujet12:/var/www/html# cp 60552B73958D2A31F05304A558895996.txt Certificat/
root@sujet12:/var/www/html# ls
60552B73958D2A31F05304A558895996.txt  Certificat  controleur  index.php  __MACOSX  modele  vue
root@sujet12:/var/www/html# cd Certificat/
root@sujet12:/var/www/html/Certificat# ls
60552B73958D2A31F05304A558895996.txt  certificate.crt  certificate.csr  private.key
root@sujet12:/var/www/html/Certificat# cd ..
root@sujet12:/var/www/html# rm 60552B73958D2A31F05304A558895996.txt
root@sujet12:/var/www/html# ls
Certificat  controleur  index.php  __MACOSX  modele  vue
root@sujet12:/var/www/html#
```

```
root@sujet12:/var/www/html# cd Certificat/
root@sujet12:/var/www/html/Certificat# rm 60552B73958D2A31F05304A558895996.txt
```

Voici après quelques analyses les modifications faites :

- Un fichier de certificat en .txt était exposé dans le répertoire où les codes sources étaient exposés. On a installé un certificat donc pour éviter tout faille de sécurité on l'a supprimé.
- Des fichiers « .zips » inutiles étaient aussi présents, on les a supprimés.

Les fichiers MVC ont bien entendu étaient gardés mais nous avons modifier des droits sur ces répertoires sensibles.

Pour sécuriser les répertoires tout en permettant à l'utilisateur www-data d'accéder et de modifier les fichiers, ce qui est souvent nécessaire pour que l'application Web fonctionne correctement, nous avons effectué différentes affectations de droits :

```
root@sujet12:/var/www/html# ls -l -a
total 36
drwxrwxrwx 8 www-data www-data 4096 12 mars  16:44 .
drwxr-xr-x 4 root     root     4096 16 janv. 13:15 ..
drwx--x--x 2 root     root     4096 12 mars  16:44 Certificat
drwxr-xr-x 3 debian1  debian1  4096 12 févr. 14:18 controleur
-rw-r--r-- 1 debian1  debian1   361 18 janv. 17:41 index.php
drwxr-xr-x 4 debian1  debian1  4096 18 janv. 21:08 __MACOSX
drwxr-xr-x 3 debian1  debian1  4096 12 févr. 14:04 modele
drwxr-xr-x 3 debian1  debian1  4096 19 janv. 10:21 vue
drwxr-xr-x 3 root     root     4096 16 janv. 13:36 .well-known
root@sujet12:/var/www/html#
```

Voici les droits initiaux.

```
root@sujet12:/var/www/html# ls -l -a
total 36
drwxrwxrwx 8 www-data www-data 4096 12 mars  16:44 .
drwxr-xr-x 4 root     root     4096 16 janv. 13:15 ..
drwx--x--x 2 root     root     4096 12 mars  16:46 Certificat
drwxr-xr-x 3 debian1  debian1  4096 12 févr. 14:18 controleur
-rw-r--r-- 1 debian1  debian1   361 18 janv. 17:41 index.php
drwxr-xr-x 4 debian1  debian1  4096 18 janv. 21:08 __MACOSX
drwxr-xr-x 3 debian1  debian1  4096 12 févr. 14:04 modele
drwxr-xr-x 3 debian1  debian1  4096 19 janv. 10:21 vue
drwxr-xr-x 3 root     root     4096 16 janv. 13:36 .well-known
root@sujet12:/var/www/html# chmod 750 Certificat controleur modele vue .well-known
root@sujet12:/var/www/html# chown www-data:www-data Certificat controleur modele vue .well-known
root@sujet12:/var/www/html# chmod 640 index.php
root@sujet12:/var/www/html# chown debian1:www-data index.php
root@sujet12:/var/www/html# ls -l -a
total 36
drwxrwxrwx 8 www-data www-data 4096 12 mars  16:44 .
drwxr-xr-x 4 root     root     4096 16 janv. 13:15 ..
drwxr-x--- 2 www-data www-data 4096 12 mars  16:46 Certificat
drwxr-x--- 3 www-data www-data 4096 12 févr. 14:18 controleur
-rw-r----- 1 debian1  www-data  361 18 janv. 17:41 index.php
drwxr-xr-x 4 debian1  debian1  4096 18 janv. 21:08 __MACOSX
drwxr-x--- 3 www-data www-data 4096 12 févr. 14:04 modele
drwxr-x--- 3 www-data www-data 4096 19 janv. 10:21 vue
drwxr-x--- 3 www-data www-data 4096 16 janv. 13:36 .well-known
root@sujet12:/var/www/html#
```

Et maintenant après modifications.

Pour détailler cette démarche voici une explication des commandes :

Fixer les permissions pour les répertoires :

- « sudo chmod 750 Certificat controleur modele vue .well-known »

Assurer que le propriétaire est www-data et que le groupe est www-data pour les répertoires :

- « sudo chown www-data:www-data Certificat controleur modele vue .well-known »

Fixer les permissions pour les fichiers :

- « sudo chmod 640 index.php »

Assurer que le propriétaire est debian1 et que le groupe est www-data pour le fichier index.php :

- « sudo chown debian1:www-data index.php »

Ces commandes nous ont permis d'ajuster les permissions de manière que le propriétaire du fichier soit l'utilisateur debian1, mais le groupe soit le groupe www-data, qui est généralement utilisé par Apache pour accéder aux fichiers.

Conclusion

Dans le cadre de ce projet de sécurisation d'un système Linux Debian 11 hébergeant un serveur et une application Web, notre groupe a mis en œuvre diverses mesures de sécurité. Après avoir mené des recherches approfondies pour identifier les vulnérabilités potentielles et les techniques d'attaque courantes, nous avons procédé à l'installation et à la configuration de plusieurs outils de sécurité. Ces mesures comprennent la mise à jour régulière du système, la surveillance des fichiers de logs, la gestion des mots de passe, la configuration de la connexion SSH par clé, l'utilisation de Fail2ban pour bloquer les attaques par force brute, la configuration d'UFW pour contrôler le trafic réseau, et la préparation de ModSecurity pour une éventuelle activation ultérieure. Nous avons pris des mesures supplémentaires telles que la génération de certificats SSL auto-signés pour le cryptage des échanges, la redirection vers HTTPS, et la désactivation de la connexion SSH en tant que root. En ajustant également les permissions des fichiers et répertoires sensibles, nous avons renforcé la sécurité de manière globale.
Ces actions ont été prises en vue de préparer le serveur à un test d'intrusion futur, dans le cadre du projet, afin de garantir sa robustesse et son intégrité face à d'éventuelles attaques.
Toutes ces tâches ont été pour nous sources d’apprentissage, en outre il est essentiel de sécuriser des systèmes informatiques de nos jours et surtout dans notre domaine de métier, cette mise en situation a pu nous initier à la sécurité informatique.

Sources

Les sources conseillées par notre enseignante :

- https://www.cert.ssi.gouv.fr
- https://www.ssi.gouv.fr/uploads/2019/02/fr_np_linux_configuration-v2.0.pdf
- https://www.ssi.gouv.fr/uploads/2013/05/anssi-guiderecommandations_mise_en_oeuvre_site_web_maitriser_standards_securite_cote_navigateurv2.0.pdf
- https://www.ssi.gouv.fr/uploads/2021/02/anssi-guide-tpe_pme.pdf

D’autres sources que nous avons consultées pour compléter nos recherches :

- https://owasp.org/www-project-top-ten/
- Les 10 principales vulnérabilités de l'OWASP - Check Point Software
- https://www.debian.org/doc/manuals/debian-faq/uptodate.fr.html
- https://www.vaadata.com/blog/fr/comment-renforcer-la-securite-de-vos-applications-web-pour-contrer-les-attaques-les-plus-courantes/
- https://raspberrytips.fr/securiser-raspberry-pi/
- https://www.root-me.org/
- https://doc.ubuntu-fr.org/fail2ban
- https://buzut.net/installer-et-parametrer-fail2ban/
- https://doc.ubuntu-fr.org/ufw
- https://www.debian.org/doc/