Réalisé par :

QUENTIN GAVILAN

RAPPORT

Renforcement De Sécurité Debian

24 avril 2023 – 30 juin 2023

SOMAIRE

INTRODUCTION

La sécurité informatique est une préoccupation majeure dans le contexte actuel, où les cyberattaques sont de plus en plus fréquentes et sophistiquées. Les entreprises et les organisations cherchent constamment des moyens d'améliorer la sécurité de leurs systèmes d'information pour protéger leurs données sensibles et prévenir les violations de sécurité. Dans ce contexte, le renforcement des systèmes d'exploitation est essentiel pour réduire les vulnérabilités et renforcer la sécurité globale.

Ce rapport présente le travail réalisé lors d'un projet de stage, dont l'objectif était de renforcer la sécurité d'une machine utilisant le système d'exploitation Debian utilisé par la suite par l'entreprise pour analyser le traffic réseau. En tant que stagiaire, ma mission consistait à concevoir le projet de bout en bout afin d'arriver à la hauteur de ce qui est demandé par le client pour cela j'ai donc suivi les recommandations de deux guides de sécurité, l'ANSSI (Agence nationale de la sécurité des systèmes d'information) et le manuel de sécurité de Debian, afin de mettre en place des mesures de sécurité appropriées.

Le projet s'est concentré sur plusieurs aspects clés de la sécurité du système, notamment, l'analyse et la suppression des services inutiles, la configuration de GRUB (le gestionnaire de démarrage), la sécurisation du noyau et système, ainsi que des mesures visant à renforcer la sécurité du réseau soit SSH ainsi que la connectivité et à configurer l'utilisation du système par les utilisateurs.

Ce rapport détaillera chaque étape du processus et présentera les mesures de sécurité mises en place, en expliquant leur importance et leur impact sur la sécurité globale du système d'exploitation Debian. De plus, nous discuterons des éventuelles limitations rencontrées et des recommandations pour améliorer davantage la sécurité du système.

MISE EN PLACE

Sécurisation du Démarrage Système

La sécurité du système au démarrage revêt une importance cruciale dans la protection globale d'un système informatique. Lorsque le système démarre, de nombreux processus et services s'exécutent dont les politiques de sécurités, ce qui en fait une phase critique où les vulnérabilités peuvent être exploitées par des attaquants.
On prend l'exemple d'une attaque de type "Evil Maid" qui consiste à manipuler le processus de démarrage en installant des logiciels malveillants ou en accédant aux données sensibles stockées sur le disque dur, un accès physique étant nécessaire ce type d'attaque est rendu complexe à mettre en œuvre dans notre cas.

En sécurisant le démarrage du système, on réduit considérablement les risques d'attaques et de compromission dès le départ. Malheureusement, dans le cadre du projet, l'accès au BIOS n'a pas été possible et la sécurisation de l'UEFI serait inutile car la machine en question est une machine virtuelle où seul l'administrateur y a accès.
Cependant, il est néanmoins nécessaire de charger certains modules et d'établir certains paramètres de sécurité pour renforcer la gestion de la mémoire et du processeur. Pour ce faire, nous devons configurer GRUB, un gestionnaire de démarrage. Nous accédons donc au fichier /etc/default/grub et y ajoutons ces paramètres :

- Grub_timeout= 0 (Pas de temps d'attente pour choisir le kernel a exécuté.)
- GRUB_CMDLINE_LINUX_DEFAULT="
  - iommu=force : Protège la mémoire système contre les accès arbitraires.
  - page_poison=on : Remplit les pages libérées avec des motifs et vérifie les motifs lors de l'allocation (pour détecter les fuites).
  - Pti=on : Force l'utilisation de la technique d'Isolation des Tables de Pages (Page Table Isolation).
  - slab_nomerge=yes : Désactive la fusion des caches slabs (allocation dynamique) compliquant les méthodes d'exploitation de la mémoire en cas de dépassement de tas
  - slub_debug=FZP :
    - F : Active les tests de cohérence des métadonnées des caches.
    - Z : Active la protection contre les dépassements (red zoning).
    - P : Active le remplissage avec des motifs empoisonnés des objets et du padding, provoquant une erreur lors de l'accès à ces zones

- spec_store_bypass_disable=seccomp : Force l'utilisation de contre-mesures pour la vulnérabilité Spectre V4.
- spectre_v2=on : Force l'utilisation de contre-mesures pour la vulnérabilité Spectre V2, évitant les attaques à prédiction de branches indirectes et les barrières de prédiction de branches indirectes pour les threads individuels.
- Mds=full,nomst : Force le système à utiliser Microarchitectural Data Sampling (échantillonnage de données micro-architecturales) pour atténuer les vulnérabilités d'Intel.
- Mce=0 : Force un kernel panic en cas d'erreurs non corrigées.
- page_alloc.shuffle=1 : Active l'allocation de pages de manière aléatoire, améliorant les performances.
- rng_core.default_quality=500 : Augmente la confiance dans le TPM (Trusted Platform Module) pour une initialisation rapide et robuste de Linux en utilisant la moitié de l'entropie qu'il fournit.
- Ipv6.disable=1 : Désactive l'ipv6 au démarrage pour le système.
- apparmor=1 : Active le support d'AppArmor, un outil de confinement des applications.
- security=apparmor : Renforce la sécurité en utilisant AppArmor comme mécanisme de sécurité.
- lsm=lockdown,yama,apparmor : Active différents mécanismes de sécurité, notamment le verrouillage du système (lockdown), Yama et AppArmor.
- lockdown=confidentiality : Améliore la confidentialité en restreignant l'accès aux informations sensibles.

Une fois cela fait on observe les différents paquets et modules installés grâce a la commande :

dpkg-query -l

On remarque que certains programmes ne sont pas utiles à notre utilisation comme :

Cron, eject, rsyslog, vim*, xauth, xxd

Qui peuvent contenir des failles de sécurité importantes que nous ne voulons pas gérer. Nous pouvons donc les supprimer du système avec la commande apt purge <paquet>

Sécurisation du Noyaux et Système

La sécurisation du noyau revêt une importance cruciale dans la protection globale d'un système d'exploitation, il est la partie centrale du système, responsable de la gestion des ressources matérielles et des communications entre les différents composants logiciels. Par conséquent, tout compromis au niveau de celui-ci peut entraîner des conséquences graves sur la sécurité et la stabilité du système. Afin de garantir sa sécurité des modifications lors de la compilation est indispensable, malheureusement la recompilation du noyau est un choix délicat à prendre car bien qu'elle puisse offrir des avantages en termes de sécurité elle présente également des inconvénients. L'un des principaux inconvénients est la perte des mises à jour de sécurité fournies par la distribution Debian. Ainsi, dans le cadre de ce projet, la décision a été prise de ne pas recompiler le noyau et de se fier aux mises à jour de sécurité régulières.

Cependant, cela ne signifie pas que nous ne pouvons pas améliorer la sécurité du noyau et du système. Au lieu de recompiler entièrement le noyau, nous avons la possibilité de modifier certaines options en utilisant le fichier sysctl.conf, situé dans le répertoire /etc.

Le fichier sysctl.conf est utilisé pour configurer divers paramètres système, y compris ceux liés à la sécurité. Nous pouvons ajouter des options spécifiques afin de renforcer la sécurité du système en tenant compte des CVE (Common Vulnerabilities and Exposures) identifiées au fil des années.

Ci-dessous, nous présentons quelques exemples d'options que nous pouvons ajouter dans le fichier sysctl.conf pour sécuriser davantage le système :

Configuration du noyau :

Restreint acces au buffer dmesg

kernel.dmesg_restrict=1

#Cache l'adresse noyaux et interface dans /proc

kernel.kptr_restrict=2

#Compte pid max supporté par le noyaux

kernel.pid_max=1000

#Interdit acces non privilégie a l'appel sys perf event open

kernel.perf_event_paranoid=2

#Restreint l'utilisation du ssystem perf

kernel.perf_cpu_time_max_percent=1

kernel.perf_event_max_sample_r

#Active ASLR

kernel.randomize_va_space=2

#Desactive les combinaisons de touche magic

kernel.sysrq=0

#Restreint l'usage du BPF

kernel.unprivileged_bpf_disabled=1

#Arrete le sys en cas de comportement inatendu du kernel

kernel.panic_on_oops=1

#Active la config LSM yama 3 aucun proc peut utiliser ptrace

kernel.yama.ptrace_scope=3

#Desactive les modules au chargement du kernel

kernel.module.disable=1

Configuration du Réseau :

#Attenue l'effet dispersion JIT

net.core.bpf_jit_harden=2

#Empeche le routage de paquet

net.ipv4.ip_forward=0

#Paquets reçu avec comme ip src en 127.0.0.1/28 drop

net.ipv4.conf.all.accept_local =0

#Refuse des paquets ICMP redirect

net.ipv4.conf.all.accept_redirects=0

net.ipv4.conf.default.accept_redirects=0

net.ipv4.conf.all.secure_redirects=0

net.ipv4.conf.default.secure_redirects=0

net.ipv4.conf.all.shared_media=0

net.ipv4.conf.default.shared_media=0

#Refuse les infos des entêtes de source route

net.ipv4.conf.all.accept_source_route =0

net.ipv4.conf.default.accept_source_route =0

#Empeche la gestion de la table ARP

net.ipv4.conf.all.arp_filter =1

#Repond au paquet ARP si l'ip src et dst sont sur le LAN

net.ipv4.conf.all.arp_ignore =2

#Refuse le routage de paquet dont ip src et dst du LAN

net.ipv4.conf.all.route_localnet =0

#Drop les paquets gratuitous ARP

net.ipv4.conf.all. drop_gratuitous_arp =1

#Verifie l'interface réseau du paquet src

net.ipv4.conf.default.rp_filter =1

net.ipv4.conf.all.rp_filter =1

#Empeche la redirection de paquet

net.ipv4.conf.default.send_redirects =0

net.ipv4.conf.all.send_redirects =0

#Ignore le paquet non conforme à RFC 1222

net.ipv4.icmp_ignore_bogus_error_responses =1

#Augmente la plage pour les ports éphémère évite donc les conflits de ports

net.ipv4.ip_local_port_range =32768 65535

#Limite les connexion TCP simultané entre 2 hôtes pour limiter l'analyse de séquence

net.ipv4.tcp_rfc1337 =1

#Haute disponibilité des SYN TCP en cas d'attaque ddos sur les connexions TCP

net.ipv4.tcp_syncookies =1

#Desactive IPV6

net.ipv6.conf.default.disable_ipv6=1

net.ipv6.conf.all.disable_ipv6=1

Configuration système :

#Desactive la création de coredump pour les exe setuid

fs.suid_dumpable=0

#Interdit l'ouverture des FIFOS qui ne sont pas la propriété de l'utilisateur

fs.protected_fifos=2

fs.protected_regular=2

#Restreint la création de lien symbolique au fichier dont utilisateur n'est pas propriétaire

fs.protected_symlinks=1

#Restreint la création de lien dure a des fichiers dont l'utilisateur n'est pas propriétaire

fs.protected_hardlinks=1

Sécurisation des Dossiers Système

L'importance de la sécurisation des dossiers /tmp et /var/tmp réside dans le fait qu'ils sont souvent utilisés comme emplacements temporaires pour les fichiers et données par de nombreuses applications et services. Ces dossiers temporaires peuvent présenter des risques de sécurité s'ils sont mal configurés ou exploités par des attaquants. Pour renforcer la sécurité de ces dossiers, voici les actions entreprises :

```
#Créer un fichier tmp de 5 GB dans la partition /tmp
        sudo dd if=/dev/zero of=/usr/tmpDSK bs=1024 count=5024000

#Créer une sauvegarde du dossier /tmp
        sudo cp -Rpf /tmp /tmpbackup

#Monter la nouvelle partition /tmp
        sudo mount -t tmpfs -o loop,noexec,nosuid,rw /usr/tmpDSK /tmp
        sudo chmod 1777 /tmp

#Copier les données de la sauvegarde dans le dossier
        sudo cp -Rpf /tmpbackup/* /tmp/
        sudo rm -rf /tmpbackup/*

#Ajouter le montage automatique du dossier tmp dans /etc/fstab
        /usr/tmpDSK /tmp tmpfs loop,nosuid,noexec,rw 0 0
        sudo mount -o remount /tmp

#On créer un lien symbolique vers le fichier que l'on a créée
        sudo mv /var/tmp /var/tmpold
        sudo ln -s /tmp /var/
        sudo cp -prf /var/tmpold/* /tmp/
        sudo rm -rf /var/tmpold/
On reboot pour appliquer les changements
```

Sécurisation Utilisateur Système

Une fois que nous avons mis en place les options pour sécuriser la partie matérielle du système, il est essentiel de se pencher sur la sécurité des utilisateurs et des applications.
Pour cela, nous devons effectuer les actions suivantes :

- Création de partitions distinctes pour un meilleur contrôle d'accès et de montage des répertoires : Il est recommandé de créer plusieurs partitions distinctes afin de pouvoir appliquer des options spécifiques à chaque partition, telles que l'interdiction d'exécution de certains fichiers ou la gestion des permissions. Cependant, dans le cadre de ce projet, nous n'avons pas prévu de partitionnement spécifique et nous ne gérons pas les cas de partitionnement.

- Verrouillage des comptes et désactivation du shell pour certains logins : Nous verrouillons tous les comptes des services, à l'exception des comptes principaux, en modifiant le fichier /etc/passwd et en remplaçant /bin/bash par /nologin. De cette manière, nous empêchons la possibilité de se connecter à ces comptes. Il est important de mettre en place cette mesure pour limiter les accès non nécessaires.

- Journalisation de toutes les créations de processus : Nous utilisons le service d'audit pour journaliser les actions effectuées par les processus. Cependant, il est essentiel de noter que la journalisation de chaque action entreprise par les processus peut générer une quantité considérable de données de journalisation. Pour gérer ces logs de manière efficace, il est recommandé d'utiliser des logiciels spécialisés et de les exporter vers un service de gestion des logs externe. Dans le cadre de ce projet, aucune action spécifique n'a été entreprise en raison de contraintes de temps et de ressources.

- Modification de l'umask pour bloquer les accès inappropriés : Nous modifions l'umask en utilisant la valeur 077. Cette modification restreint les permissions d'accès lors de la création de fichiers ou de répertoires par un utilisateur, en accordant des privilèges uniquement au propriétaire. Pour cela, nous ajoutons la ligne "umask 077" dans le fichier /etc/profile.

- Configuration du fichier sudoers pour gérer les droits des utilisateurs sudo : Nous éditons le fichier /etc/sudoers pour définir les droits des utilisateurs sudo. Nous ajoutons les règles suivantes :

```
Defaults noexec, requiretty, use_pty, umask=0077, ignore_dot, env_reset
Defaults:%sudo !noexec
Defaults:root !noexec
```

Ces règles spécifient que l'exécution de programmes via sudo est interdite (noexec), que l'exécution de sudo doit se faire à partir d'un terminal (requiretty), que l'allocation d'un pseudo terminal est autorisée pour l'exécution de sudo (use_pty), que les variables d'environnement héritées par l'utilisateur sont limitées (ignore_dot), et que l'umask est défini à 0077 pour limiter les permissions d'accès. Ces mesures renforcent la sécurité des utilisateurs et réduisent les risques liés à l'utilisation de sudo.

Mais nous remarquons que la règle noexec bloque l'exécution de sudo même pour les roots on peut donc ouvrir cette règle au root et au groupe sudo en utilisant les règles :

Defautls:%sudo !noexec

Defaults:root !noexec

- Les mises à jour automatiques jouent un rôle crucial dans le maintien sécurisé du système. Pour faciliter cette tâche, Debian propose l'outil "unattended-upgrades", qui permet de mettre à jour le système quotidiennement. Nous procédons à l'installation de ce package et vérifions que les paramètres utilisés correspondent à nos attentes.

- Nous désactivons la connexion au compte root en utilisant les commandes suivantes :
  Pour verrouiller le compte root :

  sudo passwd -l root

  Pour réactiver le compte root :

  sudo passwd -u root

#timesyncd sert a syncro le time avec d'autre time assure donc une secuirté du temps

- Nous configurons les règles TCP Wrappers pour contrôler l'accès aux services utilisant ce mécanisme. Par exemple, nous utilisons les règles suivantes :
  Pour rejeter toutes les demandes provenant de toutes les adresses IP :

  ALL: PARANOID

  Pour autoriser certaines adresses IP à se connecter en TCP :

  ALL: 192.168.0.59

  ALL: 192.168.56.11

- Nous bloquons l'accès des nouveaux utilisateurs en modifiant les paramètres suivants :
  Remplacer shell= par SHELL=/usr/sbin/nologin dans le fichier /etc/default/useradd
  Remplacer DSHELL= par DSHELL=/usr/sbin/nologin dans le fichier /etc/access.conf
  Ces actions limitent l'accès des utilisateurs à une simple connexion en empêchant l'exécution de shell interactif, renforçant ainsi la sécurité du système.

Sécurisation d'authentification PAM

Plugable Application Module est un système de gestion des connexions à des services tels que SSH ou le processus de connexion. Il offre la possibilité de renforcer la sécurité de certains services en utilisant ses fonctionnalités.
Nous créons un groupe appelé "wheel" et configurons le fichier "su" de manière à exiger que les utilisateurs appartiennent au groupe "wheel" pour pouvoir exécuter la commande "su". Pour cela, nous ajoutons la règle suivante au fichier "su" :

```
auth requisite pam_wheel.so group=wheel debug
```

Dans le fichier "other", nous mettons en place les règles suivantes pour renforcer la sécurité :

```
#Restreindre l'accès des utilisateurs à des terminaux spécifiques tty
        auth required pam_securetty.so
# Vérifier la validité des identifiants en se référant au fichier /etc/passwd, afficher un
  message d'avertissement en cas d'erreur et refuser l'authentification
        auth required pam_unix_auth.so
        auth required pam_warn.so
        auth required pam_deny.so

#Vérifier les informations des comptes (expiration, etc.), afficher un message
  d'avertissement en cas d'erreur et bloquer l'accès au compte en cas d'échec
        account required pam_unix_acct.so
        account required pam_warn.so
        account required pam_deny.so

#Gérer la validité du mot de passe, afficher un message d'avertissement lors de la
  validation du mot de passe et refuser la connexion en cas d'échec :
        password required pam_unix_passwd.so
        password required pam_warn.so
        password required pam_deny.so

#Gérer les tâches spécifiques à la session utilisateur, afficher un message d'avertissement
  et bloquer la création de session en cas d'échec du module :
        session required pam_unix_session.so
        session required pam_warn.so
        session required pam_deny.so
```

Dans le fichier "login", nous ajoutons la ligne suivante pour mettre en place un délai de 1 minute en cas d'échec :

auth required pam_faillock.so deny=3 unlock_time=60

Dans le fichier "common-password", nous avons précédemment ajouté la bibliothèque PAM Cracklib pour définir des règles lors de la création de mots de passe, mais cela s'est avéré inutile. Dans le cadre du projet nous avons optés pour la suppression de cette ligne qui modifiait le chiffrement du mot de passe. Cependant, j'ai ajouté la directive "round=11" pour renforcer la sécurité :

password [success=1 default=ignore] pam_unix.so obscure sha512 rounds=11

Dans le fichier "common-session", nous ajoutons la modification de l'UMASK via PAM :

session optional pam_umask.so umask=0077

Dans le fichier "sshd", nous sécurisons SSH en supprimant la possibilité de se connecter avec le mot de passe de l'utilisateur :

@include common-auth

Ensuite, nous ajoutons l'authentification à double facteur avec TOTP :

auth required pam_google_authenticator.so

En modifiant les fichiers de configuration liés à PAM, tels que "login", "common-password" et "common-session", nous avons pu mettre en place différentes règles de sécurité. Cela inclut la gestion des verrouillages après un certain nombre d'échecs d'authentification, la définition des politiques de mots de passe et la modification de l'UMASK par session.

En ce qui concerne la sécurisation de SSH, nous avons pris des mesures supplémentaires en désactivant la connexion par mot de passe et en introduisant l'authentification à double facteur avec TOTP (Time-based One-Time Password). Cela renforce considérablement la sécurité des connexions SSH en ajoutant une couche supplémentaire de protection.

Pour le prochain chapitre, nous aborderons l'ouverture de la sécurisation de SSH.

Sécurisation du service SSH

Secure Shell (SSH) est un protocole de communication sécurisé largement utilisé pour l'accès à distance aux serveurs et la gestion sécurisée des systèmes. Il offre des fonctionnalités telles que l'authentification sécurisée, le chiffrement des données et la confidentialité des communications. Dans ce chapitre, nous allons nous concentrer sur la sécurisation de SSH en configurant le fichier "sshd_config" pour renforcer la sécurité de nos connexions SSH.

```
#Modification du port
        Port 30778
#Activation de la journalisation de l'authentification
        SyslogFacility AUTH
#Temps accordé à l'utilisateur pour se connecter
        LoginGraceTime 2m
#Empêche la connexion en tant que root
        PermitRootLogin no
#Activation du mode strict pour vérifier la configuration utilisateur et les autorisations
        StrictModes yes
#Nombre maximal d'échecs d'authentification
        MaxAuthTries 3
#Nombre maximal de sessions simultanées par utilisateur
        MaxSessions 3
#Autorisation de connexion avec une clé publique
        PubkeyAuthentication yes
#Interdiction de l'authentification par mot de passe utilisateur
        PasswordAuthentication no
#Interdiction des mots de passe vides
        PermitEmptyPasswords no
#Activation de l'authentification par réponse challenge en utilisant les options configurées dans le fichier PAM
        ChallengeResponseAuthentication yes
        UsePam yes
#Interdiction du transfert TCP
        AllowTcpForwarding no
#Désactivation du transfert X11 pour éviter les failles connues
        X11Forwarding no
```

```
#Désactivation de l'affichage du message du jour lors de la connexion
        PrintMotd no
#Affichage du dernier journal de connexion lors de la connexion SSH
        PrintLastLog yes
#Les utilisateurs ne peuvent pas définir de variables d'environnement
        PermitUserEnvironment no
#Réduction des vulnérabilités de sécurité en désactivant la vérification des noms DNS
        UseDNS no
#Renforcement de la sécurité en utilisant uniquement le protocole 2
        Protocol 2
#Accord de privilèges réduits aux sessions pour sécuriser le système
        UsePrivilegeSeparation yes
#Méthodes d'authentification autorisées
        AuthenticationMethods publickey,keyboard-interactive
```

Après avoir modifié le fichier sshd pour renforcer au maximum sa sécurité, il est maintenant nécessaire de créer une paire de clés SSH.

Les clés utilisant l'algorithme de chiffrage edDSA offrent une sécurité supérieure à celle de RSA, suivies de Kerberos, TOTP et des mots de passe. Afin d'établir une connexion asymétrique chiffrée, il est recommandé de privilégier l'utilisation de clés. Pour ce faire, l'utilisateur doit générer une clé privée et une clé publique sur son propre hôte, puis envoyer la clé publique au serveur en demandant l'approbation de l'administrateur. Les commandes suivantes peuvent être utilisées pour effectuer ces opérations :

```
ssh-keygen -o -t ed25519
ssh-copy-id lisa@ip
```

En conclusion, les modifications apportées à SSH ont permis de restreindre les connexions distantes au système, garantissant ainsi une meilleure sécurité. Désormais, seules les personnes disposant de la paire de clés appropriée et du code TOTP sont autorisées à accéder au système. Cela renforce considérablement la protection du système contre les accès non autorisés et les attaques potentielles.

Configuration du Pare-Feu

Le pare-feu, également connu sous le nom de firewall, est une mesure de sécurité informatique qui contrôle le flux des données entre un réseau privé et un réseau public, tel qu'Internet. Il agit comme une barrière de protection en filtrant et en contrôlant le trafic réseau en fonction de règles prédéfinies. On remarque que suite à l'utilisation de l'option kernel.disable.module=1 dans sysctl.conf, de nombreux modules essentiels à l'utilisation du pare-feu sont désactivés. Afin de les activer au démarrage, nous devons spécifier les modules nécessaires dans le fichier /etc/modules. Après des recherches approfondies pour identifier les modules utiles à iptables, les modules suivants ont été ajoutés :

#Servant à la création de table permettant de rendre le pare feu utilisable

nfnetlink nf_tablesnft_compatnft_counter x_tables ip_tables

#Servant à l'utilisation de module spécifiques donnant des options à iptables

libcrc32c xt_tcpudp xt_owner xt_state

#Modules à lancer au démarrage dans le but de sécurisé le système

yama apparmor

Par la suite on attribue les règles iptables du pare-feu :

- iptables -A INPUT -s 192.168.0.0/22 -p tcp --dport 30778 -m state –state NEW,ESTABLISHED -j ACCEPT
- iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
- iptables -A OUTPUT -d 192.168.0.0/22 -p tcp --sport 30078 -m state --state ESTABLISHED -j ACCPET
- iptables -A OUTPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
- iptables -A OUTPUT -p tcp --dport 53 -m state --state NEW -j ACCEPT
- iptables -A OUTPUT -p udp --dport 53 -m state --state NEW -j ACCEPT
- iptables -A OUTPUT -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
- iptables -P INPUT -j DROP
- iptables -P FORWARD -j DROP
- iptables -P OUTPUT -j DROP

Ajout supplémentaire de sécurité

APPARMOR

AppArmor est un LSM (Linux Security Module) qui permet de limiter les droits d'un utilisateur sur le système en fonction des profils associés aux processus. Dans notre cas, l'utilisation d'AppArmor n'est pas essentielle, mais nous pouvons l'activer pour bloquer certains types d'accès de base. Pour activer AppArmor au démarrage, nous avons modifié le fichier /etc/default/grub en ajoutant apparmor=1 et security=apparmor tout en lançant le module dans /etc/module.

CHROOT :

Une fois toutes les configurations effectuées on s'attarde sur chroot permettant isoler un service de la racine dans notre cas créer un chroot pour un service n'est pas utile car on ne possède aucun service mais on peut faire en sorte d'isoler un utilisateur de la racine lors de sa connexion en ssh

Pour cela on :

#créer un dossier isoler

mkdir chroot

#utilise debootstrap pour télécharger une image réduite de debian dans le chroot

debootstrap stable /chroot http://deb.debian.org/debian

Ensuite on paramètre ssh pour qu'il utilise chroot dans le cas de connexion d'un user

Il faut comprendre que ici debootsrap utilise un système centralisé mais on peut le créer nous même de toute pièce afin de personnaliser notre service en fonction de ses besoin

Ancien module :

Le fichier "/etc/modprobe.d/blacklist.conf" est utilisé pour mettre en place une liste noire (blacklist) des modules qui ne doivent pas être utilisés sur un système. Cette liste noire est créée afin de bloquer l'utilisation de certains modules qui peuvent présenter des risques de sécurité ou qui sont obsolètes.

Les modules répertoriés dans le fichier "blacklist.conf" sont généralement des anciens protocoles ou des modules connus pour avoir des failles de sécurité exploitables. En les désactivant, on réduit les risques associés à leur utilisation.

On écris donc dans le fichier blacklist.conf ceci :

```
#Thunderbolt et Firewire peut etre utiliser pour une attaque DMA
        install firewire-core /bin/true
        install thunderbolt /bin/true
#Blacklist les protocols reseaux
        install dccp /bin/true          install sctp /bin/true
        install rds /bin/true           install tipc /bin/true
        install n-hdlc /bin/true        install ax25 /bin/true
        install netrom /bin/true        install x25 /bin/true
        install rose /bin/true          install decnet /bin/true
        install econet /bin/true        install af_802154 /bin/true
        install ipx /bin/true           install appletalk /bin/true
        install psnap /bin/true         install p8023 /bin/true
        install llc /bin/true           install p8022 /bin/true
#Block uncommon filesystems
        install cramfs /bin/true        install freevxfs /bin/true
        install jffs2 /bin/true         install hfs /bin/true
        install hfsplus /bin/true       install squashfs /bin/true
        install udf /bin/true
#Blacklist the webcam,bluetooh,ipv6,usb kernel module
        install uvcvideo /bin/true
        install btusb /bin/true
        install bluetooth /bin/true
        install ipv6 /bin/true
        install usb-storage /bin/true
```

AUTORISATION :

Lors de la gestion d'un système, il est crucial de s'assurer que les autorisations sur les dossiers sensibles sont correctement configurées. Les autorisations déterminent qui peut accéder, modifier ou exécuter des fichiers et des répertoires, et une mauvaise configuration peut compromettre la sécurité du système.

On installe l'outil d'audit yasat permettant de vérifier nos droits d'accès :

install yasat et utiliser yasat -p system_rights.test

sudo chown root:root /etc/default/grub

sudo chown root:root /etc/shadow gshadow shadow-

sudo chmod go-rwx /etc/default/grub
sudo chmod 644 /etc/group
sudo chmod 644 /etc/passwd-
sudo chmod 400 /etc/gshadow
sudo chmod 640 /etc/logrotate.conf
sudo chmod 440 /etc/sudoers
sudo chmod 600 /root/.profile
sudo chmod 600 /etc/security
sudo chmod 650 /etc/cron.d/e2scrub_all
sudo chmod o-rxw /boot
sudo chmod 644 /etc/passwd
sudo chmod 600 /etc/shadow
sudo chmod 600 /etc/shadow-
sudo chmod 640 /etc/sysctl.conf
sudo chmod 640 /etc/fstab
sudo chmod 600 /root/.bashrc
sudo chmod 600 /var/log/faillog
sudo chmod 600 /etc/crontab
sudo chmod o-rwx /etc/cron.daily/*
sudo chmod og-rxw /home/lisa
sudo chmod 700 /root /home /boot /usr/lib/modules

Lors de la sécurisation d'un système, il est important de restreindre l'utilisation de certaines commandes sensibles par les utilisateurs standards. Une étape cruciale consiste à empêcher les utilisateurs ordinaires d'exécuter la commande "su", qui permet de changer d'utilisateur et d'acquérir des privilèges élevés.

Pour réaliser cette restriction, nous utilisons la commande "dpkg-statoverride" avec les paramètres appropriés. Plus précisément, nous lançons la commande suivante :

dpkg-statoverride --update --add root wheel 4750 /bin/su

Cette commande modifie les attributs de sécurité du fichier "/bin/su" en lui attribuant les permissions spéciffiques permettant de restreindre l'accès à la commande "su" uniquement à l'utilisateur "root" et aux membres du groupe "wheel". Le mode d'exécution "4750" garantit que la commande "su" est exécutée avec les privilèges de l'utilisateur "root".

Ajout supplémentaire possible non effectué

Dans cette démarche de sécurisation du système, certaines actions n'ont pas été effectuées. Tout d'abord, aucune installation d'outil de détection d'erreurs ou de problèmes d'infiltration ou d'attaque n'a été réalisée. Ces outils permettraient de détecter les failles potentielles et les anomalies de sécurité. Cependant, cette étape a été omise et il revient au client d'installer les outils de son choix pour effectuer cette surveillance.

De plus, il n'y a pas eu d'activation de SELinux (Security-Enhanced Linux). SELinux est un mécanisme de sécurité pour les systèmes Linux qui met en œuvre des contrôles de sécurité avancés en utilisant des politiques de sécurité obligatoires. En activant SELinux, on renforcerait la sécurité globale du système en imposant des restrictions supplémentaires sur les actions des utilisateurs et des applications mais notre système n'a pas pour but à être utilisé par des utilisateurs autres que l'administrateur le choix a été pris de ne pas le paramétrer.

Gestion des logs le système n'a pas été configuré pour envoyer des logs a un services connus étant donné que seule le client possède ceci on laisse donc la liberté a celui-ci de paramétrer son système en fonction de ses normes.

CONCLUSION

Ce rapport a abordé les problématiques de sécurité d'un système et a présenté les mesures mises en place pour renforcer sa protection. L'objectif principal était d'améliorer la sécurité globale en mettant en œuvre des stratégies et des bonnes pratiques, ce qui a était effectué avec l'aide de 2 guides de la sécurité ANSSI Recommandation Hardening Linux/GNU et le Manuel de sécurité de Debian.
Les actions entreprises ont inclus la configuration d'un pare-feu robuste, la sécurisation des connexions SSH, la gestion des droits d'accès aux fichiers sensibles, la sécurisation de la mémoire, du noyau, du système et du réseau, ainsi que la désactivation des modules vulnérables. Ces mesures ont été appliquées avec succès, renforçant ainsi la sécurité du système.

Les résultats obtenus sont significatifs, avec une réduction des vulnérabilités et une amélioration globale de la sécurité. Les points forts de notre approche incluent l'utilisation de méthodes d'authentification solides, la mise en place d'un pare-feu bien configuré et la prise en compte des recommandations de sécurité.
Cependant, certains points pourraient être améliorés, tels qu'une excellente gestion des journaux avec un service dédié pour une protection supplémentaire, ainsi que l'utilisation d'outils de détection d'intrusion pour une surveillance continuent du système.

Il est important de souligner que la vigilance continue est essentielle dans le domaine de la sécurité informatique. Les menaces évoluent constamment, il est donc crucial de rester à jour, de surveiller les nouvelles vulnérabilités et de mettre régulièrement à jour les mesures de sécurité.

En conclusion, la sécurité des systèmes est une priorité absolue. Les mesures mises en œuvre dans ce rapport ont permis d'améliorer considérablement la sécurité du système, mais il est crucial de maintenir cette vigilance et de s'adapter aux nouvelles menaces. La sécurité informatique est un processus continu qui nécessite une attention constante et une adaptation aux évolutions du paysage de la sécurité.

SOURCES

ANSSI

Recommandations de Configuration d'un Système GNU/LINUX

https://www.ssi.gouv.fr/uploads/2019/02/fr_np_linux_configuration-v2.0.pdf

Manuel de sécurité Debian

https://www-debian-org.translate.goog/doc/manuals/securing-debian-manual

Book of Zeus

https://bookofzeus.com/harden-ubuntu/initial-setup