Sécuriser l'accès SSH

Commençons par **modifier le mot de passe** par défaut de l'utilisateur **root** et mettre à jour notre système

```
apt-get update
apt-get upgrade

# Puis pour changer de mot de passe:
passwd root
```

Créons un nouvel utilisateur afin de limiter l'accès à l'utilisateur root. Remplacer [myusername] par le nom d'utilisateur souhaité.

```
adduser myusername
```

Vous allez devoir ensuite remplir les diverses informations demandées. Vous pouvez laisser certaines informations vides. N'oubliez pas votre mot de passe !

On installe ensuite le paquet sudo pour permettre aux utilisateurs voulus de prendre les droits root *(Si ce n'est pas déjà le cas)*

```
apt install sudo
usermod -aG sudo myusername
```

On édite la configuration de notre SSH en: **changeant le port par défaut, bloquant la connexion via root, et en autorisant seulement le nouvel utilisateur.**

```
nano /etc/ssh/sshd_config
```

On modifie ensuite les valeurs suivantes:

```
Port 14009 # Choisissez un n° de port perso et non utilisé
PermitRootLogin no
# Temps à partir duquel le serveur arrête la connexion si elle n'aboutie pas
LoginGraceTime 30s
# Choix des utilisateurs autorisés à se connecter,
# ajouter l'user que l'on a créé
AllowUsers myusername myusername2 myusername3
```

On sauvegarde avec Ctrl+X puis on relance le SSH.

```
service ssh restart


# Ou sinon:


/etc/init.d/ssh restart
```

Si vous avez **UFW** (Uncompicated Firewall) pensez à **autoriser le nouveau port SSH**. Sinon vous pouvez faire le choix de l'installer afin de ne pas avoir à manipuler les `iptables` ou `nftables` qui peuvent être plus compliqués. C'est ce que nous ferons ici.

`UFW` est une interface de gestion de pare-feu simplifiée. Pour l'installer:

```
apt install ufw
```

Pour activer l'IPV6 (Il se fait de manière automatique normalement mais rien ne coute de vérifier):

```
nano /etc/default/ufw
```

Puis on édite:

```
IPV6=yes
```

On refuse toutes les connections entrantes et on autorise les sortantes:

```
ufw default deny incoming
ufw default allow outgoing
```

On autorise le **nouveau port SSH** ici `14009` (Remplacer par le votre) et **on active UFW**:

```
ufw allow 14009/tcp
ufw enable
```

Puis on active UFW, un avertissement qui indique que la commande peut interrompre la connexion SSH apparaîtra. On peut cependant continuer sans s'en soucier comme nous avons précédemment autorisé la connexion SSH dans notre parfeu.

Tout en gardant notre fenêtre actuelle connectée en SSH on essaye de se connecter à notre nouvel utilisateur sur notre nouveau port dans un nouveau terminal

```
ssh USERNAME@IP_ADRESS -pPORT
```

Une fois connecté en SSH si tout fonctionne vous pouvez retourner en root via la commande `su` pour continuer ce tutoriel. Il est désormais également possible d'utiliser `sudo ma_commande_admin`

Voilà pour le SSH ! *(Il est également possible de se connecter via un système de clé privée)*

Petit plus pour UFW (Vous pouvez également consulter la documentation qui sera plus complète)

Pour vérifier le status et règles de ufw:

```
ufw status
```

Pour désactiver ufw:

```
ufw disable
```

Ajouter une règle:

```
# Autoriser une connexion entrante
ufw allow [règle]
# Refuser une connexion entrante
ufw deny [règle]
```

Afficher les règles de manière numérotées:

```
ufw status numbered
```

Supprimer une règle il existe différentes manières de faire:

```
ufw delete [numéro de la règle]
ufw delete allow [règle]
ufw delete deny [règle]
```

Exemple ouverture du port 53 en TCP et UDP:

```
ufw allow 53
```

Exemple ouverture du port 53 en TCP uniquement:

```
ufw allow 53/tcp
```

Suppression de la règle précédente:

```
ufw delete allow 53/tcp
```

Remettre les paramètres par défaut:

```
ufw reset
```

Pour une utilisation avancée veuillez consulter la documentation :). Il est également possible de ne pas utiliser UFW et directement manipuler les iptables ou nftables.

Des astuces qui pourraient vous être utiles:

Demander une phrase différente de prompt lors d'une commande sudo:

Editez le fichier de configuration sudo `/etc/sudoers`

```
sudo visudo
```

Puis on rajoute les options suivantes:

```
# Lors d'un mauvais password un petit pic de sudo
Defaults insults
# Une phrase de prompt custom pour éviter le phishing
Defaults passprompt="[%p][sudo] prompt your password: "
```

Demander le mot de passe sudo à chaque commande:

Vous pouvez exiger systématiquement le mot de passe lors d'une commande sudo (par défaut, il y a un timeout de quelques minutes après la première utilisation correcte). Pour cela il suffit d'ajouter ceci dans `/etc/sudoers`:

Editez le fichier de configuration sudo `/etc/sudoers` et ajoutez y:

```
sudo visudo


# Ajoutez cette ligne, vous pouvez ajuster le timeout selon vos besoins (le nombre est en
minute)
Defaults        timestamp_timeout=0
```

Demander un autre mot de passe lors de la commande sudo:

Par exemple, si vous souhaitez demander le mot de passe root pour sudo pour X raisons:

```
sudo visudo
```

A la suite des lignes contenant le tag `Defaults` ajoutez le flag suivant:

```
Defaults     rootpw
```

Et hop le tour est joué le mot de passe root sera demandé lors d'une commande sudo plutôt que celui de l'utilisateur courant !

Restreindre l'utilisation de su

Nous pouvous faire en sorte que seul certains utilisateurs puissent utiliser la commande su. Pour cela créons un nouveau groupe `wheel` s'il n'existe pas déjà.

```
groupadd --system wheel


# Puis on ajoute les utilisateurs qui auront accès à su
usermod -aG wheel YOUR_TRUSTED_USER
```

Pour terminer il suffit ensuite de modifier deux fichiers: `/etc/pam.d/su` et `/etc/pam.d/su-l` et d'y rajouter la ligne suivante:

```
auth required pam_wheel.so use_uid
```

La surface d'attaque `su` est ainsi réduite.

Revision #18
Created 2021-01-28 09:48:55 UTC by Faces
Updated 2022-07-16 18:15:32 UTC by Faces