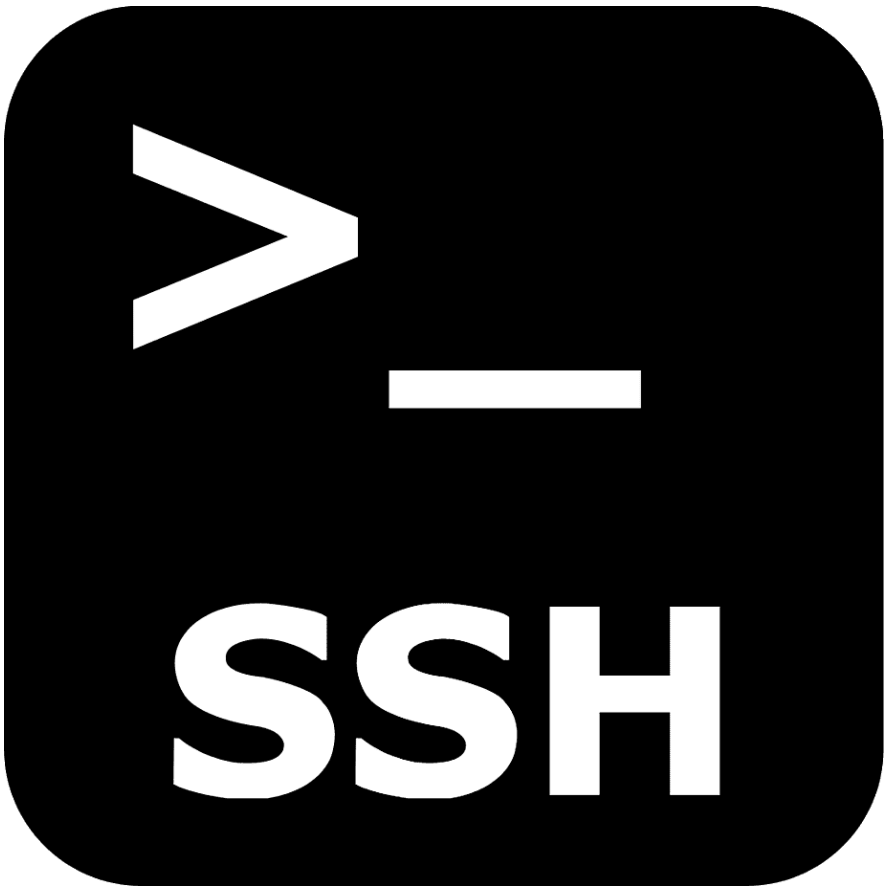

Secure Shell (SSH) est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient donc impossible d'utiliser un sniffer pour voir ce que fait l'utilisateur.

Le protocole SSH a été conçu avec l'objectif de remplacer les différents protocoles non chiffrés comme rlogin, telnet, rcp et rsh.

SOMMAIRE

Configuration du SSH

Pour accéder au fichier de configuration nous effectuons cette commande :

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

Dans ce fichier nous allons modifier les lignes suivantes :

```
Port 487

…

PermitRootLogin no

…

PubkeyAuthentication yes

…

PasswordAuthentication no

…
```

Actuellement nous avons modifier le port, nous avons désactiver la connexion avec le user « root », nous avons activer l'authentification par clés SSH et nous avons désactiver l'authentification par mot de passe.

Pour appliquer ces modifications nous allons devoir redémarrer le service SSH, avec cette commande :

```
sudo systemctl restart ssh
```

Création d'une clé SSH

Nous allons maintenant passer à la génération d'une clé SSH pour le serveur, pour faire ça nous devons effectuer cette commande :

```
ssh-keygen -t ed25519
```

Ce qui nous donneras ça :

```
Generating public/private ed25519 key pair.
Enter file in which to save the key ($HOME/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in $HOME/.ssh/id_ed25519.
Your public key has been saved in $HOME/.ssh/id_ed25519.pub.
```

Pour avoir accès au serveur, nous devons enregistrer la clé publique de notre ordinateur, en accédant à ce fichier :

```
sudo nano $HOME/.ssh/authorized_keys
```

Une fois dans ce fichier, ajoutez votre clé SSH :

```
ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIJtYkeI+Apm1CjhzwUoV2+1O94ccDsDYKX2ltKcisADy
bts@pc1
```

Configuration des notifications de connexions SSH sur Slack:

Slack est une une plate-forme de communication collaborative assez chouette à

utiliser.

Avant de « travailler » coté serveur(s), vous allez devoir créer un Wek Hook sur Slack.

```
https://votre-domaine.slack.com/apps/manage/custom-integrations
```

Une fois la manipulation effectuée, récupérez l'URL du Web Hook et conservez-la

Maintenant il s'agira de se connecter à toutes les machines qui devront envoyer des notifications, et de créer un script

```
sudo nano /etc/ssh/notify.sh
```

Ajoutez le code suivant en remplacant « YOUR_SLACK_WEBHOOK_URL » par l'URL récupérée sur Slack

```
#!/bin/sh

if [ "$PAM_TYPE" != "close_session" ]; then

 url="YOUR_SLACK_WEBHOOK_URL"

 channel="#ssh-logins"

 host="`hostname`"
```

```
 content="\"attachments\": [ { \"mrkdwn_in\": [\"text\", \"fallback\"], \"fallback\": \"SSH login: $PAM_USER connected to \`$host\`\", \"text\": \"SSH login to \`$host\`\", \"fields\": [ { \"title\": \"User\", \"value\": \"$PAM_USER\", \"short\": true }, { \"title\": \"IP Address\", \"value\": \"$PAM_RHOST\", \"short\": true } ], \"color\": \"#F35A00\" } ]"

 curl -X POST --data-urlencode "payload={\"channel\": \"$channel\", \"mrkdwn\": true, \"username\": \"ssh-bot\", $content, \"icon_emoji\": \":computer:\"}" $url &

fi
```

Rendez le script exécutable avec la commande suivante :

```
sudo chmod +x /etc/ssh/notify.sh
```

Puis modifier le fichier /etc/pam.d/sshd en y ajoutant le code suivant (en fin de fichier convient parfaitement) :

```
session optional pam_exec.so seteuid /etc/ssh/notify.sh
```

Vous pouvez maintenant recharger la configuration du serveur SSH et effectuer un test en vous connectant

```
sudo systemctl restart ssh
```

Installation et configuration de Portsentry

Pour installer le paquet lancez la commande suivante

```
sudo apt-get install portsentry
```

Lors de l'installation vous serez prévenu que par défaut portsentry ne bloque rien. Il faudra donc modifier les fichiers de configuration, mais tout d'abord occupons nous des hôtes qui seront ignorés afin de ne pas se faire soit bloquer même

```
sudo nano /etc/portsentry/portsentry.ignore
```

Ajoutons la liste des ips que vous souhaitez ne jamais bloquer

```
# IPs from /etc/portsentry/portsentry.ignore.static:


127.0.0.1/32
```

```
# dynamically fetched IPs(via ifconfig -a):

127.0.0.1

# mes IPs

xxx.xxx.xxx.xxx
```

Nous pouvons maintenant nous occuper de la configuration de portsentry :

```
sudo nano /etc/default/portsentry
```

Ici nous allons activer les modes audp et atcp, Portsentry va vérifier les ports utilisés et automatiquement « lier » les ports disponibles. C'est l'options la plus efficace (« a » signifie avancé). Avec cette options, portsentry établit une liste des ports d'écoute, TCP et UDP, et bloque l'hôte se connectant sur ces ports, sauf s'il est présent dans le fichier portsentry.ignore. Veillez donc à ce que votre fichier se compose de la manière suivante :

```
# /etc/default/portsentry

#

# This file is read by /etc/init.d/portsentry. See the portsentry.8

# manpage for details.

#

# The options in this file refer to commandline arguments (all in lowercase)

# of portsentry. Use only one tcp and udp mode at a time.

#

TCP_MODE="atcp"

UDP_MODE="audp"
```

Modifions maintenant le fichier de configuration principal :

```
sudo nano nano /etc/portsentry/portsentry.conf
```

Mettez en place le blocage en modifiant la section Ignore options de la façon suivante :

```
##################

# Ignore Options #

##################

...

# 0 = Do not block UDP/TCP scans.

# 1 = Block UDP/TCP scans.

# 2 = Run external command only (KILL_RUN_CMD)



BLOCK_UDP="1"

BLOCK_TCP="1
```

Veillez ensuite à ce que la section dropping route soit bien configurée avec la ligne suivante décommentée :

```
KILL_ROUTE="/sbin/route add -host $TARGET$ reject"
```

Meme chose pour la section TCP Wrappers qui doit être configurée ainsi :

```
KILL_HOSTS_DENY="ALL: $TARGET$ : DENY"
```

Ajoutez la ligne suivante comme external commande :

```
KILL_RUN_CMD="/sbin/iptables -I INPUT -s $TARGET$ -j DROP && /sbin/iptables -I INP
UT -s $TARGET$ -m limit --limit 3/minute --limit-burst 5 -j LOG --log-level debub --log-
prefix 'Portsentry: dropping: '"
```

Vous pouvez redémarrer portsentry et qui vous protègera au mieux d'un scan de vos ports :

```
sudo service portsentry restart
```

Installation et configuration de fail2ban

On commence par installer le paquet.

```
sudo apt-get install fail2ban
```

Le fichier /etc/fail2ban/jail.conf contient l'ensemble des plugins que vous pouvez activer pour protéger les services de votre serveur.
Mais vous ne devez pas modifier ce fichier directement. Car lors des mises à jour de votre Debian, il peut être remplacé à tout moment avec une version plus récente.

```
sudo nano /etc/fail2ban/jail.conf
```

Fail2ban permet de surveiller et de protéger de nombreux services sur votre serveur : SSH, nginx, postfix, FTP...
La configuration de chaque service est contenu dans une section spécifique [sshd] pour le service SSH qui est le seul actif par défaut.

Pour activer la surveillance d'un service, il faut placer la variable enabled à true dans la section du service.
Si vous avez modifier le port par défaut de SSH (pour ne pas utiliser 22), pensez de l'indiquer ici dans la variable port.

Pour activer et configurer les plugins de votre choix, vous devez spécifier vos paramètres personnes dans le fichier /etc/fail2ban/jail.local

En fait, Fail2ban charge les configurations dans cet ordre :

/etc/fail2ban/jail.conf

puis /etc/fail2ban/jail.d/defaults-debian.conf

et enfin /etc/fail2ban/jail.local

On crée donc notre fichier de config perso:

```
sudo nano /etc/fail2ban/jail.local
```

Et on commence donc par coller le bout de configuration ci-dessous pour activer le plugin sshd et spécifier le numéro de port SSH si vous l'avez modifié (Attention 487 est mon numéro de port à moi!)

```
[sshd]
port = 487
enabled = true
```

Ce plugin surveille les tentatives de connexions SSH sur votre serveur.
Par défaut, si Fail2ban détecte 5 tentatives de connexion erronées durant les 600 dernières secondes (soit 10 minutes), l'IP ayant essayé de se connecter sera bloquée 600s (10 min).

Ces constantes sont définies de manières globales pour tous les plugins
section [DEFAULT] du fichier /etc/fail2ban/jail.conf

```
# "bantime" is the number of seconds that a host is banned.
bantime  = 10m

# A host is banned if it has generated "maxretry" during the last "findtime"
# seconds.
findtime  = 10m

# "maxretry" is the number of failures before a host get banned.
maxretry = 5
```

Mais depuis votre fichier jail.local, vous pouvez les sur-définir au niveau de la configuration d'un plugin.

```
[sshd]
enabled = true
# 1 jour de bannissement
bantime  = 1d
```

Ou de manière globale pour tous les plugins. Noté que j'ai spécifié 1 jour en secondes ici (86400 = 24 * 60 * 60) pour vous montrer une autre manière de configurer la durée.
Vous trouverez les différents formats utilisables sur cette page

```
[DEFAULT]
# 1 jour de bannissement pour tous les plugins
bantime = 86400
```

Je vous recommande d'exclure votre adresse IP personnelle du scope de Fail2ban. Cela vous évitera de vous faire bannir de votre propre serveur !
Pour cela, ouvrez à nouveau le fichier de configuration /etc/fail2ban/jail.local

```
sudo nano /etc/fail2ban/jail.local
```

Et ajouter la variable ignoreip dans la section [DEFAULT] du fichier.
Remplacez 123.456.789.123 par votre adresse IP (ou vos adresses).

```
[DEFAULT]

ignoreip = 127.0.0.1/8 123.456.789.123
```

Reste à redémarrer le servie fail2ban pour prendre en compte les modifications de la configuration.

```
sudo systemctl restart fail2ban
```