Sécuriser votre VPS Linux

Depuis quelques années et d'autant plus depuis la crise du COVID-19 nous utilisons de plus en plus Internet. Nos objets connectés accèdent à des services complexes demandant une grande capacité de calcul. Ces services utilisent la puissance d'ordinateurs dédiés, nos serveurs. Nos serveurs gèrent dans la plupart du temps une grande quantité de données potentiellement sensibles. Tout comme le nombre d'utilisateurs augmentent le nombre d'attaques pirates ne cessent de croître. À tout cela s'ajoute de nouveaux logicielles de plus en plus complexes, qui ne sont pas à l'abri d'une erreur de code laissant une opportunité aux pirates. Sécuriser notre serveur est donc essentiel !

1- Sécuriser son SSH

La plupart des attaques sont des attaques où des "robots" vont tenter de se connecter en SSH sur notre serveur avec l’utilisateur root en tentant des combinaisons différentes de mot de passe.
Il est donc essentiel de sécuriser l'accès à notre ssh.

[Update] Ce tutoriel fait l'usage de sudo pour effectuer des commandes root (c'est le plus répandu). Il est cependant possible de faire sans: [ici](https://wonderfall.space/retour-du-root-ssh/)

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.

SSH Keys

Vous pouvez si vous le souhaitez désactiver la connection par mot de passe et utiliser une clé SSH à la place, pour cela rien de plus simple.

Commençons par générer une paire de clés pour cela tapez sur le terminal de votre ordinateur local (MobaXTerm dans mon cas).

```
ssh-keygen -t ed25519


# Au premier prompt mettez le nom de fichier que vous voulez #pour votre clé.
# A la seconde question pour la passphrase vous pouvez ne rien mettre si vous ne souhaitez pas utiliser de mot de passe à chaque fois que vous vous connectez sur votre serveur.
```

Deux fichiers seront générés dont un contenant l'extension .pub, ce fichier contiendra votre clé publique. Le second contiendra votre clé privé.

Sur votre serveur allez ensuite dans le répertoire ~/.ssh/ de l'utilisateur auquel vous souhaitez vous connecter en utilisant cette clé.

Créez un fichier `authorized_keys` s'il n'existe pas déjà:

```
nano authorized_keys


# Puis copiez votre clé publique à l'intérieur directement vous pouvez en ajoutez plusieurs à la suite
```

Soyez sûr de bien sauvegarder votre clé privée et publique dans un endroit sûr. Continuons en désactivant l'authentification par mot de passe dans le fichier de configuration SSH.

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

Une fois dans le fichier de configuration cherchez les valeurs suivantes et modifiez les comme ci-dessous :

```
PubkeyAuthentication yes
# Retirez le .ssh/authorized_keys2 qui peut servir à certaines attaques
AuthorizedKeysFile      .ssh/authorized_keys
```

```
PasswordAuthentication no
UsePAM no
```

Il nous suffit ensuite de redémarrer le service ssh en utilisant la commande `sudo systemctl restart sshd`

Vous pouvez désormais vous connecter en utilisant votre clé assurez cependant de bien importer votre clé privée dans l'utilitaire que vous utilisez. Sous MobaXterm déposez les dans: `C:\Users\VOTRE_USER\Documents\MobaXterm\home\.ssh`

2- Rester informé sur l'activité du serveur

Un serveur mail nous permettra d'obtenir un retour sur ce qui se passe sur notre serveur et de rester informer sur les points importants sans devoir continuellement regarder les fichiers logs.

C'est quoi les logs ? Ils servent à quoi ?

Les applications créent des fichiers "logs" pour garder une trace de l'activité qui c'est passé à un moment donné. Ces fichiers sont loin d'être de simples sorties de texte et peuvent être très complexes à parcourir. En cas de panne d'un service, il devient vital d'utiliser toute l'aide disponible. Les logs permettent la plupart du temps d'analyser ce qui s'est exactement passé et permettent de trouver plus facilement une solution. Logwatch est un analyseur de log très puissant qui rend la vie plus facile. Un bon fichier journal doit être aussi détaillé que possible afin d'aider l'administrateur qui a la responsabilité de maintenir le système, à trouver les informations exactes nécessaires à une panne ou pour s'assurer d'un bon fonctionnement. De ce fait les journaux, logs, sont généralement peu concis et ils contiennent des tonnes de répitions qui nécessitent des analyses et un filtrage approfondis. C'est là que Logwatch entre en jeu.

Setup du service mail

FQDN

Pour cette étape vous devez posséder un nom de domaine qui pointe sur votre serveur (DNS) et connaître votre FQDN (Fully Qualified Domain Name
Pour connaître son FQDN:

```
hostname -f
```

Pour éditer son `hostname` il suffit d'éditer:

```
nano /etc/hostname
```

Puis **remplacer le hostname actuel** par celui voulu par exemple: `server-ex52` deviendra `toto` puis on sauvegarde. Le changement sera pris en compte au prochain reboot sinon vous pouvez le réactualiser avec la commande `sudo hostname toto` puis vérifier si le changement a été pris en compte avec la commande `hostname`.

Ensuite mettons en place le FQDN pour cela faite:

```
nano /etc/hosts
```

Modifier votre fichier dans notre cas on veut rester en local *(pensez à changer domaine.tld par votre nom de domaine)*:

```
127.0.0.1 toto.domaine.tld toto
127.0.0.1 localhost
```

A la fin il devrait ressembler à cela:

```
127.0.0.1 toto.domaine.tld toto
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
```

```
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Une fois vos changements sauvegardés vérifier votre FQDN: Il suffit de taper `hostname -f`

SERVEUR MAIL POSTFIX

Maintenons installons postfix, quelques questions nous serons posées:

```
apt-get install -f postfix mailutils
```

On répond aux questions suivantes posées par l'installation :

General type of mail configuration ? Internet Site
System mail name ? domaine.tld

Puis on relance la configuration complète du paquet :

```
dpkg-reconfigure postfix
```

General type of mail configuration ? Internet Site
System mail name ? domaine.tld
Root and postmaster mail recipient ? Laisser vide
Other destinations to accept mail for ? domaine.tld, localhost.domaine.tld, localhost
Force synchronous updates on mail queue ? No
Local networks ? Laisser par défaut 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
Mailbox size limit (bytes) ? 0
Local address extension character ? Laisser par défaut (+)
Internet protocols to use ? ipv4 *all si vous avez ipv6 de config*

Pour tout problèmes vous pouvez voir votre configuration en allant voir le fichier de configuration: `/etc/postfix/main.cf` et les logs sont disponibles sous `/var/log/mail.log`

Testons notre configuration:

```
echo 'Hello World ! Je suis un email du serveur.' | mail -s 'Hello World'
mon_adresse_email@mail.com
```

Le mail peut arriver dans les spams c'est tout à fait normal ne vous inquiétez pas. Nous allons maintenant rediriger les mails de root vers notre adresse mail. Pour cela:

```
nano /etc/aliases
```

Puis rajouter à la suite:

```
root: mon_adresse_email@mail.com
```

Puis pour valider les changements:

```
newaliases
```

Testons ensuite de nouveau:

```
echo 'Hello World ! Test root mail.' | mail -s 'Hello World' root
```

Afin de ne pas avoir Postfix qui écoute inutilement sur le port 25:
Allons éditer:

```
nano /etc/postfix/master.cf
```

et commenter la ligne suivante:

```
#smtp      inet  n       -       y       -       -       smtpd
```

On relance ensuite postfix:

```
service postfix restart
```

On va mettre ensuite en place un email lorsqu'un utilisateur accède au compte root pour cela:

```
nano /root/.bashrc
```

Puis rajouter à la fin du fichier:

```
echo 'Acces Shell Root le: ' `date` 'par' `who` | mail -s 'Connexion serveur via root' root
```

Il est également possible de faire pour tout les utilisateurs et donc pour chaque connexion ssh ! Pour cela il suffit de modifier le fichier `/etc/bash.bashrc`

On ajoute ensuite la ligne suivante à la fin du fichier:

```
echo 'Acces Shell le: ' `date` 'par' `who` | mail -s 'SSH: acces sur serveur: '`hostname` root
```

Voilà nous avons fini de configurer notre serveur mail en cas de problèmes.

Monitorer ses Logs avec Logwatch

Installation de Logwatch

Les rapports créés par Logwatch sont classés selon les services (Applications) exécutés sur votre système. Les applications peuvent être définies dans le fichier de configuration de logwatch. De plus, Logwatch permet la création de scripts d'analyse personnalisés pour des besoins spécifiques

Installation du paquet:

```
apt install logwatch
```

S'il n'existe pas déjà créez le dossier `/var/cache/logwatch` nécessaire au bon fonctionnement de logwatch :

```
mkdir /var/cache/logwatch
```

Configuration de Logwatch

On va modifier le fichier de configuration se trouvant à l'adresse `/usr/share/logwatch/default.conf/logwatch.conf`, ouvrons le avec l'éditeur de notre choix.

Pour cela on va créer une copie du fichier de configuration par défaut

```
cp /usr/share/logwatch/default.conf/logwatch.conf /etc/logwatch/conf/logwatch.conf
```

Puis on l'édite:

```
nano /etc/logwatch/conf/logwatch.conf
```

On observera une longue liste de variables que l'application utilise lors de son éxécution.
Nous modifierons ou ajusterons les valeurs suivantes:

```
# Le type d'output ici on veut recevoir nos logs par mail
Output = mail
```

```
# Pour recevoir les mails au format html, c'est plus agréable à lire
Format = html
# Adresse sur laquelle vous allez recevoir les mails
MailTo = root # <= Correspondra a l'adresse précédemment définis pour root dans le setup du service mail
# Niveau de détail des logs Low | Med | High
Detail = Med

# Pour les services vous pouvez gardez la ligne Service = All mais si vous souhaitez
# recevoir seulement des rapports spécifiques vous pouvez lister chaque service comme ci dessous
# (en retirant le #). Nous garderons All dans notre cas

# Service = sendmail
# Service = http
# Service = identd
# Service = sshd2
# Service = sudo
```

Ajoutons un rapport de log pour Nginx (Si vous utilisez Nginx of course)

Pour commencer on va créer une configuration spécifique:

```
nano /etc/logwatch/conf/logfiles/nginx.conf
```

Puis nous allons copier dedans la configuration ci dessous:

```
#########################################################
# Define log file group for nginx
#########################################################

# What actual file? Defaults to LogPath if not absolute path….
LogFile = nginx/*access.log
LogFile = nginx/*access.log.1
LogFile = nginx/*error.log
LogFile = nginx/*error.log.1

# If the archives are searched, here is one or more line
# (optionally containing wildcards) that tell where they are…
#If you use a “-” in naming add that as well -mgt
Archive = nginx/*access.log*
```

```
Archive = nginx/*error.log*

# Expand the repeats (actually just removes them now)
*ExpandRepeats

# Keep only the lines in the proper date range…
*ApplyhttpDate

# vi: shiftwidth=3 tabstop=3
```

On crée ensuite le second fichier essentiel pour faire correspondre notre service *(en se basant sur la conf http existante)*:

```
cp /usr/share/logwatch/default.conf/services/http.conf /etc/logwatch/conf/services/nginx.conf
```

On l'ouvre ensuite avec notre éditeur préféré:

```
nano /etc/logwatch/conf/services/nginx.conf
```

Puis on modifie le début du fichier:

```
Title = “nginx”
LogFile = nginx
```

Et pour finir on copie le fichier du script:

```
cp /usr/share/logwatch/scripts/services/http /etc/logwatch/scripts/services/nginx
```

Puis on relance logwatch vous devriez recevoir un mail récapitulatif et par la suite un mail journalier:

```
logwatch restart
```

Et voilà logwatch vous permettra de garder une trace de ce qu'il se passe sur votre serveur quotidiennement !
Pour modifier un service reprenez l'exemple ci dessus et modifiez `http` et `nginx` par le service correspondant à votre besoin.

3 - Mise en place de Fail2Ban

Fail2Ban est le must have et permet de bannir les machines qui tentent de pirater votre serveur. Il se base sur les logs du serveur et en fonctions de règles établies afin de bannir les tentavies d'accès non autorisées.

Setup de Fail2Ban

Commençons par installer le paquet de fail2ban:

```
apt-get install -f fail2ban
```

On crée un fichier `jail.local` qui nous servira a paramétrer les différentes "prisons" en cas d'attaques.

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Puis on édite le fichier pour mettre nos paramètres de sécurités.

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

On modifiera les valeurs suivantes:

```
# L'ip avec laquelle vous accédez au serveur habituellement si fixe
ignoreip = 127.0.0.1/8 XXX.XXX.XXX.XXX  # <= REMPLACER ICI !

# Le nombre de temps qu'un utilisateur sera banni par défaut si il enfreint une règle
bantime = 7d # -1 est un ban permanent 7d équivaut à 7 jours

# La période de temps ou les "maxretry" seront comptabilisés
findtime = 1d

# maxretry le nombre d'essais ratés avant qu'un utilisateur soit banni
maxretry = 5

# Les emails de fail2ban sont envoyés sur l'email défini pour l'utilisateur root
destemail = root

# On change la ligne: action = %(action_)s pour recevoir un mail lors d'un ban
action = %(action_mwl)s

# Activation de fail2ban sur le ssh
[sshd]
```

```
enabled = true
port = 14009 # Votre port SSH
logpath = %(sshd_log)s
backend = %(sshd_backend)s
maxretry = 3 # Si on veut override le maxretry par défaut

# port = les ports à bloquer au moyen des règles iptables
# logpath = l'emplacement des fichiers de log à surveiller
# backend = le moteur de surveillance des logs.

################################################
# SI VOUS AVEZ INSTALLE ET ALLEZ UTILISER NGINX AVEC CLOUDFLARE
# utilisez real_ip_header CF-Connecting-IP; dans vos host !!
[nginx-auth]
# Désactiver en mettant false si Nginx n'est pas installé
enabled  = true
port  = http,https
filter   = nginx-auth
# Chemin vers les error log peut varier si vous utilisez npm
logpath  = /var/log/nginx/*error.log
banaction = iptables-multiport

[nginx-badbots]
# Désactiver en mettant false si Nginx n'est pas installé
enabled  = true
port  = http,https
filter = nginx-badbots
# Chemin vers les access log peut varier si vous utilisez npm
logpath = /var/log/nginx/*access.log
banaction = iptables-multiport
################################################

# Si le ban par défaut n'est pas permanent vous pouvez ajouter la jail recidive
# Si le pirate récidive le temps de ban sera augmenté
[recidive]

enabled = true
logpath  = /var/log/fail2ban.log
banaction = %(banaction_allports)s
```

```
bantime  = 3w
findtime = 7d

# Il vous suffit ensuite de rajouter "true" sur les services que vous voulez surveillez par la suite,
# et rajoutez au besoin les services absents
```

On va ensuite modifier la valeur de `dbpurge` pour l'ajuster, pour cela nous allons éditer le fichier de configuration fail2ban.

```
nano /etc/fail2ban/fail2ban.conf
```

Puis nous allons ajuster les différentes lignes si besoin:

```
# Pour éviter une boucle infini si le loglevel est en DEBUG lorsque la jail recidive est activée
loglevel = INFO

# On ajuste la valeur de dbpurge
dbpurgeage = 3w
```

Une fois cela fait et sauvegardé, nous allons rajouter nos filtres pour Nginx en accord avec les jails créés précédemments.

!! Utilisateurs Cloudflare !!

On notera que si vous utilisez Cloudflare en proxy vous ne verrez pas les vraies IPs des visiteurs. De ce fait vous devez utiliser `real_ip_header CF-Connecting-IP;` dans vos proxy host afin de récupérer la vraie IP !

Pour le premier filtre de la jail `nginx-auth` :

```
nano /etc/fail2ban/filter.d/nginx-auth.conf
```

Puis on copie colle dans le fichier sans modifier:

```
## FICHIER /etc/fail2ban/filter.d/nginx-auth.conf ##
[Definition]

failregex = no user/password was provided for basic authentication.*client: <HOST>
            user .* was not found in.*client: <HOST>
            user .* password mismatch.*client: <HOST>
```

```
ignoreregex =
```

Pour le second filtre de la jail `nginx-badbots`. Il servira a détecter les bots qui cherchent des fichiers de configuration vulnérables et failles sur votre serveur nginx:

```
nano /etc/fail2ban/filter.d/nginx-badbots.conf
```

Puis on copie colle dans le fichier sans rien modifier:

```
# Fail2Ban configuration file
# Author: Patrik 'Sikevux' Greco <sikevux@sikevux.se>

[Definition]

# Option: failregex
# Notes.: regex to match access attempts to setup.php
# Values: TEXT

failregex = ^<HOST> .*?"GET.*?\/setup\.php.*?" .*?

# Anti w00tw00t
            ^<HOST> .*?"GET .*w00tw00t.* 400

# try to access to directory
            ^<HOST> .*?"GET .*admin.* 403
            ^<HOST> .*?"GET .*admin.* 404
            ^<HOST> .*?"GET .*install.* 404
            ^<HOST> .*?"GET .*dbadmin.* 404
            ^<HOST> .*?"GET .*myadmin.* 404
            ^<HOST> .*?"GET .*MyAdmin.* 404
            ^<HOST> .*?"GET .*mysql.* 404
            ^<HOST> .*?"GET .*websql.* 404
            ^<HOST> .*?"GET .*webdb.* 404
            ^<HOST> .*?"GET .*webadmin.* 404
            ^<HOST> .*?"GET \/pma\/.* 404
            ^<HOST> .*?"GET .*phppath.* 404
            ^<HOST> .*?"GET .*admm.* 404
            ^<HOST> .*?"GET .*databaseadmin.* 404
            ^<HOST> .*?"GET .*mysqlmanager.* 404
```

```
            ^<HOST> .*?"GET .*phpMyAdmin.* 404
            ^<HOST> .*?"GET .*xampp.* 404
            ^<HOST> .*?"GET .*sqlmanager.* 404
            ^<HOST> .*?"GET .*wp-content.* 404
            ^<HOST> .*?"GET .*wp-login.* 404
            ^<HOST> .*?"GET .*typo3.* 404
            ^<HOST> .*?"HEAD .*manager.* 404
            ^<HOST> .*?"GET .*manager.* 404
            ^<HOST> .*?"HEAD .*blackcat.* 404
            ^<HOST> .*?"HEAD .*sprawdza.php.* 404
            ^<HOST> .*?"GET .*HNAP1.* 404
            ^<HOST> .*?"GET .*vtigercrm.* 404
            ^<HOST> .*?"GET .*cgi-bin.* 404
            ^<HOST> .*?"GET .*webdav.* 404
            ^<HOST> .*?"GET .*web-console.* 404
            ^<HOST> .*?"GET .*manager.* 404
# Option: ignoreregex
# Notes.: regex to ignore. If this regex matches, the line is ignored.
# Values: TEXT
#
ignoreregex =
```

Il ne nous reste plus qu'a redémarrer fail2ban :) ! Vous recevrez plusieurs mails par la même occasion mais pas de panique rien de très important cela sera plus calme par la suite.

Une fois tout cela fait on redémarre fail2ban

```
systemctl restart fail2ban
```

On vérifie que le service est bien lancé:

```
systemctl status fail2ban
```

Quelques Utilitaires pour fail2ban:
Pour voir les jails actives on peut utiliser:

```
fail2ban-client status
```

Pour voir le status d'une jail en particulier il suffit d'utiliser son nom (exemple avec `sshd`):

```
fail2ban-client status sshd
```

Pour débannir un client remplacer les fields `NOM_DE_LA_JAIL` et `IP_A_DEBANNIR`, ne pas oublier de débannir de la jail `recidive` si elle est active:

```
fail2ban-client set NOM_DE_LA_JAIL unbanip IP_A_DEBANNIR
```

(Optionnel)
Vous pouvez personnaliser vos messages d'alertes dans `/etc/fail2ban/action.d/` en modifiants ces différents fichiers:
Prenez le temps de lire les fichiers ils sont en anglais mais relativement simple à comprendre et personnaliser

```
nano /etc/fail2ban/action.d/sendmail.conf
nano /etc/fail2ban/action.d/sendmail-whois.conf
nano /etc/fail2ban/action.d/sendmail-whois-lines.conf
```

Une fois tout cela fait on redémarre fail2ban

```
systemctl restart fail2ban
```

Voilà votre magnifique douanier Fail2Ban est actif !

4 - Allez plus loin

Même si la protection ultime n’existe pas, il est toujours possible de protéger encore plus son serveur. Cependant plus on ajoute de protections plus on ajoute de contraintes. Il est donc essentiel d'évaluer votre besoin et de ne pas surcharger votre machine de sécurités inutiles.

Nous verrons ici:

- Garder son serveur à jour contre les failles de sécurité avec cron-apt

- RkHunter pour détecter les backdoors et rootkits ainsi que d'autres problèmes de sécurité.

Et une analyse du traffic en temps réel pour prévenir d'attaques:

- Portsentry, une solution pour se protéger des scans de ports. (Snort peut être une alternative mais ne sera pas traité ici)

Garder son serveur à jour avec cron-apt

Les logiciels de plus en plus complexent comportent parfois de nombreuses failles de sécurités qui ne seront découvertes qu'avec le temps. C'est pourquoi mettre son système à jour régulièrement est important. Mais comment faire si une faille de sécurité est découverte alors que vous êtes en vacances ? Malade ? Ou tout, simplement, n'avez pas de quoi accéder au serveur ?
L'avantage d'un serveur, c'est que tout peut être automatisé ! Il peut donc se mettre à jour tout seul ! Nous allons voir comment avec cron-apt. Il existe également d'autres solutions.

Installation de cron-apt

```
apt install cron-apt
```

Pour ne pas tout casser lors de certaines mise à jour on souhaite ne faire que les mises à jour de sécurité !
On crée un nouveau fichier source en redirigeant la sortie de `grep`:

```
grep security /etc/apt/sources.list > /etc/apt/security.sources.list
```

Configurons ensuite cron-apt:

```
nano /etc/cron-apt/config
```

Puis on y colle à la suite:

```
APTCOMMAND=/usr/bin/apt-get
# Le path vers le fichier source que l'on viens de créer
OPTIONS="-o quiet=1 -o Dir::Etc::SourceList=/etc/apt/security.sources.list"
# L'email pour avertir de la mise à jour MAILTO="mon_email@mail.com" ou ici root
# en accord avec l'alias que nous avons créé précédemment
MAILTO="root"
# Quand envoyer l'email au sujet des résultats de cron-apt:
# Value: error   (send mail on error runs)
#□ upgrade (when packages are upgraded)
#□ changes (mail when change in output from an action)
```

```
#  output  (send mail when output is generated)
#        always  (always send mail)
#                (else never send mail)
MAILON="upgrade"
```

On vérifie que la ligne `dist-upgrade -d -y -o APT::Get::Show-Upgraded=true` est présente:

```
nano /etc/cron-apt/action.d/3-download
```

Puis on la modifie en retirant le flag -d signifiant Download only. Si elle n'est pas présente on l'ajoute à la fin de notre fichier.

```
# Vérifier que la ligne est bien décommentée et présente sinon la rajouter. Sauvegarder puis quitter
dist-upgrade -y -o APT::Get::Show-Upgraded=true
```

Et voilà c'est fait ! `cron-apt` se lance par défaut toute les nuits à 4h du matin. Il est possible de modifier cela en éditant le fichier `/etc/cron.d/cron-apt`.

Voilà ! Votre serveur se mettra à jour automatiquement sur le point de vue de la sécurité. Cependant cela ne dispense pas de ne jamais vérifier son bon fonctionnement.

Rkhunter (Optionnel)

Rkhunter peut détecter les répertoires généralement utilisés par les rootkits. Les permissions anormales, les fichiers cachés, les chaînes suspectes dans le kernel et peut effectuer des tests spécifiques à Linux. Cela se fait en comparant les hashs de vos fichiers avec des hashs de virus et logiciels connus.

Installation de RKHunter

```
apt install rkhunter -y
```

Chercher ensuite la ligne ou il y a `WEB_CMD="/bin/false"` vous pouvez utiliser Ctrl+W sur nano. Une fois la ligne trouvée commentez la.

```
# WEB_CMD="/bin/false"
```

Puis la ligne ou il y a écris `UPDATE_MIRRORS` et mettez sa valeur à 1. Cela spécifie que le fichier mirroir doit être vérifié pour les mise à jours lors d'une update.

```
UPDATE_MIRRORS=1
```

Puis la ligne `MIRRORS_MODE` et mettre sa valeur à 0. Cela permet de spécifier à rkhunter quel mirroir utiliser lors d'une update.

```
MIRRORS_MODE=0
```

Une fois l'étape précédente terminé et enregistré on va rajouter les notifications par mail:
On ouvre le fichier situé sous `/etc/default/rkhunter` avec notre éditeur préféré:

```
nano /etc/default/rkhunter
```

Puis on modifie si besoin les variables suivantes:

```
# Pour effectuer une vérification chaque jour
CRON_DAILY_RUN="yes"
# L'adresse sur laquelle on veut recevoir les emails, ici celle associé à root
REPORT_EMAIL="root"
```

Vérifions que notre fichier de configuration est correct avec la commande:

```
rkhunter -C
```

Pour mettre à jour rkhunter vous pouvez ensuite utiliser la commande:

```
rkhunter --update
```

Puis pour vérifier le système local

```
rkhunter --check
```

Rkhunter peut avoir **de faux positifs** suite à une mise à jour par exemple dans ce cas, il faut mettre la base d'empreintes à jour avec la commande :

```
rkhunter --propupd
```

Ce sera tout pour RKhunter vous pouvez également voir les logs sous `/var/log/rkhunter.log` :)

Portsentry contre les scans de ports

Chaque minute de nombreuses tentatives d'intrusion sur des serveurs sont perpétrés par des pirates. Pour trouver une cible et avant de l'attaquer ils vont passer par une phase de reconnaissance. Certaines de ces phases sont parfois automatisées par des machines zombies. Le scan de port à l'aide de logiciel comme nmap permet de détecter les ports ouverts amenant à de potentielle exploits. Pour prévenir de tels scans rien de mieux que d'analyser le traffic vers notre serveur à l'aide de portsentry !

Nb: Portsentry ne bloque rien par défaut il se contente de logger les scans de votre serveur. Il faut le configurer, ce que l'on va tout de suite faire en commençant par white-lister certaines IP

Installation de portsentry

```
apt-get install portsentry
```

Dans cet exemple nous allons whitelist le range d'IP de Google et les nôtres :

```
nano /etc/portsentry/portsentry.ignore.static
```

Puis ajouter dans le fichier

```
# IP du serveur
x.x.x.x
# Votre IP maison si fixe par sécurité
x.x.x.x
# Plage d'IP Google
66.249.64.0/19
```

Il nous faut ensuite **modifier le fichier de configuration** pour que portsentry puisse bloquer des IPs ce qu'il ne fait pas par défaut.

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

A) On commence par modifier les variables suivantes:

```
##################
# 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"
```

B) On va modifier ensuite la section *Dropping routes*
Chercher et vérifier que la lignes suivante est bien décommentée:

```
# Newer versions of Linux support the reject flag now. This
# is cleaner than the above option.
KILL_ROUTE="/sbin/route add -host $TARGET$ reject"
```

C) La section *TCP Wrappers*
Chercher et vérifier que les lignes suivantes sont bien décommentées:

```
###############
# TCP Wrappers#
###############

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

D) Et ajouter dans la partie *External Command* :

```
###################
# External Command#
###################
KILL_RUN_CMD_FIRST = "1"

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

Pour finir on change la variable dans la partie *Scan trigger value* :

```
SCAN_TRIGGER = "1"
```

Il vaut mieux utiliser le mode **atcp** et **audp** pour une détection automatique des ports utilisés, il faut donc éditer le fichier `/etc/default/portsentry` :

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

Et on modifie:

```
# /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"
```

Puis on redémarre Portsentry et on lui permet de démarrer à chaque redémarrage de notre serveur:

```
systemctl restart portsentry
systemctl enable portsentry
```

Pour jeter un œil aux IPs bloqués.

```
cat /etc/hosts.deny
```

Pour avoir plus d'infos sur le port qui a déclenché le blocage + date/heure vous pouvez voir les différents fichiers dans le répertoire:

```
cd /var/lib/portsentry/
```

Utilitaires

Pour dé-bannir un user bloqué par erreur, cherchez son IP dans `/etc/hosts.deny` puis redémarrez portsentry. Si malgrès cela l'utilisateur reste bloqué vous pouvez tenter:

```
route del -host IP reject
```

Port Knocking

Cela permet de bloquer l'accès au réel port SSH à moins qu'une séquence spécifique de ports ne soit "knock". Ce n'est qu'à ce moment que les règles iptables permettront au port SSH d'être ouvert à l'adresse IP de celui qui aura knock cette séquence de port.

Pour utiliser UFW à la place des iptables allez directement à la fin ! (UFW)

1 - Installation des packages requis

Commençons par installer le service requis:

```
apt-get install knockd
```

Vous aurez peut être besoin d'installer le package `iptables-persistent`. Il permet de charger de manière automatique les règles `iptables` sauvegardées.

```
apt-get install iptables-persistent


# Save current IPv4 rules ? YES
# Save current IPv6 rules ? YES
```

2-Configuration avec IPTABLES

Avant de mettre en marche knockd nous allons modifier certains paramètres par défauts.
Editons le fichier de configuration:

```
nano /etc/knockd.conf
```

Modifiez ou supprimez ce qui s'y trouve afin de mettre les settings suivantes:

```
[options]
        UseSyslog

[openSSH]
        sequence    = 1555,8888,5555
        seq_timeout = 5
        command     = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
```

```
        tcpflags    = syn
        cmd_timeout = 15
        stop_command = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
```

- `sequence` correspond à la séquence de port à knock dans les 5 secondes.
- `seq_timeout` le temps imparti pour knock les ports
- `command` la commande exécuté pour ouvrir le port ssh à notre IP, Soyez sûr de bien remplacer 22 par votre port SSH.
- `tcpflags` Spécifie qu'il n'acceptera que des segments tcp
- `cmd_timeout` le temps avant que notre IP soit supprimé des iptables, ici, 15 secondes. Cela ne nous déconnectera pas de la session SSH une fois connectée mais préviendra de vous connecter sans knock les ports spécifiés.
- `stop_command` La commande pour supprimer l'accès au port SSH via votre adresse IP. Soyez sûr de bien remplacer 22 par votre port SSH.

Continuons en rajoutant une règle *iptables* afin que les personnes connectées en SSH ne soit pas déconnectées:

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Puis fermons l'accès au port SSH **(Remplacez 22 par le votre)**:

```
iptables -A INPUT -p tcp --dport 22 -j REJECT
```

Démarrons le daemon `netfilter-persistent` associé avec iptables et faisons en sorte qu'il sauvegarde nos règles iptables:

```
systemctl start netfilter-persistent
netfilter-persistent save
netfilter-persistent reload
```

(`sudo netfilter-persistent save` à utiliser à chaque changement de votre firewall que vous voulez sauvegarder)

3 - Mise en route de Knockd

Continuez en allant modifier le fichier `/etc/default/knockd` et mettez `START_KNOCKD` égal à 1.
NB: Vous aurez peut être besoin de changer la command line options afin de mettre le nom de votre interface réseau. Vous pouvez le savoir en tapant "ip addr" par défaut eth1
Redémarrez ensuite knockd `systemctl start knockd`.

Pour vous connectez depuis linux il vous suffit d'installer knockd comme précédemment et d'utiliser la commande:

```
knock -v IP PORT1 PORT2 PORT3 ...
```

Afin d'ouvrir le port SSH. Sous windows vous trouverez de nombreux projets de port knocking disponible sur github facilement installable. Comme: `https://github.com/BetterWayElectronics/port-knocker/releases/tag/1.0` pour le moment (compilez le vous même, on sait jamais).

Pour voir de l'intérieur ce que cela donne une fois que vous avec knock votre port vous pouvez utiliser:

```
tail -f /var/log/syslog
```

(PLEASE READ ME !)- Knockd Bug Not Starting at Boot

Il est probable sur certaines versions de debian comme la 9 (ça m'ai arrivé sur la 11 également □□) que knockd ne redémarre pas au reboot ce qui peut être TRES problèmatique sur un serveur remote. Si cela arrive vous pourrez sûrement vous connecter via la console en ligne de votre host provider.

Pour éviter que cela ce produise vous pouvez réaliser ces étapes:

Identifiez si nous avons le problème:

```
systemctl is-enabled knockd.service
```

cela nous retournera alors `static` !

Pour fix vous allez ensuite ajouter une section [Install] dans le fichier `/lib/systemd/system/knockd.service`:

```
nano /lib/systemd/system/knockd.service
```

Puis rajoutez à la fin:

```
[Install]
WantedBy=multi-user.target
Alias=knockd.service
```

Sauvegardez et activez knockd au démarrage:

```
systemctl enable knockd.service
```

Si vous relancez la commande `systemctl is-enabled knockd.service` vous devriez cette fois avoir la réponse `enabled` !

Hop le tour est joué !

Solutions issu de: https://bugs.debian.org

(UFW) - Si vous utilisez UFW (Ignore this if you used iptables)

Si vous utilisez UFW rien de plus simple commencez par installer le service `knockd`

```
apt-get install knockd
```

Editez le fichier de configuration de la manière suivante:

```
nano /etc/knockd.conf
```

Modifiez ou supprimez ce qui s'y trouve afin de mettre les settings suivantes et remplacez le port 22 par votre port SSH pour la signification des settings remontez à la section **2-Configuration avec IPTABLES**:

```
[options]
        UseSyslog

[openSSH]
        sequence    = 1555,8888,5555
        seq_timeout = 5
        command     = ufw allow from %IP% to any port 22
        tcpflags    = syn
        cmd_timeout = 15
        stop_command = ufw delete allow from %IP% to any port 22
```

Supprimez toute règle déjà mise en place autorisant le traffic depuis votre port SSH. Ces dernières seront gérés directement par UFW.
Puis fermez votre port SSH *(Remplacer 22 par votre port SSH)*.

```
ufw insert 1 deny from any to any port 22
```

Continuez à partir de la section **3- Mise en route de Knockd** !

et voilà :) si jamais n'hésitez pas à me contacter par discord ou mail *(disponible sur mon site)* :)