Sécuriser son serveur Linux

Marche à suivre pour sécuriser son serveur Linux et connaître les menaces

Référence : EF-SEC-LINUX

Auteur(s) :

Dorian Manzanares

Alex Falzon

Destinataire(s) :

Easyformer

Date de modification : 05/08/22

Version : 1

Sommaire

page

1 Filtrer le trafic grâce au pare-feu

Toutes les manipulations suivantes seront réalisées sur un terminal avec une distribution Debian (Debian, Ubuntu…)

1.1 Installation et configuration de Iptables

Le firewall (pare-feu en français) est l'élément indispensable pour sécuriser son serveur. Il va en effet filtrer tout le trafic en n'autorisant que les échanges permis par l'administrateur. Sans firewall correctement réglé, tous les trafics sont plus ou moins permis (c'est-à-dire qu'un attaquant peut faire ce qu'il veut chez vous) et ce genre de faille est détectable par un simple scan de ports.

Le noyau Linux offre déjà un pare-feu à l'utilisateur, qu'il est possible de configurer via le logiciel iptables (normalement contenu dans /sbin/iptables).

L'installer :

```
apt install iptables
```

Nous allons maintenant détailler le fonctionnement d'un firewall - relativement simple. Un firewall analyse tout le trafic et vérifie si chaque paquet échangé respecte bien ses règles (critères de filtrage), donc, il suffit de spécifier de bonnes règles pour interdire tout trafic superflu.

Les critères peuvent être divers (filtrer les ports, les protocoles, les adresses IP, etc). De base, nous allons spécifier nos règles sur les ports. Bien entendu, il faut être le plus strict possible quant au choix des règles ; c'est pourquoi, par défaut, tout firewall se règle en premier lieu en bloquant tout, absolument tout. Ensuite, nous allons « ouvrir » (autoriser le trafic) certains ports que nous voulons utiliser (par exemple pour un serveur web, nous allons ouvrir le port 80 afin que le site web soit accessible).

Attention :

Nous allons écrire nos règles sous forme de script bash, petite mesure de prudence si vous êtes loggué sur votre machine à distance (ssh), soyez bien sûr de ne pas vous bloquer l'accès ou - le cas échéant, de pouvoir rebooter la machine.

1.1.1 Création des règles du pare-feu

Vous pourrez à tout moment consulter les règles courantes à l'aide de la commande suivante

```
iptables -L
```

Tout d'abord, on s'occupera de la création du script

```
nano /etc/init.d/firewall
```

Il faudra ajouter le texte suivant en tout début de script :

```
# !/bin/bash
```

Il faudra effacer les règles précédentes afin de repartir de zéro

```
iptables -t filter -F iptables -t filter -X
```

Puis on pourra créer les règles suivantes (elles vont bloquer tout le trafic par défaut, à ne pas exécuter de suite si vous êtes connectés en SSH)

```
iptables -t filter -P INPUT DROP iptables -t filter -P FORWARD DROP iptables -t
filter -P OUTPUT DROP
```

On précise maintenant que les connexions établies resteront ouvertes, à l'aide de -m et –state

```
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT iptables -A
OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

On autorise notre adresse loopback

```
iptables -t filter -A INPUT -i lo -j ACCEPT
iptables -t filter -A OUTPUT -o lo -j ACCEPT
```

Maintenant que tout le trafic est bloqué, nous allons pouvoir ouvrir les ports dont nous aurons besoin.

Observons plus en détail les paramètres de iptables

• -t : vaudra par défaut « filtrer »

• -A : servira à indiquer le sens du trafic : INPUT (entrant) ou OUTPUT (sortant)

• -p : indique le protocole (TCP ou UDP en principe)

• --dport et --sport : respectivement port destination et port source (comme nous sommes le serveur, nous utiliserons principalement dport)

• -j : comment traiter le paquet (nous nous servirons d'ACCEPT et de DROP pour respectivement accepter et refuser le paquet).

Nous allons maintenant ajouter une première règle

```
iptables -t filter -A INPUT/OUTPUT -p protocole --dport port_a_ouvrir -j ACCEPT
```

Dans le cas ou on possède un serveur Web

```
iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT iptables -t filter -A
INPUT -p tcp --dport 80 -j ACCEPT
```

Notez l'importance d'ouvrir le port dans les deux sens avec OUTPUT et INPUT

Cas particulier du ping :

Le ping est basé sur un protocole particulier (ICMP) qui n'a pas de port prédéfini, nous allons autoriser le ping car c'est la méthode la plus couramment utilisée pour savoir si votre serveur est en vie.

```
iptables -t filter -A INPUT -p icmp -j ACCEPT iptables -t filter -A OUTPUT -p icmp
-j ACCEPT
```

1.1.2 Activation du pare-feu

Il est maintenant temps de démarrer notre pare-feu

```
chmod +x /etc/init.d/firewall
/etc/init.d/firewall
```

Vous pouvez maintenant tester votre configuration, et vous pouvez notamment utiliser « nmap » pour vérifier les ports ouverts.

Nous allons maintenant faire en sorte de lancer le script au démarrage du système update-rc.d firewall defaults

90 % des problèmes informatiques relèvent de l'utilisateur. C'est pourquoi, avant même de penser à sécuriser sa machine, il faut garder en mémoire quelques règles de bon sens :

• Interdire les utilisateurs sans mot de passe (ce sont d'énormes failles potentielles)

• Toujours choisir de bons mots de passe : 8 caractères minimum, pas un mot qui se trouve dans le dictionnaire, si possible des chiffres, des majuscules, des symboles... au besoin, un outil comme pwgen vous en génèrera automatiquement des bons (apt-get install pwgen)

• Maintenir son système à jour (apt-get update et apt-get upgrade)

• Toujours utiliser ssh pour l'accès à distance

1.1.3 1.1.3 Utilisation de nmap

nmap est le meilleur outil de scan de ports : il va tenter d'ouvrir des connexions sur un grand nombre de ports de votre machine afin de déterminer s'ils sont ouverts ou non.

```
apt install nmap
nmap -v ip_ou_nom_de_la_machine
```

Vous aurez alors la liste des ports ouverts. Vous pouvez aussi tester un port en particulier avec l'argument -p port. Il n'est pas recommandé d'utiliser nmap sur quiconque autre que vous-mêmes.

Si vous avez pris des mesures pour bloquer (au mieux) le scan de port, il est conseillé dans un premier temps de les désactiver le temps du scan (car un port ouvert sans que vous le sachiez est une faille), et dans un second temps de jouer avec nmap pour voir si vos règles sont efficaces ou pas (par exemple avec les options -sS, -sN ou -sI).

1.1.4 Exemple de script

Vous trouverez ici un exemple e de script basique autorisant le minimum pour un serveur web (HTTP, FTP, mail et résolution de DNS). Je vous encourage à lire des docs et des tutos plus complets si vous voulez aller plus loin dans le paramétrage de votre firewall.

```
#!/bin/sh
# Réinitialise les règles
sudo iptables -t filter -F
sudo iptables -t filter -X
# Bloque tout le trafic
sudo iptables -t filter -P INPUT DROP
sudo iptables -t filter -P FORWARD DROP
sudo iptables -t filter -P OUTPUT DROP
# Autorise les connexions déjà établies et localhost
sudo iptables -A INPUT -m state --state RELATED,ESTABLISHED -j
ACCEPT
sudo iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j
ACCEPT
sudo iptables -t filter -A INPUT -i lo -j ACCEPT
sudo iptables -t filter -A OUTPUT -o lo -j ACCEPT
# ICMP (Ping)
sudo iptables -t filter -A INPUT -p icmp -j ACCEPT
sudo iptables -t filter -A OUTPUT -p icmp -j ACCEPT
# SSH
sudo iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
sudo iptables -t filter -A OUTPUT -p tcp --dport 22 -j ACCEPT
# DNS
sudo iptables -t filter -A OUTPUT -p tcp --dport 53 -j ACCEPT
sudo iptables -t filter -A OUTPUT -p udp --dport 53 -j ACCEPT
sudo iptables -t filter -A INPUT -p tcp --dport 53 -j ACCEPT
sudo iptables -t filter -A INPUT -p udp --dport 53 -j ACCEPT
```


```
# HTTP
sudo iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT
sudo iptables -t filter -A INPUT -p tcp --dport 80 -j ACCEPT
# FTP
sudo iptables -t filter -A OUTPUT -p tcp --dport 20:21 -j ACCEPT
sudo iptables -t filter -A INPUT -p tcp --dport 20:21 -j ACCEPT
# Mail SMTP
iptables -t filter -A INPUT -p tcp --dport 25 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 25 -j ACCEPT
# Mail POP3
iptables -t filter -A INPUT -p tcp --dport 110 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 110 -j ACCEPT
# Mail IMAP
iptables -t filter -A INPUT -p tcp --dport 143 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 143 -j ACCEPT
# NTP (horloge du serveur)
sudo iptables -t filter -A OUTPUT -p udp --dport 123 -j ACCEPT
```

1.2 Configuration avancée du pare-feu

Le firewall est l'outil de base de tout système de sécurité, il permet des manipulations plus poussées que filtrer des ports.

1.2.1 Se protéger des attaques Syn Flood et DOS

Ce genre d'attaque vise à surcharger la machine de requête. Il est possible de s'en prémunir directement au niveau du firewall.

```
iptables -A FORWARD -p tcp --syn -m limit --limit 1/second -j ACCEPT
```

Le flag TCP syn engendre des demandes de connexions, et le but de cette règle est donc de les limiter à une par seconde.

Il est cependant déconseillé de monter au-delà de la seconde sous peine de gêner le contrôle de flux et la récupération d'erreur de TCP.

On peut faire de même avec les protocoles UDP et ICMP

```
iptables -A FORWARD -p udp -m limit --limit 1/second -j ACCEPT iptables -A FORWARD
-p icmp --icmp-type echo-request -m limit -- limit 1/second -j ACCEPT
```

Ce type d'attaque permet de faire tomber le serveur, mais pas d'en prendre l'accès. C'est pourquoi le risque d'en être la cible est assez mince, un logiciel simple anti-intrusion comme fail2ban est suffisant.

1.2.2 Scan de port

On peut aussi limiter un tant soit peu le scan de ports qui consiste à tester tous vos ports afin de détecter ceux qui sont ouverts, pour cela, une règle de ce genre irait :

```
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j
ACCEPT
```

C'est un peu le même principe que ci-dessus sachant qu'une connexion TCP en bon et due forme requiert trois paquets avec trois flags différents, on voit tout de suite la finesse de cette règle qui peut travailler paquet par paquet.

Il faut toutefois garder à l'esprit que cette règle basique n'est pas très efficace, c'est une protection de base.

1.2.3 Bannir une IP

Si vous repérez dans les logs ou autre une adresse IP suspecte, vous pouvez la bannir aisément au niveau du firewall via la commande

```
iptables -A INPUT -s adresse_ip -j DROP
```

Il n'est pas conseillé de bannir les IP à tour de bras.

2 Lutter contre les intrusions

Actuellement, le firewall va bloquer toutes tentatives de connexions sur les ports fermés, mais qu'en est-il des ports ouverts ?

Afin de contrôler plus précisément ce qui se passe dessus, le firewall n'est pas suffisant et nous allons devoir utiliser d'autres outils, appelés IDS (Intrusion Detection System) et IPS (Intrusion Prevention System).

Ces deux catégories de logiciels vont - comme leur nom l'indique - surveiller toute tentative d'intrusion sur le serveur

La démarche qui va suivre va vous montrer comment réagir à chaque étape d'une tentative intrusion classique, à savoir :

• Le scan de port (plus généralement, la collecte d'informations) afin de trouver les vulnérabilités

• Les attaques « simples », témoins d'une faible sécurité

• L'intrusion (via des techniques qui ne seront pas décrites ici car dépassant notre cadre de travail)

• L'installation d'un moyen de se logguer sur le serveur à volonté (si l'attaquant parvient jusqu'ici avec succès, on peut dire que la machine lui appartient).

2.1 Portsentry

Cet utilitaire permet de bloquer en temps réel la plupart des scans de port connus (même très discrets et échappant aux règles de filtrage du firewall basiques).

On rappelle au passage que scanner les ports signifie tester tous les ports d'une machine afin de déterminer ceux qui sont ouverts (les portes d'entrées), cependant, il ne faut pas paniquer si votre serveur est la cible d'un simple scan de port, cela sera monnaie courante, et si vous êtes bien protégé, le pirate passera sa route

2.1.1 Installation et configuration de Portsentry

```
apt install portsentry
```

Puis nous allons le configurer nano

```
 /usr/local/psionic/portsentry/portsentry.conf
```

ou

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

Commentez les lignes :

```
KILL_HOSTS_DENY
```

Décommentez la ligne :

```
KILL_ROUTE="/sbin/iptables -I INPUT -s $TARGET$ -j DROP"
```

Ainsi, Portsentry ajoutera une règle dans le firewall (iptables) pour rejeter les paquets en cas de scans

On démarre le logiciel, il faut le lancer deux fois, pour TCP et UDP

```
portsentry –audp
portsentry –atcp
```

Vous pouvez tester avec nmap (si vous voulez tester en local, il vous faut modifier le fichier portsentry.ignore en enlevant le localhost).

Si vous souhaitez que vos réglages restent même après un nouveau lancement de portsentry, vous devrez modifier le fichier portsentry.ignore.static .

Les ports ouverts sur la machine sont à priori sans grande protection, et sujet à des attaques simples telles que

• La tentative de connexion par brute-force ou par dictionnaire (par exemple, tester toutes les combinaisons de mots de passe pour se logguer en ssh)

• Le déni de services (surcharger le serveur de requêtes) ou plus simplement la recherche d'utilisateurs sans mots de passe...

2.2 Fail2ban

Fail2ban est un petit utilitaire qui se base sur les logs de la machine pour chercher des actions suspectes répétées (par exemple, des erreurs de mots de passe) dans un laps de temps donné.

S'il en trouve, il bannira l'IP de l'attaquant via iptables, ce type de logiciel est indispensable, car, bien que léger, il offre une bonne protection contre les attaques basiques indiquées ci-dessus.

2.2.1 Installation et configuration de Fail2ban

```
apt install fail2ban
```

Puis, pour sa configuration

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

Vous pouvez remplir le champ ci-dessous :

• destmail : indiquez une adresse mail si vous voulez recevoir des mails d'alerte de la part de fail2ban. Le niveau de protection peut être modulé via les champs suivants (notez que la configuration par défaut suffit normalement)

• bantime : temps de bannissement des IP suspectes

• maxretry : nombre de tentatives de connexion permise avant bannissement.

Notez que dans la partie JAILS (dans nano : ctrl w => rechercher JAILS) figure tous les services que fail2ban surveillera, si vous modifiez les ports par défaut, il faut les indiquer là aussi. Par exemple avec ssh.

```
nano /etc/fail2ban/jail.conf
ctrl+w => chercher [ssh]
 port : indiquer le port
```

Enregistrez et quittez.

Vous pouvez parcourir rapidement le reste des options afin de personnaliser un peu votre soft.

Enfin, pour recharger la nouvelle configuration

```
/etc/init.d/fail2ban restart
```

2.3 Snort

Snort est un outil très puissant, pouvant en fait détecter la plupart des attaques qui échapperaient à un utilitaire comme fail2ban. Bien entendu, il ne servira pas dans 90 % des cas et comme ce n'est qu'un outil de détection, ce sera à vous de rendre les mesures nécessaires s'il détecte une intrusion. Enfin, comme il analyse le trafic en temps réel, cela ralentit forcément un peu les flux

2.3.1 Installation et configuration de Snort

```
apt install snort
```

On vous demandera ensuite de sélectionner l'interface d'écoute et le réseau

On peut maintenant configurer notre service.

```
nano /etc/oinkmaster.conf
```

Dans la section "Location of rules archive", commentez la ligne

```
#url = http://www.snort.org/dl/rules/snortrules-snapshot-2_2.tar.gz
```

Et ajoutez juste en dessous la ligne suivante

```
url = http://rules.emergingthreats.net/open-nogpl/snort-2.8.4/emerging.rules.tar.gz
```

Cette commande peut nécessiter une adaptation en fonction de la version de snort. Cette version s'obtient à l'aide de

```
snort -v
```

Puis

```
oinkmaster -o /etc/snort/rules
```

Si vous avez une erreur du type

```
/usr/sbin/oinkmaster: Error: the temporary directory "/var/run/oinkmaster" does not exist or isn't writable by you.
```

Créez alors le dossier temporaire indiqué avec la commande

```
mkdir /var/run/oinkmaster
```

Puis relancez la commande

```
oinkmaster -o /etc/snort/rules
```

Il faudra pour finir, éditer un dernier fichier

```
nano /etc/snort/snort.conf
```

Et commenter quelques règles, pour en savoir plus concernant le blocage de certaines règles :

Snort - Network Intrusion Detection & Prevention System

```
#include $RULE_PATH/emerging-botcc-BLOCK.rules
#include $RULE_PATH/emerging-compromised-BLOCK.rules
#include $RULE_PATH/emerging-drop-BLOCK.rules
#include $RULE_PATH/emerging-dshield-BLOCK.rules
#include $RULE_PATH/emerging-rbn-BLOCK.rules
#include $RULE_PATH/emerging-sid-msg.map
#include $RULE_PATH/emerging-sid-msg.map.txt
```

Snort est maintenant installé et configuré, nous pouvons le démarrer

```
/etc/init.d/snort start
```

2.4 Rkhunter

Dernier volet de cette section intrusion, les backdoors.

Si par malheur un attaquant arrive à prendre possession de votre machine, il y a fort à parier qu'il y laisse une backdoor (porte dérobée) qui lui permettrait d'en reprendre le contrôle plus tard, ainsi qu'un rootkit pour la dissimuler : l'attaquant maintient ainsi un accès frauduleux à votre machine.

Rkhunter est un utilitaire qui est chargé de détecter d'éventuels rootkits sur votre serveur. Il est relativement léger (s'exécute une fois par jour par défaut.

2.4.1 Installation et configuration de Rkhunter

```
apt install rkhunter
```

Puis nous allons modifier la configuration

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

REPORT_EMAIL (indiquez un mail pour recevoir des alertes de Rkhunter)

CRON_DAILY_RUN (mettez « yes » pour une vérification quotidienne de votre machine via un cron).

Rkhunter se trompe parfois en déclarant comme infectés des fichiers sains (« faux positifs »), donc il faut être critique à l'égard des rapports. Par contre, s'il s'avère que l'alerte est justifiée, cela signifie que vous avez un rootkit ainsi qu'une faille de sécurité qui a été découverte et exploitée.

3 Surveiller les logs

La plupart des logiciels cités plus haut vous enverront des notifications par mail en cas d'alerte. Cependant, surveiller les logs est important, car ils reflètent la « vie » de votre serveur. Les logs les plus intéressants sont notamment

• /var/log/auth.log qui contient toutes les tentatives d'accès au serveur. Il peut être utile de filtrer le contenu, par exemple : cat /var/log/auth.log | grep authentication failure

• /var/log/message et /var/log/syslog contient un peu de tout (erreurs, bugs, informations, etc)

• /var/log/fail2ban est le log d'alerte de fail2ban. Cherchez notamment : cat /var/log/fail2ban | grep ban

• /var/log/snort/alert vous indiquera les logs d'alertes de Snort

• /var/log/rkhunter pour voir les rapports quotidiens de Rkhunter, attention aux erreurs trouvées (même si le risque de faux positifs existe ici).

3.1 Logwatch

Il est aussi possible d'utiliser des utilitaires qui vous simplifient un peu ce travail de lecture des logs. Logwatch notamment permet de résumer plusieurs logs afin de ne vous retourner que des anomalies si possibles.

3.1.1 Installation et configuration de Logwatch

```
apt install logwatch
nano /usr/share/logwatch/default.conf/logwatch.conf
```

Spécifiez l'option « MailTo » car logwatch envoie ses résumés de logs par mail. Il va normalement s'exécuter tous les jours (ls -l /etc/cron.daily/ | grep logwatch pour s'en assurer).

Il peut aussi être intéressant de suivre l'état du réseau et du système (monitoring) afin de détecter par exemple une brusque montée en charge, synonyme de problèmes.

Pour aller plus loin…

Le but est de chercher en général les failles de votre machine, il convient de les régler toutes, car l'attaquant peut les trouver aussi bien que vous.

Pour cela, Nessus est un des utilitaires les plus performants. Comme le logiciel est propriétaire et qu'il s'utilise via une interface graphique :

Auditer la sécurité de son réseau avec Nessus | Linux Pour Lesnuls (linux-pour-lesnuls.com)

4 Webographie

Sécuriser son serveur Linux :

Des parcours diplômants et des cours gratuits 100% en ligne - OpenClassrooms

Nessus

Try Nessus Pro free | Tenable

Snort

snort [Wiki ubuntu-fr]

Snort

Snort - Network Intrusion Detection & Prevention System