Rapport sécurisation serveur

Alexis Araujo
Samir Subra
Nino Pires
Antoine Bazire
SAE groupe 7

FAIL2BAN :

Fail2ban est un outil que l'on peut installer sur une machine UNIX**,** il va se charger de parser (lire, parcourir) les logs de différentes applications pour vérifier et détecter des comportements dis "suspects"**.** Il va par exemple savoir détecter un nombre X de tentatives d'authentification infructueuses sur une service FTP ou SSH ou détecter des requêtes anormales sur un services web tel qu'Apache2.

Le fonctionnement de Fail2ban se fait avec des prisons. Une prison est un ou plusieurs services ou ports sur lesquels vont s'appliquer des règles et dans laquelle des IP ne respectant pas ces règles vont être mises. Une fois le comportement d'une IP détectée comme suspecte, une action est effectuée pour la contrer**.** Par défaut il s'agit de bloquer l'IP en l'interdisant de communiquer avec le serveur pendant 600 secondes via des règles Iptables (pare-feu par défaut de beaucoup de distributions UNIX**).**

INSTALLATION :

apt-get update
pour être sûr d'avoir la dernière version du logiciel

apt-get install fail2ban

Maintenant que Fail2ban est installé, nous pouvons voir que sa configuration se situe dans "**/etc/fail2ban**"

Le dossier "**/etc/fail2ban/filter.d**" contient un ensemble de règles que Fail2ban va utiliser lors de la lecture des différents fichiers de logs.

Nous voyons donc un ensemble de lignes que Fail2ban va simplement essayer de retrouver dans le fichier de logs indiqué (ici "**/var/log/auth.log**" où sont écrites les tentatives de connexion SSH) pour ensuite appliquer une action en cas de correspondance.

Comme dit précédemment, une fois qu'un comportement anormal est détecté, une action va être menée afin de contrer ce comportement anormal ou au moins d'avertir l'administrateur.

Nous pouvons voir les “prisons” qui sont actuellement opérationnelles avec la commande suivante :
fail2ban-client status

qui affiche ceci :

```
root@raspberrypi:/var/log# fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd
root@raspberrypi:/var/log#
```

Dans le fichier "**jail.conf**" sont aussi indiquées des valeurs par défaut. On trouve par exemple le temps de bannissement standard qui est de 600 secondes ("bantime = 600"), le nombre de tentatives max par défaut ("maxretry=3"), le destinataire d'éventuel mail à envoyer ("destemail=root@localhost").

Nous allons maintenant pouvoir tester notre filtre ssh. Pour ce faire, nous devons tester une connexion erronée 6 fois d'affilée, ce qui devrait bannir 10 minutes l’ip suspecte.

```
Jan 18 13:37:19 raspberrypi sshd[844]: Failed password for saepi from 192.168.0.89 port 59950 ssh2
Jan 18 13:37:24 raspberrypi sshd[844]: Failed password for saepi from 192.168.0.89 port 59950 ssh2
Jan 18 13:37:28 raspberrypi sshd[844]: Failed password for saepi from 192.168.0.89 port 59950 ssh2
Jan 18 13:37:30 raspberrypi sshd[844]: Connection closed by authenticating user saepi 192.168.0.89 port 59950 [preauth]
Jan 18 13:37:30 raspberrypi sshd[844]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.0.89  user=saepi
Jan 18 13:37:33 raspberrypi sshd[854]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.0.89  user=saepi
Jan 18 13:37:35 raspberrypi sshd[854]: Failed password for saepi from 192.168.0.89 port 59952 ssh2
Jan 18 13:37:41 raspberrypi sshd[854]: Failed password for saepi from 192.168.0.89 port 59952 ssh2
```

Nous pouvons voir dans la capture d’écran ci-dessus prise dans le fichier “**/var/log/auth.log”** que quelqu’un est en train de forcer la connexion.
Après ces 6 essais, on va voir le récapitulatif des ip bannies avec la commande :
“**fail2ban-client status ssh**”

```
root@raspberrypi:~# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 1
|  |- Total failed:     6
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   192.168.0.89
```

L’ip bannie y est bien affichée, il pourra réessayer de se connecter au bout de 10mn.

HTTPS :

HTTPS (HTTP Secure) est une extension du protocole HTTP qui permet une communication sécurisée sur internet. Il utilise le protocole SSL/TLS pour chiffrer les données échangées entre un navigateur web et un serveur web.

Lorsqu'un utilisateur se connecte à un site web en utilisant HTTPS, le navigateur vérifie d'abord que le certificat SSL/TLS est valide et émis par une autorité de certification (CA) de confiance. Ensuite, une session sécurisée est établie en utilisant un système de chiffrement symétrique pour échanger des clés de chiffrement. Les données échangées entre le navigateur et le serveur sont chiffrées et déchiffrées à l'aide de ces clés.

INSTALLATION :

Pour sécuriser notre serveur Apache en utilisant le certificat X.509 HTTPS, nous avons procédé comme ceci :

On a premièrement généré un certificat auto-signé en utilisant OpenSSL :

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/pki/tls/private/localhost.key -out /etc/pki/tls/certs/localhost.crt

Nous avons ensuite configuré Apache pour utiliser ce certificat :

```
   Listen 443 https
<VirtualHost *:443>
   ServerName yourdomain.com
   SSLEngine on
   SSLCertificateFile /etc/pki/tls/certs/localhost.crt
   SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
   # ...
</VirtualHost>
```

Enfin, il nous a suffi de redémarrer Apache pour appliquer les modifications :

systemctl restart httpd

En résumé, HTTPS permet de sécuriser les communications entre un navigateur web et un serveur web en chiffrant les données échangées et en vérifiant l'authenticité du certificat SSL/TLS utilisé. Cela protège les informations sensibles telles que les mots de passe et les données de carte de crédit contre les interceptions et les attaques de type man-in-the-middle.
Il est important de noter que les étapes ci-dessus sont pour un certificat auto-signé, il est recommandé d'utiliser un certificat SSL émis par une autorité de certification tierce pour une sécurité accrue.