HIDS avec Fail2ban

Fail2ban est un **HIDS** (Host Intrusion Detection System) c'est-à-dire un système permettant de détecter des tentatives d'intrusions (**IDS**) sur une machine en analysant les logs à la recherche de comportements malveillants (succession de mots de passe erronés laissant supposer une attaque par force brute ou dictionnaire, accès à des répertoires ou pages inexistantes d'un serveur web laissant penser qu'un personne malveillante tente de faire une énumération par force brute du contenu du site…). Mais Fail2ban est capable également de bloquer ces comportements en bannissant, pour une certaine durée, l'adresse IP de l'attaquant (via l'écriture d'une règle Iptables dans le pare-feu ☺. Il agit donc comme un **IPS** (Intrusion Prevention System).

Pour ce faire, Fail2ban va se baser sur des **filtres** (= règles de détection et actions à réaliser). Le logiciel en possède déjà un certain nombre préconfigurés (pour SSH, apache2, FTP, etc.). Mais on peut ajouter d'autres filtres ou adapter leur paramétrage.

Installation

Mise à jour des dépôts et installation de Fail2ban :

```
apt install fail2ban
```

Mise en place d'un démarrage automatique après chaque redémarrage de la machine :

```
systemctl enable fail2ban
```

Paramétrage général

Normalement, les paramètres sont à écrire dans le fichier de configuration principal : **/etc/fail2ban/jail.conf** . Ce fichier contient déjà beaucoup de paramètres. Mais, ce fichier risque d'être écrasé lors d'une mise à jour du logiciel. On doit donc créer un nouveau fichier dans le dossier **/etc/fail2ban/jail.d/** . Tous les fichiers présents dans ce dossier seront toujours appelés par le fichier de configuration principale ☺. En cas de paramètres déjà présents dans **jail.conf** , c'est toujours les paramètres écrits dans le dossier **jail.d/** qui seront pris en compte.

Créer le fichier **/etc/fail2ban/jail.d/monParametrage.conf** puis saisir ce contenu :

Nano /etc/fail2ban/jail.d/monParametrage.conf

```
[DEFAULT]

# Adresses IP non concernées par la surveillance. Ici par exemple on a :
# la boucle locale et l'adresse IP du DSI (10.100.0.99)
ignoreip = 127.0.0.1 10.100.0.99

# Période de temps de recherche dans les logs
findtime = 10m

# Durée de bannissement (en minute, heure, jour, semaine : m, h, d, w)
bantime = 1h
# Durée de bannissement grandissante
bantime.increment = true
# Le 2ème bannissement sera de 24h ; le 3ème de 48h ; le 4ème de 96h, etc.
bantime.factor = 24
# Durée maximale de bannissement
bantime.maxtime = 8w

# Nombre maximal de tentatives échouées avant bannissement (dans la période
# de temps fixée par « findtime »
maxretry = 3
```

Prisons disponibles

Il est nécessaire de spécifier à Fail2ban quels services surveiller (SSH, HTTP, FTP…) en activant les « jails » (prisons) correspondantes. L'ensemble des prisons disponibles sont dans le fichier **/etc/fail2ban/jail.conf** . Exemple de contenu du fichier (NE RIEN ECRIRE DEDANS) :

Nano /etc/fail2ban/jail.conf

```
...
[sshd]
# Port à bloquer via une règle iptables
port = ssh
# Chemin des fichiers de log à surveiller
logpath = %(sshd_log)s
# Moteur de surveillance des logs
backend = %(sshd_backend)s

[apache-botsearch]
# Port à bloquer via une règle iptables
port = http,https
# Chemin des fichiers de log à surveiller
logpath = %(apache_error_log)s
# Nombre de tentatives (=lignes de log) max autorisé
maxretry = 2
...
```

Remarque : les règles pour « mettre des adresses IP en prison » sont détaillées dans des « filtres » (pour chaque prison). Cf. fin du document.

Activer une prison

Pour activer une prison, par exemple celle concernant SSH, il faut renseigner le fichier de configuration, ici : **/etc/fail2ban/jail.d/monParametrage.conf**, et définir les conditions pour « rentrer » en prison (maxretry, logpath…) et la durée de détention (bantime…) :

Nano /etc/fail2ban/jail.d/monParametrage.conf

```
[sshd]

# Activation de la prison
enabled = true
# Optionnellement, on peut préciser le port s'il n'est pas celui par défaut
port = 2222
# Optionnellement, on peut préciser le fichier de log à analyser
logpath = /var/log/auth.log
# Optionnellement, on peut préciser le nombre maximal de tentatives en échec
maxretry = 2
```

Il faut ensuite redémarrer le service :

```
systemctl restart fail2ban
```

Gestion des prisons

On peut vérifier les prisons mises en place :

```
fail2ban-client status

       Status
       |- Number of jail:      1
       `- Jail list:   sshd
```

Ou bien contrôler une prison spécifique (exemple SSH) :

```
fail2ban-client status sshd

       Status for the jail: sshd
       |- Filter
       |  |- Currently failed: 0
       |  |- Total failed:     0
       |  `- File list:        /var/log/auth.log
       `- Actions
          |- Currently banned: 0
          |- Total banned:     0
          `- Banned IP list:
```

Pour arrêter une prison, on peut utiliser la commande suivante :

```
fail2ban-client stop sshd
```

Pour démarrer une prison arrêtée, on peut utiliser la commande suivante :

```
fail2ban-client start sshd
```

Gestion du bannissement

Pour dé-bannir une IP :

```
fail2ban-client set nomPrison unbanip IP
```

Pour bannir manuellement une adresse IP:

```
fail2ban-client set nomPrison banip IP
```

Logs

Fail2ban génère des logs à l'adresse suivante :

```
tail /var/log/fail2ban.log
```

Pour voir les logs en temps réel :

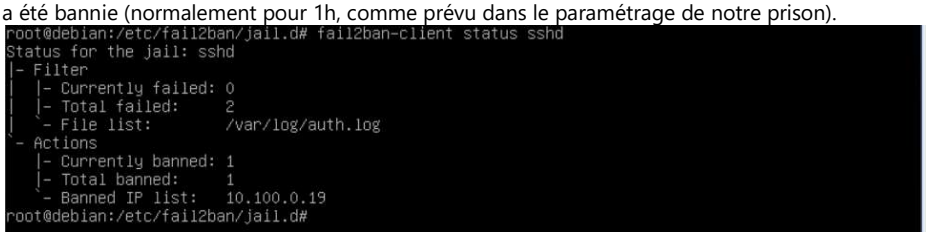
```
tail -f /var/log/fail2ban.log
```

Exemple

Après avoir mis en place une prison pour SSH pour bannir, au bout de 2 essais, les IP malveillantes (mots de passe erronés), on peut réaliser le test suivant :

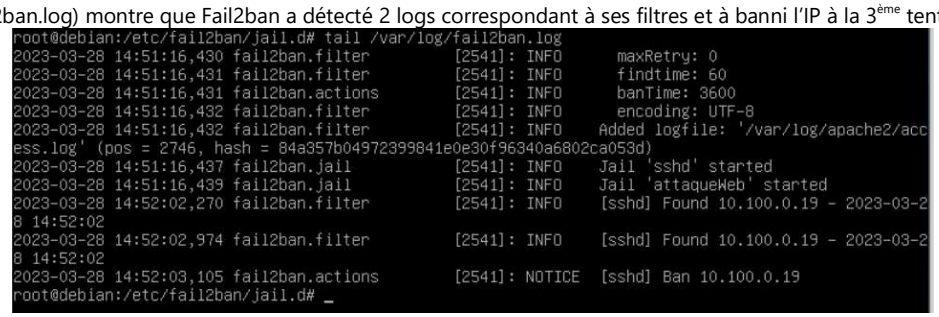
1. Se connecter en SSH avec au moins 2 mots de passe erronés (le 3ème provoquera le blocage).

2. On peut voir qu'une IP a été bannie (normalement pour 1h, comme prévu dans le paramétrage de notre prison).

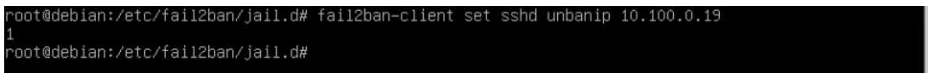


3. Ce bannissement a été matérialisé par une règle REJECT dans iptables.

4. Les logs (/var/log/fail2ban.log) montre que Fail2ban a détecté 2 logs correspondant à ses filtres et à banni l'IP à la 3ème tentative.



5. On dé-banni l'IP.



6. La prison montre que l'IP n'est plus banni (mais se souvient qu'il avait déjà banni une adresse).

7. Et maintenant on peut de nouveau se reconnecter en SSH 😊.

Alertes

Pour recevoir un mail après chaque bannissement (il faut au préalable installer « msmtp ») :

Nano /etc/fail2ban/jail.d/monParametrage.conf

```
[DEFAULT]
...
destemail = adresseEmail@gmailParExemple.com
action = %(action_mw)s
```

Filtres

Pour information (ne rien faire dans cette partie).

Des filtres prêts à l'emploi sont déjà proposés dans Fail2ban. Ces filtres contiennent des règles pour détecter certaines lignes de logs (notamment via des expressions régulières -REGEX-). On peut les modifier ou ajouter de nouveaux filtres (il faut regarder la syntaxe des logs qu'on souhaite bloquer puis spécifier, dans un filtre, qu'on recherche cette syntaxe précise 😉.

Les filtres se trouvent dans le dossier **/etc/fail2ban/filter.d** . Par exemple, le filtre pour le service SSH se trouve dans ce fichier : **/etc/fail2ban/filter.d/sshd .**

Extrait du fichier :

GNU nano 5.4 /etc/fail2ban/filter.d/sshd.conf

```
# Fail2Ban filter for openssh
#
# If you want to protect OpenSSH from being bruteforced by password
# authentication then get public key authentication working before disabling
# PasswordAuthentication in sshd_config.
#
#
# "Connection from <HOST> port \d+" requires LogLevel VERBOSE in sshd_config
#

[INCLUDES]

# Read common prefixes. If any customizations available -- read them from
# common.local
before = common.conf

[DEFAULT]

_daemon = sshd

# optional prefix (logged from several ssh versions) like "error: ", "error: PAM: " or "fatal: "
__pref = (?:(?:error|fatal): (?:PAM: )?)?
# optional suffix (logged from several ssh versions) like " [preauth]"
#__suff = (?: port \d+)?(?: \[preauth\])?\s*
__suff = (?: (?:port \d+|on \S+|\[preauth\])){0,3}\s*
__on_port_opt = (?: (?:port \d+|on \S+)){0,2}
# close by authenticating user:
__authng_user = (?: (?:invalid|authenticating) user <F-USER>\S+|.*?</F-USER>)?

# for all possible (also future) forms of "no matching (cipher|mac|MAC|compression method|key exchange method|host key type) found",
# see ssherr.c for all possible SSH_ERR_..._ALG_MATCH errors.
__alg_match = (?:(?:\w+ (?!found\b)){0,2}\w+)

# PAM authentication mechanism, can be overridden, e. g. `filter = sshd[__pam_auth='pam_ldap']`:
__pam_auth = pam_[a-z]+

[Definition]

prefregex = ^<F-MLFID>%(__prefix_line)s</F-MLFID>%(__pref)s<F-CONTENT>.+</F-CONTENT>$

cmnfailre = ^[aA]uthentication (?:failure|error|failed) for <F-USER>.*</F-USER> from <HOST>( via \S+)?%(__suff)s$
            ^User not known to the underlying authentication module for <F-USER>.*</F-USER> from <HOST>%(__suff)s$
            <cmnfailre-failed-pub-<publickey>>
            ^Failed <cmnfailed> for (?P<cond_inv>invalid user )?<F-USER>(?P<cond_user>\S+)|(?(cond_inv)(?:(?! from ).)*?|[^:]+)</F->
            ^<F-USER>ROOT</F-USER> LOGIN REFUSED FROM <HOST>
            ^[iI](?:llegal|nvalid) user <F-USER>.*?</F-USER> from <HOST>%(__suff)s$
            ^User <F-USER>\S+|.*?</F-USER> from <HOST> not allowed because not listed in AllowUsers%(__suff)s$
            ^User <F-USER>\S+|.*?</F-USER> from <HOST> not allowed because listed in DenyUsers%(__suff)s$
            ^User <F-USER>\S+|.*?</F-USER> from <HOST> not allowed because not in any group%(__suff)s$
            ^refused connect from \S+ \(<HOST>\)
            ^Received <F-MLFFORGET>disconnect</F-MLFFORGET> from <HOST>%(__on_port_opt)s:\s*3: .*: Auth fail%(__suff)s$
```

```
^G Aide       ^O Écrire      ^W Chercher    ^K Couper      ^T Exécuter    ^C Emplacement  M-U Annuler     M-A Placer la marque
^X Quitter    ^R Lire fich.  ^\ Remplacer   ^U Coller      ^J Justifier   ^_ Aller ligne  M-E Refaire     M-6 Copier
```

Création d'une prison personnelle (avec filtre associé)

1. Par exemple, on héberge le site suivant : http://10.100.0.42

2. Une personne malveillante (ou trop curieuse) peut tenter d'énumérer le site à la recherche de pages ou dossiers intéressants, par exemple : http://10.100.0.42/admin. Mais cette page n'existe pas. Cela provoque un message d'erreur (code 404 pour le protocole HTTP).

3. Un log (une trace) a été généré sur le serveur web. On voit ici 2 logs : un correspondant à la page affichée (code 200, donc page trouvée) et un correspondant à la page non trouvée (.../admin) donc avec une erreur 404. Ce dernier log va nous permettre de définir un filtre : dès que l'on verra, à l'avenir, ce log (**"GET /admin HTTP/1.1" 404**), c'est que quelqu'un essaye d'accéder à un endroit stratégique (qui heureusement n'existe pas) et doit donc être bloqué.

```
root@debian:~# tail -n2 /var/log/apache2/access.log
10.100.0.19 - - [28/Mar/2023:15:14:05 +0200] "GET / HTTP/1.1" 200 3380 "-" "Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36"
10.100.0.19 - - [28/Mar/2023:15:16:05 +0200] "GET /admin HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows
 NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36"
root@debian:~#
```

4. Création la nouvelle prison que l'on va appeler « attaqueWeb » via le fichier de configuration de Fail2ban : ***/etc/fail2ban/jail.d/monParametrage.conf***, en ajoutant les lignes suivantes :

```
Nano /etc/fail2ban/jail.d/monParametrage.conf
[attaqueWeb]
enabled = true
# On lui dit quel filtre utiliser pour détecter l'attaque (filtre qui va être créé dans l'étape suivante)
filter = attaqueWeb
# On utilise ce fichier de log pour rechercher les attaques
logpath = /var/log/apache2/access.log
# On bloque dès la 1ere tentative
maxretry = 0
```

5. Ensuite, on crée un filtre pour correspondre à la chaîne de caractères "GET /admin HTTP/1.1" 404 dans le fichier journal. Pour ce faire, on crée un nouveau fichier de filtre ***/etc/fail2ban/filter.d/attaqueWeb.conf*** avec le contenu suivant :

```
Nano /etc/fail2ban/filter.d/attaqueWeb.conf
[Definition]
failregex = ^<HOST>.*"GET /admin HTTP/1.1" 404.*

# Ou bien le filtre suivant si on veut bloquer /admin, ou /private ou /administration
#failregex = ^<HOST>.*"GET /(admin|private|administration) HTTP/1.1" 404.*

# Ou bien le filtre suivant si on veut bloquer quand la personne essaye /contact puis tout de suite après /about, puis /search
# Typiquement, ça serait l'.uvre d'une énumération par force brute (comme avec Gobuster ) via le dictionnaire suivant :
# /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt 😊
#failregex = ^<HOST>.*"GET /contact HTTP/1.1" 404.*\n.*"GET /about HTTP/1.1" 404.*\n.*"GET /search HTTP/1.1" 404.*
```

6. On redémarre le service.

```
systemctl restart fail2ban
```

7. On vérifie les prisons actives :

```
root@debian:~# fail2ban-client status
Status
|- Number of jail:      2
`- Jail list:   attaqueWeb, sshd
root@debian:~#
```

8. On imite un internaute malveillant en se rendant sur http://10.100.0.42/admin (normalement on est directement banni).

9. On vérifie la prison « attaqueWeb » : l'IP est bannie 😊


root@debian:~# fail2ban-client status attaqueWeb
Status for the jail: attaqueWeb
|- Filter
| |- Currently failed: 0
| |- Total failed: 1
| `- File list: /var/log/apache2/access.log
`- Actions
   |- Currently banned: 1
   |- Total banned: 1
   `- Banned IP list: 10.100.0.19
root@debian:~#