Fail2Ban

Un attaquant peut essayer en « brute force » toutes les combinaisons possibles pour trouver le mot de passe d'un compte utilisateur sur un service. Pour le détecter, il faut qu'une journalisation permette d'identifier un échec de connexion.

Les services (Apache, SSH, FTP, etc) ou le code ne permettent pas de détecter cela.

Il faut rajouter un utilitaire qui permet de détecter les comportements anormaux dans les accès aux services d'une machine : connexion trop fréquentes et en échec, connexions démultipliées dans un temps court, etc.

C'est le rôle de l'outil Fail2ban : de l'échec (Fail) à (2) un bannissement (Ban).

Installation

On doit d'abord installer ***fail2ban*** à partir des dépôts :

```
apt install fail2ban
```

On doit aussi disposer de l'outil iptables qui gèrera les règles de bannissement :

```
apt install iptables
```

Les principes

Fail2ban est un outil de détection d'intrusion

- Il **explore les journaux d'activités** (de la machine, d'un service, etc) pour y repérer des indications de **comportement anormaux** (par exemple une répétition de messages « connection refused » dans un temps déterminé).
- il se base sur 3 types de fichiers :
  - jail : les services qu'on surveille (nom, port, fichiers journaux étudiés, filtres appliqués, etc) avec les paramètres (nombre de tentatives considérées comme une intrusion, intervalle de temps dans lequel on détecte ce nombre, etc)
  - filter : fichiers qui décrivent les messages qu'on cherche à repérer dans les journaux
  - action : les comportements produits lors de la détection d'une intrusion, par exemple le blocage d'une IP, d'une adresse mail, d'une route dans une table de routage, etc)

Pour pouvoir détecter une intrusion, il faut donc un fichier de journalisation dans lequel des informations doivent être inscrites (échec, rejet, etc) pour que Fail2ban les repère.

Les dossiers et fichiers

Fail2ban repose sur les éléments suivants :

- Dossier de base (**/etc/fail2ban**)
  - **fail2ban.conf** : définit les configurations de base de l'outil : niveau de journalisation, fichier de journalisation, durée de conservation, etc
  - **jail.conf** : paramètres par défaut des ***jail*** (prisons ou éléments surveillés) s'ils ne sont pas redéfinis :
    - **bantime** : défini la durée de bannissement **en secondes**, et les options (rallongement, mémorisation, etc)
    - ignoreself et ignoreip : machines qui ne doivent pas être bannies
    - **maxretry** : nombre d'essais infructueux détectés avant bannissement
    - **findtime** : délai dans lequel on cherche les échecs d'action
    - banaction : comportements à appliquer en cas de détection d'intrusion
    - une **section** /[JAIL] fourni les configurations par défaut pour de nombreux services
  - les fichiers ***paths*** indiquent les chemins vers les fichiers à utiliser pour les dossiers, actions et autres filtres

Pour configurer Fail2ban, la bonne pratique est de définir des **fichiers spécifiques** (nommés ***<nom>.local***) dans les dossiers dédiés :

- **jail.d** : on y crée les configurations pour les services qu'on souhaite surveiller, soit dans des fichiers individuels, soit dans un fichier ***jail.local***
- **filter.d** : on y crée ou adapte les fichiers définissant ce qui est recherché dans les journaux pour identifier un type de comportement inadapté
- **action.d** : dossier qui décrit les comportements (blocages d'IP, ajout de règles de filtrage, journalisation, etc)

Les principales commandes

Gérer le service

```
systemctl {stop|start|restart|status} fail2ban
```

Pour bloquer une IP pour un jail donné (sshd, apache, etc)

```
fail2ban-client set <jail> banip <ip_poste>
```

Pour débloquer une IP bannie pour un jail donné (sshd, apache, etc)

```
fail2ban-client set <jail> unbanip <ip_poste>
```

Voir les journaux d'activité

```
{nano|tail} /var/log/fail2ban.log
```

Exemple de démarche pour du brute force sur SSH

On peut configurer Fail2ban pour qu'il détecte les tentatives de connexion sur SSH. L'outil permet des paramétrages plus ou moins restrictifs (mode normal, aggressive, ddos et extra). On trouvera une démarche et les explications complètes https://www.malekal.com/comment-proteger-ssh-avec-fail2ban-des-attaques-dos-bruteforce/.

Pour SSH, tout le fonctionnement est pré-paramétré (filter, actions). Il ne reste qu'à définir le **jail**.

Exemple d'une configuration de base

Dans le fichier **/etc/fail2ban/jail.d/defaults-debian.conf** (existant) ou **/etc/fail2ban/jail.d/jail.local ** (éventuellement à créer), on indiquera la **//section**// [sshd] suivante :

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

```
[sshd]
# Active la surveillance stricte de SSH
enabled = true
#mode de surveillance
mode = normal
# Port sur lequel le blocage s'appliquera (par son nom ou son numéro : SSH
ou 22)
port = ssh
# Chemin des logs d'authentification
logpath = /var/log/auth.log
# Nombre de tentatives autorisées avant bannissement (plus strict)
maxretry = 3
# Période d'échecs de connexion en minutes (30 minutes = 3000s)
findtime = 3000
# Temps de bannissement en minutes (20 minutes)
bantime = 20m
# Backend pour la gestion des journaux
backend = systemd
```

Après modification du fichier, on redémarre le service :

```
 systemctl restart fail2ban
```

Une tentative de connexion impossible générera des traces dans le journal, jusqu'au banissement de la machine concernée.

```
distant@machine.sio.bts$ tail /var/log/fail2ban.log
2025-01-16 08:38:56,246 fail2ban.jail           [11935]: INFO    Jail 'sshd'
started
2025-01-16 08:38:56,247 fail2ban.jail           [11935]: INFO    Jail
'apache' started
2025-01-16 08:41:06,536 fail2ban.filter         [11935]: INFO    [sshd]
Found 10.12.0.38 - 2025-01-16 08:41:06
2025-01-16 08:41:11,794 fail2ban.filter         [11935]: INFO    [sshd]
Found 10.12.0.38 - 2025-01-16 08:41:11
2025-01-16 08:41:16,044 fail2ban.filter         [11935]: INFO    [sshd]
```

```
Found 10.12.0.38 - 2025-01-16 08:41:15
2025-01-16 08:41:16,288 fail2ban.actions        [11935]: NOTICE  [sshd] Ban
10.12.0.38
```

Exemple de démarche pour du brute force sur une application Web

Le fichier jail.local

On doit surveiller les échecs de connexion sur l'application, donc via le service Apache. On va créer une configuration de base pour bloquer des tentatives de connexion infructueuses répétées sur le service apache (http/https) dans le fichier à créer :

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

Le contenu de la section pour apache sera par exemple :

```
[apache]    ; Nom du service qu'on surveille
enabled  = true  ; on active le service
port     = http,https ; ports qu'on surveille
filter   = apache-auth    ;  nom du filtre définissant le comportement
intrusif
logpath  = /var/log/apache*/*error.log   ; fichier de journalisation dans
lequel on cherche le comportement intrusif
maxretry = 3 ; nombre d échecs avant banissement
findtime = 60 ; délai dans lequel le nombre d échec doit être détecté
bantime = 10m ; durée du bannissement (10 minutes)
banaction = iptables-allports  #bannit l'IP sur tous les ports (il existe
d'autres options moins radicales)
```

A ce stade, le blocage ne sera pas détecté car le filtre *apache-auth* ne saura pas reconnaître l'intrusion (le fonctionnement de l'application est normal, il n'y a pas d'enregistrement d'échec par défaut)

Blocage manuel

Pour comprendre le fonctionnement, on va bloquer manuellement un poste

Remplacer <ip_poste> par une IP d'un équipement du réseau, la connexion au site Web devrait indiquer « connexion échouée » alors que les autres postes ont toujours accès.

```
fail2ban-client set apache banip <ip_poste>
```

Pour débloquer

```
fail2ban-client set apache unbanip <ip_poste>
```

Cette manipulation montre le principe mais n'est pas automatisée

Enregistrement de trace dans le journal surveillé par le jail

Pour permettre de détecter les échecs de connexion, on va générer des erreurs identifiables dans le fichier journal d'Apache pour les tentatives d'accès en échec.

Pour qu'on détecte une tentative d'intrusion, il faut que Fail2ban trouve dans un fichier journal une information qui lui permet de détecter la faille.

Par exemple, dans la page *connecter.php* qui vérifie l'identité de connexion, en cas d'erreur, avant le renvoi vers l'index.php, on inscrit l'erreur dans le journal d'activité :

- error_log enregistre dans le fichier /var/log/apache2/error.log.
- $_SERVER["REMOTE_ADDR"] fournit l'IP du poste ayant généré l'accès

```
else {
        error_log("echec de connexion :
".$_POST["compte"].":".$_SERVER["REMOTE_ADDR"]);
        header("Location: index.php");
     }
```

On pourra observer l'inscription d'une erreur dans ***/var/log/apache2/error.log***

```
tail /var/log/apache2/error.log
```

```
echec de connexion : admin:10.12.0.6, referer: http://172.20.12.102/dplace/
echec de connexion : root:10.12.0.6, referer: http://172.20.12.102/dplace/
echec de connexion : user:10.12.0.6, referer: http://172.20.12.102/dplace/
```

Le **jail** est configuré, les erreurs journalisées, mais le **filtre** ne prend toujours pas en compte l'information

Prise en compte par le filtre

Pour que les messages inscrits dans le journal soient pris en compte, il faut l'indiquer dans le filtre utilisé dans le **jail.local**.

Dans le journal, nous avons ajouté une ligne commençant par « echec » à chaque connexion erronée. Dans le **jail [apache]**, nous avons invoqué le fichier de filtre existant **/etc/fail2ban/filter.d/apache-auth.conf**.

Il faut donc que celui-ci détecte nos messages.

Editez-le filtre

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

Il faut ajouter l'expression régulière qui dit que la ligne recherchée doit commencer (^) par le terme « echec » (ajout à la dernière ligne du filtre existant).

```
failregex = ^client (?:denied by server configuration|used wrong
authentication schem>
            ^user (?!`)<F-USER>(?:\S*|.*?)</F-USER>
(?:auth(?:oriz|entic)ation failur>
            ^Authorization of user <F-USER>(?:\S*|.*?)</F-USER> to access
.*? failed\b
            ^%(auth_type)suser <F-USER>(?:\S*|.*?)</F-USER>: password
mismatch\b
            ^%(auth_type)suser `<F-USER>(?:[^']*|.*?)</F-USER>' in realm
`.+' (auth(?>
            ^%(auth_type)sinvalid nonce .* received - length is not\b
            ^%(auth_type)srealm mismatch - got `(?:[^']*|.*?)' but
expected\b
            ^%(auth_type)sunknown algorithm `(?:[^']*|.*?)' received\b
            ^invalid qop `(?:[^']*|.*?)' received\b
            ^%(auth_type)sinvalid nonce .*? received - user attempted time
travel\b
            ^(?:No h|H)ostname \S+ provided via SNI(?:, but no hostname
provided| and>
            ^echec \b
```

Journalisation du blocage

Le fichier **/var/log/fail2ban.log** conserve la trace des détections et blocages que l'outil a mis en place.

```
nano /var/log/fail2ban.log
```

On y repère les horaires précis d'enregistrement de l'échec et l'action de banissement

```
2023-12-04 11:16:22,916 fail2ban.filter [2327]: INFO [apache] Found
10.12.0.6 - 2023-12-04 11:16:22
2023-12-04 11:16:24,832 fail2ban.filter [2327]: INFO [apache] Found
10.12.0.6 - 2023-12-04 11:16:24
2023-12-04 11:16:27,536 fail2ban.filter [2327]: INFO [apache] Found
10.12.0.6 - 2023-12-04 11:16:27
2023-12-04 11:16:27,725 fail2ban.actions [2327]: NOTICE [apache] Ban
```

```
10.12.0.6
```

From:
https://wiki.sio.bts/ - **WIKI SIO : DEPUIS 2017**

Permanent link:
https://wiki.sio.bts/doku.php?id=fail2ban&rev=1737017579

Last update: **2025/01/16 08:52**