HIDS (Host-based Intrusion Detection System) // Fail2Ban

FAIL2BAN - La Sentinelle Automatique

LE CONCEPT : POURQUOI LE FIREWALL NE SUFFIT PAS ?

Imagine que **UFW** (ou tout autre FireWall) est une porte blindée. C'est solide. Si la porte est fermée (port bloqué), personne ne rentre. Mais si la porte est ouverte (port 22 pour SSH), le FireWall laisse passer tout le monde jusqu'au guichet d'authentification.

C'est là le problème. Un robot peut venir toquer à la porte 10 000 fois par seconde en essayant des clés différentes (Brute Force). Le FireWall le regardera faire sans broncher, car pour lui, la connexion est techniquement valide (TCP Handshake OK).

Fail2Ban est le vigile. Il est assis derrière la porte. Il lit le journal des entrées (/var/log/auth.log).

1. Il voit qu'une IP a raté son mot de passe 3 fois.
2. Il appelle UFW.
3. Il lui dit : *"Cette IP est interdite de séjour pour 10 minutes."*
4. UFW ferme la porte au nez de l'attaquant.

> **Analogie :** C'est les **Tourelles Automatiques** dans *Aliens*. Elles scannent, elles détectent le mouvement (l'erreur), et elles neutralisent. Simple. Brutal. Efficace.

INSTALLATION & PRÉREQUIS

L'installation est triviale, mais la survie dépend de l'ordre des opérations.

Prérequis : Avoir un pare-feu fonctionnel (UFW ou NFTables) installé et actif. Fail2Ban ne bloque rien lui-même, il donne des ordres au pare-feu.

```
sudo apt update && sudo apt install fail2ban
```

Une fois installé, le service démarre, mais il ne fait rien d'intelligent par défaut. Il faut le dresser.

CONFIGURATION : LA RÈGLE D'OR DU "LOCAL"

Fail2Ban est livré avec un fichier de configuration énorme : /etc/fail2ban/jail.conf.

> **WARNING : ON NE TOUCHE JAMAIS À jail.conf.** Si on le modifies, à la prochaine mise à jour du système (apt upgrade), les modifications seront écrasées et le serveur redeviendra une passoire.

La Méthode Pro : On crée un fichier de surcharge nommé jail.local. Fail2Ban lira d'abord le .conf, puis appliquera tes modifications du .local par-dessus.

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

On va utiliser jail.local uniquement pour définir les **règles par défaut** qui s'appliqueront à toutes tes futures prisons (SSH, Apache, Nginx...).

```
[DEFAULT]
# --- QUI ON NE BANNIT JAMAIS ---
# Toujours whitelister localhost pour ne pas que le serveur se banisse lui-même
ignoreip = 127.0.0.1/8 ::1

# --- COMBIEN DE TEMPS (LA PUNITION) ---
# 10m = 10 minutes. 1h = 1 heure.
bantime = 10m

# --- LA FENÊTRE DE TIR ---
# Si l'attaquant fait X bêtises en Y temps...
findtime = 10m

# --- LE SEUIL DE TOLÉRANCE ---
# 5 essais ratés avant le ban.
maxretry = 5

# --- L'ARME DU CRIME ---
# On utilise UFW pour bloquer.
banaction = ufw

# --- LE MOTEUR ---
# Sur Debian 12+, systemd est roi pour gérer les logs
backend = systemd
```

LA CONFIGURATION UNIVERSELLE (SSH)

Maintenant, on crée le fichier spécifique pour le SSH dans le dossier dédié jail.d/. C'est là que la magie modulaire opère.

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

> **Note** : Je préfère l'extension .local à .conf par habitude, mais .conf fonctionne aussi dans ce dossier. L'important est d'être constant.

```
[sshd]
# ON/OFF : C'est ici qu'on active la prison
enabled = true

# LE PORT :
# "ssh" correspond au port 22 par défaut.
# Si tu as changé ton port SSH (ex: 60122), mets le numéro ici !
port = ssh

# LA SURCHARGE (Optionnel) :
# Tu peux être plus méchant juste pour le SSH si tu veux
maxretry = 3

# LE MODE "RÉCIDIVE" : # Si l'IP se fait bannir plusieurs fois, la durée du ban augmente exponentiellement.
# C'est la loi des "Three Strikes". Tu joues, tu perds, tu rejoues, tu perds tout.
bantime.increment = true
```

ACTIVATION & VÉRIFICATION

Maintenant, il faut réveiller la bête.

```
sudo systemctl restart fail2ban
sudo systemctl enable fail2ban
```

Vérifier que la prison est ouverte :

```
sudo fail2ban-client status
```

Vérifier les détails de la prison SSH :

```
sudo fail2ban-client status sshd
```

LE TEST LIVE (SANS SE SUICIDER)

C'est le moment de vérité. On va vérifier si le chien de garde mord vraiment.

WARNING CRITIQUE : Ne teste **JAMAIS** depuis ta session SSH actuelle ou depuis l'IP de ton poste d'administration principal. Si tu te bannis toi-même, tu perds l'accès au serveur. Game Over.

> **La solution :** Utilise ton smartphone en 4G (Wifi coupé) avec une application comme Termius, ou demande à un collègue (que tu n'aimes pas trop) de le faire depuis sont téléphone.

Le Protocole de Test :

Sur le serveur (Session Admin) : Lance l'observation des logs en direct.

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

Tu es maintenant dans la Matrice. Tu vois tout ce que Fail2Ban pense.

Sur l'appareil attaquant (Smartphone 4G) : Tente de te connecter avec un mauvais mot de passe.

- Tentative 1... (Rien)
- Tentative 2... (Rien)
- Tentative 3... (Si tu as mis maxretry = 3)

Le Verdict : Dans ton terminal serveur, tu devrais voir apparaître une ligne satisfaisante :

```
[sshd] Ban 192.168.x.x
```

- **La Confirmation :** Réessaie de te connecter avec le smartphone.
- Résultat attendu : **Connection Refused** ou **Operation Timed Out**.
- Le serveur ne répond plus. Tu es un fantôme pour lui.

LE DROIT DE GRÂCE (UNBAN)

Tu as banni ton propre IP par erreur ? Tu as banni le PDG ? Pas de panique. Fail2Ban possède une commande client pour gérer les prisons manuellement.

Vérifier qui est en prison

```
sudo fail2ban-client status sshd
```

Il te listera le nombre de bannis et leurs IP.

Libérer un prisonnier (Unban)

```
sudo fail2ban-client set sshd unbanip <ADRESSE_IP>
```

 # (Remplace <ADRESSE_IP> par l'IP malheureuse, ex: 192.168.1.50)

C'est la carte "Sortie de Prison" du Monopoly.

Bannir manuellement quelqu'un (Le Marteau de Thor)

Si tu vois une IP suspecte dans les logs et que tu ne veux pas attendre qu'elle échoue 3 fois

```
sudo fail2ban-client set sshd banip <ADRESSE_IP>
```