Fail2Ban

Fail2Ban est un outil de protection qui analyse les logs des services exposés (SSH, applications web, reverse proxy, etc.) et bannit automatiquement les adresses IP qui présentent un comportement suspect : tentatives de connexion échouées répétées, scan d'endpoints, brute-force, etc.

Présentation

Qu'est ce que Fail2Ban ?

Fail2Ban est un outil de protection qui analyse les logs des services exposés (SSH, applications web, reverse proxy, etc.) et bannit automatiquement les adresses IP qui présentent un comportement suspect : tentatives de connexion échouées répétées, scan d'endpoints, brute-force, etc.

Concrètement, Fail2Ban fonctionne en trois briques :

Brique	Rôle
Filter	Une expression régulière qui repère une ligne de log correspondant à une tentative échouée
Jail	La configuration qui associe un filter à un service, une durée de ban, un nombre d'essais max, et une action
Action	Ce qui se passe quand le seuil est dépassé (le plus souvent : bannir l'IP via le pare-feu)

Le flux est donc : **log → filter (regex) → jail (seuils) → action (ban pare-feu).**

Sur une infra exposée sur Internet (Rakouns), chaque service accessible publiquement (Vaultwarden, Emby, Matrix, NPM, etc.) est une cible potentielle de scan automatisé et de brute-force. Sans fail2ban :

- Les logs d'authentification se remplissent de tentatives parasites.
- Les services restent exposés à du brute-force lent et discret (peu de requêtes par minute, sous le radar d'un rate-limiting applicatif).
- Aucune réponse automatique n'existe en cas d'attaque ciblée.

Spécificités de l'infra Rakouns

Sur l'infra Rakouns, la quasi-totalité des services tournent en conteneurs Docker derrière NPM (Nginx Proxy Manager) comme reverse proxy. Cela introduit deux contraintes particulières :

1. **Le ban classique d'iptables ne suffit pas avec Docker.** Docker manipule ses propres règles iptables et les insère avant les règles fail2ban standards (chaîne INPUT). Il faut donc cibler explicitement la chaîne DOCKER-USER, via une action personnalisée (iptables-docker), pour qu'un ban soit réellement effectif sur le trafic redirigé vers les conteneurs.
2. Le format des logs Docker (json-file**) nécessite le backend polling** dans la configuration jail, car le backend par défaut (auto/systemd) ne lit pas correctement ce format.

Sur le host AlmaLinux (Comms / OVH VPS), une bascule vers iptables-legacy a également été nécessaire pour la compatibilité avec les actions fail2ban, et un contexte SELinux dédié (var_log_t) a dû être appliqué aux chemins de logs Docker pour autoriser leur lecture.

Convention de nommage Rakouns

Toutes les jails créées sur l'infra Rakouns sont préfixées rakouns- (ex : rakouns-vaultwarden, rakouns-emby, rakouns-npm) afin de les distinguer clairement des jails par défaut fournies avec le paquet (sshd, recidive, etc.).

Installation

Installation générique (Debian/Ubuntu)

```
sudo apt update
sudo apt install fail2ban -y
```

Vérifier que le service est actif :

```
sudo systemctl status fail2ban
sudo systemctl enable --now fail2ban
```

Installation sur AlmaLinux/RHEL

```
sudo dnf install epel-release -y
sudo dnf install fail2ban fail2ban-systemd -y
```

```
sudo systemctl enable --now fail2ban
```

Bascule iptables-legacy (requis sur AlmaLinux)

Sur AlmaLinux, le mode `nftables` par défaut pose des soucis de compatibilité avec certaines actions fail2ban orientées iptables. Bascule en mode legacy :

```
sudo dnf install iptables-services -y
sudo alternatives --set iptables /usr/sbin/iptables-legacy
sudo alternatives --set ip6tables /usr/sbin/ip6tables-legacy
sudo systemctl enable --now iptables
sudo systemctl restart fail2ban
```

Vérifier la bascule :

```
sudo alternatives --display iptables
```

Arborescence de configuration

Fail2ban utilise deux types de fichiers, à ne **jamais** modifier dans `/etc/fail2ban/jail.conf` ou `/etc/fail2ban/filter.d/*.conf` (écrasés lors des mises à jour). Toujours passer par les fichiers `.local` :

```
/etc/fail2ban/
├── jail.conf          # NE PAS MODIFIER (fichier par défaut)
├── jail.local          # Config globale perso (créé manuellement)
├── jail.d/
│   └── rakouns-*.conf  # Une jail par service, convention Rakouns
├── filter.d/
│   ├── *.conf          # Filtres fournis par défaut
│   └── rakouns-*.conf  # Filtres personnalisés
└── action.d/
    ├── *.conf
    └── iptables-docker.conf  # Action custom pour cibler DOCKER-USER
```

Créer la config globale de base si elle n'existe pas :

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

Exemple de socle dans `jail.local` (valeurs par défaut appliquées à toutes les jails, sauf surcharge) :

```
[DEFAULT]
bantime  = 1h
findtime = 10m
maxretry = 5
backend  = auto
ignoreip = 127.0.0.1/8 192.168.1.0/24
```

> **Note —** ignoreip doit inclure ton réseau local pour éviter de te bannir toi-même pendant les tests.

Cas Docker : l'action iptables-docker

Comme évoqué en page 1, le ban standard via la chaîne `INPUT` n'a aucun effet sur le trafic redirigé vers des conteneurs Docker, car Docker insère ses propres règles dans la chaîne `DOCKER-USER` avant que la chaîne `INPUT` ne soit évaluée.

Créer `/etc/fail2ban/action.d/iptables-docker.conf` :

```
[Definition]
actionstart = iptables -N f2b-<name>
              iptables -A f2b-<name> -j RETURN
              iptables -I DOCKER-USER -j f2b-<name>

actionstop = iptables -D DOCKER-USER -j f2b-<name>
             iptables -F f2b-<name>
             iptables -X f2b-<name>

actioncheck = iptables -n -L DOCKER-USER | grep -q 'f2b-<name>[ \t]'

actionban = iptables -I f2b-<name> 1 -s <ip> -j DROP

actionunban = iptables -D f2b-<name> -s <ip> -j DROP
```

```
[Init]
name = default
```

Cette action sera référencée dans chaque jail concernée (voir Page 4).

Vérification rapide de l'installation

```
sudo fail2ban-client status
sudo fail2ban-client version
```

Filter

Principe

Un filter est un fichier .conf dans /etc/fail2ban/filter.d/ qui contient une (ou plusieurs) expression(s) régulière(s) capable(s) de repérer, dans une ligne de log, une tentative échouée, et d'en extraire l'adresse IP via le groupe nommé <HOST>.

Structure minimale :

```
[Definition]
failregex = <ta regex avec <HOST>>
ignoreregex =
```

<HOST> est une macro fail2ban qui correspond à une regex IPv4/IPv6 déjà prête à l'emploi — pas besoin de l'écrire toi-même.

Méthode générale pour écrire un filter

1. **Identifier une ligne de log représentative** d'un échec d'authentification.
2. **Repérer ce qui varie** (timestamp, IP, user) vs ce qui est fixe (le message d'erreur).
3. **Écrire la regex**, en remplaçant l'IP par <HOST> et en échappant les caractères spéciaux (., [,], etc.).
4. **Tester** avec fail2ban-regex avant de créer la jail.

Exemple générique : filter SSH (fourni par défaut, pour référence)

/etc/fail2ban/filter.d/sshd.conf (déjà présent, à ne pas modifier) repère une ligne du type :

```
Jun 20 10:15:32 host sshd[1234]: Failed password for invalid user admin from 203.0.113.42 port 51234 ssh2
```

avec une regex de la forme :

```
failregex = ^%(__prefix_line)sFailed \S+ for .* from <HOST>(?: port \d+)?(?: ssh2)?\s*$
```

Fil rouge : filter Vaultwarden

Vaultwarden journalise les échecs de connexion via Docker (stdout → json-file). Une ligne typique ressemble à :

```
[2026-06-20 10:15:32.123][vaultwarden::api::identity][WARN] Username or password is incorrect. Try again. IP: 203.0.113.42. Username: user@example.com.
```

Créer /etc/fail2ban/filter.d/rakouns-vaultwarden.conf :

```
[Definition]
failregex = ^.*Username or password is incorrect\. Try again\. IP: <HOST>\..*$
            ^.*Username or password is incorrect\. IP: <HOST>\..*$
ignoreregex =
```

```
datepattern = ^\[%%Y-%%m-%%d %%H:%%M:%%S
```

> **Note —** Plusieurs lignes dans failregex (une par ligne) sont évaluées en OU logique — utile quand le message d'erreur varie légèrement selon la version du service.

Tester un filter avant de créer la jail

Toujours valider un filter sur un vrai log (ou un extrait) avant de l'attacher à une jail, pour éviter de générer un faux sentiment de sécurité avec un filter qui ne matche jamais — ou pire, qui matche trop large.

Test sur un fichier de log classique

```
sudo fail2ban-regex /var/log/auth.log /etc/fail2ban/filter.d/sshd.conf
```

Test sur les logs Docker (cas Vaultwarden)

Les logs Docker ne sont pas dans un fichier classique par défaut, il faut soit extraire un échantillon, soit pointer directement sur le fichier json-file du conteneur :

```
# Trouver le chemin du log JSON du conteneur
docker inspect --format='{{.LogPath}}' vaultwarden

# Tester le filter contre ce fichier
sudo fail2ban-regex /var/lib/docker/containers/<container_id>/<container_id>-json.log
/etc/fail2ban/filter.d/rakouns-vaultwarden.conf
```

Lecture du résultat

fail2ban-regex affiche un résumé du type :

```
Results
=======

Failregex: 3 total
|-  #) [# of hits] regular expression
|   1) [3] ^.*Username or password is incorrect\. Try again\. IP: <HOST>\..*$
`-

Ignoreregex: 0 total

Lines: 150 lines, 0 ignored, 3 matched, 147 missed
```

3 matched confirme que le filter fonctionne. Si tu obtiens 0 matched alors que tu sais qu'il y a des échecs dans le log, retravaille la regex (espace en trop, caractère spécial non échappé, format de date différent, etc.).

Jail

Principe

La jail relie un filter à :

- une source de log (logpath ou backend pour Docker),
- des seuils (maxretry, findtime, bantime),
- une action (action, par défaut iptables classique, ou iptables-docker pour les conteneurs).

Convention Rakouns : un fichier par jail dans /etc/fail2ban/jail.d/, nommé rakouns-<service>.conf.

Structure générale d'une jail

```
[rakouns-<service>]
enabled  = true
filter   = rakouns-<service>
logpath  = <chemin ou source du log>
backend  = polling          ; obligatoire pour les logs Docker json-file
maxretry = 5
findtime = 10m
bantime  = 1h
```

```
action   = iptables-docker[name=<service>]
```

> **Note —** backend = polling est requis pour tout service conteneurisé dont les logs sont au format json-file : le backend auto/systemd ne détecte pas les nouvelles lignes correctement dans ce format.

Exemple générique : jail SSH

```
[sshd]
enabled  = true
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 5
findtime = 10m
bantime  = 1h
```

C'est le cas le plus simple : pas de Docker, pas d'action custom, le ban standard sur la chaîne INPUT suffit.

Fil rouge : jail Vaultwarden

Créer /etc/fail2ban/jail.d/rakouns-vaultwarden.conf :

```
[rakouns-vaultwarden]
enabled  = true
filter   = rakouns-vaultwarden
logpath  = /var/lib/docker/containers/<container_id>/<container_id>-json.log
backend  = polling
maxretry = 5
findtime = 10m
bantime  = 24h
action   = iptables-docker[name=vaultwarden]
```

Points clés :

- backend = polling → indispensable, sinon la jail reste en 0 currently failed même en cas d'attaque réelle.
- action = iptables-docker[name=vaultwarden] → utilise l'action custom créée en Page 2, qui cible DOCKER-USER avec une chaîne dédiée f2b-vaultwarden.
- bantime = 24h plus long que la valeur par défaut, car Vaultwarden contient des coffres de mots de passe : la tolérance au brute-force doit être minimale.

> **Note —** Le container_id change si le conteneur est recréé (mise à jour d'image, docker compose up -d --force-recreate). Un script de détection automatique du chemin de log peut éviter d'avoir à mettre à jour la jail manuellement à chaque recréation.

Jail recidive

Recommandé en complément de toutes les jails de service : bannit plus longtemps une IP déjà bannie plusieurs fois.

```
[recidive]
enabled  = true
filter   = recidive
logpath  = /var/log/fail2ban.log
banaction = iptables-allports
bantime  = 1w
findtime = 1d
maxretry = 3
```

Recharger et activer une jail

```
sudo fail2ban-client reload
# ou, pour recharger une jail spécifique sans tout relancer :
sudo fail2ban-client reload rakouns-vaultwarden
```

Vérifier qu'elle est bien prise en compte :

```
sudo fail2ban-client status
Status

|- Number of jail:      3
```

```
`- Jail list:   sshd, recidive, rakouns-vaultwarden
```

Puis le détail d'une jail précise :

```
sudo fail2ban-client status rakouns-vaultwarden
Status for the jail: rakouns-vaultwarden
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- File list:        /var/lib/docker/containers/.../....log
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
```

Vérifier le pare-feu

Pourquoi cette étape est indispensable

Une jail «active» côté fail2ban (fail2ban-client status qui répond) ne garantit **pas** que le ban est réellement appliqué au niveau du pare-feu — surtout sur l'infra Rakouns où Docker manipule ses propres règles iptables. C'est l'erreur la plus fréquente : croire qu'une jail fonctionne juste parce qu'elle est listée comme enabled.

Vérification générique : ban classique

Pour une jail standard (ex : sshd), vérifier que la règle de ban apparaît bien dans iptables :

```
sudo iptables -L f2b-sshd -n
```

Résultat attendu après un ban :

```
Chain f2b-sshd (1 references)
target     prot opt source               destination
DROP       all  --  203.0.113.42         0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```

Vérification Docker : chaîne DOCKER-USER

Pour les jails utilisant l'action iptables-docker (cas Vaultwarden et tous les services conteneurisés), c'est la chaîne DOCKER-USER qu'il faut inspecter, **pas** INPUT :

```
sudo iptables -L DOCKER-USER -n --line-numbers
```

La chaîne dédiée du service doit apparaître en première position (insérée avec -I, donc évaluée avant le reste) :

```
Chain DOCKER-USER (1 references)
num  target          prot opt source               destination
1    f2b-vaultwarden all  --  0.0.0.0/0            0.0.0.0/0
2    RETURN          all  --  0.0.0.0/0            0.0.0.0/0
```

Puis inspecter la sous-chaîne pour voir si une IP y est effectivement droppée :

```
sudo iptables -L f2b-vaultwarden -n

Chain f2b-vaultwarden (1 references)
target     prot opt source               destination
DROP       all  --  203.0.113.42         0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```

Si la chaîne f2b-vaultwarden n'apparaît pas du tout dans DOCKER-USER, l'action iptables-docker n'a pas pu s'initialiser correctement : vérifier actionstart dans iptables-docker.conf (page 2) et relire les logs fail2ban (section suivante).

Provoquer un ban de test (en sécurité)

Pour valider toute la chaîne sans attendre une vraie attaque, depuis une machine **différente** de celle utilisée pour administrer le service (sinon tu te bannis toi-même de ton accès SSH/admin) :

```
# Déclencher volontairement des échecs d'auth (exemple Vaultwarden via curl)
for i in {1..6}; do
  curl -s -o /dev/null -w "%{http_code}\n" \
    -X POST https://vault.rakouns.bzh/identity/connect/token \
    -d "username=test@test.com&password=wrongpassword&grant_type=password"
done
```

Puis vérifier côté fail2ban :

```
sudo fail2ban-client status rakouns-vaultwarden
```

L'IP de test doit apparaître dans Banned IP list. Pour débannir manuellement après test :

```
sudo fail2ban-client set rakouns-vaultwarden unbanip <IP_DE_TEST>
```

Logs fail2ban : diagnostic en cas de doute

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

Lignes à surveiller :

- [rakouns-vaultwarden] Found <IP> → un échec a été détecté par le filter (le filter fonctionne).
- [rakouns-vaultwarden] Ban <IP> → le seuil maxretry a été atteint et le ban a été déclenché (la jail fonctionne).
- Absence totale de Found malgré des échecs visibles dans les logs applicatifs → revenir à la Page 3 et retester le filter avec fail2ban-regex.
- Found présent mais jamais de Ban → vérifier maxretry/findtime, ou que ignoreip (jail.local) n'exclut pas l'IP de test par erreur.

Vérification post-reboot / post-recreate

Deux cas fréquents sur Rakouns où le ban peut «disparaître» silencieusement :

1. **Reboot du host** : les règles iptables ne sont pas persistées par défaut. Vérifier que fail2ban redémarre bien après le pare-feu/Docker (systemctl status fail2ban), et que DOCKER-USER est repeuplée par actionstart au démarrage du service.
2. **Recréation d'un conteneur** (mise à jour d'image) : le container_id change, donc le logpath de la jail devient invalide. Penser à relancer le script de détection automatique du chemin de log (mentionné en Page 4) puis fail2ban-client reload <jail>.

Checklist de vérification rapide

```
# 1. La jail est listée et active
sudo fail2ban-client status

# 2. Le filter remonte des "Found" sur le service concerné
sudo fail2ban-client status rakouns-vaultwarden

# 3. La chaîne f2b-<service> existe dans DOCKER-USER (cas Docker)
sudo iptables -L DOCKER-USER -n

# 4. Les logs confirment Found -> Ban en cas d'incident réel
sudo grep "rakouns-vaultwarden" /var/log/fail2ban.log | tail -20
```