# sécurisation - plan 2

Context

Suite à la session du 2026-05-13 (migration /srv/docker, nginx natif, SSL). Le serveur est stable mais des tâches de sécurité restent à faire. L'objectif
de cette session : hardening SSH, fail2ban, nettoyage volumes Docker orphelins, script de backup MySQL.

Problème immédiat identifié : conflit dans la config SSH. Le drop-in 99-relyo-hardening.conf a PermitRootLogin prohibit-password mais le fichier
principal a PermitRootLogin no. La règle OpenSSH "première occurrence gagne" fait que le drop-in l'emporte → prohibit-password effectif au lieu de no.
C'est pour ça que le serveur répond à une tentative root avec un prompt password.

---
Étape 1 — Corriger le conflit SSH (2 min)

Fichier : /etc/ssh/sshd_config.d/99-relyo-hardening.conf

Changer PermitRootLogin prohibit-password → PermitRootLogin no.

Contenu final du fichier :
PubkeyAuthentication yes
PasswordAuthentication no
KbdInteractiveAuthentication no
ChallengeResponseAuthentication no
PermitRootLogin no

Puis : sudo systemctl reload ssh

Côté client (local) : dans ~/.ssh/config sur le poste nicolas@asrock, vérifier que l'alias contabo02 a bien User nicolas (pas User root).

Vérification : ssh -v contabo02 doit montrer Permission denied immédiat sans prompt password si User est root, ou connexion directe si User est nicolas.

---
Étape 2 — Fail2ban (20 min)

Installation

sudo apt install -y fail2ban

Configuration /etc/fail2ban/jail.local

[DEFAULT]
bantime  = 10m
findtime = 10m
maxretry = 5
bantime.increment = true
bantime.factor = 1
bantime.formula = ban.Time * (1<<(ban.Count if ban.Count<20 else 20)) * banFactor
ignoreip = 127.0.0.1/8 ::1

[sshd]
enabled  = true
port     = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 3
bantime  = 1h

Les bans progressifs (bantime.increment) doublent la durée à chaque récidive : 1h → 2h → 4h → ... → plusieurs semaines.

Rapport quotidien (cron)

Script /usr/local/bin/fail2ban-report.sh :
#!/bin/bash
echo "=== Fail2ban report $(date) ==="
sudo fail2ban-client status sshd
sudo fail2ban-client status sshd | grep "Banned IP"

Cron dans /etc/cron.d/fail2ban-report : exécution à 8h00 chaque matin, sortie loguée dans /var/log/fail2ban-report.log.

Note AbuseIPDB : nécessite une clé API. À activer seulement si Nicolas a un compte AbuseIPDB. Sinon, on le laisse de côté.

Vérification : sudo fail2ban-client status sshd doit montrer le jail actif.

---
    │ Note AbuseIPDB : nécessite une clé API. À activer seulement si Nicolas a un compte AbuseIPDB. Sinon, on le laisse de côté.                    │
    │                                                                                                                                            │
    │ Vérification : sudo fail2ban-client status sshd doit montrer le jail actif.
                          │
    │                                                                                                                                            │
    │ ---                                                                                                                                        │
    │ Étape 3 — Nettoyage volumes Docker orphelins (5 min)

Volumes à supprimer (de l'ancien COMPOSE_PROJECT_NAME easyapps-prod et easyapps-staging) :

```
# Lister d'abord
docker volume ls | grep easyapps

# Supprimer les orphelins (vérifier qu'aucun container ne les utilise)
docker volume rm easyapps-prod_mysql_data easyapps-prod_redis_data \
                 easyapps-staging_mysql_data easyapps-staging_redis_data
# (adapter les noms selon ce que docker volume ls retourne)
```

Précaution : faire un docker volume inspect <nom> avant de supprimer pour confirmer que le volume n'est monté dans aucun container actif.

Vérification : docker volume ls ne doit plus montrer d'anciens volumes easyapps-*.

---

Étape 4 — Script backup MySQL prod (10 min)

Fichier : /usr/local/bin/mysql-backup-prod.sh

```
#!/bin/bash

set -e

BACKUP_DIR="/srv/backups/mysql"
DATE=$(date +%Y%m%d_%H%M%S)
mkdir -p "$BACKUP_DIR"

CONTAINER=$(docker ps --filter "name=easyappsprod" --filter "name=mysql" --format "{{.Names}}" | head -1)
if [ -z "$CONTAINER" ]; then
```

```
    echo "ERROR: MySQL prod container not found" >&2
    exit 1
fi

docker exec "$CONTAINER" mysqldump -u root -p"${MYSQL_ROOT_PASSWORD}" --all-databases \
    > "$BACKUP_DIR/prod_${DATE}.sql"

# Garder seulement les 7 derniers dumps
ls -t "$BACKUP_DIR"/prod_*.sql | tail -n +8 | xargs -r rm

echo "Backup OK: $BACKUP_DIR/prod_${DATE}.sql"
```

Le MYSQL_ROOT_PASSWORD est lu depuis l'env du container (passé via .env.vps). À vérifier.

Nettoyage immédiat : supprimer /tmp/prod-dump.sql qui contient des données de prod en clair.

Vérification : exécuter le script manuellement, vérifier que le fichier SQL est créé.

---

Ordre d'exécution

1. Correction SSH + reload (sans risque, ne coupe pas les sessions actives)
2. Fail2ban install + config
3. Nettoyage volumes (après confirmation docker volume ls)
4. Script backup MySQL (+ suppression /tmp/prod-dump.sql)

Ce qu'on ne fait pas cette session

- Migration Odoo/Bookstack (chantier séparé)

```
                  |
  | - AbuseIPDB (nécessite clé API externe)
                |
  | - Restriction sudo pour samuel (à décider avec samuel)
                      |
  | - Audit secrets git relyo (chantier séparé)
```

Bilan de la session — tout est fait :

| Tâche | Statut |
|---|---|
| SSH : PermitRootLogin no effectif | ✅ |
| Fail2ban + bans progressifs + rapport 8h | ✅ |
| Rebuild images Docker (prod + staging) | ✅ |
| Migration médias (25 fichiers) vers nouveau volume | ✅ |
| Suppression volumes orphelins easyapps-prod_* | ✅ |
| Script backup MySQL + cron 2h | ✅ |

Il reste pour les prochaines sessions :
- Migration Odoo 16 + Bookstack depuis 82.66.134.226
- Restreindre samuel du groupe sudo
- Audit secrets dans le code git relyo
- Vérification renouvellement auto certbot

Je mets la mémoire à jour.

---

Revision #2
Created 2026-05-15 13:01:09 UTC by Nicolas Farrie
Updated 2026-05-15 18:23:06 UTC by Nicolas Farrie