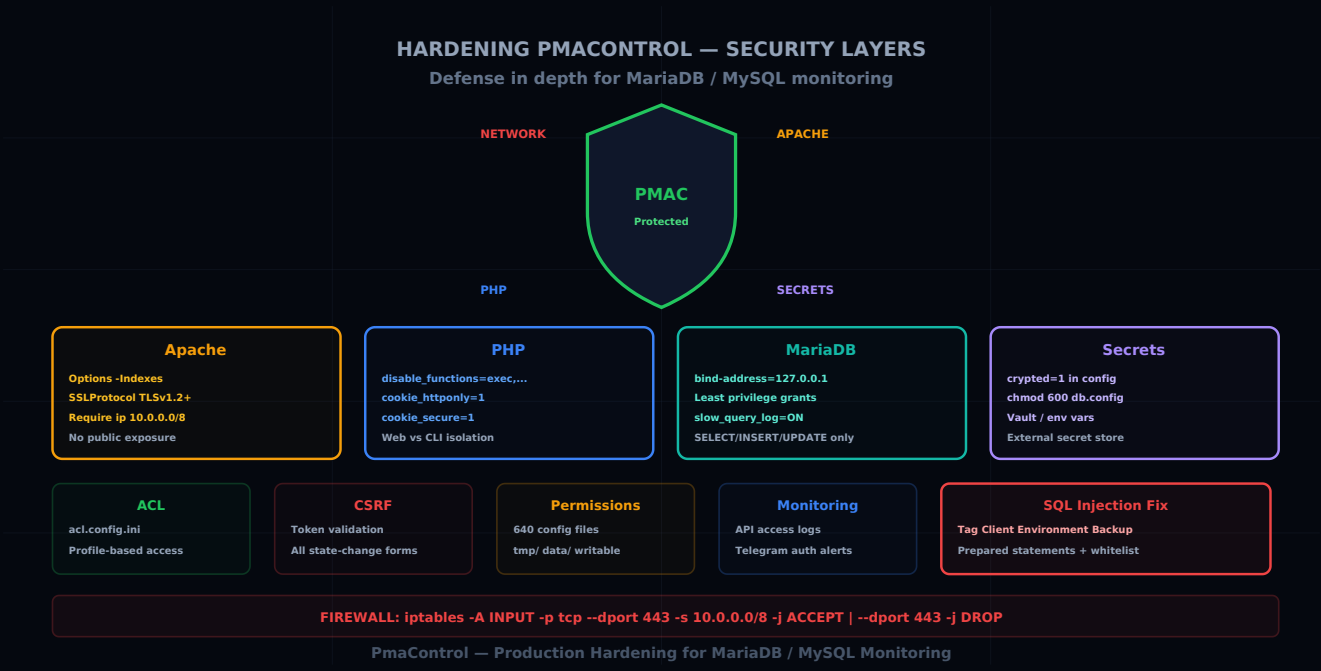
Durcir PmaControl en production : guide de sécurité complet

Aurélien LEQUOY · 13 avril 2026

PMACONTROL SECURITY HARDENING APACHE PHP MARIADB



PmaControl a les clés du royaume

PmaControl supervise vos serveurs MariaDB / MySQL de production. Il stocke les credentials de connexion, les clés SSH, les métriques de performance, la structure des bases. Si un attaquant compromet PmaControl, il a potentiellement accès à **toute votre infrastructure de bases de données**.

Ce guide détaille les mesures de durcissement à appliquer avant de mettre PmaControl en production. Il est issu d'un audit sécurité interne et couvre chaque couche : Apache, PHP, MariaDB, secrets, ACL, CSRF, permissions fichier et monitoring.

Couche 1 : Apache

Désactiver le listing de répertoires

Par défaut, Apache peut afficher le contenu des répertoires sans fichier index. C'est une fuite d'information :

```
<Directory /srv/www/pmacontrol>
    Options -Indexes
    AllowOverride All
    Require all granted
</Directory>
```

Le `-Indexes` est non négociable. Sans lui, un attaquant peut explorer la structure du projet et trouver des fichiers de configuration, des logs, des dumps.

Forcer HTTPS

PmaControl transmet des credentials en clair dans les requêtes HTTP. Sans HTTPS, un attaquant sur le réseau peut les intercepter :

```
<VirtualHost *:80>
    ServerName pmacontrol.internal.company.com
    Redirect permanent / https://pmacontrol.internal.company.com/
</VirtualHost>

<VirtualHost *:443>
    ServerName pmacontrol.internal.company.com
    SSLEngine On
    SSLCertificateFile /etc/ssl/certs/pmacontrol.pem
    SSLCertificateKeyFile /etc/ssl/private/pmacontrol.key

    # Modern TLS only
    SSLProtocol -all +TLSv1.2 +TLSv1.3
    SSLCipherSuite HIGH:!aNULL:!MD5:!3DES

    DocumentRoot /srv/www/pmacontrol
</VirtualHost>
```

Restreindre au réseau interne

PmaControl ne doit **jamais** être exposé sur Internet. Limitez l'accès au réseau interne :

```
<Location />
    Require ip 10.0.0.0/8
```

```
    Require ip 172.16.0.0/12
    Require ip 192.168.0.0/16
</Location>
```

Ou mieux encore : placez PmaControl derrière un VPN et ne l'exposez pas du tout via Apache public.

Supprimer le vhost par défaut

Le vhost par défaut d'Apache (`000-default.conf`) répond à toute requête sur l'IP du serveur. Supprimez-le :

```
a2dissite 000-default.conf
systemctl reload apache2
```

Headers de sécurité

Ajoutez les headers de sécurité HTTP :

```
Header always set X-Content-Type-Options "nosniff"
Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-XSS-Protection "1; mode=block"
Header always set Referrer-Policy "strict-origin-when-cross-origin"
Header always set Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline'; style-src 'self' 'unsafe-inline'"
```

Couche 2 : PHP

Désactiver les fonctions dangereuses

PmaControl utilise `exec()` et `shell_exec()` pour certaines opérations (SSH, collecte). La solution n'est pas de les désactiver globalement, mais d'isoler les workers qui en ont besoin.

Pour le vhost web (l'interface) :

```
; php.ini ou .user.ini dans le DocumentRoot
disable_functions = exec,shell_exec,system,passthru,popen,proc_open
expose_php = Off
```

Pour les workers CLI (Aspirateur, Listener) :

```
; php-cli.ini — ces workers ont besoin de shell_exec
disable_functions =
```

Cette séparation garantit que l'interface web ne peut pas exécuter de commandes système, même si un attaquant trouve une faille.

Sécuriser les sessions

```
session.cookie_httponly = 1
session.cookie_secure = 1
session.cookie_samesite = Strict
session.use_strict_mode = 1
session.name = PMACSESSID
```

`cookie_httponly` empêche JavaScript d'accéder au cookie de session (protection XSS). `cookie_secure` force l'envoi uniquement sur HTTPS. `cookie_samesite = Strict` protège contre le CSRF basic.

Limiter l'upload et l'exécution

```
upload_max_filesize = 2M
post_max_size = 8M
max_execution_time = 30
max_input_time = 60
memory_limit = 256M
```

PmaControl n'a pas besoin d'uploads massifs. Limitez pour réduire la surface d'attaque.

Cacher la version PHP

```
expose_php = Off
```

Cela supprime l'en-tête `X-Powered-By: PHP/8.x` des réponses HTTP.

Couche 3 : MariaDB

Restreindre les privilèges de l'utilisateur PmaControl

Après l'installation, l'utilisateur PmaControl a souvent tous les privilèges. Restreignez-le :

```
-- Révoquer les privilèges excessifs
REVOKE ALL PRIVILEGES ON *.* FROM 'pmacontrol'@'localhost';

-- Accorder uniquement ce qui est nécessaire
GRANT SELECT, INSERT, UPDATE, DELETE ON pmacontrol.* TO 'pmacontrol'@'localhost';
GRANT SELECT ON performance_schema.* TO 'pmacontrol'@'localhost';
GRANT REPLICATION CLIENT ON *.* TO 'pmacontrol'@'localhost';
GRANT PROCESS ON *.* TO 'pmacontrol'@'localhost';

FLUSH PRIVILEGES;
```

Le principe du moindre privilège : PmaControl n'a besoin que de lire les métriques et d'écrire dans sa propre base.

Lier au localhost

La base de données PmaControl ne doit écouter que sur l'interface locale :

```
[mysqld]
bind-address = 127.0.0.1
```

Si PmaControl et sa base sont sur le même serveur (configuration typique), il n'y a aucune raison d'écouter sur le réseau.

Activer le log des requêtes sensibles

```
[mysqld]
general_log = OFF          # Trop verbeux en production
slow_query_log = ON
long_query_time = 1
log_error = /var/log/mysql/error.log
```

Le slow query log permet de détecter les requêtes anormales qui pourraient indiquer une injection SQL exploitée.

Couche 4 : Secrets

Chiffrer les credentials

PmaControl stocke les credentials de connexion dans `db.config.ini.php`. Ce fichier supporte le chiffrement :

```
; configuration/db.config.ini.php
[default]
driver = mysql
host = 127.0.0.1
port = 3306
login = pmacontrol
password = "ENCRYPTED_VALUE_HERE"
database = pmacontrol
crypted = 1
```

Le flag `crypted=1` indique à PmaControl de déchiffrer le mot de passe au runtime. La clé de chiffrement est séparée du fichier de configuration.

Utiliser un secret store externe

Pour les déploiements en production critique, externalisez les secrets :

- **Vault** (HashiCorp) : PmaControl peut lire les secrets via l'API
- **AWS Secrets Manager** ou **GCP Secret Manager** : pour les déploiements cloud
- **Variables d'environnement** : le minimum viable, mieux que du texte en clair

```
# Exemple avec variables d'environnement
export PMAC_DB_PASSWORD="secret_value"
export PMAC_SSH_PASSPHRASE="ssh_secret"
```

Protéger les fichiers de configuration

```
# Propriétaire : www-data (l'utilisateur Apache)
chown root:www-data /srv/www/pmacontrol/configuration/*.php

# Permissions : lecture pour le groupe, rien pour les autres
chmod 640 /srv/www/pmacontrol/configuration/*.php

# Le fichier de credentials ne doit être lisible que par www-data
chmod 600 /srv/www/pmacontrol/configuration/db.config.ini.php
```

Couche 5 : ACL (Access Control Lists)

Réviser acl.config.ini

PmaControl a un système d'ACL basé sur les profils. Le fichier `acl.config.ini` définit quel profil peut accéder à quel contrôleur.

```
; configuration/acl.config.ini
[admin]
* = allow

[dba]
Slave = allow
Server = allow
Dashboard = allow
Backup = deny
Config = deny

[readonly]
Slave = allow
Server = allow(show)
Dashboard = allow
* = deny
```

Règles essentielles :

- **Restreindre les contrôleurs sensibles** : `Config`, `Backup`, `Install`, `Api` ne doivent être accessibles qu'aux admins
- **Créer un profil read-only** : pour les développeurs qui ont besoin de consulter sans modifier
- **Auditer régulièrement** : vérifier que de nouveaux contrôleurs ajoutés sont couverts par les ACL

Protéger les endpoints critiques

Certains endpoints sont particulièrement sensibles :

```
[admin]
Install = allow      ; Installation / réinstallation
Config = allow       ; Modification de la configuration
```

```
Api = allow          ; API REST complète
Backup = allow       ; Accès aux backups (contient des données)

[dba]
Install = deny       ; JAMAIS accessible aux non-admins
Config = deny
Api = allow(read)    ; Lecture seule via API
Backup = deny
```

Couche 6 : CSRF (Cross-Site Request Forgery)

Vérifier la présence de tokens

Chaque formulaire PmaControl doit inclure un token CSRF :

```
<form method="POST" action="/slave/start/42/">
    <input type="hidden" name="csrf_token" value="<?= $csrf_token ?>">
    <button type="submit">Start Slave</button>
</form>
```

Côté serveur, le contrôleur doit valider le token :

```
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    throw new SecurityException('Invalid CSRF token');
}
```

Actions à protéger en priorité

Les actions qui modifient l'état sont les plus critiques :

- START/STOP SLAVE
- SKIP error
- Ajout/suppression de serveur
- Modification de configuration
- Création/suppression d'utilisateur

Sans protection CSRF, un attaquant pourrait forcer un DBA connecté à arrêter la réplication d'un serveur de production en lui envoyant un lien piégé.

Couche 7 : Permissions fichier

Arborescence de permissions

```
# Répertoire principal : lisible, pas modifiable
chown -R root:www-data /srv/www/pmacontrol/
chmod -R 750 /srv/www/pmacontrol/

# Répertoires d'écriture : www-data propriétaire
chown -R www-data:www-data /srv/www/pmacontrol/tmp/
chown -R www-data:www-data /srv/www/pmacontrol/data/

# Fichiers PHP : lecture seule pour www-data
find /srv/www/pmacontrol/App/ -name "*.php" -exec chmod 640 {} \;

# Configuration : restrictif
chmod 640 /srv/www/pmacontrol/configuration/*.php
chmod 600 /srv/www/pmacontrol/configuration/db.config.ini.php
```

Le principe : `www-data` peut lire le code et écrire dans `tmp/` et `data/`. Il ne peut pas modifier le code source ni la configuration.

Couche 8 : Monitoring de la sécurité

Logger les accès API

Chaque appel à l'API REST doit être loggé avec :

- Timestamp
- IP source
- Utilisateur (token)
- Endpoint appelé
- Code de réponse

```
// Dans le middleware API
$log = sprintf(
    "[%s] %s %s %s → %d",
    date('Y-m-d H:i:s'),
```

```
    $_SERVER['REMOTE_ADDR'],
    $user->name,
    $_SERVER['REQUEST_URI'],
    http_response_code()
);
file_put_contents('/var/log/pmacontrol/api.log', $log . "\n", FILE_APPEND);
```

Alertes Telegram sur les échecs d'authentification

Configurez une alerte Telegram pour chaque échec de connexion :

```
if (!$auth->isValid()) {
    Telegram::send(
        "□□ Auth failure on PmaControl\n" .
        "IP: " . $_SERVER['REMOTE_ADDR'] . "\n" .
        "User: " . $_POST['login'] . "\n" .
        "Time: " . date('Y-m-d H:i:s')
    );
}
```

Trois échecs depuis la même IP en 5 minutes devraient déclencher un blocage temporaire.

Surveiller les fichiers de configuration

Utilisez `inotifywait` ou un outil similaire pour détecter les modifications non autorisées :

```
inotifywait -m -r /srv/www/pmacontrol/configuration/ -e modify,create,delete |
while read path action file; do
    echo "[$action] $path$file" >> /var/log/pmacontrol/config_changes.log
    # Envoyer alerte Telegram
done
```

Couche 9 : Réseau

Règles de firewall

```
# Autoriser HTTP/HTTPS uniquement depuis le réseau interne
iptables -A INPUT -p tcp --dport 80  -s 10.0.0.0/8 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -s 10.0.0.0/8 -j ACCEPT
iptables -A INPUT -p tcp --dport 80  -j DROP
```

```
iptables -A INPUT -p tcp --dport 443 -j DROP

# Autoriser MySQL uniquement en localhost
iptables -A INPUT -p tcp --dport 3306 -s 127.0.0.1 -j ACCEPT
iptables -A INPUT -p tcp --dport 3306 -j DROP
```

Pas d'exposition publique

PmaControl ne doit **jamais** être accessible depuis Internet. Même avec authentification, la surface d'attaque est trop grande :

- Les credentials des serveurs supervisés sont stockés
- Les clés SSH sont stockées
- L'interface permet d'exécuter des actions sur des serveurs de production

Si un accès distant est nécessaire, utilisez un VPN (WireGuard, OpenVPN) ou un tunnel SSH.

Couche 10 : Injections SQL — remédiation

Notre audit interne a identifié des risques d'injection SQL dans quatre contrôleurs :

Contrôleur	Risque	Remédiation
`Tag.php`	Construction dynamique de clause WHERE	Requêtes paramétrées
`Client.php`	Concaténation dans les filtres	Requêtes paramétrées
`Environment.php`	Interpolation de variable dans ORDER BY	Whitelist de colonnes
`Backup.php`	Paramètre non échappé dans LIKE	Requêtes paramétrées

Avant (vulnérable) :

```
// Tag.php — VULNÉRABLE
$sql = "SELECT * FROM tags WHERE name LIKE '%" . $_GET['search'] . "%'";
$results = $db->query($sql);
```

Après (sécurisé) :

```
// Tag.php — SÉCURISÉ
$sql = "SELECT * FROM tags WHERE name LIKE ?";
$results = $db->query($sql, ['%' . $_GET['search'] . '%']);
```

Pour les clauses ORDER BY, la whitelist est la seule solution sûre :

```
$allowed_columns = ['name', 'created_at', 'id'];
$sort = in_array($_GET['sort'], $allowed_columns) ? $_GET['sort'] : 'name';
$sql = "SELECT * FROM tags ORDER BY " . $sort;
```

Checklist de durcissement

Avant de passer PmaControl en production, validez chaque point :

- [ ] Apache : `-Indexes` activé
- [ ] Apache : HTTPS forcé
- [ ] Apache : accès restreint au réseau interne
- [ ] Apache : vhost par défaut supprimé
- [ ] PHP : fonctions dangereuses désactivées sur le vhost web
- [ ] PHP : `session.cookie_httponly = 1`
- [ ] PHP : `session.cookie_secure = 1`
- [ ] PHP : `expose_php = Off`
- [ ] MariaDB : utilisateur avec privilèges minimaux
- [ ] MariaDB : `bind-address = 127.0.0.1`
- [ ] Secrets : credentials chiffrés (`crypted=1`)
- [ ] Fichiers de config : permissions 640
- [ ] ACL : contrôleurs sensibles restreints
- [ ] CSRF : tokens sur tous les formulaires d'action
- [ ] Permissions : `tmp/` et `data/` seuls répertoires inscriptibles
- [ ] Monitoring : log des accès API
- [ ] Monitoring : alertes sur échecs d'authentification
- [ ] Réseau : firewall en place

- [ ] Réseau : pas d'exposition publique
- [ ] SQL : requêtes paramétrées dans Tag, Client, Environment, Backup

Conclusion

Sécuriser PmaControl n'est pas un luxe — c'est une obligation. L'outil a accès à vos serveurs MariaDB / MySQL de production, stocke des credentials, et peut exécuter des commandes via SSH.

Le durcissement se fait en couches : chaque couche (Apache, PHP, MariaDB, secrets, ACL, CSRF, permissions, réseau) ajoute une barrière. Si une couche est percée, les autres ralentissent l'attaquant.

La bonne nouvelle : toutes ces mesures sont standards et applicables en une journée de travail. Le coût est négligeable comparé au risque d'une compromission de votre infrastructure de bases de données.