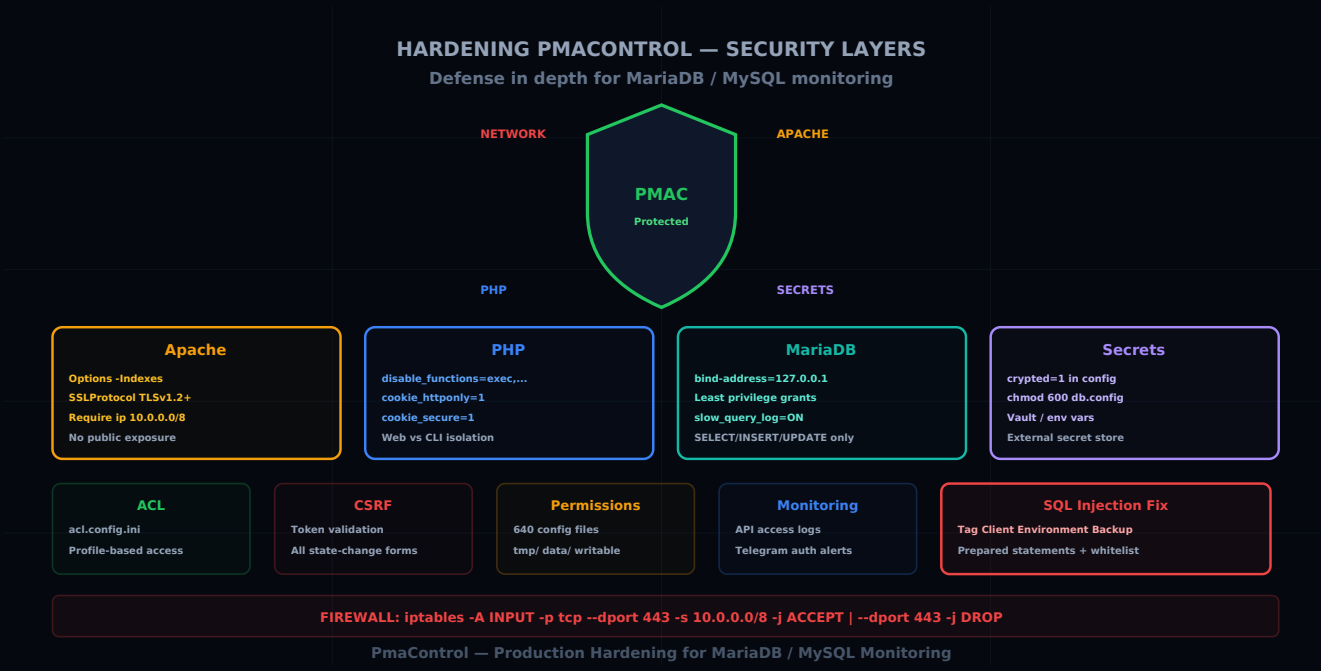
Wzmacnianie PmaControl na produkcji: kompletny przewodnik bezpieczeństwa

Aurélien LEQUOY · April 13, 2026

PMACONTROL SECURITY HARDENING APACHE PHP MARIADB



PmaControl ma klucze do królestwa

PmaControl nadzoruje Twoje produkcyjne serwery MariaDB / MySQL. Przechowuje dane uwierzytelniające, klucze SSH, metryki wydajności, struktury baz danych. Jeśli atakujący skompromituje PmaControl, potencjalnie ma dostęp do **całej infrastruktury bazodanowej**.

Ten przewodnik szczegółowo opisuje środki wzmacniania do zastosowania przed wdrożeniem PmaControl na produkcji. Pochodzi z wewnętrznego audytu bezpieczeństwa i obejmuje każdą warstwę: Apache, PHP, MariaDB, sekrety, ACL, CSRF, uprawnienia plików i monitoring.

Warstwa 1: Apache

Wyłączenie listowania katalogów

Domyślnie Apache może wyświetlać zawartość katalogów bez pliku index. To wyciek informacji:

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

`-Indexes` jest bezdyskusyjny. Bez niego atakujący może przeglądać strukturę projektu i znaleźć pliki konfiguracyjne, logi, zrzuty baz.

Wymuszenie HTTPS

PmaControl przesyła dane uwierzytelniające w postaci jawnej w żądaniach HTTP. Bez HTTPS atakujący w sieci może je przechwycić:

```
<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

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

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

Ograniczenie do sieci wewnętrznej

PmaControl **nigdy** nie powinien być wystawiony na Internet. Ogranicz dostęp do sieci wewnętrznej:

```
<Location />
    Require ip 10.0.0.0/8
    Require ip 172.16.0.0/12
```

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

Lub jeszcze lepiej: umieść PmaControl za VPN i w ogóle nie wystawiaj go przez publiczny Apache.

Usunięcie domyślnego vhosta

Domyślny vhost Apache (`000-default.conf`) odpowiada na każde żądanie na IP serwera. Usuń go:

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

Nagłówki bezpieczeństwa

Dodaj nagłówki bezpieczeństwa 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'"
```

Warstwa 2: PHP

Wyłączenie niebezpiecznych funkcji

PmaControl używa `exec()` i `shell_exec()` do niektórych operacji (SSH, zbieranie metryk).

Rozwiązaniem nie jest ich globalne wyłączenie, lecz izolacja workerów, które ich potrzebują.

Dla vhosta webowego (interfejs):

```
; php.ini lub .user.ini w DocumentRoot
disable_functions = exec,shell_exec,system,passthru,popen,proc_open
expose_php = Off
```

Dla workerów CLI (Aspirateur, Listener):

```
; php-cli.ini — te workery potrzebują shell_exec
disable_functions =
```

Ta separacja gwarantuje, że interfejs webowy nie może wykonywać poleceń systemowych, nawet jeśli atakujący znajdzie lukę.

Zabezpieczenie sesji

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

`cookie_httponly` uniemożliwia JavaScriptowi dostęp do ciasteczka sesji (ochrona XSS). `cookie_secure` wymusza wysyłanie tylko przez HTTPS. `cookie_samesite = Strict` chroni przed podstawowym CSRF.

Ograniczenie uploadu i wykonania

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

PmaControl nie potrzebuje masowych uploadów. Ogranicz, by zmniejszyć powierzchnię ataku.

Ukrycie wersji PHP

```
expose_php = Off
```

Usuwa to nagłówek `X-Powered-By: PHP/8.x` z odpowiedzi HTTP.

Warstwa 3: MariaDB

Ograniczenie uprawnień użytkownika PmaControl

Po instalacji użytkownik PmaControl często ma wszystkie uprawnienia. Ogranicz je:

```
-- Cofnięcie nadmiernych uprawnień
REVOKE ALL PRIVILEGES ON *.* FROM 'pmacontrol'@'localhost';
```

```
-- Nadanie tylko niezbędnych
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;
```

Zasada minimalnych uprawnień: PmaControl potrzebuje jedynie odczytywać metryki i zapisywać do własnej bazy.

Nasłuchiwanie na localhost

Baza danych PmaControl powinna nasłuchiwać tylko na interfejsie lokalnym:

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

Jeśli PmaControl i jego baza są na tym samym serwerze (typowa konfiguracja), nie ma powodu, by nasłuchiwać w sieci.

Włączenie logowania wrażliwych zapytań

```
[mysqld]
general_log = OFF          # Zbyt szczegółowy na produkcji
slow_query_log = ON
long_query_time = 1
log_error = /var/log/mysql/error.log
```

Slow query log pozwala wykryć nietypowe zapytania, które mogą wskazywać na wykorzystaną iniekcję SQL.

Warstwa 4: Sekrety

Szyfrowanie danych uwierzytelniających

PmaControl przechowuje dane uwierzytelniające w `db.config.ini.php`. Plik ten obsługuje szyfrowanie:

```
; 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
```

Flaga `crypted=1` informuje PmaControl, by odszyfrował hasło w runtime. Klucz szyfrowania jest oddzielony od pliku konfiguracyjnego.

Użycie zewnętrznego magazynu sekretów

Dla krytycznych wdrożeń produkcyjnych eksternalizuj sekrety:

- **Vault** (HashiCorp): PmaControl może odczytywać sekrety przez API
- **AWS Secrets Manager** lub **GCP Secret Manager**: dla wdrożeń chmurowych
- **Zmienne środowiskowe**: minimum viable, lepsze niż tekst jawny

```
# Przykład ze zmiennymi środowiskowymi
export PMAC_DB_PASSWORD="secret_value"
export PMAC_SSH_PASSPHRASE="ssh_secret"
```

Ochrona plików konfiguracyjnych

```
# Właściciel: www-data (użytkownik Apache)
chown root:www-data /srv/www/pmacontrol/configuration/*.php

# Uprawnienia: odczyt dla grupy, nic dla pozostałych
chmod 640 /srv/www/pmacontrol/configuration/*.php

# Plik z danymi uwierzytelniającymi powinien być czytelny tylko przez www-data
chmod 600 /srv/www/pmacontrol/configuration/db.config.ini.php
```

Warstwa 5: ACL (Access Control Lists)

Przegląd acl.config.ini

PmaControl ma system ACL oparty na profilach. Plik `acl.config.ini` definiuje, który profil ma dostęp do którego kontrolera.

```
; 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
```

Podstawowe zasady:

- **Ograniczyć wrażliwe kontrolery**: `Config`, `Backup`, `Install`, `Api` powinny być dostępne tylko dla adminów
- **Utworzyć profil read-only**: dla deweloperów, którzy potrzebują przeglądać bez modyfikowania
- **Regularnie audytować**: sprawdzać, czy nowo dodane kontrolery są objęte przez ACL

Ochrona krytycznych endpointów

Niektóre endpointy są szczególnie wrażliwe:

```
[admin]
Install = allow      ; Instalacja / reinstalacja
Config = allow       ; Modyfikacja konfiguracji
Api = allow          ; Pełne REST API
Backup = allow       ; Dostęp do kopii zapasowych (zawiera dane)

[dba]
Install = deny       ; NIGDY niedostępne dla nie-adminów
```

```
Config = deny
Api = allow(read)    ; Tylko odczyt przez API
Backup = deny
```

Warstwa 6: CSRF (Cross-Site Request Forgery)

Sprawdzenie obecności tokenów

Każdy formularz PmaControl musi zawierać 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>
```

Po stronie serwera kontroler musi walidować token:

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

Akcje do ochrony priorytetowej

Akcje modyfikujące stan są najbardziej krytyczne:

- START/STOP SLAVE
- SKIP error
- Dodawanie/usuwanie serwera
- Modyfikacja konfiguracji
- Tworzenie/usuwanie użytkownika

Bez ochrony CSRF atakujący mógłby zmusić zalogowanego DBA do zatrzymania replikacji serwera produkcyjnego, wysyłając mu podstępny link.

Warstwa 7: Uprawnienia plików

Drzewo uprawnień

```
# Katalog główny: czytelny, nie modyfikowalny
chown -R root:www-data /srv/www/pmacontrol/
chmod -R 750 /srv/www/pmacontrol/

# Katalogi zapisu: www-data jako właściciel
chown -R www-data:www-data /srv/www/pmacontrol/tmp/
chown -R www-data:www-data /srv/www/pmacontrol/data/

# Pliki PHP: tylko do odczytu dla www-data
find /srv/www/pmacontrol/App/ -name "*.php" -exec chmod 640 {} \;

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

Zasada: `www-data` może czytać kod i zapisywać do `tmp/` oraz `data/`. Nie może modyfikować kodu źródłowego ani konfiguracji.

Warstwa 8: Monitoring bezpieczeństwa

Logowanie dostępów API

Każde wywołanie REST API powinno być logowane z:

- Znacznik czasu
- IP źródłowe
- Użytkownik (token)
- Wywołany endpoint
- Kod odpowiedzi

```
// W 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);
```

Alerty Telegram na nieudane uwierzytelnienia

Skonfiguruj alert Telegram dla każdego nieudanego logowania:

```
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')
    );
}
```

Trzy nieudane próby z tego samego IP w 5 minut powinny wyzwolić tymczasową blokadę.

Monitorowanie plików konfiguracyjnych

Użyj `inotifywait` lub podobnego narzędzia do wykrywania nieautoryzowanych modyfikacji:

```
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
    # Wyślij alert Telegram
done
```

Warstwa 9: Sieć

Reguły firewalla

```
# Zezwolenie na HTTP/HTTPS tylko z sieci wewnętrznej
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

# Zezwolenie na MySQL tylko z localhost
iptables -A INPUT -p tcp --dport 3306 -s 127.0.0.1 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 3306 -j DROP
```

Brak ekspozycji publicznej

PmaControl **nigdy** nie powinien być dostępny z Internetu. Nawet z uwierzytelnianiem powierzchnia ataku jest zbyt duża:

- Przechowywane są dane uwierzytelniające nadzorowanych serwerów
- Przechowywane są klucze SSH
- Interfejs pozwala wykonywać akcje na serwerach produkcyjnych

Jeśli potrzebny jest zdalny dostęp, użyj VPN (WireGuard, OpenVPN) lub tunelu SSH.

Warstwa 10: Iniekcje SQL — naprawa

Nasz wewnętrzny audyt zidentyfikował ryzyka iniekcji SQL w czterech kontrolerach:

Kontroler	Ryzyko	Naprawa
`Tag.php`	Dynamiczna budowa klauzuli WHERE	Zapytania parametryzowane
`Client.php`	Konkatenacja w filtrach	Zapytania parametryzowane
`Environment.php`	Interpolacja zmiennej w ORDER BY	Whitelist kolumn
`Backup.php`	Nieescape'owany parametr w LIKE	Zapytania parametryzowane

Przed (podatne):

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

Po (bezpieczne):

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

Dla klauzul ORDER BY whitelist jest jedynym bezpiecznym rozwiązaniem:

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

Lista kontrolna wzmacniania

Przed przeniesieniem PmaControl na produkcję zwaliduj każdy punkt:

- [ ] Apache: `-Indexes` włączony
- [ ] Apache: HTTPS wymuszone
- [ ] Apache: dostęp ograniczony do sieci wewnętrznej
- [ ] Apache: domyślny vhost usunięty
- [ ] PHP: niebezpieczne funkcje wyłączone na vhoście web
- [ ] PHP: `session.cookie_httponly = 1`
- [ ] PHP: `session.cookie_secure = 1`
- [ ] PHP: `expose_php = Off`
- [ ] MariaDB: użytkownik z minimalnymi uprawnieniami
- [ ] MariaDB: `bind-address = 127.0.0.1`
- [ ] Sekrety: dane uwierzytelniające zaszyfrowane (`crypted=1`)
- [ ] Pliki konfiguracyjne: uprawnienia 640
- [ ] ACL: wrażliwe kontrolery ograniczone
- [ ] CSRF: tokeny na wszystkich formularzach akcji
- [ ] Uprawnienia: `tmp/` i `data/` jedyne katalogi zapisywalne
- [ ] Monitoring: logowanie dostępów API
- [ ] Monitoring: alerty na nieudane uwierzytelnienia
- [ ] Sieć: firewall aktywny
- [ ] Sieć: brak ekspozycji publicznej
- [ ] SQL: zapytania parametryzowane w Tag, Client, Environment, Backup

Podsumowanie

Zabezpieczenie PmaControl to nie luksus — to obowiązek. Narzędzie ma dostęp do produkcyjnych serwerów MariaDB / MySQL, przechowuje dane uwierzytelniające i może wykonywać polecenia przez SSH.

Wzmacnianie odbywa się warstwowo: każda warstwa (Apache, PHP, MariaDB, sekrety, ACL, CSRF, uprawnienia, sieć) dodaje barierę. Jeśli jedna warstwa zostanie przebita, pozostałe spowalniają atakującego.

Dobra wiadomość: wszystkie te środki są standardowe i możliwe do wdrożenia w jeden dzień pracy. Koszt jest znikomy w porównaniu z ryzykiem kompromitacji infrastruktury bazodanowej.