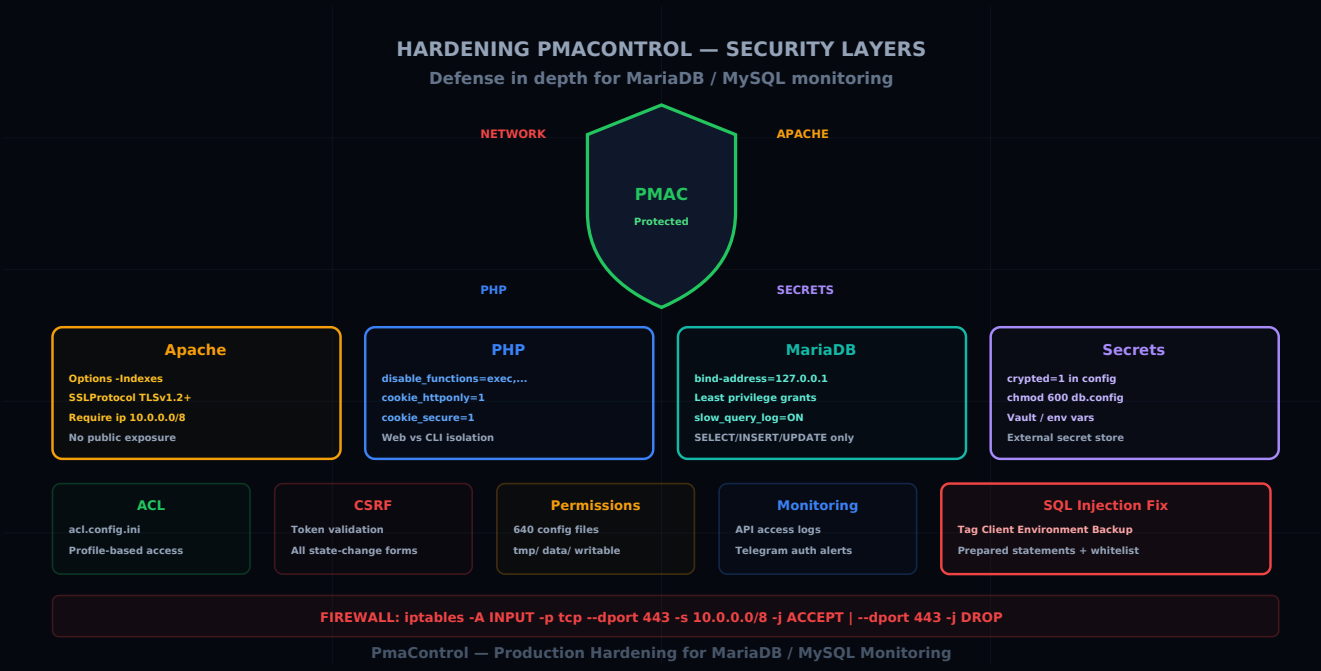
# Укрепление PmaControl в production: полное руководство по безопасности

**Aurélien LEQUOY** · April 13, 2026

PMACONTROL SECURITY HARDENING APACHE PHP MARIADB




## PmaControl хранит ключи от королевства

PmaControl осуществляет мониторинг ваших production-серверов MariaDB / MySQL. Он хранит учётные данные подключения, SSH-ключи, метрики производительности, структуру баз данных. Если злоумышленник скомпрометирует PmaControl, он потенциально получит доступ ко **всей вашей инфраструктуре баз данных**.

Это руководство описывает меры усиления защиты, которые необходимо применить перед вводом PmaControl в production. Оно основано на результатах внутреннего аудита безопасности и охватывает каждый уровень: Apache, PHP, MariaDB, секреты, ACL, CSRF, права на файлы и мониторинг.

## Уровень 1: Apache

### Отключить листинг каталогов

По умолчанию Apache может отображать содержимое каталогов без файла index. Это утечка информации:

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

`-Indexes` — безоговорочное требование. Без него злоумышленник может просматривать структуру проекта и находить конфигурационные файлы, логи, дампы.

## Принудительный HTTPS

PmaControl передаёт учётные данные в открытом виде в HTTP-запросах. Без HTTPS злоумышленник в сети может их перехватить:

```
<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

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

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

## Ограничить доступ внутренней сетью

PmaControl **никогда** не должен быть доступен из Интернета. Ограничьте доступ внутренней сетью:

```
<Location />
    Require ip 10.0.0.0/8
    Require ip 172.16.0.0/12
    Require ip 192.168.0.0/16
</Location>
```

Или ещё лучше: разместите PmaControl за VPN и не экспонируйте его через публичный Apache вовсе.

### Удалить vhost по умолчанию

Vhost по умолчанию Apache ( `000-default.conf` ) отвечает на любой запрос по IP-адресу сервера. Удалите его:

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

### Заголовки безопасности

Добавьте HTTP-заголовки безопасности:

```
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'"
```

## Уровень 2: PHP

### Отключить опасные функции

PmaControl использует `exec()` и `shell_exec()` для некоторых операций (SSH, сбор данных). Решение — не отключать их глобально, а изолировать воркеры, которым они нужны.

Для веб-vhost (интерфейс):

```
; php.ini или .user.ini в DocumentRoot
disable_functions = exec,shell_exec,system,passthru,popen,proc_open
```

```
expose_php = Off
```

Для CLI-воркеров (Aspirateur, Listener):

```
; php-cli.ini — этим воркерам нужен shell_exec
disable_functions =
```

Это разделение гарантирует, что веб-интерфейс не может выполнять системные команды, даже если злоумышленник найдёт уязвимость.

### Защитить сессии

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

`cookie_httponly` запрещает JavaScript доступ к cookie сессии (защита от XSS). `cookie_secure` принуждает отправку только по HTTPS. `cookie_samesite = Strict` защищает от базового CSRF.

### Ограничить загрузку и выполнение

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

PmaControl не нуждается в массивных загрузках. Ограничьте для уменьшения поверхности атаки.

### Скрыть версию PHP

```
expose_php = Off
```

Это убирает заголовок `X-Powered-By: PHP/8.x` из HTTP-ответов.

## Уровень 3: MariaDB

### Ограничить привилегии пользователя PmaControl

После установки пользователь PmaControl часто имеет все привилегии. Ограничьте их:

```
-- Отозвать избыточные привилегии
REVOKE ALL PRIVILEGES ON *.* FROM 'pmacontrol'@'localhost';

-- Выдать только необходимое
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;
```

Принцип минимальных привилегий: PmaControl нуждается только в чтении метрик и записи в собственную базу.

### Привязать к localhost

База данных PmaControl должна слушать только на локальном интерфейсе:

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

Если PmaControl и его база находятся на одном сервере (типичная конфигурация), нет причин слушать на сети.

### Включить лог чувствительных запросов

```
[mysqld]
general_log = OFF          # Слишком объёмный в production
slow_query_log = ON
long_query_time = 1
log_error = /var/log/mysql/error.log
```

Slow query log позволяет обнаруживать аномальные запросы, которые могут указывать на эксплуатируемую SQL-инъекцию.

## Уровень 4: Секреты

### Шифровать учётные данные

PmaControl хранит учётные данные подключения в `db.config.ini.php`. Этот файл поддерживает шифрование:

```
; 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
```

Флаг `crypted=1` указывает PmaControl расшифровать пароль во время выполнения. Ключ шифрования отделён от файла конфигурации.

### Использовать внешнее хранилище секретов

Для критически важных production-развёртываний выносите секреты наружу:

- **Vault** (HashiCorp): PmaControl может читать секреты через API
- **AWS Secrets Manager** или **GCP Secret Manager**: для облачных развёртываний
- **Переменные окружения**: минимально жизнеспособный вариант, лучше открытого текста

```
# Пример с переменными окружения
export PMAC_DB_PASSWORD="secret_value"
export PMAC_SSH_PASSPHRASE="ssh_secret"
```

### Защитить файлы конфигурации

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

# Права: чтение для группы, ничего для остальных
```

```
chmod 640 /srv/www/pmacontrol/configuration/*.php

# Файл учётных данных должен быть доступен только www-data
chmod 600 /srv/www/pmacontrol/configuration/db.config.ini.php
```

## Уровень 5: ACL (Access Control Lists)

### Проверить acl.config.ini

PmaControl имеет систему ACL на основе профилей. Файл `acl.config.ini` определяет, какой профиль может обращаться к какому контроллеру.

```
; 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
```

Ключевые правила:

- **Ограничить чувствительные контроллеры**: `Config`, `Backup`, `Install`, `Api` должны быть доступны только администраторам
- **Создать профиль read-only**: для разработчиков, которым нужно просматривать без изменения
- **Регулярно аудировать**: проверять, что вновь добавленные контроллеры покрыты ACL

### Защитить критические эндпоинты

Некоторые эндпоинты особенно чувствительны:

```
[admin]
Install = allow      ; Установка / переустановка
Config = allow       ; Изменение конфигурации
Api = allow          ; Полный REST API
Backup = allow       ; Доступ к бэкапам (содержит данные)

[dba]
Install = deny       ; НИКОГДА не доступен не-администраторам
Config = deny
Api = allow(read)    ; Только чтение через API
Backup = deny
```

## Уровень 6: CSRF (Cross-Site Request Forgery)

### Проверить наличие токенов

Каждая форма PmaControl должна включать CSRF-токен:

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

На стороне сервера контроллер должен валидировать токен:

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

### Приоритетные действия для защиты

Действия, изменяющие состояние, наиболее критичны:

- START/STOP SLAVE
- SKIP error
- Добавление/удаление сервера
- Изменение конфигурации
- Создание/удаление пользователя

Без защиты от CSRF злоумышленник мог бы заставить авторизованного DBA остановить репликацию production-сервера, отправив ему поддельную ссылку.

## Уровень 7: Права файловой системы

### Дерево прав доступа

```
# Основной каталог: читаемый, не модифицируемый
chown -R root:www-data /srv/www/pmacontrol/
chmod -R 750 /srv/www/pmacontrol/

# Каталоги для записи: www-data владелец
chown -R www-data:www-data /srv/www/pmacontrol/tmp/
chown -R www-data:www-data /srv/www/pmacontrol/data/

# PHP-файлы: только чтение для www-data
find /srv/www/pmacontrol/App/ -name "*.php" -exec chmod 640 {} \;

# Конфигурация: ограниченный доступ
chmod 640 /srv/www/pmacontrol/configuration/*.php
chmod 600 /srv/www/pmacontrol/configuration/db.config.ini.php
```

Принцип: `www-data` может читать код и писать в `tmp/` и `data/`. Он не может изменять исходный код или конфигурацию.

## Уровень 8: Мониторинг безопасности

### Логировать обращения к API

Каждый вызов REST API должен логироваться с:

- Меткой времени
- Исходным IP
- Пользователем (токен)
- Вызванным эндпоинтом
- Кодом ответа

```
// В 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);
```

### Оповещения Telegram при неудачных аутентификациях

Настройте Telegram-оповещение для каждой неудачной попытки входа:

```
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')
    );
}
```

Три неудачных попытки с одного IP за 5 минут должны вызвать временную блокировку.

### Отслеживать файлы конфигурации

Используйте `inotifywait` или подобный инструмент для обнаружения несанкционированных изменений:

```
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
    # Отправить оповещение в Telegram
done
```

## Уровень 9: Сеть

### Правила межсетевого экрана

```
# Разрешить HTTP/HTTPS только из внутренней сети
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

# Разрешить MySQL только на localhost
iptables -A INPUT -p tcp --dport 3306 -s 127.0.0.1 -j ACCEPT
iptables -A INPUT -p tcp --dport 3306 -j DROP
```

### Никакого публичного доступа

PmaControl **никогда** не должен быть доступен из Интернета. Даже с аутентификацией поверхность атаки слишком велика:

- Хранятся учётные данные наблюдаемых серверов
- Хранятся SSH-ключи
- Интерфейс позволяет выполнять действия на production-серверах

Если необходим удалённый доступ, используйте VPN (WireGuard, OpenVPN) или SSH-туннель.

## Уровень 10: SQL-инъекции — устранение

Наш внутренний аудит выявил риски SQL-инъекций в четырёх контроллерах:

| Контроллер | Риск | Устранение |
|---|---|---|
| `Tag.php` | Динамическое построение WHERE | Параметризованные запросы |
| `Client.php` | Конкатенация в фильтрах | Параметризованные запросы |
| `Environment.php` | Интерполяция переменной в ORDER BY | Белый список столбцов |
| `Backup.php` | Неэкранированный параметр в LIKE | Параметризованные запросы |

### До (уязвимо):

```
// Tag.php — УЯЗВИМО
$sql = "SELECT * FROM tags WHERE name LIKE '%" . $_GET['search'] . "%'";
```

```
$results = $db->query($sql);
```

### После (безопасно):

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

Для предложений ORDER BY белый список — единственное безопасное решение:

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

## Чек-лист укрепления

Перед вводом PmaControl в production проверьте каждый пункт:

- [ ] Apache: `-Indexes` активирован
- [ ] Apache: HTTPS принудительный
- [ ] Apache: доступ ограничен внутренней сетью
- [ ] Apache: vhost по умолчанию удалён
- [ ] PHP: опасные функции отключены для веб-vhost
- [ ] PHP: `session.cookie_httponly = 1`
- [ ] PHP: `session.cookie_secure = 1`
- [ ] PHP: `expose_php = Off`
- [ ] MariaDB: пользователь с минимальными привилегиями
- [ ] MariaDB: `bind-address = 127.0.0.1`
- [ ] Секреты: учётные данные зашифрованы ( `crypted=1` )
- [ ] Файлы конфигурации: права 640
- [ ] ACL: чувствительные контроллеры ограничены
- [ ] CSRF: токены на всех формах действий
- [ ] Права: `tmp/` и `data/` — единственные записываемые каталоги

- [ ] Мониторинг: логирование обращений к API
- [ ] Мониторинг: оповещения при неудачных аутентификациях
- [ ] Сеть: межсетевой экран настроен
- [ ] Сеть: нет публичного доступа
- [ ] SQL: параметризованные запросы в Tag, Client, Environment, Backup

## Заключение

Укрепление PmaControl — это не роскошь, а обязанность. Инструмент имеет доступ к вашим production-серверам MariaDB / MySQL, хранит учётные данные и может выполнять команды через SSH.

Укрепление проводится послойно: каждый слой (Apache, PHP, MariaDB, секреты, ACL, CSRF, права, сеть) добавляет барьер. Если один слой преодолён, остальные замедляют злоумышленника.

Хорошая новость: все эти меры стандартны и могут быть применены за один рабочий день. Затраты ничтожны по сравнению с риском компрометации вашей инфраструктуры баз данных.