Комплексный аудит безопасности и качества кода BaiMuras Platform

Дата аудита: 19 июня 2025
Репозиторий: ardakchapaev/baimuras.space
Общий объем кода: 932 строки (426 statements)

КРИТИЧЕСКИЕ ПРОБЛЕМЫ БЕЗОПАСНОСТИ

1. Flask Debug Mode в продакшене (ВЫСОКИЙ РИСК)

Файл: `src/main.py:98`
Проблема: Приложение запускается с `debug=True`, что открывает Werkzeug debugger
Риск: Позволяет выполнение произвольного кода злоумышленниками
CWE: CWE-94 (Code Injection)

```
app.run(host="0.0.0.0", port=5000, debug=True)  # КРИТИЧНО!
```

2. Привязка ко всем интерфейсам (СРЕДНИЙ РИСК)

Файл: `src/main.py:98`
Проблема: Приложение привязано к `0.0.0.0`, доступно извне
Риск: Незащищенный доступ к приложению из внешних сетей
CWE: CWE-605 (Multiple Binds to the Same Port)

УЯЗВИМОСТИ ЗАВИСИМОСТЕЙ

Flask-CORS 4.0.0 (5 уязвимостей)

1. **CVE-2024-6221** - Access-Control-Allow-Private-Network по умолчанию true
2. **CVE-2024-1681** - Log injection при debug уровне
3. **CVE-2024-6844** - Некорректная обработка '+' в URL путях
4. **CVE-2024-6866** - Case-insensitive path matching
5. **CVE-2024-6839** - Неправильная приоритизация regex паттернов

Решение: Обновить до версии 6.0.0+

Gunicorn 21.2.0 (2 уязвимости)

1. **CVE-2024-1135** - HTTP Request Smuggling через Transfer-Encoding
2. **CVE-2024-6827** - TE.CL request smuggling

Решение: Обновить до версии 23.0.0+

Werkzeug 2.3.7 (4 уязвимости)

1. **CVE-2023-46136** - DoS через crafted multipart data
2. **CVE-2024-34069** - Выполнение кода через debugger
3. **CVE-2024-49766** - Небезопасные UNC пути на Windows
4. **CVE-2024-49767** - Resource exhaustion в multipart/form-data

Решение: Обновить до версии 3.0.6+

КАЧЕСТВО КОДА (Pylint: 8.12/10)

Основные проблемы:

- **60 нарушений конвенций** (trailing whitespace, длинные строки)
- **17 предупреждений** (неиспользуемые переменные, широкие исключения)
- **3 рефакторинга** (упрощение кода)

Детализация по модулям:

1. **src/routes/api.py** - 7 широких исключений `except Exception`
2. **src/content.py** - 21 строка превышает 100 символов
3. **src/main.py** - неправильное расположение импортов
4. **src/config.py** - множественные trailing whitespace

ПЛАН ИСПРАВЛЕНИЯ (Приоритизированный)

НЕМЕДЛЕННО (Критично)

1. **Отключить debug режим в продакшене**
   ```
   python
   # Заменить в src/main.py:98
   app.run(host="127.0.0.1", port=5000, debug=False)
   ```

2. **Обновить уязвимые зависимости**
   ```
   bash
   pip install flask-cors>=6.0.0 gunicorn>=23.0.0 werkzeug>=3.0.6
   ```

В ТЕЧЕНИЕ НЕДЕЛИ (Высокий приоритет)

1. **Исправить обработку исключений**
   - Заменить `except Exception` на специфичные исключения
   - Добавить логирование ошибок

2. **Настроить CORS правильно**
   - Указать конкретные origins вместо wildcard
   - Настроить правильные headers

3. **Добавить валидацию входных данных**
   - Проверка CSRF токенов
   - Валидация форм

В ТЕЧЕНИЕ МЕСЯЦА (Средний приоритет)

1. **Улучшить качество кода**
   - Исправить все trailing whitespace
   - Разбить длинные строки
   - Переместить импорты в начало файлов

2. **Добавить тесты безопасности**
   - Unit тесты для валидации
   - Integration тесты для API

3. **Настроить CI/CD**
   - Автоматический запуск Bandit
   - Проверка зависимостей на уязвимости

РЕКОМЕНДАЦИИ ПО БЕЗОПАСНОСТИ

Конфигурация

1. Использовать переменные окружения для секретов
2. Настроить HTTPS в продакшене
3. Добавить rate limiting
4. Настроить логирование безопасности

Код

1. Добавить input sanitization
2. Использовать parameterized queries
3. Добавить authentication/authorization
4. Реализовать proper session management

Инфраструктура

1. Настроить firewall правила
2. Использовать reverse proxy (nginx)
3. Настроить мониторинг безопасности
4. Регулярные backup’ы

МЕТРИКИ КАЧЕСТВА

- **Общий рейтинг Pylint:** 8.12/10 (хорошо, но есть место для улучшения)
- **Критических ошибок:** 0
- **Предупреждений:** 17
- **Дублирования кода:** 0%
- **Покрытие документацией:** 93.75%

ЦЕЛИ НА СЛЕДУЮЩИЙ МЕСЯЦ

1. Достичь Pylint рейтинга 9.5/10
2. Устранить все уязвимости зависимостей
3. Добавить 80%+ покрытие тестами
4. Настроить автоматический security scanning
5. Реализовать proper logging и monitoring

КОНТАКТЫ ДЛЯ ВОПРОСОВ

При возникновении вопросов по данному аудиту обращайтесь к команде безопасности.

Отчет сгенерирован автоматически с использованием Bandit, Pylint, pip-audit и других инструментов анализа безопасности.