Руководство по безопасности BaiMuras Platform

Обзор безопасности

BaiMuras Platform реализует комплексный подход к безопасности, включающий защиту от основных веб-уязвимостей, безопасное управление данными и соответствие современным стандартам безопасности.

Реализованные меры безопасности

Защита от OWASP Top 10

1. Injection (A03:2021)

- **SQL Injection**: Использование SQLAlchemy ORM с параметризованными запросами
- **NoSQL Injection**: Валидация всех входных данных
- **Command Injection**: Санитизация системных команд
- **LDAP Injection**: Экранирование LDAP запросов

```
# Пример безопасного запроса
user = User.query.filter_by(email=email).first()  # Безопасно
# Вместо: db.execute(f"SELECT * FROM users WHERE email = '{email}'")  # Уязвимо
```

2. Broken Authentication (A07:2021)

- **Сильные пароли**: Минимум 8 символов, сложность
- **Защита сессий**: Secure, HttpOnly, SameSite cookies
- **Тайм-ауты**: Автоматическое завершение неактивных сессий
- **Блокировка аккаунтов**: После неудачных попыток входа

```
# Конфигурация сессий
app.config['SESSION_COOKIE_SECURE'] = True
app.config['SESSION_COOKIE_HTTPONLY'] = True
app.config['SESSION_COOKIE_SAMESITE'] = 'Lax'
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(hours=2)
```

3. Sensitive Data Exposure (A02:2021)

- **Шифрование в покое**: Шифрование чувствительных данных в БД
- **Шифрование в транзите**: HTTPS/TLS 1.3
- **Хеширование паролей**: bcrypt с солью
- **Маскирование данных**: В логах и ошибках

```
# Безопасное хеширование паролей
from werkzeug.security import generate_password_hash, check_password_hash

password_hash = generate_password_hash(password, method='pbkdf2:sha256',
salt_length=16)
```

4. XML External Entities (A04:2021)

- **Отключение XXE**: Безопасная конфигурация XML парсеров
- **Валидация XML**: Строгая схема валидации
- **Ограничение размера**: Лимиты на размер XML файлов

5. Broken Access Control (A01:2021)

- **Ролевая модель**: RBAC (Role-Based Access Control)
- **Проверка авторизации**: На каждом эндпоинте
- **Принцип минимальных привилегий**: Только необходимые права
- **Защита ресурсов**: Проверка владельца ресурса

```
# Декоратор для проверки прав доступа
@login_required
@admin_required
def admin_panel():
    # Только для администраторов
    pass
```

6. Security Misconfiguration (A05:2021)

- **Безопасные заголовки**: CSP, HSTS, X-Frame-Options
- **Отключение отладки**: В продакшн среде
- **Удаление дефолтных аккаунтов**: Нет стандартных паролей
- **Обновления**: Регулярное обновление зависимостей

```
# Безопасные HTTP заголовки
@app.after_request
def set_security_headers(response):
    response.headers['X-Content-Type-Options'] = 'nosniff'
    response.headers['X-Frame-Options'] = 'DENY'
    response.headers['X-XSS-Protection'] = '1; mode=block'
    response.headers['Strict-Transport-Security'] = 'max-age=31536000; includeSubDo-
mains'
    return response
```

7. Cross-Site Scripting (A03:2021)

- **Экранирование вывода**: Автоматическое в Jinja2
- **CSP заголовки**: Content Security Policy
- **Валидация входных данных**: Санитизация HTML
- **HttpOnly cookies**: Защита от XSS через JavaScript

```
# Content Security Policy
CSP = "default-src 'self'; script-src 'self' 'unsafe-inline'; style-src 'self' 'unsafe-
inline'"
response.headers['Content-Security-Policy'] = CSP
```

8. Insecure Deserialization (A08:2021)

- **Безопасная сериализация**: JSON вместо pickle
- **Валидация данных**: Проверка десериализованных объектов
- **Подписи**: Цифровые подписи для критичных данных

9. Using Components with Known Vulnerabilities (A06:2021)

- **Сканирование зависимостей**: Safety, Bandit
- **Автоматические обновления**: Dependabot
- **Мониторинг CVE**: Отслеживание уязвимостей
- **Минимальные зависимости**: Только необходимые пакеты

```
# Проверка уязвимостей
safety check
bandit -r src/
```

10. Insufficient Logging & Monitoring (A09:2021)

- **Подробное логирование**: Все события безопасности
- **Мониторинг в реальном времени**: Алерты на подозрительную активность
- **Аудит доступа**: Логирование всех попыток доступа
- **Ротация логов**: Безопасное хранение логов

```
# Логирование событий безопасности
import logging

security_logger = logging.getLogger('security')
security_logger.info(f'Login attempt for user {username} from IP {ip_address}')
```

Аутентификация и авторизация

Система аутентификации

- **Flask-Login**: Управление сессиями пользователей
- **Werkzeug Security**: Безопасное хеширование паролей
- **Remember Me**: Безопасные постоянные токены
- **CSRF Protection**: Flask-WTF токены

Ролевая модель

```
class Role(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64), unique=True)
    permissions = db.Column(db.Integer)

class Permission:
    READ = 1
    WRITE = 2
    DELETE = 4
    ADMIN = 8
```

Проверка прав доступа

```python
def permission_required(permission):
    def decorator(f):
        @wraps(f)
        def decorated_function(*args, **kwargs):
            if not current_user.can(permission):
                abort(403)
            return f(*args, **kwargs)
        return decorated_function
    return decorator
```

Сетевая безопасность

HTTPS/TLS конфигурация

```nginx
# Nginx SSL конфигурация
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;
ssl_prefer_server_ciphers off;
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;

# HSTS заголовок
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
```

Защита от DDoS

- **Rate Limiting**: Ограничение запросов по IP
- **Fail2Ban**: Автоматическая блокировка подозрительных IP
- **CloudFlare**: CDN с защитой от DDoS
- **Nginx Rate Limiting**: Ограничения на уровне веб-сервера

```nginx
# Rate limiting в Nginx
limit_req_zone $binary_remote_addr zone=login:10m rate=5r/m;
limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;

location /login {
    limit_req zone=login burst=5 nodelay;
}

location /api/ {
    limit_req zone=api burst=20 nodelay;
}
```

Безопасность базы данных

Конфигурация PostgreSQL

```sql
-- Создание пользователя с ограниченными правами
CREATE USER baimuras_user WITH PASSWORD 'BaiMuras2025!@#';
GRANT CONNECT ON DATABASE baimuras_db TO baimuras_user;
GRANT USAGE ON SCHEMA public TO baimuras_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO baimuras_user;

-- Настройки безопасности
ALTER SYSTEM SET ssl = on;
ALTER SYSTEM SET log_statement = 'all';
ALTER SYSTEM SET log_min_duration_statement = 1000;
```

Шифрование данных

```python
from cryptography.fernet import Fernet

class EncryptedField(db.TypeDecorator):
    impl = db.Text

    def process_bind_param(self, value, dialect):
        if value is not None:
            return encrypt_data(value)
        return value

    def process_result_value(self, value, dialect):
        if value is not None:
            return decrypt_data(value)
        return value
```

Безопасность файлов

Загрузка файлов

```python
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif', 'pdf', 'doc', 'docx'}
MAX_CONTENT_LENGTH = 16 * 1024 * 1024  # 16MB

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def secure_filename_custom(filename):
    # Дополнительная валидация имени файла
    filename = secure_filename(filename)
    # Проверка на вредоносные расширения
    if any(ext in filename.lower() for ext in ['.php', '.jsp', '.asp']):
        raise ValueError("Недопустимый тип файла")
    return filename
```

Сканирование на вирусы

```
import clamd

def scan_file_for_viruses(file_path):
    cd = clamd.ClamdUnixSocket()
    scan_result = cd.scan(file_path)
    if scan_result[file_path][0] == 'FOUND':
        raise ValueError(f"Вирус обнаружен: {scan_result[file_path][1]}")
```

Мониторинг и аудит

Логирование событий безопасности

```
import logging
from datetime import datetime

# Настройка логгера безопасности
security_logger = logging.getLogger('security')
handler = logging.FileHandler('/var/log/baimuras/security.log')
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
handler.setFormatter(formatter)
security_logger.addHandler(handler)
security_logger.setLevel(logging.INFO)

# Функции логирования
def log_login_attempt(username, ip_address, success=True):
    status = "SUCCESS" if success else "FAILED"
    security_logger.info(f'Login {status}: user={username}, ip={ip_address}')

def log_permission_denied(username, resource, ip_address):
    security_logger.warning(f'Access DENIED: user={username}, resource={resource},
ip={ip_address}')

def log_suspicious_activity(description, ip_address):
    security_logger.error(f'SUSPICIOUS: {description}, ip={ip_address}')
```

Мониторинг в реальном времени

```
# Детектор аномалий
class SecurityMonitor:
    def __init__(self):
        self.failed_attempts = {}
        self.suspicious_ips = set()

    def check_failed_login(self, ip_address):
        if ip_address not in self.failed_attempts:
            self.failed_attempts[ip_address] = 0

        self.failed_attempts[ip_address] += 1

        if self.failed_attempts[ip_address] > 5:
            self.block_ip(ip_address)
            self.send_alert(f"IP {ip_address} заблокирован за множественные неудачные
попытки входа")

    def block_ip(self, ip_address):
        # Добавление IP в черный список
        self.suspicious_ips.add(ip_address)
        # Интеграция с fail2ban или iptables
```

Процедуры реагирования на инциденты

План реагирования

1. **Обнаружение**: Автоматические алерты и мониторинг
2. **Анализ**: Определение типа и масштаба инцидента
3. **Сдерживание**: Изоляция пораженных систем
4. **Устранение**: Удаление угрозы
5. **Восстановление**: Возврат к нормальной работе
6. **Извлечение уроков**: Анализ и улучшение процедур

Контакты для экстренных случаев

- **Администратор безопасности**: admin@baimuras.space
- **Техническая поддержка**: +7 (XXX) XXX-XX-XX
- **Резервный контакт**: backup@baimuras.space

Регулярные процедуры безопасности

Еженедельно

- [ ] Проверка логов безопасности
- [ ] Анализ неудачных попыток входа
- [ ] Проверка обновлений безопасности
- [ ] Тестирование резервных копий

Ежемесячно

- [ ] Сканирование уязвимостей
- [ ] Обновление зависимостей

- [ ] Аудит пользователей и ролей
- [ ] Проверка SSL сертификатов

Ежеквартально

- [ ] Пентестирование
- [ ] Обзор политик безопасности
- [ ] Обучение персонала
- [ ] Аудит кода

Ежегодно

- [ ] Полный аудит безопасности
- [ ] Обновление плана реагирования на инциденты
- [ ] Сертификация соответствия
- [ ] Архивирование старых логов

Инструменты безопасности

Статический анализ кода

```
# Bandit - поиск уязвимостей в Python коде
bandit -r src/ -f json -o bandit_report.json

# Safety - проверка уязвимостей в зависимостях
safety check --json --output safety_report.json

# Semgrep - продвинутый статический анализ
semgrep --config=auto src/
```

Динамическое тестирование

```
# OWASP ZAP - сканирование веб-приложения
zap-baseline.py -t https://baimuras.space

# Nikto - сканер веб-сервера
nikto -h https://baimuras.space

# SQLMap - тестирование SQL инъекций
sqlmap -u "https://baimuras.space/login" --forms --batch
```

Мониторинг сети

```
# Nmap - сканирование портов
nmap -sS -O 95.140.153.181

# Wireshark - анализ трафика
tshark -i eth0 -f "port 443"

# Fail2Ban - мониторинг логов
fail2ban-client status
```

Чек-лист безопасности

Перед развертыванием

- [ ] Все пароли изменены с дефолтных
- [ ] SSL сертификаты настроены
- [ ] Файрвол сконфигурирован
- [ ] Логирование включено
- [ ] Резервное копирование настроено
- [ ] Мониторинг активирован

После развертывания

- [ ] Сканирование уязвимостей выполнено
- [ ] Пентестирование проведено
- [ ] Документация обновлена
- [ ] Команда обучена
- [ ] Процедуры реагирования протестированы

Полезные ресурсы

Стандарты и руководства

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- NIST Cybersecurity Framework (https://www.nist.gov/cyberframework)
- CIS Controls (https://www.cisecurity.org/controls/)
- ISO 27001 (https://www.iso.org/isoiec-27001-information-security.html)

Инструменты

- OWASP ZAP (https://www.zaproxy.org/)
- Bandit (https://bandit.readthedocs.io/)
- Safety (https://pyup.io/safety/)
- Semgrep (https://semgrep.dev/)

Обучение

- OWASP WebGoat (https://owasp.org/www-project-webgoat/)
- PortSwigger Web Security Academy (https://portswigger.net/web-security)
- Cybrary (https://www.cybrary.it/)

Сообщение об уязвимостях

Если вы обнаружили уязвимость в BaiMuras Platform:

1. **НЕ создавайте публичный Issue**
2. Отправьте email на: security@baimuras.space
3. Включите подробное описание уязвимости
4. Предоставьте шаги для воспроизведения
5. Укажите потенциальное влияние

Мы обязуемся:
- Ответить в течение 24 часов

- Предоставить обновления каждые 72 часа
- Исправить критические уязвимости в течение 7 дней
- Указать вас в благодарностях (по желанию)

Безопасность - это непрерывный процесс. Регулярно обновляйте и улучшайте меры защиты.