Операционный план кибербезопасности: Руководство для менеджеров и тимлидов

Руководитель службы информационной безопасности

2026-01-02

Краткое резюме для менеджеров

Данный документ представляет **практическое руководство** по реализации мер кибербезопасности на уровне команд и отделов. На основе аудита безопасности от 30 декабря 2025 года выявлены критические уязвимости, требующие **немедленных операционных действий** от руководителей среднего звена.

> ⚠ **ВНИМАНИЕ**: Обнаружена критическая утечка данных. Ваши команды должны выполнить защитные меры в течение 72 часов.

Ключевые задачи для менеджеров

Приоритет	Задача	Ответственный	Срок выполнения
Критический	Блокировка уязвимых API	DevOps/Backend команды	24 часа
🔴 **Высокий**	Внедрение заголовков безопасности	Frontend команды	48 часов
Средний	Настройка мониторинга	SRE/Ops команды	1 неделя
i Низкий	Обучение сотрудников	Все команды	2 недели

Распределение ответственности

Немедленные действия (0-72 часа):

- **Backend команды**: Закрытие REST API уязвимостей
- **DevOps команды**: Настройка безопасных конфигураций
- **QA команды**: Валидация исправлений

Краткосрочные задачи (1-4 недели):

- **Frontend команды**: Внедрение CSP и SRI
- **Infrastructure команды**: Настройка мониторинга
- **Security команды**: Процедуры реагирования на инциденты

Техническая ситуация и приоритеты

Критические уязвимости по командам

Backend/API команды - КРИТИЧЕСКИЙ ПРИОРИТЕТ

Проблема: Утечка персональных данных через REST API

- **Уязвимость**: /wp-json/wp/v2/users доступен без аутентификации
- **Риск**: Все пользовательские данные доступны публично
- **CVSS Score**: 9.1 (Критический)

Немедленные действия:

```
// СРОЧНО: Добавить в functions.php
add_filter('rest_endpoints', function($endpoints) {
  if (isset($endpoints['/wp/v2/users'])) {
    unset($endpoints['/wp/v2/users']);
  }
  return $endpoints;
});
```

Чек-лист для Backend тимлида:

- ☐ Немедленно заблокировать доступ к /wp-json/wp/v2/users
- ☐ Провести аудит всех REST API endpoints
- ☐ Внедрить аутентификацию для чувствительных endpoints
- ☐ Настроить логирование API запросов
- ☐ Протестировать изменения на staging

Frontend команды - ВЫСОКИЙ ПРИОРИТЕТ

Проблема: Отсутствие базовых заголовков безопасности

- **Уязвимости**: Нет CSP, X-Frame-Options, SRI
- **Риск**: XSS атаки, clickjacking, supply chain атаки

Практические решения:

1. **Content Security Policy (CSP)**

```
<!-- Добавить в <head> -->
<meta http-equiv="Content-Security-Policy"
    content="default-src 'self';
          script-src 'self' 'unsafe-inline' https://cdnjs.cloudflare.com;
          style-src 'self' 'unsafe-inline' https://fonts.googleapis.com;">
```

2. **Subresource Integrity (SRI)**

```
<!-- Для всех внешних ресурсов -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
      integrity="sha384-
vtXRMe3mGCbOeY7l30aIg8H9p3GdeSe4IFlP6G8JMa7o7lXvnz3GFKzPxzJdPfGK"
      crossorigin="anonymous"></script>
```

Чек-лист для Frontend тимлида:

- ☐ Внедрить CSP заголовки
- ☐ Добавить SRI для всех внешних ресурсов
- ☐ Настроить X-Frame-Options: DENY
- ☐ Протестировать совместимость с браузерами
- ☐ Обновить CI/CD pipeline для проверки заголовков

DevOps/Infrastructure команды - ВЫСОКИЙ ПРИОРИТЕТ

Проблема: Небезопасная конфигурация веб-сервера

- **Уязвимости**: Отсутствие HSTS, раскрытие версий ПО
- **Риск**: Man-in-the-middle атаки, информационная разведка

Конфигурации по серверам:

Apache (.htaccess):

```
<IfModule mod_headers.c>
   # Базовые заголовки безопасности
   Header always set X-Content-Type-Options "nosniff"
   Header always set X-Frame-Options "DENY"
   Header always set X-XSS-Protection "1; mode=block"
   Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"

   # Скрытие информации о сервере
   Header always unset Server
   Header always unset X-Powered-By
</IfModule>

# Отключение сигнатуры сервера
ServerTokens Prod
ServerSignature Off
```

Nginx:

```
server {
   # Заголовки безопасности
   add_header X-Content-Type-Options "nosniff" always;
   add_header X-Frame-Options "DENY" always;
   add_header X-XSS-Protection "1; mode=block" always;
   add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;

   # Скрытие версии
   server_tokens off;
   more_clear_headers Server;
}
```

Чек-лист для DevOps тимлида:

- ☐ Применить конфигурации безопасности

- ☐ Настроить принудительное HTTPS
- ☐ Скрыть версии серверного ПО
- ☐ Настроить логирование безопасности
- ☐ Протестировать конфигурации

Инструменты и технологии

Обязательные инструменты для команд

Мониторинг и алертинг

1. Система логирования безопасности

```
# Настройка rsyslog для событий безопасности
echo "local0.*    /var/log/security.log" >> /etc/rsyslog.conf
systemctl restart rsyslog
```

2. Мониторинг файлов конфигурации

```
#!/bin/bash
# security_monitor.sh - Скрипт для мониторинга изменений
inotifywait -m /etc/apache2/ /etc/nginx/ -e modify,create,delete \
  --format '%w%f %e %T' --timefmt '%Y-%m-%d %H:%M:%S' \
  >> /var/log/config_changes.log
```

3. Автоматическая проверка заголовков

```
#!/usr/bin/env python3
# header_check.py - Проверка заголовков безопасности
import requests
import sys

def check_security_headers(url):
    required_headers = {
        'X-Content-Type-Options': 'nosniff',
        'X-Frame-Options': ['DENY', 'SAMEORIGIN'],
        'Strict-Transport-Security': 'max-age=',
        'Content-Security-Policy': 'default-src'
    }

    try:
        response = requests.get(url, timeout=10)
        headers = response.headers

        print(f"Проверка заголовков для {url}:")
        for header, expected in required_headers.items():
            if header in headers:
                print(f"   {header}: {headers[header]}")
            else:
                print(f"× {header}: ОТСУТСТВУЕТ")
```

```
    except Exception as e:
        print(f"Ошибка: {e}")

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Использование: python3 header_check.py <url>")
        sys.exit(1)

    check_security_headers(sys.argv[1])
```

Инструменты разработки

1. Pre-commit хуки для безопасности

```
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/PyCQA/bandit
    rev: 1.7.4
    hooks:
      - id: bandit
        args: ['-r', '.']

  - repo: https://github.com/Yelp/detect-secrets
    rev: v1.4.0
    hooks:
      - id: detect-secrets
        args: ['--baseline', '.secrets.baseline']
```

2. Автоматизация SRI генерации

```
// sri-generator.js - Автоматическая генерация SRI
const crypto = require('crypto');
const fs = require('fs');
const https = require('https');

function generateSRI(url) {
    return new Promise((resolve, reject) => {
        https.get(url, (response) => {
            let data = '';
            response.on('data', chunk => data += chunk);
            response.on('end', () => {
                const hash = crypto.createHash('sha384').update(data).digest('base64');
                resolve(`sha384-${hash}`);
            });
        }).on('error', reject);
    });
}

// Использование в build процессе
```

```javascript
async function updateSRI() {
    const externalResources = [
        'https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js',
        'https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700'
    ];

    for (const url of externalResources) {
        try {
            const sri = await generateSRI(url);
            console.log(`${url}: integrity="${sri}"`);
        } catch (error) {
            console.error(`Ошибка для ${url}:`, error.message);
        }
    }
}

updateSRI();
```

CI/CD интеграция

GitLab CI конфигурация:

```yaml
# .gitlab-ci.yml - Безопасность в CI/CD
stages:
  - security-check
  - build
  - test
  - deploy

security_headers_check:
  stage: security-check
  script:
    - python3 scripts/header_check.py $CI_ENVIRONMENT_URL
  only:
    - main
    - develop

dependency_check:
  stage: security-check
  script:
    - npm audit --audit-level moderate
    - composer audit
  allow_failure: false

sri_validation:
  stage: security-check
  script:
    - node scripts/sri-generator.js
    - git diff --exit-code # Проверка, что SRI актуальны
```

```
  only:
    - main
```

Процессы и процедуры

Процедуры реагирования на инциденты

Уровень 1: Обнаружение (Команды разработки)

Индикаторы инцидента:

- Необычная активность в логах
- Алерты от систем мониторинга
- Сообщения пользователей о подозрительном поведении
- Срабатывание автоматических проверок безопасности

Немедленные действия (первые 15 минут):

1. **Не паниковать** - документировать все действия
2. **Изолировать** - отключить подозрительные сервисы
3. **Уведомить** - сообщить тимлиду и security команде
4. **Сохранить** - сделать снапшоты логов и состояния системы

Шаблон уведомления:

```
ИНЦИДЕНТ БЕЗОПАСНОСТИ
Время: [YYYY-MM-DD HH:MM]
Обнаружил: [Имя]
Система: [Название системы/сервиса]
Описание: [Краткое описание проблемы]
Предпринятые действия: [Что уже сделано]
Статус: [Активный/Локализован/Устранен]
```

Уровень 2: Анализ (Тимлиды)

Процедура анализа (30-60 минут):

1. **Сбор информации**

```
# Скрипт для сбора данных об инциденте
#!/bin/bash
INCIDENT_ID=$(date +%Y%m%d_%H%M%S)
INCIDENT_DIR="/var/log/incidents/$INCIDENT_ID"

mkdir -p $INCIDENT_DIR

# Сбор логов
cp /var/log/apache2/access.log $INCIDENT_DIR/
cp /var/log/apache2/error.log $INCIDENT_DIR/
cp /var/log/security.log $INCIDENT_DIR/

# Состояние системы
```

```
ps aux > $INCIDENT_DIR/processes.txt
netstat -tulpn > $INCIDENT_DIR/network.txt
df -h > $INCIDENT_DIR/disk_usage.txt

# Последние изменения
find /var/www -type f -mtime -1 > $INCIDENT_DIR/recent_changes.txt

echo "Данные инцидента собраны в $INCIDENT_DIR"
```

2. **Классификация инцидента**

- **Уровень 1**: Информационный (логирование, мониторинг)
- **Уровень 2**: Предупреждение (потенциальная угроза)
- **Уровень 3**: Критический (активная атака, утечка данных)

3. **Эскалация решений**

Уровень 1 → Тимлид → Документирование
Уровень 2 → Тимлид + Security → Усиленный мониторинг
Уровень 3 → Немедленная эскалация → Руководство + внешние эксперты

Процедуры обновления безопасности

Еженедельные проверки (каждый понедельник)

Чек-лист для тимлидов:

```
## Еженедельная проверка безопасности

### Системные обновления
- [ ] Проверить доступные обновления безопасности
- [ ] Запланировать установку критических патчей
- [ ] Обновить зависимости проекта (npm, composer, pip)

### Мониторинг
- [ ] Проверить логи безопасности за неделю
- [ ] Проанализировать алерты мониторинга
- [ ] Проверить работоспособность backup'ов

### Конфигурации
- [ ] Валидировать заголовки безопасности
- [ ] Проверить SSL сертификаты (срок действия)
- [ ] Аудит пользовательских доступов

### Команда
- [ ] Обсудить инциденты безопасности на ретро
- [ ] Запланировать обучение по безопасности
- [ ] Обновить документацию процедур
```

Ежемесячные аудиты

Процедура для руководителей отделов:

1. **Технический аудит**

```
#!/bin/bash
# monthly_security_audit.sh
echo "=== Ежемесячный аудит безопасности ==="
echo "Дата: $(date)"

# Проверка пользователей с sudo
echo "Пользователи с sudo правами:"
grep -Po '^sudo.+:\K.*$' /etc/group

# Проверка открытых портов
echo "Открытые порты:"
nmap -sT -O localhost

# Проверка неудачных попыток входа
echo "Неудачные попытки входа за месяц:"
grep "Failed password" /var/log/auth.log | wc -l

# Проверка размера логов
echo "Размер логов безопасности:"
du -sh /var/log/security.log
```

2. **Аудит процессов команды**

- Соблюдение процедур безопасности
- Эффективность реагирования на инциденты
- Качество документирования
- Уровень знаний команды

Мониторинг и контроль

Системы мониторинга для команд

Базовый мониторинг (обязательно для всех)

1. Мониторинг доступности и производительности

```
#!/bin/bash
# basic_monitoring.sh - Базовый мониторинг
WEBSITE="https://example.com"
LOG_FILE="/var/log/monitoring.log"

# Проверка доступности
if curl -s --head $WEBSITE | head -n 1 | grep -q "200 OK"; then
   echo "$(date): $WEBSITE - OK" >> $LOG_FILE
else
   echo "$(date): $WEBSITE - НЕДОСТУПЕН" >> $LOG_FILE
   # Отправка алерта
   echo "Сайт недоступен!" | mail -s "АЛЕРТ: Недоступность сайта"
admin@company.com
fi
```

```
# Проверка заголовков безопасности
HEADERS=$(curl -s -I $WEBSITE)
if echo "$HEADERS" | grep -q "X-Content-Type-Options"; then
    echo "$(date): Заголовки безопасности - OK" >> $LOG_FILE
else
    echo "$(date): Заголовки безопасности - ОТСУТСТВУЮТ" >> $LOG_FILE
fi
```

2. Мониторинг логов безопасности

```
#!/bin/bash
# log_monitor.sh - Мониторинг подозрительной активности
SECURITY_LOG="/var/log/security.log"
ALERT_EMAIL="security@company.com"

# Поиск подозрительных паттернов
SUSPICIOUS_PATTERNS=(
    "union.*select"
    "<script"
    "\.\./"
    "eval\("
    "base64_decode"
)

for pattern in "${SUSPICIOUS_PATTERNS[@]}"; do
    if grep -i "$pattern" $SECURITY_LOG | tail -100 | grep -q "$(date +%Y-%m-%d)"; then
        echo "Обнаружена подозрительная активность: $pattern" | \
        mail -s "АЛЕРТ: Подозрительная активность" $ALERT_EMAIL
    fi
done
```

Продвинутый мониторинг (для DevOps команд)

1. Prometheus + Grafana конфигурация

```
# prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'security-metrics'
    static_configs:
      - targets: ['localhost:9090']
    metrics_path: /metrics
    scrape_interval: 30s

  - job_name: 'web-security'
    static_configs:
      - targets: ['example.com:443']
```

```
  metrics_path: /security-check
  scheme: https
```

2. Custom метрики безопасности

```python
#!/usr/bin/env python3
# security_metrics.py - Кастомные метрики для Prometheus
from prometheus_client import start_http_server, Counter, Gauge
import time
import requests
import re

# Метрики
failed_logins = Counter('failed_logins_total', 'Количество неудачных попыток входа')
security_headers = Gauge('security_headers_present', 'Наличие заголовков безопасности', ['header'])
response_time = Gauge('security_check_response_time', 'Время ответа проверки безопасности')

def check_security_headers(url):
    """Проверка заголовков безопасности"""
    try:
        response = requests.get(url, timeout=10)
        headers = response.headers

        # Проверяем наличие важных заголовков
        security_headers.labels(header='x-content-type-options').set(
            1 if 'X-Content-Type-Options' in headers else 0
        )
        security_headers.labels(header='x-frame-options').set(
            1 if 'X-Frame-Options' in headers else 0
        )
        security_headers.labels(header='strict-transport-security').set(
            1 if 'Strict-Transport-Security' in headers else 0
        )

        return response.elapsed.total_seconds()
    except Exception as e:
        print(f"Ошибка проверки: {e}")
        return 0

def count_failed_logins():
    """Подсчет неудачных попыток входа"""
    try:
        with open('/var/log/auth.log', 'r') as f:
            content = f.read()
            failed_count = len(re.findall(r'Failed password', content))
            failed_logins._value._value = failed_count
    except Exception as e:
```

```
        print(f"Ошибка чтения логов: {e}")

if __name__ == '__main__':
    # Запуск HTTP сервера для метрик
    start_http_server(8000)

    while True:
        # Обновление метрик каждые 60 секунд
        response_time_val = check_security_headers('https://example.com')
        response_time.set(response_time_val)

        count_failed_logins()

        time.sleep(60)
```

Дашборды и отчетность

Еженедельные отчеты для тимлидов

Автоматический генератор отчетов:

```
#!/usr/bin/env python3
# weekly_security_report.py
import datetime
import json
from collections import defaultdict

def generate_weekly_report():
    """Генерация еженедельного отчета по безопасности"""

    # Период отчета
    end_date = datetime.datetime.now()
    start_date = end_date - datetime.timedelta(days=7)

    report = {
        'period': f"{start_date.strftime('%Y-%m-%d')} - {end_date.strftime('%Y-%m-%d')}",
        'summary': {},
        'incidents': [],
        'metrics': {},
        'recommendations': []
    }

    # Анализ логов безопасности
    security_events = analyze_security_logs(start_date, end_date)
    report['summary']['security_events'] = len(security_events)

    # Проверка заголовков безопасности
    headers_status = check_headers_compliance()
    report['metrics']['headers_compliance'] = headers_status
```

```python
    # Рекомендации
    if security_events:
        report['recommendations'].append("Обнаружены события безопасности - требуется анализ")

    if not headers_status['all_present']:
        report['recommendations'].append("Не все заголовки безопасности настроены")

    return report

def analyze_security_logs(start_date, end_date):
    """Анализ логов безопасности за период"""
    events = []
    try:
        with open('/var/log/security.log', 'r') as f:
            for line in f:
                # Простой парсинг - в реальности нужен более сложный
                if 'SECURITY' in line:
                    events.append(line.strip())
    except FileNotFoundError:
        pass

    return events

def check_headers_compliance():
    """Проверка соответствия заголовков безопасности"""
    import requests

    try:
        response = requests.get('https://example.com', timeout=10)
        headers = response.headers

        required_headers = [
            'X-Content-Type-Options',
            'X-Frame-Options',
            'Strict-Transport-Security',
            'Content-Security-Policy'
        ]

        present = [h for h in required_headers if h in headers]

        return {
            'total_required': len(required_headers),
            'present': len(present),
            'missing': [h for h in required_headers if h not in present],
            'all_present': len(present) == len(required_headers)
        }
    except Exception as e:
        return {'error': str(e)}
```

```python
if __name__ == '__main__':
    report = generate_weekly_report()

    # Сохранение отчета
    filename = f"security_report_{datetime.datetime.now().strftime('%Y%m%d')}.json"
    with open(filename, 'w', encoding='utf-8') as f:
        json.dump(report, f, indent=2, ensure_ascii=False)

    print(f"Отчет сохранен: {filename}")

    # Отправка по email (опционально)
    # send_report_email(report)
```

Командная работа и делегирование

Матрица ответственности (RACI)

Задача	Backend	Frontend	DevOps	QA	Security
Блокировка API уязвимостей	R	I	C	A	C
Внедрение CSP/SRI	I	R	C	A	C
Настройка заголовков сервера	I	I	R	A	C
Мониторинг безопасности	I	I	R	I	A
Обучение команды	C	C	C	C	R
Реагирование на инциденты	C	C	C	C	R

Легенда:

- **R** (Responsible) - Ответственный за выполнение
- **A** (Accountable) - Подотчетный за результат
- **C** (Consulted) - Консультируемый
- **I** (Informed) - Информируемый

Планы действий по командам

Backend команда (Приоритет 1)

Тимлид Backend: [Имя] **Срок выполнения**: 24-48 часов

Задачи:

1. **Немедленно (0-4 часа)**
   - Заблокировать доступ к /wp-json/wp/v2/users
   - Провести аудит всех REST API endpoints
   - Создать список всех публичных API
2. **Краткосрочно (1-2 дня)**
   - Внедрить аутентификацию для чувствительных endpoints
   - Настроить rate limiting

- Добавить логирование API запросов

3. **Среднесрочно (1 неделя)**
   - Провести полный security review API
   - Внедрить input validation
   - Настроить автоматические тесты безопасности

Ресурсы:

- 2 senior разработчика
- 1 junior для тестирования
- Консультации с Security командой

Критерии готовности:

- ☐ Все уязвимые endpoints заблокированы
- ☐ Аутентификация работает корректно
- ☐ Логирование настроено
- ☐ Тесты проходят успешно

Frontend команда (Приоритет 2)

Тимлид Frontend: [Имя] **Срок выполнения**: 48-72 часа

Задачи:

1. **Немедленно (0-8 часов)**
   - Внедрить базовые заголовки безопасности
   - Добавить CSP в режиме report-only
   - Протестировать на staging
2. **Краткосрочно (1-3 дня)**
   - Добавить SRI для всех внешних ресурсов
   - Перевести CSP в enforcement режим
   - Обновить build процесс
3. **Среднесрочно (1 неделя)**
   - Автоматизировать генерацию SRI
   - Настроить мониторинг CSP нарушений
   - Обучить команду best practices

Ресурсы:

- 2 frontend разработчика
- 1 DevOps для настройки CI/CD
- Поддержка QA команды

Критерии готовности:

- ☐ CSP настроен и работает
- ☐ SRI добавлен для всех ресурсов
- ☐ Нет нарушений в браузерах

- ☐ CI/CD проверяет заголовки

DevOps команда (Приоритет 2)

Тимлид DevOps: [Имя] **Срок выполнения**: 72 часа

Задачи:

1. **Немедленно (0-4 часа)**
   - Применить конфигурации безопасности к веб-серверам
   - Скрыть версии серверного ПО
   - Настроить принудительное HTTPS
2. **Краткосрочно (1-3 дня)**
   - Настроить централизованное логирование
   - Внедрить мониторинг безопасности
   - Автоматизировать проверки конфигураций
3. **Среднесрочно (1 неделя)**
   - Настроить алертинг по событиям безопасности
   - Внедрить Infrastructure as Code
   - Провести аудит всех серверов

Ресурсы:

- 2 DevOps инженера
- 1 SRE для мониторинга
- Доступ к production серверам

Критерии готовности:

- ☐ Заголовки безопасности настроены
- ☐ HTTPS принудительно включен
- ☐ Мониторинг работает
- ☐ Алерты настроены

Коммуникационный план

Ежедневные стендапы (во время кризиса)

Время: 9:00 каждый день **Участники**: Все тимлиды + Security lead **Формат**: 15 минут максимум

Структура:

1. **Статус выполнения критических задач** (5 мин)
2. **Новые проблемы и блокеры** (5 мин)
3. **Планы на день** (3 мин)
4. **Вопросы и координация** (2 мин)

Еженедельные ретроспективы

Время: Пятница, 16:00 **Участники**: Расширенная команда **Длительность**: 1 час

Повестка:

1. **Что прошло хорошо** (15 мин)
2. **Что можно улучшить** (20 мин)
3. **Уроки безопасности** (15 мин)
4. **Планы на следующую неделю** (10 мин)

Каналы коммуникации

Slack каналы:

- #security-incident - Экстренные уведомления
- #security-general - Общие вопросы безопасности
- #security-updates - Обновления и патчи

Email списки:

- security-team@company.com - Команда безопасности
- team-leads@company.com - Все тимлиды
- security-alerts@company.com - Критические алерты

Заключение и следующие шаги

Критические действия на ближайшие 72 часа

День 1 (0-24 часа):

- Backend: Блокировка уязвимых API endpoints
- DevOps: Применение базовых заголовков безопасности
- Все команды: Настройка экстренной коммуникации

День 2 (24-48 часов):

- Frontend: Внедрение CSP и базовых заголовков
- Backend: Полный аудит API безопасности
- QA: Валидация всех изменений

День 3 (48-72 часа):

- DevOps: Настройка мониторинга и алертинга
- Frontend: Добавление SRI для внешних ресурсов
- Все команды: Документирование изменений

Среднесрочные цели (1-4 недели)

Неделя 1:

- Полное внедрение всех базовых мер безопасности
- Настройка автоматизированного мониторинга
- Обучение команд процедурам безопасности

Неделя 2-3:

- Внедрение продвинутых инструментов мониторинга
- Автоматизация проверок безопасности в CI/CD
- Проведение внутреннего пентеста

Неделя 4:

- Полный аудит реализованных мер
- Документирование всех процедур
- Планирование долгосрочной стратегии

Метрики успеха

Технические метрики:

- Время обнаружения угроз: < 1 час
- Время реагирования на инциденты: < 4 часа
- Покрытие заголовками безопасности: 100%
- Количество уязвимостей: снижение на 90%

Процессные метрики:

- Соблюдение процедур безопасности: > 95%
- Время выполнения security задач: в рамках SLA
- Качество документирования: полное покрытие
- Уровень знаний команды: регулярное тестирование

Ресурсы и поддержка

Внутренние ресурсы:

- Security команда: консультации и поддержка
- DevOps команда: инфраструктурная поддержка
- QA команда: тестирование и валидация

Внешние ресурсы:

- Консультанты по безопасности (при необходимости)
- Специализированные инструменты мониторинга
- Обучающие материалы и курсы

Контакты для экстренных ситуаций:

- Security Lead: [контакт]
- DevOps Lead: [контакт]
- Дежурный администратор: [контакт]

Этот документ является живым руководством и должен обновляться по мере изменения угроз и развития наших процессов безопасности. Все тимлиды несут ответственность за его актуализацию в своих областях.