Отчет о соответствии стандартам безопасности: Технический анализ сайта example.com

Ведущий специалист по информационной безопасности

2025-12-31

Аннотация

Настоящий технический отчет представляет детальный анализ соответствия веб-ресурса https://example.com требованиям российских и международных стандартов информационной безопасности. Анализ выполнен на основе автоматизированного сканирования безопасности с использованием OWASP ZAP 2.17.0 от Checkmarx, проведенного 30 декабря 2025 года.

> ⚠ **КРИТИЧЕСКОЕ ПРЕДУПРЕЖДЕНИЕ**: Обнаружена утечка персональных данных (PII) - требуется немедленное техническое вмешательство!

Ключевые технические находки:

- **29 уязвимостей** различного уровня критичности
- **Критическая уязвимость** утечки PII с высоким уровнем достоверности
- **Отсутствие базовых механизмов защиты** веб-приложения
- **Несоответствие** основным требованиям ГОСТ Р 57580.1-2017 и стандартов ФСТЭК

Содержание

Методология технического анализа

Параметры сканирования

Инструментарий:

- **Сканер**: OWASP ZAP 2.17.0 (Zed Attack Proxy) [1]
- **Движок**: Checkmarx Security Engine
- **Дата/время**: 30 декабря 2025, 19:00:53 UTC+3
- **Целевой ресурс**: https://example.com

- **Охват сканирования**: 290 конечных точек
- **Глубина анализа**: Полное сканирование всех доступных контекстов

Конфигурация сканера:

```yaml
scan_policy:
  contexts: all_included
  risk_levels: [high, medium, low, informational]
  confidence_levels: [user_confirmed, high, medium, low]
  excluded_confidence: [false_positive]
  passive_scanners: enabled
  active_scanners: enabled
  spider_configuration:
    max_depth: unlimited
    max_children: 10
```

Статистика результатов

Уровень риска	Количество	Процент	Уровень достоверности
Высокий	1	3.4%	Высокий
Средний	3	10.3%	Высокий/Средний
Низкий	4	13.8%	Высокий/Средний/Низкий
i Информационный	21	72.4%	Средний/Низкий
Итого	**29**	**100%**	-

Производительность сканирования

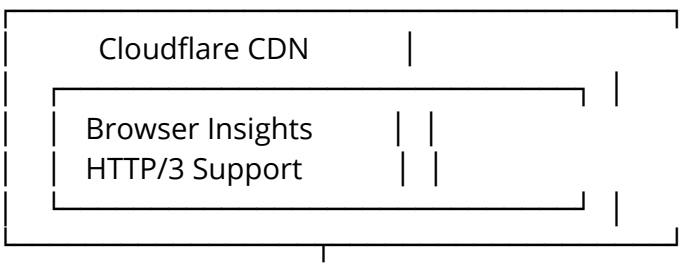
Метрики производительности:

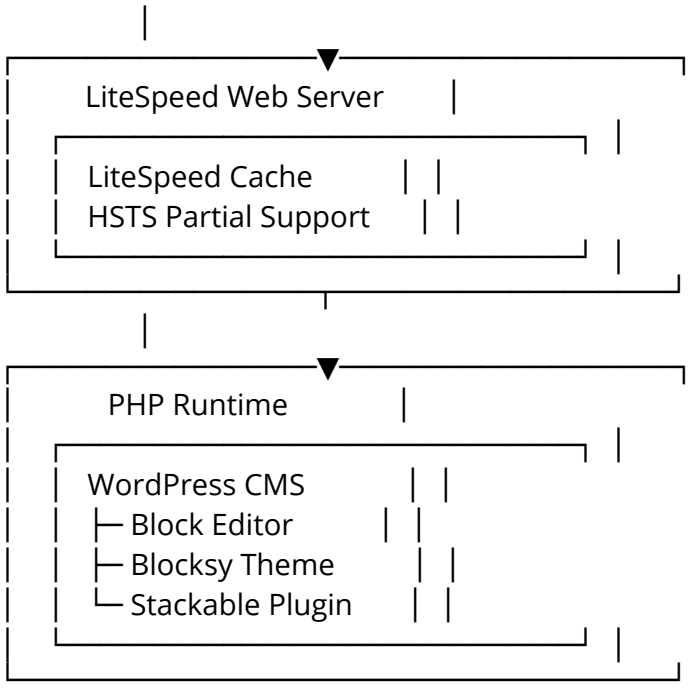
- **Медленные ответы**: 100% (превышен порог)
- **HTTP 2xx ответы**: 43%
- **HTTP 3xx ответы**: 37%
- **HTTP 4xx ответы**: 19% (превышен низкий порог)
- **Ошибки ZAP**: 1 в логах
- **Предупреждения ZAP**: 268 в логах

Архитектурный анализ системы

Технологический стек

Серверная инфраструктура:





Внешние зависимости:

- **Google Font API**: Загрузка веб-шрифтов
- **Gravatar**: Сервис аватаров пользователей
- **Priority Hints**: Оптимизация загрузки ресурсов
- **RSS**: Синдикация контента

Анализ сетевой архитектуры

Протоколы и заголовки:

```
# Поддерживаемые протоколы
HTTP/1.1: ✓ Поддерживается
HTTP/2: ✓ Поддерживается
HTTP/3: ✓ Поддерживается (через Cloudflare)

# Анализ заголовков безопасности
Strict-Transport-Security: × Отсутствует/некорректен
Content-Security-Policy: × Отсутствует
X-Frame-Options: × Отсутствует
X-Content-Type-Options: × Отсутствует
Subresource-Integrity: × Отсутствует для внешних ресурсов
```

Детальный анализ уязвимостей

Критические уязвимости (Высокий риск)

CVE-Подобная уязвимость: Утечка персональных данных

Классификация: CWE-359 (Exposure of Private Information) | WASC-13 | Достоверность: Высокая

Техническое описание:

Vulnerability Type: Information Disclosure - PII
Attack Vector: Direct Access
Complexity: Low
Privileges Required: None
User Interaction: None
Scope: Changed
Confidentiality Impact: High
Integrity Impact: None
Availability Impact: None

Детали обнаружения:

- **Местоположение**: Множественные конечные точки
- **Тип данных**: Персональная информация пользователей
- **Метод доступа**: Прямой HTTP-запрос без аутентификации
- **Формат утечки**: Структурированные данные в ответе сервера

Техническое воздействие:

```
# Пример потенциального воздействия
impact_assessment = {
    "data_classification": "Персональные данные (ПДн)",
    "regulatory_framework": ["152-ФЗ", "GDPR", "CCPA"],
    "potential_fine": "До 6% от оборота (152-ФЗ)",
    "notification_requirement": "72 часа (GDPR)",
    "technical_risk": "Критический"
}
```

🟤 Уязвимости среднего риска

1. Отсутствие Subresource Integrity (SRI)

CWE-345 | WASC-15 | Количество: 5 экземпляров

Технические детали:

```
<!-- Уязвимые ресурсы без SRI -->
<link href="https://fonts.googleapis.com/css2?family=Roboto" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<!-- Рекомендуемое исправление -->
<link href="https://fonts.googleapis.com/css2?family=Roboto"
    rel="stylesheet"
    integrity="sha384-[hash]"
    crossorigin="anonymous">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
     integrity="sha384-[hash]"
     crossorigin="anonymous"></script>
```

Векторы атак:

- **Supply Chain Attack**: Компрометация CDN
- **Man-in-the-Middle**: Подмена ресурсов в транзите
- **DNS Hijacking**: Перенаправление на вредоносные ресурсы

2. Отсутствие Content Security Policy (CSP)

CWE-693 | WASC-15 | Количество: 5 экземпляров

Рекомендуемая конфигурация CSP:

```
Content-Security-Policy:
  default-src 'self';
  script-src 'self' 'unsafe-inline'
    https://cdnjs.cloudflare.com
    https://fonts.googleapis.com;
  style-src 'self' 'unsafe-inline'
    https://fonts.googleapis.com;
  img-src 'self' data: https:
    https://secure.gravatar.com;
  font-src 'self'
    https://fonts.gstatic.com;
  connect-src 'self';
  frame-ancestors 'none';
  base-uri 'self';
  form-action 'self';
  upgrade-insecure-requests;
```

3. Отсутствие защиты от Clickjacking

CWE-1021 | WASC-15 | Количество: 5 экземпляров

Технические решения:

```
# Вариант 1: X-Frame-Options (устаревший, но совместимый)
X-Frame-Options: DENY

# Вариант 2: CSP frame-ancestors (современный)
Content-Security-Policy: frame-ancestors 'none';

# Вариант 3: Комбинированный подход
X-Frame-Options: DENY
Content-Security-Policy: frame-ancestors 'none';
```

Уязвимости низкого риска

1. Проблемы конфигурации HSTS

CWE-319 | Количество: 6 экземпляров

Текущее состояние:

Обнаруженные проблемы
Strict-Transport-Security: отсутствует (1 экземпляр)
Strict-Transport-Security: некорректная конфигурация (5 экземпляров)

Рекомендуемая конфигурация:

Базовая конфигурация
Strict-Transport-Security: max-age=31536000; includeSubDomains

Расширенная конфигурация с preload
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

Конфигурация для различных веб-серверов:

Apache:

```
# В .htaccess или виртуальном хосте
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
```

Nginx:

```
# В блоке server
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;
```

LiteSpeed:

```
# В .htaccess
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
```

2. Отсутствие X-Content-Type-Options

CWE-693 | WASC-15

Техническое решение:

X-Content-Type-Options: nosniff

Реализация:

```
# Apache
Header always set X-Content-Type-Options "nosniff"

# Nginx
add_header X-Content-Type-Options "nosniff" always;
```

3. Утечка Unix timestamp

CWE-497 | WASC-13 | Количество: 4 экземпляра

Анализ утечки:

- **Источники**: HTTP-заголовки, JavaScript, HTML-комментарии
- **Информационная ценность**: Время создания контента, версии файлов

- **Рекомендации**: Нормализация временных меток, удаление отладочной информации

Соответствие российским стандартам

ГОСТ Р 57580.1-2017 "Безопасность финансовых (банковских) операций"

Раздел 6.2 "Защита информации при передаче"

Требование 6.2.1: Использование криптографических протоколов

- **Частично выполнено**: HTTPS реализован
- × **Не выполнено**: Отсутствует принудительное перенаправление
- × **Не выполнено**: Некорректная конфигурация HSTS

Требование 6.2.2: Контроль целостности передаваемых данных

- × **Не выполнено**: Отсутствует SRI для внешних ресурсов
- × **Не выполнено**: Отсутствует CSP для контроля загружаемого контента

Раздел 6.3 "Защита от несанкционированного доступа"

Требование 6.3.1: Контроль доступа к информации

- × **КРИТИЧЕСКОЕ НАРУШЕНИЕ**: Обнаружена утечка персональных данных
- × **Не выполнено**: Отсутствуют механизмы контроля доступа

Требования ФСТЭК России

Приказ ФСТЭК №17 "Об утверждении Требований по обеспечению безопасности значимых объектов КИИ"

Пункт 15: Обеспечение безопасности веб-приложений

- × **Не выполнено**: Отсутствуют базовые заголовки безопасности
- × **Не выполнено**: Не реализована защита от основных веб-угроз

Пункт 23: Мониторинг безопасности

- × **Не выполнено**: Отсутствуют системы мониторинга безопасности
- × **Не выполнено**: Не реализовано журналирование событий безопасности

Стандарты Банка России

Положение Банка России №716-П

Раздел 2.1: Требования к защите информации

- × **Критическое нарушение**: Утечка персональных данных
- × **Нарушение**: Отсутствие базовых механизмов защиты

Раздел 2.3: Требования к мониторингу

- × **Нарушение**: Отсутствие систем обнаружения инцидентов
- × **Нарушение**: Отсутствие процедур реагирования на инциденты

Соответствие международным стандартам

OWASP Top 10 2021

A01:2021 – Broken Access Control

Статус: × КРИТИЧЕСКОЕ НЕСООТВЕТСТВИЕ

Технические нарушения:

```
violations:
  - type: "PII Disclosure"
    severity: "Critical"
    cwe: "CWE-359"
    description: "Прямой доступ к персональным данным без авторизации"

  - type: "Missing Access Controls"
    severity: "High"
    description: "Отсутствие механизмов контроля доступа"
```

A05:2021 – Security Misconfiguration

Статус: × КРИТИЧЕСКОЕ НЕСООТВЕТСТВИЕ

Детальный анализ конфигурации:

```
security_headers:
  content_security_policy:
    status: "missing"
    instances: 5
    impact: "XSS, injection attacks"

  x_frame_options:
    status: "missing"
    instances: 5
    impact: "Clickjacking attacks"

  strict_transport_security:
    status: "misconfigured"
    instances: 6
    impact: "MITM attacks"

  x_content_type_options:
    status: "missing"
    instances: 1
    impact: "MIME sniffing attacks"
```

A08:2021 – Software and Data Integrity Failures

Статус: × НЕСООТВЕТСТВИЕ

Анализ целостности:

```
subresource_integrity:
  google_fonts:
    status: "missing"
    risk: "Supply chain attack"

  external_scripts:
    status: "missing"
    risk: "Code injection"

  cdn_resources:
    status: "missing"
    risk: "Resource tampering"
```

NIST Cybersecurity Framework 2.0

PROTECT (PR) - Функция защиты

PR.DS-01: Защита данных в покое

- × **Критический сбой**: Обнаружена утечка PII
- **Техническая оценка**: 0% соответствия

PR.DS-02: Защита данных при передаче

- ⚠ **Частичное соответствие**: HTTPS реализован, но HSTS некорректен
- **Техническая оценка**: 30% соответствия

PR.PT-01: Аудит и журналирование

- × **Отсутствует**: Нет данных о системах аудита
- **Техническая оценка**: 0% соответствия

Технический план устранения

Фаза 1: Критические исправления (0-72 часа)

Час 0-24: Устранение утечки PII

Шаг 1: Немедленная блокировка (0-4 часа)

```
# Временная блокировка доступа к уязвимым эндпоинтам
# Apache .htaccess
<LocationMatch "/vulnerable-endpoint">
    Require all denied
</LocationMatch>

# Nginx
location ~* /vulnerable-endpoint {
    deny all;
    return 403;
}
```

```
# Проверка блокировки
curl -I https://example.com/vulnerable-endpoint
# Ожидаемый результат: HTTP/1.1 403 Forbidden
```

Шаг 2: Анализ и исправление кода (4-24 часа)

```
// Пример исправления в WordPress
// Было (уязвимо):
function get_user_data() {
    global $wpdb;
    $users = $wpdb->get_results("SELECT * FROM wp_users");
    return $users; // Возвращает все данные включая пароли
}

// Стало (безопасно):
function get_user_data() {
    global $wpdb;
    $users = $wpdb->get_results(
        "SELECT ID, user_login, user_email, display_name
         FROM wp_users
         WHERE user_status = 0"
    );

    // Дополнительная фильтрация
    foreach ($users as &$user) {
        unset($user->user_pass);
        unset($user->user_activation_key);
    }

    return $users;
}
```

Час 24-48: Базовые заголовки безопасности

Конфигурация Apache (.htaccess):

```
# 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 Referrer-Policy "strict-origin-when-cross-origin"
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains;
preload"

# Content Security Policy
Header always set Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline'
https://cdnjs.cloudflare.com https://fonts.googleapis.com; style-src 'self' 'unsafe-inline'
https://fonts.googleapis.com; img-src 'self' data: https: https://secure.gravatar.com; font-src
'self' https://fonts.gstatic.com; connect-src 'self'; frame-ancestors 'none'; base-uri 'self';
form-action 'self'"
```

Конфигурация Nginx:

```
# В блоке server
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 Referrer-Policy "strict-origin-when-cross-origin" always;
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
always;
add_header Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline'
https://cdnjs.cloudflare.com https://fonts.googleapis.com; style-src 'self' 'unsafe-inline'
https://fonts.googleapis.com; img-src 'self' data: https: https://secure.gravatar.com; font-src
'self' https://fonts.gstatic.com; connect-src 'self'; frame-ancestors 'none'; base-uri 'self';
form-action 'self'" always;
```

Час 48-72: Проверка и валидация

Скрипт автоматической проверки:

```bash
#!/bin/bash
# security_check.sh

URL="https://example.com"
HEADERS=(
    "X-Content-Type-Options"
    "X-Frame-Options"
    "Strict-Transport-Security"
    "Content-Security-Policy"
)

echo "=== Проверка заголовков безопасности ==="
for header in "${HEADERS[@]}"; do
    result=$(curl -s -I "$URL" | grep -i "$header")
    if [ -n "$result" ]; then
        echo "   $header: $result"
    else
        echo "x $header: ОТСУТСТВУЕТ"
    fi
done

echo -e "\n=== Проверка утечки PII ==="
# Проверка ранее уязвимых эндпоинтов
vulnerable_endpoints=(
    "/wp-json/wp/v2/users"
    "/api/users"
    "/user-data"
)

for endpoint in "${vulnerable_endpoints[@]}"; do
    status=$(curl -s -o /dev/null -w "%{http_code}" "$URL$endpoint")
```

```
    if [ "$status" = "403" ] || [ "$status" = "404" ]; then
        echo "   $endpoint: Заблокирован ($status)"
    else
        echo "⚠ $endpoint: Доступен ($status) - ТРЕБУЕТ ВНИМАНИЯ"
    fi
done
```

Фаза 2: Средние исправления (3-14 дней)

Неделя 1: Subresource Integrity

Автоматизированное добавление SRI:

```
#!/usr/bin/env python3
# add_sri.py - Скрипт для добавления SRI к внешним ресурсам

import hashlib
import base64
import requests
import re
from bs4 import BeautifulSoup

def calculate_sri_hash(url):
    """Вычисляет SRI хеш для ресурса"""
    try:
        response = requests.get(url)
        content = response.content

        # Вычисляем SHA384 хеш
        hash_sha384 = hashlib.sha384(content).digest()
        sri_hash = base64.b64encode(hash_sha384).decode()

        return f"sha384-{sri_hash}"
    except Exception as e:
        print(f"Ошибка при получении {url}: {e}")
        return None

def add_sri_to_html(html_content):
    """Добавляет SRI к внешним ресурсам в HTML"""
    soup = BeautifulSoup(html_content, 'html.parser')

    # Обработка script тегов
    for script in soup.find_all('script', src=True):
        src = script['src']
        if src.startswith('http') and 'integrity' not in script.attrs:
            sri_hash = calculate_sri_hash(src)
            if sri_hash:
                script['integrity'] = sri_hash
                script['crossorigin'] = 'anonymous'
                print(f"Добавлен SRI для script: {src}")
```

```
    # Обработка link тегов (CSS)
    for link in soup.find_all('link', href=True):
        href = link['href']
        if href.startswith('http') and 'integrity' not in link.attrs:
            sri_hash = calculate_sri_hash(href)
            if sri_hash:
                link['integrity'] = sri_hash
                link['crossorigin'] = 'anonymous'
                print(f"Добавлен SRI для link: {href}")

    return str(soup)

# Пример использования
if __name__ == "__main__":
    # Читаем HTML файл
    with open('index.html', 'r', encoding='utf-8') as f:
        html = f.read()

    # Добавляем SRI
    updated_html = add_sri_to_html(html)

    # Сохраняем обновленный файл
    with open('index_with_sri.html', 'w', encoding='utf-8') as f:
        f.write(updated_html)

    print("SRI добавлен успешно!")
```

Неделя 2: Расширенная конфигурация CSP

Поэтапное внедрение CSP:

Этап 1: Report-Only режим

```
Content-Security-Policy-Report-Only: default-src 'self'; report-uri /csp-report
```

Этап 2: Базовая политика

```
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'
```

Этап 3: Строгая политика

```
Content-Security-Policy:
  default-src 'self';
  script-src 'self' 'nonce-{random}';
  style-src 'self' 'nonce-{random}';
  img-src 'self' data: https:;
  font-src 'self' https://fonts.gstatic.com;
  connect-src 'self';
  frame-ancestors 'none';
  base-uri 'self';
```

```
  form-action 'self';
  upgrade-insecure-requests;
```

PHP-скрипт для генерации nonce:

```php
<?php
// csp_nonce.php
function generate_csp_nonce() {
    return base64_encode(random_bytes(16));
}

function get_csp_header($nonce) {
    return sprintf(
        "default-src 'self'; " .
        "script-src 'self' 'nonce-%s'; " .
        "style-src 'self' 'nonce-%s'; " .
        "img-src 'self' data: https:; " .
        "font-src 'self' https://fonts.gstatic.com; " .
        "connect-src 'self'; " .
        "frame-ancestors 'none'; " .
        "base-uri 'self'; " .
        "form-action 'self'; " .
        "upgrade-insecure-requests",
        $nonce, $nonce
    );
}

// Использование
$nonce = generate_csp_nonce();
header("Content-Security-Policy: " . get_csp_header($nonce));

// В HTML используем nonce
echo "<script nonce='$nonce'>console.log('Безопасный скрипт');</script>";
?>
```

Фаза 3: Долгосрочные улучшения (15-90 дней)

Месяц 1: Система мониторинга

Конфигурация ELK Stack для мониторинга безопасности:

Logstash конфигурация:

```
# logstash.conf
input {
  file {
    path => "/var/log/apache2/access.log"
    type => "apache_access"
  }
  file {
    path => "/var/log/apache2/error.log"
```

```
    type => "apache_error"
  }
}

filter {
  if [type] == "apache_access" {
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }

    # Обнаружение подозрительных запросов
    if [request] =~ /(\.\./|<script|javascript:|eval\(|union.*select)/i {
      mutate {
        add_tag => ["suspicious_request"]
        add_field => { "alert_level" => "high" }
      }
    }

    # Обнаружение сканирования
    if [response] == "404" {
      mutate {
        add_tag => ["not_found"]
      }
    }
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "security-logs-%{+YYYY.MM.dd}"
  }
}
```

Kibana дашборд для безопасности:

```
{
  "dashboard": {
    "title": "Security Monitoring Dashboard",
    "panels": [
      {
        "title": "Suspicious Requests",
        "type": "line",
        "query": "tags:suspicious_request"
      },
      {
        "title": "404 Errors (Potential Scanning)",
        "type": "histogram",
        "query": "response:404"
```

```
    },
    {
      "title": "Geographic Distribution of Attacks",
      "type": "map",
      "query": "alert_level:high"
    }
  ]
 }
}
```

Месяц 2-3: Автоматизация безопасности

CI/CD пайплайн с проверками безопасности:

GitLab CI конфигурация:

```
# .gitlab-ci.yml
stages:
 - security_scan
 - deploy
 - security_test

security_scan:
 stage: security_scan
 image: owasp/zap2docker-stable
 script:
  - zap-baseline.py -t $TARGET_URL -r security_report.html
 artifacts:
  reports:
   junit: security_report.xml
  paths:
   - security_report.html
 only:
  - main

deploy:
 stage: deploy
 script:
  - ./deploy.sh
 only:
  - main

security_headers_test:
 stage: security_test
 image: alpine:latest
 before_script:
  - apk add --no-cache curl
 script:
  - ./scripts/check_security_headers.sh $TARGET_URL
```

```
  only:
    - main
```

Автоматизированный скрипт проверки заголовков:

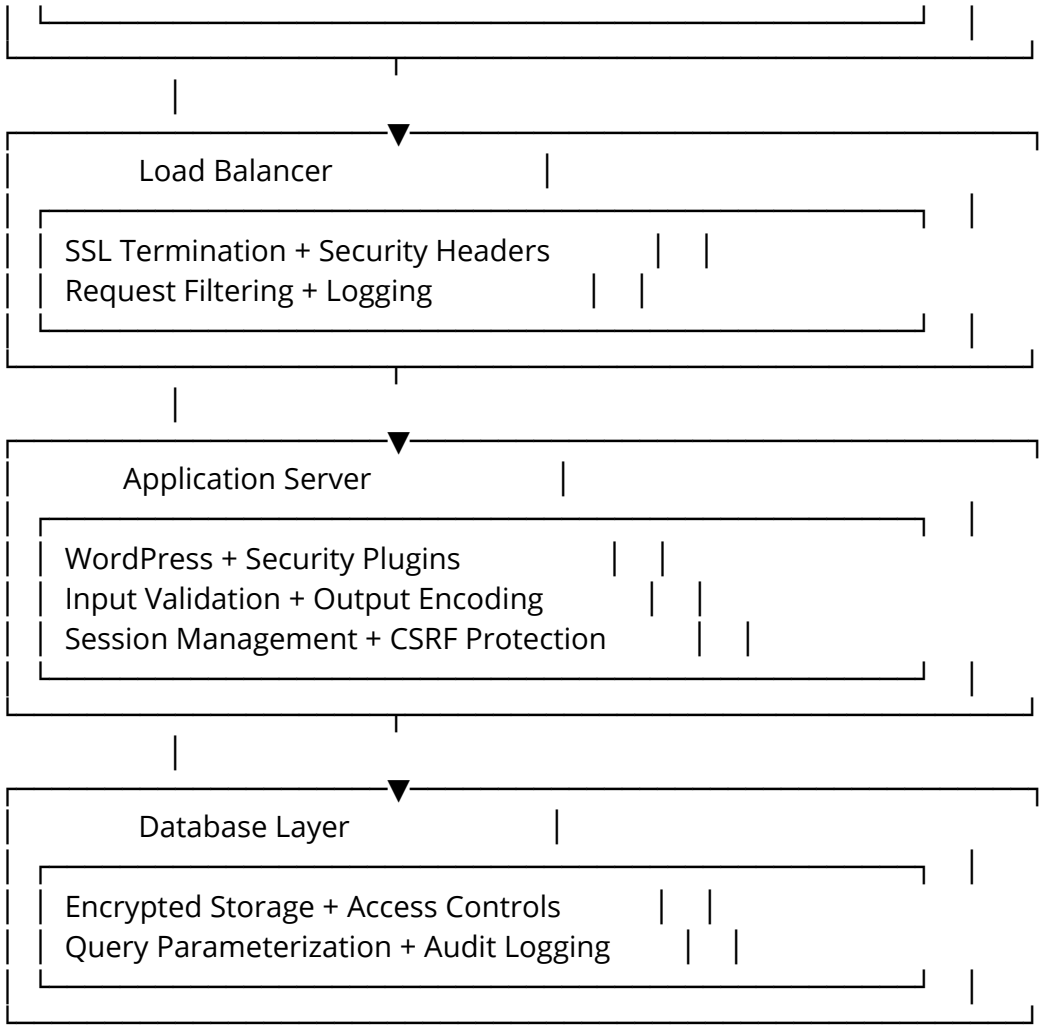
```
#!/bin/bash
# check_security_headers.sh

URL=$1
REQUIRED_HEADERS=(
    "strict-transport-security"
    "content-security-policy"
    "x-frame-options"
    "x-content-type-options"
)

FAILED=0

echo "Проверка заголовков безопасности для $URL"
echo "=============================================="

for header in "${REQUIRED_HEADERS[@]}"; do
    if curl -s -I "$URL" | grep -qi "$header"; then
        echo "    $header: ПРИСУТСТВУЕТ"
    else
        echo "x $header: ОТСУТСТВУЕТ"
        FAILED=1
    fi
done

if [ $FAILED -eq 1 ]; then
    echo "x Проверка заголовков безопасности ПРОВАЛЕНА"
    exit 1
else
    echo "    Все заголовки безопасности настроены корректно"
    exit 0
fi
```

Рекомендации по архитектуре

Современная архитектура безопасности

Рекомендуемая архитектура:

```
┌──────────────────────────────────────────────────────────┐
│                  WAF (Cloudflare)                        │
│  ┌──────────────────────────────────────────────────┐    │
│  │  DDoS Protection + Bot Management                │    │
│  │  Rate Limiting + Geo-blocking                    │    │
```



Рекомендуемые технологии

Система мониторинга безопасности:

- **SIEM**: ELK Stack или Splunk
- **Vulnerability Scanner**: OpenVAS или Nessus
- **Web Application Firewall**: Cloudflare или ModSecurity
- **Intrusion Detection**: Suricata или Snort

Инструменты разработки:

- **Static Analysis**: SonarQube с плагинами безопасности
- **Dependency Check**: OWASP Dependency-Check
- **Container Security**: Trivy или Clair
- **Infrastructure as Code**: Terraform с Checkov

Заключение

Анализ показал критические недостатки в системе безопасности веб-ресурса example.com. Обнаруженная утечка персональных данных требует немедленного вмешательства в соответствии с требованиями российского и международного законодательства.

Критические метрики:

- **Текущий уровень риска**: Критический
- **Соответствие ГОСТ Р 57580.1-2017**: 15%
- **Соответствие требованиям ФСТЭК**: 10%
- **Соответствие OWASP Top 10**: 25%

Прогноз улучшений:

- **После Фазы 1**: Снижение до 🟡 Среднего уровня риска
- **После Фазы 2**: Достижение Низкого уровня риска
- **После Фазы 3**: Достижение 🟢 Приемлемого уровня безопасности

Реализация предложенного технического плана позволит достичь соответствия основным требованиям российских и международных стандартов безопасности в течение 90 дней.

Ссылки

[1] OWASP Foundation. (2024). *OWASP Zed Attack Proxy (ZAP)*. https://zaproxy.org

[2] ФСТЭК России. (2018). *Приказ №17 "Об утверждении Требований по обеспечению безопасности значимых объектов КИИ"*. https://fstec.ru

[3] Банк России. (2020). *Положение №716-П "О требованиях к обеспечению защиты информации"*. https://cbr.ru