Отчет по аудиту безопасности веб-приложения

1. Cross-Site Scripting (XSS)

Проблема:

Приложение уязвимо к XSS-атакам, так как не все пользовательские данные экранируются перед выводом на страницу. Например, в файле admin.php вывод данных пользователя не защищен от инъекции JavaScript-кода.

Решение:

Реализовано экранирование всех выводимых данных с помощью функции htmlspecialchars().

```
// Пример исправленного кода
echo htmlspecialchars($user['name'], ENT_QUOTES, 'UTF-8');

// Для массивов
$languages = array_map(function($lang) {
    return htmlspecialchars($lang, ENT_QUOTES, 'UTF-8');
}, $user['languages']);
echo implode(', ', $languages);
```

2. Раскрытие информации (Information Disclosure)

Проблема:

Системные ошибки (например, ошибки подключения к БД) выводятся пользователям, что может раскрыть конфиденциальную информацию о структуре приложения.

Решение:

Настроено логирование ошибок в файл вместо вывода на экран. Чувствительные данные удаляются из сессии после использования.

```
// В начале каждого PHP-файла
ini_set('display_errors', '0');
error_reporting(E_ALL);
ini_set('log_errors', '1');
```

```
ini_set('error_log', __DIR__.'/error.log');

// После показа учетных данных
unset($_SESSION[‘credentials']);
```

3. SQL-инъекции (SQL Injection)

Проблема:

Учетные данные базы данных хранятся непосредственно в коде приложения, что небезопасно при утечке исходного кода.

Решение:

Учетные данные вынесены в отдельный конфигурационный файл. Все SQL-запросы используют подготовленные выражения.

```
// config.php (вне корневой директории)
return [
    'db' => [
        'dsn' => 'mysql:host=localhost;dbname=mydb',
        'username' => 'user',
        'password' => 'password'
    ]
];

// Использование в коде
$config = require '../config.php';
$pdo = new PDO(
    $config['db']['dsn'],
    $config['db']['username'],
    $config['db']['password']
);
```

4. Подделка межсайтовых запросов (CSRF)

Проблема:

Формы приложения не защищены от CSRF-атак, что позволяет злоумышленнику выполнять действия от имени авторизованного пользователя.

Решение:

Реализована система CSRF-токенов для всех форм.

```
// Генерация токена при старте сессии
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}

// Добавление в форму
<input type="hidden" name="csrf_token" value="<?=
$_SESSION['csrf_token'] ?>">

// Проверка при обработке формы
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die('Неверный CSRF-токен');
}
```

5. Уязвимости включения файлов (Include)

Проблема:

Использование внешних ресурсов (Google Fonts) создает потенциальный риск, если эти ресурсы будут скомпрометированы.

Решение:

Все внешние ресурсы (шрифты) перенесены на локальный сервер.

```
<!-- Было: -->
<link href="https://fonts.googleapis.com/css?family=Open+Sans"
rel="stylesheet">

<!-- Стало: -->
<link href="/fonts/open-sans.css" rel=«stylesheet">
```

6. Уязвимости загрузки файлов (Upload)

Проблема:

Хотя функционал загрузки файлов в текущей версии отсутствует, важно предусмотреть меры защиты на будущее.

Решение:

Подготовлен код для безопасной реализации загрузки файлов:

```
$allowedTypes = ['image/jpeg', 'image/png'];
$uploadDir = __DIR__.'/../uploads/';

if (in_array($_FILES['file']['type'], $allowedTypes)) {
    $ext = pathinfo($_FILES['file']['name'], PATHINFO_EXTENSION);
    $filename = uniqid().'.'.$ext;
    move_uploaded_file($_FILES['file']['tmp_name'], $uploadDir.
$filename);
}
```

Заключение

В результате аудита были выявлены и устранены следующие уязвимости:

- Реализована защита от XSS через экранирование всех выводимых данных
- Устранены утечки информации через отключение вывода ошибок и очистку сессии
- Улучшена защита от SQL-инъекций через вынос конфигурации и использование PDO
- Добавлена защита от CSRF с помощью токенов
- Устранены риски включения файлов через локальное хранение ресурсов
- Подготовлены меры защиты для будущего функционала загрузки файлов

Все изменения соответствуют лучшим практикам веб-безопасности и обеспечивают надежную защиту приложения.