

# Отчет по аудиту безопасности веб-приложения

---

## 1. Cross-Site Scripting (XSS)

### Проблема:

Приложение уязвимо к XSS, так как не все данные, выводимые на страницу, экранируются. Например, в admin.php строка `implode(' ', $user['languages'])` может содержать вредоносный JavaScript.

### Решение:

Использовать функцию `htmlspecialchars()` с параметрами `ENT_QUOTES` и `UTF-8` для всех данных, выводимых на страницу. Экранировать динамические строки, такие как результат `implode()`.



### Что сделано:

Добавлено экранирование для всех полей в таблице admin.php (id, fio, phone, email, birthdate, gender, languages, bio). Проверен index.php — вывод ошибок и значений формы уже использует `htmlspecialchars()`.

## 2. Раскрытие информации (Information Disclosure)

### Проблема:

Ошибки подключения к базе данных (например, `$e->getMessage()`) в index.php, save.php и admin.php могут показать структуру базы или другие детали. Учетные данные (`$_SESSION['credentials']`) в index.php показываются в открытом виде и не удаляются из сессии после отображения.

## Решение:

Отключить вывод ошибок в продакшене и логировать их в файл. Удалять чувствительные данные сессии после их показа. Добавить `ini_set('display_errors', '0')` в начале каждого PHP-файла.

```
Пример кода (исправление в index.php):
<?php if ($credentials): ?>
<div class="success-box">
  <p>Ваши учетные данные:</p>
  <p>Логин: <?php echo htmlspecialchars($credentials['login'], ENT_QUOTES, 'UTF-8'); ?></p>
  <p>Пароль: <?php echo htmlspecialchars($credentials['password'], ENT_QUOTES, 'UTF-8'); ?></p>
  <p>Сохраните эти данные!</p>
</div>
<?php unset($_SESSION['credentials']); ?>
<?php endif; ?>
```

```
Пример кода (обработка ошибок в index.php):
try {
    $pdo = new PDO(DB_DSN, DB_USERNAME, DB_PASSWORD);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    error_log("Ошибка подключения: " . $e->getMessage());
    die("Ошибка сервера. Попробуйте позже.");
}
```

## Что сделано:

Заменен вывод ошибок на безопасное сообщение во всех файлах. Добавлено удаление `$_SESSION['credentials']` после отображения.

## 3. SQL-инъекции (SQL Injection)

### Проблема:

Приложение использует подготовленные запросы PDO, что защищает от SQL-инъекций, но учетные данные базы данных жестко закодированы в коде (index.php, save.php, admin.php), что опасно при утечке.

### Решение:

Вынести учетные данные в отдельный файл config.php за пределами веб-корня. Убедиться, что все запросы используют подготовленные выражения (уже сделано).

Пример кода (новый config.php):

```
<?php
define('DB_DSN', 'mysql:host=localhost;dbname=u68860;charset=utf8');
define('DB_USERNAME', 'u68860');
define('DB_PASSWORD', '8500150');
?>
```

Пример кода (обновленный index.php):

```
<?php
session_start();
require_once '../config.php';

try {
    $pdo = new PDO(DB_DSN, DB_USERNAME, DB_PASSWORD);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    error_log("Ошибка подключения: " . $e->getMessage());
    die("Ошибка сервера. Попробуйте позже.");
}
?>
```

### Что сделано:

Создан config.php и подключен во всех файлах. Все SQL-запросы используют PDO с параметрами.

## 4. Cross-Site Request Forgery (CSRF)

### Проблема:

Формы в index.php, save.php и admin.php не защищены от CSRF-атак, что позволяет злоумышленнику отправить поддельный запрос от имени пользователя.

### Решение:

Добавить CSRF-токены в каждую POST-форму. Генерировать уникальный токен для сессии и проверять его на сервере.

```
Пример кода (добавление токена в index.php):
<?php
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
?>
<form action="save.php?action=update" method="POST">
    <input type="hidden" name="csrf_token" value="<?php echo htmlspecialchars($_SESSION['csrf_token'],
ENT_QUOTES, 'UTF-8'); ?>" />
    <!-- Остальная форма -->
</form>
```

```
Пример кода (проверка токена в save.php):
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token'])
    {
        $errors['csrf'] = 'Недействительный CSRF-токен';
    }
}
```

### Что сделано:

Добавлены CSRF-токены во все формы (index.php, admin.php). Реализована проверка токенов в save.php и admin.php.

## 5. Уязвимости включения файлов (Include)

## Проблема:

Динамических включений файлов (include с пользовательским вводом) нет, но подключение Google Fonts в index.php и admin.php создает риск, если внешний ресурс будет скомпрометирован.

## Решение:

Хранить шрифты локально в папке fonts/. Удалить подключение Google Fonts из HTML.

```
Пример кода (обновленный style.css):
@font-face {
  font-family: 'Titillium Web';
  src: url('fonts/TitilliumWeb-Regular.ttf')
  format('truetype');
}
@font-face {
  font-family: 'Titillium Web';
  src: url('fonts/TitilliumWeb-Bold.ttf') format('truetype');
  font-weight: 700;
}
body {
  font-family: 'Titillium Web', Arial, sans-serif;
  /* Остальной CSS */
}
```

```
Пример кода (удаление Google Fonts в
index.php: <link href="style.css">
```

**Что сделано:** Скачан шрифт Titillium Web и добавлен в fonts/. Удалены ссылки на Google Fonts из index.php и admin.php.


## 6. Уязвимости загрузки файлов (Upload)

### Проблема:

Приложение не поддерживает загрузку файлов, поэтому прямых уязвимостей нет. Но если функционал добавят, могут возникнуть риски (например, загрузка PHP-скриптов).

### Решение:

Проверять тип и размер файлов. Сохранять файлы за пределами веб-корня. Использовать уникальные имена файлов.



```
Пример кода (гипотетический код для save.php):
if (isset($_FILES['profile_picture'])) {
    $allowed_types = ['image/jpeg', 'image/png'];
    $file = $_FILES['profile_picture'];
    if (!in_array($file['type'], $allowed_types)) {
        $errors['file'] = 'Разрешены только JPEG и PNG';
    } elseif ($file['size'] > 2 * 1024 * 1024) {
        $errors['file'] = 'Размер файла не должен превышать 2 МБ';
    } else {
        $upload_dir = '/var/uploads/';
        $filename = uniqid() . '-' . basename($file['name']);
        move_uploaded_file($file['tmp_name'], $upload_dir . $filename);
    }
}
```

### Что сделано:

Подготовлен предупреждающий код для безопасной загрузки файлов на будущее.

## Заключение

Аудит выявил уязвимости XSS, раскрытия информации и CSRF, а также потенциальные риски включения и загрузки файлов. Все проблемы исправлены:

- Экранирование вывода для защиты от XSS.
- Скрытие ошибок и очистка сессии для предотвращения раскрытия информации.
- Вынос учетных данных в config.php для защиты от SQL-инъекций.
- Добавление CSRF-токенов для защиты форм.
- Локальное хранение шрифтов для устранения рисков включения.
- Превентивные меры для загрузки файлов.