Аудит Безопасности

1. Уязвимость: XSS

Необходимо реализовать защиту от вредоносного JavaScript кода. Конкретно в моём коде не оказалось защиты в index.php в полях bio, fio и phone. Добавим экранирование. Экранирование — это преобразование специальных символов в безопасный формат, чтобы они воспринимались как текст, а не как часть кода. Проще говоря экранирование заменяет опасные символы на их html сущности.

Добавим функцию setValue:

```
function setValue($field) {
   return isset($_COOKIE[$field]) ? htmlspecialchars($_COOKIE[$field]) : '';
}

echo htmlspecialchars($_POST['bio'], ENT_QUOTES, 'UTF-8');
```

В admin.php вывод из бд проводится без экранирования:

```
<td><?php echo htmlspecialchars($user['fio']); ?></td>
<td><?php echo htmlspecialchars($user['email']); ?></td>
```

2. Уязвимость: Information Disclosure

Уязвимость Information Disclosure возникает, когда приложение раскрывает конфиденциальную информацию. В admin.php и index.php выводятся PDO исключения.

Было:

```
catch (PDOException $e) {
            echo 'Ошибка: ' . $e->getMessage();
            exit();
```

Стало:

```
catch (PDOException $e) {
            die('Ошибка.');
```

3. Уязвимость: SQL Injection

SQL Injection — это атака, при которой злоумышленник внедряет вредоносный SQL-код в запросы к базе данных. У меня это защита уже была с прошлых работ. Все мои SQL-запросы используют prepare() и execute().

Примеры:

```
$stmt = $db->prepare("SELECT * FROM admins WHERE username = ?");
        $stmt->execute([$_SERVER['PHP_AUTH_USER']]);
        $admin = $stmt->fetch(PDO::FETCH_ASSOC);
```

```
$stmt = $db->prepare("UPDATE applications SET fio = ?, phone = ?, email
= ?, birth_date = ?, gender = ?, bio = ? WHERE id = ?");
            $stmt->execute([
                        $_POST['fio'], $_POST['phone'], $_POST['email'],
$_POST['birth_date'], $_POST['gender'], $_POST['bio'], $_SESSION['id']
            ]);
```

4. Уязвимость: CSRF

CSRF — это атака, при которой злоумышленник заставляет жертву выполнить нежелательные действия в веб-приложении, где она аутентифицирована.

Добавим токен в сессию:

```
session_start();
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

Реализуем проверку токена при обработке POST:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        if  (!isset($_POST['csrf_token'])  ||  $_POST['csrf_token']  !==
$_SESSION['csrf_token']){
        die('Ошибка CSRF: неверный токен!');
    }
```

Добавим токен в форму:

```
<form method="POST" style="display:inline;">
                                    <input type="hidden" name="csrf_token"
value="<?= $_SESSION['csrf_token'] ?>">
                                            <input type="hidden" name="id"
value="<?php echo $user['id']; ?>">
```

5. Уязвимость: Include и Upload

File Inclusion — это уязвимость, позволяющая злоумышленнику включать произвольные файлы с сервера или даже с удаленных ресурсов. Конкретно в моём коде уязвимость нет, так как `include('form.php')` в index.php не зависит от пользовательского ввода.

Уязвимость File Upload возникает, когда приложение позволяет пользователям загружать файлы без должной проверки. В моём коде отсутствует загрузка файлов.