Отчет о проведении аудита безопасности приложения.

I. **XSS** уязвимости возникают, когда приложение позволяет вводить данные, которые затем отображаются пользователю без должной проверки и очистки.

Пример проблемного кода:

```
    $messages[] = sprintf('Вы можете <a href="login.php">войти</a> с логином <strong>%s</strong>
и паролем <strong>%s</strong> для изменения данных.',
        strip_tags($_COOKIE['login']),
        strip_tags($_COOKIE['pass']));
```

Исправление ошибок:

```
$messages[] = sprintf(
    'Вы можете <a href="login.php">войти</a> с логином <strong>%s</strong> и паролем <strong>%s</strong> для изменения данных.',
    htmlspecialchars($_COOKIE['login']),
    htmlspecialchars($_COOKIE['pass'])
);
```

В данном случае, я использовала **htmlspecialchars** для безопасного вывода данных.

II. **Information Disclosure** уязвимости возникают, когда приложение раскрывает информацию, которая может быть использована злоумышленниками для атаки на систему. Явных примеров раскрытия информации в моем коде не обнаружено.

III. **SQL Injection** уязвимости возникают, когда приложение позволяет пользователю вставлять произвольный SQL-код в запросы к базе данных. Мой код защищен от данных уязвимостей, так как используются подготовленные запросы. Вот несколько примеров:

```
$stmt = $db->prepare("SELECT * FROM application_languages WHERE id = ?");
$stmt->execute([$_SESSION['uid']]);
```

```
$stmt = $db->prepare("UPDATE application SET names = ?, phones = ?, email = ?, dates = ?, gender = ?, biography = ? WHERE id = ?");
$stmt->execute([$_POST['names'], $_POST['phone'], $_POST['email'], $_POST['data'], $_POST['gender'], $_POST['biography'], $_SESSION['uid']]);
```

IV. **CSRF (Cross-Site Request Forgery)** уязвимости позволяют злоумышленнику заставить пользователя выполнить нежелательное действие на сайте, где он аутентифицирован.

В моем коде не было никакой защиты от данного вида уязвимостей, поэтому были добавлены CSRF-токены в формы:

```php
// Начало сессии
session_start();

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

А также их проверки:

```php
// Проверка CSRF-токена при обработке формы
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (empty($_POST['csrf_token']) || !hash_equals($_SESSION['csrf_token'], $_POST['csrf_token'])) {
        die('Неверный CSRF-токен.');
    }
}
```

V. Уязвимости типа **Include** возникают, когда приложение включает файлы без должной проверки, что может позволить злоумышленнику загрузить произвольные файлы. Поэтому вместо include(‘form.php’) было добавлено:

```php
    $file = 'form.php';
if (file_exists($file)) {
    include($file);
} else {
    echo "Файл не найден.";
}
```

File Upload уязвимости возникают, когда злоумышленник может загрузить произвольный файл на сервер. Но в нашем коде пользователь ничего не загружает на сервер.