**1. Cross-Site Scripting (XSS)**

**Уязвимость:**

XSS-уязвимость позволяет злоумышленнику вставлять вредоносные скрипты, которые могут быть выполнены в браузере другого пользователя.

**Защита:**

* Всегда экранируйте выводимые данные.
* Используйте встроенные функции для экранирования данных.

**Исправленный код:**

$values['names'] = isset($\_COOKIE['names\_value']) ? htmlspecialchars($\_COOKIE['names\_value'], ENT\_QUOTES, 'UTF-8') : '';

$values['phone'] = isset($\_COOKIE['phone\_value']) ? htmlspecialchars($\_COOKIE['phone\_value'], ENT\_QUOTES, 'UTF-8') : '';

$values['email'] = isset($\_COOKIE['email\_value']) ? htmlspecialchars($\_COOKIE['email\_value'], ENT\_QUOTES, 'UTF-8') : '';

$values['data'] = isset($\_COOKIE['data\_value']) ? htmlspecialchars($\_COOKIE['data\_value'], ENT\_QUOTES, 'UTF-8') : '';

$values['biography'] = isset($\_COOKIE['biography\_value']) ? htmlspecialchars($\_COOKIE['biography\_value'], ENT\_QUOTES, 'UTF-8') : '';

### 2. Information Disclosure

#### Уязвимость:

Скрипт может выводить внутренние сообщения и информацию, которая не предназначена для конечного пользователя.

#### Защита:

* Убедитесь, что отладочные сообщения не выводятся в продакшн-режиме.
* Не выводите внутреннюю информацию об ошибках пользователю.

#### Исправленный код:

ini\_set('display\_errors', '0');

error\_reporting(E\_ALL);

### 3. SQL Injection

#### Уязвимость:

SQL Injection позволяет злоумышленнику выполнять произвольные SQL-запросы.

#### Защита:

* Используйте подготовленные выражения с параметризованными запросами.
* Никогда не вставляйте данные напрямую в SQL-запросы.

#### Исправленный код:

$db = new PDO('mysql:host=localhost;dbname=u67508', 'u67508', '2263537', array(PDO::ATTR\_PERSISTENT => true));

$stmt = $db->prepare("UPDATE application SET names = ?, phones = ?, email = ?, data = ?, gender = ?, biography = ? WHERE id = ?");

$stmt->execute([$\_POST['names'], $\_POST['phone'], $\_POST['email'], $\_POST['data'], $\_POST['gender'], $\_POST['biography'], $\_SESSION['uid']]);

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

#### Уязвимость:

CSRF позволяет злоумышленнику выполнять действия от имени пользователя без его ведома.

#### Защита:

* Используйте CSRF-токены для форм и проверяйте их на сервере.

#### Пример кода:

session\_start();

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

if (empty($\_POST['csrf\_token']) || $\_POST['csrf\_token'] !== $\_SESSION['csrf\_token']) {

die('CSRF token validation failed');

}

}

// Генерация CSRF-токена

$\_SESSION['csrf\_token'] = bin2hex(random\_bytes(32));

Ваш скрипт на PHP обрабатывает данные формы и сохраняет их в базу данных. Однако в нём присутствуют несколько уязвимостей, таких как XSS, SQL Injection и Information Disclosure. Также отсутствует защита от CSRF. Давайте рассмотрим каждую из этих уязвимостей и предложим методы их устранения.

### 1. Cross-Site Scripting (XSS)

#### Уязвимость:

XSS-уязвимость позволяет злоумышленнику вставлять вредоносные скрипты, которые могут быть выполнены в браузере другого пользователя.

#### Защита:

* Всегда экранируйте выводимые данные.
* Используйте встроенные функции для экранирования данных.

#### Исправленный код:

php

Копировать код

$values['names'] = isset($\_COOKIE['names\_value']) ? htmlspecialchars($\_COOKIE['names\_value'], ENT\_QUOTES, 'UTF-8') : ''; $values['phone'] = isset($\_COOKIE['phone\_value']) ? htmlspecialchars($\_COOKIE['phone\_value'], ENT\_QUOTES, 'UTF-8') : ''; $values['email'] = isset($\_COOKIE['email\_value']) ? htmlspecialchars($\_COOKIE['email\_value'], ENT\_QUOTES, 'UTF-8') : ''; $values['data'] = isset($\_COOKIE['data\_value']) ? htmlspecialchars($\_COOKIE['data\_value'], ENT\_QUOTES, 'UTF-8') : ''; $values['biography'] = isset($\_COOKIE['biography\_value']) ? htmlspecialchars($\_COOKIE['biography\_value'], ENT\_QUOTES, 'UTF-8') : '';

### 2. Information Disclosure

#### Уязвимость:

Скрипт может выводить внутренние сообщения и информацию, которая не предназначена для конечного пользователя.

#### Защита:

* Убедитесь, что отладочные сообщения не выводятся в продакшн-режиме.
* Не выводите внутреннюю информацию об ошибках пользователю.

#### Исправленный код:

php

Копировать код

ini\_set('display\_errors', '0'); error\_reporting(E\_ALL);

### 3. SQL Injection

#### Уязвимость:

SQL Injection позволяет злоумышленнику выполнять произвольные SQL-запросы.

#### Защита:

* Используйте подготовленные выражения с параметризованными запросами.
* Никогда не вставляйте данные напрямую в SQL-запросы.

#### Исправленный код:

php

Копировать код

$db = new PDO('mysql:host=localhost;dbname=u67508', 'u67508', '2263537', array(PDO::ATTR\_PERSISTENT => true)); $stmt = $db->prepare("UPDATE application SET names = ?, phones = ?, email = ?, data = ?, gender = ?, biography = ? WHERE id = ?"); $stmt->execute([$\_POST['names'], $\_POST['phone'], $\_POST['email'], $\_POST['data'], $\_POST['gender'], $\_POST['biography'], $\_SESSION['uid']]);

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

#### Уязвимость:

CSRF позволяет злоумышленнику выполнять действия от имени пользователя без его ведома.

#### Защита:

* Используйте CSRF-токены для форм и проверяйте их на сервере.

#### Пример кода:

php

Копировать код

session\_start(); if ($\_SERVER['REQUEST\_METHOD'] == 'POST') { if (empty($\_POST['csrf\_token']) || $\_POST['csrf\_token'] !== $\_SESSION['csrf\_token']) { die('CSRF token validation failed'); } } // Генерация CSRF-токена $\_SESSION['csrf\_token'] = bin2hex(random\_bytes(32));

В форму добавьте скрытое поле с CSRF-токеном:

html

Копировать код

<input type="hidden" name="csrf\_token" value="<?php echo htmlspecialchars($\_SESSION['csrf\_token'], ENT\_QUOTES, 'UTF-8'); ?>">

### 5. File Inclusion (Include)

В данном примере уязвимость с включением файлов отсутствует, так как включение файлов не производится динамически на основе пользовательских данных.

### 6. File Upload (Загрузка файлов)

В данном примере нет обработки загрузки файлов. Однако общие рекомендации включают:

* Проверку MIME-типа файла.
* Ограничение допустимых типов файлов.
* Проверку размера загружаемого файла.
* Сохранение загруженных файлов в безопасное место.

### Общие исправления

Также добавим защиту от XSS и SQL Injection везде, где это необходимо:

foreach ($\_POST as $key => $value) {

$\_POST[$key] = htmlspecialchars($value, ENT\_QUOTES, 'UTF-8');

}