Министерство Образования, Культуры и Исследований

Молдавский Государственный Университет

Факультет Математики и Информатики

Департамент Информатики

**Лабораторная работа №2**

По курсу: Безопасность web приложений.

На тему: Методы предотвращения XSS-injection в веб приложениях.

Выполнила: Гуцу Даниил,

группа IA2102

Проверила: Плешка Наталья

Кишинев, 2024

# **Требования к лабораторной работе**

1. Продемонстрируйте как можно использовать язык JavaScript для XSS атак, через контролы созданной формы, непроверяя входные данные, перенаправив пользователя на другой сайт или вставив вместо всей страницы (все что было между <body>...</body>) спорную фотографию. То есть необходимо вместо ожидаемого контента вывести другой, неожиданный для пользователя.
2. Добавьте скрипты проверок, чтобы не было возможностей XSS атак.

**Что такое XSS-инъекция?**

**XSS (Cross-Site Scripting), «межсайтовый скриптинг»** — тип атаки на веб-системы или веб приложения, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере какого-то пользователя при открытии им этой страницы).

Уязвимость возникает из-за недостаточной фильтрации данных, которые пользователь отправляет для вставки в веб-страницу.

**Существует два направления атак:** активные и пассивные

**Активные** (называются и хранимыми или постоянными) – это вид атак, когда хакер пытается найти уязвимость в фильтре сайта. Нужно при помощи комбинации тегов и символов создать такой запрос, чтобы сайт его понял и выполнил команду. Как только дыра в безопасности найдена, в существующий запрос можно вложить «вредокод», который, к примеру, будет пересылать в удобное хакеру место.

**Пассивные** (называются еще и отражёнными или непостоянными)– требуют непосредственного вмешательства субъекта атаки. Суть заключается в том, чтобы заставить жертву перейти по вредоносной ссылке для выполнения «вредокода». Такой тип атак более сложный в реализации, ведь необходимо обладать не только техническими, но и психологическими знаниями.

**Как бороться с XSS-атаками?**

* **Экранирование данных**

Прежде чем вставлять данные пользователя на веб-страницу, убедитесь, что они проходят экранирование. Используйте функции экранирования, такие как htmlspecialchars (в PHP) для замены опасных символов на HTML-сущности.

* **Санитизация пользовательского ввода**

Проверяйте и очищайте пользовательский ввод от потенциально вредоносного кода. Можно использовать библиотеки для фильтрации ввода.

* **Корректные заголовки Content-Type**

Устанавливайте правильные заголовки Content-Type, чтобы браузер знал, как правильно интерпретировать содержимое страницы.

* **Валидация данных на сервере**

Проверяйте входные данные на сервере перед их обработкой и выводом на страницу.

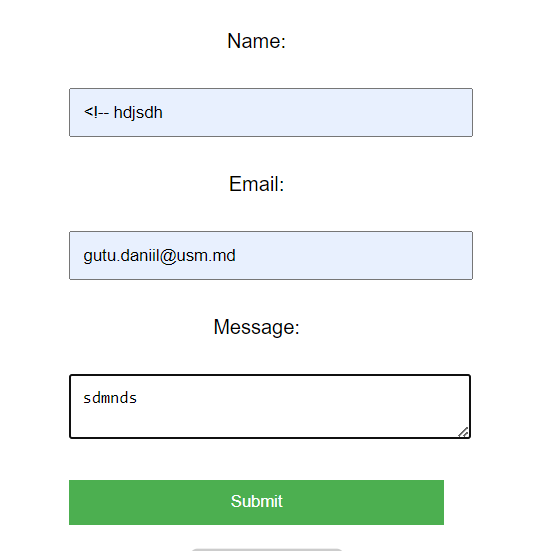
* **Обновление и патчи**

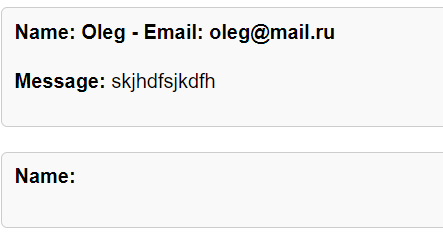
Регулярно обновляйте программное обеспечение, библиотеки и фреймворки, чтобы использовать последние безопасные версии и исправить возможные уязвимости.

Применяя сочетание этих методов, можно существенно снизить риски XSS-атак и обеспечить безопасность базы данных и приложения.

# **Попытки XSS-атак**

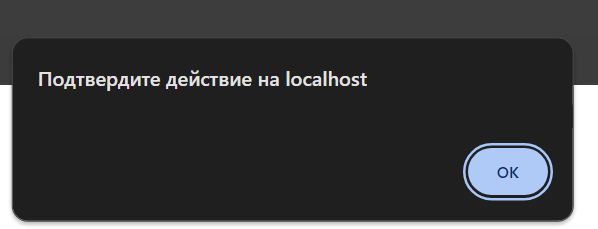
1. Вводим символы для обозначения, что начался комментарий в HTML (<!--) – после сохранения этих символов в БД все, что будет следовать после них, будет закоментированно и не будет отображаться пользователю.





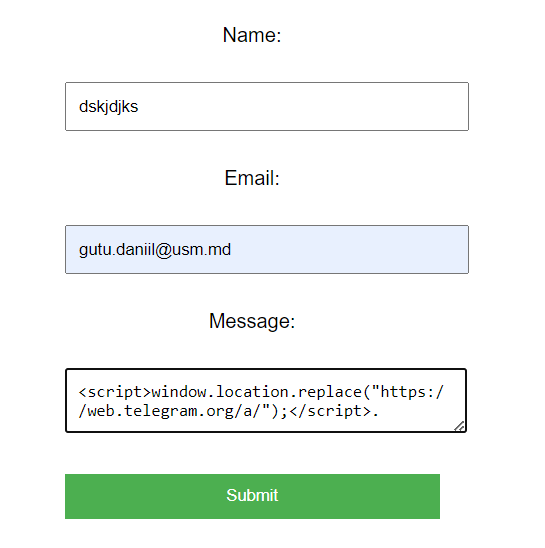
1. Заменяем контент бесконечным сообщением в alert. В комментарий вписываю такой скрипт <script>while(true){alert()}</script>.



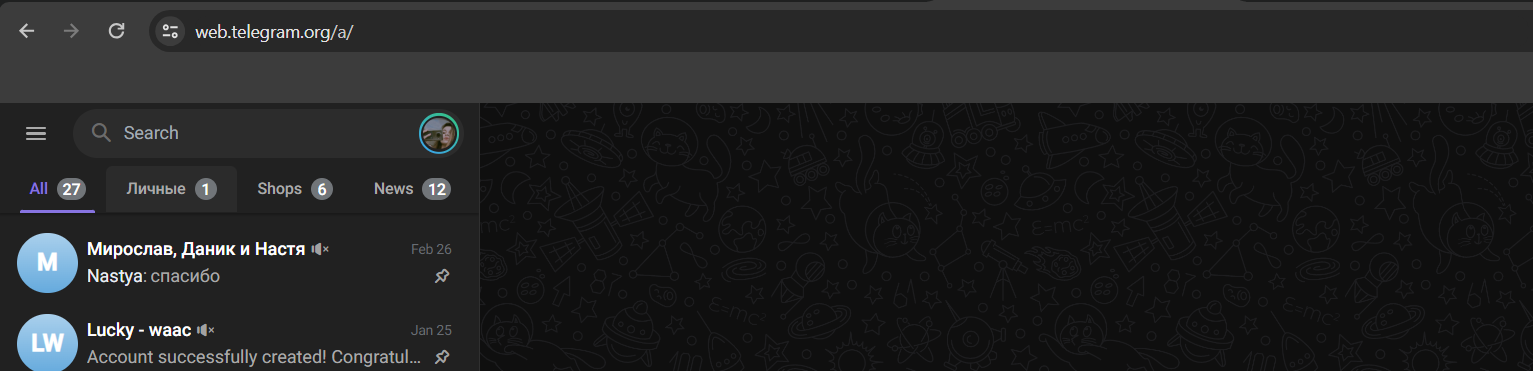


Теперь, ни один пользователь не сможет перейти на страницу с отзывами. Будет бесконечный alert.

1. Перенаправление пользователя на другой сайт при помощи JavaScript.



Теперь, после отправки формы меня перенаправило не на content.php, а на совершенно другой сайт.



Таким способом можно перенаправить на страницу с практически идентичным названием, которое будет отличаться, к примеру, всего на одну букву. Это делается с целью «кражи» клиентов или кражи личных данных клиентов компании-конкурент.

Как мы видим, весь существующий контент был заменен на картинку. Теперь, пользователи не смогут ознакомиться со страничкой отзывов.

Обеспечив безопасность от XSS атак своему приложению, проведем тест системы на ожидаемую безопасность.

**Как обеспечена безопасность в приложении?**

**При создании отзыва (внесения новых данных в БД):**

include 'sanitize.php';

$user = sanitize($\_POST['user']);

$email = sanitize($\_POST['email']);

$message = sanitize($\_POST['message']);

<?php

function sanitize($input) {

    // Защита от SQL инъекций и XSS атак

    $input = trim($input);

    $input = stripslashes($input);

    $input = htmlspecialchars($input);

    return $input;

}

?>

function validateForm() {

    var user = document.getElementById("user").value;

    var email = document.getElementById("email").value;

    var message = document.getElementById("message").value;

    if (user == "" || email == "" || message == "") {

        alert("All fields must be filled out");

        return false;

    }

    // Проверка наличия тегов и скриптов во введенных данных

    var htmlTags = /<(.|\n)\*?>/g;

    if (htmlTags.test(user) || htmlTags.test(email) || htmlTags.test(message)) {

        alert("HTML tags are not allowed");

        return false;

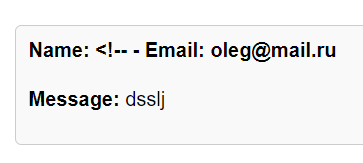
    }

    return true;

}

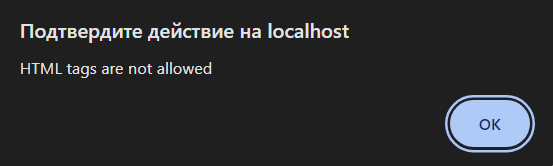
**Попытки XSS-атак в безопасной версии**

1. Вводим символы для обозначения, что начался комментарий в HTML (<!--) – после сохранения этих символов в БД все, что будет следовать после них, будет закоментированно и не будет отображаться пользователю.



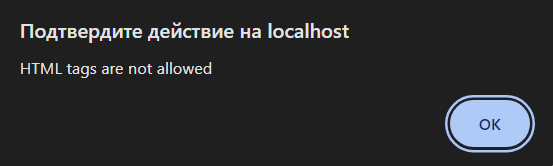
Эта проверка успешно пройдена, символы были экранированы и весь контент продолжает отображаться на странице.

1. Заменяем контент бесконечным сообщением в alert. В комментарий вписываю такой скрипт <script>while(true){alert()}</script>.



Страница успешно выведена.

1. Перенаправление пользователя на другой сайт при помощи JavaScript.



Мы были переадресованы на верную страницу, тест пройден успешно.

# **Вывод**

В результате выполнения лабораторной работы были изучены и применены методы предотвращения XSS-инъекций в веб-приложениях. Эти методы повышают безопасность приложения, предотвращая возможные атаки и обеспечивая безопасный пользовательский опыт. Эффективное применение указанных методов является ключевым аспектом создания безопасных веб-приложений.