DOI 10.37882/2223-2966.2025.04-2.34

ПОСТРОЕНИЕ СИСТЕМЫ ПОИСКА УЯЗВИМЫХ К МЕЖСАЙТОВОМУ СКРИПТИНГУ СУЩНОСТЕЙ ВЕБ-ПРИЛОЖЕНИЙ

BUILDING A SYSTEM FOR SEARCHING FOR WEB APPLICATION ENTITIES VULNERABLE TO CROSS-SITE SCRIPTING

G. Shipulin
A. Shabalin
S. Spevakova

Summary. The article is devoted to the issues related to the search for web application entities vulnerable to cross-site scripting. The paper considered the concept of cross-site scripting (XSS vulnerabilities), its main types (Reflected XSS, Stored XSS and DOM-based XSS), and described a general approach to finding this type of vulnerability in web applications. In addition, based on the analysis of existing automated search tools for web application entities vulnerable to cross-site scripting (XSS vulnerability scanners), their main limitations are identified (insufficient accuracy in identifying vulnerable entities and lack of support for processing all types of entities), the architecture of the developed system for searching for web application entities vulnerable to cross-site scripting and the relationship of its modules. The developed system covers the analysis of various types of web application entities, including user input elements, URL parameters, HTTP protocol headers, and dynamic DOM tree elements.

Keywords: cross-site scripting, web application, vulnerability scanning, security analysis, vulnerability scanners.

Шипулин Георгий Фаризович
кандидат юридических наук, доцент, Российский технологический университет (РТУ МИРЭА), г. Москва
podumai_nad@mail.ru
Шабалин Александр Денисович
Московский Политехнический Университет
ne_ponyal@list.ru
Спевакова Светлана Викторовна
ассистент, Московский Политехнический Университет
sspev@yandex.ru

Аннотация. Статья посвящена рассмотрению вопросов, связанных с поиском сущностей веб-приложения, уязвимых к межсайтовому скриптингу. В работе были рассмотрены понятие межсайтового скриптинга (XSS-уязвимости), его основные виды (Reflected XSS, Stored XSS и DOM-based XSS), описан общий подход поиска данного вида уязвимостей в веб-приложениях. Кроме того, на основе анализа существующих автоматизированных средств поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений (сканеров XSS-уязвимостей), выявлены их основные ограничения (недостаточная точность определения уязвимых сущностей и отсутствие поддержки обработки всех типов сущностей), описана архитектура разработанной системы поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений и взаимосвязь ее модулей. Разработанная система охватывает анализ сущностей веб-приложений разного типа, включая элементы пользовательского ввода, параметры URL-адреса, заголовки HTTP-протокола и динамические элементы DOM-дерева.

Ключевые слова: межсайтовый скриптинг, веб-приложение, поиск уязвимостей, анализ защищенности, сканеры уязвимостей.

Межсайтовый скриптинг (XSS-уязвимость) — уязвимость веб-приложений, при которой возможно внедрение в веб-страницу вредоносного кода с последующим его выполнением как на стороне веб-приложения, так и на стороне клиента. [1]

Основной целью XSS-атак является выполнение произвольного кода на стороне клиента или веб-приложения, что может привести к краже аутентификационных данных пользователя, компрометации его сессий, перенаправлению на фишинговые сайты и пр.

В актуальной версии OWASP Top 10 (2021) межсайтовый скриптинг не был выделен в отдельную категорию, однако включен в категорию A3 Injection, куда также входят SQL-инъекции, инъекции команд и др. [2] Несмотря на изменения в обновленной классификации OWASP Top 10, межсайтовый скриптинг остается одной из наиболее распространенных и опасных уязвимостей веб-приложений.

В рамках рассмотрения уязвимостей веб-приложений принято считать, что полезная нагрузка — это скрипт (код), внедряемый в веб-страницу с целью его выполнения на стороне пользователя (клиентская сторона), а вектор атаки — это способ или метод, посредством которого выполняется доставка полезной нагрузки. [3]

Выделяют следующие три основных вида межсайтового скриптинга:

1. Reflected XSS — данный вид XSS-уязвимости состоит отсутствии проверки веб-сервером входящих параметров, переданных в пользовательском запросе, с последующим ответом веб-сервера. Если ответ при этом форматируется тегами (или

попадает в скрипт), то результат полезной нагрузки будет отображен с учетом форматирования. Главной особенностью реализации данной уязвимости заключается в том, что полезная нагрузка не хранится на веб-сервере, а «отражается» в браузере клиента при его запросе к веб-странице.
2. Stored XSS — данный вид XSS-уязвимости возникает при наличии возможности сохранения полезной нагрузки на стороне веб-сервера (без форматирования), который вернется пользователю при открытии соответствующей веб-страницы.
3. DOM-based XSS — данный вид XSS-уязвимости применим к модели DOM (структурированное представление HTML-документа в виде иерархического дерева, в котором каждый элемент представляется отдельным объектом) и основывается на возможности получения клиентом веб-приложения данных из «недоверенных» источников. [3]

Анализ реализованных инструментальных средств с открытым исходным кодом поиска сущностей в веб-приложении, подверженных XSS-уязвимостям, показал, что большинство из них ориентированы на выявление только одного вида — Reflected XSS. [4, 5] Стоит также отметить, что по данным исследований 2020 года, XSS-уязвимости встречаются в 30–40 % всех веб-приложений. [6]

Таким образом, представляется целесообразным разработка системы поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений, которая обеспечивает анализ защищенности веб-приложения ко всем видам XSS-уязвимостей, включая DOM-based XSS и Stored XSS. Также предъявляемыми требованиями к разрабатываемой системе являются масштабируемость, высокая скорость работы и минимизация ложных срабатываний.

Общий подход поиска уязвимых к межсайтовому скриптингу сущностей независимо от вида состоит из следующих этапов:

1. Построение карты веб-приложения.

Перед проведением анализа конкретных сущностей, находящихся в коде веб-страниц, требуется определение его структуры самого веб-приложения для выявления всех доступных к обращению веб-страниц, что может быть реализовано как посредством прямого перебора URL-адресов (брутфорса), так и посредством применения технологии спайдеринга, заключающейся в обходе доступных веб-страниц на основе определения их взаимосвязей. [7]

2. Парсинг веб-страницы и построение дерева DOM.

На основе результатов работы первого этапа проводится анализ каждой веб-страницы, в рамках чего выполняется идентификация всех возможных точек входа, через которые возможна отправка на веб-сервер полезной нагрузки, а также восстановление DOM-дерева, что позволяет определять в нем уязвимые сущности при обработке динамических элементов.

3. Формирование и отправка запроса к веб-приложению.

После определения потенциально уязвимых сущностей выполняется подготовка соответствующих запросов к веб-странице, включая выбор тестовой полезной нагрузки и вектора атаки в зависимости от сущности, уязвимой к межсайтовому скриптингу, включая: формы пользовательского ввода; параметры URL-строки; заголовки HTTP-протокола; встроенные JavaScript-сценарии и обработчики событий DOM, а также от вида XSS-уязвимости.

4. Обработка ответов веб-приложения.

Данный этап состоит в обработке ответа веб-приложения на отправленный запрос с полезной нагрузкой в отношении тестируемой сущности (определение наличия/отсутствия XSS-уязвимости), а также в определении функционирующих средств защиты в веб-приложении.

На основе общего подхода поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений, а также требований к разрабатываемой системе поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений была предложена архитектура, состоящая из следующих модулей:

1. Модуль сканирования веб-страниц. Выполняет анализ структуры веб-приложения, выявляя все потенциально уязвимые элементы, такие как формы ввода данных, параметры URL-строки, заголовки HTTP и встроенные JavaScript-сценарии.
2. Модуль подготовки запросов с полезной нагрузкой. Данный модуль отвечает за выбор полезной нагрузки и подходящего вектора атаки в зависимости от типа потенциально уязвимой сущности и от вида XSS-уязвимости, включая Reflected XSS, Stored XSS и DOM-based XSS.
3. Модуль отправки запросов с полезной нагрузкой. Предназначен для формирования и отправки HTTP-запросов к веб-приложению, предварительно подготовленных модулем подготовки запросов с полезной нагрузкой.
4. Модуль обработки ответов веб-приложения. В рамках данного модуля проверяется, была ли «отражена» или сохранена полезная нагрузка в коде веб-страницы, а также определяется возможность выполнения полезной нагрузки в браузере пользователя.

5. Модуль журналирования и генерации отчетов. Результат выполнения работы каждого модуля фиксируется в журнале, на основе чего формируется отчет, содержащий информацию о найденных уязвимых сущностях, виде межсайтового скриптинга, используемой полезной нагрузке при обнаружении, а также векторе атаки.
6. Модуль управления, который представляет интерфейс взаимодействия пользователя с системой (ее модулями) посредством командной строки.

Модули системы (1–5) взаимодействуют по принципу конвейера, последовательно отрабатывая, при этом за конфигурацию и процесс работы модулей отвечает модуль управления (6).

При программной реализации описанной архитектуры системы использовался интерпретируемый язык программирования Python ввиду большого количества библиотек и фреймворков, обеспечивающих реализацию общего подхода поиска всех видов XSS-уязвимостей.

Для обеспечения сетевого взаимодействия и отправки HTTP-запросов использовалась библиотека Requests (функции requests.get(), requests.post(), requests.headers()). [8] Для парсинга HTML-кода веб-страниц и определения реализованных в веб-приложении защитных механизмов были применены функции библиотеки BeautifulSoup. [9] Эмуляция «браузерного» поведения реализована с помощью библиотеки Selenium, что обеспечивает проверку динамических изменений веб-страницы и потенциальных уязвимостей, возникающих при обработке данных в JavaScript. [10] Создание событий (результат работы каждого модуля), их ведение и сохранение результатов запуска системы было реализовано с помощью функций встроенного модуля JSON языка программирования Python (функции json.dump(), logger.info()).

Для оценки эффективности функционирования разработанной системы был проведен сравнительный анализ с двумя другими системами поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений с открытым исходным кодом (сканерами XSS-уязвимостей). В качестве тестовой среды использовалось веб-приложение Mutillidae, предназначенное для получения практических компетенций поиска и эксплуатации уязвимостей веб-приложений.

Количественные результаты поиска уязвимых к межсайтовому скриптингу сущностей по каждому виду XSS-уязвимости представлены в таблице 1.

Таблица 1.

Сравнение систем поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений

Программное средство	Вид XSS-уязвимости	Количество обнаруженных уязвимых сущностей на веб-странице	Время отработки (сек.)
XSSer	Reflected XSS	2	28
DSXS	Reflected XSS, Stored XSS	4	22
Разработанное программное средство	Reflected XSS, Stored XSS, DOM-based XSS	6	27

Таким образом, были рассмотрены основные виды межсайтового скриптинга, описан общий подход поиска уязвимых к межсайтовому скриптингу сущностей веб-приложений. Описана архитектура разработанной системы поиска уязвимых сущностей, а также отражены результаты проведенного сравнительного анализа разработанной системы с существующими, которые подтверждают эффективность предложенного решения, позволяющего выявлять все основные виды XSS-уязвимостей.

ЛИТЕРАТУРА

1. Cybersecurity Club, IIT Bombay: Cross Site Scripting (XSS) // Cybersecurity Club, IIT Bombay URL: https://csea-iitb.github.io/IITBreachers-wiki/2020/07/22/XSS.html (дата обращения: 01.03.2025).
2. Owasp: Top 10 Web Application Security Risks // Owasp.org URL: https://owasp.org/www-project-top-ten/# (дата обращения: 02.03.2025).
3. Habr: XSS атакует! Краткий обзор XSS уязвимостей // Habr URL: https://habr.com/ru/companies/alfa/articles/717896/ (дата обращения: 10.03.2025).
4. Kali tools: XSSer // Kali tools URL: https://kali.tools/?p=1758 (дата обращения: 12.03.2025).
5. GitHub: DSXS // GitHub URL: https://github.com/stamparm/DSXS (дата обращения: 13.03.2025).
6. Acunetix: 2020 web application vulnerability report // v8.2 Infographic Report URL:https://www.acunetix.com/resources/report/Acunetix_2020_Web_Application_Vulnerability_Report_Key_Takeaways.pdf (дата обращения: 15.03.2025).
7. Шипулин Г.Ф., Приймак А.Е. Использование техник спайдеринга и скрапинга при оценке состояния защищенности веб-приложений // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки. — 2024. — №11. — С. 185–187. DOI 10.37882/2223-2966.2024.11.45.
8. Requests: HTTP for Humans // Requests docs URL: https://requests.readthedocs.io/en/latest/ (дата обращения: 19.03.2025).
9. Beautiful Soup Documentation // Read the Docs URL: https://beautiful-soup.readthedocs.io/en/latest/# (дата обращения: 18.03.2025).
10. Selenium.dev: The Selenium Browser Automation Project // Selenium docs URL: https://www.selenium.dev/documentation/ (дата обращения: 20.03.2025).