УДК 004.492.2

# ИНСТРУМЕНТЫ ЗАЩИТЫ ОТ МЕЖСАЙТОВОГО СКРИПТИНГА

**Лужнова Екатерина Евгеньевна,** студент, специальность 10.05.01 Компьютерная безопасность, Оренбургский государственный университет, Оренбург
e-mail: pussy.kat@inbox.ru

Научный руководитель: **Фот Юлия Дмитриевна,** кандидат технических наук, доцент, доцент кафедры компьютерной безопасности и математического обеспечения информационных систем, Оренбургский государственный университет, Оренбург
e-mail: fotulia@mail.ru

***Аннотация.*** *Актуальность статьи обоснована значением межсайтового скриптинга как наиболее популярного метода хищения личной информации пользователей из приложений в сети Интернет, что требует подробного изучения данного явления и методов защиты от него. Методами исследования выступает библиографический анализ и формализация методов защиты от межсайтового скриптинга. Целью статьи выступает изучение сущности и особенностей межсайтового скриптинга как уязвимого программного обеспечения, используемого в информационно-телекоммуникационной сети Интернет. Изучены особенности реализации и примеры кодов сценариев, уязвимых к трем основным типам межсайтового скриптинга. Проведён обзор методов защиты от XSS-уязвимости, особое внимание уделено способам проверки интерпретации данных со стороны клиента и со стороны сервера. Сформулировано правило, позволяющее снизить вероятность межсайтового скриптинга.*

***Ключевые слова:*** *уязвимость программного обеспечения, межсайтовый скриптинг, компьютерная безопасность.*

***Благодарности:*** **с**татья подготовлена в рамках исследования, проводимого в ходе реализации стратегического проекта «Технологии и кадры для ОПК», выполняемого по программе стратегического академического лидерства «Приоритет-2030».

***Для цитирования:*** Лужнова Е. Е. Инструменты защиты от межсайтового скриптинга // Шаг в науку. – 2022. – № 4. – С. 46–48.

# CROSS-SITE SCRIPTING PROTECTION TOOLS

**Luzhnova Ekaterina Evgenievna,** student, specialty 10.05.01 Computer Security**,** Orenburg State University, Orenburg
e-mail: pussy.kat@inbox.ru

Research advisor: **Fot Julia Dmitrievna,** Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of Computer Security and Mathematical Support of Information Systems, Orenburg State University, Orenburg
e-mail: fotulia@mail.ru

***Abstract.*** *The relevance of the article is justified by the importance of cross-site scripting as the most popular method of stealing users' personal information from applications on the Internet, which requires a detailed study of this phenomenon and methods of protection against it. The research methods are bibliographic analysis and formalization of methods of protection against cross-site scripting. The purpose of the article is to study the essence and features of cross-site scripting as a vulnerability of software used in the Internet information and telecommunications network. The implementation features and examples of script codes vulnerable to the three main types of cross-site scripting are studied. The review of methods of protection against XSS vulnerability is carried out, special attention is paid to ways of checking the interpretation of data from the client and from the server side. A rule has been formulated to reduce the likelihood of cross-site scripting.*

***Key words:*** *software vulnerability, cross-site scripting, computer security.*

***Acknowledgements:*** this article was prepared as part of research conducted during the implementation of the strategic project «Technologies and personnel for the defense industry», carried out under the program of strategic academic leadership «Priority 2030».

***Cite as:*** Luzhnova, E. E. (2022) [Cross-site scripting protection tools]. *Shag v nauku* [Step into science]. Vol. 4, pp. 46–48.

Межсайтовый скриптинг (XSS – Cross-Site Scripting) – «это тип уязвимости программного обеспечения, свойственный web-приложениям (путем обхода ограничений безопасности браузера)» [4], что позволяет внедрить определенный клиентский сценарий в страницы сайта, просматриваемые другими пользователями. Межсайтовый скриптинг обычно представлен в виде специально созданной злоумышленниками гиперссылки, сохраняющей учетные данные пользователя. Такой вид уязвимости программного обеспечения используется хакерами для того, чтобы:

– изменить настройки программы, используемой пользователями;

– разместить ложную рекламу, направляющую посетителей сети Интернет на сайт-двойник или опасный электронный ресурс;

– похитить формы токенов, помогающих проводить CSRF-атаки [5].

Уязвимость XSS обладает популярностью в связи с низкой защищенностью web-приложений. Согласно результатам исследования компании «Ростелеком-Солар» по итогам 2021 года, определено, что российские компании имеют низкий уровень защищенности от XSS-атак, так как 94% уязвимостей в исследуемых организациях имеют достаточно высокий уровень критичности, при этом 75% из них исправить можно с помощью использования уже прописанных патчей, которые компании не применяют, также зачастую не обновляют вовремя системы безопасности [2].

Выделяют три основных типа XSS-уязвимостей:

1. Постоянный (хранимый) XSS – «наиболее разрушительный тип атак, предполагающий хранение вредоносного кода на сайте или сервере, при этом каждое обращение к оригинальной странице выполняет в браузере внедренный код. Классическим примером такой формы уязвимости являются форумы, на которых разрешено оставлять комментарии в HTML-формате без ограничений. При осуществлении некорректной фильтрации входные данные сохраняются в базе данных на сервере или записываются в файлы, выводимые в браузер пользователю» [3].

2. Непостоянный (отраженный) XSS – вариант атаки, использующий предоставляемые пользователями в строке запроса или HTML-форме данные для создания ответа клиенту без обработки. При этом пользователю необходимо перейти по специально сгенерированной ссылке.

3. XSS в DOM-модели (Document Object Model) – реализуется через не программный интерфейс, не зависящий от платформы и языка программирования, который предоставляет программам и сценариям «доступ к содержимому HTML- и XML-документов и изменяет их содержимое, структуру и оформление» [7]. Примером данной уязвимости служит «сценарий, получающий данные из URL через «location.*DOM» или посредством XMLHttpRequest-запроса и использующий их без фильтрации для создания динамических HTML объектов» [3].

Защита от межсайтового скриптинга может быть осуществлена с помощью использования функции «htmlspecialchars()» или её аналога «htmlentities()». Многие авторы указывают, что в HTML существуют сущности или мнемоники, предполагающие написание определенной последовательности символов в сам HTML, например «©». При этом браузер будет отображать соответствующий этой мнемонике символ, например, знак копирайта © [8].

При запуске функции «htmlspecialchars()» некоторые символы (кавычки, угловые скобки и т. д.) в строке заменяются на соответствующие им мнемоники, браузер при том выводит данную строку на экран как строку, не пытаясь выполнять её как код. Когда в форму на сайте вводится текст «<script>alert(‘hello’)</script>», функция «htmlspecialchars()» переведёт его в «<script> alert(‘hello’)</script>», при этом браузер выведет на экран преобразованную строку, не восприняв такой код как JavaScript.

Изучив мнения некоторых специалистов по компьютерной безопасности, сформулируем правило, позволяющее значительно снизить вероятность межсайтового скриптинга: «все данные, предоставленные пользователем, нужно передавать в DOM исключительно в виде строк» [1]. Но при этом данное правило функционирует не во «всех приложениях, так как во многих из них есть функции, позволяющие пользователям передавать данные в DOM» [6]. Правило можно конкретизировать: «необходимо запрещать передачу в DOM присланных пользователем данных, не прошедших очистку» [1].

Существуют способы проверки правильности интерпретации данных, переданных в DOM, «как на стороне клиента, так и на стороне сервера» [6].

Многие авторы указывают, что распознавание строк в JavaScript происходит следующим образом:

«const isString = function(x) {if (typeof x === ‘string’ || x instanceof String) {return true;} return false;};» [6].

Для проверки правильности интерпретации данных необходимо отнести числа к напоминающим строки (string-like) объектам. Также возможно воспользоваться относительно неизвестным побочным эффектом функции:

JSON.parse(): const isStringLike = function(x) {try {return JSON.stringify(JSON.parse(x)) === x;} catch (e) {console.log(‘not string-like’); } };

Встроенная функция JavaScript «JSON.parse()» преобразовывает текст в объект JSON. Сделать такое преобразование возможно для чисел и строк, но

сложные объекты, такие как функции, не соответствуют формату, совместимому с JSON, а значит, для них такое преобразование невозможно.

Вторым способом выступает изучение корректности интерпретации строкового или напоминающего строку объекта, так как данные объекты могут ошибочно интерпретироваться функцией DOM как элементы или преобразовываться в них. Чаще всего данные пользователей внедряются в DOM с помощью элементов «innerText» или «innerHTML». Но безопаснее использовать «innerText», потому что он подвергает очистке все, что выглядит как HTML-тег, представляя его в виде строки:

const userString = 'hello, world!;
const div = document.querySelector('#userComment');
div.innerText = userString;

Эндрю Хоффман считает, что «использование при добавлении строк или похожих на строки объектов в DOM элемента «innerText» вместо «innerHTML» – оптимальная практика. Элемент «innerText», просматривая HTML-теги как строки, выполняет их очистку, тогда как элемент «innerHTML» при загрузке в DOM интерпретирует теги HTML именно как теги» [6]. Но, так как у каждого браузера существует свой вариант реализации введенной информации, то даже прошедший обработку элемент «innerText» ошибочно считать полностью безопасным.

Таким образом, определено, что риск XSS можно смягчить в сети или в базе данных клиента, при этом идеальным направлением приложения усилий будет клиент, так как XSS-атаки обычно осуществляются на стороне клиента. Выявлено, что для предотвращения реализации всех трех типов межсайтового скриптинга можно использовать чисто программные методы, при этом созданный код должен формировать централизованную функцию, которая будет автоматически добавлять данные в DOM для всего приложения в целом.

**Литература**

1. Губенков А. А., Рыбалкин С. М. Разработка модуля браузера для защиты от атак типа «межсайтовый скриптинг» // Математические методы в технике и технологиях. – 2014. – № 8 (67). – С. 281–282.
2. Итоги контроля уязвимостей российских компаний за 2021 год: отчет компании «Ростелеком-Солар» [Электронный ресурс]. – Режим доступа: https://rt-solar.ru/upload/iblock/7a9/kc1dio23g2v2x657ne6xo0nu2zjlvqie/Itogi-kontrolya-uyazvimostey-rossiyskikh-kompaniy-za-2021-god.pdf (дата обращения: 06.05.2022).
3. Комкова О. Е., Ушаков К. Е. Исследование XSS-уязвимостей сервисных веб-приложений // Будущее машиностроения России: сборник докладов Двенадцатой Всероссийской конференции молодых ученых и специалистов (с международным участием), Москва, 24–27 сентября 2019 года. – Москва: Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет), 2019. – С. 898–903.
4. Мохамед А. Е. Полное пособие по межсайтовому скриптингу [Электронный ресурс]. – Режим доступа: https://www.securitylab.ru/analytics/432835.php (дата обращения: 06.05.2022).
5. Тарасов М. Уязвимость Cross Site Scripting (XSS): практическое руководство для хакеров [Электронный ресурс]. – Режим доступа: https://timcore.ru/wp-content/uploads/2021/01/Уязвимость-Cross-Site-Scripting-XSS-Практическое-руководство.pdf (дата обращения: 31.03.2022).
6. Хоффман Э. Безопасность веб-приложений: пер. с англ. / Э. Хоффман. – СПб.: Питер – 2021. – 336 с.
7. XSS уязвимость и защита от XSS. – URL: http://lifeexample.ru/razrabotka-i-optimizacia-saita/xss-uyazvimost-i-zashhita-ot-xss.html (дата обращения: 31.03.2022).
8. YongHao Li Cross-Site-Scripting (XSS) – Attacking and Defending. – URL: https://www.theseus.fi/bitstream/handle/10024/13013/Li_Yonghao.pdf?sequence=1 (дата обращения: 31.03.2022).

Статья поступила в редакцию: 29.05.2022; принята в печать: 25.10.2022.
Автор прочитал и одобрил окончательный вариант рукописи.