СИСТЕМНЫЙ АНАЛИЗ, УПРАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ / SYSTEM ANALYSIS, MANAGEMENT AND PROCESSING OF INFORMATION

DOI: https://doi.org/10.60797/itech.2025.5.1

АНАЛИЗ УЯЗВИМОСТЕЙ СТАНДАРТА CONTENT SECURITY POLICY С ЦЕЛЬЮ ПОВЫШЕНИЯ ЗАЩИТЫ ВЕБ-САЙТОВ

Обзор

Охонько Ф.С.[1, *]
[1] Tential, Пенсакола, Соединенные Штаты Америки

* Корреспондирующий автор (philip.okhonko[at]gmail.com)

Аннотация

В статье выполняется анализ уязвимостей стандарта Content Security Policy с целью повышения защиты веб-сайтов. Отмечено, что данный стандарт обладает уязвимостями, позволяющими злоумышленникам успешно реализовывать атаки, основанные на внедрении вредоносного кода в тело веб-страницы. Обоснована актуальность исследования уязвимостей стандарта CSP с целью повышения защиты веб-сайтов. Рассмотрены основные аспекты реализации защиты веб-ресурса с использованием стандарта CSP. Представлен состав наиболее известных методик обхода директив данного стандарта. Описан процесс реализации XSS атаки с использованием директивы «unsafe-inline». Выявлено, что подобная организация атаки позволит осуществить перехват пользовательских данных незаметно для пользователей и администраторов безопасности. Сделан вывод о том, что использование разработчиками директивы unsafe-inline не обеспечивает должного уровня защиты от XSS атак, а в качестве альтернативы предложено внедрение более эффективной политики CSP, настраиваемой в соответствии с рекомендациями специалистов в области информационной безопасности.

Ключевые слова: информационная безопасность, Content Security Policy, Cross-Site Scripting, защита веб-сайтов, уязвимости стандарта защиты.

ANALYSIS OF VULNERABILITIES IN THE CONTENT SECURITY POLICY STANDARD FOR IMPROVING THE PROTECTION OF WEBSITES

Review article

Okhonko F.S.[1, *]
[1] Tential, Pensacola, USA

* Corresponding author (philip.okhonko[at]gmail.com)

Abstract

This article analyses vulnerabilities in the Content Security Policy standard to improve the protection of websites. It is noted that this standard has vulnerabilities that allow attackers to successfully implement attacks based on the introduction of malicious code into the body of a web page. The relevance of CSP vulnerability research to improve the protection of websites is substantiated. The main aspects of implementing web resource protection using the CSP standard are examined. The composition of the most known methods of bypassing the directives of this standard is presented. The process of realization of XSS attack using "unsafe-inline" directive is described. It is found that such an attack organization will allow to intercept user data unnoticed by users and security administrators. It is concluded that the use of unsafe-inline directive by the developers does not provide a proper level of protection against XSS attacks, and as an alternative, it is proposed to implement a more effective CSP policy, configured in accordance with the recommendations of experts in the field of information security.

Keywords: information security, Content Security Policy, Cross-Site Scripting, website protection, security standard vulnerabilities.

Введение

Обеспечение защиты данных в рамках функционирования веб-ресурсов уже давно стало острым вопросом. Реализуется данного рода безопасность различными образами, а одним из актуальных механизмов стал отдельный уровень, именуемый Content Security Policy. Этот механизм направлен на обнаружение и устранение атак, основанных на межсайтовом скриптинге и внедрению данных злоумышленника в передаваемые запросы. Для работы использует ряд директив, позволяющих гибко настроить механизмы защиты. Однако, несмотря на высокий уровень популярности данный стандарт защиты веб-сервисов и приложений обладает рядом недостатков, изучение которых позволит определить способы их устранения для широкого круга специалистов. Однако фактически, несмотря на наличие целого набора директив, обеспечивающих разные направления защиты от инъекции кода, злоумышленники находят методы их обхода. Именно поэтому в рамках данной статьи исследуются данные методы, а также выявляется методика перенаправления запросов, которая ранее нигде не была освещена.

Основные результаты

Начать статью стоит с рассмотрения определения атаки типа Stored XSS. Так называют уязвимость в веб-системах, связанная с возможностью внедрения злоумышленником на страницу вредоносного программного кода (чаще всего на JavaScript) [3]. В результате успешной реализации данной атаки злоумышленник может осуществить перехват пары логин-пароль, личных данных пользователя или иной личной информации [6]. По факту, атаки XSS основаны на

доверии браузера к отображаемому им контенту, полученному с серверной платформы, в результате чего он запускает на исполнение все полученные скрипты.

Современные веб-сервисы уже давно реализуются так, чтобы устранить возможность реализации XSS-уязвимостей злоумышленниками. Также в данном направлении работают и разработчики веб-браузеров, которые используют инструменты валидации входной информации, а также политики Same-Origin Policy (определяет разрешения на доступ к данным для скриптов) и Content Security Policy (определяет перечень доверенных источников скриптов). Однако уязвимости ещё имеют место, и в рамках данной статьи предлагается рассмотреть таковые в составе политики CSP.

Content Security Policy (CSP) представляет собой дополнительный уровень обеспечения защиты веб-сервисов и приложений, направленный на устранение возможности реализации атак типа внедрение скриптов и выполнение нежелательного кода. Реализуется защита с применением CSP посредством формирования состава разрешенных для загрузки ресурсов источников [5], [9]. В том случае, если ресурсы или скрипты будут принадлежать какому-то из источников, не включенных в сформированный перечень, он попросту не будет загружен браузером. Таким образом устраняется возможность выполнения стороннего кода и реализации атак по внедрению вредоносного кода, результаты выполнения которых могут быть абсолютно различными – начиная от перехвата данных и заканчивая нарушением работы сайта [2].

Стандарт CSP обладает несколькими версиями, которые обеспечивают полный уровень совместимости. Исключение составляет вторая версия стандарта, имеющая ряд несоответствий в плане совместимости с остальными версиями. Однако это все равно позволяет браузерам работать с серверами, используя различные версии стандарта. В случае если сервером не будет предложен заголовок CSP, то браузером будет попросту применена стандартная политика для данного источника [1], [10].

Для того чтобы сформировать перечень доверенных источников, в рамках стандарта CSP на веб-странице должны быть размещены специализированный заголовок Content-Security-Policy и набор директив, к которым относятся:

- img-src – настройка источников для загрузки изображений;
- media-src – настройка источников для загрузки медиаконтента (видео, анимации, аудио);
- script-src – настройка источников для загрузки сценариев и скриптов для веб-страницы;
- frame-src – настройка источников для загрузки веб-элементов;
- default-src – резервная директива. В том случае, если в одной или нескольких директивах, представленных выше, отсутствуют аргументы, то браузер использует источники, указанные в данной директиве [4].

За счет использования представленного перечня директив разработчики могут указать для загрузки домен, поддомен, либо отдельную страницу, а также указать правила взаимодействия. Для этого при описании директивы могут быть использованы две директивы: "none" запрещает, а "self" разрешает использование ресурсов, находящихся на указанном источнике.

Помимо общих правил для отдельных директив существуют собственные правила. Так, существует правило "unsafe-inline", которое разрешает использование JS-скриптов, размещаемых непосредственно в коде веб-страницы. И если данное правило не прописать, то любой скрипт, загруженный не из доверенного источника, будет заблокирован. Правило "unsafe-eval" позволяет разрешить или заблокировать (по умолчанию) выполнение динамического кода в процессе его работы [10].

Для того чтобы получить возможность использования политики CSP, следует выполнить настройку возврата соответствующего заголовка на веб странице. В том случае, если разработчик хочет протестировать политику CSP, существует другая директива – Content-Security-Policy-Report-Only. При её использовании не будет происходить блокировка загрузки скриптов, однако все зафиксированные нарушения и отклонения будут внесены в специальные отчеты, которые отправляются системному администратору.

Однако, несмотря на, казалось бы, простоту реализации, и при этом возможности гибкой настройки разрешенных источников для веб-ресурсов, CSP имеет уязвимости, посредством которых злоумышленники могут реализовать угрозу XSS [8].

Одна из выявленных уязвимостей завязана на использовании правила "unsafe-inline" в директиве "script-src". Казалось бы, данная директива позволяет ограничить возможности передачи конфиденциальной информации на сторону. В том случае, если данная директива не включена, то разработчикам необходимо вручную внести огромнейший список ресурсов, на которые эти данные могут быть переданы. А некорректная настройка данной директивы может привести к ограниченной работоспособности приложения. Именно по этой причине разработчики зачастую используют данную директиву, считая, что она позволит обеспечить защиту от передачи информации на другие ресурсы, однако на практике это не всегда так.

В результате проведенного исследования было установлено, что если использовать метод редиректа, то будет получена возможность получения всех необходимых параметров. Для этого необходимо сформировать URL-ссылку, в которой в составе GET-запроса будут указаны необходимые параметры, которые требуется получить. Пример такого запроса на рисунке 1:

```
const e = encodeURIComponent;
const url = `http://attacker-server.com/?u=${e(location.href)}&c=${e(document.cookie)}
&d=${e(document.documentElement.innerHTML)}&l=${e(JSON.stringify(localStorage))}`;
document.location = url;
```

Рисунок 1 - Процесс формирования URL-ссылки и передача конфиденциальных данных на сервер-перехватчик

DOI: https://doi.org/10.60797/itech.2025.5.1.1

Представленная выше ссылка позволит получить cookie сессии, однако данный вариант обладает определенным недостатком – атака предполагает переадресацию пользователя на сервер-перехватчик, что по итогу может нарушить работу атакуемого веб-сайта. Для устранения этого недостатка подход к реализации перехвата данных необходимо доработать. Сперва необходимо реализовать сервер-перехватчик таким образом, чтобы он принимал данные не возвращая ответ и удерживая соединение открытым. Затем произвести переход по ссылке на сервер-перехватчик с указанием необходимых для перехвата параметров, после чего вызвать процедуру "window.stop", который отменит перенаправление на сервер-перехватчик и сделает атаку незаметным для пользователя.

Визуально ход перехвата с применением данного метода представлен на рисунке 2 и может быть описан тремя этапами.

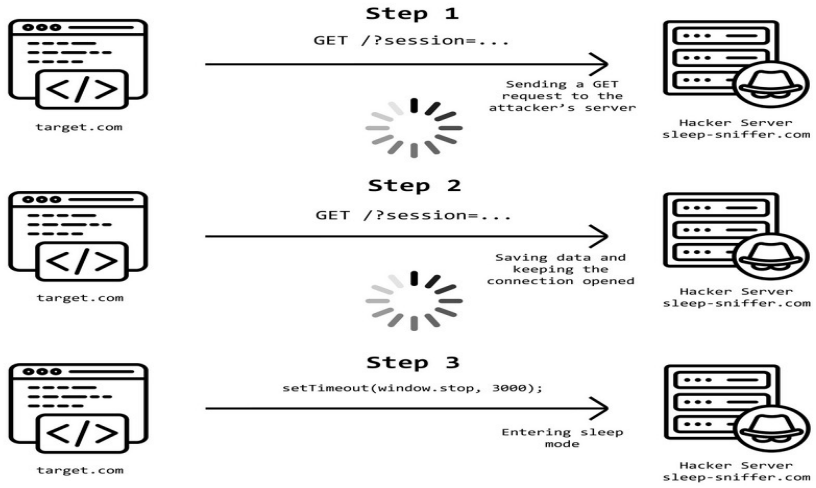

Рисунок 2 - Процесс реализации XSS атаки на страницу, использующую CSP-директиву "unsafe-inline"
DOI: https://doi.org/10.60797/itech.2025.5.1.2

На первом шаге выполняется переадресация на указанный сервер sleep-sniffer.com с отправкой указанных атрибутов сессии. При этом на странице отображается индикатор загрузки страницы, то есть пользователь считает, что происходит загрузка страницы и ожидает завершения данного процесса. На втором шаге происходит получение запроса сервером-перехватчиком, он сохраняет полученные в составе запроса параметры, а далее удерживает соединение открытым, при этом не отправляя ответ. Особенностью данного метода является тот факт, что в момент ожидания ответа на запрос браузер считает, что страница работает нормально, все скрипты на ней выполняются, а пользователь видит индикатор загрузки и ожидает её завершения. По истечении трех секунд происходит выполнение setTimeout. Это будет переходом на третий шаг – при выполнении метода window.stop происходит отмена загрузки абсолютно всех ресурсов – мультимедиа, шрифты, стили оформления и прочего. При этом не выполненные запросы будут просто отменены. То есть веб-страница будет работать далее, все необходимые данные будут переданы в составе запроса на сервер sleep-sniffer.com, а пользователь не будет ничего подозревать. Для реализации метода перехвата данных с использованием редиректа была написана утилита, окно которой представлено на рисунке 3 [3]. Она позволяет указать состав параметров, которые будут переданы в рамках запроса, а также адреса атакуемой страницы и сервера-перехватчика.

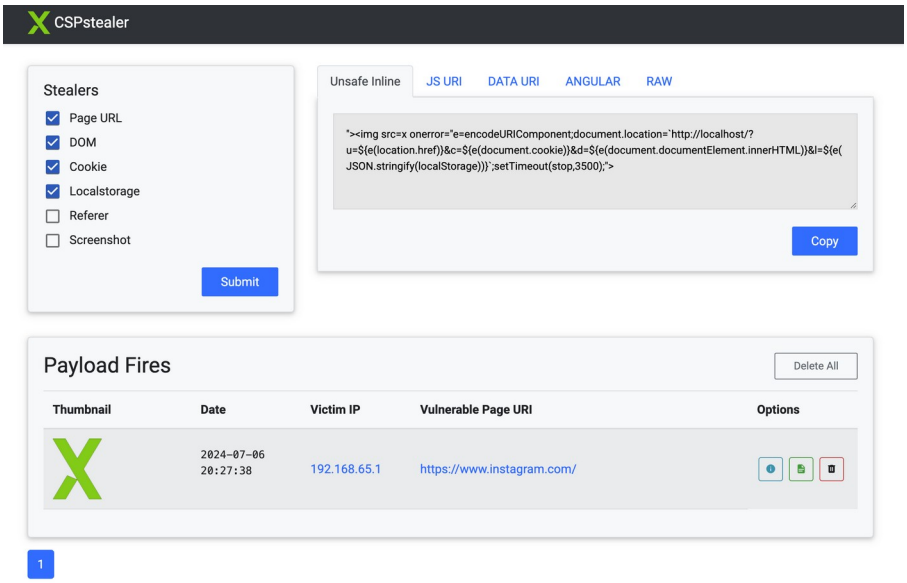

Рисунок 3 - Окно утилиты, реализующей запросы с редиректом различных наборов данных [3]
DOI: https://doi.org/10.60797/itech.2025.5.1.3

Для того чтобы устранить описанную возможность редиректа с запросом необходимых параметров в CSP, была добавлена директива "navigate-to", которая позволяет указать перечень внешних источников, на которые разрешен редирект с текущей страницы. С одной стороны, она позволяет задать список доменов или адресов, на которые возможно выполнение запросов. С другой – реализация подобного списка требует колоссальных затрат на формирование данного перечня, а также будет приводить к тому, что этот перечень необходимо будет постоянно актуализировать. По этой причине единственным способом обеспечения высокого уровня безопасности будет отказ от использования директивы unsafe-inline и внедрение более эффективной политики CSP, несмотря на все требуемые для этого трудовые и финансовые затраты.

Стандарт CSP включает множество директив и их разнообразные конфигурации, что в ряде случаев может создавать уязвимости, посредством которых злоумышленники могут реализовать угрозу XSS [7]. Также существуют другие известные методы обхода данного стандарта:

- внедрение скрипта посредством использования механизмов загрузки файлов;
- внедрение скрипта с использованием публичной CDN;
- внедрение скрипта посредством использования механизма Third-Party EndPoints + JSONP;
- внедрение скрипта с использованием процедур Angular;

Однако данные методы предполагают только внедрение кода на веб-страницу, что не решает проблему передачи конфиденциальных данных со страницы на сервер атакующего. Как и в случае с включенной опцией unsafe-inline, будет действовать ограничение, которое разрешает коммуникацию только с доверенными веб-ресурсами, что препятствует отправке данных со страницы на сервер атакующего. В таком случае можно использовать представленный выше метод в сочетании с другими известными методиками обхода стандарта CSP для обхода данного ограничения.

Таблица 1 - Пример использования полезной нагрузки сгенерированной с помощью утилиты CSPStealer и метода обхода CSP посредством использования механизма Third-Party EndPoints + JSONP

DOI: https://doi.org/10.60797/itech.2025.5.1.4

"><script src="/api/jsonp?callback=(function(){e=encodeURIComponent;document.location=`http://attacker-server.com/?u=${e(location.href)}&c=${e(document.cookie)}&d=${e(document.documentElement.innerHTML)}&l=${e(JSON.stringify(localStorage))}`;setTimeout(stop,3500);})();//"></script>

Заключение

В результате использования любой из перечисленных уязвимостей злоумышленники получают возможность перехвата различного рода информации, в том числе и данные для авторизации. А их комбинация расширяет возможные методики реализации XSS-атак, и добавляет работы специалистам по безопасности.

Подытоживая, следует отметить тот факт, что несмотря на наличие механизма CSP, направленного на защиты от инъекций программного кода в веб-страницу, качество его работы в большинстве случае оставляет желать лучшего, а злоумышленники находят все новые и новые способы его обхода. В рамках исследования был продемонстрирован существующий критический риск использования директивы unsafe-inline. По этой причине единственным способом

обеспечения высокого уровня безопасности будет отказ от использования директивы unsafe-inline и внедрение более эффективной политики CSP, несмотря на все требуемые для этого трудовые и финансовые затраты.

Конфликт интересов

Не указан.

Рецензия

Все статьи проходят рецензирование. Но рецензент или автор статьи предпочли не публиковать рецензию к этой статье в открытом доступе. Рецензия может быть предоставлена компетентным органам по запросу.

Conflict of Interest

None declared.

Review

All articles are peer-reviewed. But the reviewer or the author of the article chose not to publish a review of this article in the public domain. The review can be provided to the competent authorities upon request.

Список литературы / References

1. Content Security Policy (CSP). — URL: https://developer.mozilla.org/ru/docs/Web/HTTP/CSP (accessed: 15.08.2024).
2. Content Security Policy (CSP) Bypass. — URL: https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass (accessed: 22.08.2024).
3. GitHub – sysmustang/csp-stealer: Tool to retrieve web-page secrets and bypass Content Security Policy. — URL: https://github.com/sysmustang/csp-stealer (accessed: 22.08.2024).
4. Petkova L. Content security policy validation / L. Petkova // Security & Future. — 2019. — Vol. 3. — Issue 1. — P. 6–9.
5. Ганжур А.П. Политика безопасности контента / А.П. Ганжур, А.С. Отакулов, Н.В. Дьяченко // Молодой исследователь Дона. — 2021. — № 6 (33). — С. 41–44.
6. Использование перехвата форм для обхода CSP. — URL: https://teletype.in/@callme_txt/CONTENT-SECURITY-POLICY (дата обращения: 16.08.2024).
7. Кнышов В.А. Механизм защиты от XSS-атак Trusted Types / В.А. Кнышов, А.В. Свищев // Моя профессиональная карьера. — 2023. — Т. 3. — № 48. — С. 46–51.
8. Правила директив: «Content Security Policy: ключ 'unsafe-inline' в директивах CSP». — URL: https://csplite.com/ru/csp97/ (дата обращения: 17.08.2024).
9. Челухин В.А. Сетевые уязвимости / В.А. Челухин, Е.П. Душкин // Наука, инновации и технологии: от идей к внедрению. — Комсомольск-на-Амуре : Комсомольский-на-Амуре государственный университет, 2022. — С. 300–302.
10. Добрышин М.М. Модель сетевых атак типа xss- и sql-инъекций на вебресурсы, учитывающая различные уровни сложности их реализации / М.М. Добрышин, Д.Е. Шугуров, Д.Л. Беляев // Известия ТулГУ. Технические науки. — 2021. — № 2. — С. 196–204.

Список литературы на английском языке / References in English

1. Content Security Policy (CSP). — URL: https://developer.mozilla.org/ru/docs/Web/HTTP/CSP (accessed: 15.08.2024).
2. Content Security Policy (CSP) Bypass. — URL: https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass (accessed: 22.08.2024).
3. GitHub – sysmustang/csp-stealer: Tool to retrieve web-page secrets and bypass Content Security Policy. — URL: https://github.com/sysmustang/csp-stealer (accessed: 22.08.2024).
4. Petkova L. Content security policy validation / L. Petkova // Security & Future. — 2019. — Vol. 3. — Issue 1. — P. 6–9.
5. Ganzhur A.P. Politika bezopasnosti kontenta [Content Security Policy] / A.P. Ganzhur, A.S. Otakulov, N.V. Dyachenko // Molodoj issledovatel' Dona [Young Researcher of the Don]. — 2021. — № 6 (33). — P. 41–44. [in Russian]
6. Ispol'zovanie perehvata form dlja obhoda CSP [Using form interception to bypass CSP]. — URL: https://teletype.in/@callme_txt/CONTENT-SECURITY-POLICY (accessed: 16.08.2024). [in Russian]
7. Knyshov V.A. Mehanizm zashhity ot XSS-atak Trusted Types [Trusted Types XSS attack protection mechanism] / V.A. Knyshov, A.V. Svishchev // Moja professional'naja kar'era [My Professional Career]. — 2023. — Vol. 3. — №. 48. — P. 46–51. [in Russian]
8. Pravila direktiv: "Content Security Policy: kljuch 'unsafe-inline' v direktivah CSP" [Rules of directives: "Content Security Policy: the 'unsafe-inline' key in CSP directives"]. — URL: https://csplite.com/ru/csp97/ (accessed: 17.08.2024). [in Russian]
9. Chelukhin V.A. Setevye ujazvimosti [Network vulnerabilities] / V.A. Chelukhin, E.P. Dushkin // Nauka, innovacii i tehnologii: ot idej k vnedreniju [Science, innovation and technology: from ideas to implementation]. — Komsomolsk-on-Amur : Komsomolsk-on-Amur State University, 2022. — P. 300–302. [in Russian]
10. Dobryshin M.M. Model' setevyh atak tipa xss- i sql-in'ekcij na vebresursy, uchityvajushhaja razlichnye urovni slozhnosti ih realizacii [Model of network attacks such as xss and sql injections on web resources, taking into account different levels of complexity of their implementation] / M.M. Dobryshin, D.E. Shugurov, D.L. Beljaev // Izvestija TulGU. Tehnicheskie nauki [Bulletin of Tula State University. Technical Sciences]. — 2021. — № 2. — P. 196–204. [in Russian]