УДК 004.056

ПОЛИТИКА БЕЗОПАСНОСТИ КОНТЕНТА

А. П. Ганжур, А. С. Отакулов, Н. В. Дьяченко
Донской государственный технический университет (г. Ростов-на-Дону, Российская Федерация)

Рассмотрены методики повышения безопасности программного обеспечения на примере DevSecOps. Политика безопасности контента (CSP) — это стандарт компьютерной безопасности, обеспечивающий дополнительный уровень защиты от межсайтовых сценариев (XSS), clickjacking (перехвата кликов) и других атак с использованием кода, основанных на выполнении вредоносного контента в контексте доверенной веб-страницы. Используя подходящие директивы CSP в заголовках ответа в виде протокола передачи гипертекста (HTTP), можно выборочно указать источники данных, которые должны быть разрешены в соответствующем веб-приложении. Описано, как использовать заголовки CSP для защиты сайтов от атак XSS и других попыток обойти политику безопасности контента.

Ключевые слова: информационная безопасность, компьютерная безопасность, вредоносный контент, политика CSP, защита сайтов.

CONTENT SECURITY POLICY

A. P. Ganzhur, A. S. Otakulov, N. V. Dyachenko
Don State Technical University, (Rostov-on-Don, Russian Federation)

The paper discusses an issue related to the methodology for improving software security using the example of DevSecOps. A Content Security Policy (CSP) is a computer security standard that provides an additional layer of protection against cross-site scripting (XSS), clickjacking, and other code-based attacks that execute malicious content in the context of a trusted web page. By using suitable CSP directives in HTTP response headers, you can selectively specify which data sources should be allowed in your web application. This article describes how to use CSP headers to protect sites from XSS attacks and other attempts to bypass CSP policy.

***Keywords*:** information security, computer security, malicious content, CSP policy, site protection.

Введение. CSP — это защитная мера от любых атак, которые основаны на выполнении вредоносного контента в надежном веб-контексте или на других попытках обойти политику одного и того же происхождения. С помощью CSP возможно ограничить источники данных, разрешенные веб-приложением, указав соответствующую директиву в заголовке ответа HTTP.

Снижение риска межсайтового скриптинга. Основная цель CSP — смягчение и обнаружение XSS-атак, которые используют доверие браузера к контенту, полученному с сервера [1, 2]. Браузер жертвы подвергается выполнению вредоносных скриптов, потому что он доверяет источнику контента.

CSP позволяет администраторам серверов уменьшить или исключить возможность злоумышленника запускать XSS, указав браузеры доменов Интернета, которые должны рассматриваться в качестве законных источников исполняемых сценариев. Совместимые с CSP браузеры запускают сценарии, содержащие исходные файлы из доменов белого списка, и

игнорируют все другие, включая встроенный сценарий и атрибуты обработки событий посредством языка разметки гипертекста (HTML).

Смягчение перехвата пакетов и принудительное применение протокола HTTPS в качестве расширения HTTP. Помимо внесения в белый список доменов, с которых браузер может загружать контент, серверы также могут указывать разрешенные протоколы. Например, сервер может указать, что браузеры должны загружать контент через HTTPS.

Комплексная политика защиты передачи данных включает в себя не только реализацию HTTPS, но также пометку всех файлов cookie с помощью логического атрибута secure и автоматическое перенаправление страниц HTTP на HTTPS. Кроме того, сайты могут использовать заголовки HTTPS, чтобы браузеры подключались только через зашифрованные каналы.

Примеры заголовков CSP. Веб-сервер может добавлять к каждому ответу заголовок HTTP под названием Content-Security-Policy. Имеется возможность установить следующие свойства в заголовке CSP:

1. Default-src — необязательный метод, если другие атрибуты не определены. В большинстве случаев значение этого свойства — self, то есть браузер может загружать ресурсы только с текущего веб-сайта.

2. Script-src — места, из которых могут быть загружены внешние скрипты. Если ваш веб-сайт или приложение не использует клиентские сценарии, следует установить значение none.

3. Img-src — места, из которых можно получить изображения.

4. Media-src — места, из которых можно получить мультимедийные данные, например, видео.

5. Object-src — места, из которых можно получить плагины.

6. Manifest-src — места, из которых можно получить манифесты приложений.

7. Frame-ancestors — места, из которых другая веб-страница может быть загружена с помощью элементов frame, iframe, объекта управления, встраивания или апплета.

8. Form-action — URL-адреса, которые можно использовать как часть действия в атрибуте тега <form>. Это означает, что браузер ограничивает отправку результатов формы. Действие формы не возвращается к методу default-src, поэтому это является обязательным свойством, если используются элементы формы на соответствующем сайте.

9. Plugin-types — набор подключаемых модулей, которые можно вызывать с помощью объектов управления, встраиваемых приложений или апплетов, определенных с использованием MIME-типов.

10. Base-uri — разрешает использование URL-адресов в атрибуте src любого тега.

Уязвимости, предотвращаемые CSP. Злоумышленники могут попытаться получить доступ к ресурсам по незащищенному протоколу. В этом случае целесообразно использовать CSP для принудительного применения протокола HTTPS к любому значению, определенному в атрибутах * -src, добавив префикс https: // к любому URL-адресу в белом списке. Таким образом, ресурсы никогда не будут загружаться через незашифрованное HTTP-соединение. Можно добиться того же эффекта, добавив свойство block-all-mixed-content. Кроме того, CSP может предотвратить следующие распространенные уязвимости:

— беззнаковые встроенные операторы языка разметки CSS в тегах <style>;

— беззнаковый встроенный язык JavaScript в тегах <script>;

— динамический язык CSS с использованием метода CSSStyleSheet.insertRule ();

— динамический код Javascript с использованием функции eval ().

Лучше всего хранить скрипт и CSS в отдельных файлах, на которые ссылается HTML-страница. Если необходимо, можно включить ее с использованием ключевых слов unsafe-eval и unsafe-inline.

Рекомендации по использованию CSP. Сложные веб-приложения более чувствительны к XSS, поэтому использование CSP в этом случае является целесообразным. Использовать CSP желательно для любого приложения, которое управляет конфиденциальными данными, например, для административных пользовательских интерфейсов, консолей управления устройствами или любых продуктов, содержащих файлы, документы или сообщения, созданные пользователями. В современных фреймворках добавить стандарт CSP легко, и он может обеспечить высокую окупаемость инвестиций с точки зрения дополнительной безопасности.

CSP может быть не лучшим выбором в случаях использования статических приложений, размещенных на их собственных доменах или поддоменах, без входа в систему или файлов cookie, приложений, которые взаимодействовали с XSS в прошлом, а также имеют известные уязвимости в шаблонах или фреймворках, которые они используют. В этом случае целесообразно вложить средства в исправление уязвимого кода, потому что CSP сам по себе не обеспечивает достаточной защиты. Его следует добавлять поверх защищенного приложения, не имеющего известных уязвимостей.

Заключение. Лучший способ реализации политики безопасности контента — это добавить CSP задним числом ко всему веб-сайту, т. е. определить полностью пустой белый список, по сути блокируя все. Первоначально запустить CSP в режиме «только отчет» означает, что браузер оценивает правила, но еще не блокирует контент. Затем целесообразно оценить ошибки и принять решение: какие из них следует добавить в список или запретить.

Определенную сложность представляет решение о блокировке. Например, если используется сценарий, размещенный через сети доставки CDN, и разрешены адреса, то принимается весь трафик, исходящий из этих CDN, который может включать вредоносную составляющую.

Запуск CSP в режиме отчета в течение нескольких недель или, самое большее, нескольких месяцев должен выявить все возможные случаи ошибок. Когда набор правил охватывает все подходящие варианты использования, необходимо отключить только отчет и блокировать ресурсы, которых нет в белом списке.

Библиографический список

1. Флорен, М. В. Организация управления доступом / М. В. Флорен // Защита информации. Инсайд. — 2019. — № 3. — С. 83–97.

2. Тарасов, Ю. Контрольно-пропускной режим на предприятии / Ю. Тарасов // Защита информации. Инсайд. — 2020. — № 1. — С. 43–45.

Об авторах:

Ганжур Алексей Петрович, старший преподаватель кафедры «Вычислительные системы и информационная безопасность» Донского государственного технического университета (344003, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), Iamganzhur@yandex.ru

Дьяченко Никита Владимирович, студент кафедры «Вычислительные системы и информационная безопасность» Донского государственного технического университета (344003, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), nikita7890@yandex.ru

Отакулов Артур Собирович, студент кафедры «Вычислительные системы и информационная безопасность» Донского государственного технического университета (344003, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), diletants23z@gmail.com

About the Authors:

Ganzhur, Aleksey P., Senior Lecturer Department of «Computing Systems and Information Security», Don State Technical University (1, Gagarin sq., Rostov-on-Don, RF, 344003), Iamganzhur@yandex.ru

Dyachenko, Nikita V., Student Department of «Computing Systems and Information Security», Don State Technical University (1, Gagarin sq., Rostov-on-Don, RF, 344003), nikita7890@yandex.ru

Otakulov, Artur S., Student Department of «Computing Systems and Information Security» Don State Technical University (1, Gagarin sq., Rostov-on-Don, RF, 344003), diletants23z@gmail.com