УДК 004.056.5

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ CONTENT SECURITY POLICY ДЛЯ ЗАЩИТЫ ПОЛЬЗОВАТЕЛЕЙ WEB-ПРИЛОЖЕНИЙ

Давыдовский С.В.
Белорусский государственный университет информатики и радиоэлектроники, г. Минск
Научный руководитель: Лещев А.Е., м.т.н.

В данной статье рассматривается технология Content Security Policy, обеспечивающая безопасность пользователей Web-приложений, раскрываются детали ее использования и делается вывод о ее применимости.

Сегодня Web-приложения используются повсеместно. Их используют в том числе финансовые организации, учреждения здравоохранения, туристические и транспортные организации. Все они хранят конфиденциальную информацию о своих клиентах, которая, попав в руки злоумышленника, может нанести большой вред. Потеря доверия клиентов ведет к большим убыткам. Поэтому безопасность является одной из главных забот разработчиков.

Вся безопасность web-приложений основана на принципе одинакового источника (Same Origin Policy), который ограничивает доступ к данным других источников. Однако существуют методы, позволяющие его обойти. Одним из самых распространенных способов атаки на Web-приложения до сих пор остается “межсайтовый скриптинг” (Cross Site Scripting – XSS). По данным OWASP, в 2013 году XSS находился на третьем месте в списке ключевых рисков Web-приложений. XSS позволяет злоумышленнику внедрять свой код на страницу Web-приложения. Этот код выполнится на компьютере клиента, скачавшего эту страницу. Таким образом, злоумышленник может получить доступ к данным пользователя. Традиционные методы борьбы с XSS – кодирование и валидация входных данных. Недостатком этих методов является то, что приходится рассматривать каждый пользовательский ввод и проверять его на вредоносность. Крупные сайты имеют множество форм для ввода клиентских данных, так что уследить за всем становится сложно.

Новый механизм защиты был разработан в Mozilla Foundation. Этот механизм получил название “Политика защиты контента” (Content Security Policy - CSP). Он не защищает от внедрения вредоносного кода, но существенно снижает его вредоносный эффект. CSP определяет список доверенных источников и велит браузеру отправлять запросы только к ним. Таким образом, даже если злоумышленнику удастся внедрить свой код, он не сможет получить никакой информации, поскольку браузер клиента заблокирует запрос на недоверенный источник. В ноябре 2012 года CSP стал кандидатом в рекомендации W3C. В июле 2015 года ему на смену пришел CSP Level 2. По данным *caniuse.com,* на октябрь 2015 года поддержка браузерами CSP 1.0 составляет 87%, CSP Level 2–55%.

По умолчанию браузеры работают без CSP. Для его включения необходимо, чтобы страницы Web-приложения имели дополнительный заголовок в HTTP-ответе: *Content-Security-Policy*. Значение заголовка – строка, определяющая одну или более политик безопасности, которые будут использоваться на странице.

Формат CSP-заголовка (абзацы и отступы не должны присутствовать в заголовке):

Content-Security-Policy:
directive source-expression, source-expression, …;
directive …;
…

directive – строка, определяющая тип ресурса, source-expression – шаблон, описывающий один или более источников, откуда ресурсы могут быть загружены. Список доступных директив:

- *base-uri*
- *child-src*
- *connect-src*
- *default-src*
- *font-src*
- *form-action*
- *frame-ancestors*
- *frame-src*
- *img-src*
- *media-src*
- *object-src*
- *plugin-types*
- *report-uri*
- *sandbox*
- *script-src*
- *style-src*

Формат *source-expression*: *протокол://домен:порт*. Домен может начинаться с *, что означает любой поддомен. Порт и протокол могут быть * (любой порт и любой протокол), также они могут быть опущены. Можно указать только протокол (к примеру, *https*). Дополнительно, *source-expression* может иметь следующие значения:

- *'none'* – запрещает все источники;
- *'self'* – разрешает только источники, обслуживаемые текущим хостингом (но не поддоменами);
- *'unsafe-inline'* – разрешает использовать инлайн-скрипты, инлайн-стили, *javascript:URLs*;
- *'unsafe-eval'* – разрешает использовать кодогенерацию в JavaScript.

По умолчанию, все ресурсы открыты для загрузки из любых источников. Директива *default-src* позволяет переопределить это поведение.
Пример CSP-заголовка:
Content-Security-Policy:
script-src 'self' scripts.example.com;
media-src 'none';
*img-src *;*
default-src 'self' http://.example.com*

В данном примере страница может загружать скрипты только из текущего хостинга и из *scripts.example.com*, аудио- и видеофайлы недоступны с любых источников, картинки могут быть загружены откуда угодно, все остальные ресурсы могут быть доступны из текущего хостинга и с любого поддомена *example.com*.

CSP можно включить также напрямую в разметке страницы, используя тэг <*meta*> с атрибутом *http-equiv*. Пример:
<*meta http-equiv="Conten-Security-Policy" content="script-src 'self' scripts.example.com; media-src 'none'; img-src *; default-src 'self' http://*.example.com"*>
В <*meta*> нельзя использовать директивы *frame-ancestors*, *report-uri* и *sandbox*.

CSP 1.0 позволяет либо полностью запретить инлайн-скрипты, либо разрешить их использование (по умолчанию, инлайн-скрипты отключены, для их включения нужно указать *'unsafe-inline'* у директивы *script-src).* Поскольку самая большая угроза от XSS-атак – внедрение кода на страницу – рекомендуется полностью отказаться от них. Если же все-таки без них нельзя обойтись CSP, Level 2 предлагает решения. Можно добавить атрибут *nonce* к инлайн-скриптам. Значение атрибута *nonce* – уникальное токен-значение, которое меняется при каждом запросе. В CSP-заголовке, соответственно, у директивы *script-src* нужно указать тот же самый токен. Таким образом, известные инлайн-скрипты (с атрибутом nonce) будут выполнены, а неизвестные (без атрибута *nonce* либо если значение *nonce* не совпало со значением в директиве) будут заблокированы. К примеру: <*script nonce="EDNnf03nceIOfn39fn3e9h3sdfa"*>*…*</*script*> *Content-Security-Policy: script-src 'nonce-EDNnf03nceIOfn39fn3e9h3sdfa'* Есть и другой способ. Можно взять хэш от инлайн-скрипта (от кода между <*script*>*…*</*script*>) и в директиве указать это хэш вместе с алгоритмом хэширования. Пример: <*script*>*alert('Hello, world.');*</*script*> *Content-Security-Policy: script-src 'sha256-qznLcsROx4GACP2dm0UCKCzCG-HiZ1guq6ZZDob_Tng='*CSP способен отправлять уведомления о блокированных источниках на сервер, так что можно проанализировать и исправить уязвимости. Уведомление передается в формате JSON, где указаны заблокированные источники, нарушенные директивы и другая информация. Для того, чтобы браузер отправлял POST-запросы на сервер, необходимо указать директиву *report-uri* и путь, куда посылать уведомления, к примеру: *Content-Security-Policy: default-src 'self'; …; report-uri /my_csp_report_parser*. Существует заголовок *Content-Security-Policy-Report-Only*, использование которого не блокирует недоверенные источники, но посылает о них отчеты на сервер. Можно использовать два заголовка одновременно.

Использование CSP значительно повышает защиту от XSS-атак, однако эта защита не является стопроцентной. Злоумышленники могут обойти CSP с помощью внедрения собственных директив, могут использовать новые способы отправки данных на сторонние сайты без JavaScript (CSS Attribute Reading, SVG-кейлоггер и др.). Несмотря на это, CSP – большой шаг в сторону безопасного веба. Его можно и нужно использовать уже сейчас.

Список цитированных источников

1. World Wide Web Consortium [Электронный ресурс], 2015. – Режим доступа : http://www.w3.org/TR/CSP2. – Дата доступа: 10.10.2015.
2. An Introduction to Content Security Policy [Электронный ресурс] / Mike West, 2015. – Режим доступа: http://www.html5rocks.com/en/tutorials/security/content-security-policy. – Дата доступа: 10.10.2015.

УДК 004.4

ТИПИЗАЦИЯ ФОРМИРОВАНИЯ ПЕЧАТНЫХ ФОРМ ДЛЯ СИСТЕМ ЭКОНОМИЧЕСКОЙ НАПРАВЛЕННОСТИ

Дулуб В.В.
Брестский государственный технический университет, г. Брест
Научный руководитель: С.В.Мухов, к.т.н., доцент

При проектировании компьютерных систем экономической направленности вследствие их специфики, как правило, выделяют списковые и итоговые печатные формы. При этом, как правило, используют фильтр для уточнения области выбираемых данных типа интервального задания временного интервала и указания объекта, для которого необходимо сформировать печатную форму.

Для реализации формирования печатной формы необходимо предварительно сформировать промежуточный набор данных с помощью соответствующего запроса и передачу сформированного набора данных в программу генерации печатной формы с использованием соответствующего шаблона печатной формы с указанием уровней отчета и суммируемых полей. Для реализации запроса с указанием фильтра будем использовать таблицу «Настройки системы» для ввода и хранения параметров фильтра. Для ввода параметров фильтра можно использовать типизированную экранную форму ввода данных в таблицу «Настройки системы». В случае формирования списковой печатной формы, включающей все объекты распечатываемой таблицы, таблица «Настройки системы» не используется. В системах экономической направленности, как правило, в качестве фильтра используется задание интервала времени для выбираемых операций и объект или группа объектов для выборки в печатную форму. В качестве объекта в системах экономической направленности могут служить синтетические счета, субсчета и признаки аналитического учета, используемые в балансовых экономических моделях.

При формировании печатной формы типа *список всех объектов таблицы* выполняем:
– создаем запрос с использованием на входе данной таблицы для всех или только необходимых для отчета полей;
– создаем отчет на базе ранее сделанного запроса.

В качестве примера такой печатной формы может служить распечатка произвольного небольшого справочника, в котором задание фильтров не имеет смысла.

При формировании печатной формы типа список объектов таблицы с использованием фильтра выполняем:
– создаем запрос с использованием на входе данной таблицы для всех или только необходимых для отчета полей и таблицы «Настройки системы»;
– добавляем в список выбираемых полей необходимые для создания фильтра поля из таблицы «Настройки системы». Отметим, что ввод полей в таблицу «Настройки системы» выполняется с помощью типизированной экранной формы «Ввод …. для формирования ….»;