МЕЖСАЙТОВЫЕ АТАКИ С ВНЕДРЕНИЕМ СЦЕНАРИЯ (XSS)

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

А.С. Михайлов

Саломатин С.Б. – канд. техн. наук

Рассмотрено вопросы, связанные незаконное использование JavaScript, правила безопасности, чтобы предотвратить возможную XSS атаку, и предоставлено собственное исследование, связанное с проверкой сайтов на наличие организованной безопасности веб-сайтов.

Введение

Во время межсайтовой атаки с внедрением сценария (XSS) атакующая сторона внедряет в легальную Web-страницу вредоносный код, который затем запускает вредоносный сценарий на стороне клиента. При посещении пользователем зараженной страницы сценарий загружается в браузер пользователя и там запускается. Эта схема имеет множество разновидностей. Вредоносный сценарий может получать доступ к cookie-файлам браузера, сеансовым маркерам или другой чувствительной информации, хранящейся в браузере. Межсайтовый скриптинг (XSS) — пожалуй, самый типичный вид уязвимостей, широко распространённых в веб-приложениях. По статистике, около 65 % сайтов в той или иной форме уязвимы для XSS-атак.

Цель работы состоит в разработке и исследовании программ защиты информации от межсайтового скриптинга.

1. Межсайтовые атаки с внедрением сценария

Рассматриваются следующие схемы возможные атаки в рамках схемы работы межсайтовой атаки с внедрением сценария.

Хранимые XSS (постоянные) - один из самых опасных типов уязвимостей, так как позволяет злоумышленнику получить доступ к серверу и уже с него управлять вредоносным кодом (удалять, модифицировать).

Отраженные XSS (непостоянные) - в этом случае вредоносная строчка выступает в роли запроса жертвы к зараженному веб-сайту.

DOM-модели - в этом варианте возможно использование как хранимых XSS, так и отраженных. Суть заключается в следующем: злоумышленник создает URL-адрес, который заранее содержит вредоносный код, и отправляет его по электронной почте или любым другим способом пользователю, человек переходит по этой ссылке, зараженный сайт принимает запрос, исключая вредоносную строку, на странице у пользователя выполняется сценарий, в результате чего загружается вредоносный скрипт и злоумышленник получает cookies.

2. Правила безопасности

Защититься от XSS возможно, но защита должна применяться последовательно, без исключений и упрощений, желательно с самого начала разработки веб-приложения. Внедрение защиты на более поздних этапах может быть дорогостоящим делом.

Проверка входных данных. Проверка ввода — только первая линия защиты веб-приложения. При таком типе защиты мы знаем лишь то, как сейчас используются ненадёжные данные, и на этапе получения данных не можем предсказать, где и как они будут дальше применяться. Сюда относятся практически все текстовые данные, так как мы всегда должны обеспечивать пользователю возможность написания кавычек, угловых скобок и других символов.

Проверка работает лучше всего благодаря предотвращению XSS-атак на данные, которым присущи предельные значения. Допустим, целое число не должно содержать специфические для HTML символы. Параметры, такие как название страны, должны соответствовать заранее заданному списку реальных стран, и т. д.

Проверка входных данных помогает контролировать данные с определённым синтаксисом. Например, допустимый URL-адрес должен начинаться с префикса http:// или https://, а не с гораздо более опасных конструкций javascript: или data:. По сути, все адреса, полученные из непроверенных входных данных, должны проверяться на наличие этих тегов. Экранирование URI

javascript: или data: имеет такой же эффект, как экранирование легального URL-адреса. То есть вообще никакого эффекта.

Хотя проверка входных данных не может блокировать всю зловредную полезную нагрузку при XSS-атаке, она способна остановить наиболее очевидные типы атаки.

Экранирование (а также кодирование). Экранирование данных на выходе позволяет гарантировать, что данные не будут ошибочно восприняты принимающим парсером или интерпретатором. Очевидные примеры — знаки «меньше» и «больше», которые обозначают HTML-теги. Если позволить этим символам быть вставленными из ненадёжных входных данные, злоумышленник сможет вводить новые теги, которые браузер будет отрисовывать. Обычно эти символы заменяются последовательностями > и $lt;.

Замена символов предполагает сохранение смысла экранированных данных. Экранирование просто заменяет символы, имеющие определённое значение, альтернативными. Обычно используется шестнадцатеричное представление или что-то более читабельное, например, HTML-последовательности (если их применение безопасно).

Способ экранирования зависит от того, какого типа содержимое внедряется. Экранирование HTML-кода отличается от экранирования JavaScript, которое, в свою очередь, отличается от экранирования адресов. Применение неверной экранирующей стратегии для определённого контекста способно привести к неэффективности защиты, к созданию уязвимости, которой может воспользоваться злоумышленник.

Заключение

Разработанные алгоритмы, реализованные в виде программы позволяют повысить эффективность защиты веб-приложения от атак. Они значительно упрощают процесс тестирования веб-ресурса разработчиком, благодаря функционалу формирования отчета с рекомендациями по устранению найденных уязвимостей.

Список литературы

1. Элхади А.М. Полное пособие по межсайтовому скриптингу // [Электронный ресурс].
2. Элхади А.М. Уязвимости веб-приложений: пора анализировать исходный код // [Электронный ресурс].
3. Джатана Н., Агравал А., Собти.К. Пост эксплуатация XSS: продвинутые методы и способы защиты// [Электронный ресурс].