МЕЖСАЙТОВАЯ АТАКА С ВНЕДРЕНИЕМ СЦЕНАРИЯ

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

Каленик К.Г.

Новиков В.И. – кандидат техн. наук, доцент, УО"ВГКС"

Межсайтовая атака с внедрением сценария (XSS) на текущий день является одной из самых распространенных атак на уровне приложения, которые хакеры используют для незаконного проникновения в Web-приложения. XSS – это атака, направленная не на сервер или сайт напрямую, а на клиентов этого сайта и их конфиденциальную информацию.

Важно отметить, что XSS сама по себе не является уязвимостью, а XSS лишь одним из возможных способов эксплуатации уязвимости определенного класса.

Целью такой атаки могут быть cookies, или другая информация, идентифицирующая пользователя. Располагая регистрационными данными легитимного клиента, атакующий получает возможность действовать от его имени при работе с сайтом, что наверняка приведёт к взлому всей системы безопасности с похищением или фальсификацией данных о пользователе.

Для получения cookie злоумышленник, после выявления XSS уязвимостей, создает ссылку с вредоносным кодом. В этом коде данные из куков передаются на сторонний сервер, на котором обрабатываются и сохраняются, затем пользователь перенаправляется на страницу обратно. Все это происходит незаметно для человеческого глаза, и поэтому атака остается необнаруженной.

Атаки, построенные по принципу межсайтингового скриптинга, можно разделить на два типа:

Активные XSS атаки – подразумевают внедрение ссылки в саму страницу ресурса.

Пассивные XSS атаки – пользователь сам должен вставить ссылку в адресную строку или просто кликнуть по ней.

Всё это осуществляются через браузер жертвы: атакующий старается заставить пользователя перейти по злонамеренной ссылке. Это может быть реализовано многими способами от размещения ссылки на изображении до отправки электронного сообщения.

Самый простой пример XSS атаки можно представить запросом:

http://index.php?name=guest<script>alert('XSS')</script>

Чаще всего в современные браузеры по умолчанию встроена система безопасности, предотвращающая такие атаки. Они защищают от XSS прямого действия, но могут пропустить более сложные, например хранимые XSS и основанные на DOM XSS атаки. Одна из брешей в системе защиты браузера основана на «доверительной» политике доменов.

До того как фильтр XSS в браузере обработает исходящий HTTP запрос, он проверит, откуда этот запрос отправлен. Считается, что системе нет смысла рассматривать запросы, отправленные с того же домена, т.к. это простая трата ресурсов. Однако XSS может прихоть от любого источника. Злоумышленник может просто применить отраженную XSS атаку, и перенаправить запрос на «доверенный» домен средствами JavaScript.

Для борьбы с самой атакой может быть достаточно экранирования всех строк, попадающие в HTML-документ. Однако устранять необходимо не XSS, а уязвимость её породившую, что влечёт за собой необходимость защищенной обработки данных. Такую обработку можно реализовать при помощи определения степени доверия к данным, типизации и валидации всей входящей информации, а также приведение выходных данных к виду, безопасному для принимающей стороны.

Предположим, на сайте есть форма для размещения комментариев, которые будут отображаться сразу же после добавления. Злоумышленник может попытаться разместить комментарий, содержащий JavaScript код. После отправки формы, все данные переправляются на сервер и сохраняются в базе данных. После этого информация извлекаются из базы и новый комментарий отображается на HTML странице, вместе с внедрённым вредоносным кодом. Он может перенаправлять пользователя на какую-то вредоносную страницу или на фишинговый сайт.

Для защиты приложений, можно пропустить входной поток данных через функцию strip_tags(), которая удалит все присутствующие теги. Тогда при отображении данных в браузере, используется функция htmlentities(), преобразующая все возможные символы в соответствующие HTML-сущности.

Эти правила актуальны для любых атак, обусловленных уязвимостью этого класса. Например, для SQL-инъекций, внедрений команд и т.д. И, хотя эти атаки давно известны, и разработчики давно знают, что данным, поступающим на сервер от клиента доверять нельзя, они зачастую не задумываются о том, что обратное также верно, причем по тем же самым причинам. Тем не менее, если бы защищенная обработка данных была реализована и на стороне клиента, это позволило бы свести к минимуму риски клиентских атак, связанные с эксплуатацией уязвимостей на сервере.

Список использованных источников:

1. XSS: the easiest way to hack your website in 2014 [Электронный ресурс] – электронные данные – Switzerland: 2014. – Режим доступа: https://www.htbridge.com/blog/xss_the_easiest_way_to_hack_your_website_in_2014.html, свободный.
2. Sebastian Lekies, Ben Stock, Martin Johns. A tale of the weaknesses of current client-side XSS filtering [Текст]: доклад / конф. Black Hat USA 2014.