CERT-UA

XSS или уязвимость которой уязвимы 100% государственных веб-ресурсов Украины

Автори:
Міснік Олексій
Антонішин Михайло
Computer Emergenscy Response Team of Ukraine – CERT-UA
04119, Україна, м. Київ, вул. Мельникова, 83б
тел.: +380 44 281 88 25
факс: +380 44 489 31 33
www: cert.gov.ua

1. Что такое XSS?

XSS расшифровывается как “Cross-site scripting” (межсайтовый скриптинг). Буква "X", в сокращении, используется для того чтобы не было путаницы, ведь абривиатура "CSS" имеет ещё одну расшифровку – “Cascading Style

Sheets (Каскадные таблицы стилей)”. Года 3-4 назад на различных сайтах хакерской тематики можно было найти множество статей в которых данный вид атак именовался как "CSS". Из-за этого люди, которые не знали что такое XSS, думали что CSS и XSS это разные виды атак но во многом схожие. Данный тип атак интересен тем, что работает не по системе “клиент атакует сервер”, а совсем наоборот – “сервер атакует клиента”. То есть вредоносный код должна вызвать жертва (а с этим иногда бывают трудности). Итак, что же такое XSS? Уязвимости типа XSS представляют собой вставку произвольного HTML/JavaScript-кода в результат работы скрипта, когда сценарий не фильтрует поступившие от пользователя данные или фильтрует их недостаточно.

Исторически сложилось так что программисты не понимают всей опасности уязвимостей данного типа. Спросите любого программиста чем опасна XSS-уязвимость и почти 99% ответят Вам что с помощью этой уязвимости взломщик сможет только украсть cookies. Из-за этого многие сайты сейчас страдают XSS-уязвимостями. И если разработчики платно/бесплатно-распространяемого веб-приложения стараются устранить подобные уязвимости как можно быстрее, то разработчики отдельных сайтов просто не хотят заморачивать себе этим голову и оставляют всё как есть. Практически то же самое твориться и с большинством взломщиков (по крайней мере в нашей стране точно). Многие думают что уязвимости типа XSS можно использовать только для кражи cookies. Почему то в любых статьях и рассказах бывают следующие фразы связанные с XSS – "Я нашёл одну XSS но cookies через неё угнать нельзя" или "Я нашёл XSS и угнал cookies администратора". Создаётся впечатление что подобные уязвимости можно использовать только в одном направлении – краже cookies. Как кажется мне – источник этого в том что в большинстве статей приводятся именно такие примеры, поэтому новички полностью верят тому что XSS только для кражи cookies и нужен. Это не так. Что же можно сделать с помощью XSS? На этот вопрос есть огромное количество ответов. Но самый верный, пожалуй, будет следующий – "Всё что может сделать с помощью браузера пользователь".

Фактически, злоумышленник, используя XSS-уязвимость, может делать всё что ему угодно – нажимать на кнопки, проходить по ссылкам, заполнять формы, отправлять их и много чего ещё. Всё ограничивается только фантазией и знаниями взломщика. Но сейчас почему то основная масса людей обитающих на форумах хакерской тематики не хочет ни включать фантазию, ни пополнять знания.

Теперь давайте рассмотрим основные цели, преследуемые взломщиками, при использовании XSS-уязвимостей.

1. Кража cookies или идентификатора сессии.

2. Вывод надписей типа "Hacked by Vasya"

3. Вирусная атака на отдельного пользователя или группу пользователей.

Как и стоило ожидать чаще всего XSS применяется для воровства cookies или перехвата сессии. Причину популярности данной цели мы обсуждали ранее.

Вторая, по популярности, цель (каким бы смешным это не казалось) – шуточные атаки с выводом различных сообщений. Так поступают обычные неопытные новички которым важен сам факт того что они произвели какую-то атаку.

Третья причина – вирусная атака. Но сейчас вредоносный код через подобные уязвимости встраивают в странички только небольшие хакерские группы желающие построить или увеличить свой бот-нет. В реально огромных масштабах подобные заражения производят с помощью кражи паролей к FTP-аккаунтам.

2. Виды XSS.

Теперь обсудим классификацию XSS-уязвимостей. Все XSS-уязвимости по своей природе делятся на 2 основных вида – пассивные и активные. Пассивная XSS – это XSS который сработает только при передаче пользователем определённых данных, то есть при атаке Вы должны каким либо образом заставить пользователя вызвать выполнение вредоносного кода – это является самым большим и основным недостатком уязвимостей данного вида. При использовании пассивных XSS необходимые для атаки данные передаются чаще всего в ссылке (методом GET), либо в поле какой-либо формы (метод POST), что встречается в очень редких случаях.

Естественно заставить пользователя пройти по нужной Вам ссылке намного проще чем заставить его вписать опасный код в форму самому.Здесь так-же нужно обратить внимание на то что более-менее продвинутый в области безопасности пользователь может сразу заподозрить неладное. Ситуация осложняется ещё и тем что подобные атаки в основном бывают нацелены на администраторов уязвимого ресурса. А такие люди точно смогут понять что к чему.

Ещё одно неудобство данного способа в том что ссылки с опасным кодом, выполняющим самые простые действия, получаются очень огромные. Что бы не сильно заморачиваться с этим можно просто создать страничку которая

перебросит жертву по специально сформированной ссылке в которой уже имеется опасный код. Рассмотрим тепер активные (хранимые) XSS-уязвимости. Данный тип XSS представляет из себя уязвимость с помощью которой можно осуществить непосредственную вставку HTML/JS-кода в тело определённой страницы. Данные для использования активной XSS не всегда должны поставляться через какую-либо форму. Из некоторых методов атак, описанных ниже, Вы узнаете что XSS нападение может быть осуществлено, например, через специально сформированное изображение или Flash-анимацию. Основной выигрыш со стороны взломщика здесь в том что ему не нужно заставлять пользователя самого вызывать вредоносный код. Достаточно того что бы пользователь зашёл на определённую страницу (не путайте с переходом по специально сформированной ссылке), где уже имеется опасный код, оставленный взломщиком. Для лучшего понимания представьте следующую ситуацию – например каждый день пользователи заходят на определённый сайт и читают новости. В один прекрасный момент появляется взломщик, оставляет комментарий к определённой новости, содержащий вредоносный код, и все пользователи, просмотревшие эту новость (а следовательно и комментарии к ней), попадают под действие опасного кода. То есть всё пришло к тому что взломщик, после публикации опасного кода, больше в этой атаке на прямую не участвует. Так же подобный тип XSS-уязвимостей хорош тем что с его помощью можно осуществлять массовые нападения, тогда как пассивные XSS-уязвимости используются, в основном, для атак на отдельных пользователей. Как Вы наверное уже догадались - активный XSS, в основном, распространён в гостевых книгах, в различных форумах и блогах. Вобщем там, где пользователь может оставлять то что могут увидеть многие пользователи.

3. Обнаружение XSS-уязвимости.

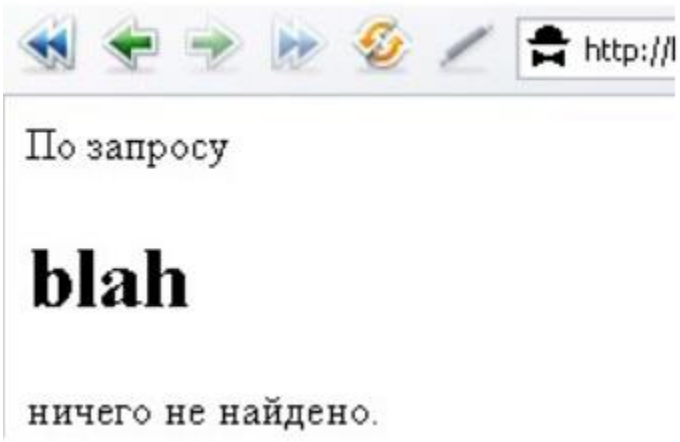
Сейчас мы обсудим основные места веб-приложений которые можно использовать для поиска XSS-уязвимости. Самое главное, при поиске XSS, это искать скрипты которые отображают пришедшие от пользователя данные в чистом виде. Введённая пользователем информация не обязательно должна отображаться на экране, она может находиться в любой части html-кода, главное что бы она ничем не фильтровалась или фильтровалась недостаточно. Вообще, при подобных проверках, лучше сразу смотреть html-код потому что по внешнему виду страницы иногда трудно определить что именно осталось в её коде. Для начала возьмём, конечно же, ссылки. То есть GET-метод передачи данных. Рассмотрим самый простой пример. Возьмём PHP-скрипт следующего содержания:

```
<?php
print "По запросу {$_GET['s']} ничего не найдено.";
?>
```

Сохраним его на хосте localhost под именем test.php. Теперь пройдите по следующей ссылке:

http://localhost/test.php?s=<h1>blah</h1>

В результате Вы увидите следующий ответ скрипта:



Как видите – данные переданные в параметре 's' успешно обработались браузером как html-код. Подобные уязвимости

чаще всего относятся к пассивным XSS. Так же подобные уязвимости могут появиться при создании на сайте печатных

версий каких-либо текстов. Часто программисты для формирования подобных ссылок используют следующий код:

print "Версия для печати";

То есть к скрипту, к которому Вы обращаетесь, добавляются все переданные параметры и добавляется ещё один параметр который указывает скрипту на то что нужно вывести версию для печати. В таких случаях URI по которому Вы обратились вообще не фильтруется и в него можно попытаться вставить html-код. Что бы не повреждать при этом самого URI можно указать опасный код в конце ссылки, в содержимом какого ни будь несущественного парметра. Попробуйте пройти по ссылке

http://localhost/test.php?id=1

и взгляните на получившийся html-код:

Версия для печати

А теперь возьмём код закрытия текущего параметра (адреса ссылки) и тега <a>:

'>

код показа сообщения с цифрами "123":

<script>alert(123)</script>

и код открывающий следующий тег с любым параметром:

<b i='

У нас получается следующая ссылка:

http://localhost/test.php?id=1&i='><script>alert(123)</script><b i='

Пройдя по ней Вы увидите следующее сообщение:

Обратите внимание на то что для перехода по таким ссылками нужно использовать IE. Другие браузеры перекодируют подобные символы в их ASCII-аналоги. Например браузеры Opera и Mozilla превратят эту ссылку вот во что:

http://localhost/test.php?id=1&i='%3E%3Cscript%3Ealert(123)%3C/script%3E%3Cb%20i=' соответсвенно в такой ссылке код не выполнится.

Взглянем на получившийся html-код:

<**script**>**alert(123)**</**script**><**b%20i='**&print=1'>Версия для печати

То что в нём выделено жирным шрифтом это содержимое нашего параметра "i".

Так же, очень часто, XSS-уязвимости существуют в JavaScript-сценариях в которых используется функция eval(). Данная функция выполняет переданный ей текст как JavaScript-код. Практически на 99% сайтов, использующих эту функцию в своих сценариях, данные передающиеся ей не фильтруются. К тому же эта функция используется буквально повсеместно. Рассмотрим небольшой пример – страничку с одним полем и 2 функциями обработки данных. Вот её код:

```
<html>
<head>
<title></title>
</head>
<body>
<form action='?' onSubmit='check(this.field_1.value);'>
<input type='text' name='field_1'>
<input type='submit'>
</form>
<script>
function check(str)
{
    // Здесь расположен какой-либо код
    eval("check_this('"+str+"')");
```

```
}
function check_this(str)
{
        alert(str);
}
</script>
</body>
</html>
```

Как видно из кода - при нажатии на кнопку отправки формы вызывается функция check(), которой передаётся то что Вы

ввели в форме. В ней же выполняется функция eval(), помещённая там просто для примера.

Если в текстовом поле нашей странички набрать слово test и отправить форму то в функции eval() получится

следующий код:

```
check_this('test');
```

А теперь давайте введём вот что:

```
test');alert('123
```

Тогда при отправке формы мы увидим уже 2 сообщения. В функции eval() сформируется следующий код:

```
check_this('test');alert('123');
```

Таким образом мы добились выполнения постороннего кода. Подобный способ атак, связанный с функцией eval(), может быть использован как и в пассивных, так и в активных XSS-уязвимостях. Метод, в принципе, не новый, но почему-то никто не обращает на него внимания. Хотя вероятность обнаружения подобных уязвимостей очень велика как и в обычных сайтах, так и в популярных распространяемых веб-приложениях. Видимо никто не занимается анализом подобных уязвимостей только потому что анализ JavaScript-сценариев не имеет большого

распространения – все стремяться исследовать исходный код самого приложения, а вот про JS-сценарии почему-то забывают. Похожее использование имеют XSS-уязвимости при которых данные, введённые пользователем, вставляются прямо в какой-либо обработчик события, например так:

<img src='...' onClick='код'>

Но такие варианты XSS-уязвимостей не очень распространены.

Так же есть ещё достаточно много узких мест которые могут быть использованы для проведения XSS-нападений

– об этом в следующей версии.