

第一次ActiveX Fuzzing测试

转载

[weixin_30569153](#) 于 2016-03-11 16:34:00 发布 36 收藏

原文链接: <http://www.cnblogs.com/Ox9A82/p/5266255.html>

版权

接着上一篇的看雪Exploit me试题。

这道题给出了一个ActiveX的DLL，挖掘这个DLL中的漏洞。

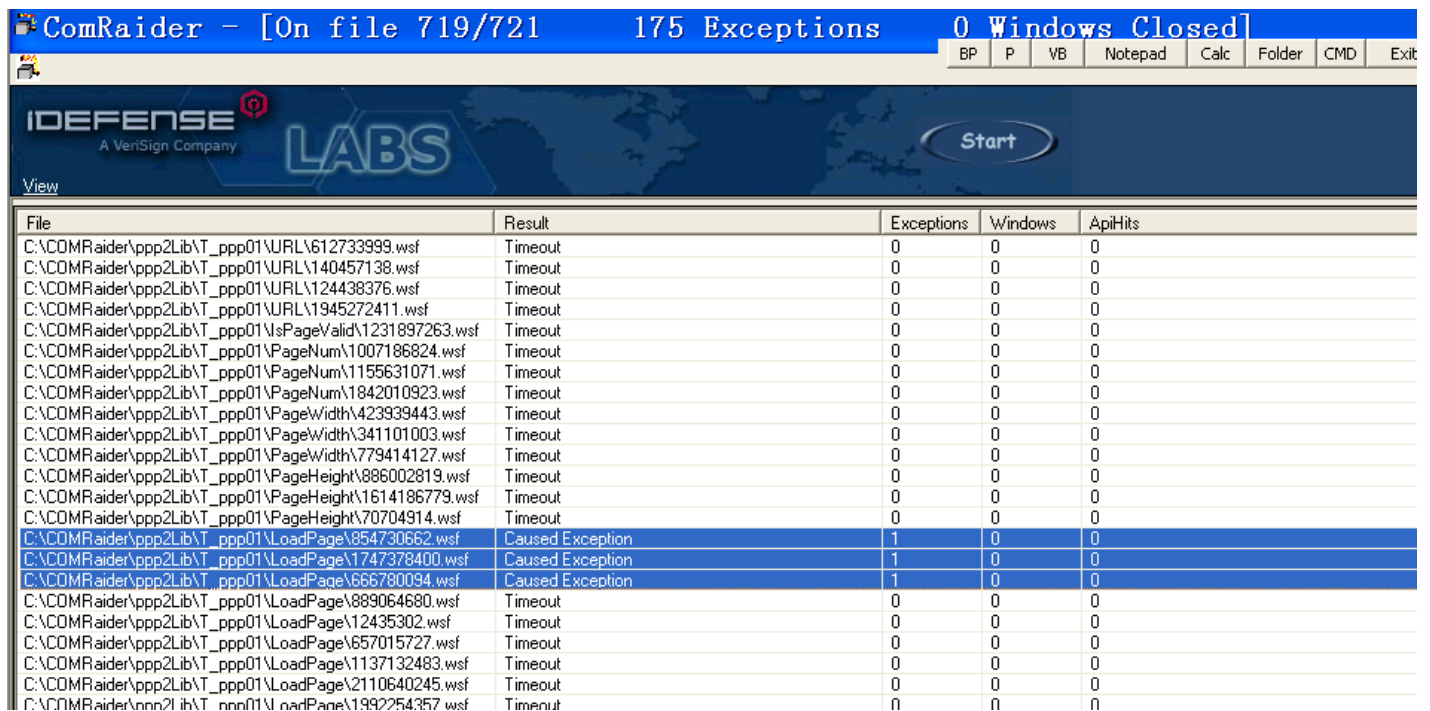
由于从来没有接触过ActiveX的Fuzzing，所以找了一些文章来看。自己动手试验了一下。

根据提示，使用了Comraider来作为Fuzzing工具。这个工具比较老了，找了好久才找到下载地址

<http://down.51cto.com/data/1100082>

根据出题者的意图，应该是先对这个控件进行Fuzzing，然后根据结果进行分析得出漏洞。最后要这个漏洞写出poc，使用堆喷来完成。

实验环境是ie6+xp sp3



The screenshot shows the ComRaider application interface. The title bar reads "ComRaider - [On file 719/721 175 Exceptions 0 Windows Closed]". Below the title bar, there are buttons for "BP", "P", "VB", "Notepad", "Calc", "Folder", "CMD", and "Exit". The main area features the "DEFENSE LABS" logo and a "Start" button. Below this is a table with the following columns: File, Result, Exceptions, Windows, and ApiHits.

File	Result	Exceptions	Windows	ApiHits
C:\COMRaider\ppp2Lib\T_ppp01\URL\612733999.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\URL\140457138.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\URL\124438376.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\URL\1945272411.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\IsPageValid\1231897263.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageNum\1007186824.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageNum\1155631071.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageNum\1842010923.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageWidth\423939443.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageWidth\341101003.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageWidth\779414127.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageHeight\886002819.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageHeight\1614186779.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\PageHeight\70704914.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\854730662.wsf	Caused Exception	1	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\1747378400.wsf	Caused Exception	1	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\686780094.wsf	Caused Exception	1	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\889064680.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\12435302.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\657015727.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\1137132483.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\2110640245.wsf	Timeout	0	0	0
C:\COMRaider\ppp2Lib\T_ppp01\LoadPage\1992254357.wsf	Timeout	0	0	0

根据Fuzzing的结果可知，LoadPage存在漏洞，看了一下测试的payload，是这样的。

```
854730662.wsf - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?XML version='1.0' standalone='yes' ?>
<package><job id='DoneInVBS' debug='false' error='true'>
<object classid='clsid:7F5E27CE-4A5C-11D3-9232-0000B48A05B2' id='target' />
<script language='vbscript'>

'File Generated by COMRaider v0.0.134 - http://labs.idefense.com

'Wscript.echo typename(target)

'for debugging/custom prolog
targetFile = "C:\WINDOWS\system32\syclover.dll"
prototype = "Sub LoadPage ( ByVal URL As String , ByVal x As Long , ByVal y As Long , ByVal Zoom As Single )"
memberName = "LoadPage"
progid = "ppp2Lib.T_ppp01"
argCount = 4

arg1=String(256, "A")
arg2=1
arg3=1
arg4=1

target.LoadPage arg1 ,arg2 ,arg3 ,arg4

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

根据256个A，猜想是个栈溢出。

用od调试这个ActiveX控件，这里费了一些时间。因为从来没有调试过ActiveX控件，不知道怎么定位那些函数，因为那些并不是导出函数。后来发现，原来oleaut32.dll的

DispCallFunc函数会调用这个ActiveX中的函数，也就是只要断在DispCallFunc就可以了。

进入之后发现程序逻辑很复杂，有很多子函数调用，逆向分析一时找不到头绪。于是想到了用OD自动单步执行，来定位是执行到哪一条指令时导致的异常。结果如图

Address	Disassembly	Comment
04573F00	shr ecx,0x2	
04573F03	rep movs dword ptr es:[edi],dword ptr ds:	
04573F05	mov ecx,eax	
04573F07	and ecx,0x3	
04573F0A	xor al,al	
04573F0C	rep movs byte ptr es:[edi],byte ptr ds:	
04573F0E	pop edi	syclover.04572D00
04573F0F	pop esi	syclover.04572D00
04573F10	pop ebx	syclover.04572D00
04573F11	add esp,0x10C	
04573F17	retn 0x8	
04573F1A	pop edi	syclover.04572D00
04573F1B	pop esi	syclover.04572D00
04573F1C	xor al,al	
04573F1E	pop ebx	syclover.04572D00
04573F1F	add esp,0x10C	
04573F25	retn 0x8	
04573F28	nop	
04573F29	nop	
04573F2A	nop	
04573F2B	nop	
04573F2C	nop	
04573F2D	nop	
04573F2E	nop	

可以看到是4573F25处的retn指令导致的异常，此时栈中的返回地址已被覆盖，导致了retn到不可访问的地址，导致了异常。

用ida找到这一段程序后可以发现，这是一个strcmp的误用导致的问题。

```

sub_10013DC0 proc near
var_10C= byte ptr -10Ch
var_108= qword ptr -108h
String= byte ptr -100h
arg_0= dword ptr 4
arg_4= dword ptr 8

sub     esp, 10Ch
mov     edx, ecx
or      ecx, 0FFFFFFFh
xor     eax, eax
push   ebx
push   esi
push   edi
mov     edi, [esp+118h+arg_4]
repne  scasb
not     ecx
sub     edi, ecx
lea    ebx, [esp+118h+String]
mov     eax, ecx
mov     esi, edi
mov     edi, ebx
shr     ecx, 2
rep  movsd
mov     ecx, eax
mov     eax, [edx+3F94h]
and     ecx, 3
cmp     eax, 1
rep  movsb
jnz     loc_10013F1A

```

rep movsd导致了漏洞的发生。

300个字节的数据刚好可以淹没返回地址。

转载于:<https://www.cnblogs.com/Ox9A82/p/5266255.html>



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)