熊猫烧香分析

原创

Stronger_99 于 2019-05-26 11:50:03 发布 2663 收藏 2

在这里主要分析一下熊猫烧香病毒的初始化部分。

病毒文件可以在看雪里下载。

先对熊猫烧香病毒进行一下手动查杀，由于电脑里的软件在查杀病毒时有滞后性，所以学习手动查杀还是很有必要的。

我们先来排查可疑进程。

先把样本放到虚拟机里面，查看任务管理器发现进程数是26

然后运行病毒样本发现任务管理器消失了，再打开也是一闪而过，那么也就说明这个病毒文件已经对电脑产生了影响。

我们可以利用tasklist命令进行查看：

```
C:\Documents and Settings\Administrator>tasklist

图像名                       PID 会话名              会话#       内存使用
========================= ====== ================ ======== ============
System Idle Process            0 Console                    0         28 K
System                         4 Console                    0        296 K
smss.exe                     508 Console                    0        436 K
csrss.exe                    656 Console                    0      5,176 K
winlogon.exe                 680 Console                    0      6,612 K
services.exe                 724 Console                    0      5,252 K
lsass.exe                    736 Console                    0      1,444 K
vmacthlp.exe                 896 Console                    0      2,620 K
svchost.exe                  912 Console                    0      4,984 K
svchost.exe                 1012 Console                    0      4,436 K
svchost.exe                 1108 Console                    0     19,896 K
svchost.exe                 1280 Console                    0      3,620 K
explorer.exe                1452 Console                    0     15,384 K
svchost.exe                 1460 Console                    0      4,508 K
spoolsv.exe                 1628 Console                    0      6,100 K
scardsvr.exe                1668 Console                    0      2,744 K
svchost.exe                 1968 Console                    0      3,352 K
vmtoolsd.exe                 204 Console                    0     11,436 K
rundll32.exe                 936 Console                    0      3,616 K
vmtoolsd.exe                1048 Console                    0     13,580 K
ctfmon.exe                  1344 Console                    0      3,388 K
wuauclt.exe                 1420 Console                    0      5,396 K
spoclsv.exe                  960 Console                    0      4,900 K
conime.exe                  1000 Console                    0      3,188 K
cmd.exe                     3032 Console                    0      2,788 K
tasklist.exe                3292 Console                    0      4,560 K
wmiprvse.exe                3332 Console                    0      5,788 K
                                                  https://blog.csdn.net/Stronger_99
C:\Documents and Settings\Administrator>_
```

通过和上面的对比就可以知道多出了一个名为spoclsv.exe的程序，然后将其删掉就可以了，

```
C:\Documents and Settings\Administrator>taskkill /f /im 960
成功: 已终止 PID 为 960 的进程。
```

这样排查可疑进程就结束了。

下面来检查系统的启动项。在运行命令里面输入msconfig查看启动项：

在这里可以看到spoclsv已经是自启动的一部分了，还可以看到它的路径，在这里将前面那个√删掉，点击确定就可以了。

下面就需要删除病毒了。

通过刚刚的操作已经知道了病毒的位置，利用del /f spoclsv.exe命令将其删除就可以了。

在查看C盘里的内容时明显发现了多出了`autorun.inf`和`setup.exe`，将其删除就可以了：

回到`cmd`查看，在隐藏属性里发现了这两个

将其删除：

手动查杀到这里就结束，之后重启电脑就可以了。

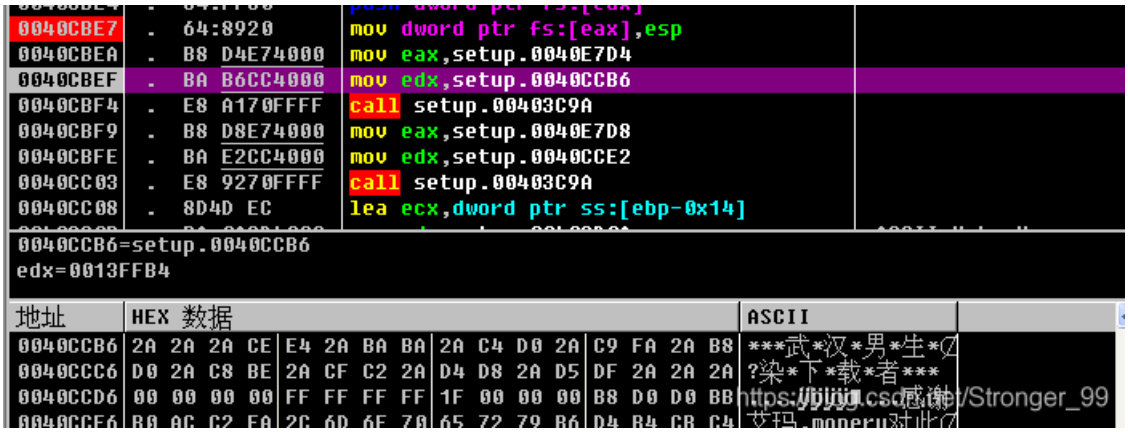
下面进行逆向分析：

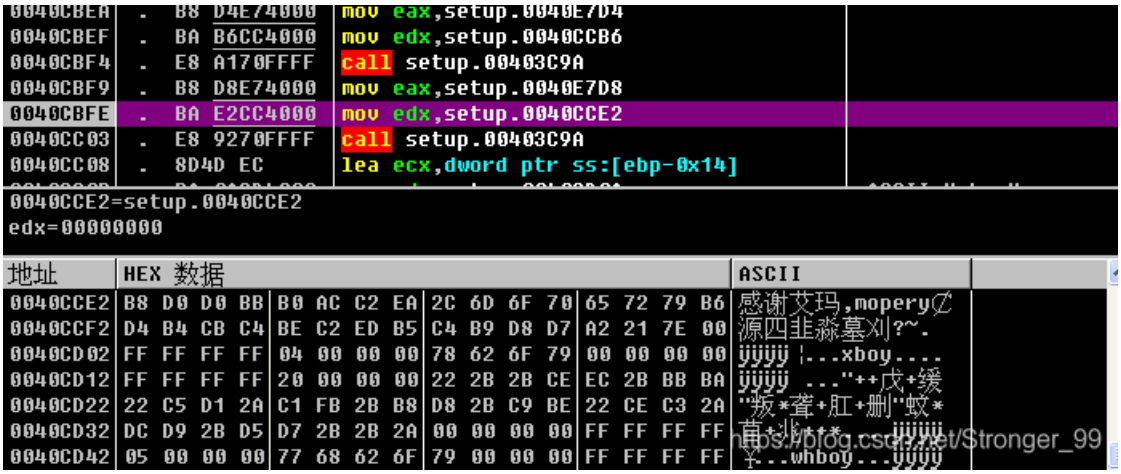
先进行查壳，如果有壳的话需要进行脱壳，通过peid可以看到这个并没有加壳。

使用ida载入病毒文件，首先可以看到接连调用了两个一样的函数

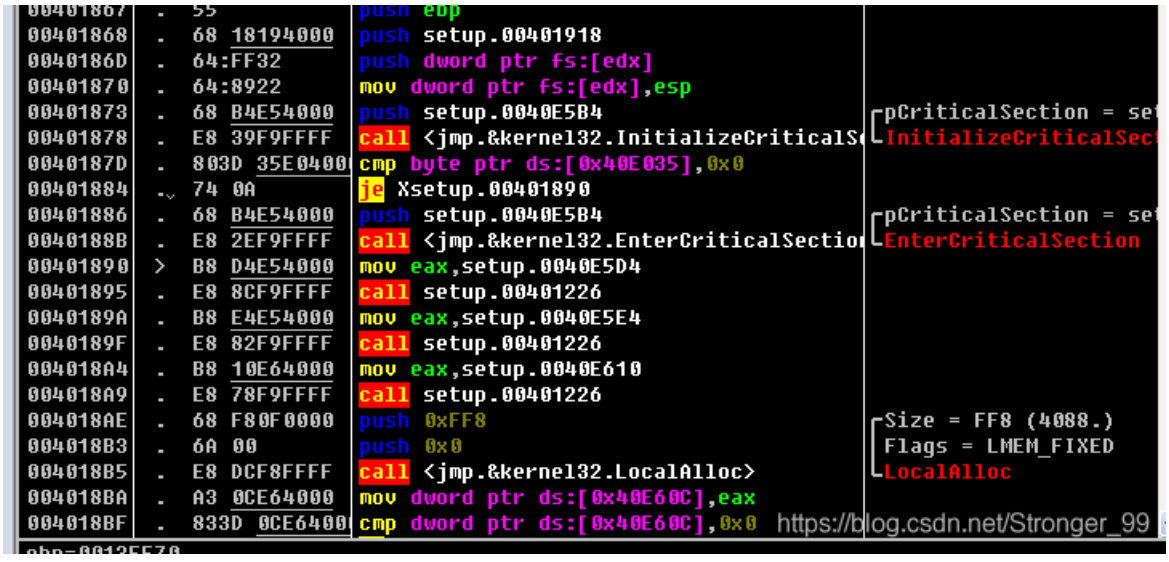
```
E:0040CBE4                push    dword ptr fs:[eax]
E:0040CBE7                mov     fs:[eax], esp
E:0040CBEA                mov     eax, offset dword_40E7D4
E:0040CBEF                mov     edx, offset word_40CCB6
E:0040CBF4                call    sub_403C9A
E:0040CBF9                mov     eax, offset unk_40E7D8
E:0040CBFE                mov     edx, offset word_40CCE2
E:0040CC03                call    sub_403C9A
E:0040CC08                lea     ecx, [ebp-14h]
```

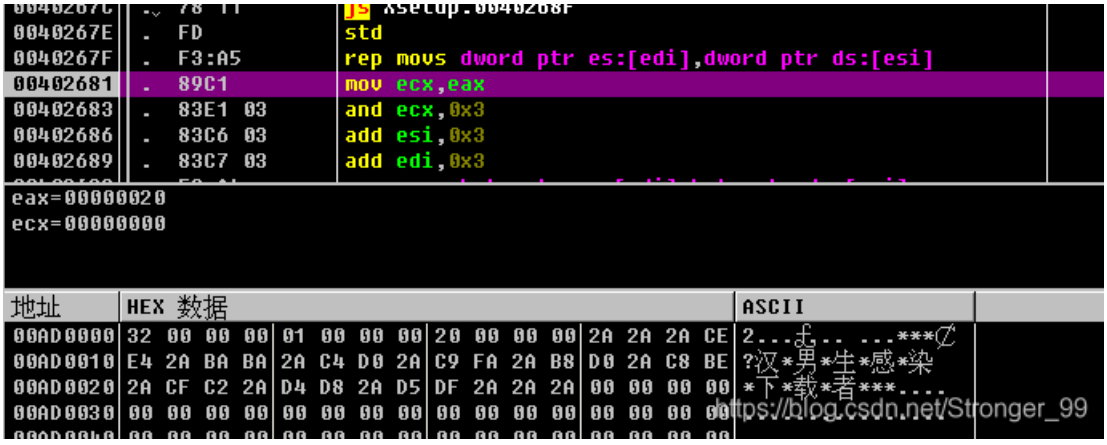
使用OD看到了两个字符串





函数sub_403C9A就是分配一定大小的内存空间和字符串的拷贝

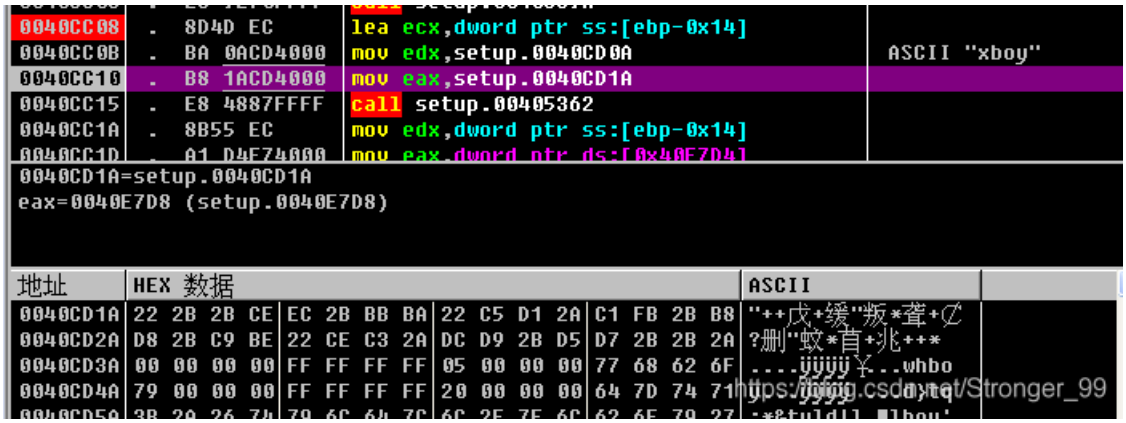




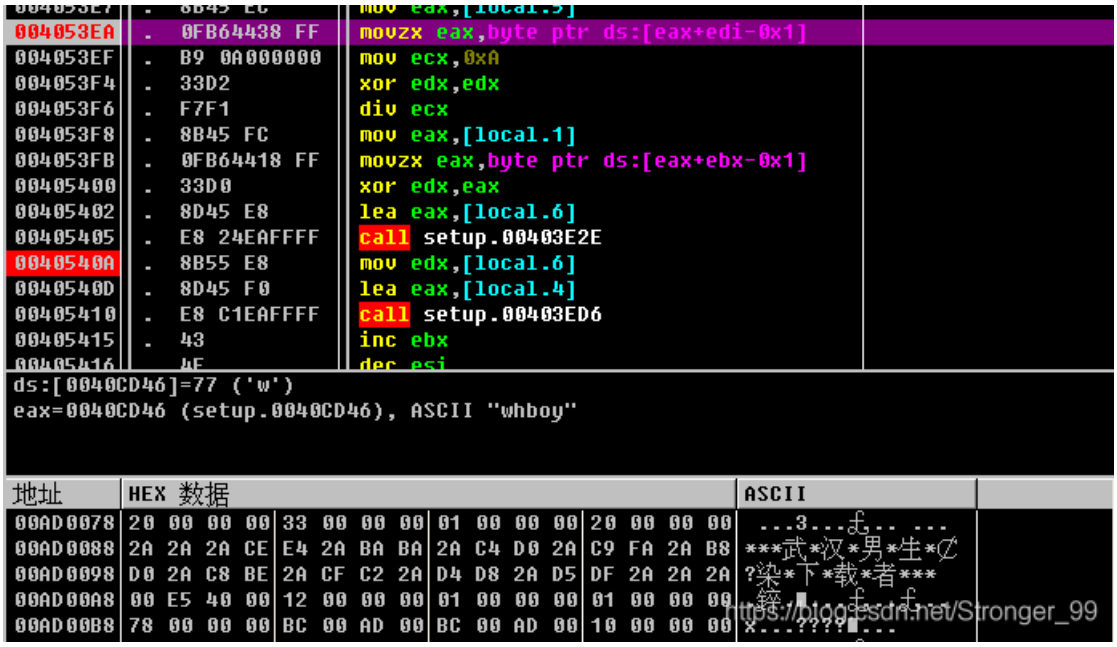
可以将其重命名为allospaceandcopystring。（重命名是为了后面看到是更加清楚明了，具体的名字根据自己，自己明白就好）

继续分析下一个call：

第一个字符串时xboy，第二个是一堆乱码。

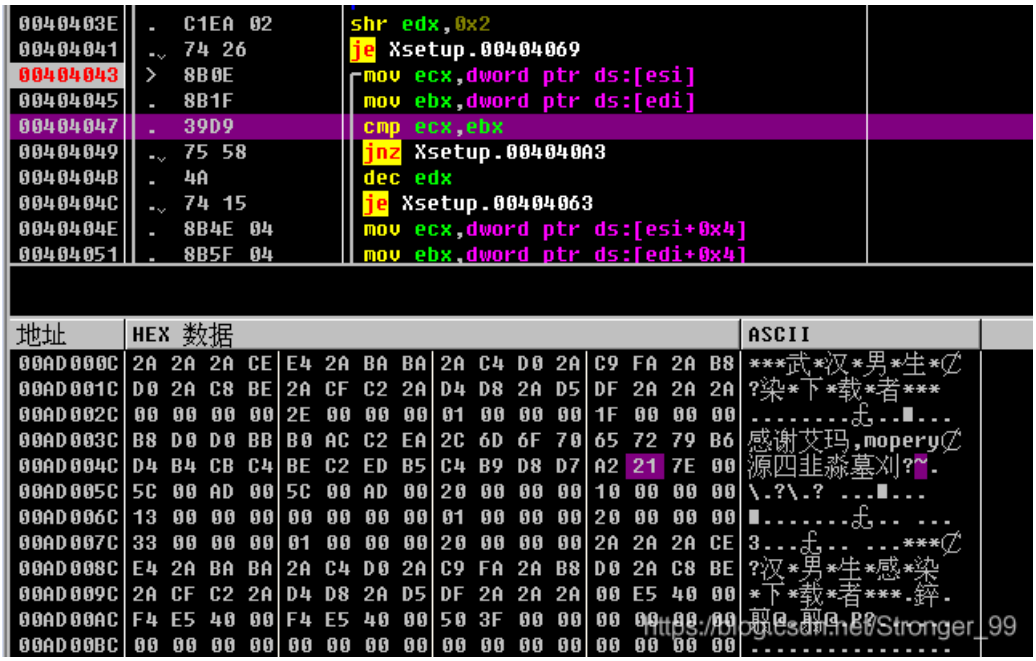


继续分析，看到了一个循环，这里就是进行解码的操作。



将其重命名为decode

函数sub_40401A就是将上面解密后的字符串进行比较



可以看到如果比较成功的话就会跳转到loc_40CC32，继续进行解密和比较的操作

```
.MaskPE:0040CC22                call    sub_40401A
.MaskPE:0040CC27                jz      short loc_40CC32
.MaskPE:0040CC29                push    0
.MaskPE:0040CC2B                call    ExitProcess_0
.MaskPE:0040CC30 ; ---------------------------------------------------------------------------
.MaskPE:0040CC30                jmp     short loc_40CC83
.MaskPE:0040CC32 ; ---------------------------------------------------------------------------
.MaskPE:0040CC32
.MaskPE:0040CC32 loc_40CC32:                            ; CODE XREF: .MaskPE:0040CC27↑j
.MaskPE:0040CC32                lea     ecx, [ebp-18h]
.MaskPE:0040CC35                mov     edx, offset aWhboy ; "whboy"
.MaskPE:0040CC3A                mov     eax, offset word_40CD56
.MaskPE:0040CC3F                call    sub_405362
.MaskPE:0040CC44                mov     edx, [ebp-18h]
.MaskPE:0040CC47                mov     eax, offset word_40CD82
.MaskPE:0040CC4C                call    sub_40401A
.MaskPE:0040CC51                jz      short loc_40CC5C
.MaskPE:0040CC53                push    0
.MaskPE:0040CC55                call    ExitProcess_0
.MaskPE:0040CC5A ; ---------------------------------------------------------------------------
.MaskPE:0040CC5A                jmp     short loc_40CC83
```

可以将其重命名为comstring

然后会跳转到loc_40CC5C：

```
.MaskPE:0040CC5C ; ---------------------------------------------------------------------------
.MaskPE:0040CC5C
.MaskPE:0040CC5C loc_40CC5C:                            ; CODE XREF: .MaskPE:0040CC51↑j
.MaskPE:0040CC5C                call    sub_408042
.MaskPE:0040CC61                call    sub_40CAD2
.MaskPE:0040CC66                call    sub_40C9F2
.MaskPE:0040CC6B                jmp     short loc_40CC73
.MaskPE:0040CC6D ; ---------------------------------------------------------------------------
```

分析第一个call 函数sub_408042

先是进行空间申请：

```
00408042                push    ebp
00408043                mov     ebp, esp
00408045                mov     ecx, 84h
0040804A
0040804A loc_40804A:                            ; CODE XREF: sub_408042+D↓j
0040804A                push    0
0040804C                push    0
```

下面就分析每一个call

函数sub_40277E调用了函数GetModuleFileNameA

```
.MaskPE:00402796                push    105h            ; nSize
.MaskPE:0040279B                lea     eax, [esp+118h+Filename]
.MaskPE:0040279F                push    eax             ; lpFilename
.MaskPE:004027A0                push    0               ; hModule
.MaskPE:004027A2                call    GetModuleFileNameA
.MaskPE:004027A7                mov     ecx, eax
.MaskPE:004027A9                mov     edx, esp
```

利用OD跟随：

```
0040279B  .  8D4424 04      lea eax,dword ptr ss:[esp+0x4]
0040279F  .  50             push eax                        PathBuffer
004027A0  .  6A 00          push 0x0                        hModule = NULL
004027A2  .  E8 5BE9FFFF    call <jmp.&kernel32.GetModuleFileNameA>   GetModuleFileNameA
004027A7  .  8BC8           mov ecx,eax
004027A9  .  8BD4           mov edx,esp
004027AB  .  8BC3           mov eax,ebx
eax=00000036
ecx=7C93003D (ntdll.7C93003D)
地址      HEX 数据                                          ASCII
0013FA3C  43 3A 5C 44 6F 63 75 6D 65 6E 74 73 20 61 6E 64   C:\Documents and
0013FA4C  20 53 65 74 74 69 6E 67 73 5C 41 64 6D 69 6E 69    Settings\Admini
0013FA5C  73 74 72 61 74 6F 72 5C D7 C0 C3 E6 5C 73 65 74   strator\桌面\set
0013FA6C  75 70 2E 65 78 65 00 00 88 FA 13 00 02 00 00 00   up.exe..?.....
```


可以将其重命名为getpathandname

分析sub_405686

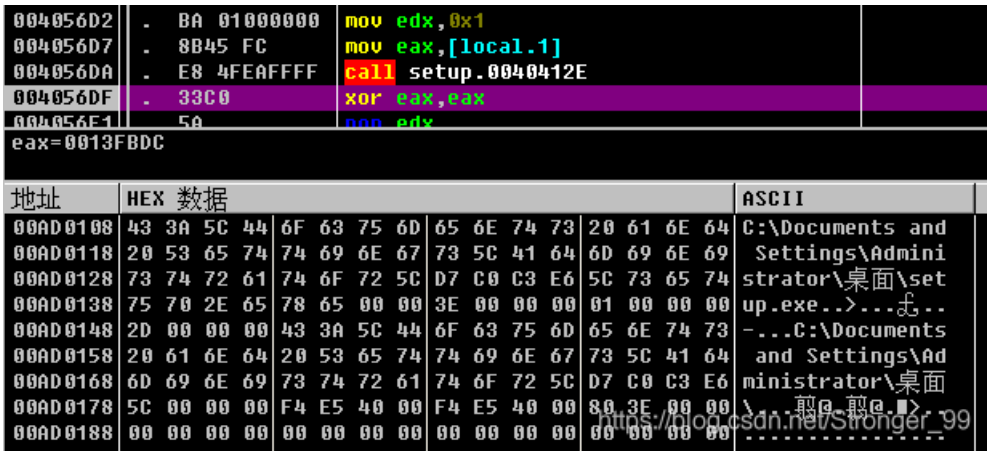
先是对刚刚获取的路径从后往前与 “\” “/” ” :” 进行比较，

```
MaskPE:004056AF                 mov     ebx, eax
MaskPE:004056B1                 cmp     ebx, 1
MaskPE:004056B4                 jl      short loc_4056CF
MaskPE:004056B6
MaskPE:004056B6 loc_4056B6:                             ; CODE XREF: sub_40568E
MaskPE:004056B6                 mov     eax, [ebp+var_4]
MaskPE:004056B9                 mov     al, [eax+ebx-1]
MaskPE:004056BD                 cmp     al, '\'
MaskPE:004056BF                 jz      short loc_4056CF
MaskPE:004056C1                 cmp     al, '/'
MaskPE:004056C3                 jz      short loc_4056CF
MaskPE:004056C5                 cmp     al, ':'
MaskPE:004056C7                 jz      short loc_4056CF
MaskPE:004056C9                 dec     ebx
MaskPE:004056CA                 cmp     ebx, 1
MaskPE:004056CD                 jge     short loc_4056B6
```



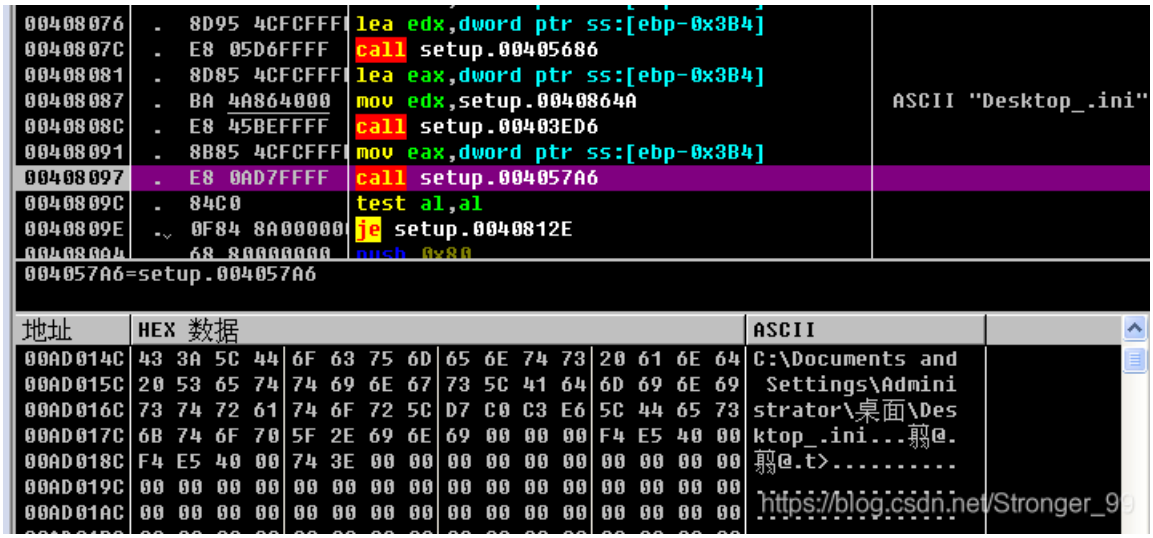
```
004056B4  .v 7C 19          jl Xsetup.004056CF
004056B6  > 8B45 FC         mov eax,[local.1]
004056B9  . 8A4418 FF       mov al,byte ptr ds:[eax+ebx-0x1]
004056BD  . 3C 5C           cmp al,0x5C
004056BF  .v 74 0E          je Xsetup.004056CF
004056C1  . 3C 2F           cmp al,0x2F
004056C3  .v 74 0A          je Xsetup.004056CF
004056C5  . 3C 3A           cmp al,0x3A
004056C7  .v 74 06          je Xsetup.004056CF
004056C9  . 4B              dec ebx
004056CA  . 83FB 01         cmp ebx,0x1
ds:[00AD013D]=65 ('e')
al=08 (Backspace)
地址      HEX 数据                                          ASCII
00AD0108  43 3A 5C 44 6F 63 75 6D 65 6E 74 73 20 61 6E 64   C:\Documents and
00AD0118  20 53 65 74 74 69 6E 67 73 5C 41 64 6D 69 6E 69    Settings\Admini
00AD0128  73 74 72 61 74 6F 72 5C D7 C0 C3 E6 5C 73 65 74   strator\桌面\set
00AD0138  75 70 2E 65 78 65 00 00 F4 E5 40 00 F4 E5 40 00   up.exe..?.....
00AD0148  BC 3E 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ?...............
```


继续向下分析可以知道，函数是为了获取除了文件名以外的路径。



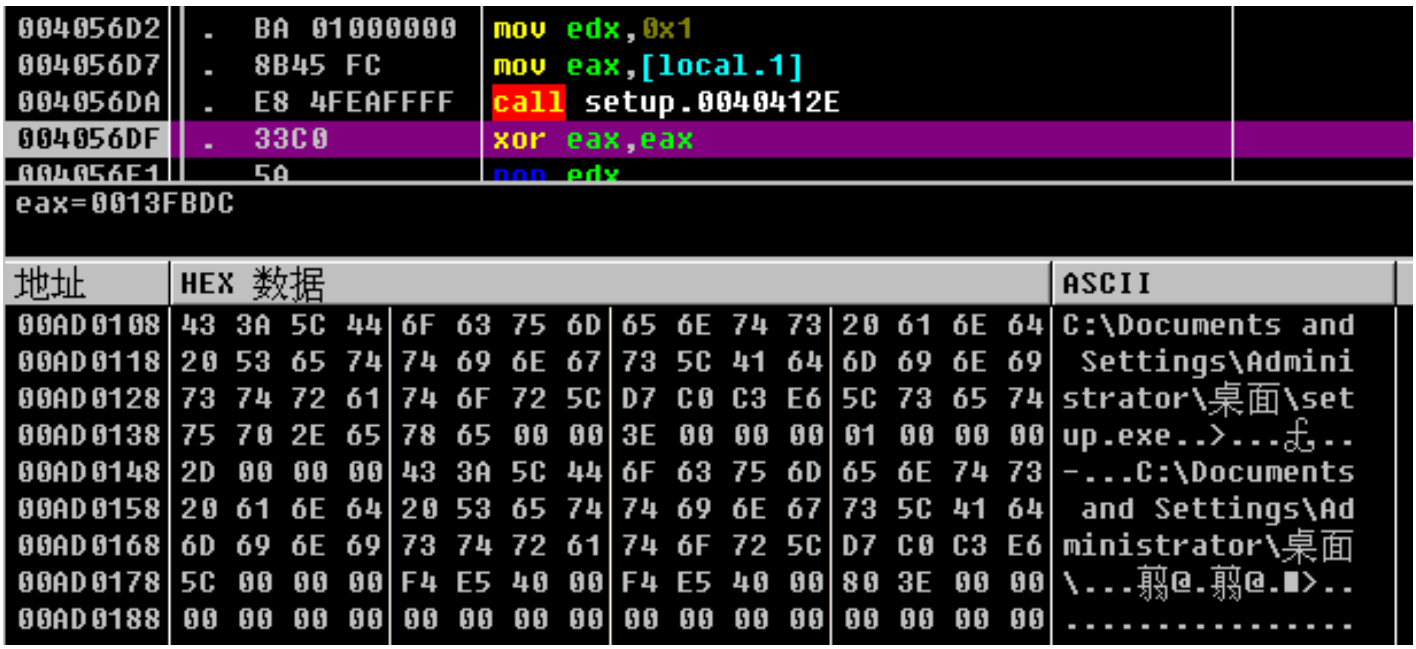
将其重命名为getpath

继续下一个call



这个就是将刚刚去掉文件名的路径与字符串Desktop_.ini结合成一个新的路径。可以将其重命名为stringconnect

函数sub_4057A6的目的就是验证刚刚连接的路径是否存在。



可以将其重命名为`checkfileexist`

由于前面的一些小秘密的操作继续往下看就会很清楚了，如果当前目录下存在Desktop_.ini，就会将其属性改为正常，然后停止一毫秒后对其进行删除

```
.MaskPE:0040809E                jz      loc_40812E
.MaskPE:004080A4                push    80h             ; dwFileAttributes
.MaskPE:004080A9                lea     edx, [ebp+var_3C0]
.MaskPE:004080AF                xor     eax, eax
.MaskPE:004080B1                call    getpathandname
.MaskPE:004080B6                mov     eax, [ebp+var_3C0]
.MaskPE:004080BC                lea     edx, [ebp+var_3BC]
.MaskPE:004080C2                call    getpath
.MaskPE:004080C7                lea     eax, [ebp+var_3BC]
.MaskPE:004080CD                mov     edx, offset aDesktopIni ; "Desktop_.ini"
.MaskPE:004080D2                call    stringconnect
.MaskPE:004080D7                mov     eax, [ebp+var_3BC]
.MaskPE:004080DD                call    sub_4040CE
.MaskPE:004080E2                push    eax             ; lpFileName
.MaskPE:004080E3                call    SetFileAttributesA
.MaskPE:004080E8                push    1               ; dwMilliseconds
.MaskPE:004080EA                call    Sleep
.MaskPE:004080EF                lea     edx, [ebp+var_3C8]
.MaskPE:004080F5                xor     eax, eax
.MaskPE:004080F7                call    getpathandname
.MaskPE:004080FC                mov     eax, [ebp+var_3C8]
.MaskPE:00408102                lea     edx, [ebp+var_3C4]
.MaskPE:00408108                call    getpath
.MaskPE:0040810D                lea     eax, [ebp+var_3C4]
.MaskPE:00408113                mov     edx, offset aDesktopIni ; "Desktop_.ini"
.MaskPE:00408118                call    stringconnect
.MaskPE:0040811D                mov     eax, [ebp+var_3C4]
.MaskPE:00408123                call    sub_4040CE
.MaskPE:00408128                push    eax             ; lpFileName
.MaskPE:00408129                call    DeleteFileA
```


继续分析未知意图的call：

函数sub_4040CE就是完成一个验证，将其重命名为checkpath,

```
laskPE:004040CE                                         ; sub_404622+134p ...
laskPE:004040CE                 test    eax, eax
laskPE:004040D0                 jz      short loc_4040D4
laskPE:004040D2                 retn
laskPE:004040D2 ; ---------------------------------------------------------------------------
laskPE:004040D3 byte_4040D3     db 0                    ; DATA XREF: sub_4040CE:loc_4040D4↓o
laskPE:004040D4 ; ---------------------------------------------------------------------------
laskPE:004040D4
laskPE:004040D4 loc_4040D4:                             ; CODE XREF: sub_4040CE+2↑j
laskPE:004040D4                 mov     eax, offset byte_4040D3
laskPE:004040D9                 retn
laskPE:004040D9 sub_4040CE      endp
```

函数sub_4078E2：

可以观察到在步过sub_4078E2时数据窗口显示了大量的字符，整体来看就是写入病毒信息到内存，将其改名为writevirusinfotomemory。

```
004081DF   .  BA 03000000   mov edx,0x3
004081E4   .  E8 A5BDFFFF   call setup.00403F8E
004081E9   .  8B85 20FCFFFF mov eax,dword ptr ss:[ebp-0x3E0]
004081EF   .  8D95 24FCFFFF lea edx,dword ptr ss:[ebp-0x3DC]
004081F5   .  E8 34D1FFFF   call setup.0040532E
004081FA   .  8B95 24FCFFFF mov edx,dword ptr ss:[ebp-0x3DC]
00408200   .  58            pop eax
00408201   .  E8 14BEFFFF   call setup.0040401A
00408206   .  0F84 02010000 je setup.0040830E
堆栈 ss:[0013FBB0]=00AF7EC4, (ASCII "C:\WINDOWS\system32\drivers\spoclsv.exe")
eax=004081E9 (setup.004081E9)
```

地址	HEX 数据																ASCII
00AF7EC4	43	3A	5C	57	49	4E	44	4F	57	53	5C	73	79	73	74	65	C:\WINDOWS\syste
00AF7ED4	6D	33	32	5C	64	72	69	76	65	72	73	5C	73	70	6F	63	m32\drivers\spoc
00AF7EE4	6C	73	76	2E	65	78	65	00	00	00	00	00	00	00	00	00	lsv.exe.....
00AF7EF4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

sub_403C46我并没有发现它的具体功能是什么，在利用OD动态调试时发现了标志位Z从0变成了1，那么久可以将其重命名为setzerpflag。

sub_403ECE就是获取pe文件的长度，可以将其重命名为getpefilelen。

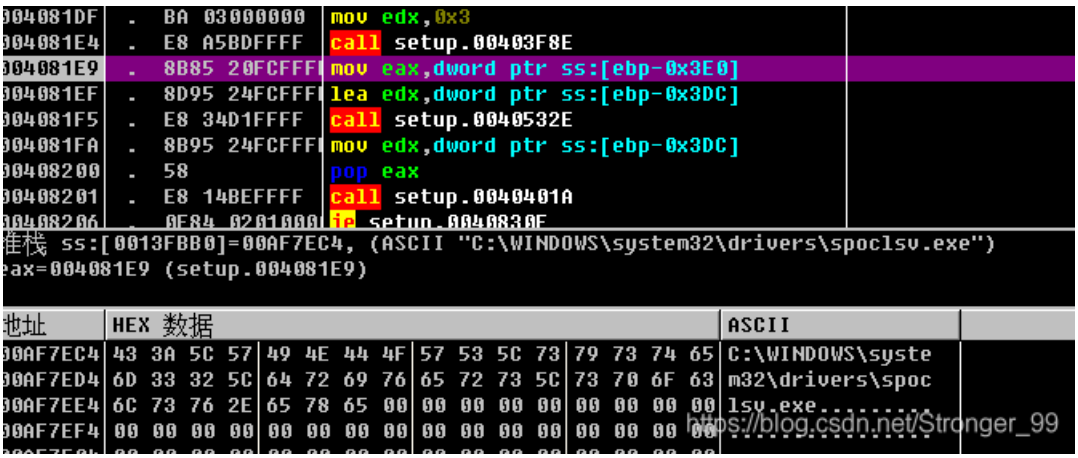
在获取长度后会执行一个跳转jmp　short loc_408181

继续向下分析，这个函数主要就是调用了函数CharUpperBuffA，把缓冲区中指定的字符转换成大写，将其重命名为toupper。

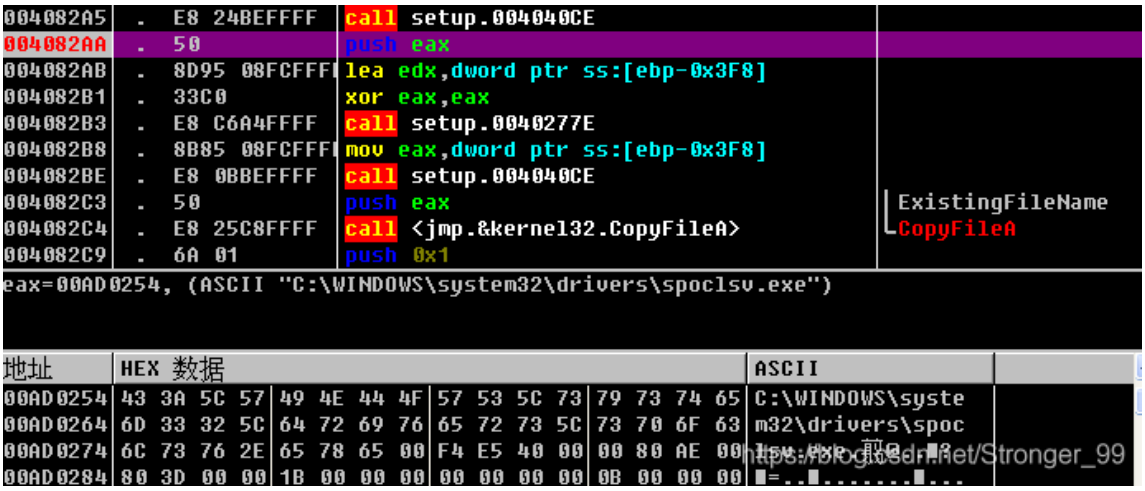
```
.MaskPE:0040532E
.MaskPE:0040532E toupper        proc near               ; CODE XREF: sub_408042
.MaskPE:0040532E                                        ; sub_408042+1B3↓p ...
.MaskPE:0040532E                push    ebx
.MaskPE:0040532F                push    esi
.MaskPE:00405330                push    edi
.MaskPE:00405331                mov     edi, edx
.MaskPE:00405333                mov     esi, eax
.MaskPE:00405335                mov     eax, esi
.MaskPE:00405337                call    getpefilelen
.MaskPE:0040533C                mov     ebx, eax
.MaskPE:0040533E                mov     eax, esi
.MaskPE:00405340                call    checkpath
.MaskPE:00405345                mov     edx, eax
.MaskPE:00405347                mov     eax, edi
.MaskPE:00405349                mov     ecx, ebx
.MaskPE:0040534B                call    sub_403D36
.MaskPE:00405350                test    ebx, ebx
.MaskPE:00405352                jle     short loc_40535D
.MaskPE:00405354                push    ebx             ; cchLength
.MaskPE:00405355                mov     eax, [edi]
.MaskPE:00405357                push    eax             ; lpsz
.MaskPE:00405358                call    CharUpperBuffA
```

下一个call是sub_4054BE，调用了GetSystemDirectoryA，获取系统的路径，可以将其重命名为getsysdir。

通过OD跟随可以知道函数sub_403F8E我主要目的就是将上面获取的路径与两个字符串进行拼接，将其重命名为twostringconnect。



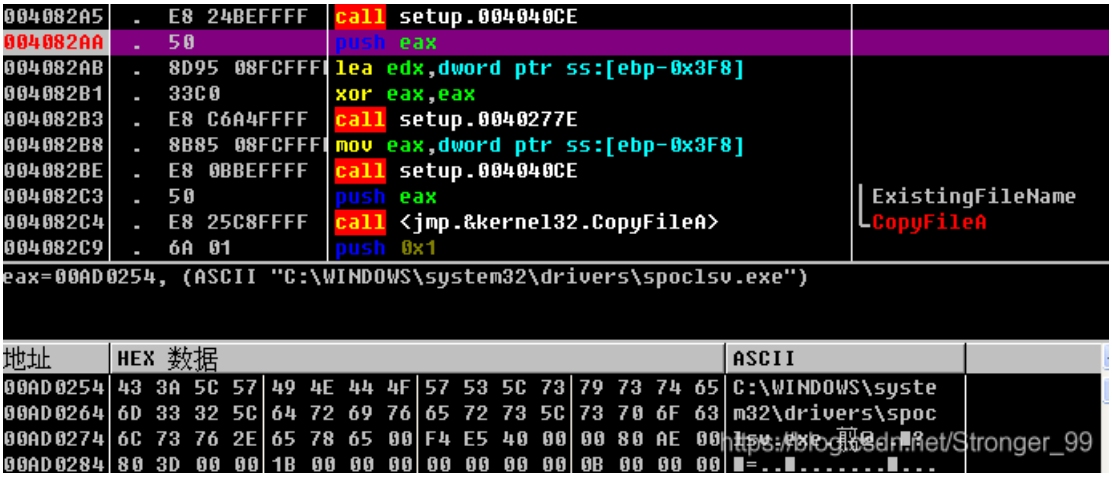
通过分析可以知道函数sub_4060D6目的是查找进程spoclsv.exe，如果就将其结束掉，重命名为searchandterminateprocess



继续向下看就会看到函数是刚刚分析过的，理解其意思就会方便很多，这也就体现了重命名的好处。

下面看到了CopyFileA

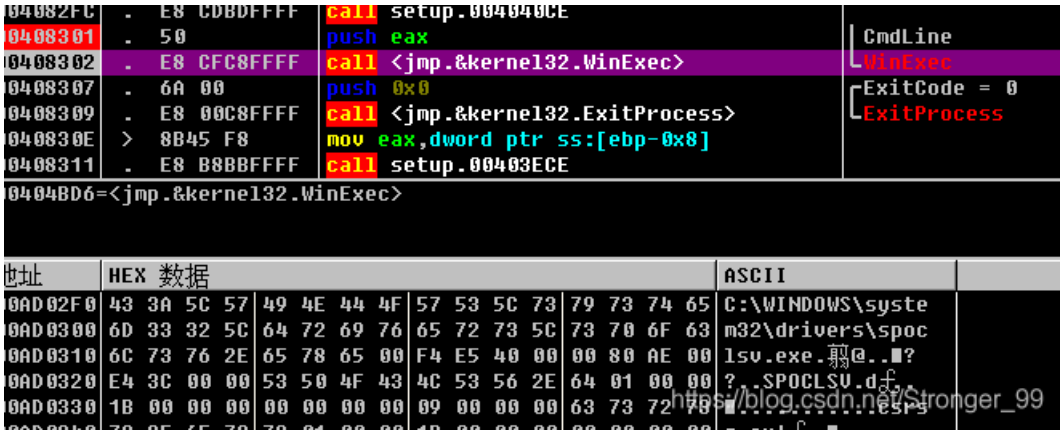
```
.MaskPE:004082A5                call    checkpath
.MaskPE:004082AA                push    eax             ; lpNewFileName
.MaskPE:004082AB                lea     edx, [ebp+var_3F8]
.MaskPE:004082B1                xor     eax, eax
.MaskPE:004082B3                call    getpathandname
.MaskPE:004082B8                mov     eax, [ebp+var_3F8]
.MaskPE:004082BE                call    checkpath
.MaskPE:004082C3                push    eax             ; lpExistingFileName
.MaskPE:004082C4                call    CopyFileA
.MaskPE:004082C9                push    1
```



这里就是将病毒文件改为spoclsv.exe，复制到目录下。

继续向下看到了WinExec，结合OD可以知道现在就是运行程序，然后就退出了。

```
MaskPE:004082F6                mov     eax, [ebp+var_3F8]
MaskPE:004082FC                call    checkpath
MaskPE:00408301                push    eax             ; lpCmdLine
MaskPE:00408302                call    WinExec
MaskPE:00408307                push    0               ; uExitCode
MaskPE:00408309                call    ExitProcess_0
MaskPE:0040830E ; ---------------------------------------------------------------
```



这里就是一个call接着一个call分析，然后弄清楚整体代码的意思，大家在自己分析的时候也不要拘泥一些小的细节。好了，

熊猫烧香病毒的初始化部分分析就到这里了。