

熊猫烧香分析

原创

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

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

本文链接：https://blog.csdn.net/Stronger_99/article/details/90574427

版权

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

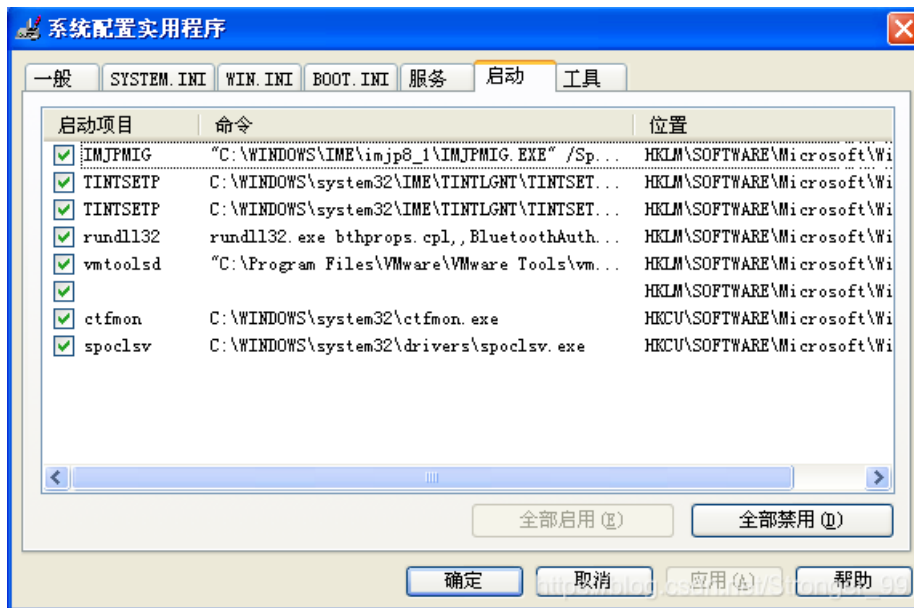
```
C:\Documents and Settings\Administrator>
```

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 spocslv.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

```
C:\>dir /ah
驱动器 C 中的卷没有标签。
卷的序列号是 5C99-7933

C:\> 的目录

2019-05-25 10:20                81 autorun.inf
2019-05-25 10:37                211 boot.ini
2008-04-14 20:00           322,730 bootfont.bin
2015-10-09 14:19                  0 IO.SYS
2015-10-09 14:19                  0 MSDOS.SYS
2008-04-14 20:00           47,564 NTDETECT.COM
2008-04-14 20:00           257,728 ntldr
2019-05-25 09:48       805,306,368 pagefile.sys
2019-05-25 10:21    <DIR>          RECYCLER
2006-12-28 20:04           60,416 setup.exe
2015-10-09 14:30    <DIR>          System Volume Information
          9 个文件      805,995,098 字节
          2 个目录  39,115,202,560 可用字节

C:\> https://blog.csdn.net/Stronger_99
```

将其删除：

```
C:\>del /ah /f autorun.inf
C:\>del /ah /f setup.exe

C:\>dir /ah
驱动器 C 中的卷没有标签。
卷的序列号是 5C99-7933

C:\> 的目录

2019-05-25 10:37                211 boot.ini
2008-04-14 20:00           322,730 bootfont.bin
2015-10-09 14:19                  0 IO.SYS
2015-10-09 14:19                  0 MSDOS.SYS
2008-04-14 20:00           47,564 NTDETECT.COM
2008-04-14 20:00           257,728 ntldr
2019-05-25 09:48       805,306,368 pagefile.sys
2019-05-25 10:21    <DIR>          RECYCLER
2015-10-09 14:30    <DIR>          System Volume Information
          7 个文件      805,934,601 字节
          2 个目录  39,115,264,000 可用字节

C:\> https://blog.csdn.net/Stronger_99
```

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

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

使用OD看到了两个字符串

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。

0040C008	8D4D EC	lea ecx,dword ptr ss:[ebp-0x14]	
0040C00B	BA 0ACD4000	mov edx,setup.0040CD0A	ASCII "xboy"
0040C010	B8 1ACD4000	mov eax,setup.0040CD1A	
0040C015	E8 4887FFFF	call setup.00405362	
0040C01A	8B55 EC	mov edx,dword ptr ss:[ebp-0x14]	
0040C01D	A1 D4F74000	mov eax,dword ptr ds:[0x40F7D4]	
0040CD1A=setup.0040CD1A eax=0040E7D8 (setup.0040E7D8)			
地址	HEX 数据	ASCII	
0040CD1A	22 2B 2B CE EC 2B BB BA	22 C5 D1 2A C1 FB 2B B8	"**+戊+缓"叛*葺+0
0040CD2A	D8 2B C9 BE 22 CE C3 2A	DC D9 2B D5 D7 2B 2B 2A	?删"蚊*自+兆***
0040CD3A	00 00 00 00 FF FF FF FF	05 00 00 00 77 68 62 6F	...uuuuY..whbo
0040CD4A	79 00 00 00 FF FF FF FF	20 00 00 00 64 7D 74 71	https://www.csdn.net/Stronger_99
0040CD50	3B 20 26 74 70 6C 64 70	6C 2E 7E 6C 62 6E 70 27	*#tu1d11_11boy'

继续分析，看到了一个循环，这里就是进行解码的操作。

004053E7	8B45 EC	mov eax,[local.5]	
004053EA	0FB64438 FF	movzx eax,byte ptr ds:[eax+edi-0x1]	
004053EF	B9 0A000000	mov ecx,0xA	
004053F4	33D2	xor edx,edx	
004053F6	F7F1	div ecx	
004053F8	8B45 FC	mov eax,[local.1]	
004053FB	0FB64418 FF	movzx eax,byte ptr ds:[eax+ebx-0x1]	
00405400	33D0	xor edx,eax	
00405402	8D45 E8	lea eax,[local.6]	
00405405	E8 24EAFFFF	call setup.00403E2E	
0040540A	8B55 E8	mov edx,[local.6]	
0040540D	8D45 F0	lea eax,[local.4]	
00405410	E8 C1EAFFFF	call setup.00403ED6	
00405415	43	inc ebx	
00405416	4E	dec esi	
ds:[0040CD46]=77 ('w') eax=0040CD46 (setup.0040CD46), ASCII "whboy"			
地址	HEX 数据	ASCII	
00AD0078	20 00 00 00 33 00 00 00	01 00 00 00 20 00 00 00	...3...武...生
00AD0088	2A 2A 2A CE E4 2A BA BA	2A C4 D0 2A C9 FA 2A B8	***武*汉*男*生*0
00AD0098	D0 2A C8 BE 2A CF C2 2A	D4 D8 2A D5 DF 2A 2A 2A	?染*下*载*者***
00AD00A8	00 E5 40 00 12 00 00 00	01 00 00 00 01 00 00 00	...3...武...生
00AD00B8	78 00 00 00 BC 00 AD 00	BC 00 AD 00 10 00 00 00	https://www.csdn.net/Stronger_99

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

0040403E	-	C1E8 02	shr edx,0x2
00404041	~v	74 26	je Xsetup.00404069
00404043	>	8B0E	mov ecx,dword ptr ds:[esi]
00404045	-	8B1F	mov ebx,dword ptr ds:[edi]
00404047	-	39D9	cmp ecx,ebx
00404049	~v	75 58	jnz Xsetup.004040A3
0040404B	-	4A	dec edx
0040404C	~v	74 15	je Xsetup.00404063
0040404E	-	8B4E 04	mov ecx,dword ptr ds:[esi+0x4]
00404051	-	8B5F 04	mov ebx,dword ptr ds:[edi+0x4]

地址	HEX 数据	ASCII
00AD000C	2A 2A 2A CE E4 2A BA BA 2A C4 D0 2A C9 FA 2A B8	***武*汉*男*生*感
00AD001C	D0 2A C8 BE 2A CF C2 2A D4 D8 2A D5 DF 2A 2A 2A	?染*下*载*者***
00AD002C	00 00 00 00 2E 00 00 00 01 00 00 00 1F 00 00 00志.....
00AD003C	B8 D0 D0 BB B0 AC C2 EA 2C 6D 6F 70 65 72 79 B6	感谢艾玛,mopery感
00AD004C	D4 B4 CB C4 BE C2 ED B5 C4 B9 D8 D7 A2 21 7E 00	源四非森墓刘?~.
00AD005C	5C 00 AD 00 5C 00 AD 00 20 00 00 00 10 00 00 00	\.?\.?
00AD006C	13 00 00 00 00 00 00 00 01 00 00 00 20 00 00 00志.....
00AD007C	33 00 00 00 01 00 00 00 20 00 00 00 2A 2A 2A CE	3...志...***感
00AD008C	E4 2A BA BA 2A C4 D0 2A C9 FA 2A B8 D0 2A C8 BE	?汉*男*生*感*染
00AD009C	2A CF C2 2A D4 D8 2A D5 DF 2A 2A 2A 00 E5 40 00	*下*载*者***. 鏗.
00AD00AC	F4 E5 40 00 F4 E5 40 00 50 3F 00 00 00 00 00 00	http://bbs.csdn.net/Stronger_99
00AD00BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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 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
.MaskPE:0040CC5F

```

可以将其重命名为comstring

然后会跳转到loc_40CC5C:

```

.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 ; -----
.MaskPE:0040CC6D

```

分析第一个call 函数sub_408042

先是进行空间申请:

```

00408042
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:00402798          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.https://blog.csdn.net/Stronger_99

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          loc_4056B6: ; CODE XREF: sub_405686
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_405686
```

004056B4	7C 19	j1 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	74 0E	je Xsetup.004056CF	
004056C1	3C 2F	cmp al, 0x2F	
004056C3	74 0A	je Xsetup.004056CF	
004056C5	3C 3A	cmp al, 0x3A	
004056C7	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.https://blog.csdn.net/Stronger_99
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。

004056D2	. BA 01000000	mov edx,0x1	
004056D7	. 8B45 FC	mov eax,[local.1]	
004056DA	. E8 4FEAFFFF	call setup.0040412E	
004056DF	. 33C0	xor eax,eax	
004056E1	. 50	pop edx	
eax=0013FBDC			
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 3E 00 00 00 01 00 00 00	up.exe.>...>..	
00AD0148	2D 00 00 00 43 3A 5C 44 6F 63 75 6D 65 6E 74 73	...C:\Documents	
00AD0158	20 61 6E 64 20 53 65 74 74 69 6E 67 73 5C 41 64	and Settings\Ad	
00AD0168	6D 69 6E 69 73 74 72 61 74 6F 72 5C D7 C0 C3 E6	ministrator\桌面	
00AD0178	5C 00 00 00 F4 E5 40 00 F4 E5 40 00 80 3E 00 00	\...>..	
00AD0188	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

将其重命名为getpath

继续下一个call

00408076	. 8D95 4CF0FFF	lea edx,dword ptr ss:[ebp-0x3B4]	
0040807C	. E8 05D6FFFF	call setup.00405686	
00408081	. 8D85 4CF0FFF	lea eax,dword ptr ss:[ebp-0x3B4]	
00408087	. BA 4A864000	mov edx,setup.0040864A	ASCII "Desktop_.ini"
0040808C	. E8 45BEFFFF	call setup.00408ED6	
00408091	. 8B85 4CF0FFF	mov eax,dword ptr ss:[ebp-0x3B4]	
00408097	. E8 0AD7FFFF	call setup.004057A6	
0040809C	. 84C0	test al,al	
0040809E	. 0F84 8A000000	jc setup.0040812E	
004080A4	. 68 80000000	push 0x80	
004057A6=setup.004057A6			
地址	HEX 数据	ASCII	
00AD014C	43 3A 5C 44 6F 63 75 6D 65 6E 74 73 20 61 6E 64	C:\Documents and	
00AD015C	20 53 65 74 74 69 6E 67 73 5C 41 64 6D 69 6E 69	Settings\Admini	
00AD016C	73 74 72 61 74 6F 72 5C D7 C0 C3 E6 5C 44 65 73	strator\桌面\Des	
00AD017C	6B 74 6F 70 5F 2E 69 6E 69 00 00 00 F4 E5 40 00	ktop_.ini...>..	
00AD018C	F4 E5 40 00 74 3E 00 00 00 00 00 00 00 00 00 00	>.....	
00AD019C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00AD01AC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	https://blog.csdn.net/Stronger_99	

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。

004056D2	. BA 01000000	mov edx,0x1	
004056D7	. 8B45 FC	mov eax,[local.1]	
004056DA	. E8 4FEAFFFF	call setup.0040412E	
004056DF	. 33C0	xor eax,eax	
004056E1	. 50	pop edx	
eax=0013FBDC			
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 3E 00 00 00 01 00 00 00	up.exe.>...>..	
00AD0148	2D 00 00 00 43 3A 5C 44 6F 63 75 6D 65 6E 74 73	...C:\Documents	
00AD0158	20 61 6E 64 20 53 65 74 74 69 6E 67 73 5C 41 64	and Settings\Ad	
00AD0168	6D 69 6E 69 73 74 72 61 74 6F 72 5C D7 C0 C3 E6	ministrator\桌面	
00AD0178	5C 00 00 00 F4 E5 40 00 F4 E5 40 00 80 3E 00 00	\...>..	
00AD0188	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

可以将其重命名为checkfileexist

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。

```
004081DF  . BA 03000000  mov  edx, 0x3
004081E4  . E8 A5BDFFFF  call setup.00403F8E
004081E9  . 8B85 20FCFFF  mov  eax, dword ptr ss:[ebp-0x3E0]
004081EF  . 8D95 24FCFFF  lea  edx, dword ptr ss:[ebp-0x3DC]
004081F5  . E8 34D1FFFF  call setup.0040532E
004081FA  . 8B95 24FCFFF  mov  edx, dword ptr ss:[ebp-0x3DC]
00408200  . 58           pop  eax
00408201  . E8 14BEFFFF  call setup.0040401A
00408206  . 0E84 0201000  jmp setup.0040830F
挂栈 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\sys
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	lsu.exe.....
00AF7EF4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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

04082FC	- E8 CDBDFFFF	call setup.004040CE	
0408301	- 50	push eax	CmdLine
0408302	- E8 CFC8FFFF	call <jmp.&kernel32.WinExec>	WinExec
0408307	- 6A 00	push 0x0	ExitCode = 0
0408309	- E8 00C8FFFF	call <jmp.&kernel32.ExitProcess>	ExitProcess
040830E	> 8B45 F8	mov eax,dword ptr ss:[ebp-0x8]	
0408311	- E8 B8BBFFFF	call setup.00403ECE	
0404BD6=<jmp.&kernel32.WinExec>			

地址	HEX 数据	ASCII
0AD02F0	43 3A 5C 57 49 4E 44 4F 57 53 5C 73 79 73 74 65	C:\WINDOWS\system
0AD0300	6D 33 32 5C 64 72 69 76 65 72 73 5C 73 70 6F 63	m32\drivers\spoc
0AD0310	6C 73 76 2E 65 78 65 00 F4 E5 40 00 00 80 AE 00	lsu.exe.弱@.!?
0AD0320	E4 3C 00 00 53 50 4F 43 4C 53 56 2E 64 01 00 00	? .SPOCLSU.d.f
0AD0330	1B 00 00 00 00 00 00 00 09 00 00 00 63 73 72 70	https://blog.csdn.net/Stronger_99

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。