纯小白系列(一)之PC病毒分析

原创

Youngs0xFF 于 2017-05-03 10:35:21 发布 1282 收藏
分类专栏： 病毒木马分析 文章标签： 病毒分析 木马分析

目前才开通博客，整理并转移以前写的文章

本文于2015年4月1号发表于看雪，2015年7月22号发表于52pojie。

以前博客：http://youngs-rsr.blog.163.com/

首选先自我介绍一下吧，我在病毒分析方面纯小白！
此篇文章是自己第一次分析病毒，第一次写病毒分析报告！
此分析报告是前几天面试一家安全公司时需要上交的题目！
自己的基础方面:C/C++、逆向等有一定入门基础！所以打算自己边学边写一个病毒分析系列的教程吧，希望对那些和我一样的新手有所帮助吧！如有笔误或者分析有误时，大神们勿喷！

一、 名词解释
母体：指病毒主文件（即2E6682932F826269B0F84A93AAB9E609.85A681D7）
子体：指由母体从资源中释放出来的文件

二、 研究环境及分析工具
系统环境：虚拟机中的XP环境
研究工具：OD 、IDA、procexp（行为分析）、procmon、PEID等
分析人：Youngs

三、 行为分析
1、 用OD加载母体程序，直接按F9运行时，在procexp中可以看到母体释放出一个子体daemonupd.exe格式的程序，并创建进程运行！如图一所示：

图一

2、用PEID给母体查壳，显示为Nullsoft PiMP Stub [Nullsoft PiMP SFX] *，如下图所示

图二

因为不知道这个是什么壳，于是我百度一下，结果百度上给出的是以下回答：
不是特别的壳，只是用Nullsoft Installation System制作的安装程序。Nullsoft Installation System（NSIS）是一个把程序，数据和文档等，用Script 方式，制作成"安装程序"的软件。
只是它的母体只是一个用NSIS来制作的一个exe程序。

四、 调试分析
1、打开ida加载母体程序后，可以看到一些重要的API和CALL，了解整个过程的操作流程。看了其伪代码大概知道是获取临时路径，然后操作文件相关。

2、大致流程路线：

3、用OD加载母体程序，首先是其母体程序的路径，如图三所示:

图三

然后获取完整路径，用"(即22)作为结束符判断，如图四所示：

图四

通过脚本程序（后面会介绍到脚本）获取零时文件：

代码:

```
0040588C  |.  56                |PUSH ESI                                ; /TempName
0040588D  |.  8D45 08           |LEA EAX,DWORD PTR SS:[EBP+0x8]          ; |
00405890  |.  6A 00             |PUSH 0x0                                ; |Unique = 0x0
00405892  |.  50                |PUSH EAX                                ; |Prefix="nsa"(此值是随时变化的)
00405893  |.  FF75 0C           |PUSH DWORD PTR SS:[EBP+0xC]             ; |Path="C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\"
00405896  |.  0055 0A           |ADD BYTE PTR SS:[EBP+0xA],DL            ; |DL=9 此时"nsa"中的a+9=t 从而名字为"nsj"
00405899  |.  FF15 D8704000     |CALL DWORD PTR DS:[<&KERNEL32.GetTempFi>; \GetTempFileNameA
0040589F  |.  85C0              |TEST EAX,EAX
004058A1  |.  75 0D             |JNZ SHORT 2E668293.004058B0
004058A3  |.  85FF              |TEST EDI,EDI
004058A5  |.^ 75 D0             \JNZ SHORT 2E668293.00405877
```

然后我们会发现在C:\Documents and Settings\Administrator\Local Settings\Temp路径下会发现 这样一个零时文件。执行完后回到那个调用CALL的位置：

因为不知道创建这个零时文件有什么作用，然后我们接着继续往下分析，会发现其回继续跳出回来主call，其实通过GetTempPathA就可以猜出是在上面的目录下进行操作的，如下图：

然后接着继续看，当看到下面的时候，我们会发现之前的分析没有啥意义了，个人理解为是在干扰分析用的又或者可以理解为一种壳吧。

代码:

```
004033A8  |. /74 7E             JE SHORT 2E668293.00403428
004033AA  |> |68 00A04200       PUSH 2E668293.0042A000                 ; /FileName="C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\nsj3.tmp"
004033AF  |. |FF15 44714000     CALL DWORD PTR DS:[<&KERNEL32.DeleteFile>; \DeleteFileA
004033B5  |.  56                PUSH ESI                               ; /Arg1
004033B6  |.  E8 B7F8FFFF       CALL 2E668293.00402C72                 ; \在路径C:\Documents and Settings\Administrator\Local Settin
```

刚才分析的文件又重新删除掉了，并又创建了nsd4.tmp方式与之前一样！

跟着我们继续跟入CALL中，然后会发现这样一段，是在 C:\Documents and Settings\Administrator\Local Settings\Application Data\NVIDIA Corporation\Update 目录下创建一下文件夹的方法，由于文件夹为null，所以没出现新文件夹。如下图：

下面这段代码是创建google下update的目录：
代码:

```
00402A0C  |.  83E1 0F       AND ECX,0xF[/size]
[size=3]00402A0F  |.  C1F8 04       SAR EAX,0x4[/size]
[size=3]00402A12  |.  FF348A        PUSH DWORD PTR DS:[EDX+ECX*4]        ; /Arg2[/size]
[size=3]00402A15  |.  C1E0 0A       SHL EAX,0xA                          ; |[/size]
[size=3]00402A18  |.  05 709B4000   ADD EAX,2E668293.00409B70            ; |ASCII "Google\Update"[/size]
[size=3]00402A1D  |.  50            PUSH EAX                             ; |Arg1[/size]
[size=3]00402A1E  |.  E8 65310000   CALL 2E668293.00405B88               ; \创建CALL
```

以下是在 C:\Documents and Settings\Administrator\Local Settings\Application Data\NVIDIA Corporation\Update\ 路径下创建新文件 d83665e11921a3e0525e1d4d9e1d04f1.exe

```
0040583D  /$  FF7424 04     PUSH DWORD PTR SS:[ESP+0x4]          ; /FileName[/size]
[size=3]00405841  |.  FF15 78704000 CALL DWORD PTR DS:[<&KERNEL32.GetFil>; \GetFileAttributesA[/size]
[size=3]00405847  |.  8BC8          MOV ECX,EAX[/size]
[size=3]00405849  |.  6A 00         PUSH 0x0                             ; /hTemplateFile = NULL[/size]
[size=3]0040584B  |.  41            INC ECX                              ; |[/size]
[size=3]0040584C  |.  F7D9          NEG ECX                              ; |[/size]
[size=3]0040584E  |.  1BC9          SBB ECX,ECX                          ; |[/size]
[size=3]00405850  |.  23C8          AND ECX,EAX                          ; |[/size]
[size=3]00405852  |.  51            PUSH ECX                             ; |Attributes=0[/size]
[size=3]00405853  |.  FF7424 14     PUSH DWORD PTR SS:[ESP+0x14]         ; |Mode=CREATE_ALWAYS[/size]
[size=3]00405857  |.  6A 00         PUSH 0x0                             ; |pSecurity = NULL[/size]
[size=3]00405859  |.  6A 01         PUSH 0x1                             ; |ShareMode = FILE_SHARE_READ[/size]
[size=3]0040585B  |.  FF7424 1C     PUSH DWORD PTR SS:[ESP+0x1C]         ; |Access = GENERIC_WRITE[/size]
[size=3]0040585F  |.  FF7424 1C     PUSH DWORD PTR SS:[ESP+0x1C]         ; |FileName="C:\Documents and Settings\Administrator\Local Set
[size=3]00405863  |.  FF15 90704000 CALL DWORD PTR DS:[<&KERNEL32.Create>; \CreateFileA[/size]
[size=3]00405869  \.  C2 0C00       RETN 0xC
```

如下图所示：是创建出来的程序

然后通过调用此call

创建进程，运行它
Call内代码如下：
代码:

```
004053C6  /$  55            PUSH EBP
004053C7  |.  8BEC          MOV EBP,ESP
004053C9  |.  83EC 10       SUB ESP,0x10
004053CC  |.  8D45 F0       LEA EAX,DWORD PTR SS:[EBP-0x10]
004053CF  |.  C705 A8244200>MOV DWORD PTR DS:[0x4224A8],0x44
004053D9  |.  50            PUSH EAX                          ; /pProcessInfo
004053DA  |.  33C0          XOR EAX,EAX                       ; |
004053DC  |.  68 A8244200   PUSH 2E668293.004224A8            ; |pStartupInfo = 2E668293.004224A8
004053E1  |.  50            PUSH EAX                          ; |CurrentDir => NULL
004053E2  |.  50            PUSH EAX                          ; |pEnvironment => NULL
004053E3  |.  50            PUSH EAX                          ; |CreationFlags => 0
004053E4  |.  50            PUSH EAX                          ; |InheritHandles => FALSE
004053E5  |.  50            PUSH EAX                          ; |pThreadSecurity => NULL
004053E6  |.  50            PUSH EAX                          ; |pProcessSecurity => NULL
004053E7  |.  FF75 08       PUSH DWORD PTR SS:[EBP+0x8]       ; |CommandLine
004053EA  |.  50            PUSH EAX                          ; |ModuleFileName => NULL
004053EB  |.  FF15 D0704000 CALL DWORD PTR DS:[<&KERNEL32.Create>; \CreateProcessA
004053F1  |.  85C0          TEST EAX,EAX
004053F3  |.  74 0C         JE SHORT 2E668293.00405401
004053F5  |.  FF75 F4       PUSH DWORD PTR SS:[EBP-0xC]       ; /hObject
004053F8  |.  FF15 EC704000 CALL DWORD PTR DS:[<&KERNEL32.CloseH>; \CloseHandle
004053FE  |.  8B45 F0       MOV EAX,DWORD PTR SS:[EBP-0x10]
00405401  |>  C9            LEAVE
00405402  \.  C2 0400       RETN 0x4
```

运行后的程序如下图：

随后的代码中，直接关闭其句柄了：
代码:

```
00401E87  |> \FF75 08       PUSH DWORD PTR SS:[EBP+0x8]       ; /hObject
00401E8A  |>  FF15 EC704000 CALL DWORD PTR DS:[<&KERNEL32.CloseH>; \CloseHandle
```

接着其创建注册表，操作过程用 可以查看到：

代码:

```
00402328  |.  53            PUSH EBX                          ; /pDisposition
00402329  |.  51            PUSH ECX                          ; |pHandle
0040232A  |.  8B0D 503F4200 MOV ECX,DWORD PTR DS:[0x423F50]   ; |
00402330  |.  83C9 02       OR ECX,0x2                        ; |
00402333  |.  53            PUSH EBX                          ; |pSecurity
00402334  |.  51            PUSH ECX                          ; |Access
00402335  |.  53            PUSH EBX                          ; |Options
00402336  |.  53            PUSH EBX                          ; |Class
00402337  |.  53            PUSH EBX                          ; |Reserved
00402338  |.  50            PUSH EAX                          ; |SubKey="Software\Microsoft\Windows\CurrentVersion\Run"
00402339  |.  57            PUSH EDI                          ; |hKey=HKEY_CURRENY_USER
0040233A  |.  C745 FC 01000>MOV DWORD PTR SS:[EBP-0x4],0x1    ; |
00402341  |.  FF15 20704000 CALL DWORD PTR DS:[<&ADVAPI32.RegCre>; \RegCreateKeyExA
```

紧接着会出现设置注册表的键值
代码:

```
0040238E  |> \50            PUSH EAX                          ; /BufSize=95
0040238F  |.  57            PUSH EDI                          ; |Buffer="C:\Documents and Settings\Administrator\Local Settings\Appl
00402390  |.  FF75 D0       PUSH DWORD PTR SS:[EBP-0x30]      ; |ValueType = REG_SZ
00402393  |.  53            PUSH EBX                          ; |Rserved = 0x0
00402394  |.  FF75 BC       PUSH DWORD PTR SS:[EBP-0x44]      ; |ValueName = "NvUpdService"
00402397  |.  FF75 08       PUSH DWORD PTR SS:[EBP+0x8]       ; |hKey = OxDC
0040239A  |.  FF15 04704000 CALL DWORD PTR DS:[<&ADVAPI32.RegSet>; \RegSetValueExA
```

监控的操作为：

紧接着关闭注册表：
代码：

```
0040247C  |> \57            PUSH EDI                             ; /hKey
0040247D  |>  FF15 1C704000 CALL DWORD PTR DS:[<&ADVAPI32.RegClo>; \RegCloseKey
```

继续跟进,会发现在 C:\Documents and Settings\Administrator\Local Settings\Application Data\Google\Update中创建一个空目录，

代码：

[Asm] 纯文本查看 复制代码

```
004015D2  |.  53            |
PUSH
EBX
; /pSecurity=0


004015D3  |.  57            |
PUSH
EDI
; |Path = "C:\Documents and Settings\Administrator\Local Settings\Application Data\Google\Update"


004015D4  |.  8A06          |
MOV
AL
,
BYTE
PTR
DS
:[
ESI
]
; |


004015D6  |.  881E          |
MOV
BYTE
PTR
 DS
:[
ESI
],
BL
; |
```

然后调用 SHFileOperationA函数来复制d83665e11921a3e0525e1d4d9e1d04f1.exe并重命名为 gupdate.exe：

代码：

[Asm] 纯文本查看 复制代码

```
004021CC  |.  50
PUSH
EAX
; /Arg2


004021CD  |.  53
PUSH
EBX
; |Arg1


004021CE  |.  8975 A4
MOV
DWORD
PTR
 SS
:[
EBP
-0x5C],
ESI
; |


004021D1  |.  897D A8
MOV
DWORD
PTR
 SS
:[
EBP
-0x58],
EDI
; |
```

紧接着给gupdate.exe创建注册表，代码如下：
代码:

[Asm] 纯文本查看 复制代码

```
00402328  |.  53
PUSH
EBX
; /pDisposition


00402329  |.  51
PUSH
ECX
; |pHandle


0040232A  |.  8B0D 503F4200
MOV
ECX
,
DWORD
PTR
DS
:[0x423F50]
; |


00402330  |.  83C9 02
OR
ECX
,0x2
; |


00402333  |.  53
PUSH
EBX
; |pSecurity
```

接着设置其注册表的键值，代码如下：
代码:

[Asm] 纯文本查看 复制代码

```
0040238E  |> \50
PUSH
EAX
; /BufSize=95

0040238F  |.  57
PUSH
EDI
; |Buffer="C:\Documents and Settings\Administrator\Local Settings\Application Data\Google\Update\gupdate.exe /app
2B42CDC8B1EDBFEC23AA442F8F7EF3D9"

00402390  |.  FF75 D0
PUSH
DWORD
PTR
 SS
:[
EBP
-0x30]
; |ValueType = REG_SZ

00402393  |.  53
PUSH
EBX
; |Rserved = 0x0

00402394  |.  FF75 BC
PUSH
DWORD
PTR
 SS
```

接下来就是关闭注册表：
代码:

[Asm] 纯文本查看 复制代码

```
0040247C  |> \57
PUSH
EDI
; /hKey

0040247D  |>  FF15 1C704000
CALL
DWORD
PTR
 DS
:[<&ADVAPI32.RegClo>; \RegCloseKey
```

更换了目录，对C:\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows目录操作
代码:

[Asm] 纯文本查看 复制代码

```
004015D2  |.  53            |
PUSH
EBX
1           ; /pSecurity


004015D3  |.  57            |
PUSH
EDI
; |Path = "C:\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows"
2


004015D4  |.  8A06          |
MOV
AL
,
BYTE
PTR
DS
3           :[
ESI
]
; |


004015D6  |.  881E          |
MOV
4           BYTE
PTR
 DS
:[
ESI
],
BL
; |
```

调用SHFileOperationA函数来复制 d83665e11921a3e0525e1d4d9e1d04f1.exe并重命名为 winupdate.exe：
代码:

[Asm] 纯文本查看 复制代码

```
004021CC  |.  50
PUSH
EAX
01          ; /Arg2


004021CD  |.  53
PUSH
EBX
; |Arg1
02


004021CE  |.  8975 A4
MOV
DWORD
PTR
 SS
:[
EBP
03          -0x5C],
ESI
; |


004021D1  |.  897D A8
MOV
DWORD
04          PTR
 SS
:[
EBP
-0x58],
EDI
; |
```

把其中的一个字符串装换为宽字符：
代码:

[Asm] 纯文本查看 复制代码

```
0040211B  |.  68 00040000
PUSH
0x400
; /WideBufSize = 400 (1024.)


00402120  |.  56
PUSH
ESI
; |WideCharBuf => 2E668293.00409368


00402121  |.  6A FF
PUSH
-0x1
; |StringSize = FFFFFFFF (-1.)


00402123  |.  FF75 D0
PUSH
DWORD
PTR
 SS
:[
EBP
-0x30]
; |StringToMap = "C:\Documents and Settings\Administrator\「开始」菜单\程序\启动\winupdate.lnk"


00402126  |.  BF 05400080
MOV
EDI
,0x80004005
; |
```

紧接着去搜索C:\Documents and Settings\Administrator\Local Settings\Application Data\NVIDIA Corporation\Update\daemonupd.exe"

代码:

[Asm] 纯文本查看 复制代码

```
00405527  |.  50
PUSH
EAX
; /pFindFileData


00405528  |.  56
PUSH
ESI
; |FileName="C:\Documents and Settings\Administrator\Local Settings\Application Data\NVIDIA
Corporation\Update\daemonupd.exe"


00405529  |.  03DF
ADD
EBX
,
EDI
; |


0040552B  |.  FF15 40714000
CALL
DWORD
PTR
 DS
:[<&KERNEL32.FindFi>; \FindFirstFileA
```

然后就自己调用MoveFileA函数来重命名程序了,

代码:

[Asm] 纯文本查看 复制代码

```
            0040165E  |.  57
            PUSH
            EDI
1           ; /NewName = "C:\Documents and Settings\Administrator\Local Settings\Application Data\NVIDIA
Corporation\Update\daemonupd.exe"


            0040165F  |.  56
            PUSH
            ESI
            ; |ExistingName = "C:\Documents and Settings\Administrator\Local Settings\Application Data\NVIDIA
2   Corporation\Update\d83665e11921a3e0525e1d4d9e1d04f1.exe"


            00401660  |.  FF15 70704000
            CALL
            DWORD
            PTR
             DS
3           :[<&KERNEL32.MoveFi>; \MoveFileA
```

执行完上面代码后,则出现下图情况：

继续跟入，我们会发现，程序创建进程直接运行起 daemonupd.exe

代码:

[Asm] 纯文本查看 复制代码

```
            004053D9  |.  50
            PUSH
            EAX
01          ; /pProcessInfo


            004053DA  |.  33C0
            XOR
            EAX
            ,
            EAX
02          ; |


            004053DC  |.  68 A8244200
            PUSH
            2E668293.004224A8
            ; |pStartupInfo = 2E668293.004224A8
03


            004053E1  |.  50
            PUSH
            EAX
            ; |CurrentDir => NULL


04          004053E2  |.  50
            PUSH
            EAX
            ; |pEnvironment => NULL


            004053E3  |.  50
            PUSH
```

关闭句柄，代码如下：
代码:

[Asm] 纯文本查看 复制代码

```
00402753  |> \FF75 08
PUSH
DWORD
PTR
 SS
:[
EBP
+0x8]
; /hObject


00402756  |.  FF15 EC704000
CALL
DWORD
PTR
 DS
:[<&KERNEL32.CloseH>; \CloseHandle
```

继续搜索以前的原有程序 d83665e11921a3e0525e1d4d9e1d04f1.exe是否存在

代码:

[Asm] 纯文本查看 复制代码

```
00405521  |.  8D85 B8FEFFFF
LEA
EAX
,
DWORD
PTR
SS
:[
EBP
-0x148]


00405527  |.  50
PUSH
EAX
; /pFindFileData


00405528  |.  56
PUSH
ESI
; |FileName="C:\Documents and Settings\Administrator\Local Settings\Application Data\NVIDIA
Corporation\Update\d83665e11921a3e0525e1d4d9e1d04f1.exe"


00405529  |.  03DF
ADD
EBX
,
EDI
; |


0040552B  |.  FF15 40714000
CALL
```

继续检查2E6682932F826269B0F84A93AAB9E609.85A681D7是否存在，代用代码和上面一样
接着调用DeleteFileA来删除母体程序，由于我用OD已经加载了，所以删除失败，正常是可以自己删除母体程序的

代码:

[Asm] 纯文本查看 复制代码

```
004055A0  |.  57           |
PUSH
EDI
; /FileName


004055A1  |.  FF15 44714000 |
CALL
DWORD
PTR
 DS
:[<&KERNEL32.Delet>; \DeleteFileA
```

后面的代码都是对2B42CDC8B1EDBFEC23AA442F8F7EF3D9操作的，搜索目录、终止进程、关闭句柄，因为我用OD加载了 所以无法对其操作！
其代码如下：
代码:

[Asm] 纯文本查看 复制代码

```
004055D3  |> \8D85 B8FEFFFF |LEA EAX,DWORD PTR SS:[EBP-0x148]
004055D9  |.  50            |PUSH EAX                                  ; /pFindFileData
004055DA  |.  FF75 08       |PUSH DWORD PTR SS:[EBP+0x8]               ; |hFile
004055DD  |.  FF15 3C714000 |CALL DWORD PTR DS:[<&KERNEL32.FindN>]     ; \FindNextFileA
```

五、 继续分析daemonup.exe文件
用OD加载的时候会显示数据被压缩过，可能是加了某种壳吧！打乱了所有的顺序，用"删除模块分析"都没什么反应，由于水平有限暂到这里吧
我们会发现无法正常双击去运行它，然后我又重新让母体程序运行起来，用Process Monitor会发现如下的情况

应该是一个远控木马程序，因为它不停的发包和接包！

六、 带着好奇心打开NSIS文件
1、弄玩上面的，又百度了一下NSIS，在卡饭论坛上看到有这样的回答，说是可以用7z-zip解压一些文件，于是在虚拟机中我用7z解压打开了母体程序。我一个朋友也解压出来了，然后也跟着试试！看到如下信息：

暂时百度后只知道nsis是执行脚本。
结合图一和图6，我们可以看到图6中的exe在图一中都没显示过，初步估计是操作很快，人眼无法识别出来，但是在我们动态调试的时候，显示有！
2、以文本文档的方式打开[NSIS].nsi文件，我们会发现正如百度知道所说，这里面全部是一些操作文件、注册表等的代码，部分代码如下：
代码:

[Asm] 纯文本查看 复制代码

```
Exec "$\"$INSTDIR\daemonupd.exe$\" /exit 2B42CDC8B1EDBFEC23AA442F8F7EF3D9"

DeleteRegValue HKCU Software\Microsoft\Windows\CurrentVersion\Run $_0_

DeleteRegValue HKCU Software\Microsoft\Windows\CurrentVersion\Run $_4_

Sleep 1000

Delete $INSTDIR\daemonupd.exe

Delete $SMSTARTUP\$_2_.lnk

Delete $LOCALAPPDATA\Microsoft\Windows\$_2_.exe

Delete $LOCALAPPDATA\$_1_\$_3_.exe
```

因为之前用OD调试过，会发现母体程序基本都是按照此代码来操作注册表和文件等的
好了，分析报告就到这里吧！
因为自己没有分析病毒的经验，也就是自己一步一步的跟进去找到相关的call去理解的！
第一次写这样的文档，也不是文科生，叙述没有条理，有些混乱！大家凑合看吧！有错误的地方也希望大家能指正，我也能多学习学习！
以后我会边工作边研究，也会把自己分析过的一些病毒分析出来，希望自己能对和我一样是纯小白的人有点帮助吧！