windows样本高级静态分析之识别汇编中C代码结构(switch)

yong夜 / 2019-10-14 10:13:20 / 浏览数 3422 安全技术 二进制安全 顶(0) 踩(0)

目标

通过分析代码结构来理解一个恶意样本的总体功能。

本篇主要通过分析样本了解switch语句

分析流程

1.基础静态分析

2.基础动态分析

3.高级静态分析

实践过程

实例1

Lab06-03.exe

基础静态分析

导入函数

```
InternetOpenUrlA
InternetCloseHandle
InternetReadFile
InternetGetConnectedState
InternetOpenA
RegSetValueExA
RegOpenKeyExA
CreateDirectoryA
CopyFileA
DeleteFileA
GetFileType
WriteFile
```

字符串

```
http://www.practicalmalwareanalysis.com/cc.htm
Software\Microsoft\Windows\CurrentVersion\Run
C:\Temp\cc.exe
C:\Temp
Error 1.1: No Internet
Success: Internet Connection
Error 2.3: Fail to get command
Error 2.2: Fail to ReadFile
Error 2.1: Fail to OpenUrl
Internet Explorer 7.5/pma
Error 3.2: Not a valid command provided
Error 3.1: Could not set Registry value
Malware
Success: Parsed command is %c
```

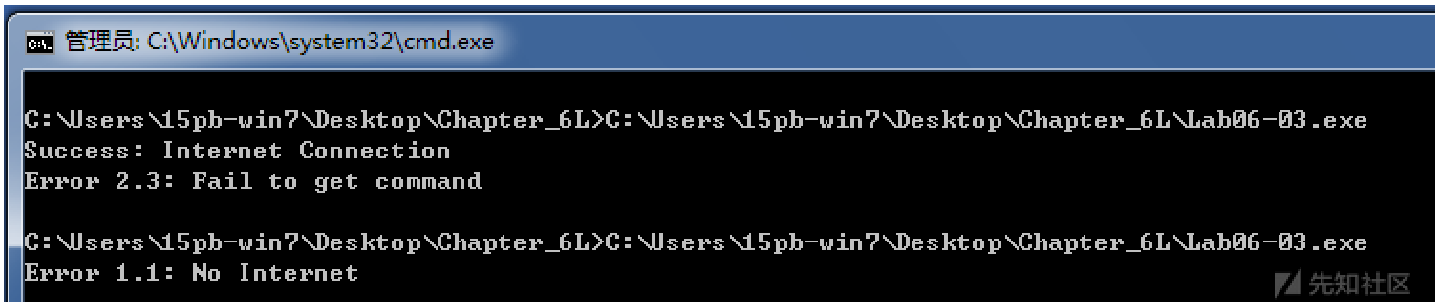
根据api和字符串可以判断：

1.存在联网访问 http://www.practicalmalwareanalysis.com/cc.htm 网址操作并且通过字符串中的错信息可以判断可能存在解析网页来获取命令来执行

2.写注册表来是实现自启动

3.产生衍生文件C:\Temp\cc.exe

基础动态分析



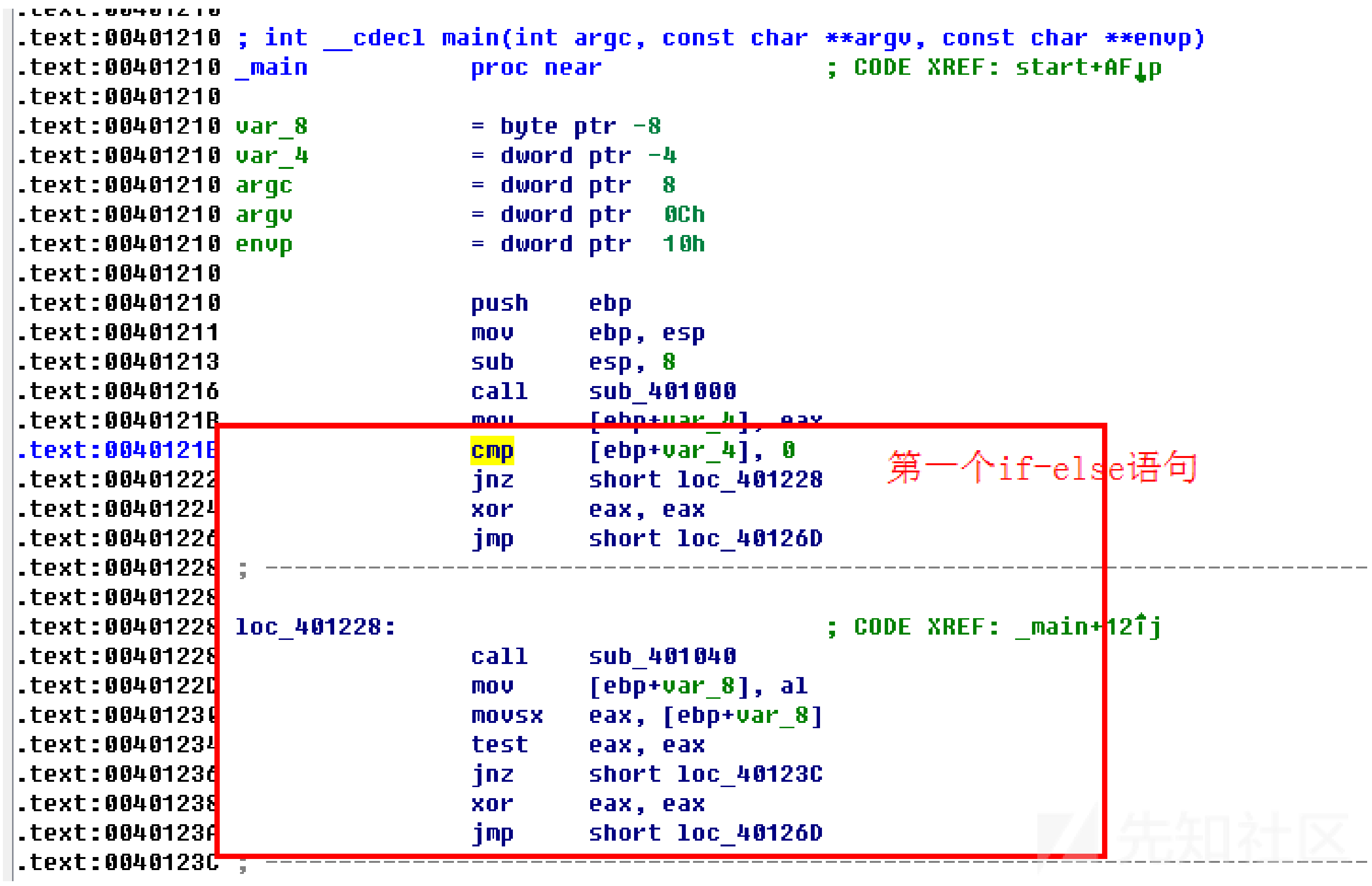
```
C:\Users\15pb-win7\Desktop\Chapter_6L>C:\Users\15pb-win7\Desktop\Chapter_6L\Lab06-03.exe
Success: Internet Connection
Error 2.3: Fail to get command

C:\Users\15pb-win7\Desktop\Chapter_6L>C:\Users\15pb-win7\Desktop\Chapter_6L\Lab06-03.exe
Error 1.1: No Internet
```

和之前分析一样，根据不同网络状态返回打印内容，接着通过高级静态分析来看程序后续操作

高级静态分析

直接跟如main函数进行分析



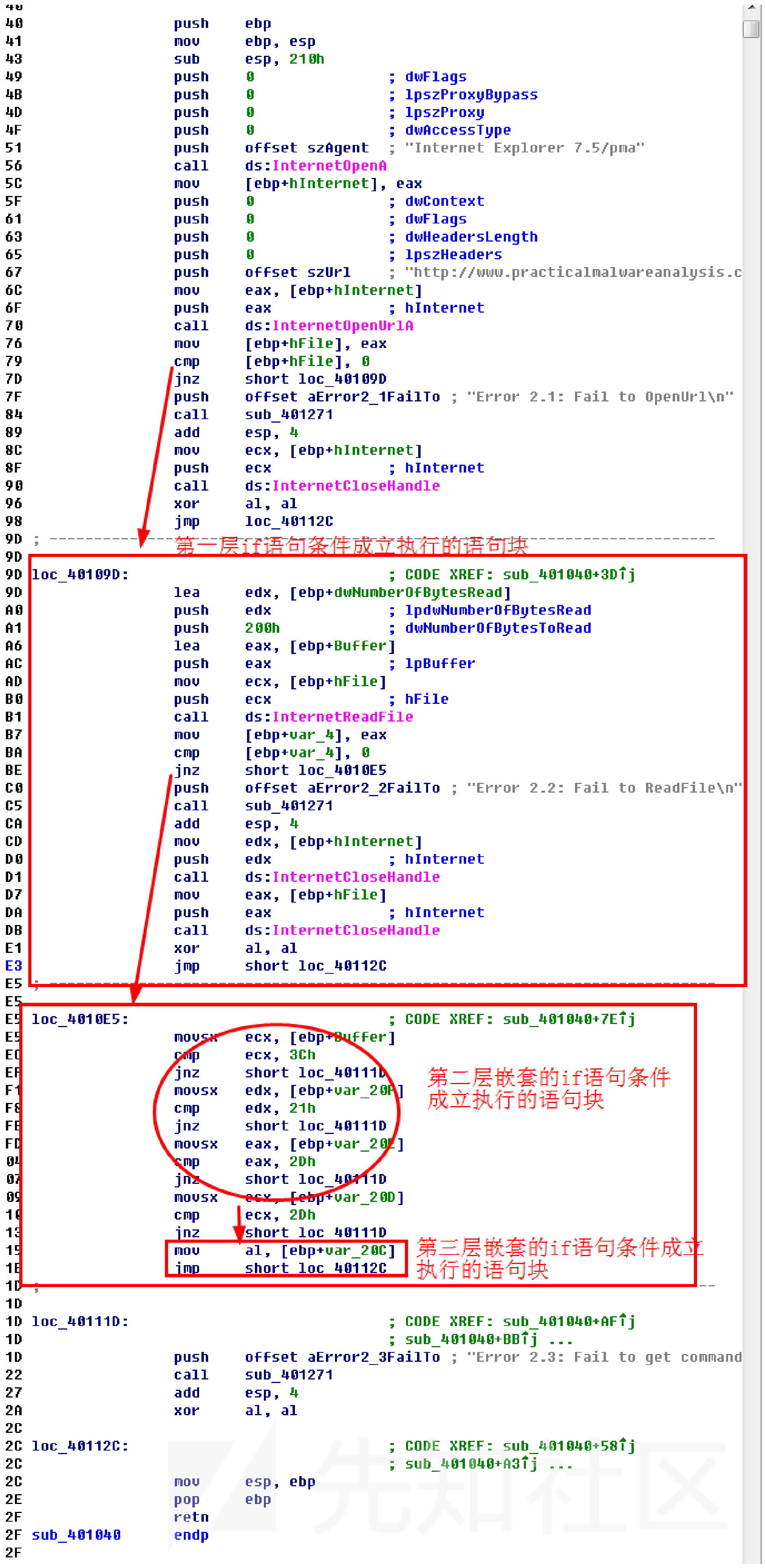
```
.text:00401210
.text:00401210 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401210 _main           proc near               ; CODE XREF: start+AF↓p
.text:00401210
.text:00401210 var_8           = byte ptr -8
.text:00401210 var_4           = dword ptr -4
.text:00401210 argc            = dword ptr  8
.text:00401210 argv            = dword ptr  0Ch
.text:00401210 envp            = dword ptr  10h
.text:00401210
.text:00401210                 push    ebp
.text:00401211                 mov     ebp, esp
.text:00401213                 sub     esp, 8
.text:00401216                 call    sub_401000
.text:0040121B                 mov     [ebp+var_4], eax
.text:0040121E                 cmp     [ebp+var_4], 0
.text:00401222                 jnz     short loc_401228
.text:00401224                 xor     eax, eax
.text:00401226                 jmp     short loc_40126D
.text:00401228 ; ---------------------------------------------------------------------------
.text:00401228
.text:00401228 loc_401228:                             ; CODE XREF: _main+12↑j
.text:00401228                 call    sub_401040
.text:0040122D                 mov     [ebp+var_8], al
.text:00401230                 movsx   eax, [ebp+var_8]
.text:00401234                 test    eax, eax
.text:00401236                 jnz     short loc_40123C
.text:00401238                 xor     eax, eax
.text:0040123A                 jmp     short loc_40126D
.text:0040123C ; ---------------------------------------------------------------------------
```

cmp指令，脑子里立刻浮现一个if-else语句流程图，将跳转后的语句和紧跟跳转指令后的指令填入对应的if和else语句块中。

判断条件：sub_401000函数返回结果，即联网状态

if(条件成立)：调用sub_401040函数获取返回结果，如果返回结果不为0则太跳转到loc_40123C，所以接下来分析sub_401040

else(条件不成立)：eax置0，并且跳转到main函数结尾

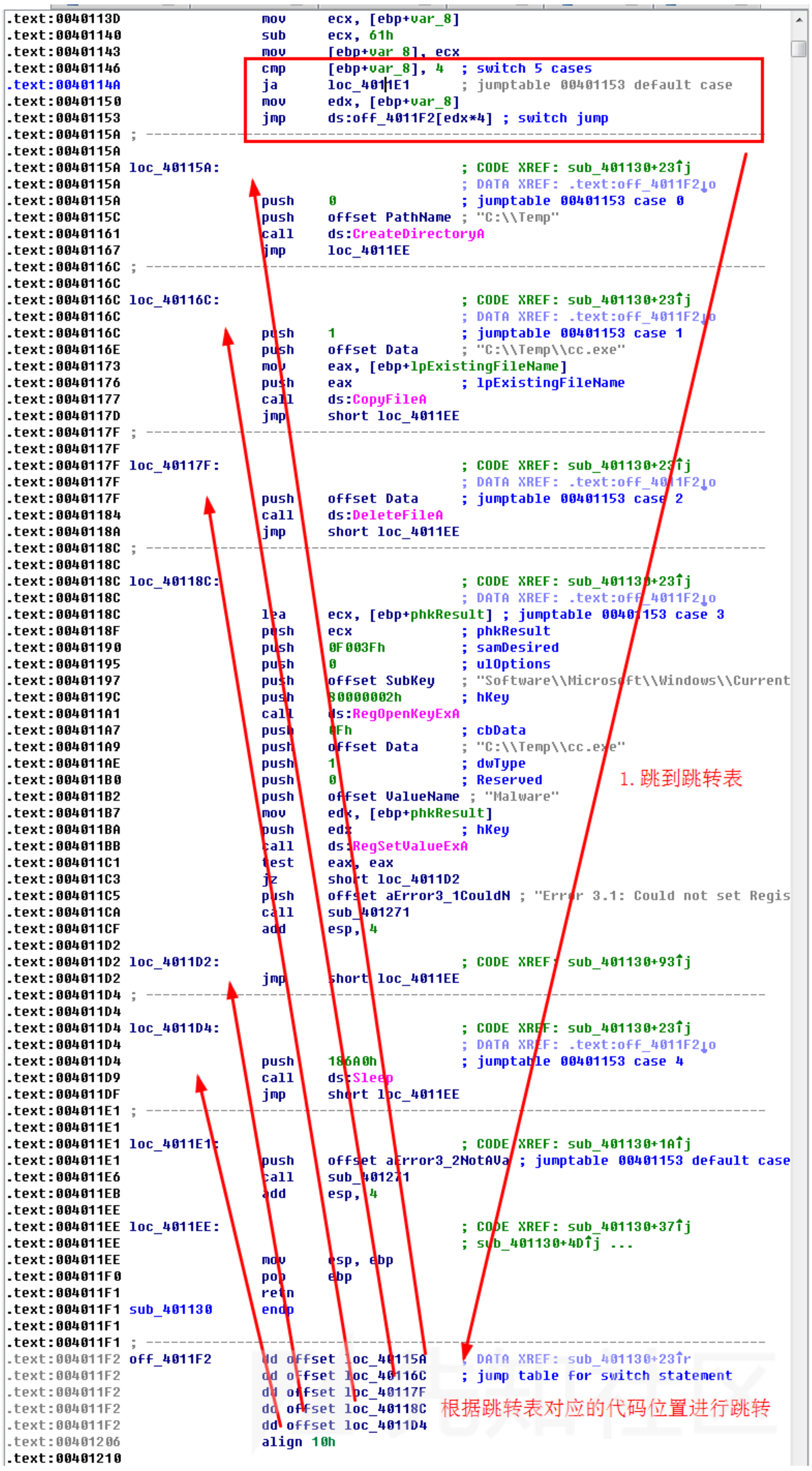

第一层if语句条件成立执行的语句块
第二层嵌套的if语句条件成立执行的语句块
第三层嵌套的if语句条件成立执行的语句块

```
40
40                push    ebp
41                mov     ebp, esp
43                sub     esp, 210h
49                push    0               ; dwFlags
4B                push    0               ; lpszProxyBypass
4D                push    0               ; lpszProxy
4F                push    0               ; dwAccessType
51                push    offset szAgent  ; "Internet Explorer 7.5/pma"
56                call    ds:InternetOpenA
5C                mov     [ebp+hInternet], eax
5F                push    0               ; dwContext
61                push    0               ; dwFlags
63                push    0               ; dwHeadersLength
65                push    0               ; lpszHeaders
67                push    offset szUrl    ; "http://www.practicalmalwareanalysis.c
6C                mov     eax, [ebp+hInternet]
6F                push    eax             ; hInternet
70                call    ds:InternetOpenUrlA
76                mov     [ebp+hFile], eax
79                cmp     [ebp+hFile], 0
7D                jnz     short loc_40109D
7F                push    offset aError2_1FailTo ; "Error 2.1: Fail to OpenUrl\n"
84                call    sub_401271
89                add     esp, 4
8C                mov     ecx, [ebp+hInternet]
8F                push    ecx             ; hInternet
90                call    ds:InternetCloseHandle
96                xor     al, al
98                jmp     loc_40112C
9D ; ---------------------------------------------------------------------------
9D
9D loc_40109D:                            ; CODE XREF: sub_401040+3D↑j
9D                lea     edx, [ebp+dwNumberOfBytesRead]
A0                push    edx             ; lpdwNumberOfBytesRead
A1                push    200h            ; dwNumberOfBytesToRead
A6                lea     eax, [ebp+Buffer]
AC                push    eax             ; lpBuffer
AD                mov     ecx, [ebp+hFile]
B0                push    ecx             ; hFile
B1                call    ds:InternetReadFile
B7                mov     [ebp+var_4], eax
BA                cmp     [ebp+var_4], 0
BE                jnz     short loc_4010E5
C0                push    offset aError2_2FailTo ; "Error 2.2: Fail to ReadFile\n"
C5                call    sub_401271
CA                add     esp, 4
CD                mov     edx, [ebp+hInternet]
D0                push    edx             ; hInternet
D1                call    ds:InternetCloseHandle
D7                mov     eax, [ebp+hFile]
DA                push    eax             ; hInternet
DB                call    ds:InternetCloseHandle
E1                xor     al, al
E3                jmp     short loc_40112C
E5 ; ---------------------------------------------------------------------------
E5
E5 loc_4010E5:                            ; CODE XREF: sub_401040+7E↑j
E5                movsx   ecx, [ebp+Buffer]
EC                cmp     ecx, 3Ch
EF                jnz     short loc_40111D
F1                movsx   edx, [ebp+var_20F]
F8                cmp     edx, 21h
FB                jnz     short loc_40111D
FD                movsx   eax, [ebp+var_20E]
04                cmp     eax, 2Dh
07                jnz     short loc_40111D
09                movsx   ecx, [ebp+var_20D]
10                cmp     ecx, 2Dh
13                jnz     short loc_40111D
15                mov     al, [ebp+var_20C]
1B                jmp     short loc_40112C
1D ; ---------------------------------------------------------------------------
1D
1D loc_40111D:                            ; CODE XREF: sub_401040+AF↑j
1D                                        ; sub_401040+BB↑j ...
1D                push    offset aError2_3FailTo ; "Error 2.3: Fail to get command
22                call    sub_401271
27                add     esp, 4
2A                xor     al, al
2C
2C loc_40112C:                            ; CODE XREF: sub_401040+58↑j
2C                                        ; sub_401040+A3↑j ...
2C                mov     esp, ebp
2E                pop     ebp
2F                retn
2F sub_401040     endp
2F
```

sub_401040: 第一层也就是最外层的if语句判断是否可以打开http://www.practicalmalwareanalysis.com/cc.htm
，如果可以打开则条件成立，进入嵌套的第二层if语句，判断是否可以读取该网页文件，如果可以则进入嵌套的第三层if语句，判断读取的内容是否以<!--开头，如果条件成

我们假设满足条件：可以访问到网页文件并且网页文件以<!--开头，返回数据后，我们进入loc_40123C主要分析sub_401130函数：

switch语句（if+跳转表）



```
.text:0040113D                 mov     ecx, [ebp+var_8]
.text:00401140                 sub     ecx, 61h
.text:00401143                 mov     [ebp+var_8], ecx
.text:00401146                 cmp     [ebp+var_8], 4  ; switch 5 cases
.text:0040114A                 ja      loc_4011E1      ; jumptable 00401153 default case
.text:00401150                 mov     edx, [ebp+var_8]
.text:00401153                 jmp     ds:off_4011F2[edx*4] ; switch jump
.text:0040115A ; ---------------------------------------------------------------------------
.text:0040115A
.text:0040115A loc_40115A:                             ; CODE XREF: sub_401130+23↑j
.text:0040115A                                         ; DATA XREF: .text:off_4011F2↓o
.text:0040115A                 push    0               ; jumptable 00401153 case 0
.text:0040115C                 push    offset PathName ; "C:\\Temp"
.text:00401161                 call    ds:CreateDirectoryA
.text:00401167                 jmp     loc_4011EE
.text:0040116C ; ---------------------------------------------------------------------------
.text:0040116C
.text:0040116C loc_40116C:                             ; CODE XREF: sub_401130+23↑j
.text:0040116C                                         ; DATA XREF: .text:off_4011F2↓o
.text:0040116C                 push    1               ; jumptable 00401153 case 1
.text:0040116E                 push    offset Data     ; "C:\\Temp\\cc.exe"
.text:00401173                 mov     eax, [ebp+lpExistingFileName]
.text:00401176                 push    eax             ; lpExistingFileName
.text:00401177                 call    ds:CopyFileA
.text:0040117D                 jmp     short loc_4011EE
.text:0040117F ; ---------------------------------------------------------------------------
.text:0040117F
.text:0040117F loc_40117F:                             ; CODE XREF: sub_401130+23↑j
.text:0040117F                                         ; DATA XREF: .text:off_4011F2↓o
.text:0040117F                 push    offset Data     ; jumptable 00401153 case 2
.text:00401184                 call    ds:DeleteFileA
.text:0040118A                 jmp     short loc_4011EE
.text:0040118C ; ---------------------------------------------------------------------------
.text:0040118C
.text:0040118C loc_40118C:                             ; CODE XREF: sub_401130+23↑j
.text:0040118C                                         ; DATA XREF: .text:off_4011F2↓o
.text:0040118C                 lea     ecx, [ebp+phkResult] ; jumptable 00401153 case 3
.text:0040118F                 push    ecx             ; phkResult
.text:00401190                 push    0F003Fh         ; samDesired
.text:00401195                 push    0               ; ulOptions
.text:00401197                 push    offset SubKey   ; "Software\\Microsoft\\Windows\\Current
.text:0040119C                 push    80000002h       ; hKey
.text:004011A1                 call    ds:RegOpenKeyExA
.text:004011A7                 push    0Fh             ; cbData
.text:004011A9                 push    offset Data     ; "C:\\Temp\\cc.exe"
.text:004011AE                 push    1               ; dwType
.text:004011B0                 push    0               ; Reserved
.text:004011B2                 push    offset ValueName ; "Malware"
.text:004011B7                 mov     edx, [ebp+phkResult]
.text:004011BA                 push    edx             ; hKey
.text:004011BB                 call    ds:RegSetValueExA
.text:004011C1                 test    eax, eax
.text:004011C3                 jz      short loc_4011D2
.text:004011C5                 push    offset aError3_1CouldN ; "Error 3.1: Could not set Regis
.text:004011CA                 call    sub_401271
.text:004011CF                 add     esp, 4
.text:004011D2
.text:004011D2 loc_4011D2:                             ; CODE XREF: sub_401130+93↑j
.text:004011D2                 jmp     short loc_4011EE
.text:004011D4 ; ---------------------------------------------------------------------------
.text:004011D4
.text:004011D4 loc_4011D4:                             ; CODE XREF: sub_401130+23↑j
.text:004011D4                                         ; DATA XREF: .text:off_4011F2↓o
.text:004011D4                 push    186A0h          ; jumptable 00401153 case 4
.text:004011D9                 call    ds:Sleep
.text:004011DF                 jmp     short loc_4011EE
.text:004011E1 ; ---------------------------------------------------------------------------
.text:004011E1
.text:004011E1 loc_4011E1:                             ; CODE XREF: sub_401130+1A↑j
.text:004011E1                 push    offset aError3_2NotAVa ; jumptable 00401153 default case
.text:004011E6                 call    sub_401271
.text:004011EB                 add     esp, 4
.text:004011EE
.text:004011EE loc_4011EE:                             ; CODE XREF: sub_401130+37↑j
.text:004011EE                                         ; sub_401130+4D↑j ...
.text:004011EE                 mov     esp, ebp
.text:004011F0                 pop     ebp
.text:004011F1                 retn
.text:004011F1 sub_401130      endp
.text:004011F1
.text:004011F1 ; ---------------------------------------------------------------------------
.text:004011F2 off_4011F2      dd offset loc_40115A    ; DATA XREF: sub_401130+23↑r
.text:004011F2                 dd offset loc_40116C    ; jump table for switch statement
.text:004011F2                 dd offset loc_40117F
.text:004011F2                 dd offset loc_40118C
.text:004011F2                 dd offset loc_4011D4
.text:00401206                 align 10h
.text:00401210
```

根据上一步从网页中获取的数据来得到对应的edx值，从而根据找到跳转表对应的位置进行跳转并执行相应代码。

这里有：

- 创建目录
- 复制当前程序到C:\Temp\cc.exe
- 删除C:\Temp\cc.exe
- 设置C:\Temp\cc.exe对应的开启自启动注册表键值Malware

小结

分析到这里基本完毕。

主要恶意行为就是通过从网页中获取的指令来执行对样本的隐藏、删除、自启动以及创建目录的操作。

switch补充

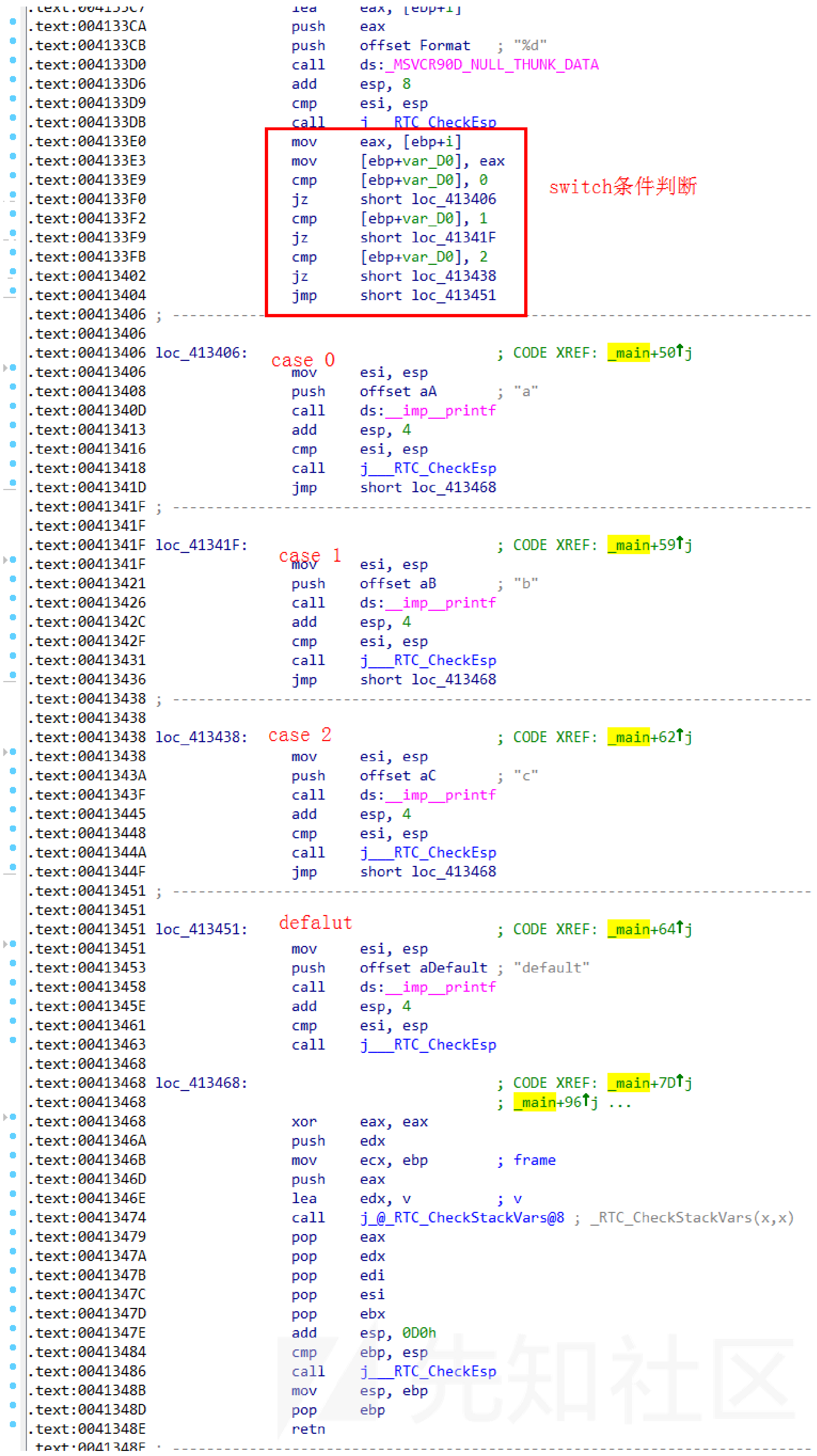
上面的实例中介绍到了switch的一种跳转表的跳转形式，下面补充一种纯用if语句进行的跳转：

真实代码

```
#include <stdio.h>


void main()
{
   int i = 0;
   scanf("%d", &i);
   switch(i)
   {
   case 0:
       printf("a");
       break;
   case 1:
       printf("b");
       break;
   case 2:
       printf("c");
       break;
   default:
       break;
   }
}
```

汇编：



```
.text:004133C7                 lea     eax, [ebp+i]
.text:004133CA                 push    eax
.text:004133CB                 push    offset Format   ; "%d"
.text:004133D0                 call    ds:_MSVCR90D_NULL_THUNK_DATA
.text:004133D6                 add     esp, 8
.text:004133D9                 cmp     esi, esp
.text:004133DB                 call    j___RTC_CheckEsp
.text:004133E0                 mov     eax, [ebp+i]
.text:004133E3                 mov     [ebp+var_D0], eax
.text:004133E9                 cmp     [ebp+var_D0], 0
.text:004133F0                 jz      short loc_413406
.text:004133F2                 cmp     [ebp+var_D0], 1
.text:004133F9                 jz      short loc_41341F
.text:004133FB                 cmp     [ebp+var_D0], 2
.text:00413402                 jz      short loc_413438
.text:00413404                 jmp     short loc_413451
.text:00413406 ; ---------------------------------------------------------------------------
.text:00413406
.text:00413406 loc_413406:                             ; CODE XREF: _main+50↑j
.text:00413406                 mov     esi, esp
.text:00413408                 push    offset aA       ; "a"
.text:0041340D                 call    ds:__imp__printf
.text:00413413                 add     esp, 4
.text:00413416                 cmp     esi, esp
.text:00413418                 call    j___RTC_CheckEsp
.text:0041341D                 jmp     short loc_413468
.text:0041341F ; ---------------------------------------------------------------------------
.text:0041341F
.text:0041341F loc_41341F:                             ; CODE XREF: _main+59↑j
.text:0041341F                 mov     esi, esp
.text:00413421                 push    offset aB       ; "b"
.text:00413426                 call    ds:__imp__printf
.text:0041342C                 add     esp, 4
.text:0041342F                 cmp     esi, esp
.text:00413431                 call    j___RTC_CheckEsp
.text:00413436                 jmp     short loc_413468
.text:00413438 ; ---------------------------------------------------------------------------
.text:00413438
.text:00413438 loc_413438:                             ; CODE XREF: _main+62↑j
.text:00413438                 mov     esi, esp
.text:0041343A                 push    offset aC       ; "c"
.text:0041343F                 call    ds:__imp__printf
.text:00413445                 add     esp, 4
.text:00413448                 cmp     esi, esp
.text:0041344A                 call    j___RTC_CheckEsp
.text:0041344F                 jmp     short loc_413468
.text:00413451 ; ---------------------------------------------------------------------------
.text:00413451
.text:00413451 loc_413451:                             ; CODE XREF: _main+64↑j
.text:00413451                 mov     esi, esp
.text:00413453                 push    offset aDefault ; "default"
.text:00413458                 call    ds:__imp__printf
.text:0041345E                 add     esp, 4
.text:00413461                 cmp     esi, esp
.text:00413463                 call    j___RTC_CheckEsp
.text:00413468
.text:00413468 loc_413468:                             ; CODE XREF: _main+7D↑j
.text:00413468                                         ; _main+96↑j ...
.text:00413468                 xor     eax, eax
.text:0041346A                 push    edx
.text:0041346B                 mov     ecx, ebp        ; frame
.text:0041346D                 push    eax
.text:0041346E                 lea     edx, v          ; v
.text:00413474                 call    j_@_RTC_CheckStackVars@8 ; _RTC_CheckStackVars(x,x)
.text:00413479                 pop     eax
.text:0041347A                 pop     edx
.text:0041347B                 pop     edi
.text:0041347C                 pop     esi
.text:0041347D                 pop     ebx
.text:0041347E                 add     esp, 0D0h
.text:00413484                 cmp     ebp, esp
.text:00413486                 call    j___RTC_CheckEsp
.text:0041348B                 mov     esp, ebp
.text:0041348D                 pop     ebp
.text:0041348E                 retn
.text:0041348E ; ---------------------------------------------------------------------------
```

cmp + jz + jmp实现的switch流程

点击收藏 | 0 关注 | 2

上一篇：通过Dpapi获取Windows身份凭证 下一篇：从Linux到Windows栈溢出利用

1. 0 条回复
    - 动动手指，沙发就是你的了！