

Microsoft Office CTaskSymbol Use-After-Free Vulnerability

17/08/2015

Software:	Microsoft Office
Affected Versions:	MS Office 2013 SP1 (x86, x64)
CVE Reference:	CVE-2015-1642
Author:	Yong Chuan, Koh (@yongchuank)
Severity:	Important
Vendor:	Microsoft
Vendor Response:	Fix released as part of MS15-081

Description

Microsoft Office is a suite of desktop applications consisting of Word, Powerpoint, Excel, Outlook and various other productivity applications. The applications are affected by a use-after-free vulnerability while parsing a specially crafted Office file as a result of the application loading the `CTaskSymbol` COM object in memory.

Impact

If persuaded to open the crafted Office file, a successful exploitation would allow an attacker to run arbitrary code in the context of the target application.

Cause

The vulnerability exists because Microsoft Office incorrectly dereferences the `CTaskSymbol` object after it is freed.

Interim Workaround

Avoid opening Office files from untrusted sources or view them in Protected-View mode.

Solution

Users should apply MS15-081 updates from Microsoft.

Technical Details

As WINWORD tries to load the `CTaskSymbol` COM (UUID: 44F9A03B-A3EC-4F3B-9364-08E0007F21DF), with page-heap enabled, it crashes at the following address:

```

Command - Pid 1812 - WinDbg:6.3.9600.17298 X86
(714.304): C++ EH exception - code e06d7363 (first chance)
(714.160): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Program Files\Microsoft Office\Office15\wwlib.dll -
eax=2da16fc0 ebx=1b18928 ecx=2cf2afc4 edx=00ff94f4 esi=2cf2afb8 edi=1cfd08f58
eip=59bc8c4b esp=00ff94cc ebp=00ff94dc iopl=0         nv up ei pl zr ac pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010216
nmcmdngr!ATL::CCoMContainedObject<CTaskSymbol>::QueryInterface+0x13:
59bc8c4b 8b08          mov     ecx,dword ptr [eax] ds:0023:2da16fc0=????????
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Program Files\Microsoft Office\Office15\WINWORD.EXE -
0:000> ub .iu .
nmcmdngr!ATL::CCoMContainedObject<CTaskSymbol>::QueryInterface+0x2:
59bc8c3a 55          push   ebp
59bc8c3b 8bec        mov     ebp,esp
59bc8c3d 56          push   esi
59bc8c3e 8b7508      mov     esi,dword ptr [ebp+8]
59bc8c41 ff7510      push  dword ptr [ebp+10h]
59bc8c44 ff750c      push  dword ptr [ebp+0Ch]
59bc8c47 8b463c      mov     eax,dword ptr [esi+3Ch]
59bc8c4a 50          push   eax
nmcmdngr!ATL::CCoMContainedObject<CTaskSymbol>::QueryInterface+0x13:
59bc8c4b 8b08        mov     ecx,dword ptr [eax]
59bc8c4d ffd1        call   dword ptr [ecx]
59bc8c4f 85c0        test   eax,eax
59bc8c51 791b        jns   nmcmdngr!ATL::CCoMContainedObject<CTaskSymbol>::QueryInterface+0x36 (59bc8c6e)
59bc8c53 8d4e34      lea   ecx,[esi+34h]
59bc8c56 3b4e3c      cmp   ecx,dword ptr [esi+3Ch]
59bc8c59 7413        je    nmcmdngr!ATL::CCoMContainedObject<CTaskSymbol>::QueryInterface+0x36 (59bc8c6e)
59bc8c5b ff7510      push  dword ptr [ebp+10h]
0:000> !heap -p -a eax
address 2da16fc0 found in
DPH_HEAP_ROOT @ 1031000
in free-ed allocation ( DPH_HEAP_BLOCK:          VirtAddr      VirtSize)
                             2de31d9c:          2da15000      3000
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Windows\system32\verifier.dll -
69738fc2 verifier!VerifierDisableFaultInjectionExclusionRange+0x000003232
772b9620 ntdll!RtlDebugFreeHeap+0x00000032
7727a51d ntdll!RtlpFreeHeap+0x00005ec8d
7721b545 ntdll!RtlFreeHeap+0x00000475
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Program Files\Common Files\Microsoft Shared\Office15\aso.dll -
5f366ed8 mso!MsoFreePv+0x0000000a0
5f366e7f mso!MsoFreePv+0x000000047
5f366e36 mso!Ordinal3195+0x00000011
5f3af62b mso!Ordinal3679+0x000002b5
5f4ed009 mso!Ordinal5092+0x00000042
5f4bee1c mso!Ordinal5013+0x000000614
5f3ac55c mso!Ordinal7961+0x00000486
5f3a98fe mso!Ordinal5928+0x00000161f
7724226c ntdll!_RtlUserThreadStart+0x00000020
77242245 ntdll!_RtlUserThreadStart+0x0000001b

```

```

Command - Pid 1812 - WinDbg:6.3.9600.17298 X86
0:000> k
ChildEBP RetAddr
00ff94dc 621b514a nmcmdngr!ATL::CCoMContainedObject<CTaskSymbol>::QueryInterface+0x13
WARNING: Stack unwind information not available. Following frames may be wrong.
00ff94fc 627ccded wlib!DllGetLCID+0x523408
00ff9538 627ca1be wlib!wdGetApplicationObject+0xcc5f6
00ff9558 61ecedd9 wlib!wdGetApplicationObject+0xc99c7
00ff957c 61b1923c wlib!DllGetLCID+0x23d097
00ff95a4 61b16012 wlib!DllGetClassObject+0xd4268
00ffc198 61b15c3e wlib!DllGetClassObject+0xd103e
00ffd824 61b1c92a wlib!DllGetClassObject+0xd0c6a
00ffd840 61b60a84 wlib!DllGetClassObject+0xd7956
00ffd858 61b60955 wlib!DllGetClassObject+0x11bab0
00ffd944 61ab02f9 wlib!DllGetClassObject+0x11b981
00ffda04 61aae2fb wlib!DllGetClassObject+0x6b325
00ffda24 61aac2c5 wlib!DllGetClassObject+0x69327
00ffda48 61aa9593 wlib!DllGetClassObject+0x672f1
00fffac0 011d15c4 wlib!DllGetClassObject+0x645bf
00fffae4 011d1558 WINWORD+0x15c4
00fffb78 752217ad WINWORD+0x1558
00fffb84 7724226c KERNEL32!BaseThreadInitThunk+0xe
00fffb88 77242245 ntdll!_RtlUserThreadStart+0x20
00fffbdb 00000000 ntdll!_RtlUserThreadStart+0x1b

```

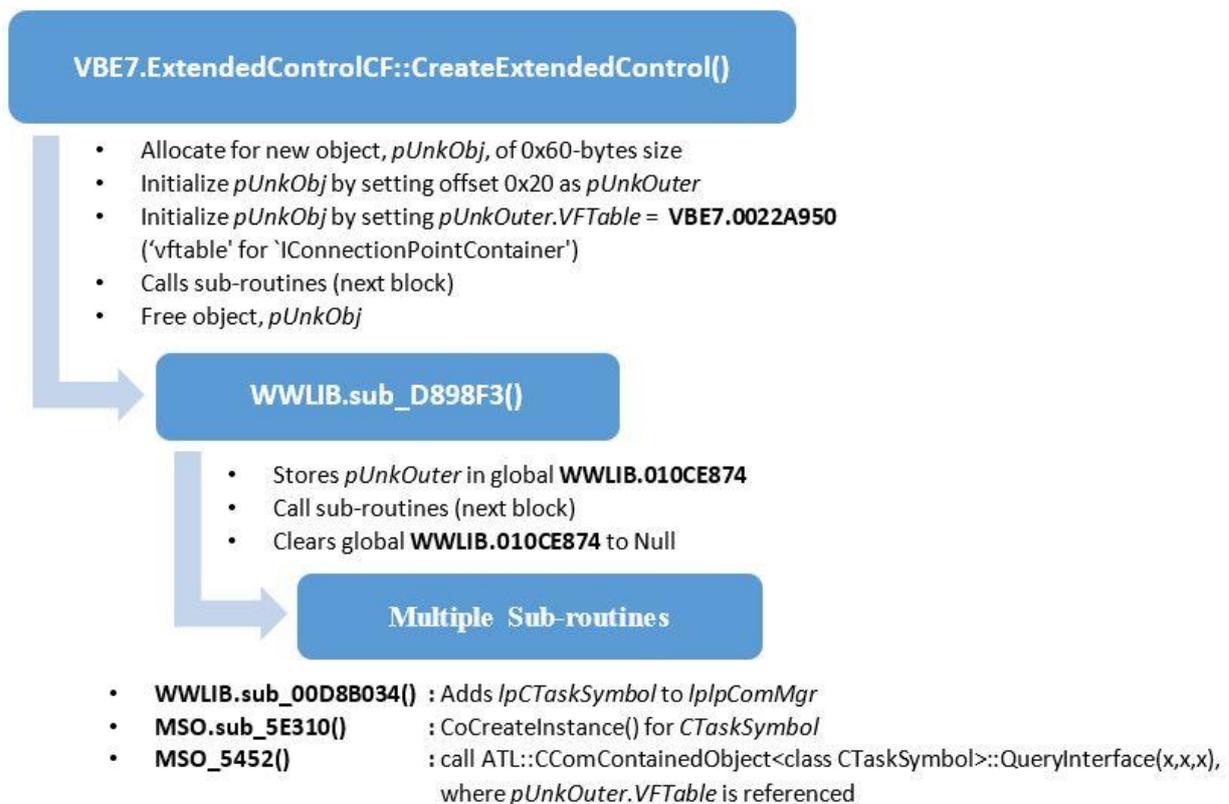
In the first screenshot, the invalid `VFTable` read at address `0x59bc8c4b` occurs because the object has already been freed. With some reverse-engineering, it was found that this `VFTable` belongs to the aggregate object's `IUnknown` interface, `pUnkOuter`, of the `CTaskSymbol` object:

```

CoCreateInstance (
    rclsid      = {44F9A03B-A3EC-4F3B-9364-08E0007F21DF},
    pUnkOuter   = aggregate object's IUnknown interface,
    dwClsContext = CLSCTX_INPROC_SERVER|CLSCTX_INPROC_HANDLER,
    riid        = {00000000-0000-0000-C000-000000000046},
    ppv         = lpOutput
)

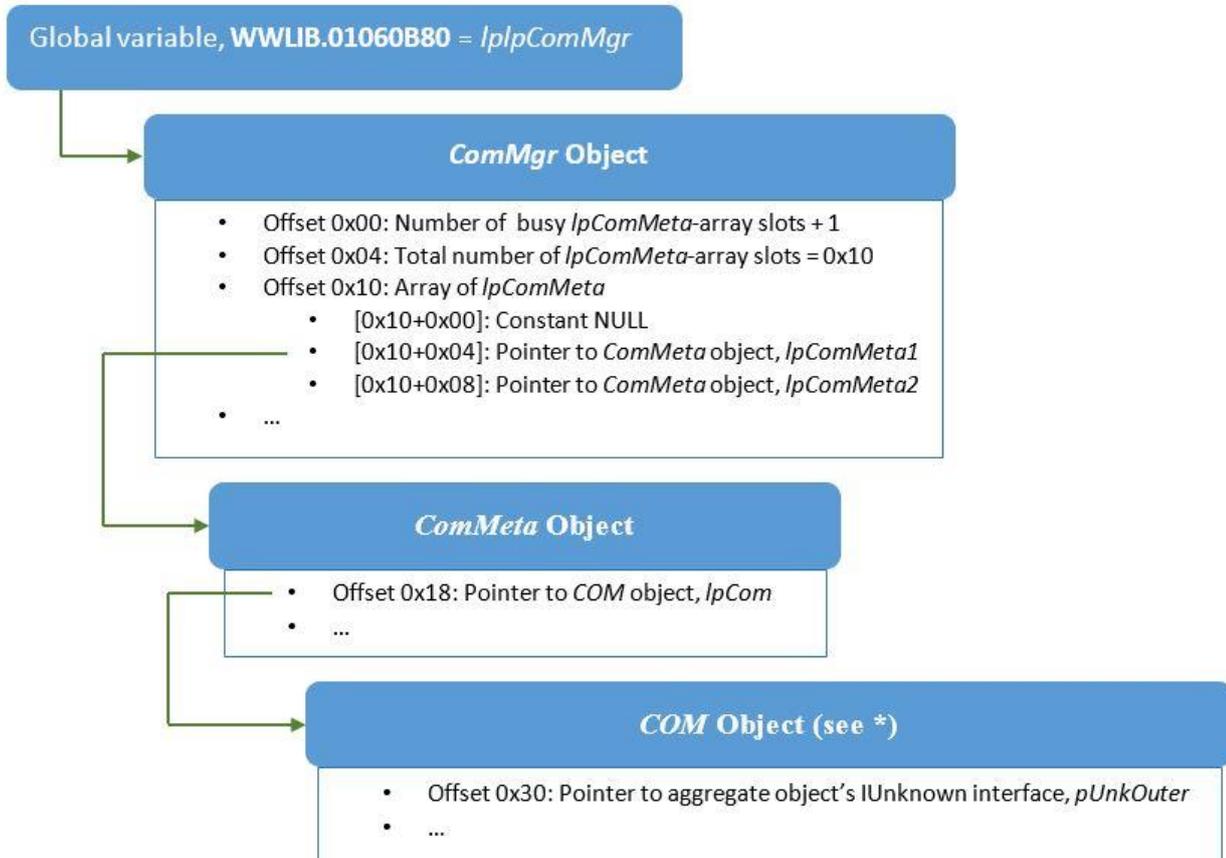
```

This `pUnkOuter IUnknown` interface object is allocated, used and freed in the following sequence:



The series of actions to load a COM object starts in `VBE7.ExtendedControlCF::CreateExtendedControl()` where it allocates a new 0x60 bytes `pUnkObj` object, sets `pUnkOuter` to its +0x20 offset, calls into `WWLIB.sub_D898F3()` and finally frees `pUnkObj`. In `WWLIB.sub_D898F3()`, the target `pUnkOuter` is first stored in global `WWLIB.010CE874`, calls into some subroutines and finally clears the reference in `WWLIB.010CE874`. Among the subroutines, the `CTaskSymbol` COM is instantiated with `pUnkOuter` and then added to the internal COM manager, `lpComMgr`.

As its name implies, the `lpComMgr` COM manager help WINWORD tracks all COM objects that are embedded into the current document. Its structure and relationship to `pUnkOuter` of the `CTaskSymbol` COM is illustrated:



* In this case, `lpCom` refers to `TaskSymbol` COM (ie: `lpCTaskSymbol`)

As the diagram above shows, `lpComMgr` maintains an array of references to all embedded COM objects, whose details are described in each `lpCom` object. Among the details is the COM's aggregate object's `IUnknown` interface, `pUnkOuter`, at offset +0x30.

At this point, there are 2 references to `pUnkOuter`, one in global `WWLIB.010CE874` and the other in `lpComMgr`. As WINWORD exits `VBE7.ExtendedControlCF::CreateExtendedControl()`, the `pUnkOuter` is freed (as part of `pUnkObj`) and its reference in `WWLIB.010CE874` is cleared. However it does not remove the reference in `lpComMgr -> lpComMeta -> lpCTaskSymbol -> pUnkOuter`. As a result, in a subsequent call to `ATL::CComContainedObject<class CTaskSymbol>::QueryInterface(x, x, x)`, the invalid/freed `lpCTaskSymbol -> pUnkOuter -> VTable` is being referenced. The following screenshot illustrates these:

Detailed Timeline

Date:	Summary:
27/02/2015	MWR Labs reported issue to Microsoft
28/02/2015	Microsoft acknowledge receipt and start investigation of issue
02/04/2015	Microsoft replied that issue is assigned CVE-2015-1642
11/08/2015	Microsoft released patch as part of MS15-081
17/08/2015	MWR Labs released advisory