How to better fuzz Directx kernel at present

刘晓亮

Qihoo 360 IceSword Lab

About Me

- Security Researcher of Qihoo 360 IceSword Lab
- Main focus: Bug Hunting, Windows Kernel
- twitter:flame陆(https://twitter.com/flame36987044)
- weibo:flamelxl

About IceSword Lab

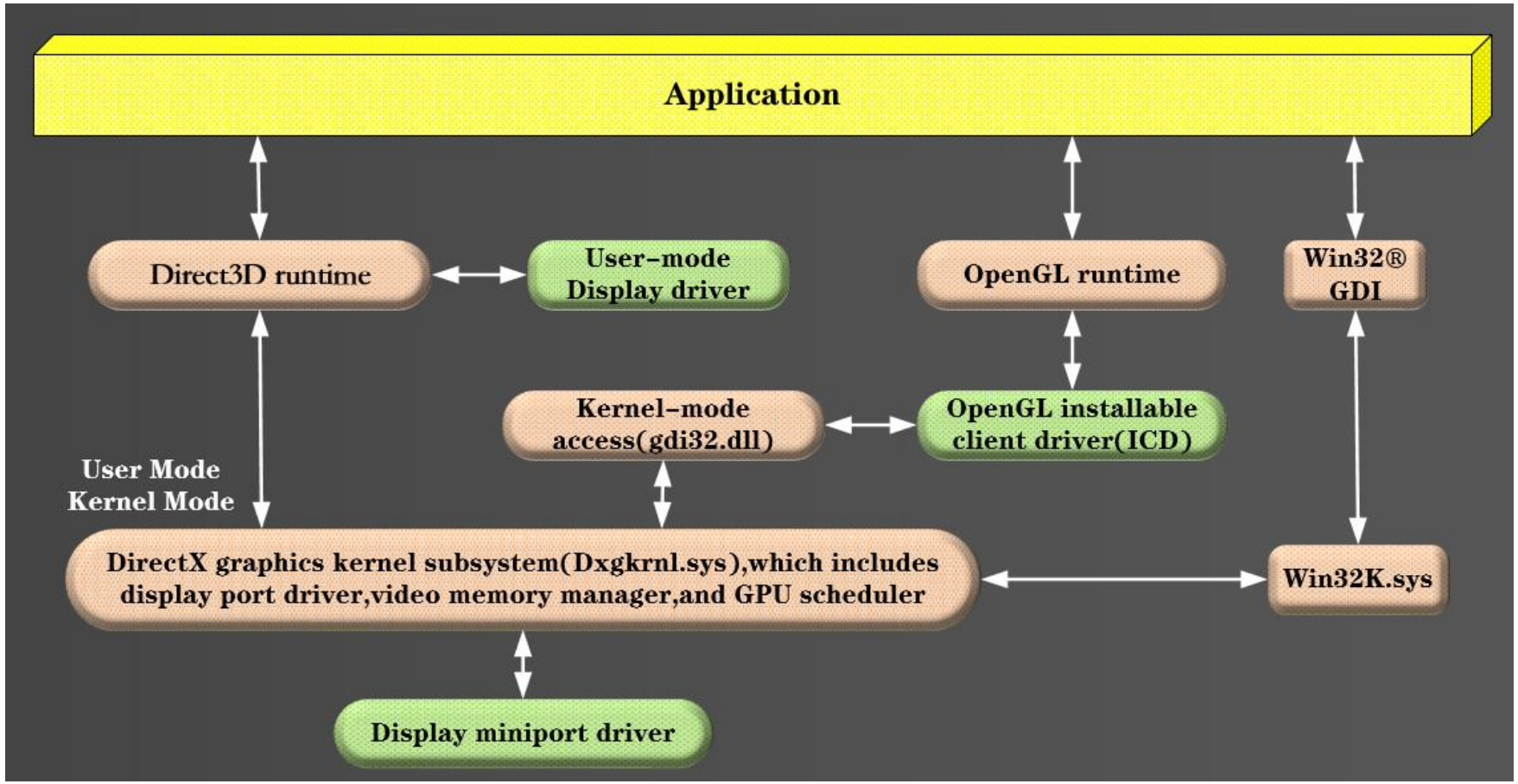
- About Leader
  - 360 Group Fellow (VP).
  - Chief Scientist of 360 Enterprise Group.
  - Author of the famous Anti rootkit software IceSword.
- Team members include
  - Top 5 of Qualcomm's vulnerability mining ranking
  - MSRC TOP 100 in 2016/2017/2018/2019
  - Outstanding driver development team and many others

Presentation Outline

- About DirectX
- Attack surface
- Fuzzing
- Case Study
- Summary&Reflection

About DirectX

- WDDM Architecture diagram



About DirectX

- Dxgkrnl.sys Exports

unction name
DXGDEVICE::QueryLastCompletedPres
DXGDEVICE::RemoveAllocationsWithc
DXGDEVICE::RemoveDirectFlipAlloc
DXGDEVICE::RemovePrimaryAllocatic
DXGDEVICE::RemoveResourceFromDevi
DXGDEVICE::RemoveVidPnOwnership(v
DXGDEVICE::ReportAllocationState(
DXGDEVICE::ReportDeviceAllocatior
DXGDEVICE::ReportDeviceResources(
DXGDEVICE::ReportDeviceSyncObject
DXGDEVICE::ReportState(void)
DXGDEVICE::Reset(void)
DXGDEVICE::Stop(uchar)
DXGDEVICE::SuspendResumeEscape(bc
DXGDEVICE::TrimAllDmaPoolsToMinim
DXGDEVICE::UnpinAllDirectFlipAllc
DXGDEVICE::UnpinDeviceAllocations
DXGDEVICE::UnpinDeviceResources(v
DXGDEVICE::UnpinDirectFlipAllocat

Name	Address
DxgkMapGpuVirtualAddress	000BAA50
DxgkMarkDeviceAsError	0019E373
DxgkNetDispGetNextChunkInfo	0016FBF6
DxgkNetDispQueryMiracastDisplayDeviceStatus	0016FF32
DxgkNetDispQueryMiracastDisplayDeviceSupport	001700DA
DxgkNetDispStartMiracastDisplayDevice	0017017A
DxgkNetDispStopMiracastDisplayDevice	0017030C
DxgkOfferAllocations	000BA0F0
DxgkOpenAdapterFromDeviceName	000AD60C
DxgkOpenAdapterFromHdc	000BCF3A
DxgkOpenAdapterFromLuid	000AEED4
DxgkOpenBundleObjectNtHandleFromName	001B8492
DxgkOpenKeyedMutex2	001CEF6C
DxgkOpenKeyedMutex	001CF1BE
DxgkOpenKeyedMutexFromNtHandle	001B85D4
DxgkOpenNtHandleFromName	001B8826
DxgkOpenProtectedSessionFromNtHandle	001C657E
DxgkOpenResource	000BCD48
DxgkOpenResourceFromNtHandle	000AD83A

About DirectX

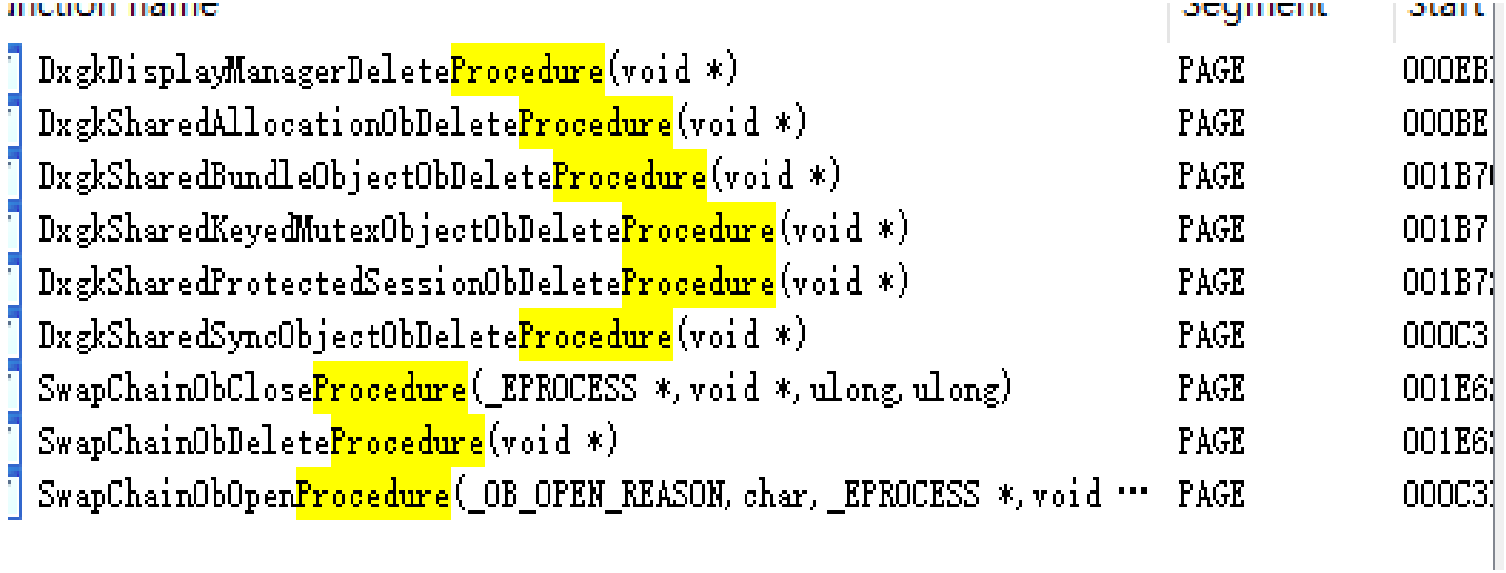
Kernel	R3	Syscall
DXGDEVICE	DeviceHandle	D3DKMTCreateDevice
DXGADAPTER	AdapterHandle	D3DKMTEnumAdapters
DXGRESOURCE	ResourceHandle	D3DKMTCreateAllocation
DXGALLOCATION	AllocationHandle	D3DKMTCreateAllocation
DXGSYNC	SyncHandle	D3DKMTCreateSynchronizationObject
DXGCONTEXT	ContextHandle	D3DKMTCreateCreateContext

```
18 = (_DWORD *)DXGQUOTAALLOCATOR<1,1265072196>::op
f ( v18 )
 v21 = (DXGCONTEXT *)DXGCONTEXT::DXGCONTEXT(v18, v
lse
 v21 = 0;
f ( v21 )

 v22 = DXGCONTEXT::Initialize(v21, a7, a8);
 if ( v22 < 0 )
 {
   DXGCONTEXT::DestroyContext(v21, 0);
   DXGCONTEXT::~DXGCONTEXT(v21);


  v13 = operator new[](0x4B677844u, 0x70u, (POOL_TYPE)512)
  v16 = v15;
  if ( v13 )
    v12 = (DXGKEYEDMUTEX *)DXGKEYEDMUTEX::DXGKEYEDMUTEX(v1
  if ( v12 )
  {
    v20 = DXGKEYEDMUTEX::Initialize(v12);
    if ( v20 >= 0 )
    {
      DXGKEYEDMUTEX::AcquireReference(v12);
      DXGFASTMUTEX::Acquire((DXGFASTMUTEX *)(v24 + 280));


 v14 = DXGQUOTAALLOCATOR<1,1265072196>::operator new(0x468u);
 if ( v14 )
   v17 = (DXGDEVICE *)DXGDEVICE::DXGDEVICE(v14, v15, (int)v9, a8, (
 else
   v17 = 0;
 if ( v17 )
 {
   v18 = DXGDEVICE::Initialize(v17, a6, a7);
   if ( v18 >= 0 )
   {
     if ( *((_DWORD *)v17 + 41) == 2 )
```

How to do?

Attack surface

- First attack surface：
  - Find where some unreleased memory is released



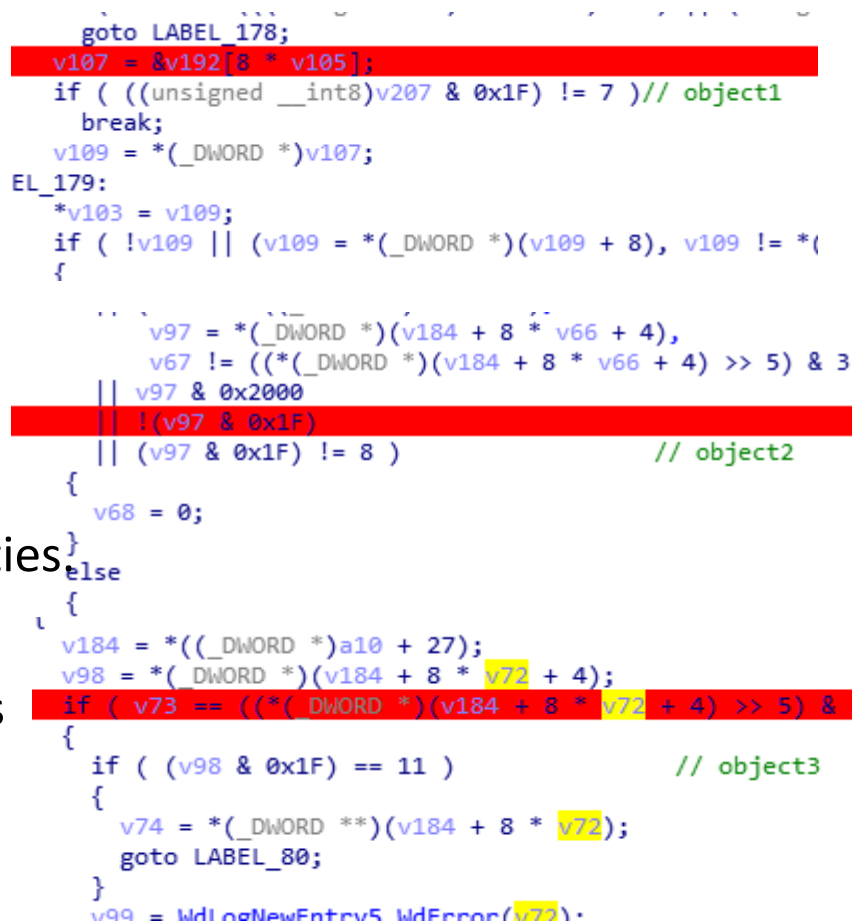
Attack surface

- Second attack surface
  - sunch as:

Function	Object mark	flag
DxgkCreateSynchronizationObject	8	D3DDDI_SYNCHRONIZATIONOBJECT_FLAGS
DxgkCreateSynchronizationObject2	8	D3DDDI_SYNCHRONIZATIONOBJECT_FLAGS
DxgkCreateContextVirtual	7	D3DDDI_CREATECONTEXTFLAGS
DxgkCreateAllocation	4	D3DKMT_CREATEALLOCATIONFLAGS
DxgkCreateKeyedMutex2	9	D3DKMT_CREATEKEYEDMUTEX2_FLAGS

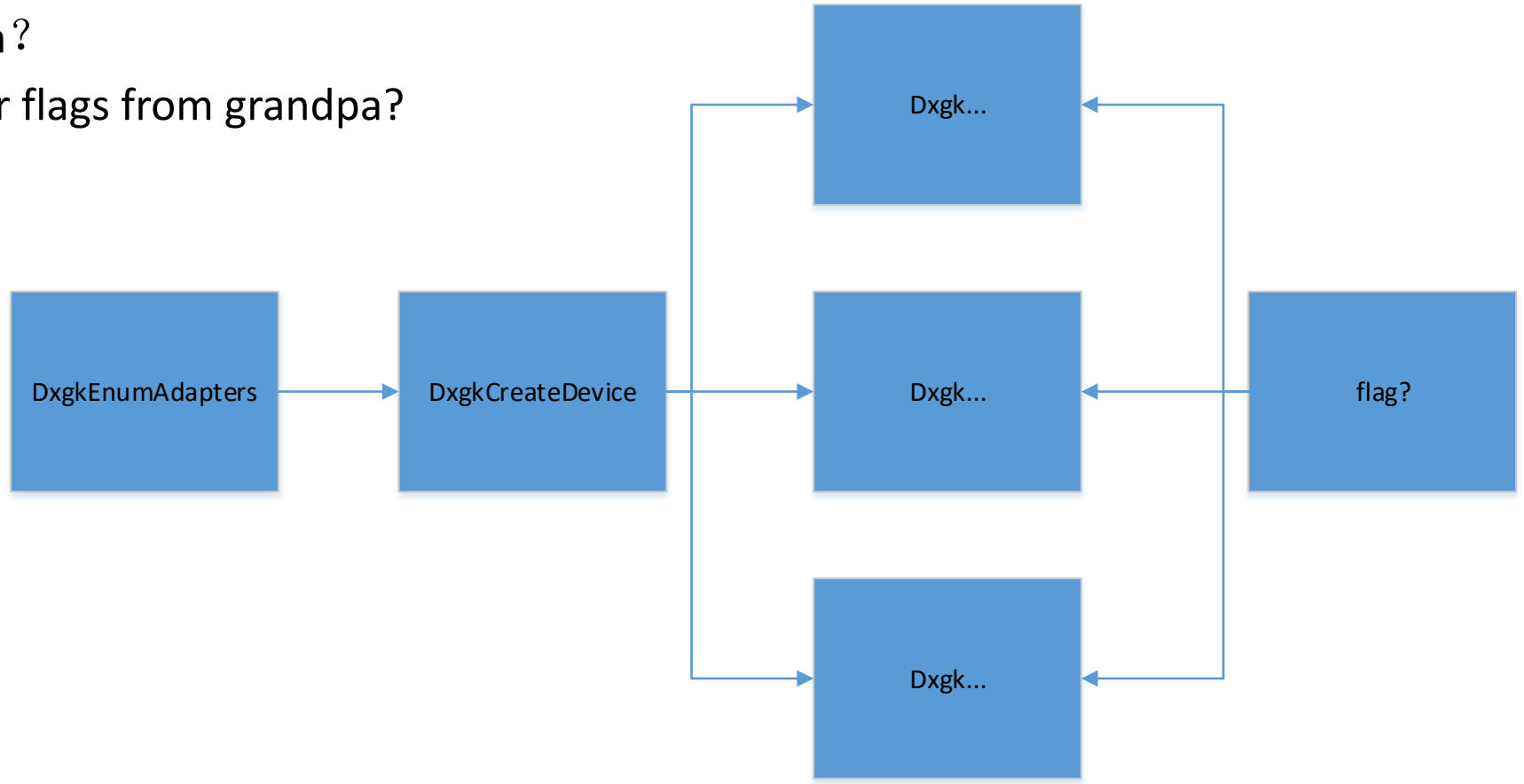
Attack surface

- Second attack surface:
- **In the kernel:**
  - A function can have different parameter
  - A function can create objects of different properties.
  - The flag determines their call path and Attributes in the kernel！



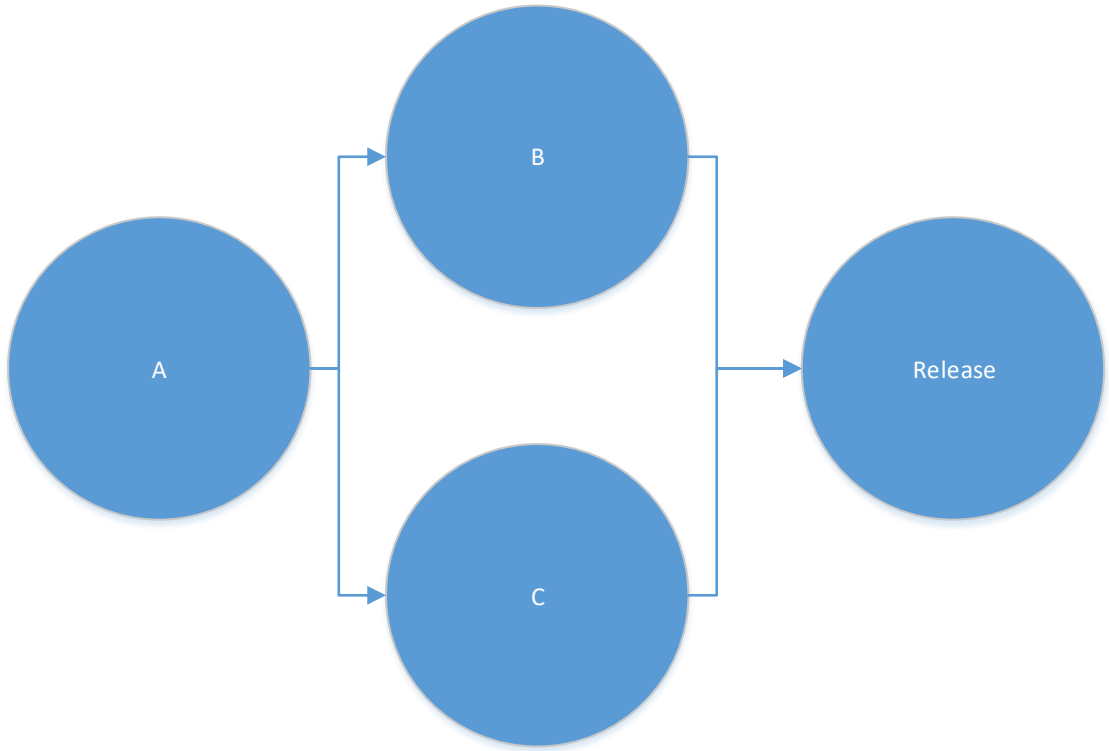
Attack surface

- Parent.flag decide son？
- Grandson needs other flags from grandpa?



Attack surface

- Third attack surface:
  - Reverse engineering
  - understanding undisclosed functional relationships
  - Establish corresponding structural relationships
  - and functional dependencies



- CVE-2020-0714

```
if ( *((PKTHREAD *)this + 4) != KeGetCurrentThread() )
{
  v6 = WdLogNewEntry5_WdAssertion(v5, v4);
  *(_DWORD *)(v6 + 12) = 2624;
  WdLogEvent5_WdAssertion(v6);
}
v7 = (char *)this + (*((_DWORD *)a2 + 2) != 0 ? 96 : 68);
if ( *((_DWORD *)a2 + 1) )
{
  if ( *((_DWORD *)v7 + 4) != 1 )
  {
    v8 = WdLogNewEntry5_WdError(v5);
    *(_DWORD *)(v8 + 12) = 2636;
    WdLogEvent5_WdError(v8);
    v9 = 0xC00000BB;
LABEL_6:
    v10 = a2;
    goto LABEL_31;
  }
  if ( *(_DWORD *)(*((_DWORD *)this + 38) + 8) == *(_DWORD *)(*((_DWORD *)this + 38) + 12) )
```

```
  v29 = operator new[](0x4B677844u, 4 * v24 | -((unsigned __int64)v24 >> 30 != 0), PagedPool);
  if ( !v29 )
  {
    v7 = WdLogNewEntry5_WdLowResource();
    *(_DWORD *)(v7 + 12) = 1264;
    WdLogEvent5_WdLowResource(v7);
    DXGETWPROFILER_BASE::PopProfilerEntry((DXGETWPROFILER_BASE *)&v26);
    if ( dword_5E3D0 & 2 && Microsoft_Windows_DxgKrnlEnableBits & 0x2000 )
      McTemplateK0q(v8, v26);
    return -1073741801;
  }
  Dst = v29;
```

```
  v9 = operator new[](0x4B677844u, 4 * v31 | -((unsigned __int64)v31 >> 30 != 0), PagedPool);
  v3 = v9;
  v37 = v9;
  if ( !v9 )
  {
    v10 = WdLogNewEntry5_WdLowResource();
    *(_DWORD *)(v10 + 12) = 1276;
    WdLogEvent5_WdLowResource(v10);
    DXGETWPROFILER_BASE::PopProfilerEntry((DXGETWPROFILER_BASE *)&v33);
    if ( v35 && byte_70181 & 0x20 )
      McTemplateK0q(v11, v33);
    return -1073741801;
  }
  memset(v9, 0, 4 * v6);
  v32 = v3;
}
```

start to fuzz

Then we found these

```
a : 00000000`00000013 00000000`000000c4 ffffcb81`ae36bf10 fffff802`4c6a4140 : nt!DbgBreakPointWithStatus
d : 00000000`00000003 ffffcb81`ae36bf10 fffff802`4c8064e0 00000000`000000c4 : nt!KiBugCheckDebugBreak+0x12
4 : ffffa98b`b3417a00 fffff802`4c7149a6 ffffa98b`ae1d69d0 00000000`00001000 : nt!KeBugCheck2+0x8a5
b : 00000000`000000c4 00000000`00000013 00000000`00001e9e ffffa98b`ae1d69c0 : nt!KeBugCheckEx+0x104
9 : ffffa98b`ae1d69d0 00000000`00010202 ffffa98b`b32b9d70 fffff809`6978371d : nt!ExFreePoolSanityChecks+0x11b
2 : ffffa98b`b32b9d70 ffff820a`dd180fd8 00000000`00000000 ffffa98b`08000000 : nt!VerifierExFreePoolWithTag+0x39
'e : ffff820a`dd180ff8 00000000`00000000 ffff820a`dd180fd8 ffff820a`dd180fc0 : dxgmms2!VIDMM_FENCE_STORAGE_PAGE::FreeStorage+0x2a
```

```
WARNING: Stack unwind information not available. Following frames may be wrong.
aefdf2a0 813ffc92 0000001e c0000005 89c01b93 nt!KeBugCheckEx
aefdf2bc 813a69e2 aefdf7e8 814ab328 aefdf3b0 nt!KeRegisterNmiCallback+0x184
aefdf2e0 813a69b4 aefdf7e8 814ab328 aefdf3b0 nt!ExRaiseStatus+0xce
aefdf3a0 8129499e aefdf7e8 aefdf3b0 00010037 nt!ExRaiseStatus+0xa0
aefdf7cc 8139fc11 aefdf7e8 00000000 aefdf8c4 nt!RtlInitUnicodeStringEx+0x11ae
aefdf838 813a44df 00000000 00000000 00000000 nt!Kei386EoiHelper+0x309
aefdf8dc 812c0463 aefdf8d0 00000000 00000000 nt!Kei386EoiHelper+0x4bd7
aefdfb80 89bb8d42 aefdfb94 8139e42e 012930e8 nt!ExReleasePushLockSharedEx+0x123
```

and other

```
a295b814 81997b01 00000050 c8294ff8 00000002 nt!KiBugCheck2+0xc6
a295b834 818a7348 00000050 c8294ff8 00000002 nt!KeBugCheckEx+0x19
a295b890 81930fac a295ba1c 81930fac a295ba1c nt!MiSystemFault+0xc58
a295b978 819acb81 00000002 c8294ff8 00000000 nt!MmAccessFault+0x12c
a295b978 900857c7 00000002 c8294ff8 00000000 nt!KiTrap0E+0x2d5
a295bae0 900ec757 9a9b9a16 0146b6a8 332ff718 dxgkrnl!DXGPAGINGQUEUE::RemoveReference+0x11
```

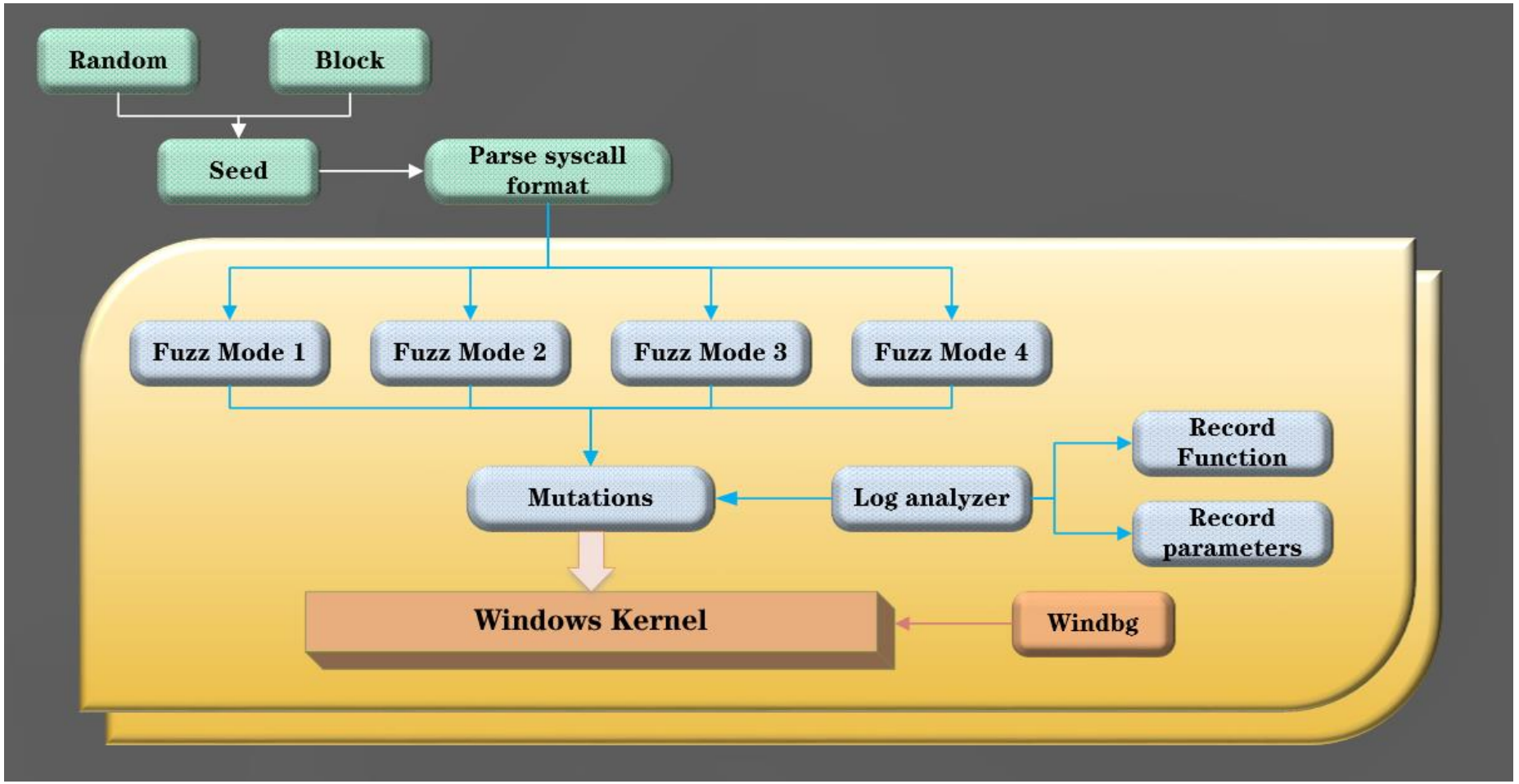
fuzz framework

- The recent fuzz about the Linux kernel has talked about a lot of things about blocks,like this

```
void func1(){
 open(...)
 read / write(...)
close(...)
}
```

- so we have also combined a lot of blocks.
- Then randomly call them,Sometimes the A function is selected, sometimes the B function is selected, but some functions must be called...

Fuzz Framework



Case Study

CVE-2020-1258

```
581ee40 ffff8508`d581ed98 : nt!KeBugCheckEx
491b0 fffff804`49669070 : nt!KiBugCheckDispatch+0x69
0103 00000000`00000000 : nt!KiFastFailDispatch+0xd0
0060 fffff804`497123a9 : nt!KiRaiseSecurityCheckFailure+0x325
000000 ffffaf0b`3d3d3100 : dxgmms2!VIDMM_PROCESS_FENCE_STORAGE::FreeSharedFenceStorageSlot+0x7a
0001 00000000`00000000 : dxgmms2!VIDMM_GLOBAL::FreeFenceStorageSlot+0x30
0000 00000000`00000000 : dxgmms2!VidMmFreeFenceStorageSlot+0x9
0 00000000`00000000 : dxgkrnl!DXGSYNCOBJECT::~DXGSYNCOBJECT+0x9b
00 ffffaf0b`58433910 : dxgkrnl!DXGSYNCOBJECT::Destroy+0x10b
50 00000000`00000000 : dxgkrnl!DXGGLOBAL::DestroySyncObject+0xff
9760 00000000`00000000 : dxgkrnl!DXGPROTECTEDSESSION::~DXGPROTECTEDSESSION+0xd4
0000 ffffaf0b`40a04c20 : dxgkrnl!DXGPROTECTEDSESSION::`scalar deleting destructor'+0xe
0 01000000`00100000 : dxgkrnl!ADAPTER_DISPLAY::DestroyProtectedSession+0x171
00 ffffaf0b`41334c00 : dxgkrnl!DXGPROTECTEDSESSION::DestroyProtectedSession+0xcc
000000 00000000`00000000 : dxgkrnl!DxgkSharedProtectedSessionObDeleteProcedure+0x77
```

CVE-2020-1258

DXGGLOBAL::CreateSyncObject；

```
mov     ecx, edi
push    ebx
push    [ebp+var_4]
call    ??0DXGSYNCOBJECT@@IAE@PAVDXGGLOBAL@@PAU_D3DDDI_SYNCHRONIZATIONOBJE
mov     edx, [ebp+arg_0]
lea     ecx, [edi+0D8h] ; this
push    edx             ; struct ADAPTER_RENDER *
call    ??0DXGADAPTERSYNCOBJECT@@QAE@PAVADAPTER_RENDER@@@Z ; DXGADAPTERSYN

                        ; CODE XREF: DXGGLOBAL::CreateSyncObject(ADAPTER_R
test    edi, edi
jz      loc_13738A
mov     eax, [edx+8]
mov     al, [eax+6Dh]
mov     [edi+0CDh], al
call    ds:__imp__PsGetCurrentProcess@0 ; PsGetCurrentProcess()
push    eax             ; _DWORD
```

DXGPROTECTEDSESSION::Initialize

```
mov     esi, [ebp+arg_8]
mov     [edi+48h], eax
mov     eax, [ecx+84h]
mov     ecx, [ebp+arg_C]
mov     [edi+4Ch], eax
mov     eax, [ebp+DxgSyncObject]
mov     eax, [eax]
mov     [edi+44h], eax  ; DxgSync->DxgProtectSession+44h
mov     eax, [edx]
mov     [edi+34h], eax
mov     [edi+38h], esi
mov     eax, [ecx]
mov     [edi+3Ch], eax
mov     eax, [ebp+arg_10]
mov     [edi+40h], eax
mov     eax, [ebp+DxgSyncObject]
and     dword ptr [eax], 0
lea     eax, [edi+30h]
and     dword ptr [edx], 0
xor     edx, edx        ; _DWORD
```

CVE-2020-1258

D3DKMTShareObjects/DestroyWindow->

DxgkSharedProtectedSessionObDeleteProcedure->

DXGPROTECTEDSESSION::~DXGPROTECTEDSESSION->

DXGGLOBAL::DestroySyncObject

```
                        ; CODE XREF: DXGPROTECTEDSESSION::~DXGPROTEC
mov     eax, [esi+44h]  ; [esi+44h] is DxgSyncObject
xor     ebx, ebx
test    eax, eax
jz      short loc_4428F
push    ebx             ; unsigned int
push    eax             ; struct DXGSYNCOBJECT *
call    ?GetGlobal@DXGGLOBAL@@SGPAV1@XZ ; DXGGLOBAL::GetGlobal(void)
mov     ecx, eax        ;  Release DxgSyncObject
call    ?DestroySyncObject@DXGGLOBAL@@QAEXPAVDXGSYNCOBJECT@@I@Z ; DX
mov     [esi+44h], ebx

                        ; CODE XREF: DXGPROTECTEDSESSION::~DXGPROTEC
```

CVE-2020-0732

- VIDMM_PROCESS_FENCE_STORAGE::AllocateFenceStorageSlot->ExAllocatePoolWithTag->AllocMemory

```
11
12  v2 = (KSPIN_LOCK *)this;
13  if ( VIDMM_PROCESS_FENCE_STORAGE::FindAvailableFenceStorageSlot(this, a2) )
14    return 0;
15  AllocMemory = (VIDMM_FENCE_STORAGE_PAGE *)ExAllocatePoolWithTag((POOL_TYPE)512, 0x40u, 0x34346956u);
16  if ( AllocMemory )
17    v5 = VIDMM_FENCE_STORAGE_PAGE::VIDMM_FENCE_STORAGE_PAGE(AllocMemory, (struct VIDMM_PROCESS_FENCE_STORAGE *)v2);
18  else
19    v5 = 0;
20  if ( !v5 )
21    return -1073741801;
22  v7 = VIDMM_FENCE_STORAGE_PAGE::Init(v5);      // double free
23  if ( v7 >= 0 )
```

CVE-2020-0732

- VIDMM_FENCE_STORAGE_PAGE::Init->
- VIDMM_FENCE_STORAGE_PAGE::FreeStorage
  - free AllocMemory+12(esi+30)

- VIDMM_FENCE_STORAGE_PAGE::`scalar deleting destr
- ->VIDMM_FENCE_STORAGE_PAGE::FreeStorage
  - free AllocMemory+12(esi+30)

```
void __thiscall VIDMM_FENCE_STORAGE_PAGE::FreeStorage(VIDMM_FENCE_STORAGE_PAGE *this)
{
  VIDMM_FENCE_STORAGE_PAGE *AllocMemory; // esi
  void *v2; // eax
  void *v3; // ecx

  AllocMemory = this;
  if ( *((_BYTE *)this + 52) )
    MmUnlockPages(*((PMDL *)this + 12));
  v2 = (void *)*((_DWORD *)AllocMemory + 12);
  if ( v2 )
    ExFreePoolWithTag(v2, 0);
  if ( *((_DWORD *)AllocMemory + 11) )
  {
    MmUnmapViewInSystemSpace(*((PVOID *)AllocMemory + 11));
    *((_DWORD *)AllocMemory + 11) = 0;
  }
  v3 = (void *)*((_DWORD *)AllocMemory + 10);
  if ( v3 )
  {
    ObfDereferenceObject(v3);
    *((_DWORD *)AllocMemory + 10) = 0;
  }
}
```

CVE-2020-0732

```
1: kd> ub   dxgmms2!VIDMM_FENCE_STORAGE_PAGE::FreeStorage+2a
dxgmms2!VIDMM_FENCE_STORAGE_PAGE::FreeStorage+0xd:
fffff809`69761f35 740a          je      dxgmms2!VIDMM_FENCE_STORAGE_PAGE::FreeStorage+0x19 (fffff809`69761f41)
fffff809`69761f37 488b4958      mov     rcx,qword ptr [rcx+58h]
fffff809`69761f3b ff1547c6fdff  call    qword ptr [dxgmms2!_imp_MmUnlockPages (fffff809`6973e588)]
fffff809`69761f41 488b4b58      mov     rcx,qword ptr [rbx+58h]
fffff809`69761f45 4885c9        test    rcx,rcx
fffff809`69761f48 7408          je      dxgmms2!VIDMM_FENCE_STORAGE_PAGE::FreeStorage+0x2a (fffff809`69761f52)
fffff809`69761f4a 33d2          xor     edx,edx
fffff809`69761f4c ff1526c6fdff  call    qword ptr [dxgmms2!_imp_ExFreePoolWithTag (fffff809`6973e578)]
```

```
STACK_TEXT:
ffffcb81`ae36bda8 fffff802`4c86971a : 00000000`00000013 00000000`000000c4 ffffcb81`ae36bf10 fffff802`4c6a4140 : nt!DbgBreakPointWithStatus
ffffcb81`ae36bdb0 fffff802`4c8690fd : 00000000`00000003 ffffcb81`ae36bf10 fffff802`4c8064e0 00000000`000000c4 : nt!KiBugCheckDebugBreak+0x12
ffffcb81`ae36be10 fffff802`4c7f3ec4 : ffffa98b`b3417a00 fffff802`4c7149a6 ffffa98b`ae1d69d0 00000000`00001000 : nt!KeBugCheck2+0x8a5
ffffcb81`ae36c520 fffff802`4cdb546b : 00000000`000000c4 00000000`00000013 00000000`00001e9e ffffa98b`ae1d69c0 : nt!KeBugCheckEx+0x104
ffffcb81`ae36c560 fffff802`4cd96429 : ffffa98b`ae1d69d0 00000000`00010202 ffffa98b`b32b9d70 fffff809`6978371d : nt!ExFreePoolSanityChecks+0x11b
ffffcb81`ae36c5a0 fffff809`69761f52 : ffffa98b`b32b9d70 ffff820a`dd180fd8 00000000`00000000 ffffa98b`08000000 : nt!VerifierExFreePoolWithTag+0x39
ffffcb81`ae36c5d0 fffff809`69702a7e : ffff820a`dd180ff8 00000000`00000000 ffff820a`dd180fd8 ffff820a`dd180fc0 : dxgmms2!VIDMM_FENCE_STORAGE_PAGE::FreeStorage+0x2a
ffffcb81`ae36c600 fffff809`69715c7e : ffffa98b`b32b9d70 00000000`00001000 00000000`00000000 00000000`00001000 : dxgmms2!VIDMM_FENCE_STORAGE_PAGE::`scalar deleting destructor'+0xe
ffffcb81`ae36c630 fffff809`69761d5f : ffffa98b`b3417a00 00000000`00000001 ffffa98b`8d99f000 00000000`ffffec77 : dxgmms2!VIDMM_PROCESS_FENCE_STORAGE::AllocateFenceStorageSlot+0x13416
ffffcb81`ae36c680 fffff809`69761ccd : ffffa98b`b34179d0 00000000`00000000 ffffa98b`8d99f000 fffff802`4cd95d78 : dxgmms2!VIDMM_GLOBAL::AllocateFenceStorageSlot+0x57
```

Result

- CVE-2020-0622
- CVE-2020-0690
- CVE-2020-0709
- CVE-2020-0714
- CVE-2020-0732
- CVE-2020-0746
- CVE-2020-0888
- CVE-2020-1140
- more.....

Summary&Reflection

- what about other ?
- Increased coverage?
- Next new attack surface?

Thanks!