

# hAFL1:

## Our Journey of Fuzzing Hyper-V and Discovering a 0-Day

---

Peleg Hadar & Ophir Harpaz




# About Us

## Peleg Hadar (@peleghd)

- Senior Security Researcher  
at  SafeBreach
- Windows Internals, vulnerability research, hypervisors
- Former Black Hat USA Speaker (2020)

## Ophir Harpaz (@ophirharpaz)

- Senior Security Researcher  
at  Guardicore
- Excited by low level
- Author of [begin.re](#)



Hyper-V

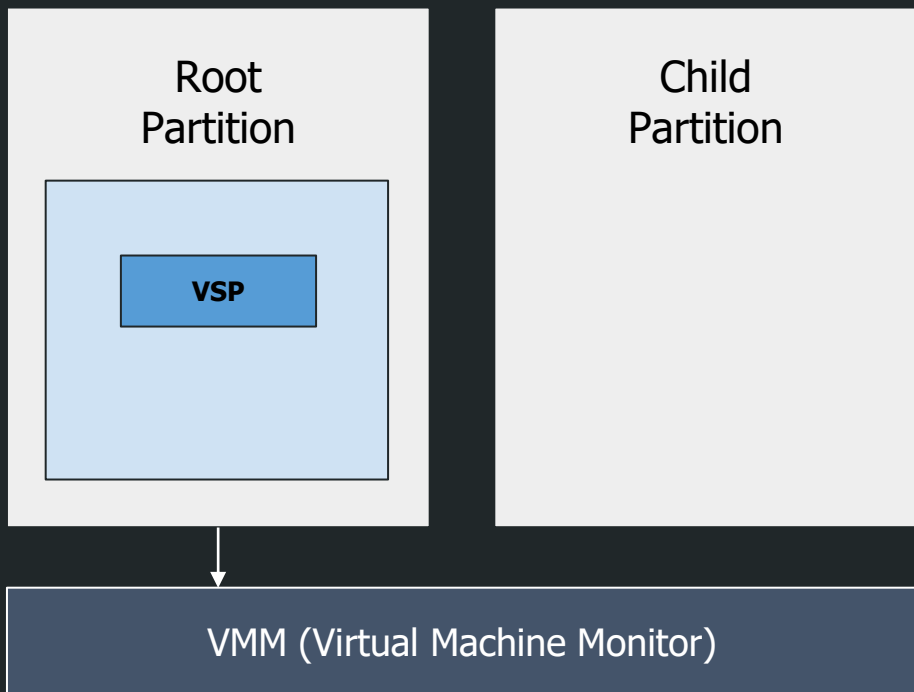


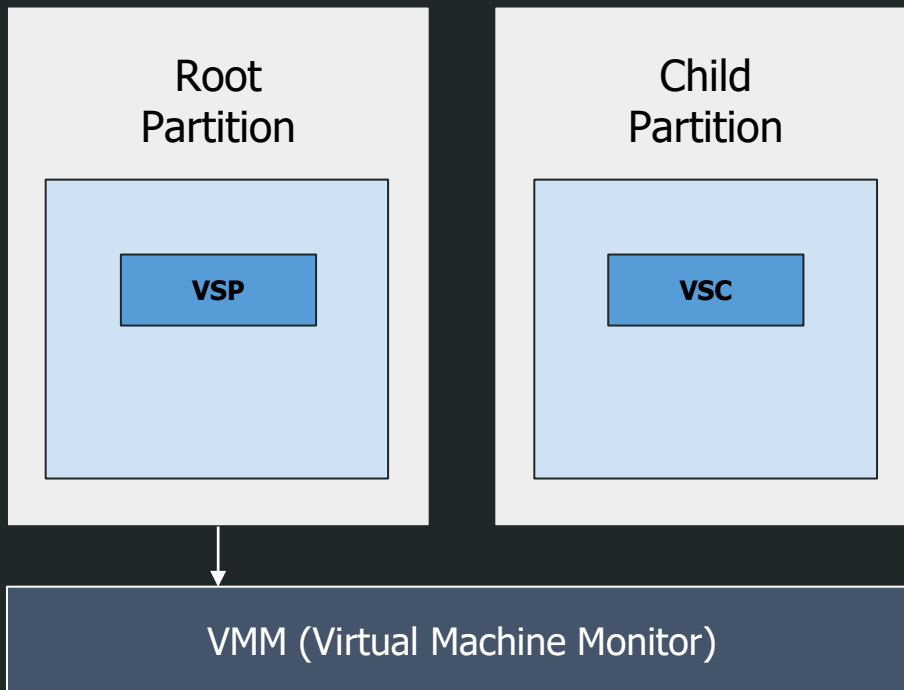
Root  
Partition

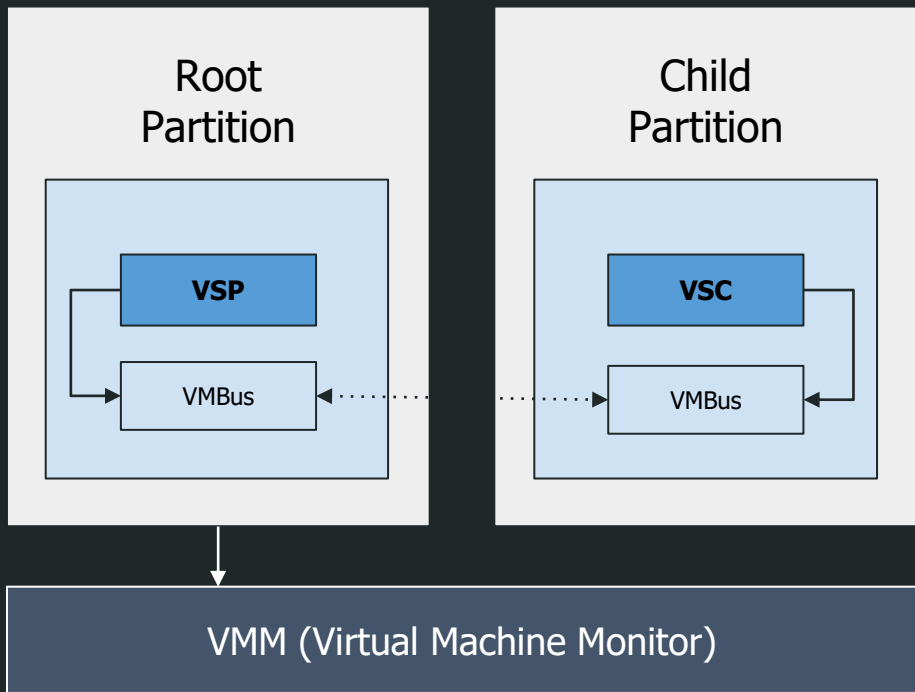
Child  
Partition



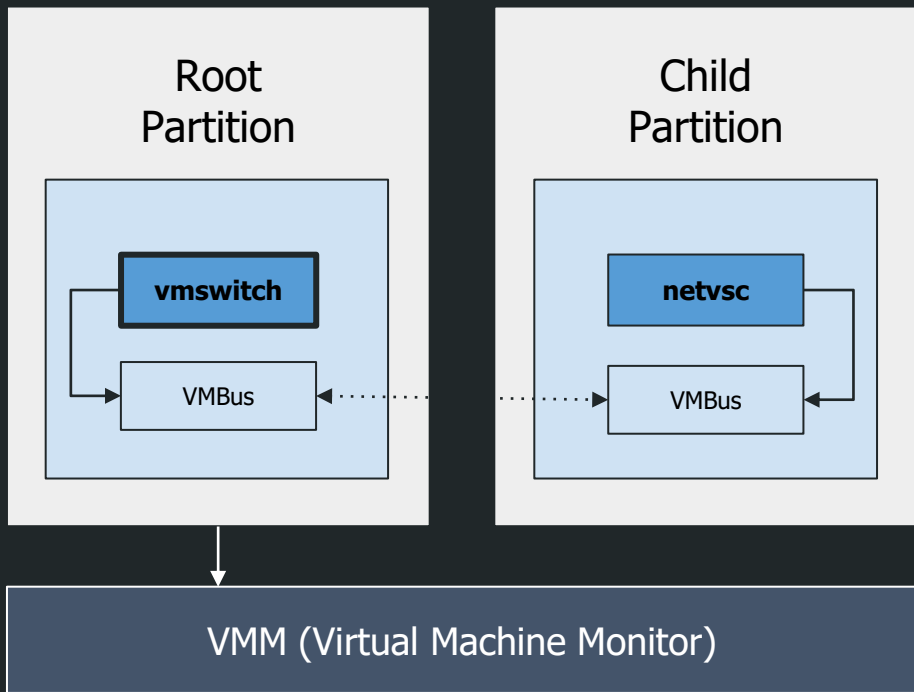
VMM (Virtual Machine Monitor)











Why **fuzzing**?

---

## **Fuzzing Infrastructure**

**Fuzzing Infrastructure**

**Harness**

**Fuzzing Infrastructure**

**Coverage  
Guidance**

**Harness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

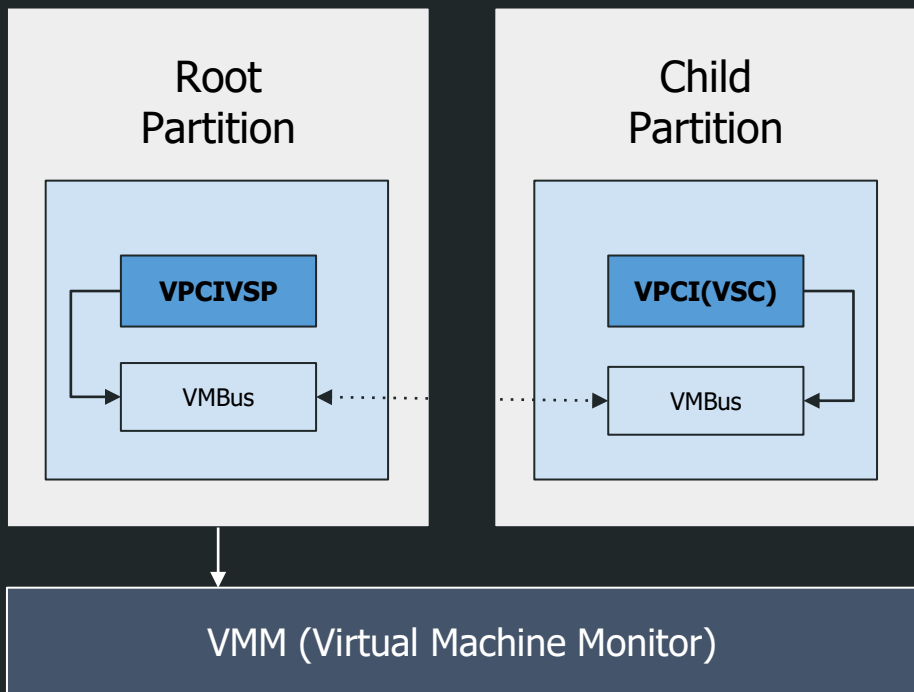
**Crash  
Monitoring**

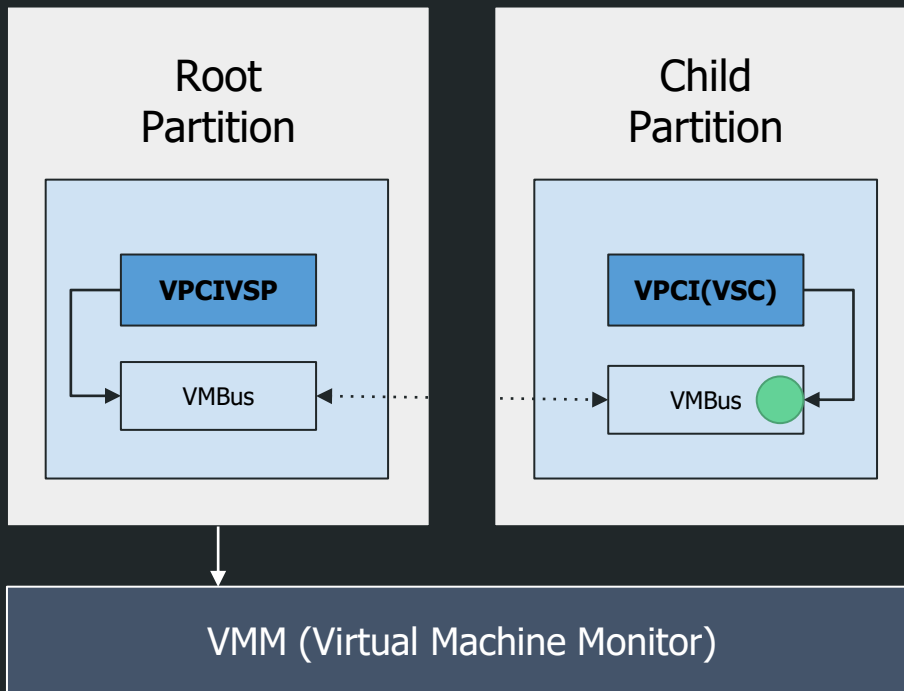
**Structure  
Awareness**

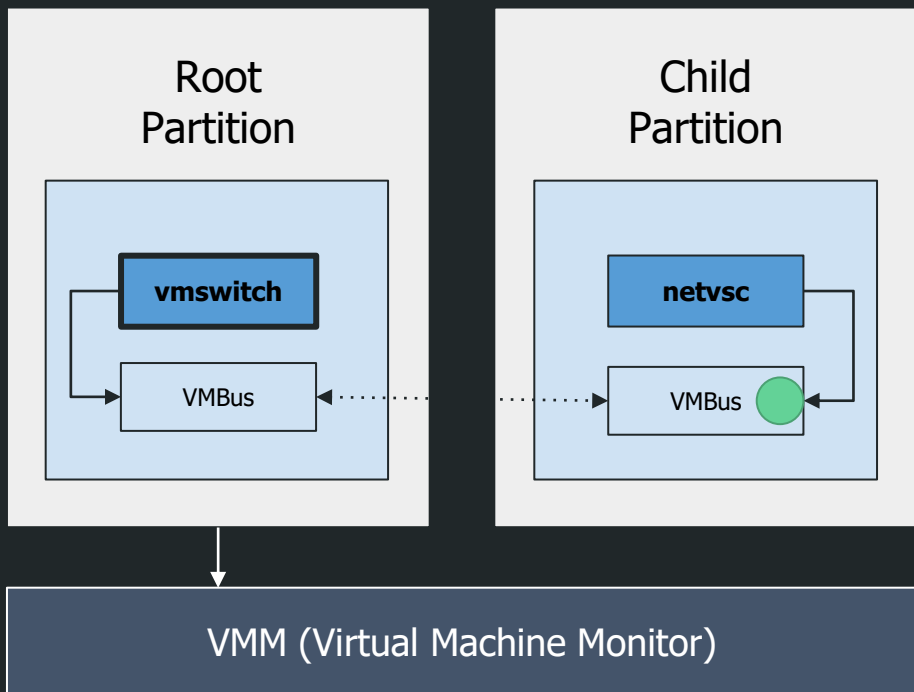


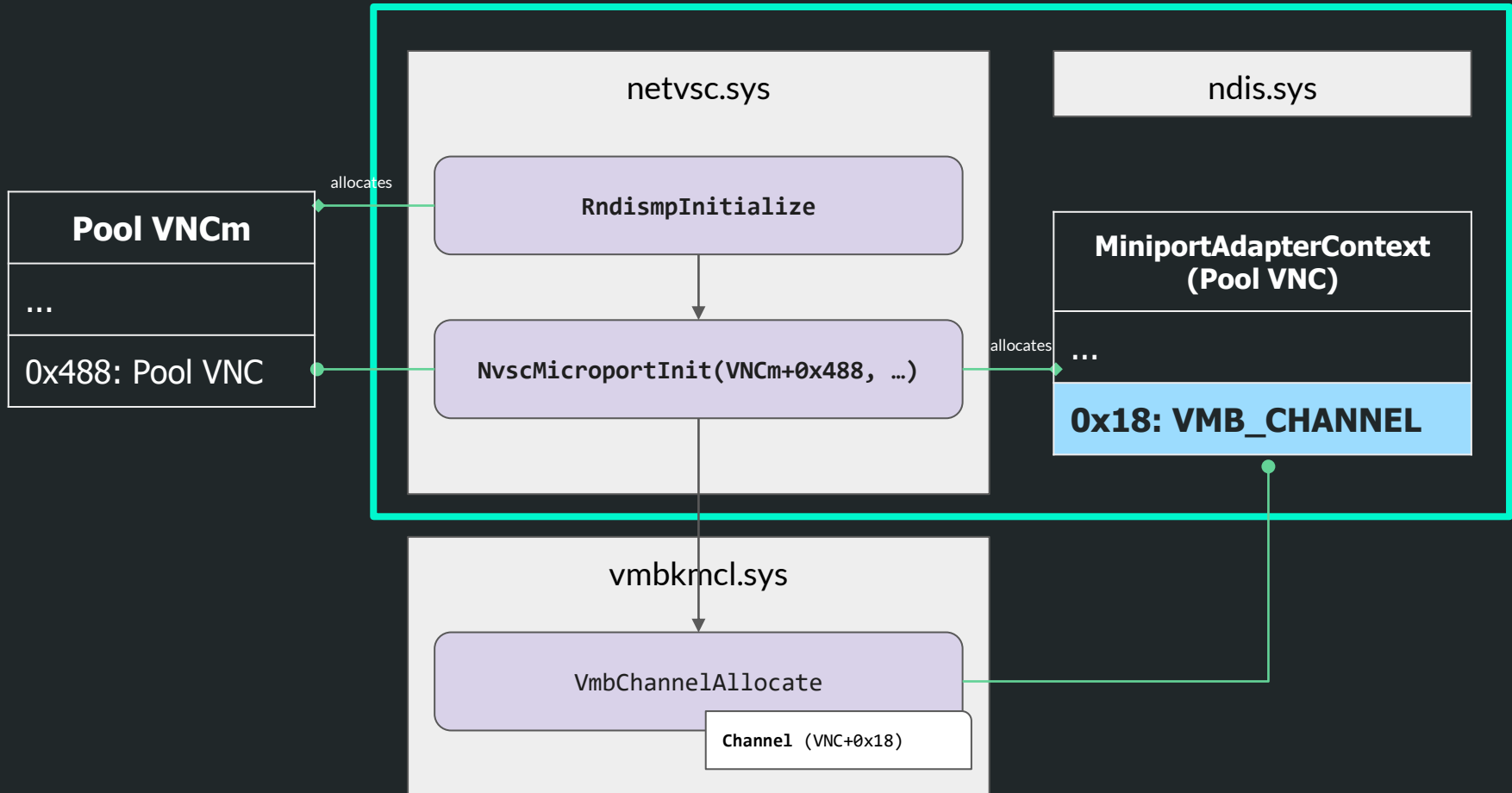
# *Fuzzing Para-Virtualized Devices in Hyper-V*

---









ndis.sys

ndis.sys

ndisMiniportList

ndis.sys

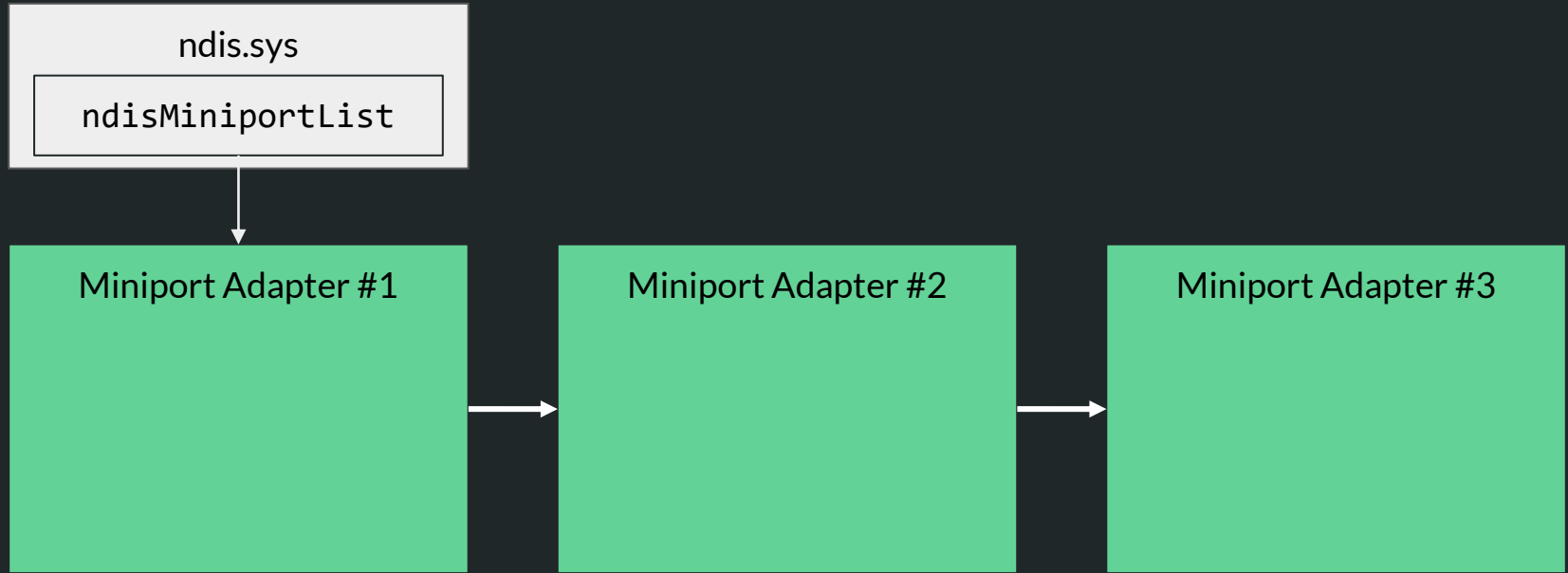
ndisMiniportList

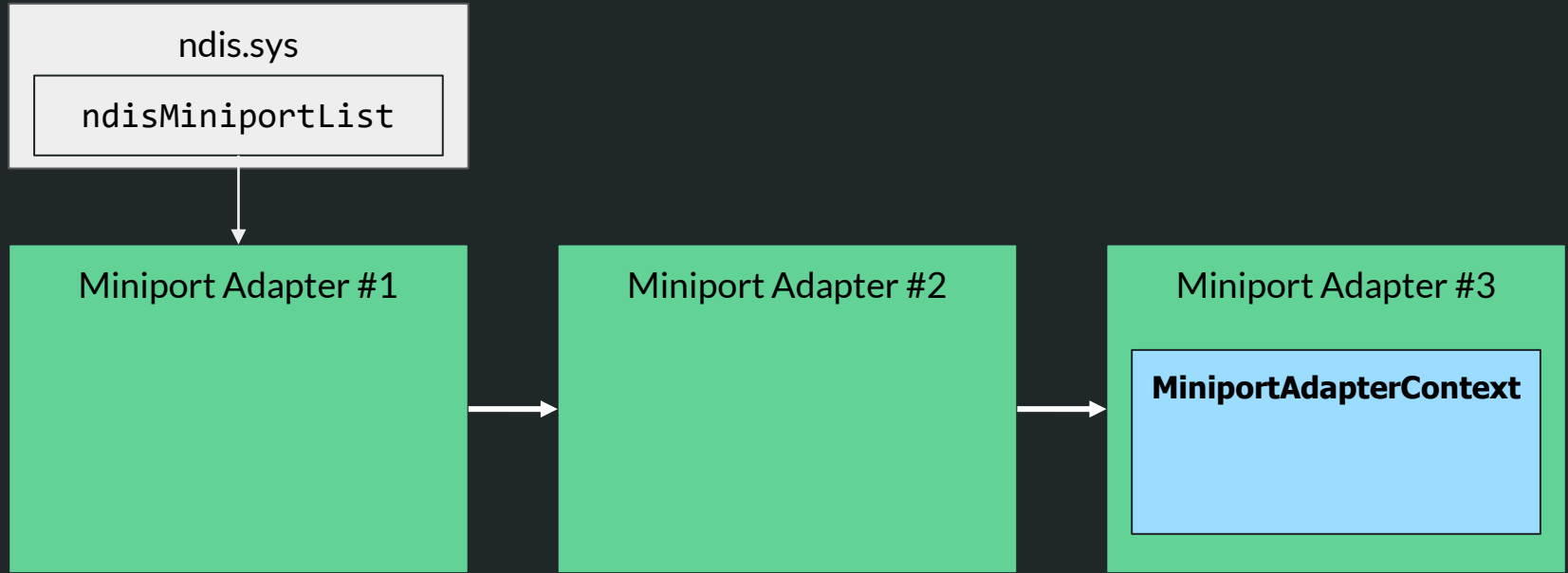


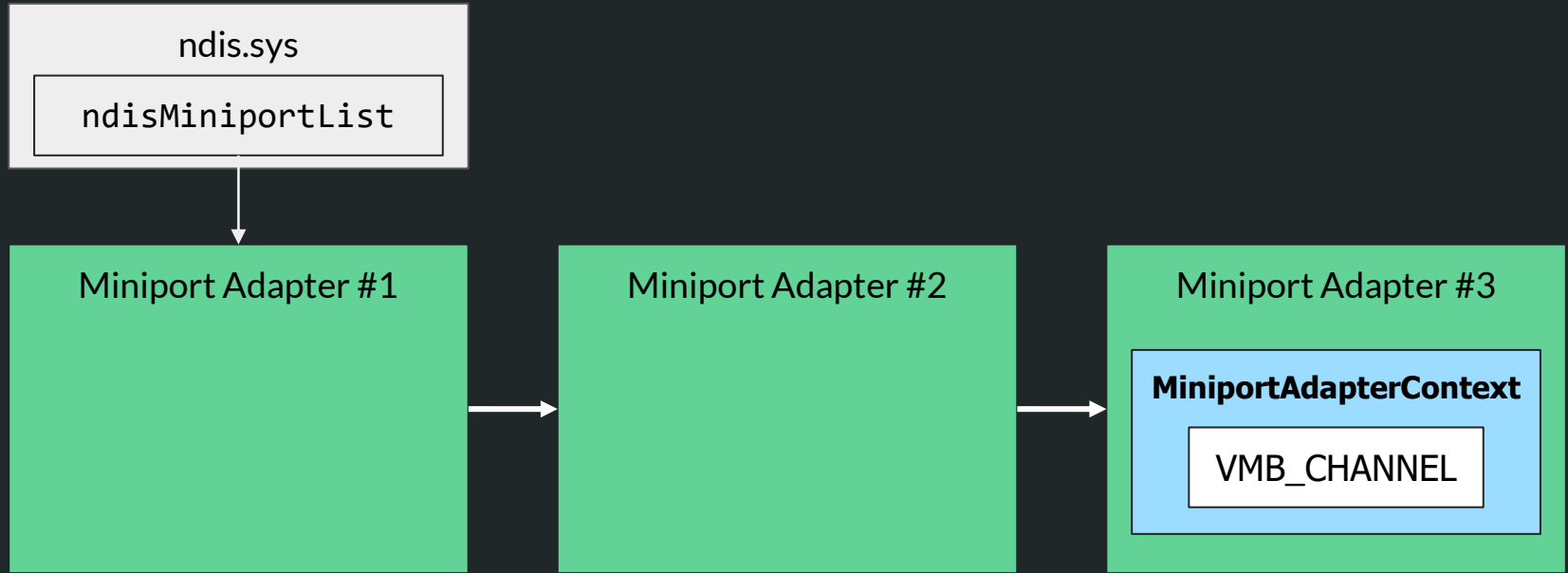
Miniport Adapter #1

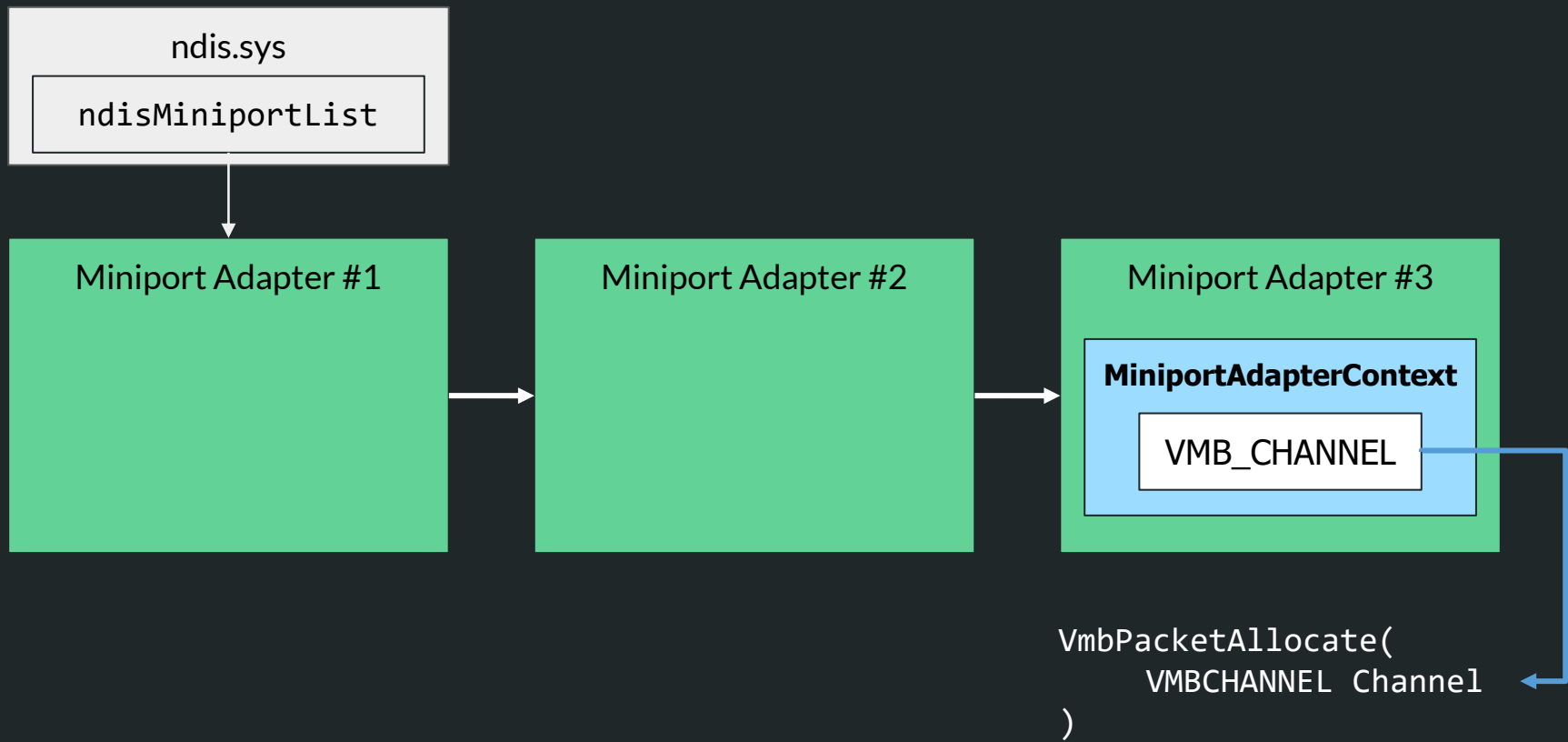












```
for (currMiniport; currMiniport;  
    currMiniport = currMiniport + nextGlobalMiniportOffset))  
{  
    status = NdisMQueryAdapterInstanceName(&currName, currMiniport);  
    stringCompareRes = RtlCompareUnicodeString(&currName, &ourName, caseInsensitive);  
  
    if (stringCompareRes == 0)  
    {  
        currContext = currMiniport->MiniportAdapterContext;  
  
        if (currContext) {  
            PoolVNC = currContext + poolVNCOffset;  
            if (PoolVNC) {  
                channel = PoolVNC + vmbChannelOffset;  
            }  
        }  
        ...  
    }  
}
```

```
for (currMiniport; currMiniport;
     currMiniport = currMiniport + nextGlobalMiniportOffset))
{
    status = NdisMQueryAdapterInstanceName(&currName, currMiniport);
    stringCompareRes = RtlCompareUnicodeString(&currName, &ourName, caseInsensitive);

    if (stringCompareRes == 0)
    {
        currContext = currMiniport->MiniportAdapterContext;

        if (currContext) {
            PoolVNC = currContext + poolVNCOffset;
            if (PoolVNC) {
                channel = PoolVNC + vmbChannelOffset;
            }
        }
        ...
    }
}
```

```
for (currMiniport; currMiniport;  
    currMiniport = currMiniport + nextGlobalMiniportOffset))  
{  
    status = NdisMQueryAdapterInstanceName(&currName, currMiniport);  
    stringCompareRes = RtlCompareUnicodeString(&currName, &ourName, caseInsensitive);  
  
    if (stringCompareRes == 0)  
    {  
        currContext = currMiniport->MiniportAdapterContext;  
  
        if (currContext) {  
            PoolVNC = currContext + poolVNCOffset;  
            if (PoolVNC) {  
                channel = PoolVNC + vmbChannelOffset;  
            }  
        }  
        ...  
    }  
}
```



```
for (currMiniport; currMiniport;
     currMiniport = currMiniport + nextGlobalMiniportOffset))
{
    status = NdisMQueryAdapterInstanceName(&currName, currMiniport);
    stringCompareRes = RtlCompareUnicodeString(&currName, &ourName, caseInsensitive);

    if (stringCompareRes == 0)
    {
        currContext = currMiniport->MiniportAdapterContext;

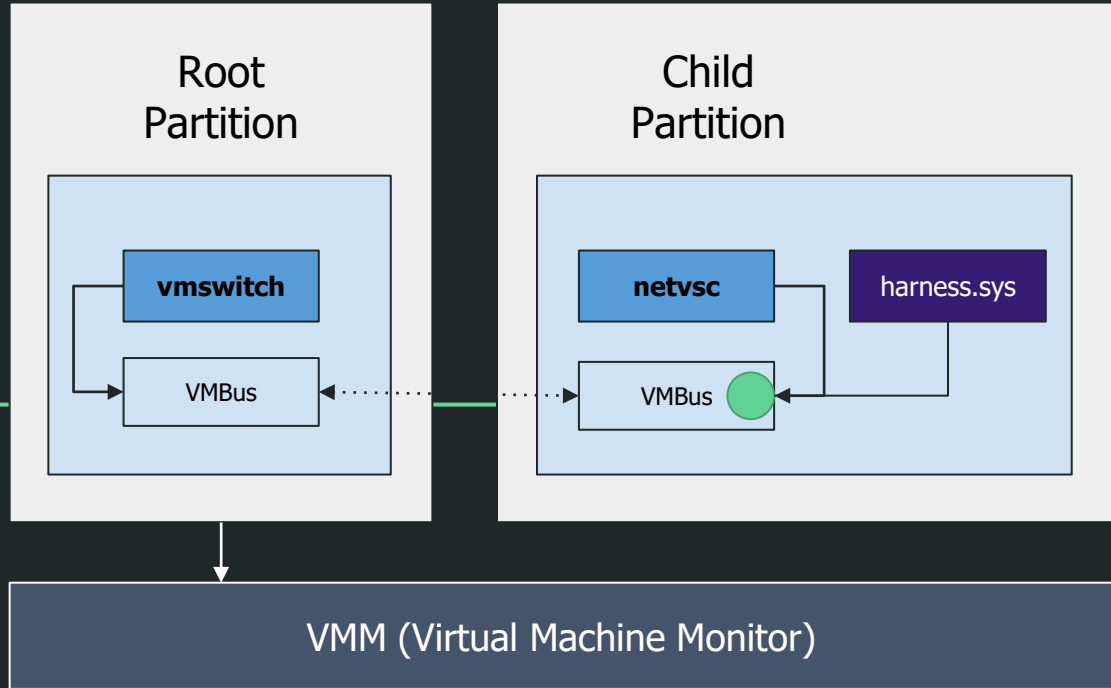
        if (currContext) {
            PoolVNC = currContext + poolVNCOffset;
            if (PoolVNC) {
                channel = PoolVNC + vmbChannelOffset;
            }
        }
        ...
    }
}
```

```
for (currMiniport; currMiniport;
     currMiniport = currMiniport + nextGlobalMiniportOffset))
{
    status = NdisMQueryAdapterInstanceName(&currName, currMiniport);
    stringCompareRes = RtlCompareUnicodeString(&currName, &ourName, caseInsensitive);

    if (stringCompareRes == 0)
    {
        currContext = currMiniport->MiniportAdapterContext;

        if (currContext) {
            PoolVNC = currContext + poolVNCOffset;
            if (PoolVNC) {
                channel = PoolVNC + vmbChannelOffset;
            }
        }
        ...
    }
}
```

Now to  
*vmswitch's* side.



The *EvtVmbChannelProcessPacket* callback function is invoked when a packet arrives in the incoming ring buffer.

## Syntax

C++

 Copy

```
EVT_VMB_CHANNEL_PROCESS_PACKET EvtVmbChannelProcessPacket;  
  
void EvtVmbChannelProcessPacket(  
    VMBCHANNEL Channel,  
    VMBPACKETCOMPLETION Packet,  
    PVOID Buffer,  
    UINT32 BufferLength,  
    UINT32 Flags  
)  
{...}
```

The *EvtVmbChannelProcessPacket* callback function is invoked when a packet arrives in the incoming ring buffer.

## Syntax

C++

 Copy

```
EVT_VMB_CHANNEL_PROCESS_PACKET EvtVmbChannelProcessPacket;
```

```
void EvtVmbChannelProcessPacket(  
    VMBCHANNEL Channel,  
    VMBPACKETCOMPLETION Packet,  
    PVOID Buffer,  
    UINT32 BufferLength,  
    UINT32 Flags  
)  
{...}
```

```
    VmbChannelInitSetProcessPacketCallbacks(*Channel,  
    VmsVmNicPvtKmclProcessPacket,  
    VmsVmNicPvtKmclProcessingComplete);
```

# Packet Types

- NVSP
  - Initialization
  - Control and management
  - **RNDIS**
    - Communication between host and network adapter

```
enum {
    NVSP_MSG_TYPE_NONE = 0,
    /* Init Messages */
    NVSP_MSG_TYPE_INIT = 1,
    NVSP_MSG_TYPE_INIT_COMPLETE = 2,
    NVSP_VERSION_MSG_START = 100,
    /* Version 1 Messages */
    NVSP_MSG1_TYPE_SEND_NDIS_VER = NVSP_VERSION_MSG_START,
    NVSP_MSG1_TYPE_SEND_RECV_BUF,
    NVSP_MSG1_TYPE_SEND_RECV_BUF_COMPLETE,
    NVSP_MSG1_TYPE_REVOKE_RECV_BUF,
    NVSP_MSG1_TYPE_SEND_SEND_BUF,
    NVSP_MSG1_TYPE_SEND_SEND_BUF_COMPLETE,
    NVSP_MSG1_TYPE_REVOKE_SEND_BUF,
    NVSP_MSG1_TYPE_SEND_RNDIS_PKT,
    NVSP_MSG1_TYPE_SEND_RNDIS_PKT_COMPLETE,
    /* Version 2 messages */
    NVSP_MSG2_TYPE_SEND_CHIMNEY_DELEGATED_BUF,
    NVSP_MSG2_TYPE_SEND_CHIMNEY_DELEGATED_BUF_COMP,
    NVSP_MSG2_TYPE_REVOKE_CHIMNEY_DELEGATED_BUF,
    NVSP_MSG2_TYPE_RESUME_CHIMNEY_RX_INDICATION,
    // SNIP
    NVSP_MSG2_MAX = NVSP_MSG2_TYPE_ALLOC_CHIMNEY_HANDLE_COMP,
    /* Version 4 messages */
    NVSP_MSG4_TYPE_SEND_VF_ASSOCIATION,
    NVSP_MSG4_TYPE_SWITCH_DATA_PATH,
    NVSP_MSG4_TYPE_UPLINK_CONNECT_STATE_DEPRECATED,
    NVSP_MSG4_MAX = NVSP_MSG4_TYPE_UPLINK_CONNECT_STATE_DEPRECATED,
    /* Version 5 messages */
    NVSP_MSG5_TYPE_OID_QUERY_EX,
    NVSP_MSG5_TYPE_OID_QUERY_EX_COMP,
    NVSP_MSG5_TYPE_SUBCHANNEL,
    NVSP_MSG5_TYPE_SEND_INDIRECTION_TABLE,
    NVSP_MSG5_MAX = NVSP_MSG5_TYPE_SEND_INDIRECTION_TABLE,
    /* Version 6 messages */
    NVSP_MSG6_TYPE_PD_API,
    NVSP_MSG6_TYPE_PD_POST_BATCH,
    NVSP_MSG6_MAX = NVSP_MSG6_TYPE_PD_POST_BATCH
};
```



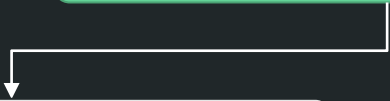
VmsVmNicPvtKmc1ProcessPacket



VmsVmNicPvtKmc1ProcessPacket

NVSP\_MSG\_TYPE\_INIT

VmsVmNicPvtHandleInitMessage



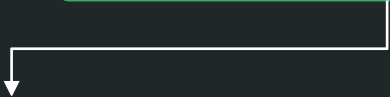




VmsVmNicPvtKmc1ProcessPacket

NVSP\_MSG\_TYPE\_INIT

VmsVmNicPvtHandleInitMessage





VmsVmNicPvtKmc1ProcessPacket

NVSP\_MSG\_TYPE\_INIT

VmsVmNicPvtHandleInitMessage

NVSP\_MSG1\_TYPE\_SEND\_RNDIS\_PKT

VmsVmNicPvtVersion1HandleRndisSendMessage



VmsVmNicPvtKmc1ProcessPacket

NVSP\_MSG\_TYPE\_INIT

VmsVmNicPvtHandleInitMessage

NVSP\_MSG1\_TYPE\_SEND\_RNDIS\_PKT

VmsVmNicPvtVersion1HandleRndisSendMessage

NVSP\_MSG\_TYPE\_\*

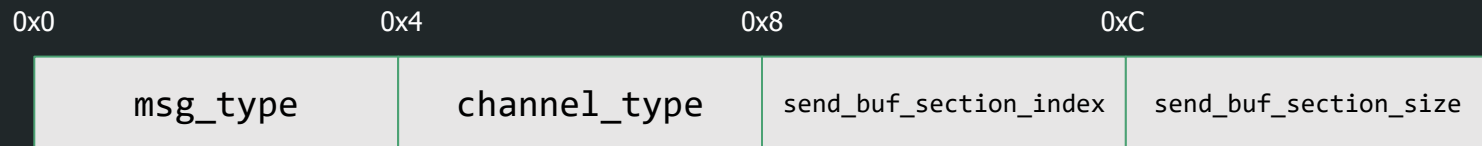
MessageHandlerTable[msg\_type]





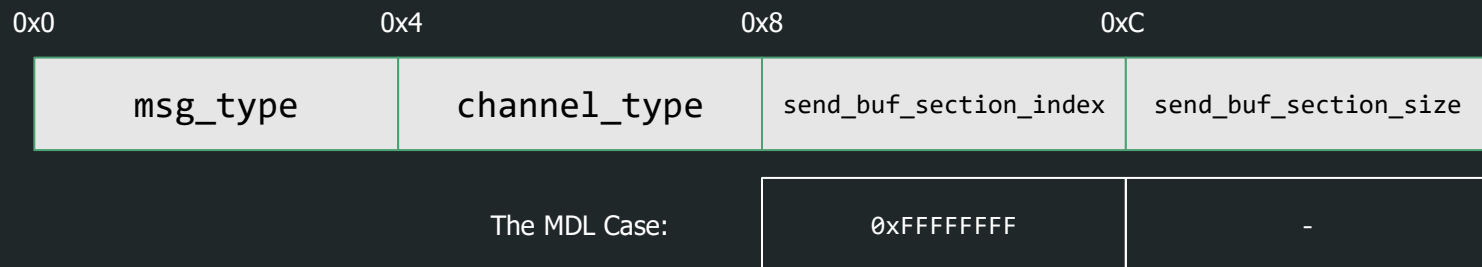
```
VmsVmNicPvtVersion1HandleRndisSendMessage(VMBCHANNEL pChannel,  
void* Buffer,  
PMDL pMdl,  
void* PacketCompletionContext)
```

# Buffer



- RNDIS data can be transmitted via either:
  - The VMBus's send buffer

# Buffer



- RNDIS data can be transmitted via either:
  - The VMBus's send buffer
  - **An MDL - ✓**

# Using MDLs

```
VmsVmNicPvtVersion1HandleRndisSendMessage(VMBCHANNEL pChannel,  
void* Buffer,  
PMDL pMdl,  
void* PacketCompletionContext)
```

Buffer:

NVSP_MSG1_TYPE_SEND_RNDIS_PKT	0x1	0xFFFFFFFF	0x0
-------------------------------	-----	------------	-----

MDL (RNDIS Message):

MessageType	rndis_message_container
-------------	-------------------------

# Testing Our Harness

## CVE-2019-0717

```
*** Fatal System Error: 0x00000050
                                (0xFFFF9F0903203000,0x0000000000000000,0xFFFFF80124E1E723,0x0000000000000002)
```

```
Driver at fault:
```

```
*** vmswitch.sys - Address FFFFF80124E1E723 base at FFFFF80124DD0000, DateStamp 5eeb51be
```

```
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.
```

```
For analysis of this file, run !analyze -v
```

```
nt!DbgBreakPointWithStatus:
```

```
fffff801`21e69bd0 cc int 3
```

```
1: kd> kp
```

#	Child-SP	RetAddr	Call Site
<a href="#">00</a>	ffffe98c`cefbcc38	fffff801`21f3ebe2	nt!DbgBreakPointWithStatus
<a href="#">01</a>	ffffe98c`cefbcc40	fffff801`21f3e367	nt!KiBugCheckDebugBreak+0x12
<a href="#">02</a>	ffffe98c`cefbcca0	fffff801`21e62057	nt!KeBugCheck2+0x957
<a href="#">03</a>	ffffe98c`cefbdb3c0	fffff801`21f05731	nt!KeBugCheckEx+0x107
<a href="#">04</a>	ffffe98c`cefbdb400	fffff801`21d762e7	nt!MiSystemFault+0x12ffa1
<a href="#">05</a>	ffffe98c`cefbdb540	fffff801`21e6fac9	nt!MmAccessFault+0x327
<a href="#">06</a>	ffffe98c`cefbdb6e0	fffff801`24e1e723	nt!KiPageFault+0x349
<a href="#">07</a>	ffffe98c`cefbdb878	fffff801`24df6da1	vmswitch!lmemcmp+0x93
<a href="#">08</a>	ffffe98c`cefbdb880	fffff801`24df5ed1	vmswitch!VmsMpCommonPvtSetRequestCommon+0x175
<a href="#">09</a>	ffffe98c`cefbdb930	fffff801`24df59f3	vmswitch!VmsMpCommonSetRequest+0xa5
<a href="#">0a</a>	ffffe98c`cefbdb9b0	fffff801`24e174b9	vmswitch!VmcVmNicPvtRndisDeviceSetRequest+0x92
<a href="#">0b</a>	ffffe98c`cefbdba20	fffff801`24e0e457	vmswitch!RndisDevHostHandleSetMessage+0xb9
<a href="#">0c</a>	ffffe98c`cefbdbd00	fffff801`21e69bd0	vmswitch!RndisDevHostControlMessageHandleRoutine+0x107



## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

# What We Expected

# What We Expected

KVM (L0)

kAFL

# What We Expected

KVM (L0)

kAFL

Hyper-V (L1) - Host

# What We Expected

KVM (L0)

kAFL

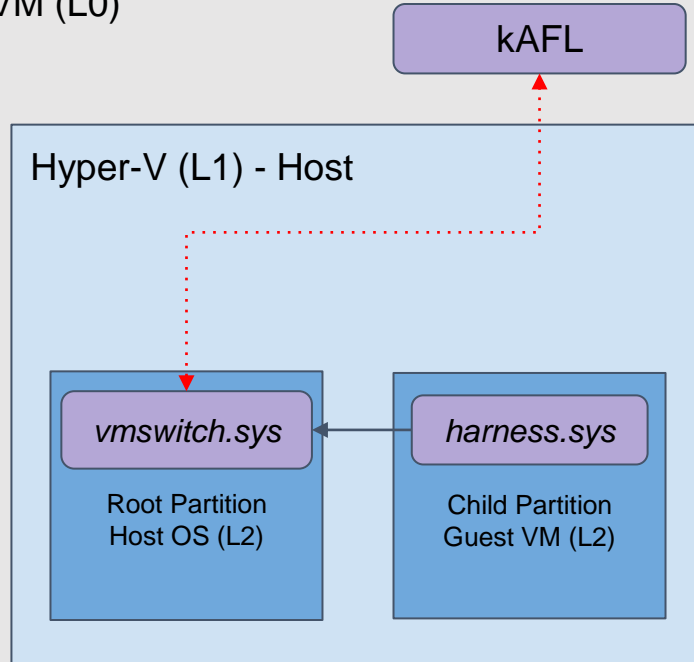
Hyper-V (L1) - Host

*vmswitch.sys*

*harness.sys*

Root Partition  
Host OS (L2)

Child Partition  
Guest VM (L2)



# Problem: Nested Virtualization

- L2 ↔ L0
  - MSR\_IA32\_RTIT\_CTL (Intel PT)
  - SHM / Nested GPA R/W
  - Hypercall handling
- To be continued :)



**Fuzzing from L2**

# Fuzzing from L1



# Solution: Fuzzing From L1

- VT-x is **disabled** and yet *vmswitch.sys* is **loaded**

# Solution: Fuzzing From L1

- VT-x is **disabled** and yet *vmswitch.sys* is **loaded**
- Our harness driver will call `*HandleRndisSendMessage` function with our fuzzing inputs

```
VmsVmNicPvtVersion1HandleRndisSendMessage(VMBCHANNEL pChannel,  
void* Buffer,  
PMDL pMdl,  
void* PacketCompletionContext)
```

# No VMBus channel exists

**No working partitions**



**VMBus is not initialized**



**No valid VMB\_CHANNEL pointer on *vmswitch's***

**and**

# No VMBus channel exists

**No working partitions**



**VMBus is not initialized**



**No valid VMB\_CHANNEL pointer on *vmswitch's* end**

```
VmsVmNicPvtVersion1HandleRndisSendMessage(VMBCHANNEL pChannel,  
void* Buffer,  
PMDL pMdl,  
void* PacketCompletionContext)
```

# Initializing a Channel

- Luckily, `VmsVmNicMorph` initializes most of the structures for us, including a VMBus Channel

# Huston, we have a problem!

- Calling `VmsVmNicMorph` crashed the system!



# Huston, we have a problem!

- Calling `VmsVmNicMorph` crashed the system!
- Patching out VMBus-related logic
  - Fortunately, VMBus doesn't interfere with the data
  - We had to disable PatchGuard

# Disabling *PatchGuard*

## Setting **KdDebuggerEnabled**

might cause performance  
overhead

---

Option #1

# Disabling *PatchGuard*

## Setting **KdDebuggerEnabled**

might cause performance  
overhead

---

Option #1

## Using **EfiGuard**

An open-source UEFI bootkit which  
disables PatchGuard & Driver  
Signature Enforcement

---

Option #2

## What we did

KVM (L0)

## What we did

KVM (L0)

kAFL

# What we did

KVM (L0)

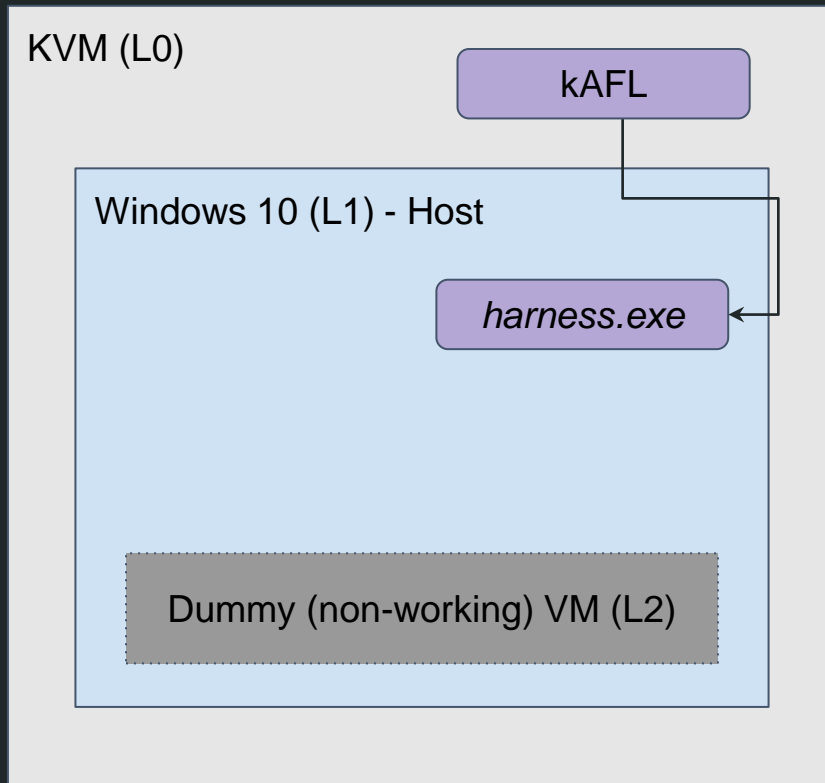
kAFL

Windows 10 (L1) - Host

Dummy (non-working) VM (L2)

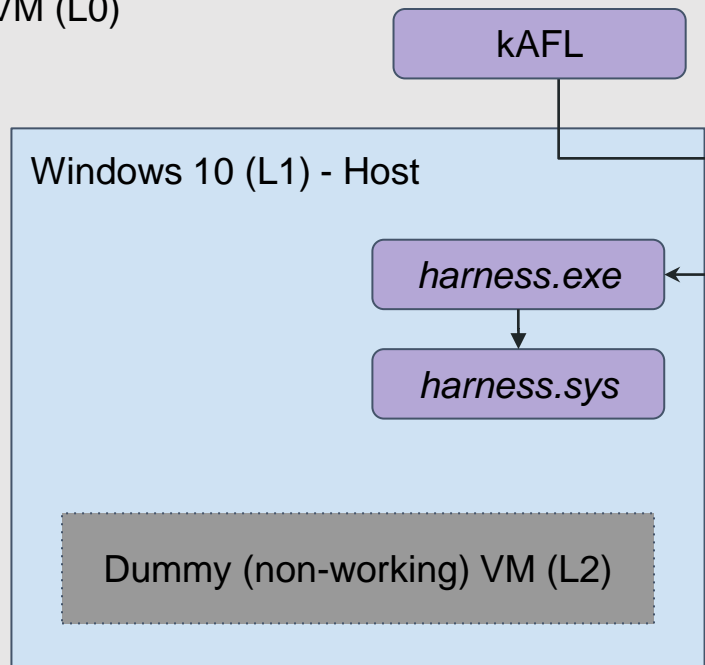


# What we did



# What we did

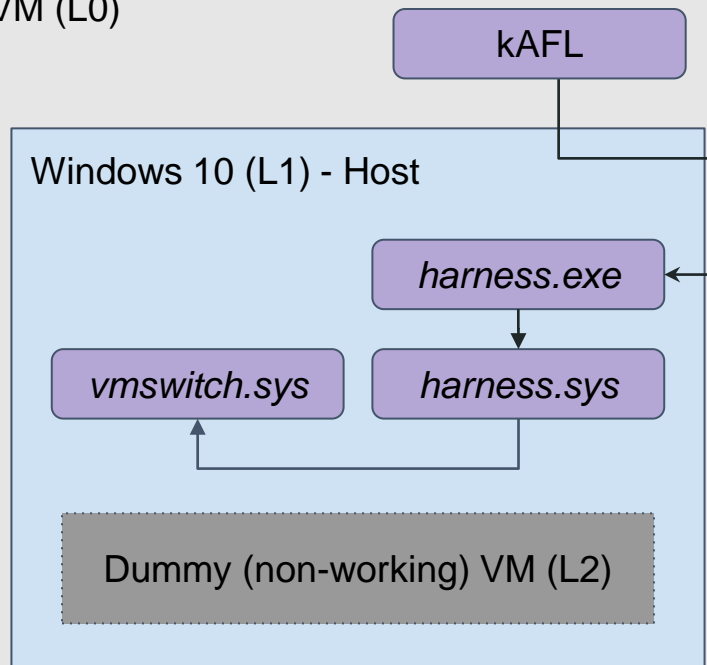
KVM (L0)





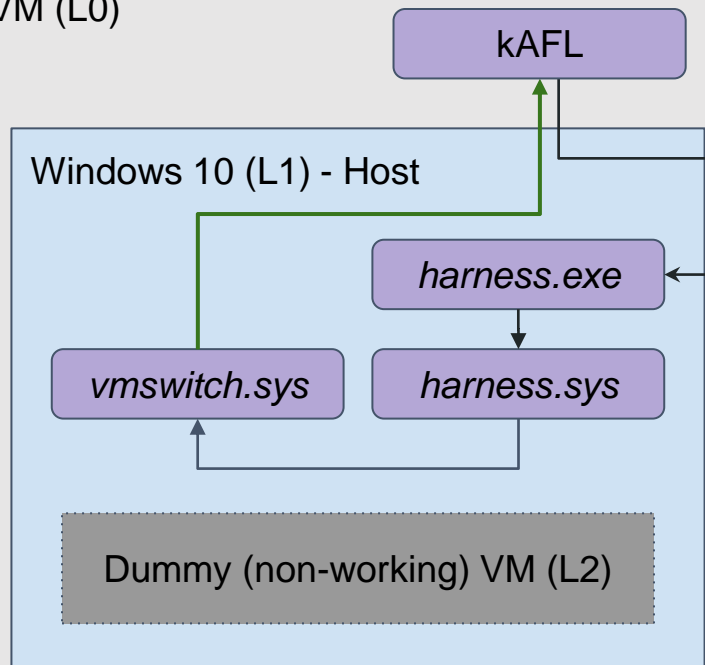
# What we did

KVM (L0)



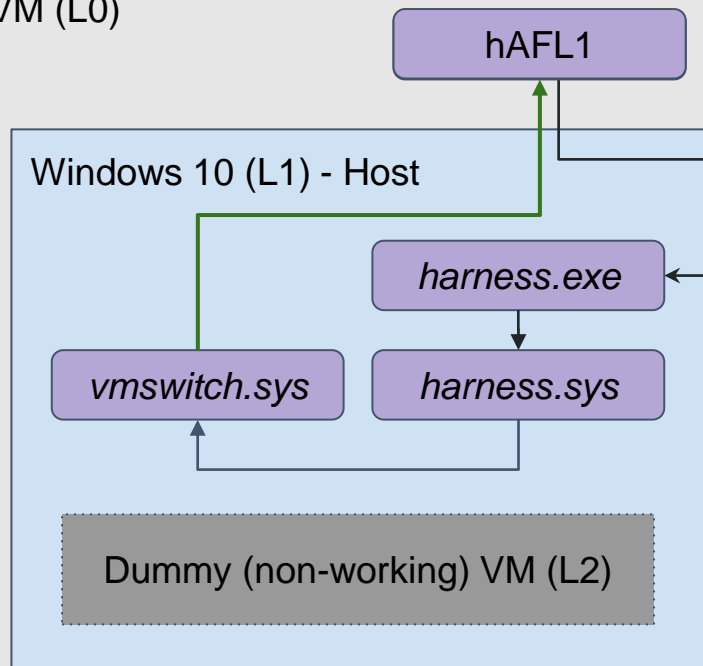
# What we did

KVM (L0)



# What we did

KVM (L0)



## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## Fuzzing Infrastructure

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

# Coverage Guidance

- kAFL supports coverage-guidance out-of-the-box
  - Using Intel PT
  - Per-process coverage using CR3 filtering

# Coverage Guidance

- **Problem:** Low basic-block count



# Coverage Guidance

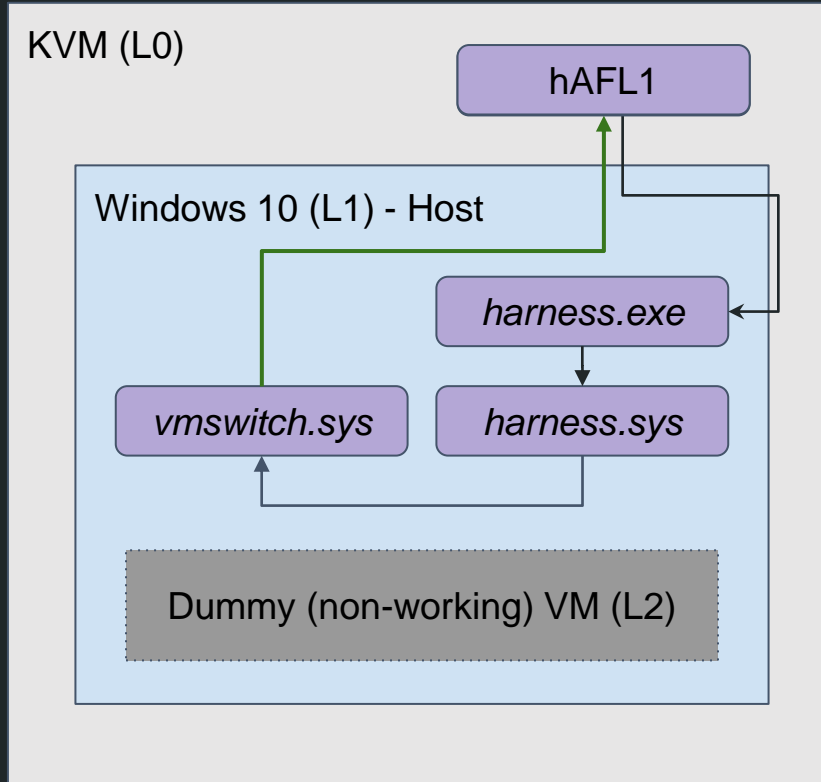
- **Problem:** Low basic-block count
- *vmswitch* processes incoming packets in a multithreaded manner

# VMSwitch Multithreaded Processing

CR3: 0x0000000000000000

Intel PT: ON

Context: harness.exe

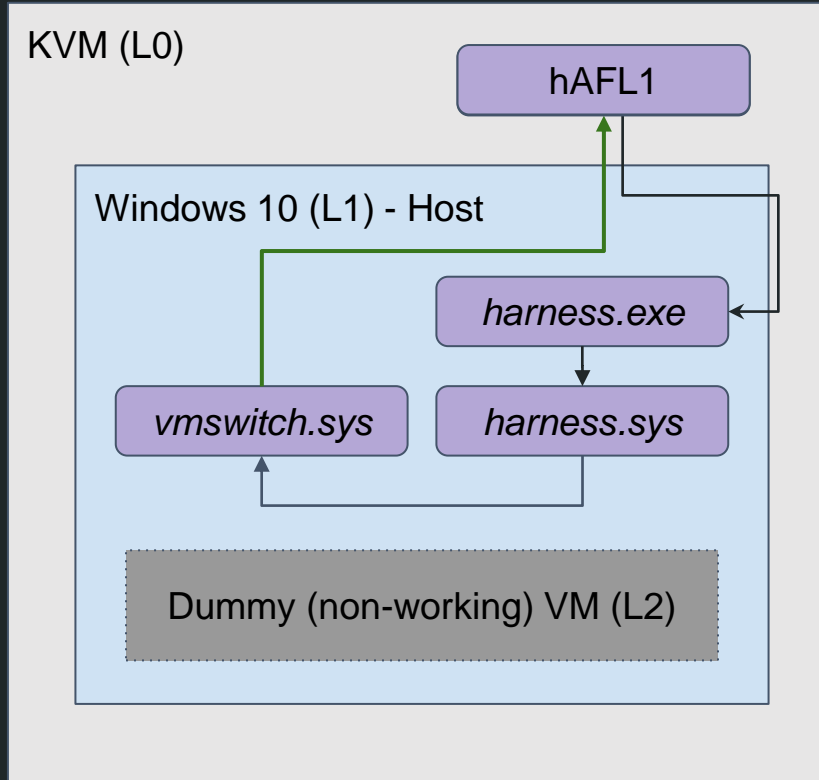


# VMSwitch Multithreaded Processing

CR3: 0x0000000000000000

Intel PT: ON

Context: harness.exe

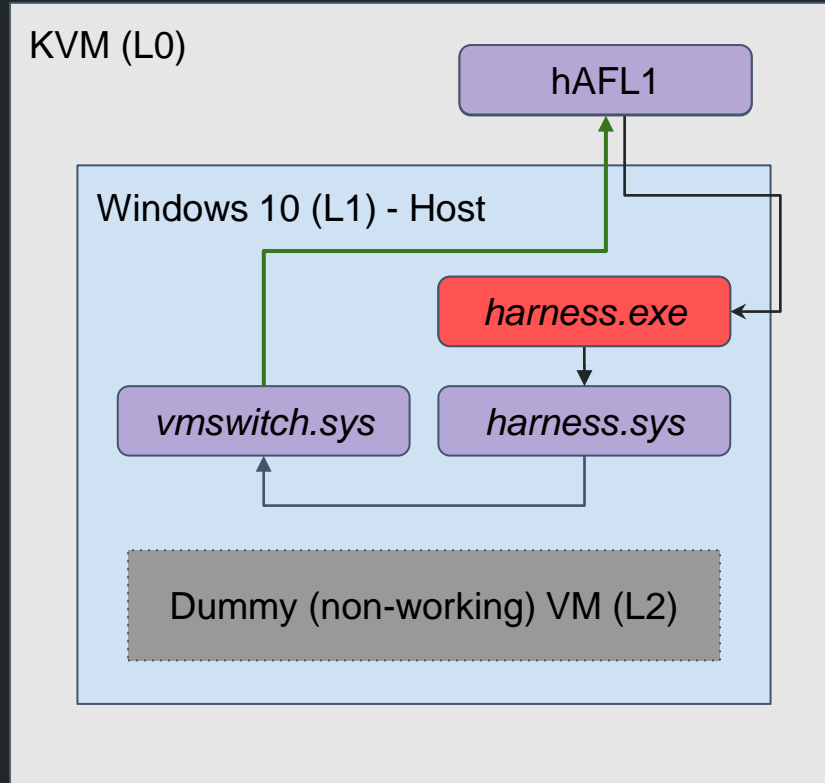


# VMSwitch Multithreaded Processing

CR3: 0x0000000000000000

Intel PT: ON

Context: harness.exe

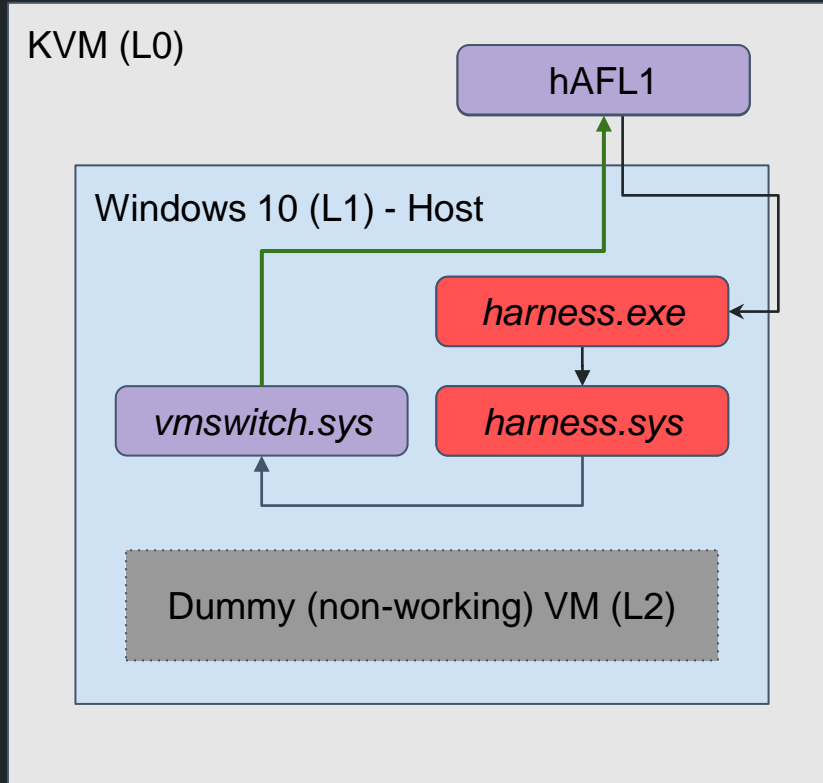


# VMSwitch Multithreaded Processing

CR3: 0x0000000000000000

Intel PT: ON

Context: harness.exe



# VMSwitch Multithreaded Processing

KVM (L0)

hAFL1

Windows 10 (L1) - Host

*harness.exe*

*vmswitch.sys*

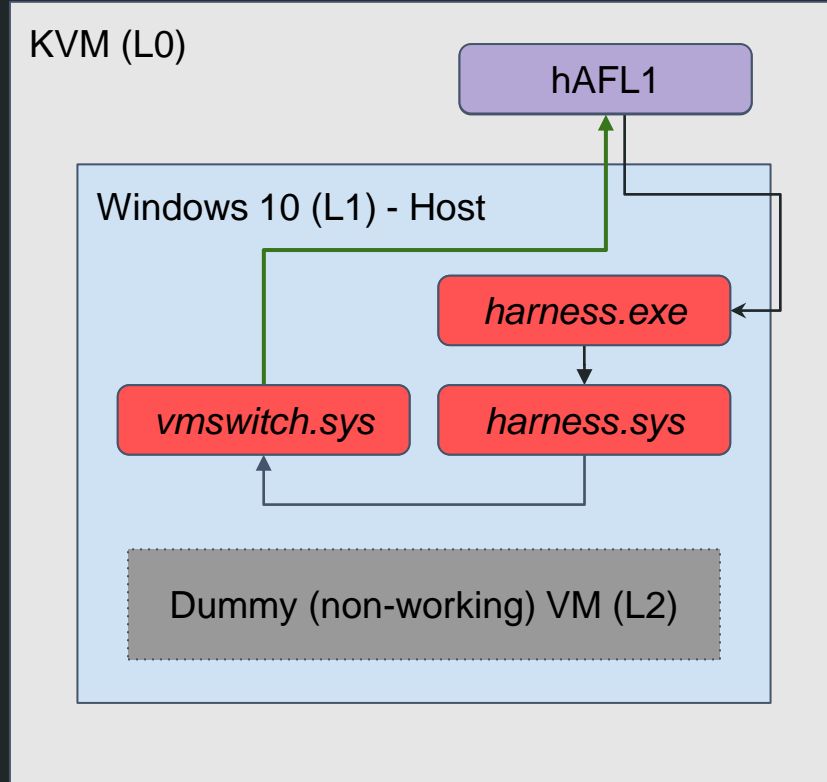
*harness.sys*

Dummy (non-working) VM (L2)

CR3: 0x0000000000000000

Intel PT: ON

Context: harness.exe

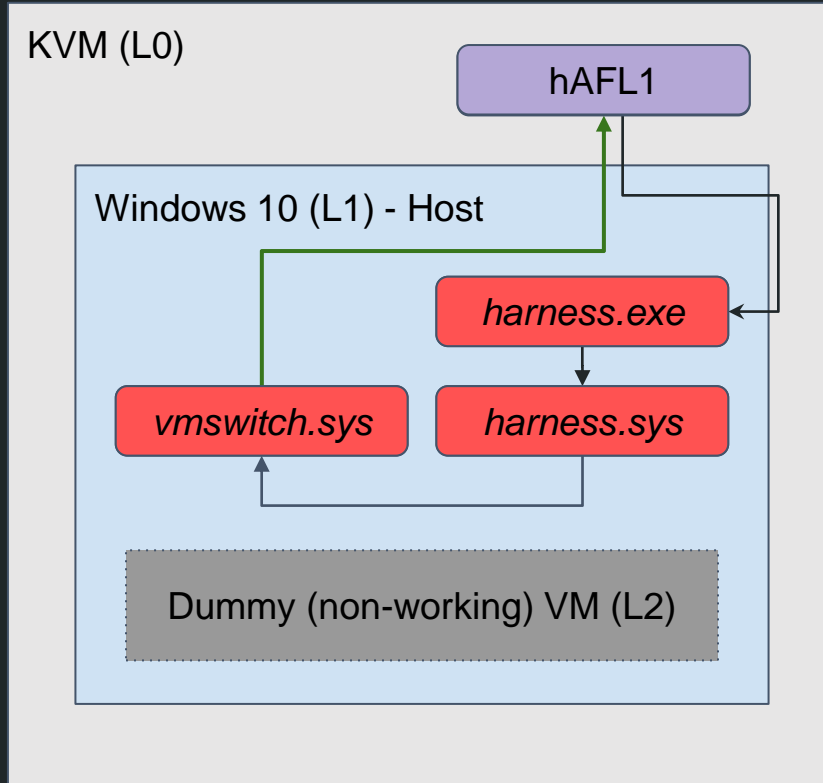


# VMSwitch Multithreaded Processing

CR3: 0x11111111111111111111111111111111

Intel PT: OFF

Context: System



# Coverage Guidance

- **Solution:** disable Intel-PT CR3 filtering
  - This enabled us to monitor all executions in *vmswitch* address region





## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

# Crash Monitoring

- Registered a BugCheck callback to send crash's stack trace
  - Based on Xen's implementation
- Module name is retrieved by maintaining a linked list of loaded modules

```
case KVM_EXIT_KAFL_CRASH_DUMP:  
    handle_hypercall_kafl_crash_dump(run, cpu);  
    ret = 0;  
    break;  
case KVM_EXIT_KAFL_CRASH_DUMP_SIZE:  
    handle_hypercall_kafl_crash_dump_size(run, cpu);  
    ret = 0;  
    break;
```

```
~/kAFL-1 # python3 uniq_crashes.py /root/crashes addresses.pickle
```

```
/root/crashes/9dc81003df5331465de698a310e09f3162edc3de.log  
vmswitch.sys!VmsIfrInfoParamsNdisOidRequest + 0x57  
vmswitch.sys!VmsIfrInfoParams_OID_SWITCH_NIC_REQUEST + 0x145  
vmswitch.sys!VmsIfrInfoParamsNdisOidRequestBuffer + 0x99  
vmswitch.sys!RndisDevHostHandleSetExMessage + 0x210  
vmswitch.sys!RndisDevHostHandleControlMessage + 0x1c3  
vmswitch.sys!RndisDevHostControlMessageWorkerRoutine + 0x1b  
vmswitch.sys!RndisDevHostDispatchControlMessage + 0x1a9  
vmswitch.sys!VmsVmNicPvtKmc1ProcessingCompleteInternal + 0x12e  
vmswitch.sys!VmsVmNicPvtVersion1HandleRndisSendMessage + 0x228  
fuzzer.sys + 00000000000030A1
```

```
=====
```

```
/root/crashes/127ac58f2ddda05991ae8ad3e366a77b26242dd4.log  
vmswitch.sys!VmsIfrInfoParams_OID_SWITCH_NIC_REQUEST + 0xfb  
vmswitch.sys!VmsIfrInfoParamsNdisOidRequestBuffer + 0x99  
vmswitch.sys!RndisDevHostHandleSetExMessage + 0x210  
vmswitch.sys!RndisDevHostHandleControlMessage + 0x1c3  
vmswitch.sys!RndisDevHostControlMessageWorkerRoutine + 0x1b  
vmswitch.sys!RndisDevHostDispatchControlMessage + 0x1a9  
vmswitch.sys!VmsVmNicPvtKmc1ProcessingCompleteInternal + 0x12e  
vmswitch.sys!VmsVmNicPvtVersion1HandleRndisSendMessage + 0x228  
fuzzer.sys + 00000000000030A1  
fuzzer.sys + 0000000000001E19
```

```
=====
```

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**



## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

# Structure Awareness

- **Protocol Buffers**
- We integrated Libprotobuf-Mutator (LPM) into hAFL1 by adding a state

```
message RNDISMessage {
  fixed32 ndis_msg_type = 10;
  fixed32 msg_len = 11;
  oneof rndis_msg_container {
    REMOTE_NDIS_PACKET_MSG packet_msg = 1;
    REMOTE_NDIS_INITIALIZE_MSG init_msg = 2;
    REMOTE_NDIS_HALT_MSG halt_msg = 3;
    REMOTE_NDIS_QUERY_MSG query_msg = 4;
    REMOTE_NDIS_SET_MSG set_msg = 5;
    REMOTE_NDIS_RESET_MSG reset_msg = 6;
    REMOTE_NDIS_KEEPALIVE_MSG keepalive_msg = 8;
  }
}
```

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

## **Fuzzing Infrastructure**

**Harness**

**Coverage  
Guidance**

**Crash  
Monitoring**

**Structure  
Awareness**

[kAFL v0.2 ]

Runtime:	14m59s	#Execs:	2.7M	Exec/s:	3055	Slaves:	6/8
CPU Used:	43%	User:	10.2%	Guest:	28.7%	Stability:	100%
Mem Used:	21.7%	Avail:	97.0G	Total:	125.7G%	Reset/s:	0

Path Info:	Bitmap Stats:		Findings:	
Total:	13	Crash:	0 (N/A)	None Yet
Seeds:	1	Edges:	0 (N/A)	None Yet
Favs:	13	p(col):	0 (N/A)	None Yet
Norm:	0	Pending:	0.0%	Regular: 13 (N/A) 5m22s

Yld: Init:	5	Grim:	0	Redq:	0	Det:	6	Hvc:	1
Fav: Init:	0	Rq/Gr:	0	Det:	0	Hvc:	0	Fin:	13
Nrm: Init:	0	Rq/Gr:	0	Det:	0	Hvc:	0	Fin:	0

> Slave 0:	afl_splice	node:	4	fav/lvl:	2/1	exec/s:	515
Slave 1:	afl_splice	node:	9	fav/lvl:	4/2	exec/s:	499
Slave 2:	afl_havoc	node:	6	fav/lvl:	2/1	exec/s:	508
Slave 3:	afl_splice	node:	7	fav/lvl:	4/1	exec/s:	503
Slave 4:	afl_havoc	node:	3	fav/lvl:	1/1	exec/s:	510
Slave 5:	afl_havoc	node:	12	fav/lvl:	2/2	exec/s:	518

[Payload Info]

Parent:	1	Size:	20B	Bytes:	20	Bits:	0	Exit:	r
---------	---	-------	-----	--------	----	-------	---	-------	---

0x00000000:	50 00 00 00 36 00 00 00 58 00 00 00 1b 20 10 00	....6...X.....
0x00000010:	1a 00 00 00	....

CVE-2021-28476

---



`RndisDevHostControlMessageWorkerRoutine`

**RndisDevHostControlMessageWorkerRoutine**



**RndisDevHostHandleSetMessage**

**RndisDevHostControlMessageWorkerRoutine**



**RndisDevHostHandleSetMessage**



**VmsIfrInfoParamsNdisOidRequestBuffer**

**RndisDevHostControlMessageWorkerRoutine**



**RndisDevHostHandleSetMessage**



**VmsIfrInfoParamsNdisOidRequestBuffer**




**VmsIfrInfoParams\_OID\_SWITCH\_NIC\_REQUEST**

# OID\_SWITCH\_NIC\_REQUEST

- Encapsulates & forwards OID requests to an external adapter
- **Supposed** to be sent only by the extensible switch (root partition)

# Syntax

C++

 Copy

```
typedef struct _NDIS_SWITCH_NIC_OID_REQUEST {
    NDIS_OBJECT_HEADER    Header;
    ULONG                 Flags;
    NDIS_SWITCH_PORT_ID   SourcePortId;
    NDIS_SWITCH_NIC_INDEX SourceNicIndex;
    NDIS_SWITCH_PORT_ID   DestinationPortId;
    NDIS_SWITCH_NIC_INDEX DestinationNicIndex;
    PNDIS_OID_REQUEST     OidRequest;
} NDIS_SWITCH_NIC_OID_REQUEST, *PNDIS_SWITCH_NIC_OID_REQUEST;
```

```
;
; Fetch OidRequest from OID_SWITCH_NIC_REQUEST
;

loc_FFFF80D7C000A3B:
mov     r10, [rbp+18h]
test    r10, r10
jz      loc_FFFF80D7C039CB0
```

```
cmp     cs:WPP_RECORDER_INITIALIZED, rbx
jz      short loc_FFFF80D7C000A92
```

```
; START OF FUNCTION CHUNK FOR VmsIfrInfoParams_OID_SWITCH_NIC_REQUEST

loc_FFFF80D7C039CB0:
cmp     cs:WPP_RECORDER_INITIALIZED, rbx
jz      short loc_FFFF80D7C039CE9
```

```
mov     ecx, 4
call    VmsWppTraceLevelText
mov     edx, [rsi]
mov     r9d, 13Bh
;
; THIS INSTRUCTION CRASHES
;
mov     ecx, [r10+20h]
mov     [rsp+98h+var_48], ecx
mov     [rsp+98h+var_50], r12
mov     [rsp+98h+var_58], rax
shr     edx, 2
and     edx, 7Fh
mov     dword ptr [rsp+98h+var_60], edx
mov     dword ptr [rsp+98h+var_68], r14d
mov     [rsp+98h+var_70], r15
call    WPP_RECORDER_SF_sddssd
mov     r10, [rbp+18h]
```

```
mov     rcx, cs:VmsIfrLog
lea     rax, aOid_switch_nic ; "!\"OID_SWITCH_NIC_REQUEST with NULL Oid\"...
mov     [rsp+98h+var_70], rax
mov     r9d, 13Ch
lea     rax, WPP_dd3b18751bd6349f642b36ecd20a66b8_Traceguids
mov     r8d, 13h
mov     [rsp+98h+string], rax
call    WPP_RECORDER_SF_s
```

# Consequences

- Trivially, denial of service by crashing the Hyper-V host  
→ DoS of Azure
- The bug is an arbitrary pointer dereference which can potentially lead to a **guest-to-host RCE** (CVSS 9.9)



Demo

File Home View Breakpoints Time Travel Model Scripting Command Source

Step Out Step Out Back Restart  
 Step Into Step Into Back Stop Debugging  
 Step Over Step Over Back Go Back Detach

Flow Control Reverse Flow Control End Preference

```
Command
nt!DbgBreakPointWithStatus:
fffff807`411fd920 cc          int     3
kd> g
Break instruction exception - code 80000003 (first chance)
*****
*
* You are seeing this message because you pressed either
* CTRL+C (if you run console kernel debugger) or,
* CTRL+BREAK (if you run GUI kernel debugger),
* on your debugger machine's keyboard.
*
* THIS IS NOT A BUG OR A SYSTEM CRASH
*
* If you did not intend to break into the debugger, press the "g"
* press the "Enter" key now. This message might immediately reapp
* does, press "g" and "Enter" again.
*
*****
nt!DbgBreakPointWithStatus:
fffff807`411fd920 cc          int     3
kd> g
nt!DebugService2+0x5:
fffff807`411fd975 cc          int     3
kd> g
```

\*BUSY\* Debuggee is running...

Disassembly

Address: @scopeip  Follow current instruction

```
00000000 00000000 ?? ???
00000000 00000001 ?? ???
00000000 00000002 ?? ???
00000000 00000003 ?? ???
00000000 00000004 ?? ???
00000000 00000005 ?? ???
00000000 00000006 ?? ???
00000000 00000007 ?? ???
00000000 00000008 ?? ???
00000000 00000009 ?? ???
00000000 0000000a ?? ???
00000000 0000000b ?? ???
00000000 0000000c ?? ???
00000000 0000000d ?? ???
00000000 0000000e ?? ???
00000000 0000000f ?? ???
```

L1 on GC-PC-161 - Virtual Machine Connection

File Action Media View Help

L2 on DESKTOP-0015N7R - Virtual Machine Connection

File Action Media View Help

Administrator: C:\Windows\system32\cmd.exe

Microsoft Windows [Version 10.0.17763.379]  
 (c) 2018 Microsoft Corporation. All rights reserved.

```
C:\Windows\system32>c:\Users\ophir\Desktop
'c:\Users\ophir\Desktop' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\system32>cd c:\Users\ophir\Desktop
c:\Users\ophir\Desktop>devcon_x64.exe remove Fuzzer
No devices were removed.

c:\Users\ophir\Desktop>devcon_x64.exe remove Root\Fuzzer
No devices were removed.

c:\Users\ophir\Desktop>devcon_x64.exe install Fuzzer.inf Root\Fuzzer
Device node created. Install is complete when drivers are installed...
VULN:Updating drivers for Root\Fuzzer from c:\Users\ophir\Desktop\Fuzzer.inf.
Drivers installed successfully.

c:\Users\ophir\Desktop>PacketSender_2.exe
```

Device Setup

Let's finish setting up  
 If this time still works for you, select OK to get set up.

OK Dismiss

Status: Running

Type here to search

7:16 AM 5/9/2021

<https://github.com/SB-GC-Labs/hAFL1>



# Thank you.

---

@peleghd

@ophirharpaz

 SafeBreach

 Guardicore

# Q&A

---

@peleghd

 SafeBreach

@ophirharpaz

 Guardicore