Breaking Theoretical Limits: The Gap Between Virtual NICs and Physical Network Cards

Quan Luo, Qian Chen | December 2023

About Us

Quan Luo
@TrueUnitySect

OS Virtualization
Network Protocol

Qian Chen
@cq674350529

IoT
Network Protocol

A Ben

OS Browser
Network Protocol

Ruiqi Chen
@kevinoclam2

Web Windows

Hang An
@HangAn54637220

Linux Kernel

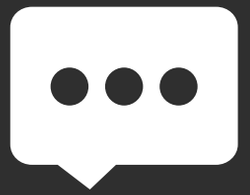
奇安信代码安全实验室
— Qi'anxin Codesafe Team —

Focus on software source code security analysis and binary vulnerability research

Agenda

Introduction

Hyper-V Network Module Research



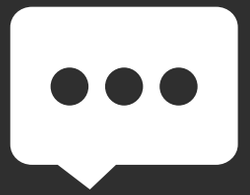
Vulnerability Analysis

Summary

Agenda

Introduction

Hyper-V Network Module Research

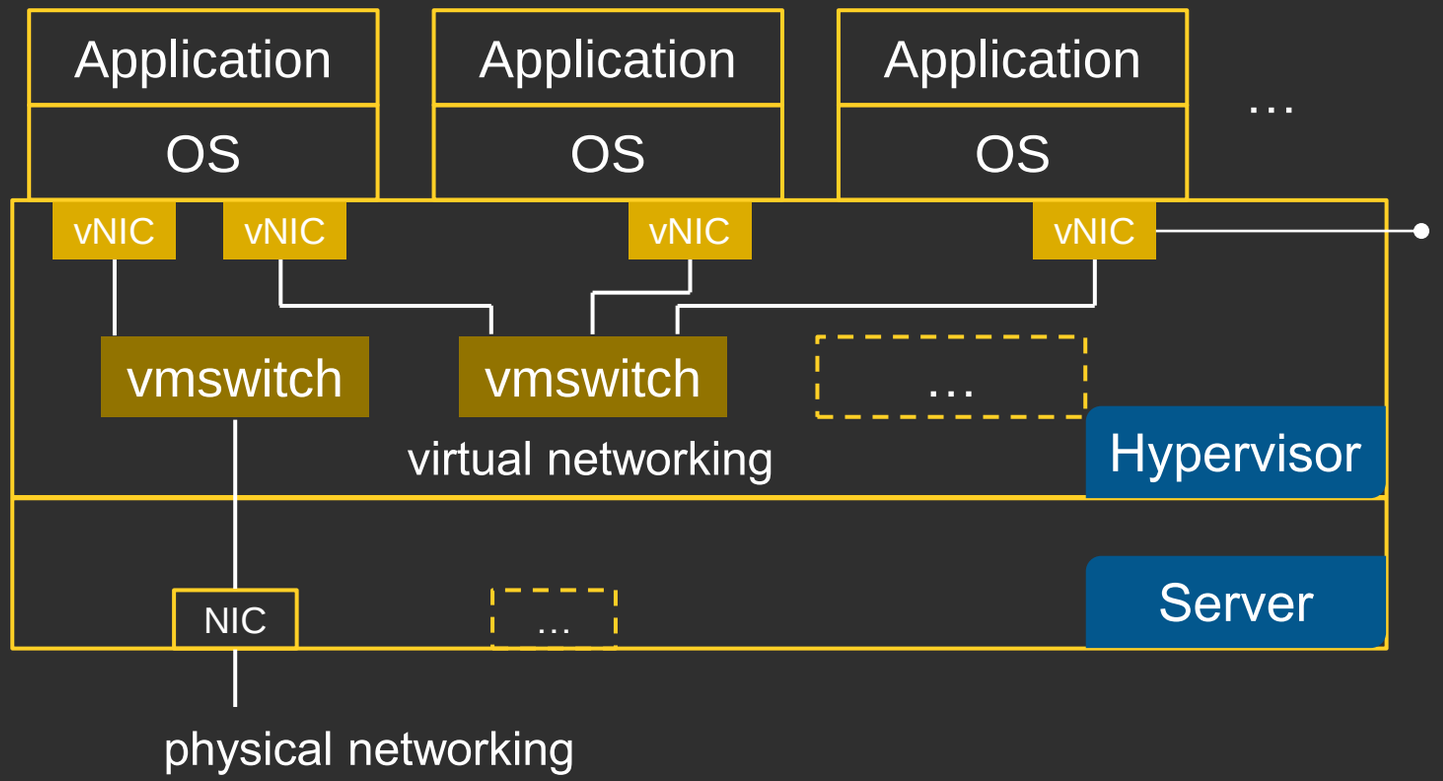


Vulnerability Analysis

Summary

Virtualization Technology

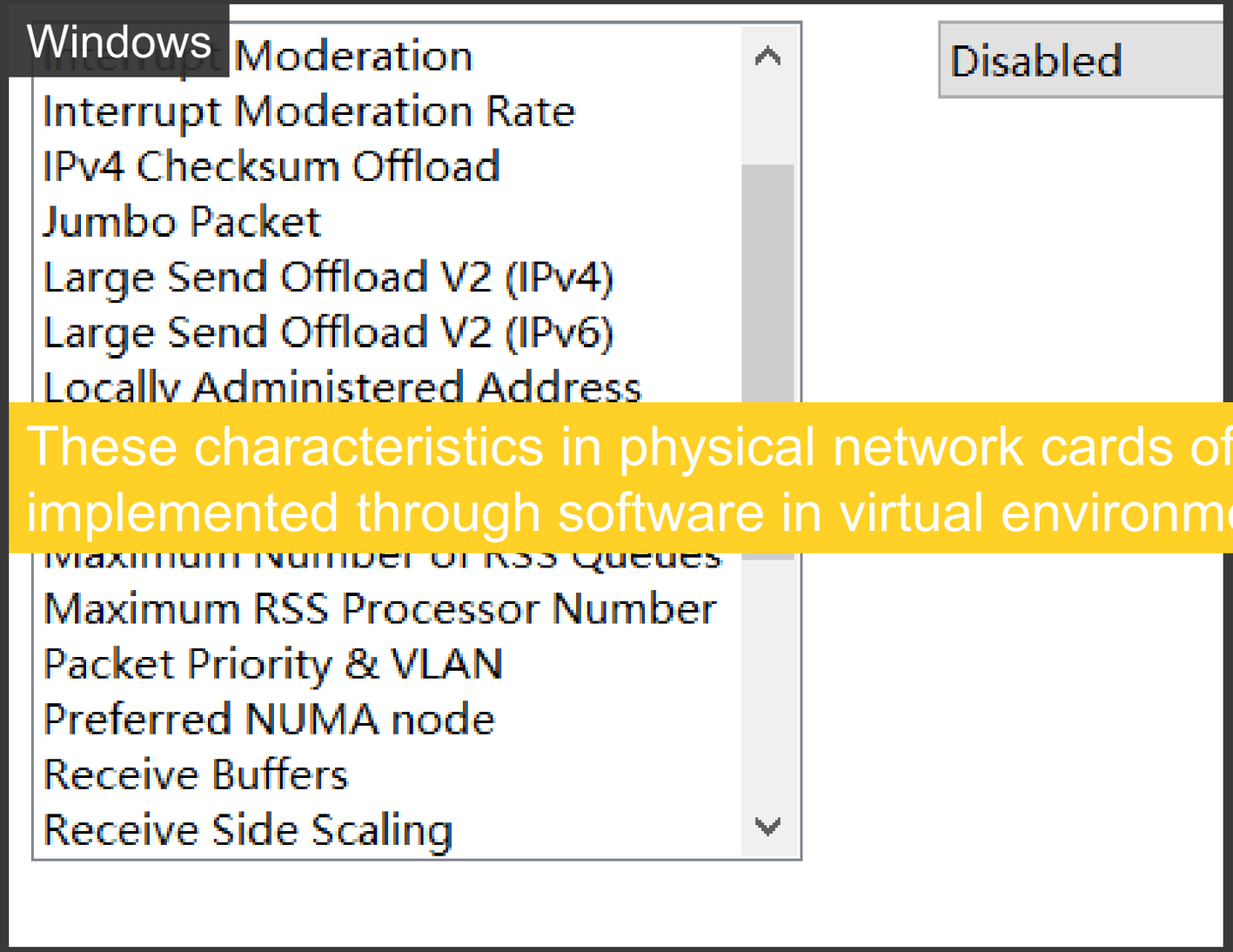
- Provide the foundational technology for creating and managing virtual resources like virtual servers and virtual networks



- provide functionalities like Open vSwitch (SDN) and communication between adjacent virtual machines
- serve as a fundamental and low-level infrastructure, which is an appealing target for virtual machine escape

Network Interface Card (NIC) Characteristics

Windows

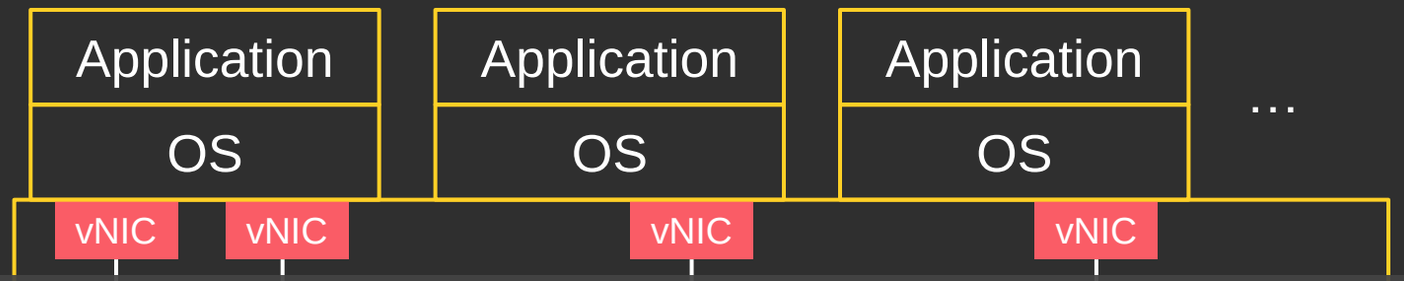


Linux

```
abc]# ethtool -S ens160
tics:
  Tx Queue#: 0
    TSO pkts tx: 0
    TSO bytes tx: 0
    ucast pkts tx: 87
    ucast bytes tx: 10546
    mcast pkts tx: 11
    mcast bytes tx: 818
    bcast pkts tx: 5
    drv dropped tx total: 0
       too many frags: 0
       giant hdr: 0
       hdr err: 0
       tso: 0
    ring full: 0
    pkts linearized: 0
    hdr cloned: 0
    giant hdr: 0
```

These characteristics in physical network cards often need to be simulated and implemented through software in virtual environments.

Virtual NIC

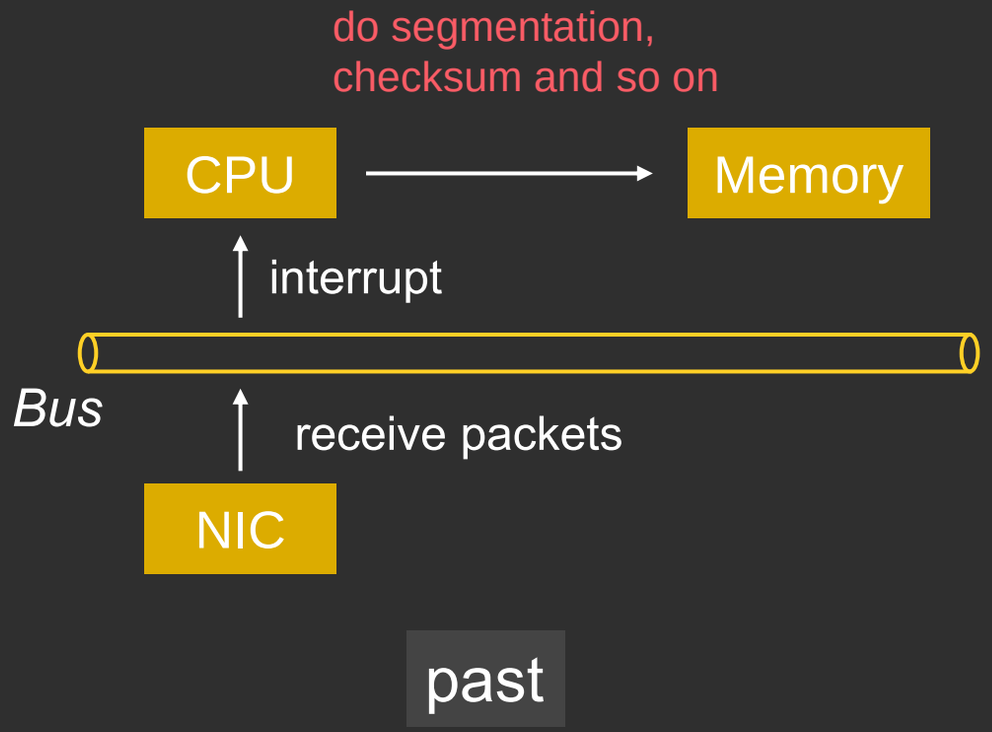


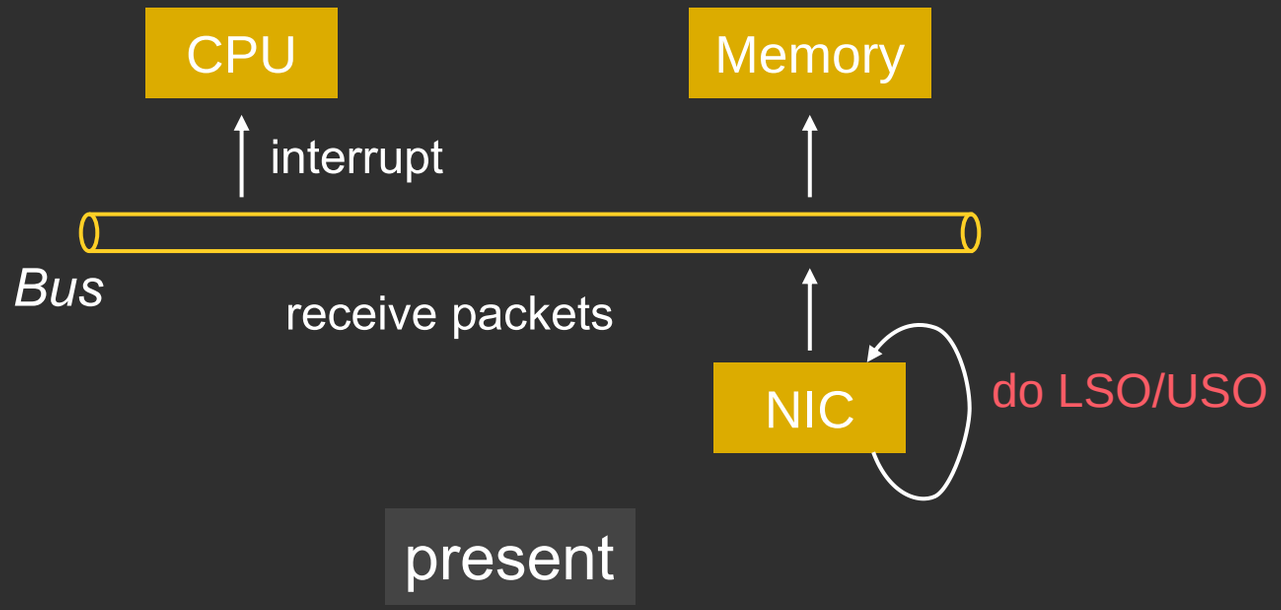
- UDP Segmentation Offload (USO): offload the task of segmenting large UDP packets into small fragments from CPU to NIC
- Large Send Offload (LSO): offload the task of segmenting large TCP packets into small fragments from CPU to NIC
- …

implementation in software

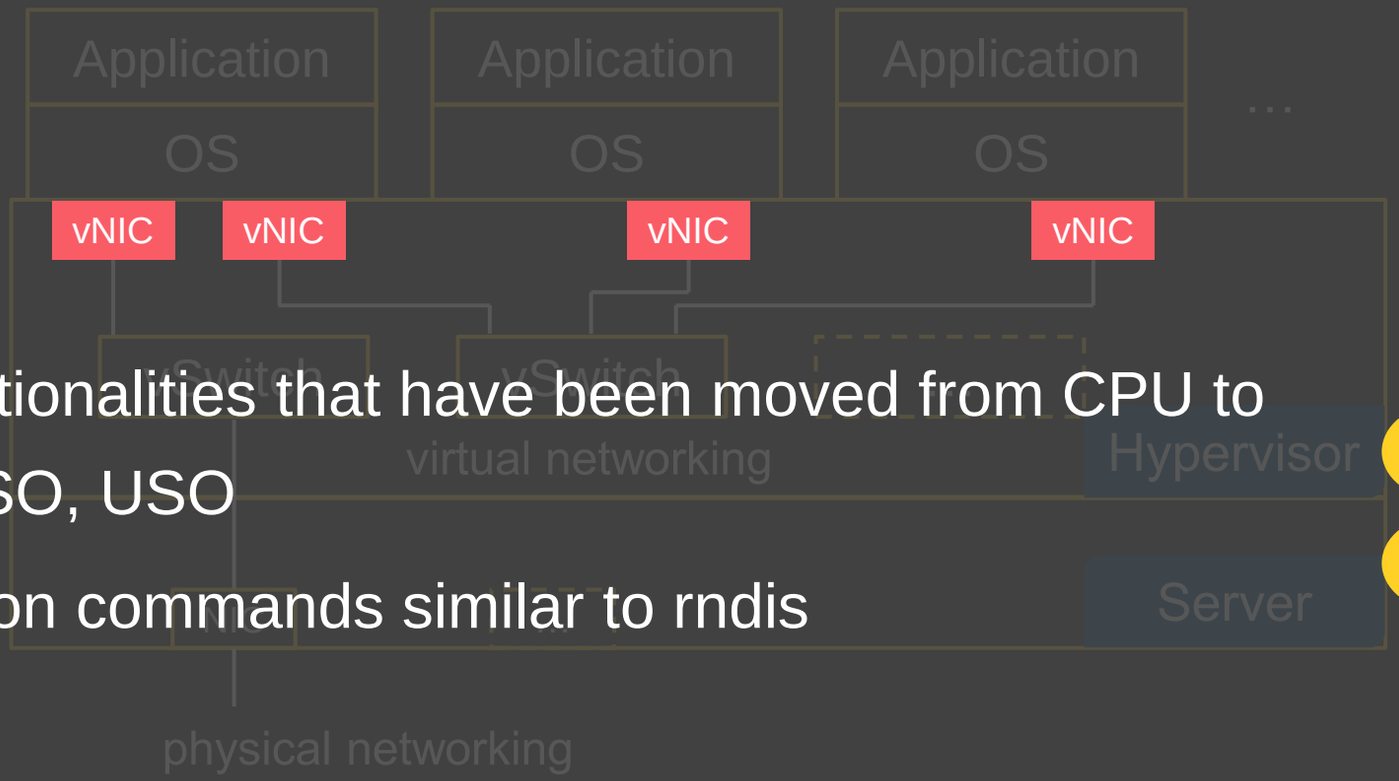
Virtual NIC

- Category: E1000, E1000e, VMXNET, VMXNET2, VMXNET3, …
- Primary feature: provide functionalities that have been migrated from CPU to NIC





Past Research Focus



- Those functionalities that have been moved from CPU to NIC, like LSO, USO
- Configuration commands similar to rndis

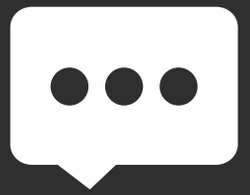
fuzzing

code review

Agenda

Introduction

Hyper-V Network Module Research



Vulnerability Analysis

Summary

Choose code review
when fuzzing yields no promising results

reverse engineering the vmswitch module

No.	Time	Source	Destination	Protocol	Length	Info
2	69.159055	fe80::db90:748e:fc5f:e62f	fe80::acc6:5128:792d:5005	ICMPv6	69741	Unknown (86)
3	69.159055	fe80::db90:748e:fc5f:e62f	fe80::acc6:5128:792d:5005	ICMPv6	69741	Unknown (86)
4	74.177271	fe80::db90:748e:fc5f:e62f	fe80::acc6:5128:792d:5005	ICMPv6	69741	Unknown (86)
5	74.177271	fe80::db90:748e:fc5f:e62f	fe80::acc6:5128:792d:5005	ICMPv6	69741	Unknown (86)

```
> Frame 2: 69741 bytes on wire (557928 bits), 69741 bytes captured (557928 bits) on interfac
> Ethernet II, Src: Microsof_be:bc:00 (00:15:5d:be:bc:00), Dst: VMware_3e:02:d1 (00:0c:29:3e
v Internet Protocol Version 6, Src: fe80::db90:748e:fc5f:e62f, Dst: fe80::acc6:5128:792d:500
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 0 (Jumbogram)
    Next Header: IPv6 Hop-by-Hop Option (0)
    Hop Limit: 255
    Source Address: fe80::db90:748e:fc5f:e62f
    Destination Address: fe80::acc6:5128:792d:5005
  > IPv6 Hop-by-Hop Option
  > Routing Header for IPv6 (Source Route)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
  > Routing Header for IPv6 (Unknown type 86)
```

```
00000000  00 0c 29 3e 02 d1 00 15  5d be bc 00 86 dd 60 00
00000010  00 00 00 00 00 ff fe 80  00 00 00 00 00 00 db 90
00000020  74 8e fc 5f e6 2f fe 80  00 00 00 00 00 00 ac c6
00000030  51 28 79 2d 50 05 2b 00  c2 04 00 01 10 10 2b ff
00000040  00 00 06 d5 00 15 0c ba  73 60 00 00 00 00 56 ff
00000050  02 00 00 00 00 00 10 11  12 13 14 15 16 17 18 19
00000060  1a 1b 1c 1d 1e 1f 20 21  22 23 24 25 26 27 28 29
00000070  2a 2b 2c 2d 2e 2f 30 31  32 33 34 35 36 37 56 56
00000080  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000090  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000000a0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000000b0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000000c0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000000d0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000000e0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000000f0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000100  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000110  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000120  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000130  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000140  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000150  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000160  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000170  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000180  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
00000190  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000001a0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000001b0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000001c0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
000001d0  56 56 56 56 56 56 56 56  56 56 56 56 56 56 56 56
```

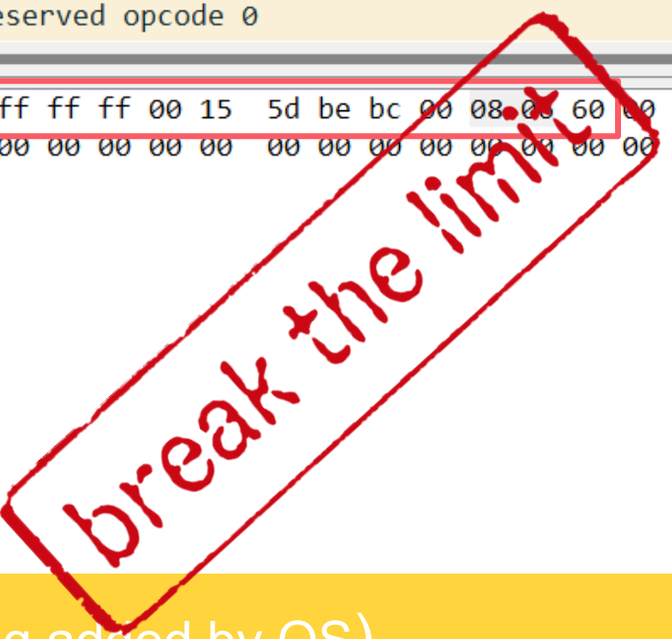
Break the limit

A single ICMPv6 packet whose length is bigger than 65535

No.	Time	Source	Destination	Protocol	Length	Info
23	0.168495	Microsof_be:bc:00	Broadcast	ARP	34	Reserved opcode 0
24	0.168520	Microsof_be:bc:00	Broadcast	ARP	34	Reserved opcode 0
25	0.168545	Microsof_be:bc:00	Broadcast	ARP	34	Reserved opcode 0
26	0.252788	Microsof_be:bc:00	Broadcast	ARP	34	Reserved opcode 0
27	0.252925	Microsof_be:bc:00	Broadcast	ARP	34	Reserved opcode 0

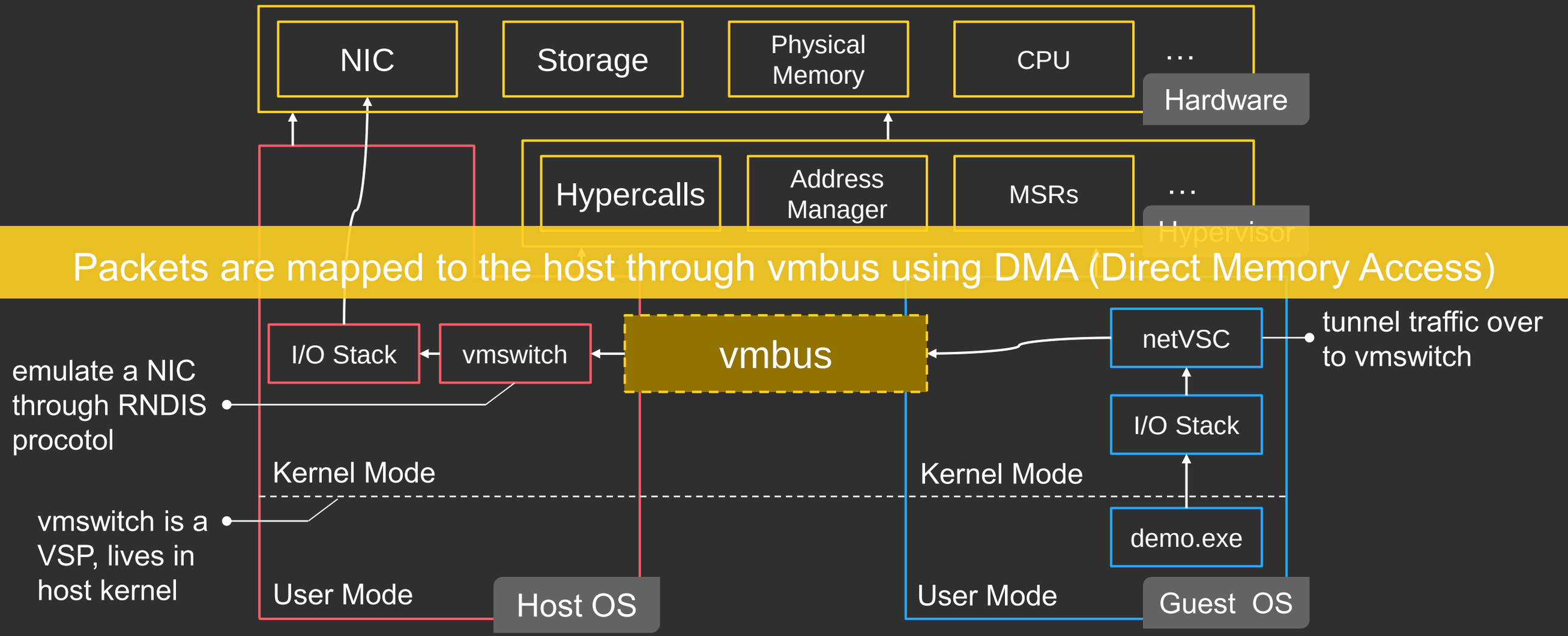
```
> Frame 23: 34 bytes on wire (272 bits), 34 bytes captured (272 bits) on interface \Device\NPF
v Ethernet II, Src: Microsof_be:bc:00 (00:15:5d:be:bc:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: Microsof_be:bc:00 (00:15:5d:be:bc:00)
    Type: ARP (0x0806)
    Trailer: 000000000000000000000000
v Address Resolution Protocol (reserved)
    Hardware type: Unknown (24576)
    Protocol type: Unknown (0x0000)
    Hardware size: 0
    Protocol size: 0
    Opcode: reserved (0)
```

```
0000  ff ff ff ff ff ff 00 15  5d be bc 00 08 00 60 00
0010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0020  00 00
```

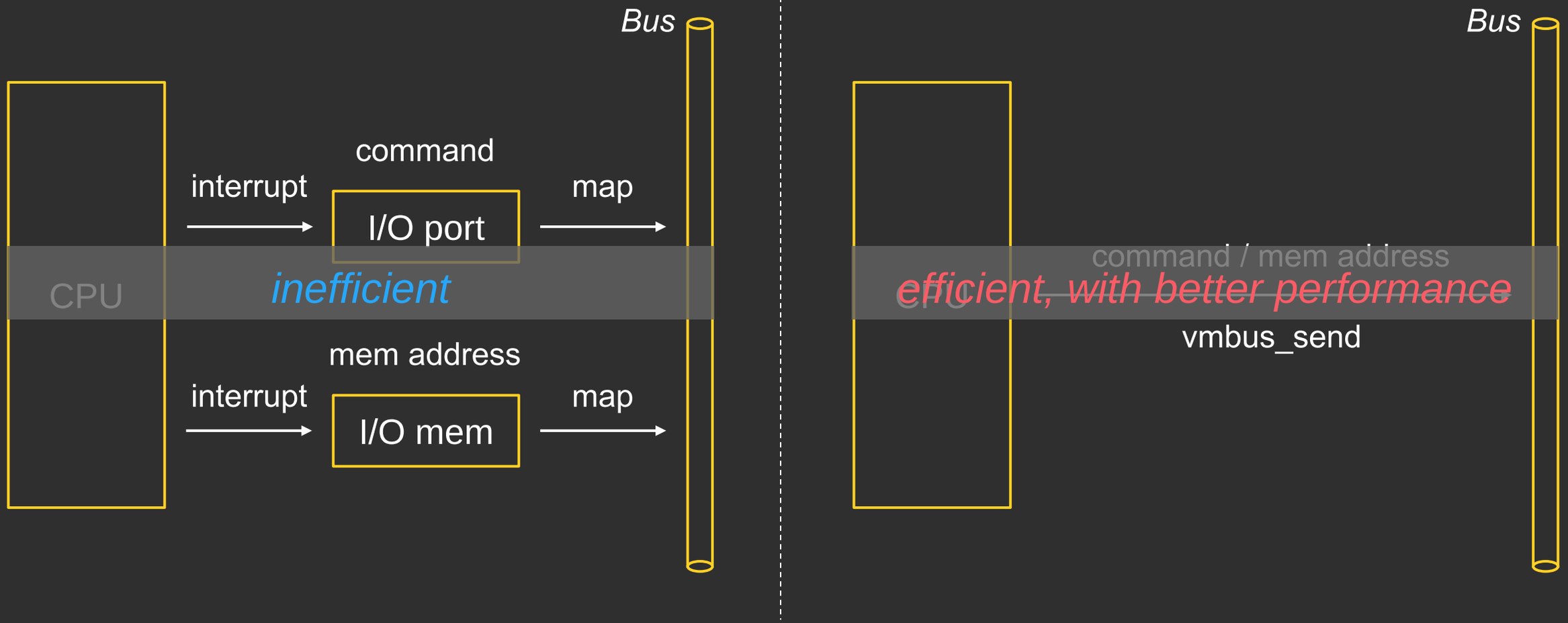


A single ARP packet whose length is only 15 (extra padding added by OS)

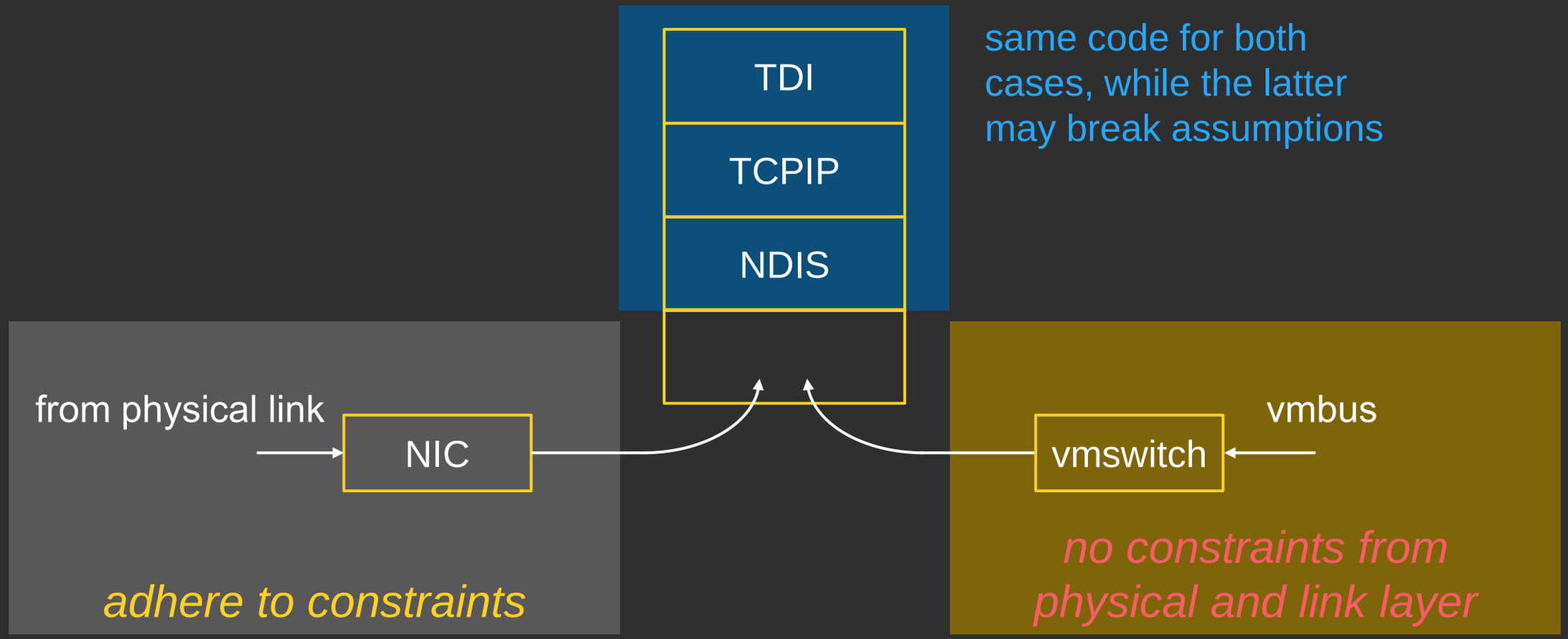
Packet Transmission in Hyper-V



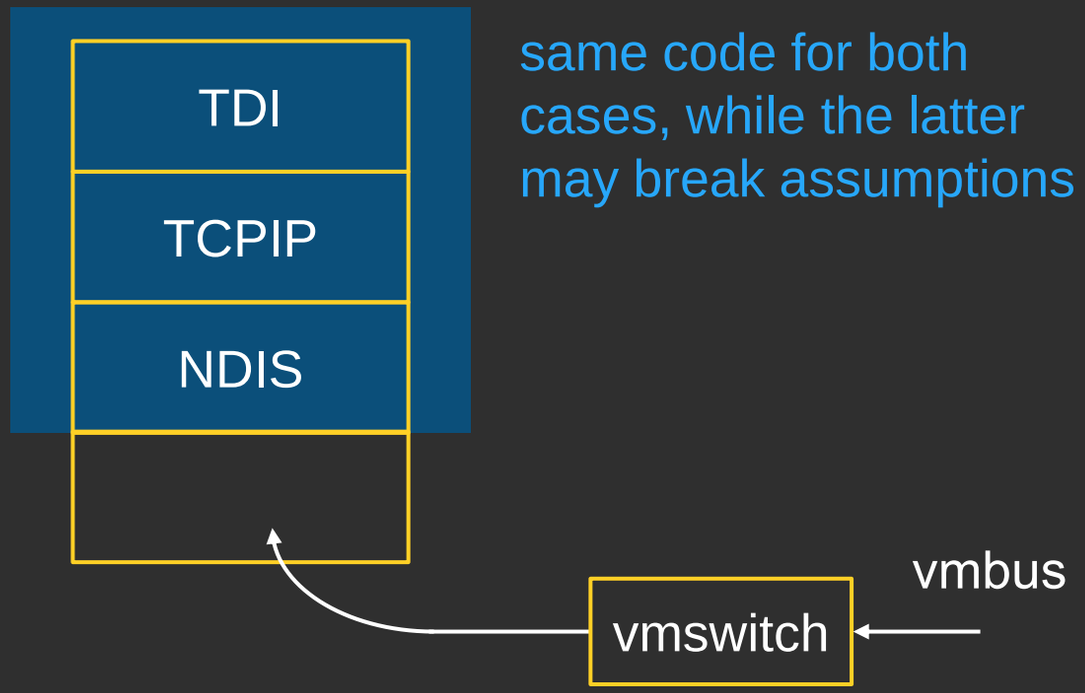
I/O Port vs vmbus



How Packets Reaching Network I/O Stack



Call Stack for Packets in vmswitch



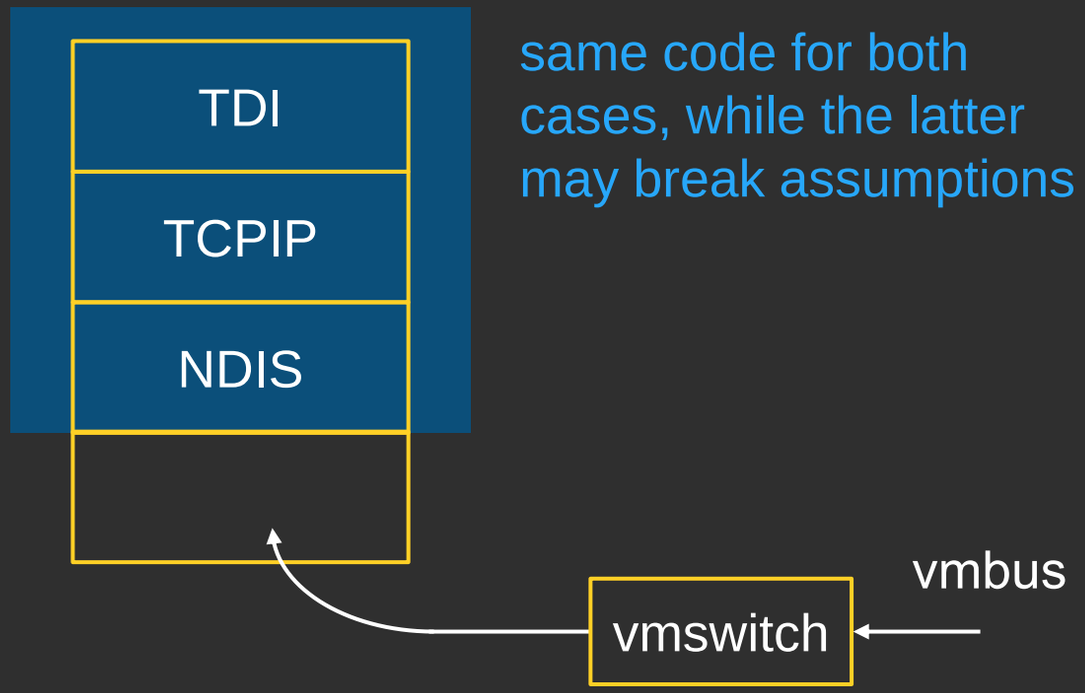
same code for both cases, while the latter may break assumptions

```
vmswitch!VmsVmNicPvtRndisDeviceSendPackets
vmswitch!RndisDevHostHandlePacketMessages+0x212
vmswitch!VmsVmNicPvtKmclProcessingComplete+0x1e3
vmbkmclr!InpFillAndProcessQueue+0x2d0
vmbkmclr!KmclpVmbusIsr+0x126
vmbusr!ParentRingInterruptDpc+0x62
nt!KiExecuteAllDpcs+0x335
nt!KiRetireDpcList+0x910
nt!KyRetireDpcList+0x5
nt!KiDispatchInterruptContinue
```

call stack

1. transform from a message to packet
2. enter the protocol processing function (protocol handler) registered in vmswitch for NDIS

Call Stack for Packets in vmswitch



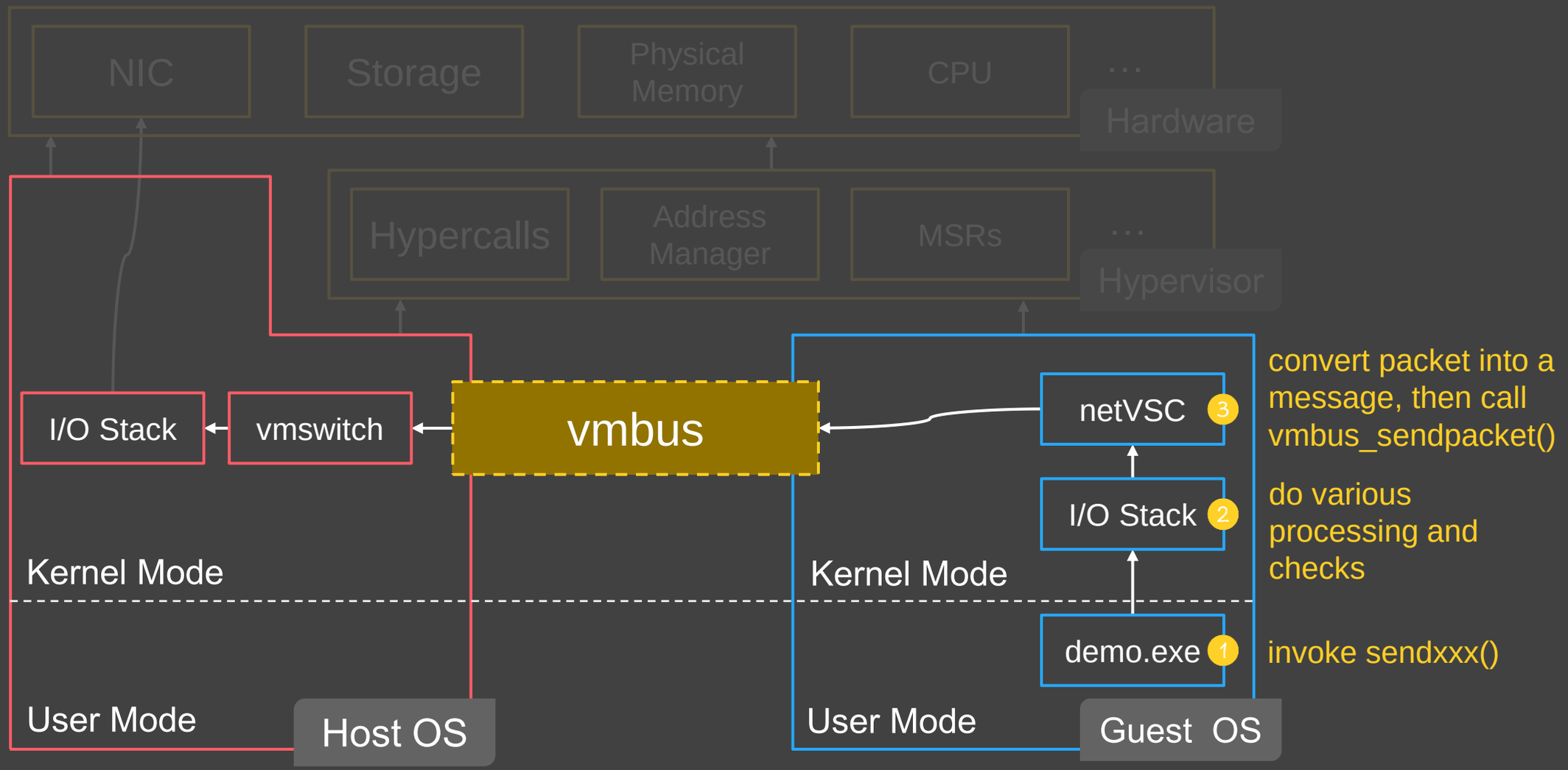
same code for both cases, while the latter may break assumptions

```
vmswitch!RndisDevHostDeviceIndicatePackets
vmswitch!RndisDevDeviceIndicatePackets+0x4a
vmswitch!VmsVmNicPvtPacketForward+0x496
vmswitch!VmsRouterDeliverNetBufferLists+0x81a
vmswitch!VmsExtPtReceiveNetBufferLists+0x193
NDIS!ndisMIndicateNetBufferListsToOpen+0x11e
NDIS!ndisMTopReceiveNetBufferLists+0x267bc
NDIS!ndisCallReceiveHandler+0x47
NDIS!NdisMIndicateReceiveNetBufferLists+0x735
vmswitch!VmsExtMpIndicatePackets+0xa55
vmswitch!VmsExtMpSendNetBufferLists+0x
```

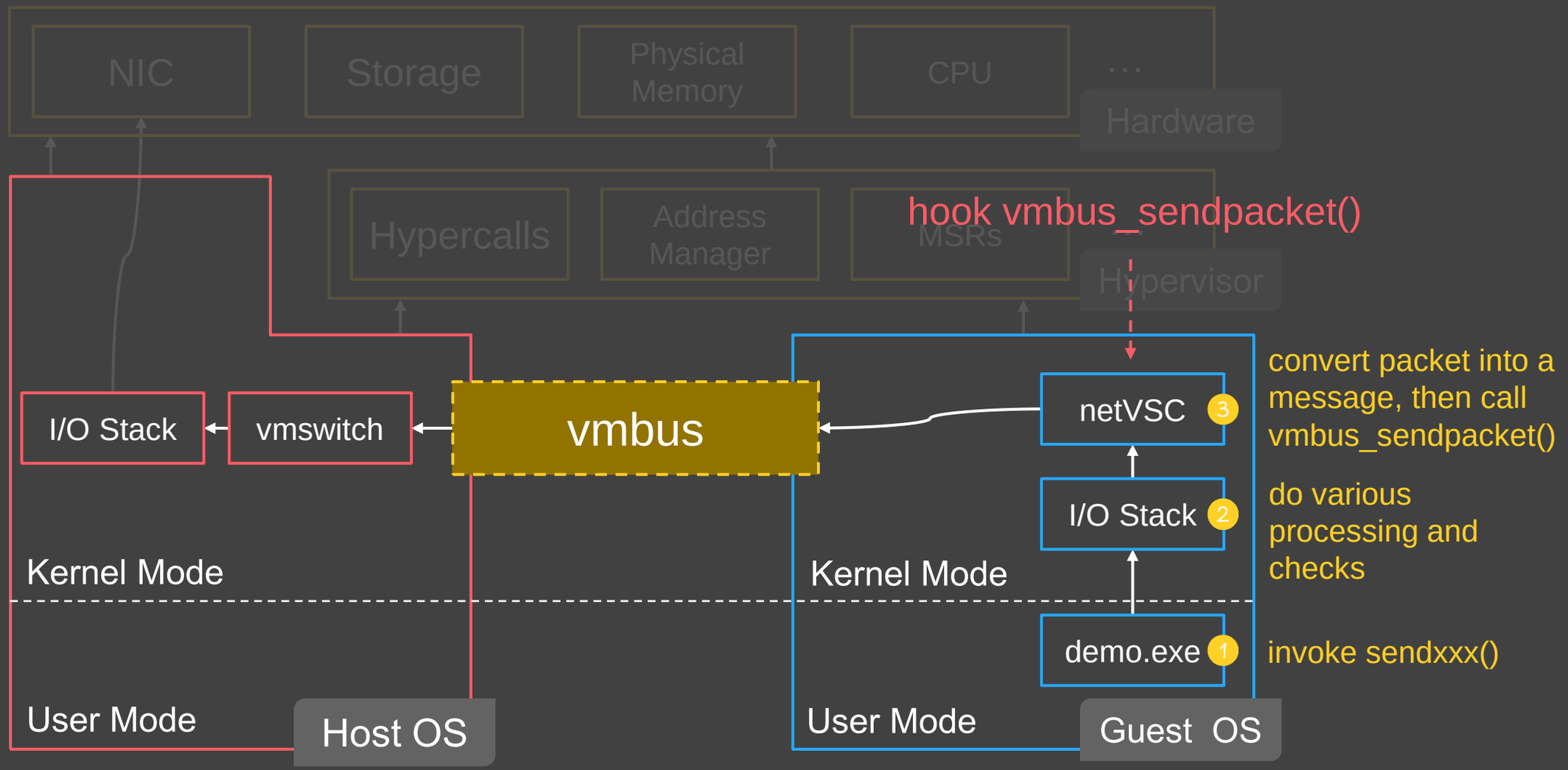
call stack

1. reach VmsVmNicPvtPacketForward() after a series of filtering, verification, addressing
2. invoke the corresponding handler on the protocol stack to send the packet

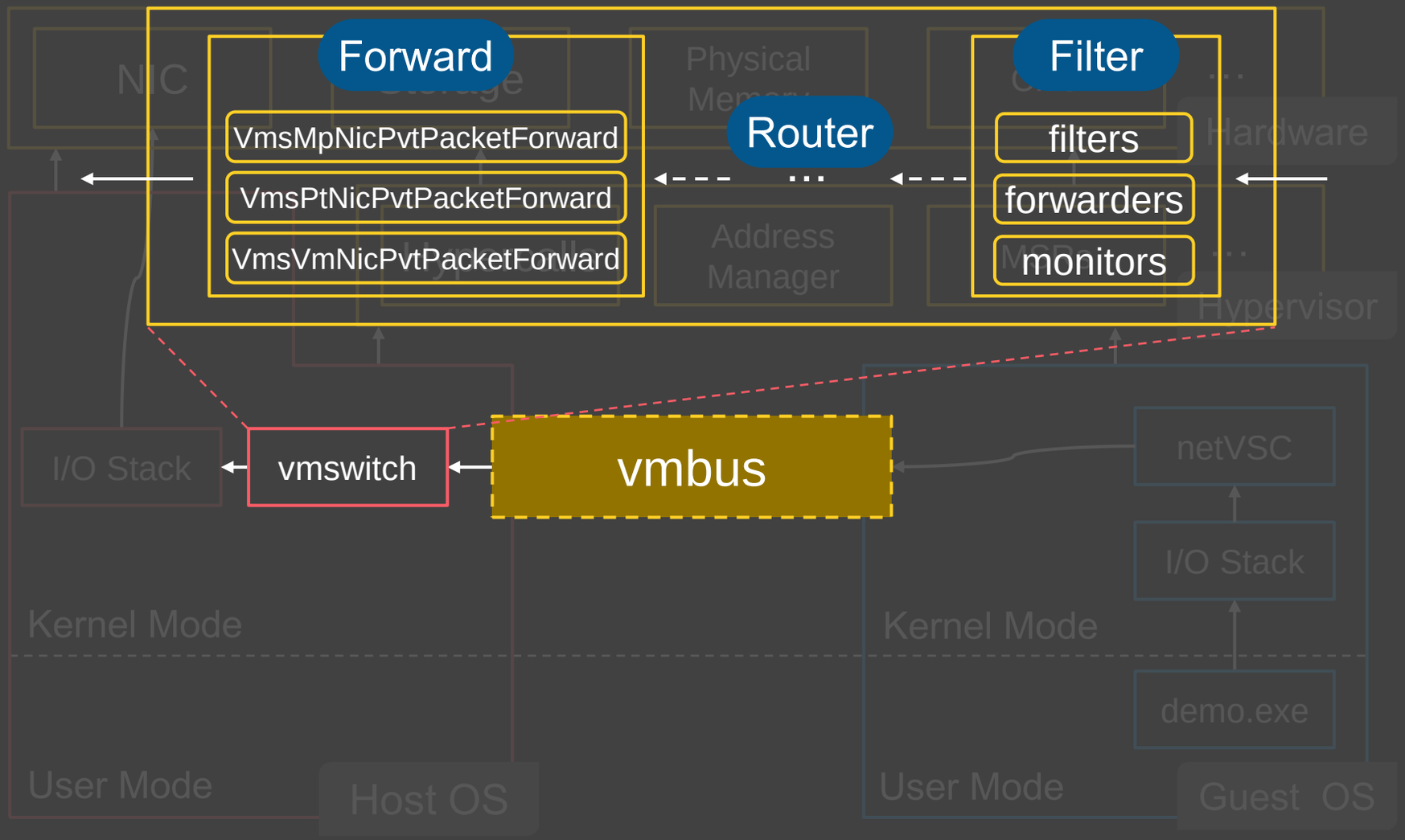
How to Send Normal Packets



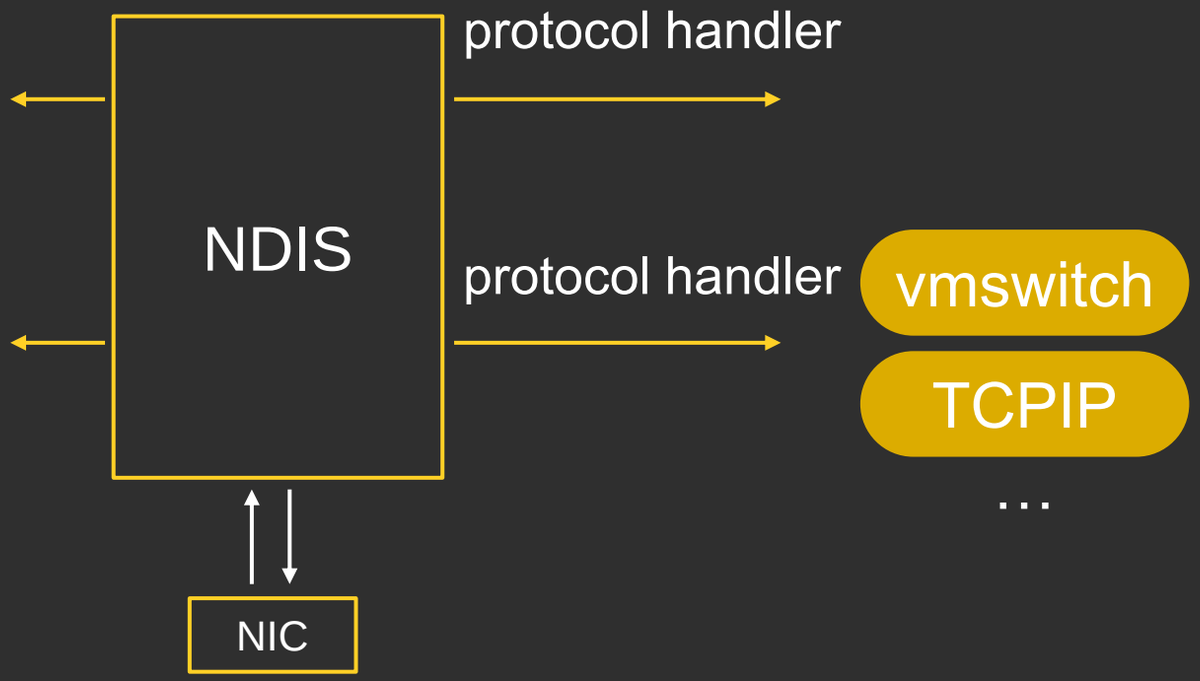
How to Send “Anormal” Packets



Packet Process Flow in vmswitch



NDIS Network Interface Architecture

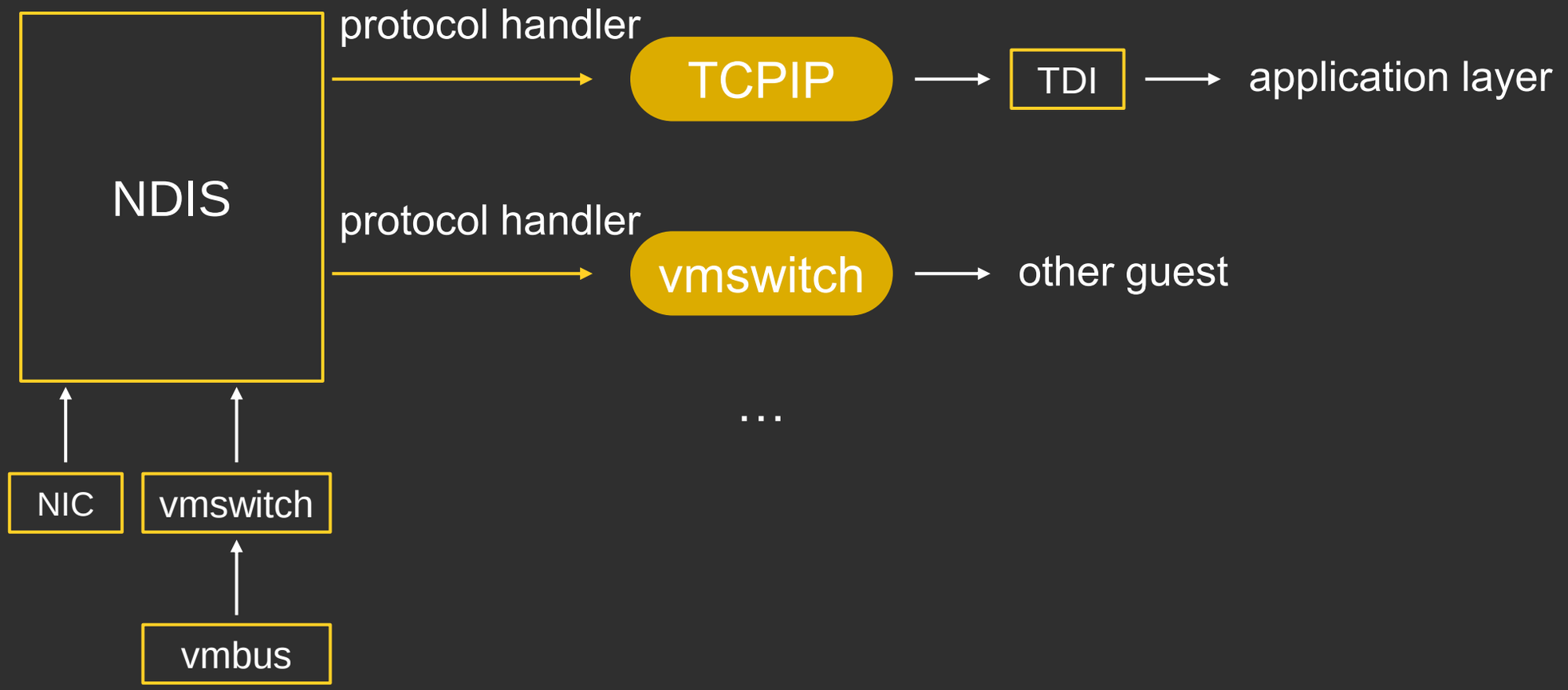


- vmswitch can be considered as a filtering driver stacked on top of NDIS
- Many of the function pointers in vmswitch are treated as dispatch function pointers for NDIS

vmswitch Stacking Behavior

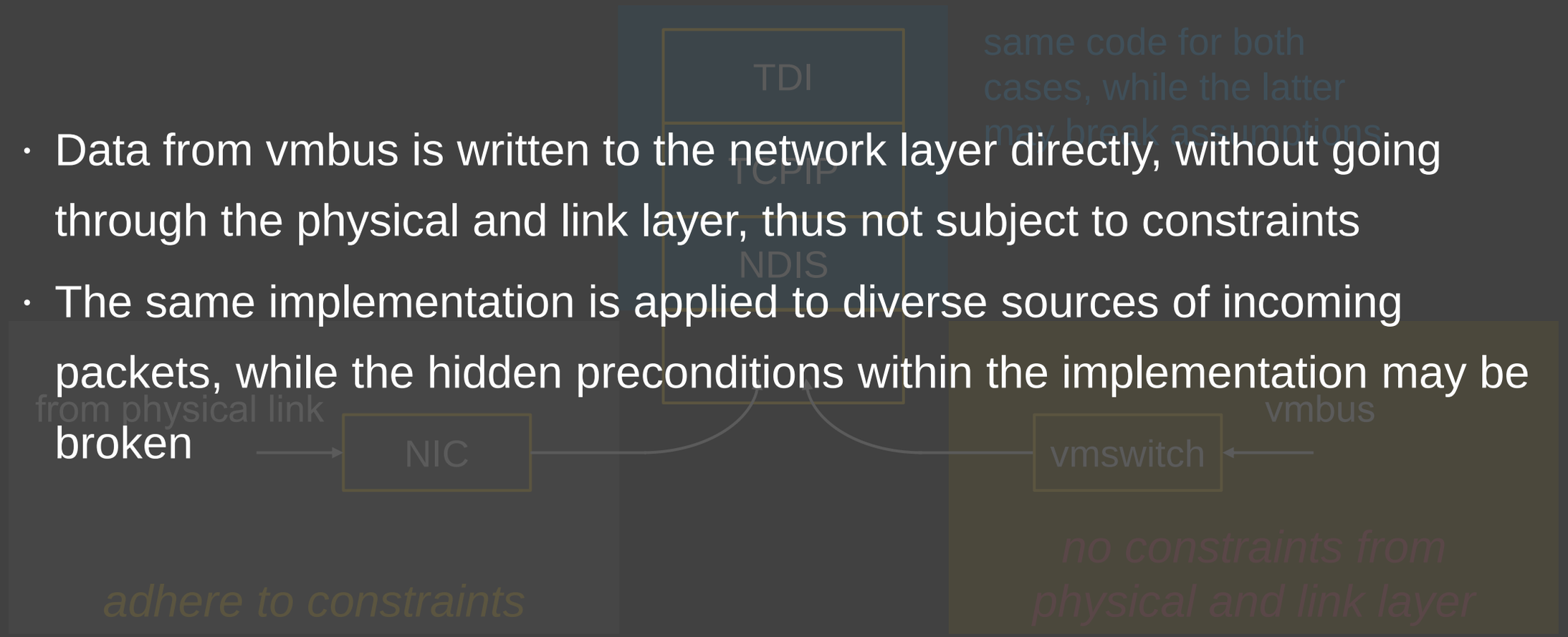
```
// ...
RtlInitUnicodeString(&DestinationString, L"VMSP");
ProtocolCharacteristics.Header = 8389269;
// ...
ProtocolCharacteristics.OpenAdapterCompleteHandlerEx = VmsPtNicOpenAdapterCompleteEx;
ProtocolCharacteristics.CloseAdapterCompleteHandlerEx = VmsPtNicCloseAdapterCompleteEx;
// ...
ProtocolCharacteristics.UninstallHandler = VmsPtNicUninstall;
v12 = NdisRegisterProtocolDriver(0i64, &ProtocolCharacteristics, &VmsProtocolHandle);
/* ... */
RtlInitUnicodeString(&v35, L"Hyper-V Virtual Switch Extension Filter");
RtlInitUnicodeString(&v36, L"{529B8983-9625-49A5-8284-CE944FD8E242}");
RtlInitUnicodeString(&v37, L"VMSVSF");
FilterDriverCharacteristics.SetOptionsHandler = VmsExtFilterSetFilterModuleOptions;
FilterDriverCharacteristics.SetFilterModuleOptionsHandler = VmsExtFilterSetFilterModuleOptions;
// ...
FilterDriverCharacteristics.SendNetBufferListsHandler = VmsExtFilterSendNetBufferLists;
// ...
v18 = NdisFRegisterFilterDriver(DriverObject, 0i64, &FilterDriverCharacteristics,
&VmsVswitchFilterHandle);
```

Processing Routine



Our Findings

- Data from vmbus is written to the network layer directly, without going through the physical and link layer, thus not subject to constraints
- The same implementation is applied to diverse sources of incoming packets, while the hidden preconditions within the implementation may be broken



Agenda

Introduction

Hyper-V Network Module Research

Vulnerability Analysis

Summary

CVE-2021-24074

Integer Overflow

Windows TCP/IP Remote Code Execution Vulnerability

CVE-2021-24074
Security Vulnerability

Released: Feb 9, 2021

Assigning CNA: Microsoft

CVE-2021-24074

CVSS:3.1 9.8 / 8.5

Exploitability

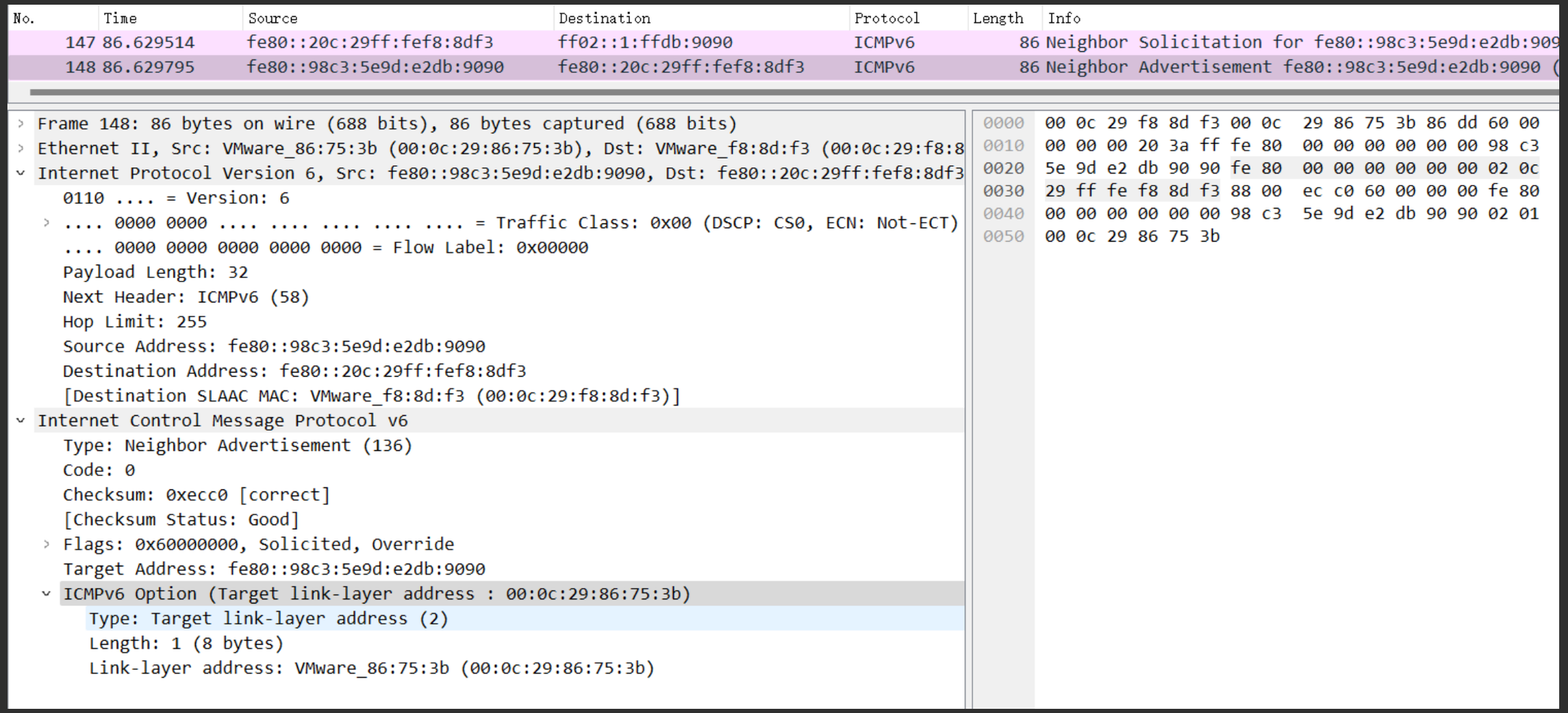
The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly disclosed	Exploited	Exploitability assessment
No	No	Exploitation More Likely

Caused by a single ICMPv6 packet whose length is bigger than 65535

CVE-2021-24074

Integer Overflow



No.	Time	Source	Destination	Protocol	Length	Info
147	86.629514	fe80::20c:29ff:fef8:8df3	ff02::1:ffdb:9090	ICMPv6	86	Neighbor Solicitation for fe80::98c3:5e9d:e2db:909
148	86.629795	fe80::98c3:5e9d:e2db:9090	fe80::20c:29ff:fef8:8df3	ICMPv6	86	Neighbor Advertisement fe80::98c3:5e9d:e2db:9090 (

```
> Frame 148: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
> Ethernet II, Src: VMware_86:75:3b (00:0c:29:86:75:3b), Dst: VMware_f8:8d:f3 (00:0c:29:f8:8
v Internet Protocol Version 6, Src: fe80::98c3:5e9d:e2db:9090, Dst: fe80::20c:29ff:fef8:8df3
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 32
    Next Header: ICMPv6 (58)
    Hop Limit: 255
    Source Address: fe80::98c3:5e9d:e2db:9090
    Destination Address: fe80::20c:29ff:fef8:8df3
    [Destination SLAAC MAC: VMware_f8:8d:f3 (00:0c:29:f8:8d:f3)]
v Internet Control Message Protocol v6
    Type: Neighbor Advertisement (136)
    Code: 0
    Checksum: 0xecc0 [correct]
    [Checksum Status: Good]
  > Flags: 0x60000000, Solicited, Override
    Target Address: fe80::98c3:5e9d:e2db:9090
  v ICMPv6 Option (Target link-layer address : 00:0c:29:86:75:3b)
      Type: Target link-layer address (2)
      Length: 1 (8 bytes)
      Link-layer address: VMware_86:75:3b (00:0c:29:86:75:3b)
```

```
0000  00 0c 29 f8 8d f3 00 0c  29 86 75 3b 86 dd 60 00
0010  00 00 00 20 3a ff fe 80  00 00 00 00 00 00 98 c3
0020  5e 9d e2 db 90 90 fe 80  00 00 00 00 00 00 02 0c
0030  29 ff fe f8 8d f3 88 00  ec c0 60 00 00 00 fe 80
0040  00 00 00 00 00 00 98 c3  5e 9d e2 db 90 90 02 01
0050  00 0c 29 86 75 3b
```

CVE-2021-24074

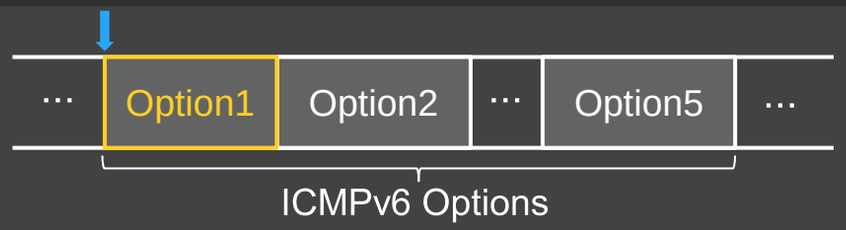
Integer Overflow

```
tcpip!Ipv6pHandleRouterAdvertisement
tcpip!Icmpv6ReceiveDatagrams+0x32b
tcpip!IppDeliverListToProtocol+0xf0
tcpip!IppProcessDeliverList+0x62
tcpip!IppReceiveHeaderBatch+0x214
tcpip!IppFlcReceivePacketsCore+0x315
tcpip!FlpReceiveNonPreValidatedNetBufferListChain+0x271
tcpip!FlReceiveNetBufferListChainCalloutRoutine+0xc2
nt!KeExpandKernelStackAndCalloutInternal+0x85
tcpip!FlReceiveNetBufferListChain+0xb6
NDIS!NdisMIndicateReceiveNetBufferLists+0x31c
vmswitch!VmsMpNicPvtPacketForward+0x238
vmswitch!VmsRouterDeliverNetBufferLists+0x390
vmswitch!VmsExtPtReceiveNetBufferLists+0x193
NDIS!ndisMIndicateNetBufferListsToOpen+0x11e
NDIS!ndisMTopReceiveNetBufferLists+0x267bc
NDIS!ndisCallReceiveHandler+0x47
NDIS!NdisMIndicateReceiveNetBufferLists+0x735
```

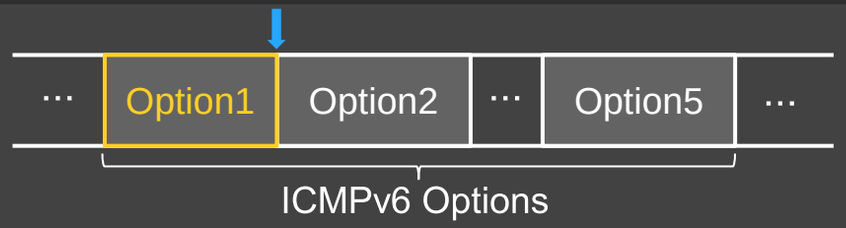
The control flow, originating from the vmswitch module, eventually enters the tcpip module

call stack

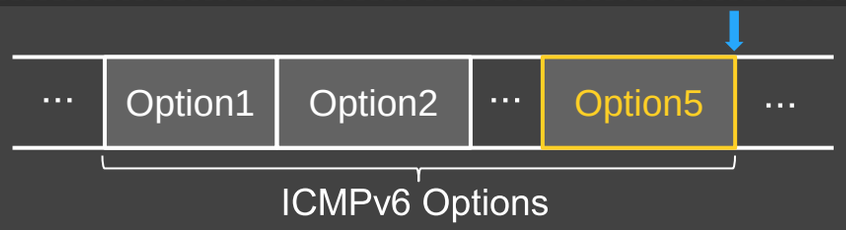
```
VOID Ipv6pHandleRouterAdvertisement(ICMPV6_MESSAGE *Icmpv6, IP_REQUEST_CONTROL_DATA *Args) {
    // ...
    USHORT ParsedLength; // (1)
    /* ... Validate the Router Advertisement ... */
    /* ... Get the Router Advertisement header ... */
    Advertisement = NetioGetDataBuffer(NetBuffer, sizeof(ND_ROUTER_ADVERT_HEADER), &AdvertisementBuffer, 1, 0);
    ParsedLength = sizeof(ND_ROUTER_ADVERT_HEADER);
    /* ... */
    while (Ipv6pParseTlvOption(NetBuffer, &Type, &Length)) { // (2) sanity-check the options
        switch (Type) {
            case ND_OPT_SOURCE_LINKADDR: // ...
            case ND_OPT_MTU: // ...
            case ND_OPT_PREFIX_INFORMATION: // ...
            case ND_OPT_ROUTE_INFO: // ...
        }
        // Move forward to the next option.
        // Keep track of the parsed length, so we can use it below to back up.
        NetioAdvanceNetBuffer(NetBuffer, Length); // (3)
        ParsedLength += Length;    // (4)
    }
    // ...
    NetioRetreatNetBuffer(NetBuffer, ParsedLength, 0);    // (5)
    // ...
```



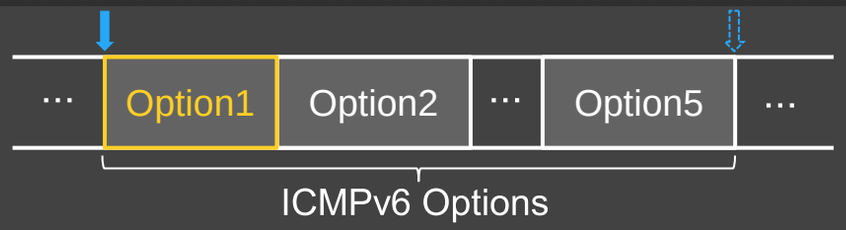
```
VOID Ipv6pHandleRouterAdvertisement(ICMPV6_MESSAGE *Icmpv6, IP_REQUEST_CONTROL_DATA *Args) {
    // ...
    USHORT ParsedLength; // (1)
    /* ... Validate the Router Advertisement ... */
    /* ... Get the Router Advertisement header ... */
    Advertisement = NetioGetDataBuffer(NetBuffer, sizeof(ND_ROUTER_ADVERT_HEADER), &AdvertisementBuffer, 1, 0);
    ParsedLength = sizeof(ND_ROUTER_ADVERT_HEADER);
    /* ... */
    while (Ipv6pParseTlvOption(NetBuffer, &Type, &Length)) { // (2) sanity-check the options
        switch (Type) {
            case ND_OPT_SOURCE_LINKADDR: // ...
            case ND_OPT_MTU: // ...
            case ND_OPT_PREFIX_INFORMATION: // ...
            case ND_OPT_ROUTE_INFO: // ...
        }
        // Move forward to the next option.
        // Keep track of the parsed length, so we can use it below to back up.
        NetioAdvanceNetBuffer(NetBuffer, Length); // (3)
        ParsedLength += Length;    // (4)
    }
    // ...
    NetioRetreatNetBuffer(NetBuffer, ParsedLength, 0);    // (5)
    // ...
```



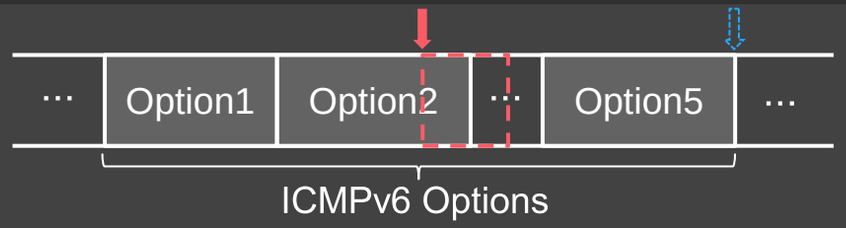
```
VOID Ipv6pHandleRouterAdvertisement(ICMPV6_MESSAGE *Icmpv6, IP_REQUEST_CONTROL_DATA *Args) {
    // ...
    USHORT ParsedLength; // (1)
    /* ... Validate the Router Advertisement ... */
    /* ... Get the Router Advertisement header ... */
    Advertisement = NetioGetDataBuffer(NetBuffer, sizeof(ND_ROUTER_ADVERT_HEADER), &AdvertisementBuffer, 1, 0);
    ParsedLength = sizeof(ND_ROUTER_ADVERT_HEADER);
    /* ... */
    while (Ipv6pParseTlvOption(NetBuffer, &Type, &Length)) { // (2) sanity-check the options
        switch (Type) {
            case ND_OPT_SOURCE_LINKADDR: // ...
            case ND_OPT_MTU: // ...
            case ND_OPT_PREFIX_INFORMATION: // ...
            case ND_OPT_ROUTE_INFO: // ...
        }
        // Move forward to the next option.
        // Keep track of the parsed length, so we can use it below to back up.
        NetioAdvanceNetBuffer(NetBuffer, Length); // (3)
        ParsedLength += Length;    // (4)
    }
    // ...
    NetioRetreatNetBuffer(NetBuffer, ParsedLength, 0);    // (5)
    // ...
```



```
VOID Ipv6pHandleRouterAdvertisement(ICMPV6_MESSAGE *Icmpv6, IP_REQUEST_CONTROL_DATA *Args) {
    // ...
    USHORT ParsedLength; // (1)
    /* ... Validate the Router Advertisement ... */
    /* ... Get the Router Advertisement header ... */
    Advertisement = NetioGetDataBuffer(NetBuffer, sizeof(ND_ROUTER_ADVERT_HEADER), &AdvertisementBuffer, 1, 0);
    ParsedLength = sizeof(ND_ROUTER_ADVERT_HEADER);
    /* ... */
    while (Ipv6pParseTlvOption(NetBuffer, &Type, &Length)) { // (2) sanity-check the options
        switch (Type) {
            case ND_OPT_SOURCE_LINKADDR: // ...
            case ND_OPT_MTU: // ...
            case ND_OPT_PREFIX_INFORMATION: // ...
            case ND_OPT_ROUTE_INFO: // ...
        }
        // Move forward to the next option.
        // Keep track of the parsed length, so we can use it below to back up.
        NetioAdvanceNetBuffer(NetBuffer, Length); // (3)
        ParsedLength += Length;    // (4)
    }
    // ...
    NetioRetreatNetBuffer(NetBuffer, ParsedLength, 0);    // (5)
    // ...
```



```
VOID Ipv6pHandleRouterAdvertisement(ICMPV6_MESSAGE *Icmpv6, IP_REQUEST_CONTROL_DATA *Args) {
    // ...
    USHORT ParsedLength; // (1)
    /* ... Validate the Router Advertisement ... */
    /* ... Get the Router Advertisement header ... */
    Advertisement = NetioGetDataBuffer(NetBuffer, sizeof(ND_ROUTER_ADVERT_HEADER), &AdvertisementBuffer, 1, 0);
    ParsedLength = sizeof(ND_ROUTER_ADVERT_HEADER);
    /* ... */
    while (Ipv6pParseTlvOption(NetBuffer, &Type, &Length)) { // (2) sanity-check the options
        switch (Type) {
            case ND_OPT_SOURCE_LINKADDR: // ...
            case ND_OPT_MTU: // ...
            case ND_OPT_PREFIX_INFORMATION: // ...
            case ND_OPT_ROUTE_INFO: // ...
        }
        // Move forward to the next option.
        // Keep track of the parsed length, so we can use it below to back up.
        NetioAdvanceNetBuffer(NetBuffer, Length); // (3)
        ParsedLength += Length;    // (4) integer overflow
    }
    // ...
    NetioRetreatNetBuffer(NetBuffer, ParsedLength, 0);    // (5)
    // ...
```



CVE-2022-30223

Out-of-bounds Read

Windows Hyper-V Information Disclosure Vulnerability

CVE-2022-30223
Security Vulnerability

Released: Jul 12, 2022

Assigning CNA: Microsoft

CVE-2022-30223

Impact: Information Disclosure Max Severity: Important

CVSS:3.1 5.7 / 5.0

Exploitability

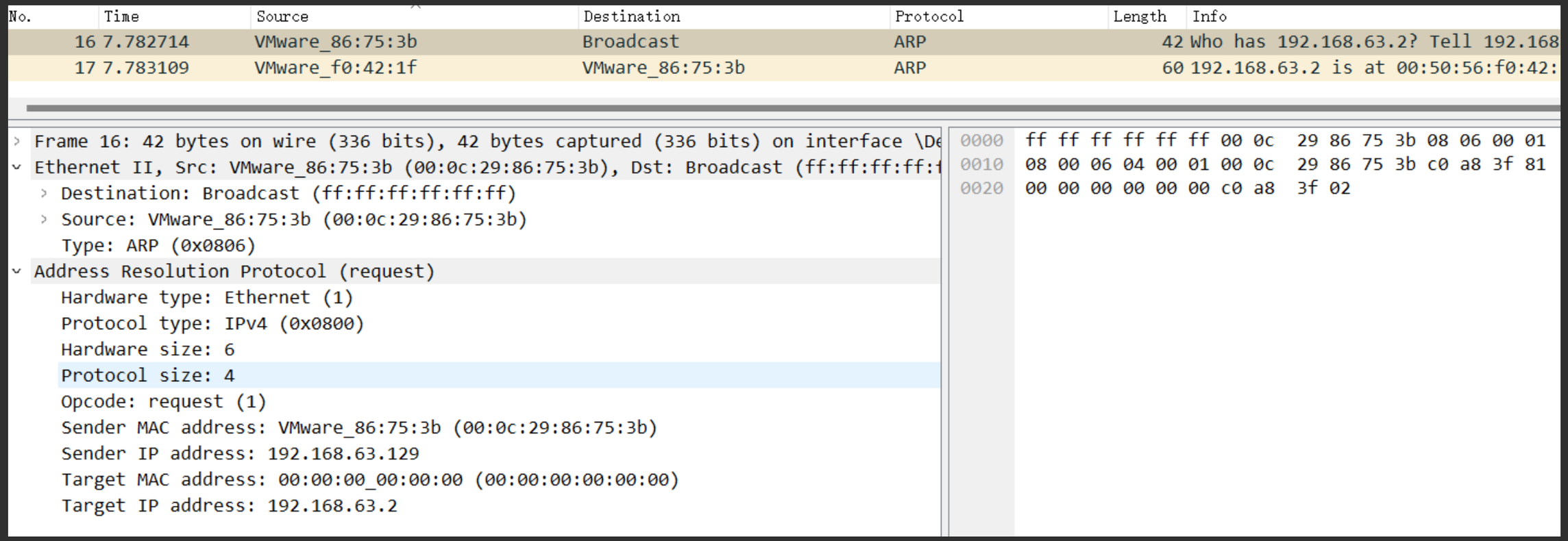
The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly disclosed	Exploited	Exploitability assessment

Caused by a single ARP packet whose length is only 15

CVE-2022-30223

Out-of-bounds Read



CVE-2022-30223

Out-of-bounds Read

```
vmswitch!VmsNblHelperCreateCloneNbl
vmswitch!VmsMpNicPvtPacketForward+0x308
vmswitch!VmsRouterDeliverNetBufferLists+0x81a
vmswitch!VmsExtPtReceiveNetBufferLists+0x193
NDIS!ndisMIndicateNetBufferListsToOpen+0x11e
NDIS!ndisMTopReceiveNetBufferLists+0x267bc
NDIS!ndisCallReceiveHandler+0x47
NDIS!NdisMIndicateReceiveNetBufferLists+0x735
vmswitch!VmsExtMpIndicatePackets+0xa55
vmswitch!VmsExtMpSendNetBufferLists+0x5a8
```

call stack

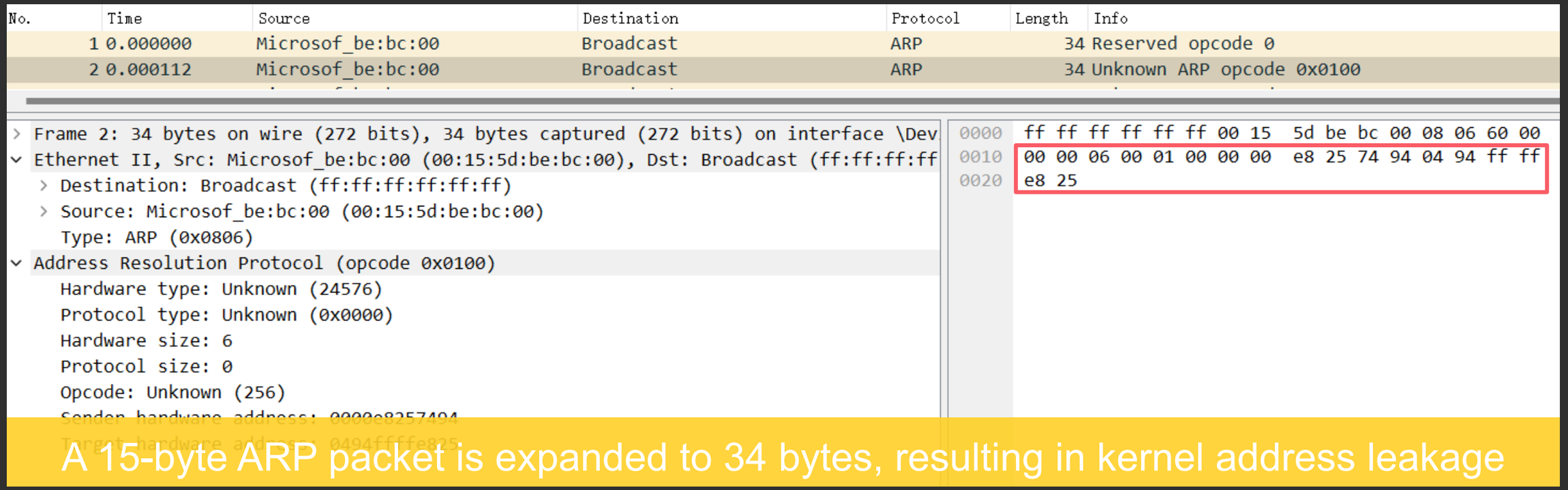
```
__int64 VmsNblHelperCreateCloneNbl(PNET_BUFFER_LIST SrcNetBufferList, NDIS_HANDLE NetBufferListPoolHandle, NDIS_HANDLE
NetBufferPoolHandle, char a4, char a5, char a6, int a7, __int64 a8) {
  // ...
  v11 = v10_SrcNetBufferList->NetBufferListInfo[0];
  if ( v11 && ((unsigned __int8)v11 & 0x1C) != 0 ) {
    // ...
    if ( ((unsigned __int8)v11 & 4) != 0 ) {
      // ...
LABEL_14:
      v57 = v12;
      NdisAdvanceNetBufferListDataStart(v10_SrcNetBufferList, v12, 0, 0i64);
      v56 = 1;
      goto LABEL_16;
    }
    if ( ((unsigned __int8)v11 & 8) == 0 ) {
      v12 = 34;  // (1)
      goto LABEL_14;
    }
    // ...
  }
  // ...
LABEL_16:
  // ...
  v21 = v12;    // (2)
  /* ... */
  while ( 1 ) {
    // ...
    v19_dstNetBufferList = NdisCopyFromNetBufferToNetBuffer(v26, 0, v21, v24, 0, &BytesCopied); // (3)
    // ...
```

C++

```
NDIS_EXPORTED_ROUTINE NDIS_STATUS NdisCopyFromNetBufferToNetBuffer(
  [in]  NET_BUFFER       *Destination,
  [in]  ULONG            DestinationOffset,
  [in]  ULONG            BytesToCopy,
  [in]  NET_BUFFER const *Source,
  [in]  ULONG            SourceOffset,
  [out] ULONG            *BytesCopied
);
```

CVE-2022-30223

Out-of-bounds Read



A 15-byte ARP packet is expanded to 34 bytes, resulting in kernel address leakage

CVE-XXXX-XXXX (not fixed yet)

NULL pointer deference

caused by a packet with only 8-byte IP header

RE: Re: Microsoft Bounty Program: Out-of-Scope Notification Case 71449 CRM:0022001410

安全浏览模式

发件人：Microsoft Security Response Center<secure@microsoft.com>

收件人：MSFT Bounty<bounty@microsoft.com> a4651386@163.com<a4651386@163.com>

抄送人：Microsoft Security Response Center<secure@microsoft.com> Microsoft Security Response Center<secure@microsoft.com>

Microsoft Security Response Center<secure@microsoft.com>

时　间：2022年09月24日 01:37 (星期六)

Hello Quan,

I'm sorry for the frustration in MSRC's outcome of this case. Since your test environment is using VMWare and ours is using Hyper-V, might we suggest we align our testing environments? To that end might we suggest that you create a new POC using only Microsoft Hyper-V and submit that POC as a new case submission. That would allow us to rotate the assessment engineer to a fresh set of eyes.

Thank you again for working with MSRC.

Regards,
Duncan

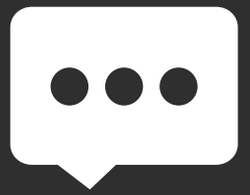
Microsoft Security...	RE: Re: Microsoft Bounty Program: Out-of-Scope Notification Case 7144...	2022-09-24
MSFT Bounty	RE: Microsoft Bounty Program: Out-of-Scope Notification Case 71449 CR...	2022-09-23
Microsoft Security...	RE: MSRC Case 71449 CRM:0022001410	2022-04-22
Microsoft Security...	MSRC Case 71449 CRM:0022001410	2022-04-20

Demo

Agenda

Introduction

Hyper-V Network Module Research



Vulnerability Analysis

Summary

What We Have Talked

- Virtual NIC is not total identical to physical network card. And the gap between them may break the protocol stack implementations, resulting in severe vulnerabilities
- An in-depth analysis of multiple vulnerabilities discovered by breaking the theoretical limits outlined by RFC
- A new point to guide the code review or fuzzing routine when targeting virtual NICs

Thanks!


TrueUnitySect


a4651386@163.com