Securing Self-Virtualizing Ethernet Devices

Igor Smolyar, Muli Ben-Yehuda, Dan Tsafrir

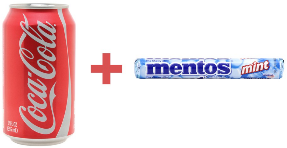
TECHNION
Israel Institute
of Technology

In a Nutshell

- Two technologies, **SRIOV** and **Ethernet Flow Control**, that are individually secure . . . . but collectively vulnerable
- We show **a new attack** where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a **vulnerability in self-virtualizing Ethernet NICs**
- We show **how to build** an attack-resistant NIC

In a Nutshell

- Two technologies, **SRIOV** and **Ethernet Flow Control**, that are individually secure . . . . but collectively vulnerable
- We show **a new attack** where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a **vulnerability in self-virtualizing Ethernet NICs**
- We show **how to build** an attack-resistant NIC



In a Nutshell

- Two technologies, **SRIOV** and **Ethernet Flow Control**, that are individually secure . . . . but collectively vulnerable
- We show **a new attack** where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a **vulnerability in self-virtualizing Ethernet NICs**
- We show **how to build** an attack-resistant NIC

In a Nutshell

- Two technologies, **SRIOV** and **Ethernet Flow Control**, that are individually secure . . . . but collectively vulnerable
- We show **a new attack** where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a **vulnerability in self-virtualizing Ethernet NICs**
- We show **how to build** an attack-resistant NIC

In a Nutshell

- Two technologies, **SRIOV** and **Ethernet Flow Control**, that are individually secure . . . . but collectively vulnerable
- We show **a new attack** where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a **vulnerability in self-virtualizing Ethernet NICs**
- We show **how to build** an attack-resistant NIC

Attack Ramifications

- To **defend** against the attack, you have to either:
  - **Trust** your virtual machines, or
  - **Give up** on flow control functionality and **lose performance**, or
  - **Modify** device hardware/firmware
- **All** evaluated vendors are vulnerable

Attack Ramifications

- To **defend** against the attack, you have to either:
  - **Trust** your virtual machines, or
  - **Give up** on flow control functionality and **lose performance**, or
  - **Modify** device hardware/firmware
- **All** evaluated vendors are vulnerable

Attack Ramifications

- To **defend** against the attack, you have to either:
  - **Trust** your virtual machines, or
  - **Give up** on flow control functionality and **lose performance**, or
  - **Modify** device hardware/firmware
- **All** evaluated vendors are vulnerable

Attack Ramifications

- To **defend** against the attack, you have to either:
  - **Trust** your virtual machines, or
  - **Give up** on flow control functionality and **lose performance**, or
  - **Modify** device hardware/firmware
- **All** evaluated vendors are vulnerable

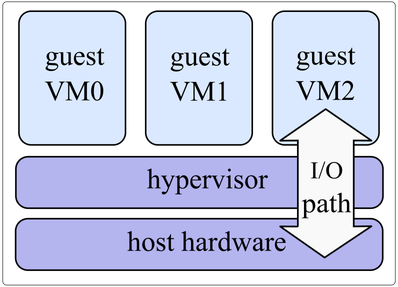
Attack Ramifications

- To **defend** against the attack, you have to either:
  - **Trust** your virtual machines, or
  - **Give up** on flow control functionality and **lose performance**, or
  - **Modify** device hardware/firmware
- **All** evaluated vendors are vulnerable

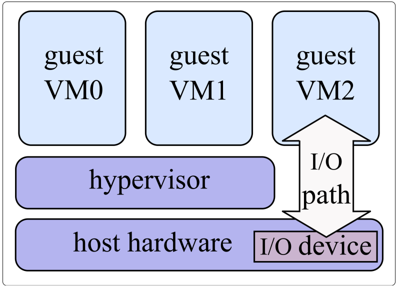
Attack Ramifications

- To **defend** against the attack, you have to either:
  - **Trust** your virtual machines, or
  - **Give up** on flow control functionality and **lose performance**, or
  - **Modify** device hardware/firmware
- **All** evaluated vendors are vulnerable

Background: Types of I/O Virtualization



Emulation & Para-virtualization



Direct I/O Device Assignment

Background: Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Limitation: not scalable** 5–10 I/O devices per host, **but** 50–100 virtual machines per host
- **Self-virtualizing devices are scalable!** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

Background: Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Limitation: not scalable** 5–10 I/O devices per host, **but** 50–100 virtual machines per host
- **Self-virtualizing devices are scalable!** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

Background: Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Limitation: not scalable** 5–10 I/O devices per host, **but** 50–100 virtual machines per host
- **Self-virtualizing devices are scalable!** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

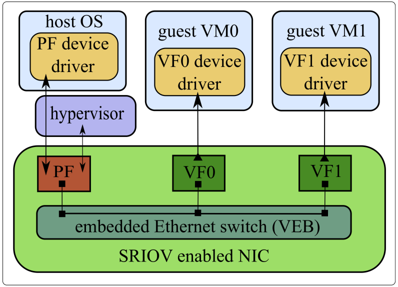
Background: Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Limitation: not scalable** 5–10 I/O devices per host, **but** 50–100 virtual machines per host
- **Self-virtualizing devices are scalable!** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

Single Root I/O Virtualization (SRIOV)

Each SRIOV capable device consists of at least one Physical Function (PF) and multiple virtual partitions called Virtual Functions (VF)

- **PF** is a standard PCIe function with full configuration space. Can control entire PCI device and perform I/O operations
- **VF** is a “lightweight” PCI function that implements only a subset of standard PCI functionality, mostly performs I/O



SRIOV NIC in a virtualized environment

Where is SRIOV Used?

- **HPC** with SRIOV it is possible to virtualize HPC setups.
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic appliance uses SRIOV technology to share the internal network

Where is SRIOV Used?

- **HPC** with SRIOV it is possible to virtualize HPC setups.
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic appliance uses SRIOV technology to share the internal network

Where is SRIOV Used?

- **HPC** with SRIOV it is possible to virtualize HPC setups.
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic appliance uses SRIOV technology to share the internal network

Where is SRIOV Used?

- **HPC** with SRIOV it is possible to virtualize HPC setups.
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic appliance uses SRIOV technology to share the internal network

Ethernet Flow Control

- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

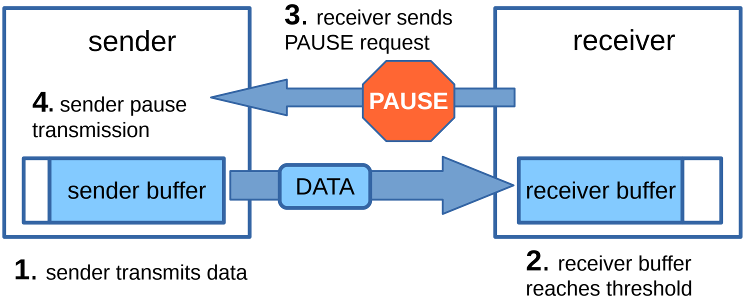
- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time

Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time



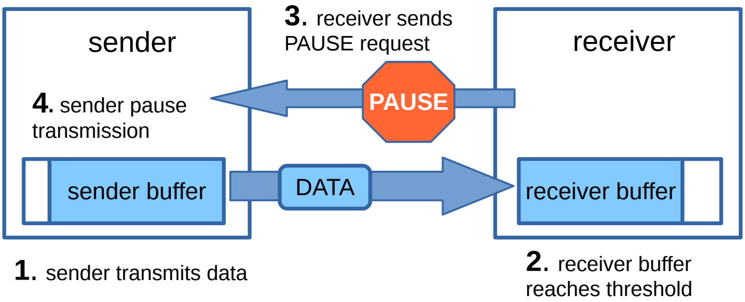
Ethernet Flow Control

❶ The sender (e.g. Ethernet switch) transmits data faster than the receiver can process

❷ The receiver (e.g. host's Ethernet NIC) runs out of space

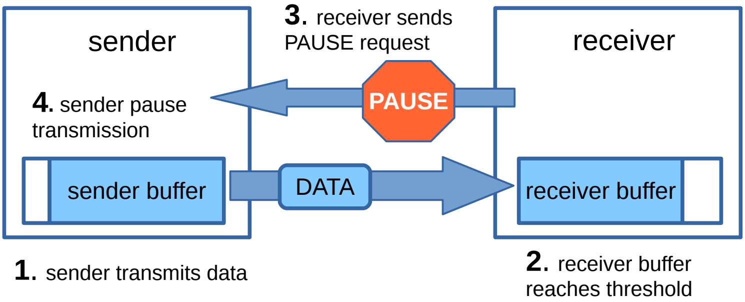
❸ The receiver sends the sender a MAC control frame with a pause request

❹ The sender stops transmitting data for requested period of time



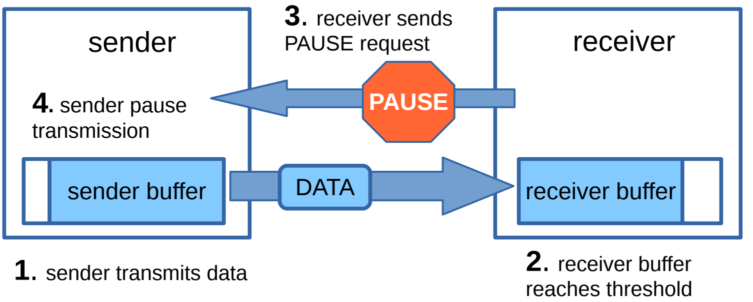
Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time



Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time



Ethernet Flow Control

- When receiver runs out of the buffers, what is the required pause frame rate to stop the transmission completely?

Ethernet Flow Control

- When receiver runs out of the buffers, what is the required pause frame rate to stop the transmission completely?

Ethernet Flow Control

- When receiver runs out of the buffers, what is the required pause frame rate to stop the transmission completely?

link speed [Gbps]	single frame pause time [ms]	frame rate required to stop transmission [frames/second]
1	33.6	30
10	3.36	299
40	0.85	1193

Attacking VMs via Flow Control

- **Flow Control** works on link-level
- **Link is shared** between VMs; all VMs with direct access to the VFs of the same PF share the same physical link to the edge switch
- **Each FC Pause Frame** halts traffic on the *entire* link
- **All VFs** associated with this PF are affected

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

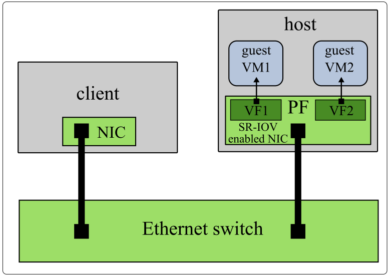
The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues...
- **Until the malicious VM sends the next pause frame**

Attack Evaluation—Setup

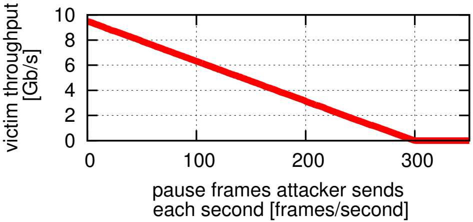
Our testbed consists of two servers: one acting as the client and the other as the host with SRIOV capable NIC

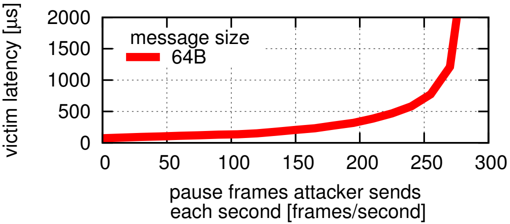
- VF1 assigned to VM1 and VF2 to VM2
- traffic generated between VM1 and the client
- VM2 attacks VM1 by sending PAUSE frames to the switch



Pause Frames Attack Results using SRIOV 10GbE NIC

- On the left, victim **throughput** as a function of pause frames rate
- On the right, victim **latency** as a function of pause frames rate





What do we propose?

- **Filter** outbound traffic transmitted by a VF
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF
- **In ideal world** we would modify NIC's hardware/firmware to filter out malicious pause frames
- But ... in reality NIC uses **proprietary closed firmware and hardware** ☹
- So, we built a Virtualization-Aware Network Flow Controller (VANFC)—**a software-based prototype** of an SRIOV Ethernet NIC with pause frame filtering.

What do we propose?

- **Filter** outbound traffic transmitted by a VF
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF
- **In ideal world** we would modify NIC's hardware/firmware to filter out malicious pause frames
- But ... in reality NIC uses **proprietary closed firmware and hardware** ☹
- So, we built a Virtualization-Aware Network Flow Controller (VANFC)—**a software-based prototype** of an SRIOV Ethernet NIC with pause frame filtering.

What do we propose?

- **Filter** outbound traffic transmitted by a VF
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF
- **In ideal world** we would modify NIC's hardware/firmware to filter out malicious pause frames
- But . . . in reality NIC uses **proprietary closed firmware and hardware** ☹
- So, we built a Virtualization-Aware Network Flow Controller (VANFC)—**a software-based prototype** of an SRIOV Ethernet NIC with pause frame filtering.

What do we propose?

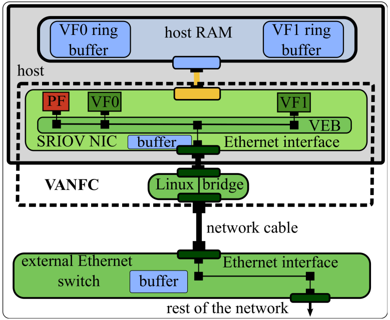
- **Filter** outbound traffic transmitted by a VF
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF
- **In ideal world** we would modify NIC's hardware/firmware to filter out malicious pause frames
- But . . . in reality NIC uses **proprietary closed firmware and hardware** ☹
- So, we built a Virtualization-Aware Network Flow Controller (VANFC)—**a software-based prototype** of an SRIOV Ethernet NIC with pause frame filtering.

What do we propose?

- **Filter** outbound traffic transmitted by a VF
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF
- **In ideal world** we would modify NIC's hardware/firmware to filter out malicious pause frames
- But . . . in reality NIC uses **proprietary closed firmware and hardware** ☹
- So, we built a Virtualization-Aware Network Flow Controller (VANFC)—**a software-based prototype** of an SRIOV Ethernet NIC with pause frame filtering.

The VANFC design

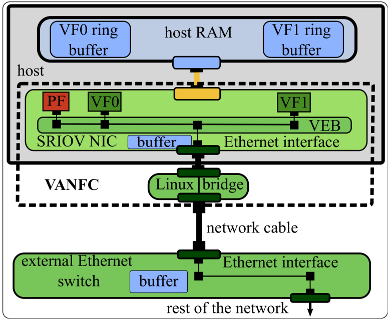
- **Internal switch** replicates Ethernet switch
- We add **a software based extension** to the NIC internal switch
- Switch extension **filters malicious pause frames** with source address of a VFs



Schema of VANFC

The VANFC design

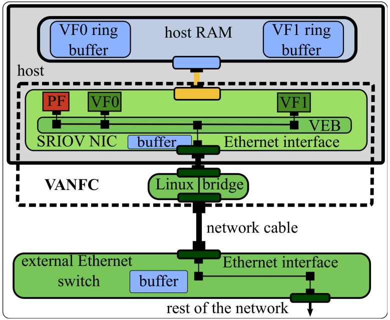
- **Internal switch** replicates Ethernet switch
- We add **a software based extension** to the NIC internal switch
- Switch extension **filters malicious pause frames** with source address of a VFs



Schema of VANFC

The VANFC design

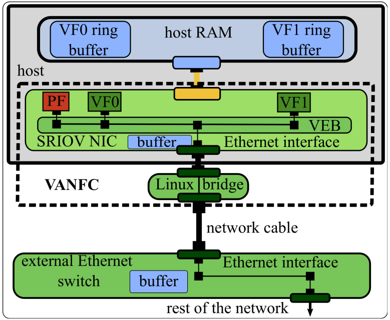
- **Internal switch** replicates Ethernet switch
- We add **a software based extension** to the NIC internal switch
- Switch extension **filters malicious pause frames** with source address of a VFs



Schema of VANFC

The VANFC design

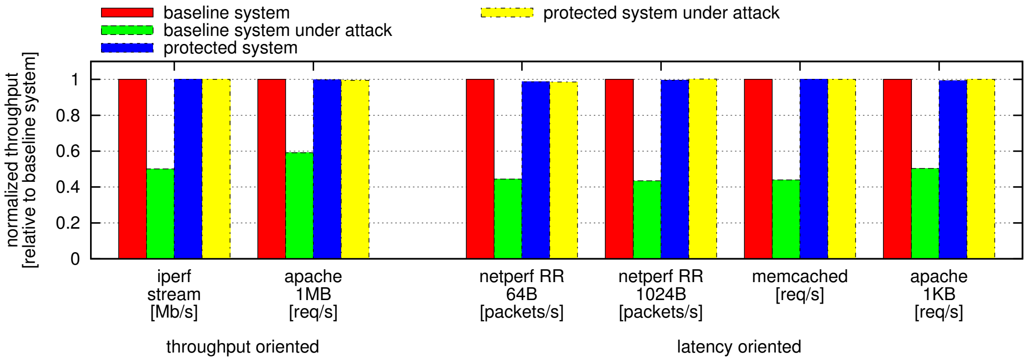
- **Internal switch** replicates Ethernet switch
- We add **a software based extension** to the NIC internal switch
- Switch extension **filters malicious pause frames** with source address of a VFs



Schema of VANFC

VANFC performance evaluation results in 10GbE setup

- We attack with 150 pause frames per second, to reduce perfromance to 50%
- **VANFC completely blocks the attack and introduces no performance penalty**



Current Situation in the Industry

- **Mellanox**
  - updated firmware for ConnectX-3 NICs provided for testing and planned to be released in August
  - fix for ConnectX-4 is in progress
- **Qlogic (Broadcom chip)**
  - updated driver/firmware provided for testing
  - fix is based on the proposed solution of filtering malicious pause frames sent from VF's
  - public release is in progress
- **Other major vendors were notified**, no solution provided yet

Current Situation in the Industry

- **Mellanox**
  - updated firmware for ConnectX-3 NICs provided for testing and planned to be released in August
  - fix for ConnectX-4 is in progress
- **Qlogic (Broadcom chip)**
  - updated driver/firmware provided for testing
  - fix is based on the proposed solution of filtering malicious pause frames sent from VF's
  - public release is in progress
- **Other major vendors were notified**, no solution provided yet

Current Situation in the Industry

- **Mellanox**
  - updated firmware for ConnectX-3 NICs provided for testing and planned to be released in August
  - fix for ConnectX-4 is in progress
- **Qlogic (Broadcom chip)**
  - updated driver/firmware provided for testing
  - fix is based on the proposed solution of filtering malicious pause frames sent from VF's
  - public release is in progress
- **Other major vendors were notified**, no solution provided yet

Current Situation in the Industry

- **Mellanox**
  - updated firmware for ConnectX-3 NICs provided for testing and planned to be released in August
  - fix for ConnectX-4 is in progress
- **Qlogic (Broadcom chip)**
  - updated driver/firmware provided for testing
  - fix is based on the proposed solution of filtering malicious pause frames sent from VF's
  - public release is in progress
- **Other major vendors were notified**, no solution provided yet

Conclusions

- Removing host from the I/O path requires adding **filtering functionality** to the hardware
- Current implementation of Ethernet SRIOV devices is **incompatible with Ethernet flow control**
- We show the **new attack** that exploits vulnerability in Ethernet SRIOV devices
- Our solution, the VANFC, **prevents the attack** and imposes no overhead
- **Future work:** extend our findings to SRIOV InfiniBand, Fiber Channel, NVMe SSD and GPGU

Thank You

Questions?