TOKYO, JAPAN / DECEMBER 11-13, 2025

BOSTON UNIVERSITY

IBM Research

Red Hat

VFIO: Very Frightening I/O? Taming Wild Guests and their PCIe Config-Space Abuse

Chathura Rajapaksha*, Sandhya Koteshwara#, Bandan Das+, Apoorve Mohan#,
Hubertus Franke#, Ajay Joshi*, Manuel Egele*

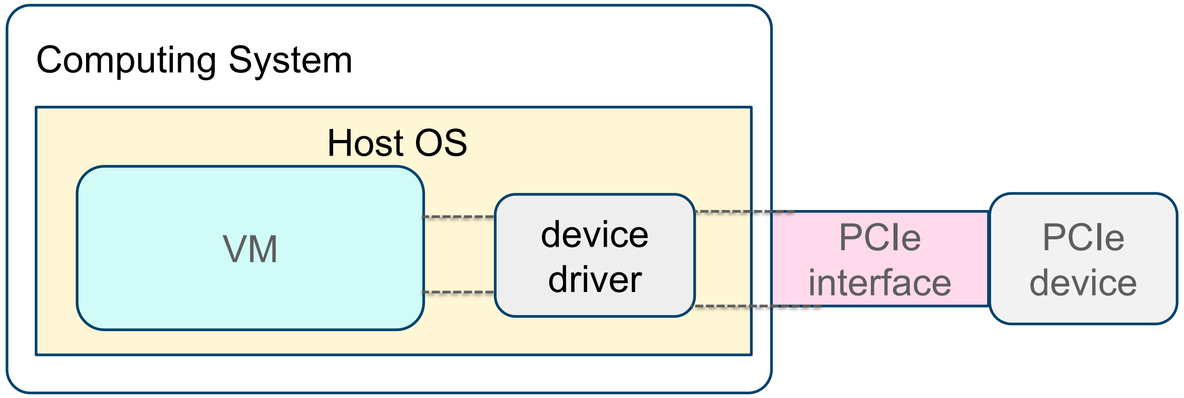
*Boston University; #IBM Research; +Red Hat

Outline

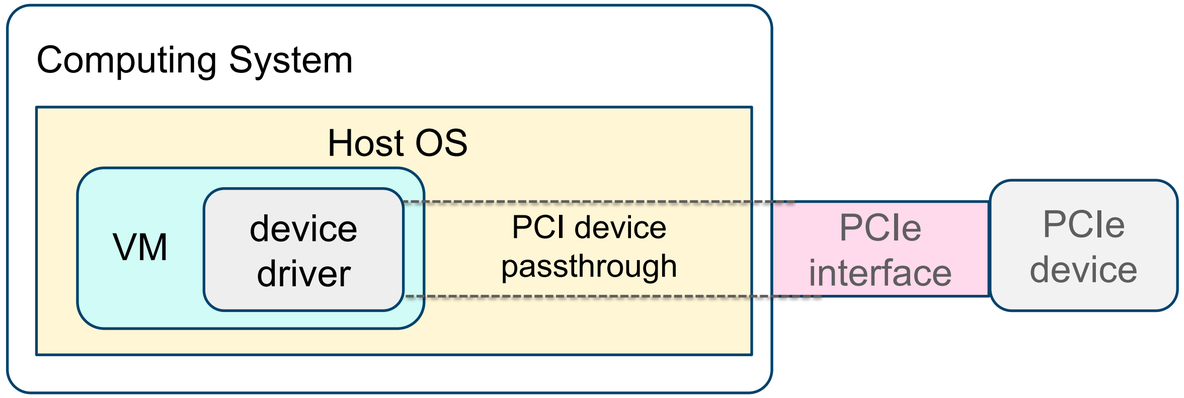
- **Background:** PCI(e) Passthrough, Configuration Space, and Mappings
- **Problem:** RAS Risks with PCIe Passthrough
- **Observation:** Exploring RAS risks with Fuzzing, Root Cause
- **Mitigation:** Proposed VFIO Patch
- **Discussion:** Community Feedback, Vendor Discussions, Next Steps

Background: PCI(e) Passthrough

- The multiple layers involved in VM I/O operations introduce additional overhead, which reduces I/O performance compared to direct hardware access.

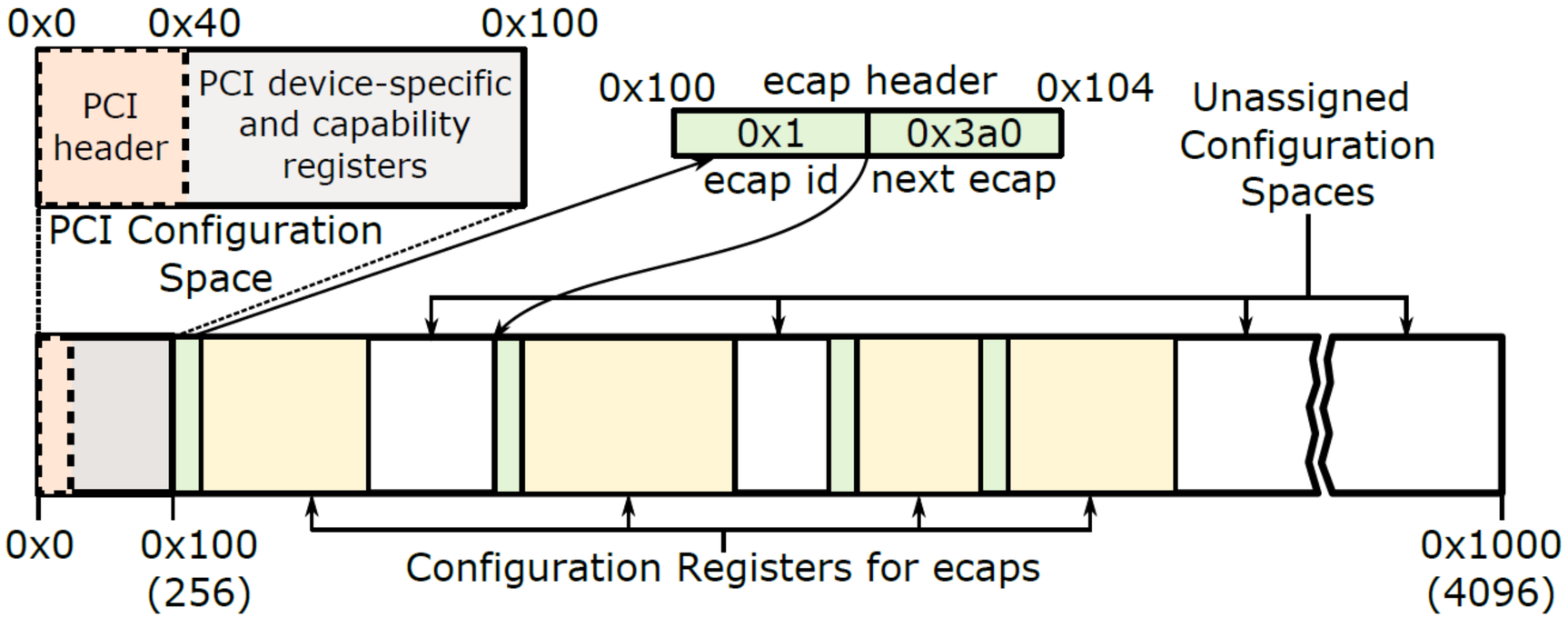


- PCI(e) passthrough, also known as direct device assignment, allows a physical PCI(e) device to be directly assigned to a VM, enabling the VM to achieve near-native I/O performance.

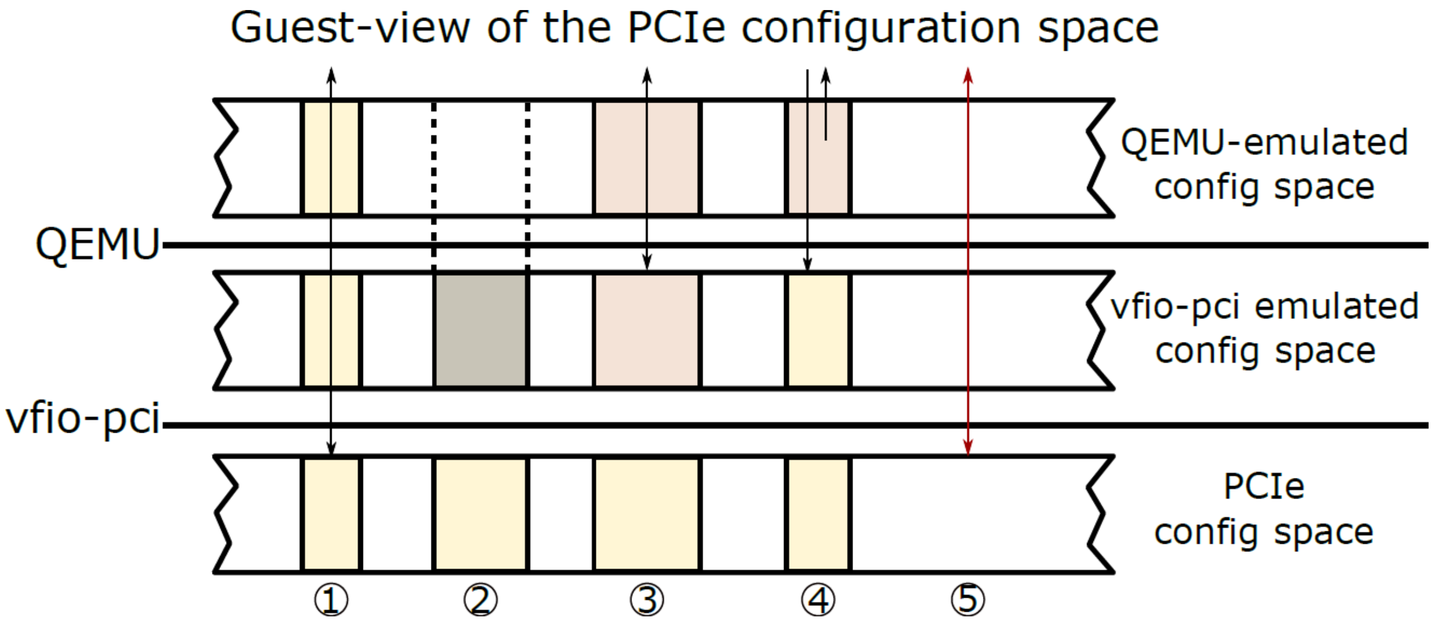


Background: PCI(e) Configuration Space

- Legacy config space: 0-255 bytes (PCI & PCIe devices)
- Extended config space: 256-4096 bytes (PCIe devices only)
- In the extended configuration space, many regions are unassigned



Background: PCI(e) configuration space mappings

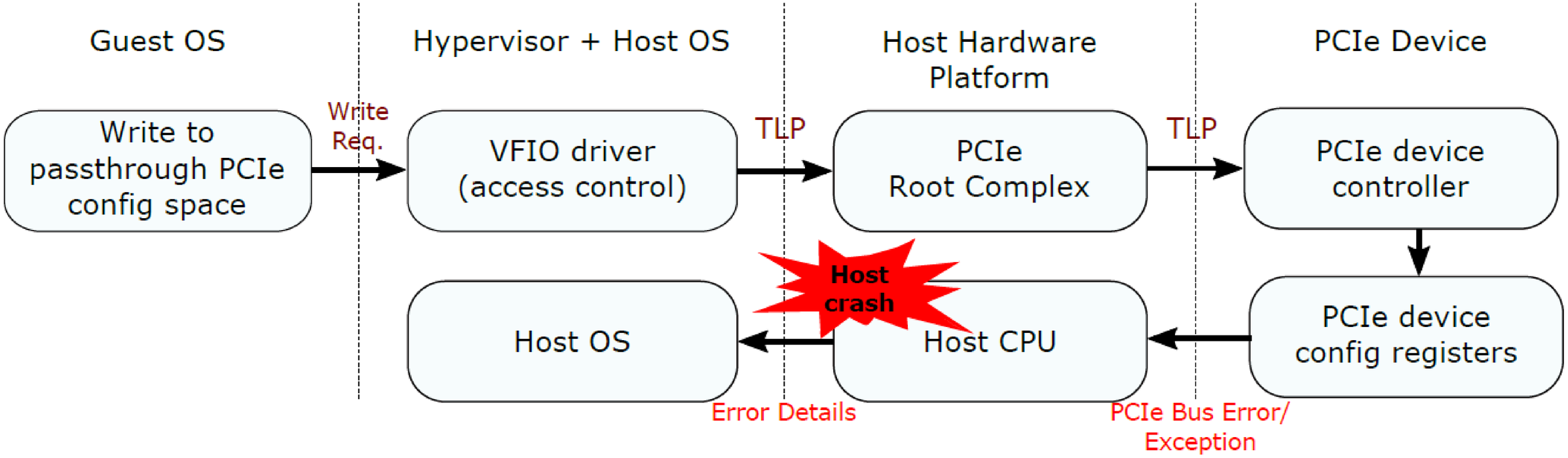


1. Passthrough register
2. Register hidden by vfio-pci
3. Register emulated by vfio-pci
4. Register emulated by QEMU
5. Unassigned PCI registers.

Problem: RAS risks with PCI(e) passthrough

Host crash:

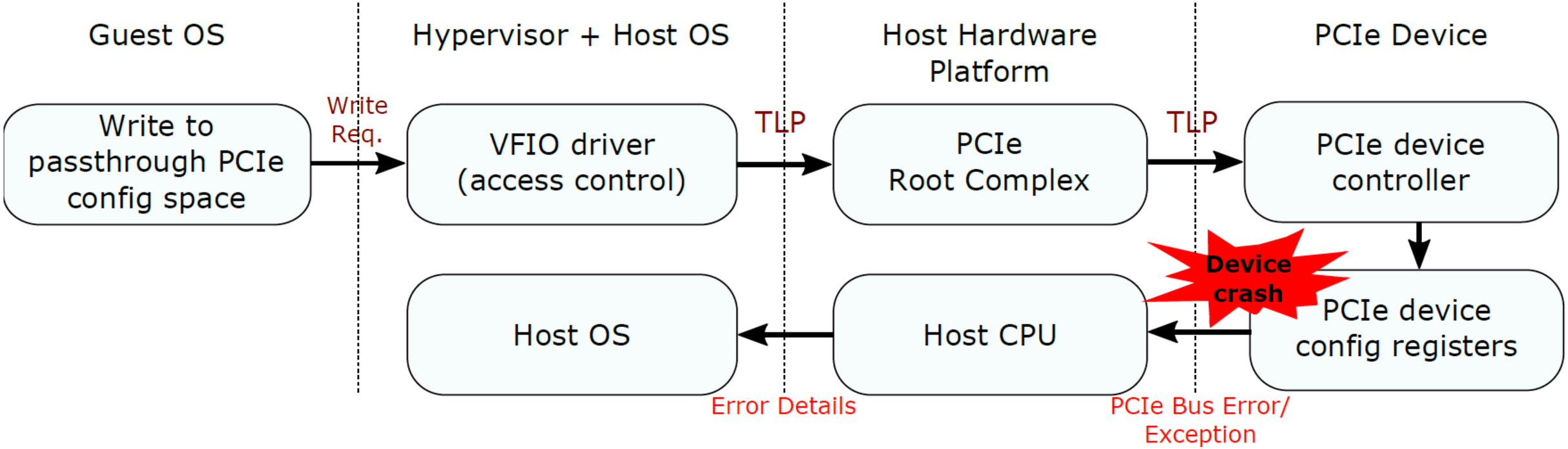
- Fatal PCI(e) errors can be induced from Guest
- Results in exceptions that cannot be handled by CPU



Problem: RAS risks with PCI(e) passthrough

Device crash:

- Device in unrecoverable state
- Unavailable for reassignment – requires system reboot



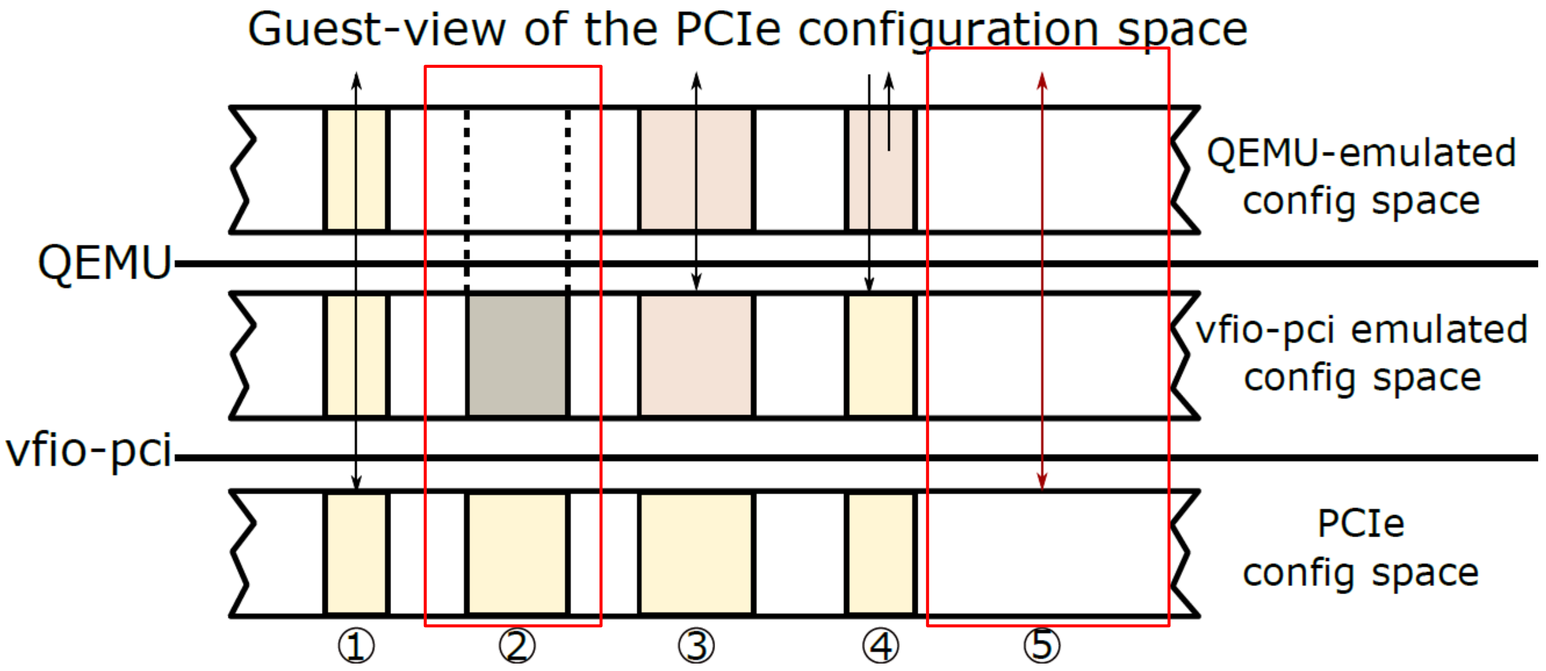
Observations: Exploring RAS risks

- Fuzzing* was used to randomly write data at random offsets of PCI configuration space
- 10 RAS failures were observed in 4 device categories

Failure No.	Device type	Instance type	Trigger criteria	Error type	PCIe capability region of the offset	PCIe config space region
1	NVMe	VM	Single Write	Host system crash/device reset	Unassigned	Extended
2	NVMe	VM	Single Write	Device crash	Unassigned	Extended
3	NVMe	VM	Single Write	Host system crash	Unassigned	Extended
4	NVMe	VM	Single Write	Device crash	Unassigned	Extended
5	GPU	VM	Single Write	Host system crash	Unassigned	Extended
6	GPU	VM	Multiple Writes	Host system crash	Unassigned	Extended
7	GPU	VM	Multiple Writes	Host system crash	Unassigned	Extended
8	NIC	VM	Single Write	Host system crash	Power Management	Legacy
9	SmartNIC	Bare-metal	Single Write	Device crash	Investigated by vendor	Extended
10	SmartNIC	Bare-metal	Multiple Writes	Device crash/loss of connectivity	Investigated by vendor	Extended

*Open-source fuzz tool developed in this work: https://github.com/IBM/fuzz-testing

Observations: Root cause



1. Passthrough register
2. Register hidden by vfio-pci
3. Register emulated by vfio-pci
4. Register emulated by QEMU
5. Unassigned PCI registers.

Mitigation: Proposed Patch

- Ideally, device vendors should block bad writes to PCIe config space
- Until then, cloud providers should have ability to block/mask PCIe config space writes
- Proposed patch blocks unassigned config space access only

[RFC PATCH] vfio/pci: Block and audit accesses to unassigned config regions

1. Support for blocking guest accesses to unassigned PCI configuration space
    - Ability to bypass this access control for specific devices
    - Three module parameters: **`block_pci_unassigned_write`**, **`block_pci_unassigned_read`**, **`uaccess_allow_ids`**

2. Auditing support for config space accesses to unassigned regions
    - Controlled via a new Kconfig option **`CONFIG_VFIO_PCI_UNASSIGNED_ACCESS_AUDIT`**
    - New audit event type, **`AUDIT_VFIO`**, has been introduced

Discussion and Feedback on proposed patch

Importance and Scope:

- **Critical for RAS** vs Security Issue?
- Can device generate same behavior via **MMIO or IO register** access?
- **PCIe Physical Function** vs Virtual Function?

Implementation:

- **Toggle switch to turn on/off to protect cloud providers?** Trapping access to device can lead to **limitations in use and performance**?
- **Per Device** vs System Wide?
- **Long-Term Solution**: We **need to robustly design HW** to have full containment as well (including managing errors)
- **Easier to add an option to qemu** to make it block any address not in a cap chain than to add a bunch of PCI ID tables and detection to the kernel
- **VFIO** vs generic PCI function?

Discussion: Device vendor responsibilities

- Although PCIe 2.0 introduced a standard mechanism for exposing vendor-specific capabilities (VSEC), vendors instead place these registers in unassigned regions
- Many unassigned configuration space regions contain registers for internal features or customer-specific requirements.
- Device vendors already have PF/VF. Device vendors should distinguish the Host and Guest environment. Maybe in the form of a list that can be enumerated by host and blocked accordingly

Robust device design is the right approach but is a long-term solution. Meanwhile, we need a short-term solution for immediate deployment and for systems already in production

Discussion: Next Steps

Where do we go from here?

- Codesign vs independent ownership
- Extending exploration to other areas: MMIO, BAR, VF ..
- Turn off switch for cloud vendors
- Upstreaming the proposed patch
- Ad-hoc mitigation techniques:
  - i440x chipset which prevents access to extended config space
  - Kernel lockdown node prohibiting all guest writes

BOSTON UNIVERSITY

IBM Research

Red Hat

Thank You!

東京 2025
LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

BOSTON UNIVERSITY

IBM Research

Red Hat

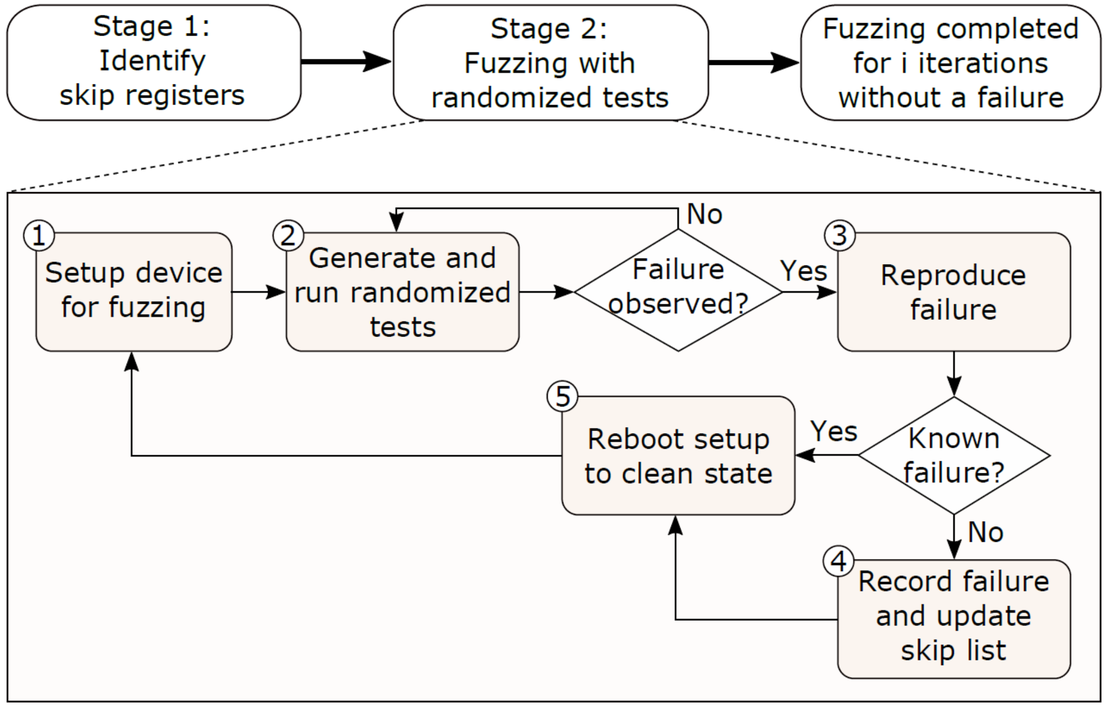
Backup Slides

東京 2025
LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

Exploring RAS risks with fuzzing

- Fuzzing, or fuzz testing, is a popular software testing technique that involves running target software with random or mutated inputs to discover bugs.

- We fuzzed the passthrough PCIe devices via the PCI config space exposed to the VMs by performing config space reads and writes.



Detailed table

Failure No	Device			CPU (x86)		Instance Type	Trigger Criteria	Error Type	PCIe capability region of the offset	PCIe config space region
	Vendor	Type	Model	Vendor	Model					
1	A	NVMe	dev1	X	CPU 1	VM	SW*	Host system crash/device reset	unassigned	Extended
2							SW*	Device crash	unassigned	Extended
3	A	NVMe	dev2	X	CPU 2	VM	SW	Host system crash	unassigned	Extended
4	A	NVMe	dev2	Y	CPU 3	VM	SW	Device crash	unassigned	Extended
5	B	GPU	dev3	X	CPU 4	VM	SW	Host system crash	unassigned	Extended
6	B	GPU	dev4	Y	CPU 5	VM	MW	Host system crash	unassigned	Extended
7	B	GPU	dev5	X	CPU 6	VM	MW	Host system crash	unassigned	Extended
8	C	NIC	dev6	Y	CPU 7	VM	SW	Host system crash	Power Management	Legacy
9	D	SmartNIC	dev7	X	CPU 8	BM	SW	Device crash	Investigated by vendor	Extended
10	E	SmartNIC	dev8	X	CPU 8	BM	MW	Device crash/loss of connectivity	Investigated by vendor	Extended

* [RFC PATCH 0/2] vfio/pci: Block and audit accesses to unassigned config regions

@ 2025-04-26 21:22 Chathura Rajapaksha
2025-04-26 21:22 ` [RFC PATCH 1/2] block accesses to unassigned PCI " Chathura Rajapaksha
` (2 more replies)
0 siblings, 3 replies; 15+ messages in thread
From: Chathura Rajapaksha @ 2025-04-26 21:22 UTC (permalink / raw)
To: kvm
Cc: Chathura Rajapaksha, Alex Williamson, Paul Moore, Eric Paris,
Giovanni Cabiddu, Xin Zeng, Yahui Cao, Bjorn Helgaas, Kevin Tian,
Niklas Schnelle, Yunxiang Li, Dongdong Zhang, Avihai Horon,
linux-kernel, audit

Some PCIe devices trigger PCI bus errors when accesses are made to
unassigned regions within their PCI configuration space. On certain
platforms, this can lead to host system hangs or reboots.

The current vfio-pci driver allows guests to access unassigned regions
in the PCI configuration space. Therefore, when such a device is passed
through to a guest, the guest can induce a host system hang or reboot
through crafted configuration space accesses, posing a threat to
system availability.

This patch series introduces:
1. Support for blocking guest accesses to unassigned
   PCI configuration space, and the ability to bypass this access control
   for specific devices. The patch introduces three module parameters:

   block_pci_unassigned_write:
   Blocks write accesses to unassigned config space regions.

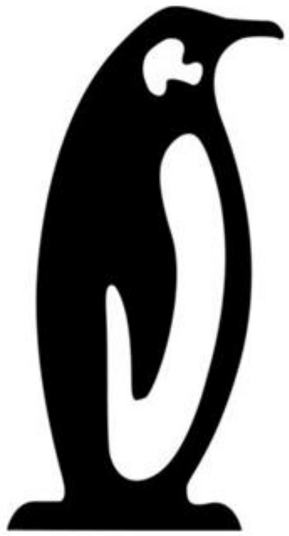
   block_pci_unassigned_read:
   Blocks read accesses to unassigned config space regions.

   uaccess_allow_ids:
   Specifies the devices for which the above access control is bypassed.
   The value is a comma-separated list of device IDs in
   <vendor_id>:<device_id> format.

   Example usage:

Open questions/Future work

- Open source fuzz - pointer
- Fuzzing exposed BAR space/MMIO is a WiP
- Failures have only been observed in PF so far. VF needs further investigation
- What happens if there is no VFIO?
  - Control plane can be affected in BM
- Other approaches to mitigation:
  - i440x chipset which prevents access to extended config space
  - Kernel lockdown node prohibiting all guest writes



東京 2025
LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025