CSE290X Project Proposal: Drywall: Recover from the Covert-Channel in CXL.cache by shutting down the CXL.cache Device

Yiwei Yang
University of California, Santa Cruz
Santa Cruz, California
yyang363@ucsc.edu

Yangyu Chen
Chongqing University
Chongqing, China
cyyself@cqu.edu.cn

ABSTRACT

Emerging CXL.cache protocol provides a cache-able interconnect based on the PCIe link, we know that the PCIe device can actually hurt the code not only in the outer PCIe Bus but inside the CPU running processes[5]. If the type1 CXL device shuts down by the attacker, the online cached cache line, which is exclusive, will drain down the running process. If the process is running in the kernel, it may get UCEs[1], which may possibly shut the kernel and leak sensitive information. We will come up with a sneaky way of detecting this by knowing the ownership and state of each cache line in real-time and will give hooks on the CXL fabric manager to kill the corresponding process, setting the cache line to invalid and giving the kernel proper way to escape the possibility of crashes. If the process is running in TDX[4], although, the exclusive cache line state can make automatically protect the cache from re-accessing the IOMMU which is automatically protected from this kind of attack, we still need to use CXL software-hardware co-design to make sure the state in the guest kernel does not hurt by reverse traversing the Nested Page. If the attacker is able to modify the cache line information physically, we should be cast the same thing as we did in bare metal.

1 INTRODUCTION

With PCIe devices already have many attacks like timing side-channel attacks based on timing differences in ioctl[6] to the different topology of the devices, and mitigation of fuzzing kernel[5] by PCIe DMA cache line level firewall protection, we found the topic interesting of timing side-channel on CXL or physical shutdown one CXL devices to attack the kernel location. Since the feature of CXL.cache enables devices to take ownership of cache lines inside the CPU and cached there. It's very dangerous if we shut down the CXL.cache devices(or it's offline accidentally) if they are currently having ownership of the cache line. If the cache line happens to be in the kernel, the next time we fetched from a killed device or mal-constructed device a malicious cache line, it will crash the kernel to some extent. By possible constructions of payloads, which for the kernel is the covert channel, we can destroy the kernel inside the enclave also, if the device has some cache lines inside the kernel and the TDX TEE-IO only enclaves the IOMMU part.

2 BACKGROUND

As CXL spec declared [3], non-demand uncorrected errors detected by a device (e.g., memory scrub logic in CXL device memory controller) that does not support the Memory Error Logging and Signaling Enhancements will be signaled to the device driver via a device MSI or MSI-X. Any corrected memory errors will be signaled to the device driver via a device MSI or MSI-X. The driver may choose to deallocate memory pages that have repeated errors. Neither the platform firmware nor the OS directly deals with these errors. An eRCD device may implement the capabilities described, in which case a device driver is not required.

3 PROPOSED DESIGN

Here's the proposed design of both bare metal and TDX-enabled container's threat model.

3.1 Threat Model for Bare Metal Machine

We suppose the CXL.cache devices. OSes are trusted. Attackers, which run some process in the kernel, have some way to in software(using side channel or user-space sudo promotion) know the device occupies what cache line in what code section, kill the device timing precisely, and let the cache line points to the payloads cache line. The payload cache line can therefore crash the kernel or print the sensitive information.

We want to have a modified kernel together with CXL fabric manage hardware-software[a] co-design to protect from the crash of process which may leak the information and prevent the kernel crash.

3.2 Threat Model for Containers with TDX TEE-IO[4]

We suppose the CXL.cache devices, and OSes are trusted and the hypervisor is untrusted. The virtual machine are applying enclave to protect the data access to CXL.cache devices. The attacker shuts one of the CXL.cache devices and knows the device occupies what cache line in what code section, emulating the device and giving some payloads to the kernel's desired location of cache line. The TDX protect that if the cache line's state is exclusive, it will never go out of the enclave. So the attacker need to assure the cache line state is not exclusive to the CXL.cache, which can be done by a fined-grained CXL.cache firewall.

After constructing the attack, we can simply prevent the specified downed devices to be exclusive and sending invalid to that cache line after the OS is aware of the downed devices.

4 PROPOSED EVALUATION

Our proposed evaluation will be first implemented on the QEMU for PoC and then for future submission to the conference and wait until the physical devices are coming we will implement real-world recovery resolutions.

4.1 QEMU implementation

We first implement the naive CXL devices that run basic type-1 CXL.cache operations. Then we first run some software that can observe the kernel and kills the CXL.cache device on demand to show whether the attacker is feasible or not for both the bare metal and TDX TEE-IO case. At last, we come up with a software-hardware co-design to prevent the leakage and crash by killing the process and invalidating the cache line.

4.2 Physical Device implementation

Here, we think we can make it simpler to show the proof of concept. We decide to use the intel sapphire rapids 6438y and Bittware CXL FPGA. We will first test the timing of hot unplugging the device to see the effect of the Linux kernel mechanism and try to make the qemu-emulated state machine we implemented in the kernel happen in the physical devices.

5 RELATED WORK

We describe the two lines of this work from which Drywall takes inspiration:

(1) The fuzzing technique [5], recently published on USENIX Sec23, based on a firewall protecting every DMA access to the PCIe devices through IOMMU and DMA controller will be filtered by the hook. And the author designed fuzzing techniques for real-world applications' access patterns to better hit the kernel's bugs.
(2) The ioctl[2] inside the Linux kernel for PCIe attached devices like FPGA etc. has a timing difference for side-channel attacks evaluation.

REFERENCES

[1] D. from Alibaba. Uce bugs, 2019.
[2] I. Giechaskiel, S. Tian, and J. Szefer. Cross-vm covert-and side-channel attacks in cloud fpgas. *ACM Transactions on Reconfigurable Technology and Systems*, 16(1): 1–29, 2022.
[3] Intel. Cxl specification, 2022.
[4] Intel. Tdx, 2023.
[5] S. Qin, F. Hu, B. Zhao, T. Yin, and C. Zhang. Kextfuzz: Macos iommu firewall inspection fuzzing. *USENIX Security 23*, 16(1):1–29, 2022.
[6] M. Tan, J. Wan, Z. Zhou, and Z. Li. Invisible probe: Timing attacks with pcie congestion side-channel. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 322–338. IEEE, 2021.