
OPEN SOURCE SUMMIT
China 2019

Preventing DMA Attacks from Untrusted Devices

Lu Baolu
OTC, Intel

Agenda

- Preface
- Device Classification
- Vulnerabilities 1: IOMMU default on
- Vulnerabilities 2: Fatal ATS
- Vulnerabilities 3: Bounce Page
- Vulnerabilities 4: Strict Mode
- Beyond IOMMU
- Call for action
- Take away
- Reference

Statement

- All vulnerabilities described in these slides are silicon vendor agnostic. Please don't think these are Intel-specific.
- Intel engineers are helping the community to address all these vulnerabilities. This will benefit all architectures.

官渡之战

A famous war won by weakness happened 1800 years ago in China.

The turning point was an adviser named 许攸 who betrayed 袁绍 and turn to 曹操.

Xu knows everything about Yuan and he turned the whole war around.

Information security is so so so important. We learned this lesson 1800 years ago.

DMA attack

Unfortunately, today, after 1800 years, we still face the problem of information security.

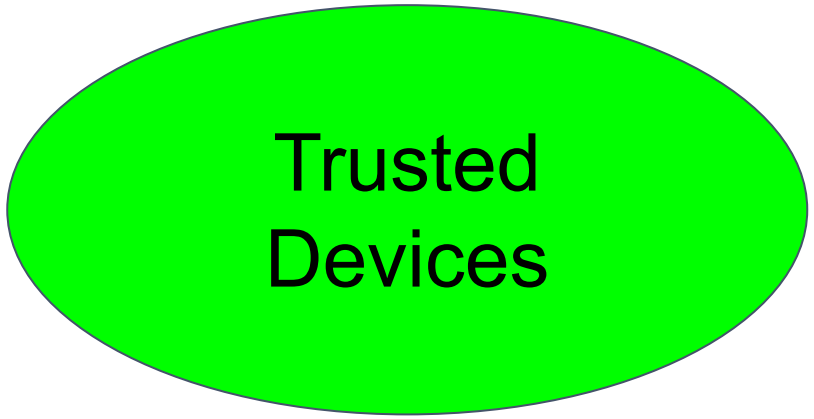
A malicious peripheral is able to attack and steal data from your computer if Direct Memory Access (DMA) is allowed. (Thunderbolt)

The silicon vendors provide IOMMU to isolate the DMA accesses from a device. But is that enough? No.

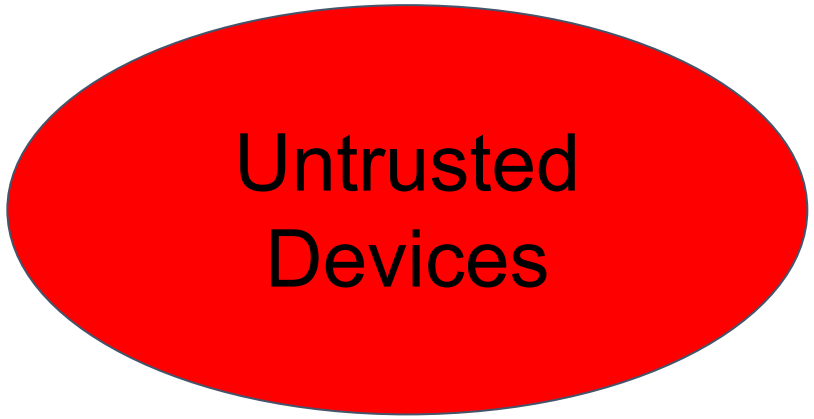
Before we start ...

- A trusted PCIe device always
  - DMAs into the designated buffers and not overrun.
  - never DMAs unless the driver asks it to do.
  - reports right error when DMA results in failure
- We don’t trust the external devices to the same degree as the devices built in the system.
- Security and performance are always a balance.
- Hence ...
  - For trusted devices, we focus on data transmit performance.
  - For untrusted devices, we focus on security.

Device Classification

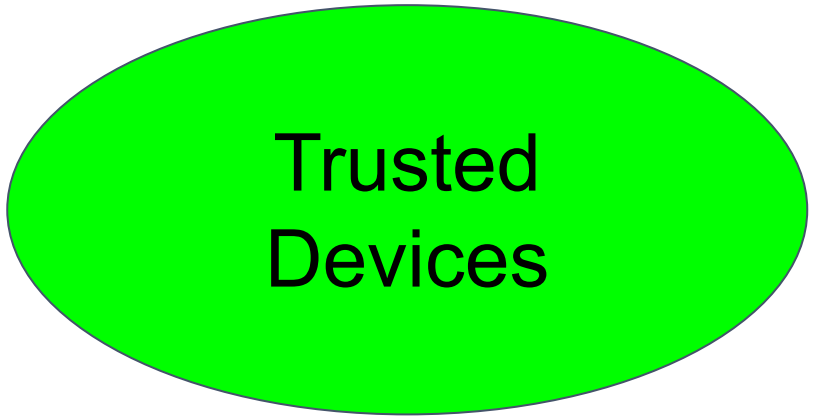


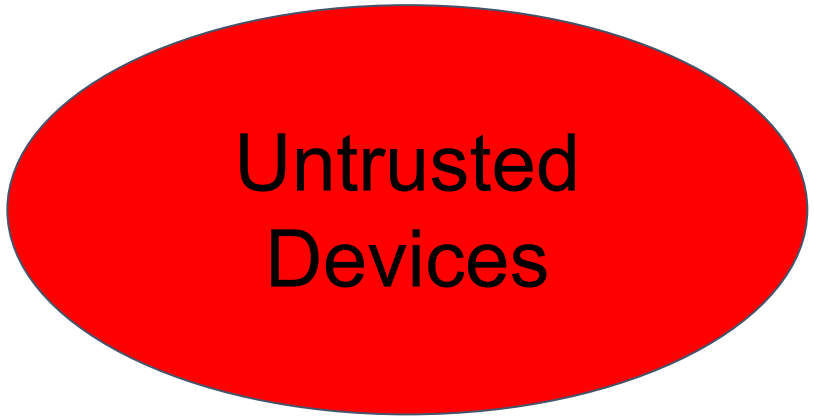
- Devices shipped together with the computer.
- Devices in chassis.
- Peripherals from trusted vendors
- … …



- Peripherals hot plugged through external port
  - Thunderbolt
  - PCI/PCIe slots
- … …

Device Classification





- Linux kernel is able to probe untrusted PCI device and mark it with flag in pci device structure.
  - https://lkml.org/lkml/2018/11/12/2325
  - Available since v5.0
- Linux kernel could have per-device sysfs switch so that the end user could classify the devices by their own.

V1: Is IOMMU on duty?

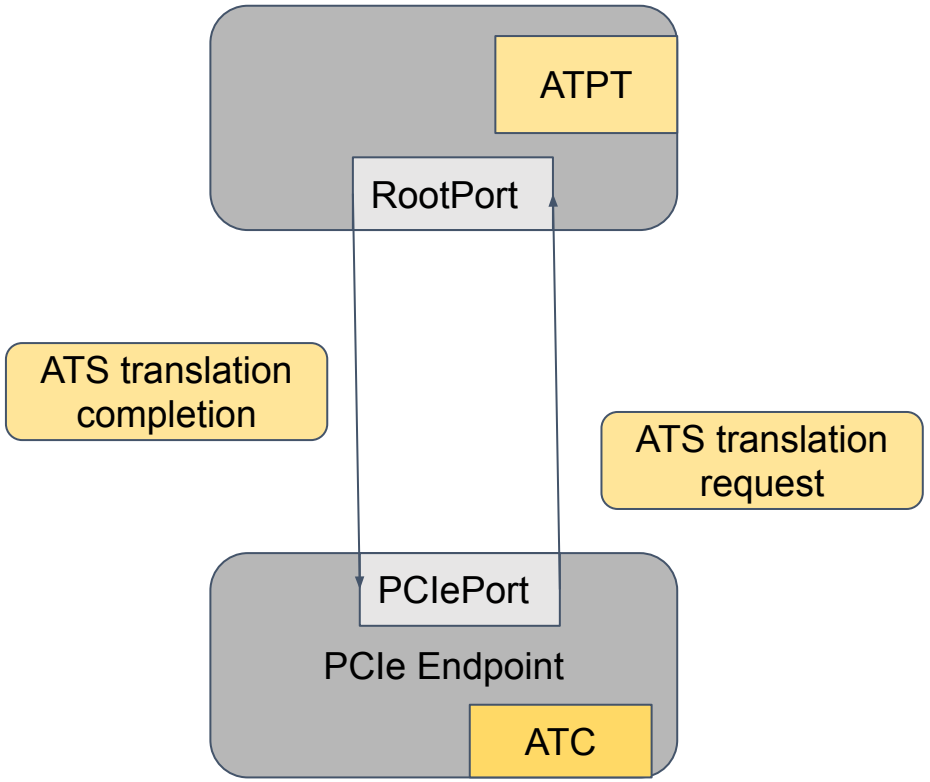
The silicon vendors provide IOMMU, but recent research[1] show that:

- macOS is the only system that turns on iommu protection by default
- Only the latest enterprise version of Windows provides any sort of protection, and the process to turn it on is very convoluted.
- FreeBSD 10.3, Ubuntu 14.04 and 16.04 and Red Hat Enterprise Linux 7.1, the iommu was switched off by default.

- Force IOMMU on if there are untrusted devices.
  - Linux kernel got fixed since v5.0
  - https://lkml.org/lkml/2018/11/12/2324

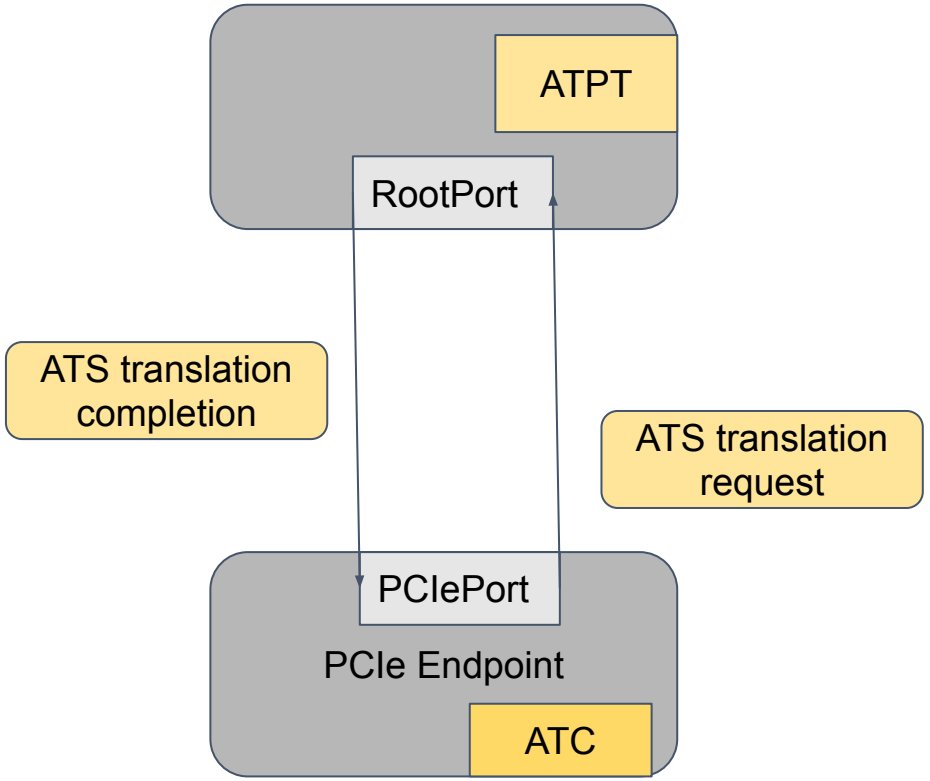
[1] https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-934.pdf

V2: Fatal ATS



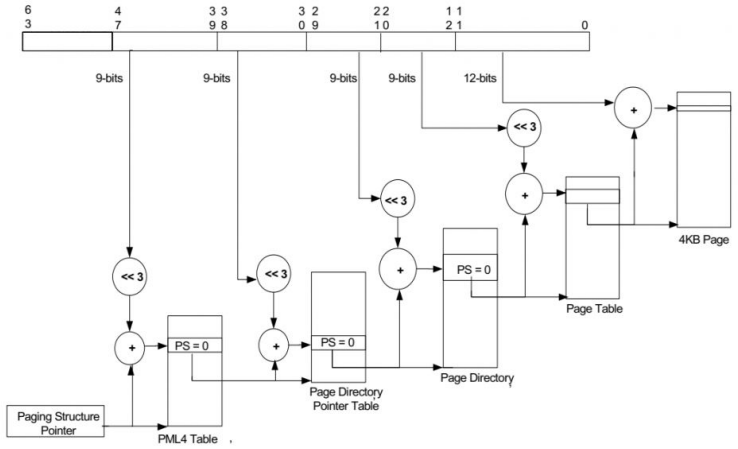
- ATS allows a PCIe endpoint to send an address translation request to IOMMU and cache the translation result in the local cache. IOMMU will be bypassed if the device uses the translated address for DMA.
- Linux blindly enabled ATS when it found the capability.
- A malicious device may claim the ATS capability and attack the system with faked translated addresses.

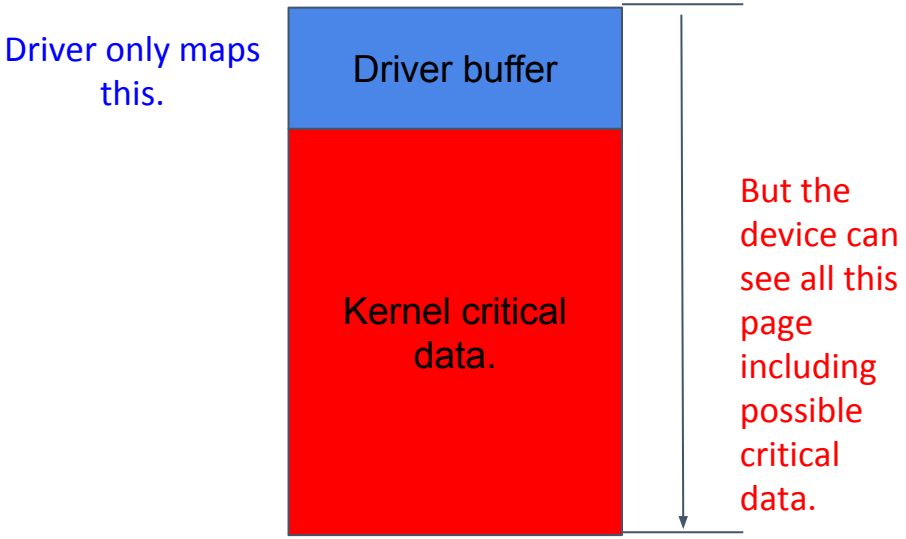
V2: Fatal ATS



- Short term solution
  - Disable ATS for untrusted devices
  - Linux kernel enhanced since v5.1
  - https://lkml.org/lkml/2019/2/28/1398
- Long term solution
  - Secure ATS support
  - PCIe ECN: PCIe* Device Security Enhancements
  - https://www.intel.com/content/www/us/en/io/pci-express/pcie-device-security-enhancements-spec.html

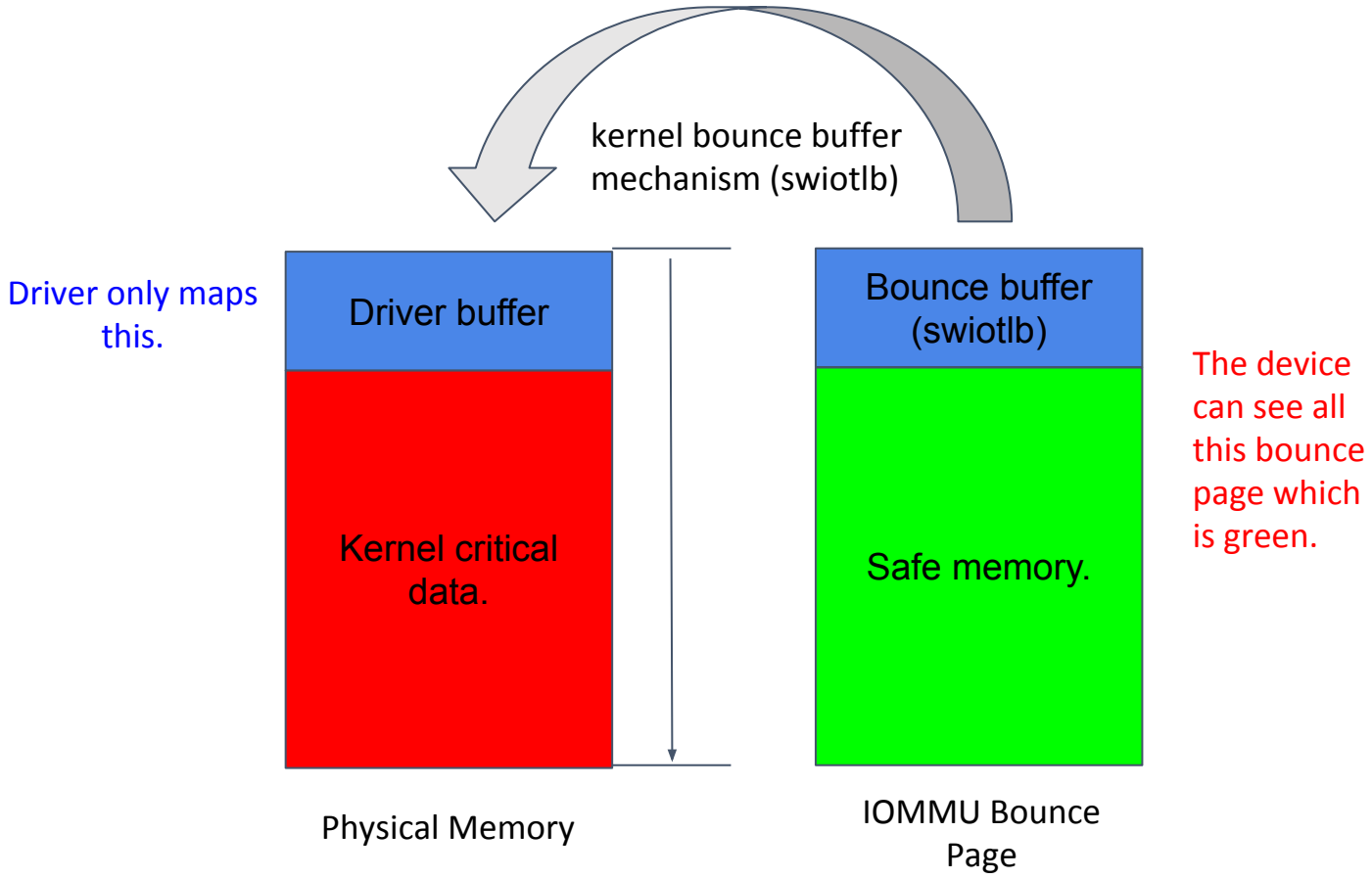
V3: Bounce Page





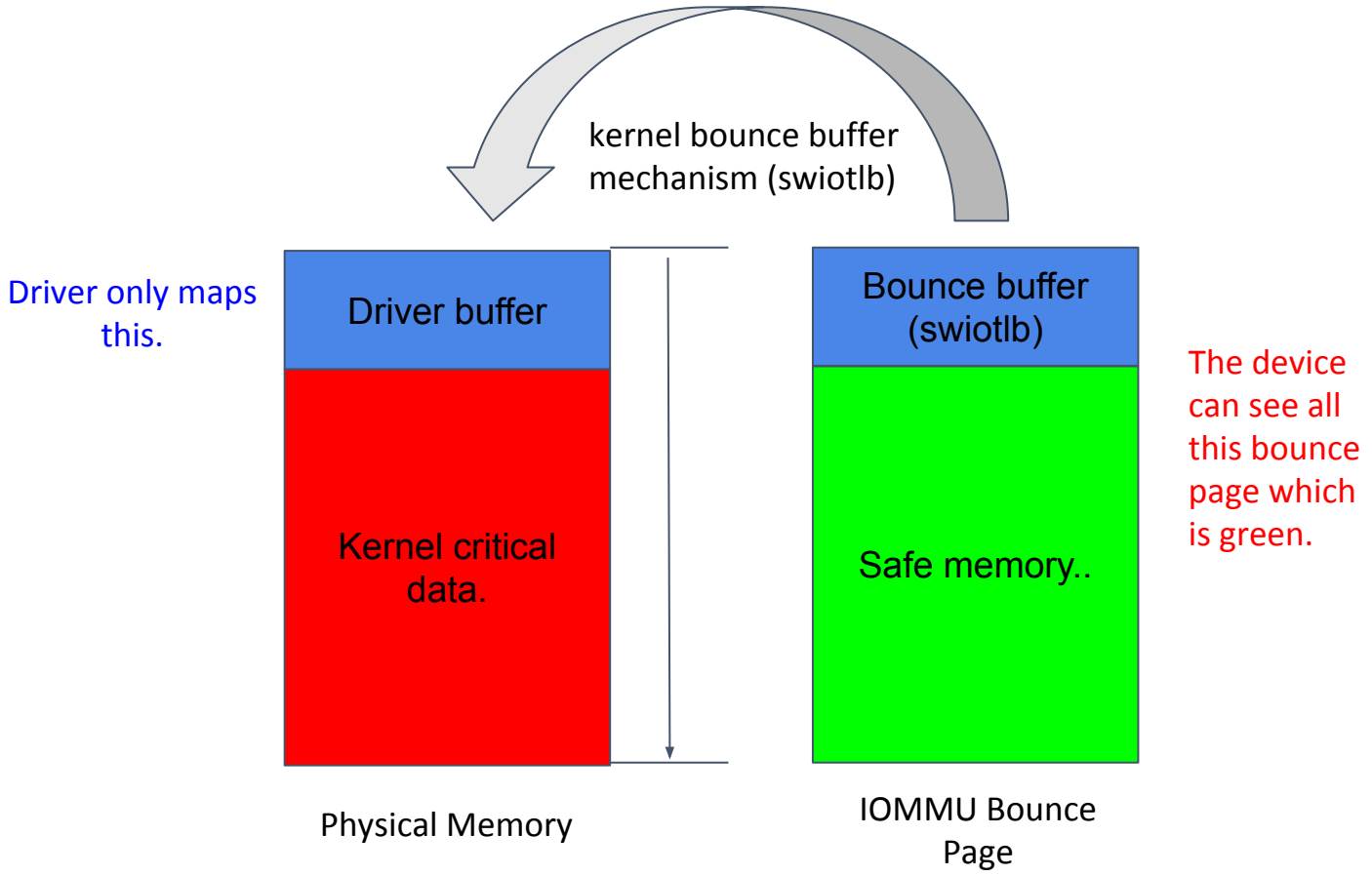
- Minimum IOMMU window size is one page (4k)
- Drivers may map buffers not filling the whole IOMMU window
- This allows device access to possibly unrelated memory
- Malicious device can exploit this to perform a DMA attack

V3: Bounce Page



- IOMMU driver allocates full page size bounce buffer
- Data is copied (bounced) between bounce and physical buffer
- IOMMU window is opened to this bounce buffer instead
- The device cannot access memory outside of the IOMMU window

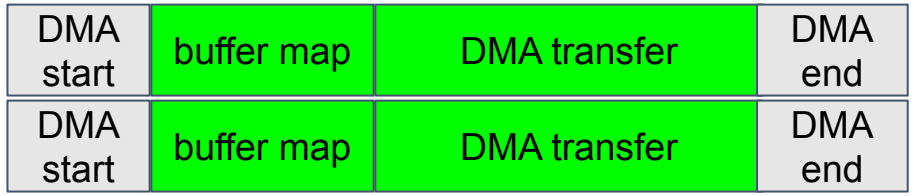
V3: Bounce Page



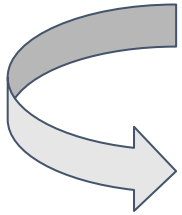
- The latest bounce page implementation was posted here for discussion https://lkml.org/lkml/2019/6/2/187
- Enhance IOMMU to provide sub-page protection
  - Hardware cost for sub-page protection is high.
  - ROI needs to evaluate relative to software based solution.

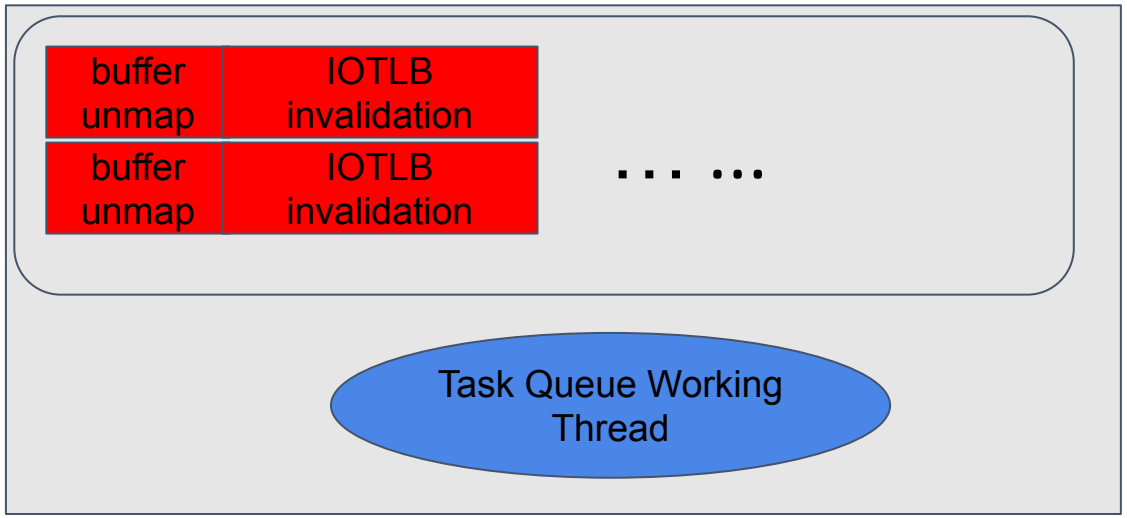
V4: Strict Mode

DMA sequence in IOMMU strict mode



DMA sequence in IOMMU lazy mode





V4: Strict Mode

- Lazy mode by default
  - IOMMU driver does not tear down the mapping when driver unmaps the buffer.
  - Instead it will schedule a thread that does it later on with the idea of tearing down multiple mappings at once.
  - This is due to performance reasons.
- During this time period the device can still access the memory even if it is not supposed to do so.
  - Malicious device can exploit this for DMA attack.
- Use strict mode for untrusted device
  - Linux kernel enhanced since v5.2
  - https://lkml.org/lkml/2019/4/12/13

Beyond IOMMU(抛砖引玉)

- Attack window between boot and IOMMU enabling.
  - Use VT-d PMR (Protect Memory Region) range registers to guard platform memory during the window.
  - Available since kernel v2.6.25.
- Most drivers that encounter an unrecoverable error cause kernel panic bringing down the platform and resulting in DOS attack.
  - Identify and enhance device drivers
- System software ignores malicious activity reported by IOMMU.
  - After N malicious activity report by IOMMU system software should disable DMA from offending device.
- Software identifies devices using manufacture and device-id that are publicly available and can be spoofed.
  - Cryptographic handshake to attest device.
  - PCIe* ECN under discussion in PCISIG
  - https://www.intel.com/content/www/us/en/io/pci-express/pcie-device-security-enhancements-spec.html

Call for action

When developing a device driver, please

- Avoid using panic():
  panic("%s: Illegal queue state.", dev->name);
- Avoid using BUG_ON(con), use below instead
  if (WARN_ON(con))
  return;
- Try to use DMA buffers which are driver specific and page aligned.
- Specify correct DMA direction and attribution parameters when map a DMA buffer.

Take Away

- Both silicon and platform provide technologies to protect against DMA attacks.
- OS relaxed the control of DMA in order to transmit data more effectively.
- We are enhancing Linux kernel by
  - Classify devices into different trust levels;
  - For trusted devices, keep optimizations for efficient data transfer;
  - For untrusted devices, fix the security vulnerabilities against DMA attack;
  - Investigating enhancements in PCIe* and IOMMU implementation for secure and efficient data transfer.
    - Secure ATS support
    - IOMMU sub-page protection

Reference

1. https://christian.kellner.me/2019/02/27/thunderclap-and-linux/
2. https://thunderclap.io/
3. https://www.repository.cam.ac.uk/handle/1810/274352
4. https://lwn.net/Articles/786558/
5. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-934.pdf

Suggestions? Questions? Comments? Please!