Thunderclap: Exploring Vulnerabilities in Operating System IOMMU Protection via DMA from Untrustworthy Peripherals

Robert N. M. Watson

R209 - 9 October 2025

A bit of metadata

- This paper was originally published at the Network and Distributed Systems Security (NDSS) Symposium 2019.
- The work took place over several years, including multi-year interactions with vendors to **update threat models** and **remediate multiple vulnerabilities**.

- Ultimately the results of this work not only included several important security updates and product design changes, but also changes to the USB 4 specification.
- There has been considerable work building on this, both around I/O security and also for device-driver validation through fuzzing.

- I started with the conference slide deck “as presented” but trimmed/rearranged/edited some slides to give a more focused presentation.
- “Do as we say and not as we do.” (Sorry)

THUNDERCLAP

The Perils of Peripherals

A. Theodore Markettos[†], Colin Rothwell[†], Brett F. Gutstein[†*],

Allison Pearce[†], Peter G. Neumann[‡], Simon W. Moore[†], Robert N. M. Watson[†]

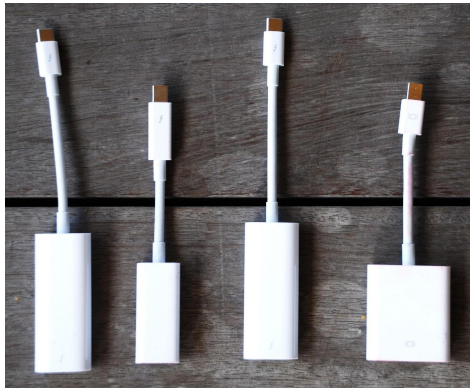
[†]University of Cambridge
Dept. Computer Science and Technology

[‡]SRI International

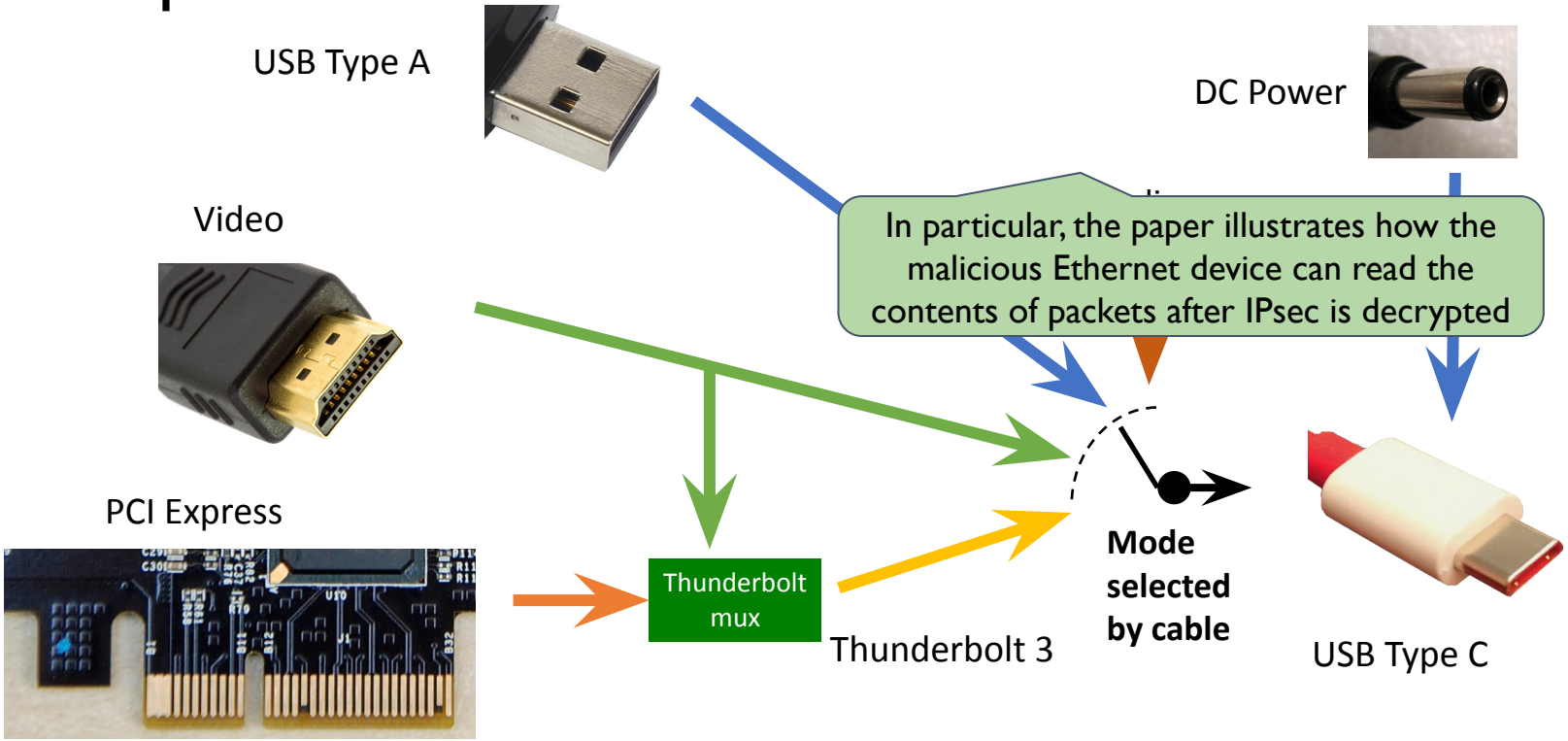
[*]Rice University

Smaller laptops, more external peripherals

- Laptops getting smaller, more devices are going external
  - Chargers, dongles, docking stations
  - Common to borrow external peripherals (power, dongles, displays) from others
- Performance is increasingly more of a constraint
- Security?



USB-C convergence: fewer plugs is great, but now we can't tell protocol from the connector



Security?

- USB is a packet-based protocol
  - like the internet, only little scrutiny
  - attackers craft bad messages
  - reprogram devices to send bad messages
  - trip up and exploit device drivers
  - defences: firewalls, filtering, fuzzing etc
- Thunderbolt carries PCI Express, which is a memory-based protocol
  - DMA: *direct memory access*
  - access the full state of your machine
  - read your files, your passwords
  - inject arbitrary code...

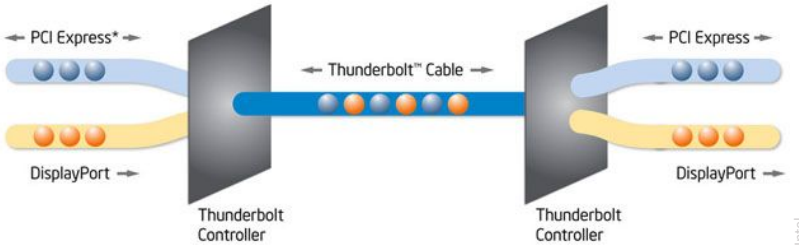
ars TECHNICA

BIZ & IT —

This thumbdrive hacks computers. "BadUSB" exploit makes devices turn "evil"

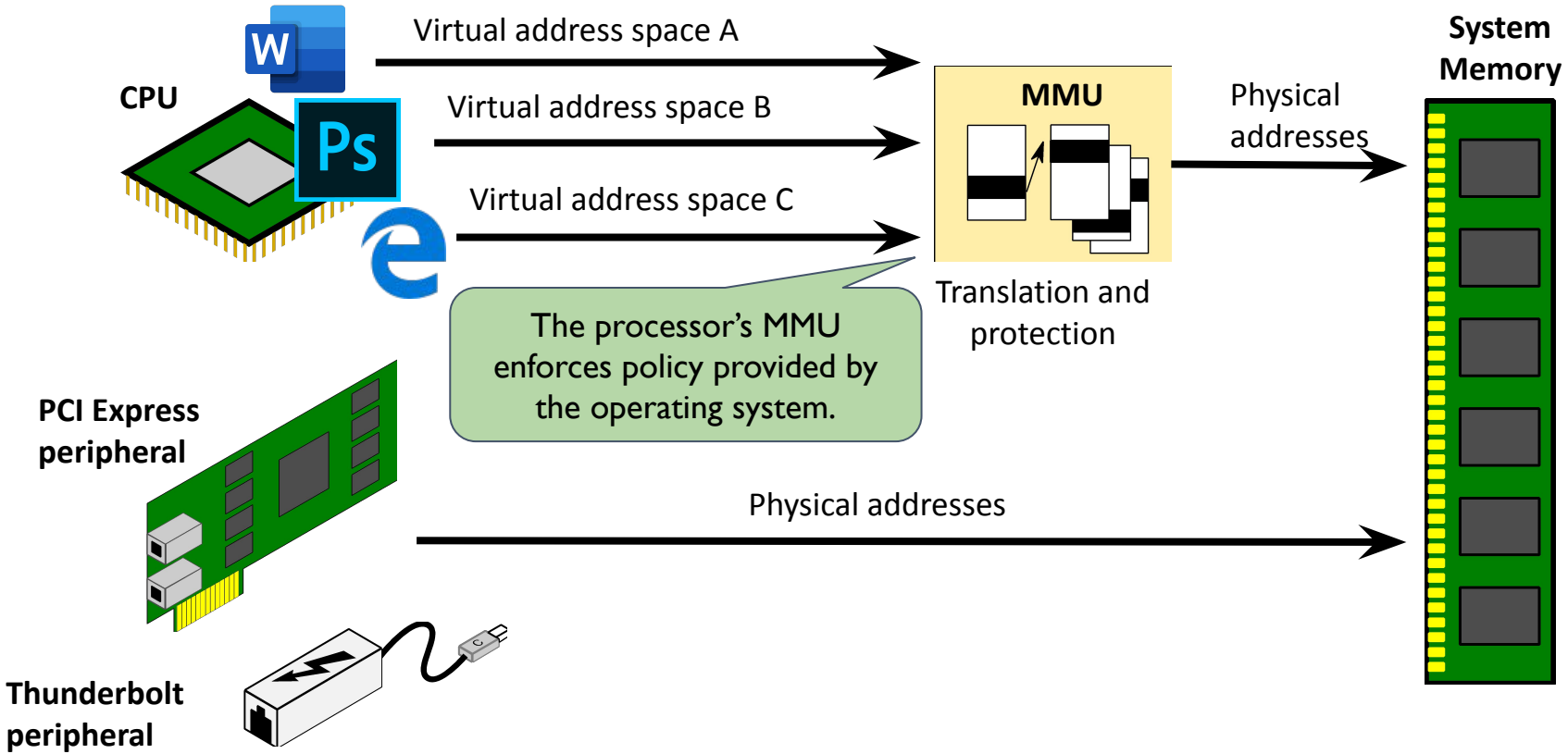
Researchers devise stealthy attack that reprograms USB device firmware.

DAN GOODIN - 7/31/2014, 2:21 PM

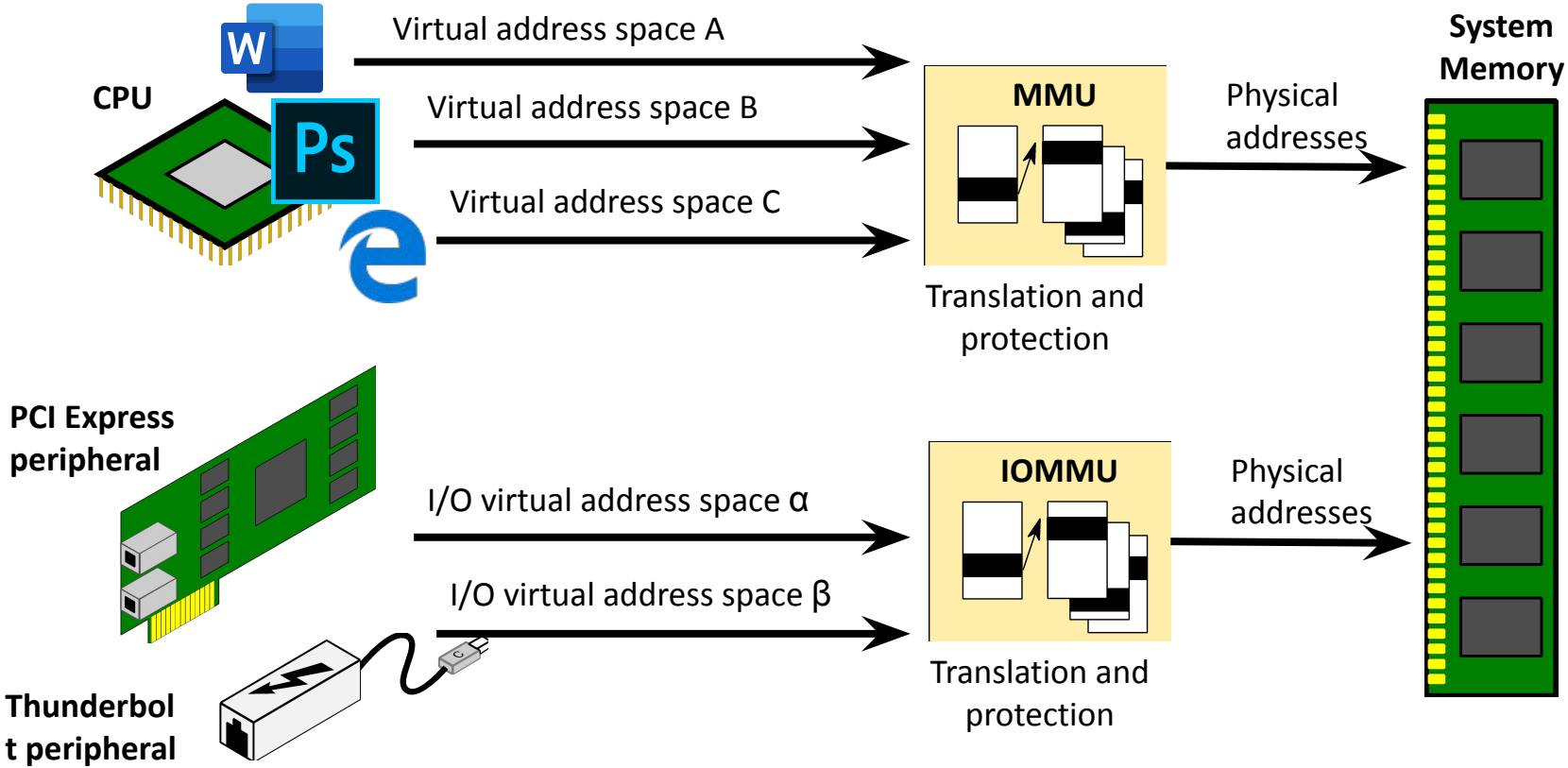


- USB Type C carries both, and power and video, on the same cable

Memory Management Unit (MMU): process isolation



I/O Memory Management Unit (IOMMU): device isolation



IOMMU protection against malicious devices

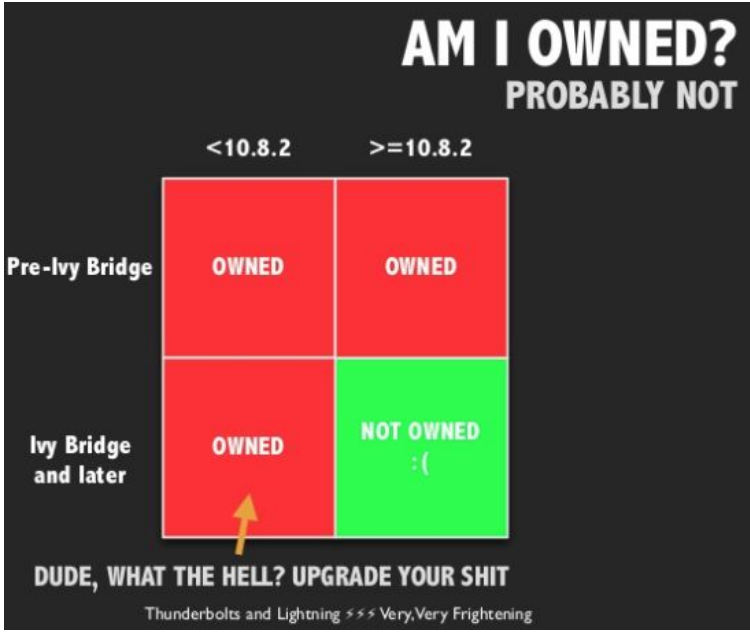
- ✗ Windows 7 / 8 : don't use the IOMMU, all memory exposed
- ✗ Windows 10 Home/Pro : didn't use the IOMMU
- ✔ MacOS ≥10.8.2 : IOMMU enabled by default
- ✗ Linux : supported, but IOMMU rarely enabled by default
- ✗ FreeBSD : supported, but not enabled by default
- ✗ IOMMU often disabled in default firmware settings (BIOS, UEFI)

Current state of the world is not good

Our work assumes that the OS vendor is at least vaguely trying...
What is the attack surface if they turned on IOMMU protection?

Attacks from a real device

- general understanding: "when the IOMMU is enabled, attacks are foiled"
  - these are simple memory-probing attacks
  - no interactions with driver or kernel
- actually, the attack surface is much more nuanced
- what attack surface does a real I/O device have?
  - what accesses can it make?
  - how does it interact with the device driver stack?
  - as the OS increasingly trusts it, what extra vulnerabilities does it open up?

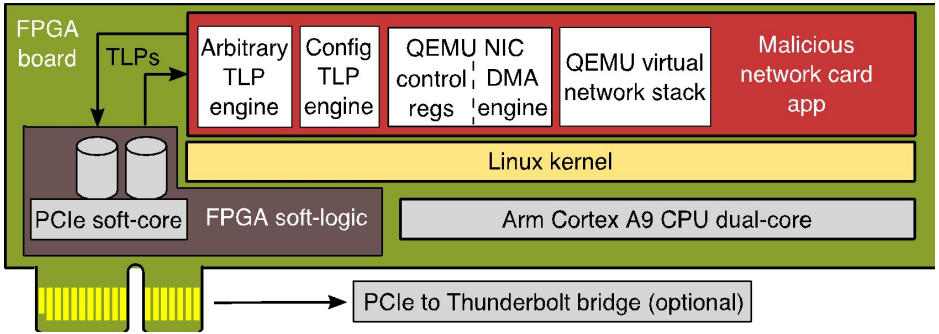


snare and rzn, *Thunderbolts and Lightning – Very Very Frightening* (2014)

This isn't so much about attacking the IOMMU itself, but how OS **uses** the IOMMU

Thunderclap: a research platform for I/O security

- We built a fake network card (NIC):
  - software device model of an Intel E1000 PCIe ethernet card from QEMU
    - software = easy to change, add malicious behavior
  - run it on a CPU on an FPGA (Arm Cortex A9 on Intel Arria 10, running Ubuntu)
    - FPGA logic can send and receive arbitrary PCIe packets
    - QEMU model responds to PCIe packets and generates 'DMA' like a real NIC
  - runs on FPGA dev boards, attached via PCIe or Thunderbolt dock
  - hardware/software open sourced
  - designed physical embodiments
    - Thunderbolt dock implant
    - malicious projector, charger
    - not fully engineered/productized
    - not released at this time



These are all in (important) "narrative elements": Once we have the structure of the attack, how might it be deployed?

Attack: MacOS data leakage and root shell

- MacOS architecture
  - all devices share one page map
    - network card can't read/write kernel or apps memory, but can access USB buffers, framebuffer
  - mbufs are allocated in a single block and exposed to all devices at boot time
    - access all of the network data all of the time – traffic for other network cards/wifi, VPN plaintext, etc
- Breaking existing protections
  - Kernel-Address Space Layout Randomization (KASLR) can be broken due to leaked symbol from USB driver
  - free() function pointer and 3 parameters from mbuf allow launching a root shell

```
struct mbuf {
  ...
  struct m_ext;
  ...
  // internal buffer
  char M_databuf[224];
};

struct m_ext {
  // external buffer pointer
  caddr_t ext_buf;
  // free() function pointer
  void (*ext_free)(caddr_t,
      u_int, caddr_t);
  u_int   ext_size;
  ...
  struct ext_ref {
    u_int32_t refcnt;
    // buffer is external flag
    u_int32_t flags;
  } *ext_refflags;
};
```

This is an example of an "exploit chain," a concept we will return to over the term

Attack variants: FreeBSD and Linux

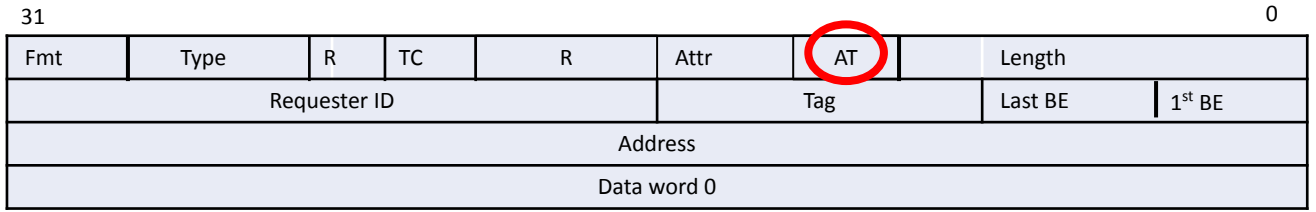
- FreeBSD
  - one page map per device
  - see other network traffic co-located on pages (traffic for other NICs, VPN plaintext)
  - no KASLR: root shell attack works

In particular, the paper illustrates how the malicious Ethernet device can read the contents of packets after IPsec is decrypted

- Linux
  - one page map per device
  - data and metadata on different pages – can't overwrite free() pointer
  - general kernel allocator used by driver
    - see Unix domain socket traffic (as used by SSH agent)
    - kernel NAT jump tables, potentially lots more...

Attack: Linux IOMMU bypass

- PCIe has a feature called Address Translation Services (ATS)
- Allows PCIe to carry pre-translated addresses
  - Performance mitigation to cache translations locally, don't have to go inter-socket on a multi-socket server
- 'Pre-translated addresses' means we can generate memory reads/writes to arbitrary physical addresses with no IOMMU interposing
- Set Thunderclap to advertise PCIe configuration registers saying it supports ATS
- Linux sees this and enables ATS on the PCIe switches
- Set a bit in the PCIe packet header saying an address is pre-translated
- We've completely bypassed IOMMU protection!



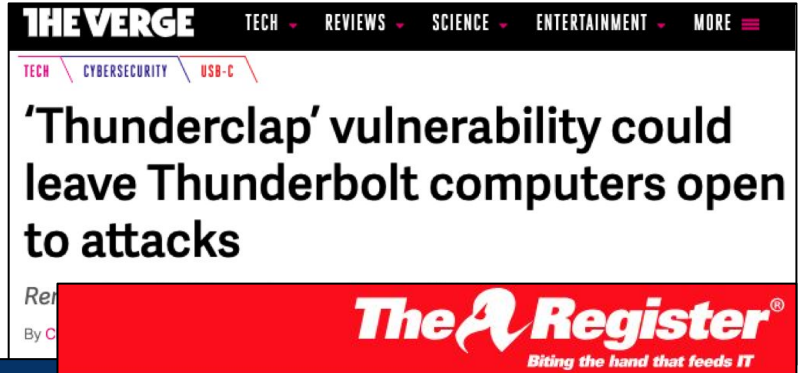
MemoryWrite32 TLP

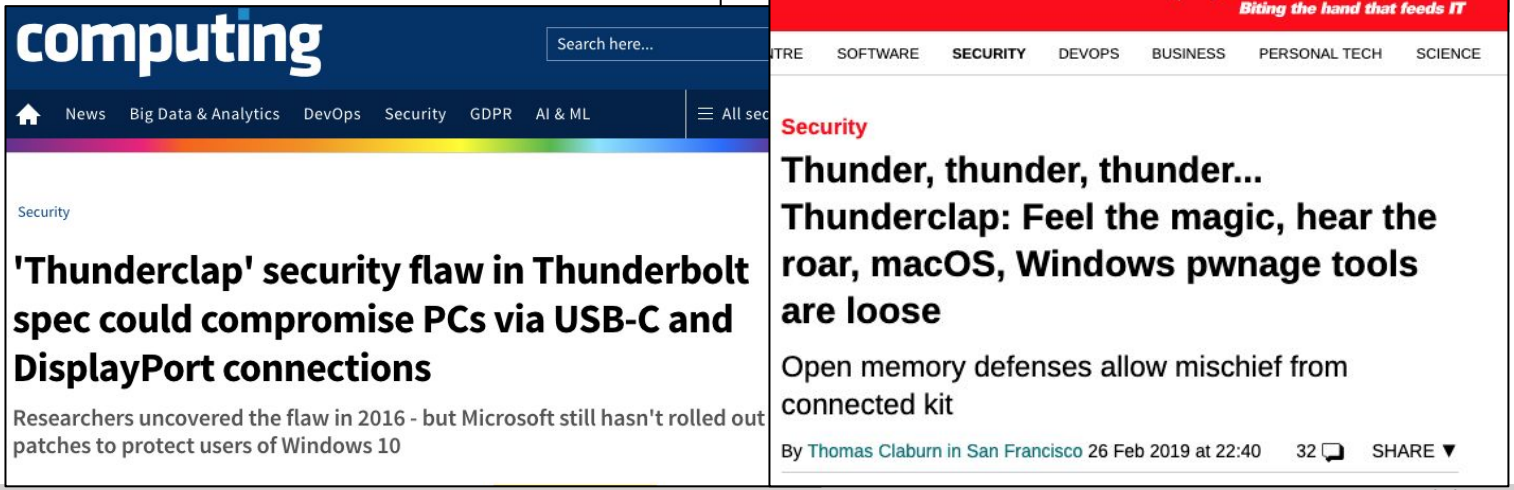
Mitigations and impact

- Collaborating with vendors since 2016
- Apple mitigated specific exploit in MacOS 10.12.4
  - encrypt the kernel pointer, hide the flags
- Microsoft shipped Kernel DMA Protection for Thunderbolt 3 in Windows 10 1803
  - IOMMU enabled for Thunderbolt devices (only)
  - Requires post-1803 firmware, ie new products only
- Intel enabled IOMMU for Thunderbolt in Linux 4.21 (now 5.0rc), disabled ATS
  - Thunderbolt devices are now less trusted than internal ones
- Major laptop vendor: we won't ship Thunderbolt until we understand this attack vector better
- Eternal vigilance: DMA turning up in numerous new places – PCIe in phones, SD card 7.0, NVMe over Ethernet...

Coverage in the popular press

- A new attack vector
- Defences aren't up to scratch
- What can we do about it?
- What lessons can we learn?





The Register: We became boffins .. their highest praise!

"The aforementioned research platform, dubbed Thunderclap, and the associated paper represent the work of **assorted academic and think tank boffins**: …"

Mitigations and impact

- Best practice guidelines
- Engaging with the future

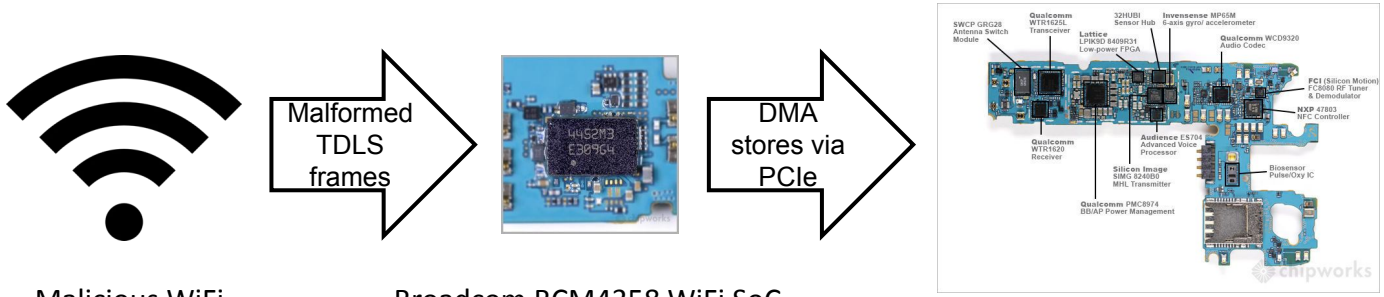
Conclusions

- We present the IOMMU attack surface as a new and rich field for vulnerabilities
- Open sourced Thunderclap, a research platform that allows exploration from an FPGA
- Told some stories of attacks across four major OS platforms
  - including a complete IOMMU bypass
- Vendors shipped mitigations to our attacks which are already fielded
- Solving the problem in the general case is a lot harder than it appears... we're working on it!
- NDSS paper, source code and FAQ: **thunderclap.io**

THUNDERCLAP

Example 1: WiFi microcontroller attacks on mobile devices (Google Project Zero 2017)

Photos: chipworks



Malicious WiFi Access Point

Broadcom BCM4358 WiFi SoC running Broadcom firmware

Snapdragon 801 (BB/AP SoC) running Android

- Gal Beniamini (Google Project Zero) research into WiFi-based attacks
- Compromised Android and iOS devices via vulnerable Broadcom Wifi SoC
  - Malformed 802.11 packets triggers **classical buffer overflow in firmware**
  - Escalation: heap corruption→ **arbitrary code execution on SoC microcontroller**
- Escalate via DMA over PCIe to obtain privilege on application processor
  - **IOMMU unused** by Android operating system on many phones
  - IOMMU used in iOS; **exploited descriptor-ring race condition** w/device driver
- Cross-SoC attack exploits vulnerable I/O core to attack application core

Possible discussion questions

- Why is the comparison made by the authors between the system-call interface and the I/O interface so apt?
- Why was a tangible demonstration of these techniques so important to seeing these issues addressed?
- What caused Microsoft to take several years to change its stance on whether DMA attacks were “in scope”?
- Why is the ATS bit a “good idea” in some contexts vs. a “terrible idea” in others?
- What are the ethical considerations in releasing Thunderclap as open-source hardware and software?
- Does good use of an IOMMU solve this general class of problems involving malicious peripherals?