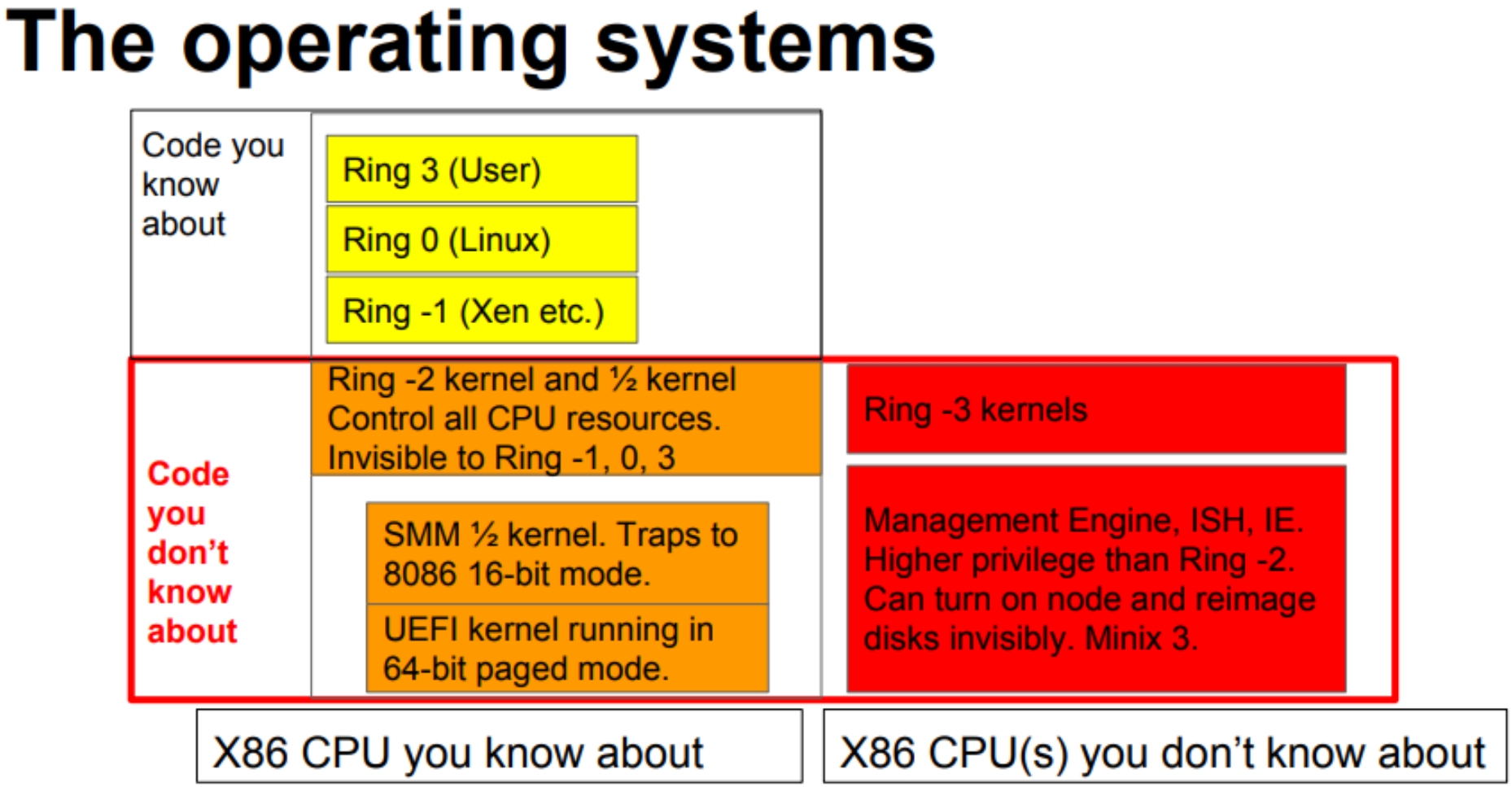
Hardware Exploitation

Ring -1, -2, -3

Exploiting Ring <0

“Replace Your Exploit-Ridden Firmware with Linux - Ronald Minnich, Google”

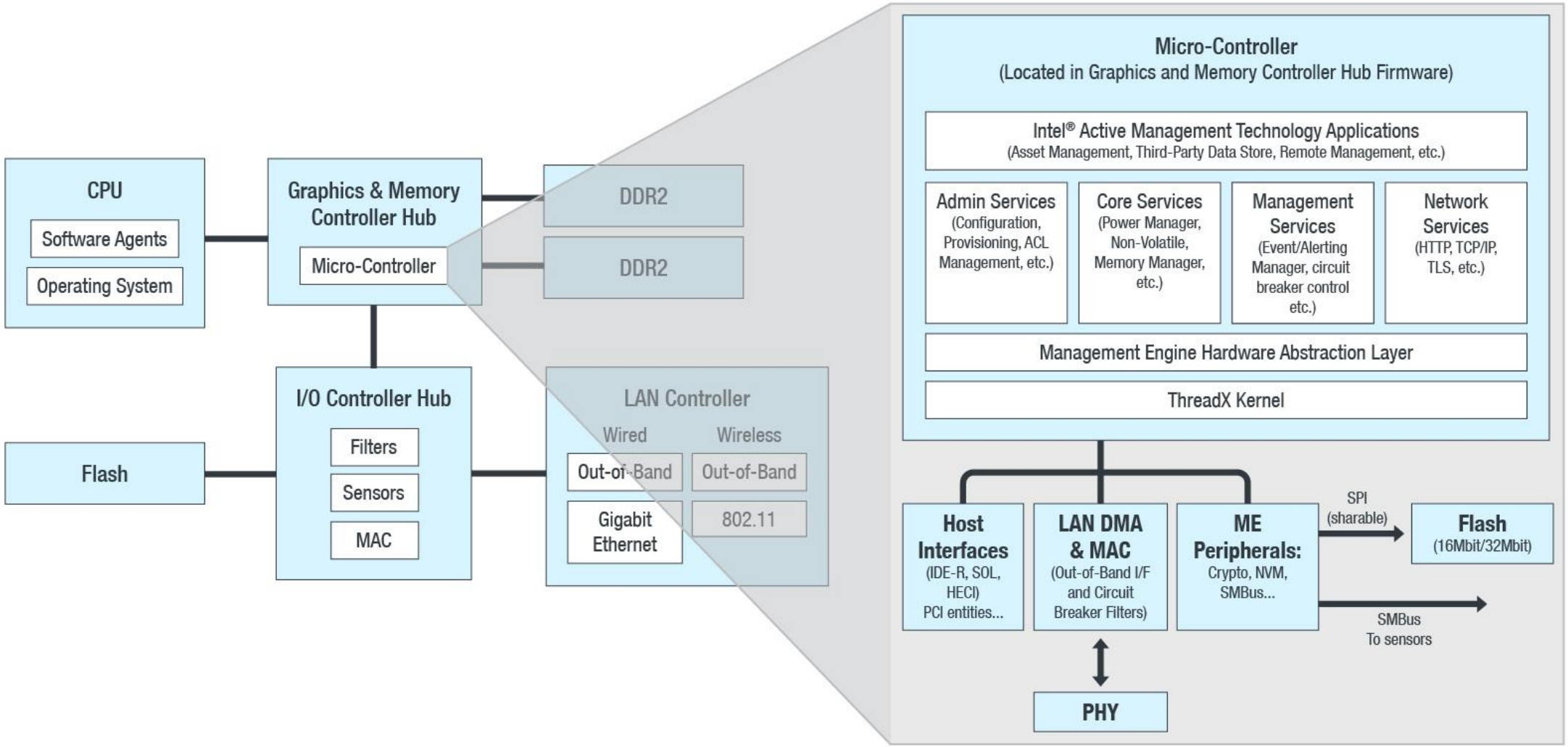


Ring -1, -2, -3

- Ring -1: Hypervisor
  - ESX, HyperV, Xen etc
  - Makes it possible to run multiple kernels (VM's) at the same time
- Ring -2: SMM
  - System Management Mode
  - 16 bit mode
  - Handling of interrupts
- Ring -3: Intel ME
  - Management Engine
  - Separate Microprocessor (!)
  - Works if your computer is off
  - Has TCP/IP stack
  - Minix OS
  - Access to Screen (KVM)
  - Intel AMT

Intel ME

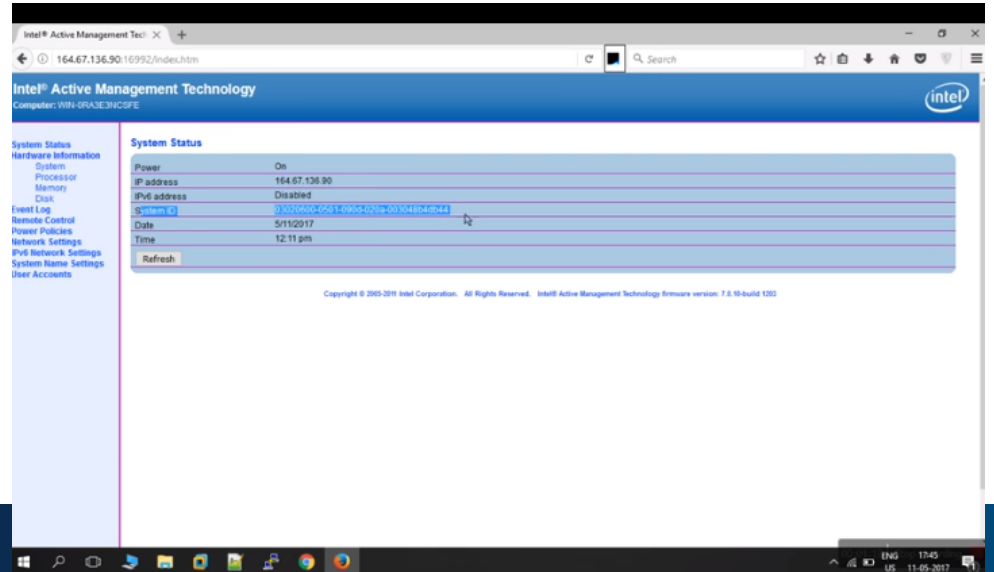
https://blog.trendmicro.com/trendlabs-security-intelligence/mitigating-cve-2017-5689-intel-management-engine-vulnerability/



Intel ME / AMT Bug

Using a web-based control panel, accessible from **port 16992 and 16993,** which comes pre-installed on the chipset, an administrator can remotely manage a system.
The **Intel AMT Web Interface works even when the system is turned off,** as long as the platform is connected to a line power and a network cable, as it operates independently of the operating system.

To exploit this logical flaw in Intel AMT Web Interface, **all an unauthorized attacker needs to do is send nothing (null)** into user_response to the server.

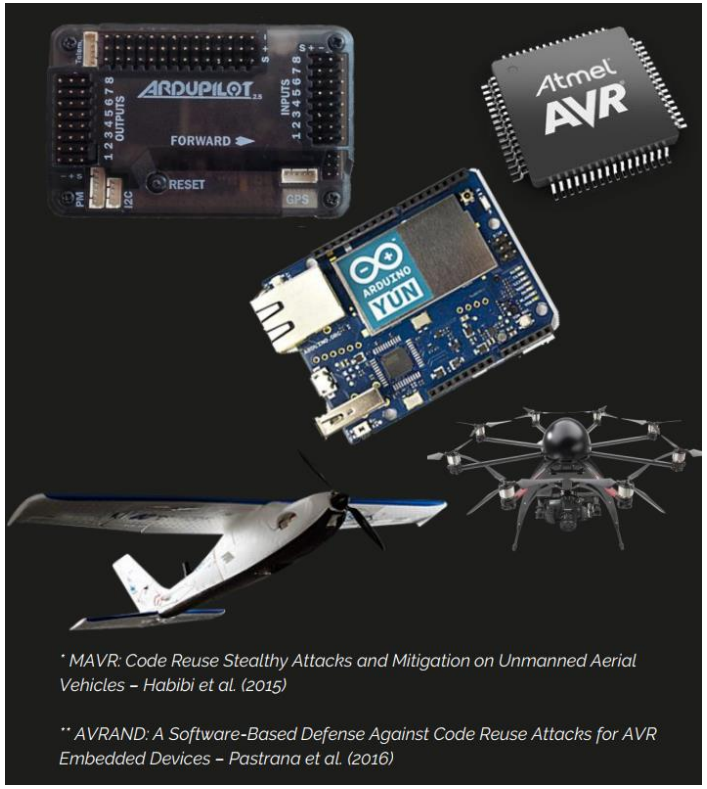


Embedded Systems

Non Intel/AMD/ARM

Embedded Systems

- Non Linux/Windows Systems
- ESP8266, ROTS, ESP32, Zephyr, Fuchsia OS (Google), …
- Reference:
  - Jos Wetzels
  - http://samvartaka.github.io/work

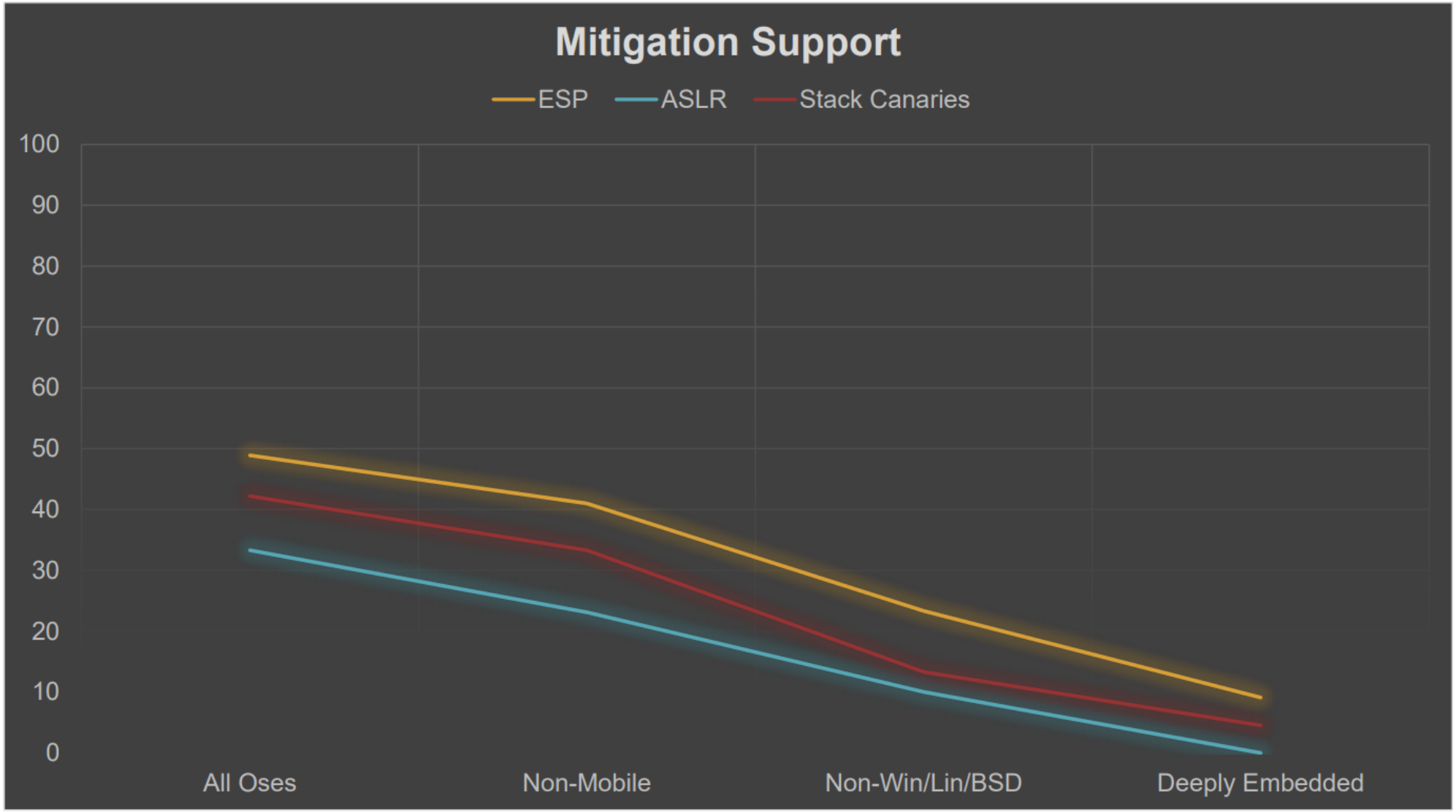


Hardware Selection

- Selected 78 Popular Embedded '*Core Families*'
- Evaluated for Hardware Dependency Support

https://hardwear.io/document/rtos-exploit-mitigation-blues-hardwear-io.pdf

Embedded Systems



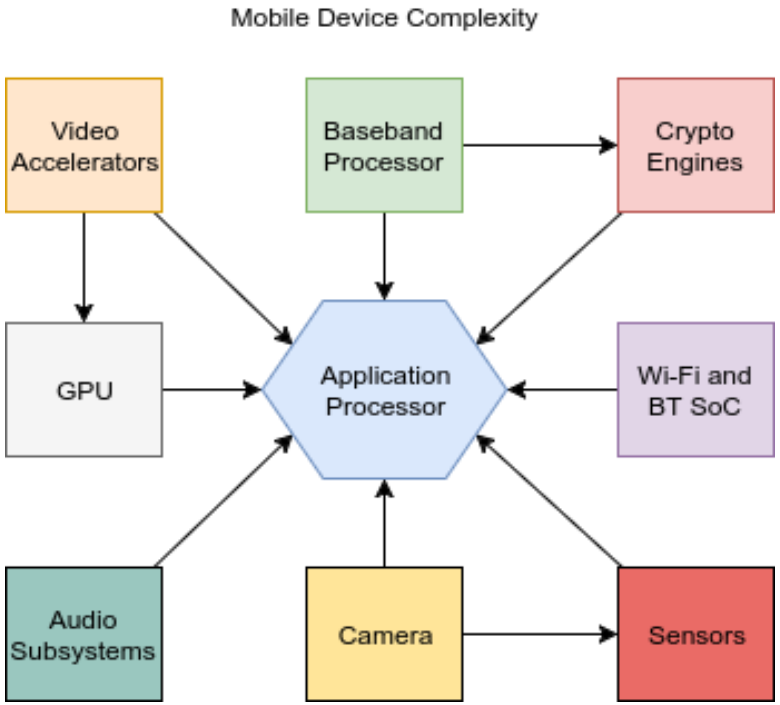
https://hardwear.io/document/rtos-exploit-mitigation-blues-hardwear-io.pdf

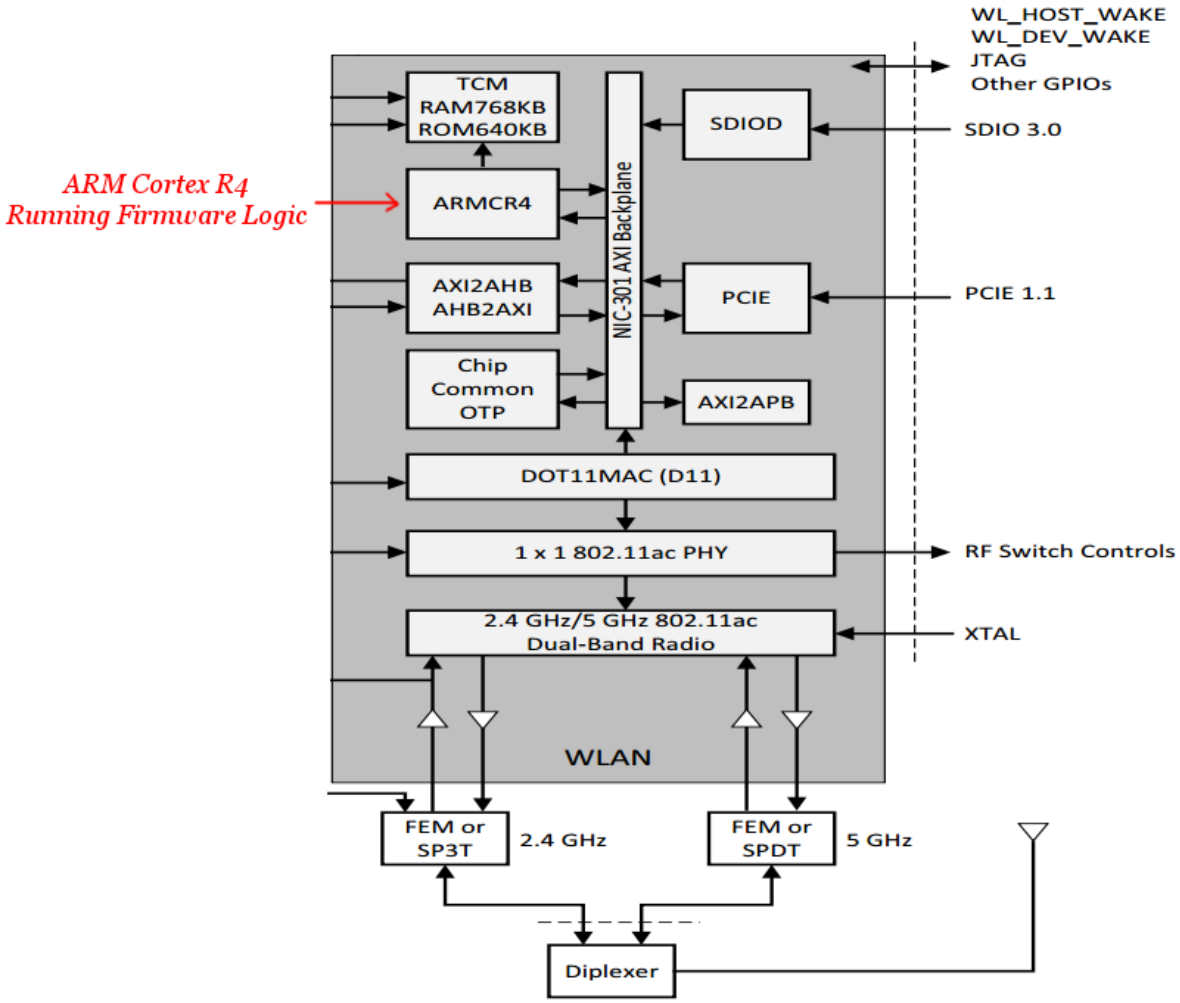
Embedded Systems

- Have their own CPU implementation, instruction set
- May or may not have Exploit Mitigation
  - DEP (CPU+OS)
  - ASLR (OS)
  - Stack Canaries (compiler)
- May or may not enable them by default
- May or may not have to create your own ROP technique (JOP, SOP,…)
- Have to create your own shellcode

Exploiting Broadcom's Wi-Fi Stack

https://googleprojectzero.blogspot.ch/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html





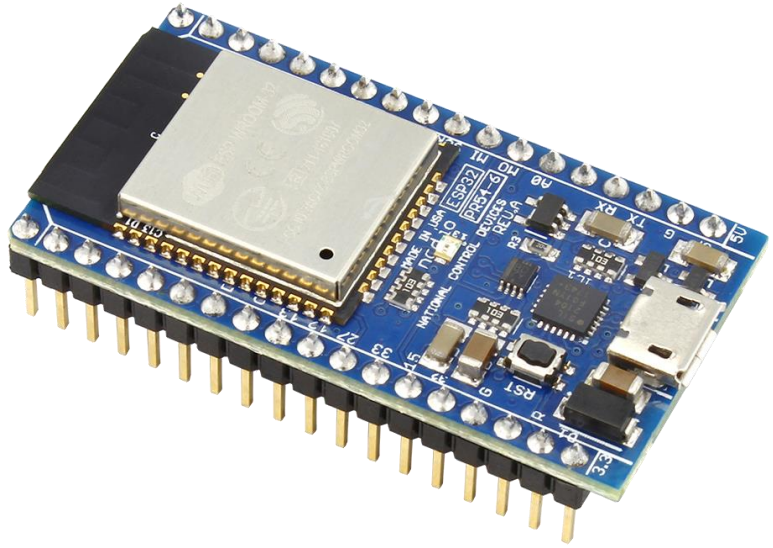
Exploiting Broadcom's Wi-Fi Stack

https://googleprojectzero.blogspot.ch/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html

- Exploit for WiFi Chip Broadcom (Nexus 5, 6, 6P, Samsung, iPhones, …)
- Via WiFi frames
- Heap overflow (heap massage)
- After RCE: Escalate to Host OS

We've seen that while the firmware implementation on the Wi-Fi SoC is incredibly complex, it still lags behind in terms of security. Specifically, **it lacks all basic exploit mitigations – including stack cookies, safe unlinking and access permission protection**

Microcontrollers



ESP8266, ESP32

- No real OS (no processes, filesystem)
  - One single process. TCP/IP, drivers etc. as library
- Wifi, bluetooth, ...
- Stack is (by hardware) not executable

Gadgets: real or fake

What is perfect and what is fake exactly?

load_regs:

```
l32i.n a12, a1, 4;
l32i.n a13, a1, 8;
l32i.n a14, a1, 0xc;
l32i.n a15, a1, 0x10;
l32i.n a0,  a1, 0;
addi   a1,  a1, 0x20;
ret.n
```

Load data from stack into register
A12 = *(A1 + 4)
…
A0 = *(A1 + 0)
A1 = *(A1 + 32)

write_mem:

```
s32i.n a14, a15, 0x0;
l32i   a0,  a1,  0;
addi   a1,  a1,  0x4;
ret.n
```

Store the data in register a14
Into address stored in 15

*(A15+0) = a14
A0 = *(A1 + 0)
A1 = *(A1 + 4)

sync:

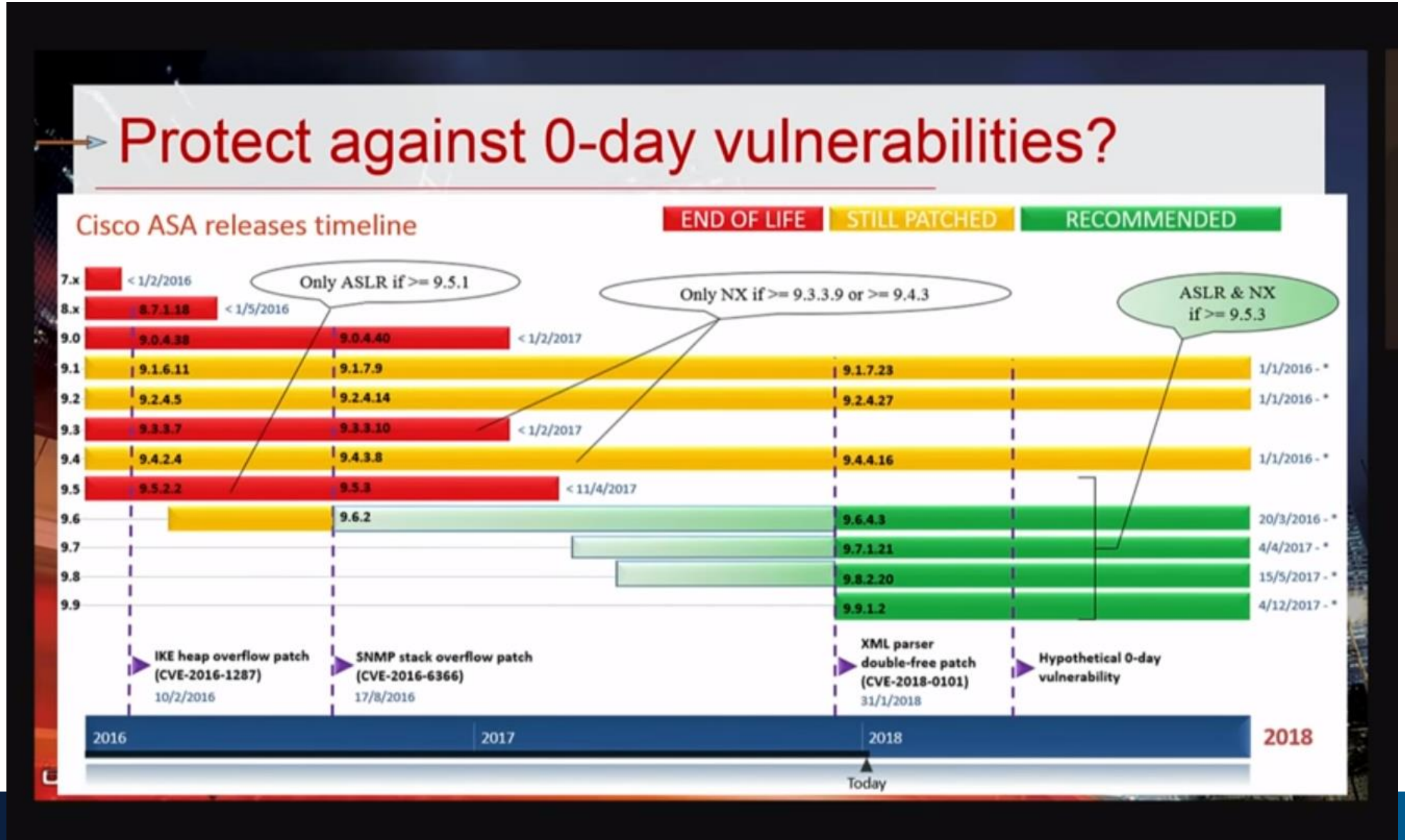
```
isync;
isync;
l32i a0, a1, 0;
addi a1, a1, 0x4;
```

Sync (flush caches etc)
So new code is visible

- https://def.camp/wp-content/uploads/dc2017/Day%201_Carel%20&%20Philip_xtensa_exploitation_DRAFT.PDF

What about ASA (Cisco)

https://www.youtube.com/watch?v=eDyxBgIUaR8



Attacking Hardware

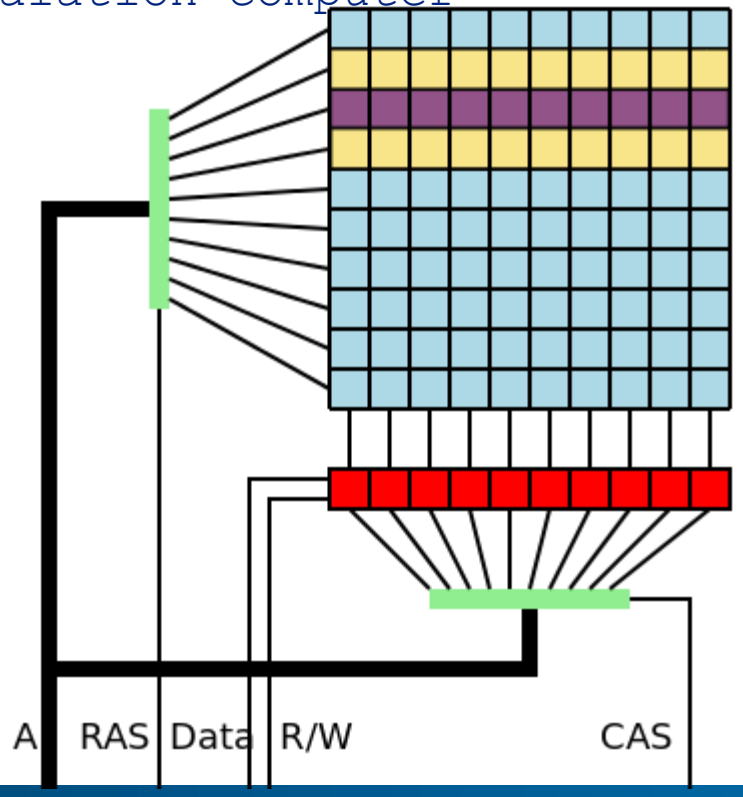
Local Kernel Exploits - Hardware - Rowhammer

RAM Attack: Rowhammer

Row hammer is an unintended side effect in (DRAM) that causes memory cells to leak their charges and interact electrically between themselves, possibly **altering the contents of nearby memory rows** that were not addressed in the original memory access.

The row hammer effect has been used in some privilege escalation computer security exploits

- Write into other memory cells in RAM (bypass OS/CPU protection)
- Integrity of data not guaranteed
- Who is affected? Everyone!
- What is the fix?
  - …



Local Kernel Exploits - Hardware - Meltdown/Spectre

- Meltdown / Spectre
  - Read memory of kernel, of other userspace processes, of other vm's, …
  - Confidentially of “protected” memory pages not guaranteed
    - Stealing encryption keys
    - Bypass kASLR
  - Can be exploited via Browser (JavaScript) (need code execution first)
  - Who is affected? Everyone! (Intel, ~AMD, ARM, …)

Meltdown

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

If your computer has a vulnerable processor and runs an unpatched operating system, it is not safe to work with sensitive information without the chance of leaking the information. This applies both to personal computers as well as cloud infrastructure. Luckily, there are software patches against Meltdown.

Spectre

Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre

Spectre is harder to exploit than Meltdown, but it is also harder to mitigate. However, it is possible to prevent specific known exploits based on Spectre through software patches.

As expected, there's more! - RIDL/Fallout

The **RIDL** and **Fallout** speculative execution attacks allow attackers to **leak private data** across arbitrary security boundaries on a victim system, for instance compromising data held in the cloud or leaking your data to malicious websites. Our attacks leak data by exploiting the 4 newly disclosed *Microarchitectural Data Sampling* (or **MDS**) side-channel vulnerabilities in Intel CPUs. Unlike existing attacks, our attacks can leak arbitrary *in-flight* data from CPU-internal buffers (*Line Fill Buffers*, *Load Ports*, *Store Buffers*), including data never stored in CPU caches. We show that existing defenses against speculative execution attacks are inadequate, and in some cases actually make things worse. Attackers can use our attacks to leak sensitive data despite mitigations, due to vulnerabilities deep inside Intel CPUs.

RIDL

RIDL (*Rogue In-Flight Data Load*) shows attackers can exploit MDS vulnerabilities to mount practical attacks and leak sensitive data in real-world settings. By analyzing the impact on the CPU pipeline, we developed a variety of practical exploits leaking in-flight data from different internal CPU buffers (such as *Line-Fill Buffers* and *Load Ports*), used by the CPU while loading or storing data from memory.

We show that attackers who can run unprivileged code on machines with recent Intel CPUs - whether using shared cloud computing resources, or using JavaScript on a malicious website or advertisement - can steal data from other programs running on the same machine, across any security boundary: other applications, the operating system kernel, other VMs (e.g., in the cloud), or even secure (SGX) enclaves.

We will present our paper on these attacks at the 40th IEEE Symposium on Security and Privacy, on May 20th 2019.

VUSec CISPA

Read the RIDL paper | Cite

Fallout

Fallout demonstrates that attackers can leak data from *Store Buffers*, which are used every time a CPU pipeline needs to store *any* data. Making things worse, an unprivileged attacker can then later *pick* which data they leak from the CPU's Store Buffer.

We show that Fallout can be used to break Kernel Address Space Layout Randomization (KASLR), as well as to leak sensitive data written to memory by the operating system kernel.

Ironically, the recent hardware countermeasures introduced by Intel in recent Coffee Lake Refresh i9 CPUs to prevent Meltdown make them *more* vulnerable to Fallout, compared to older generation hardware.

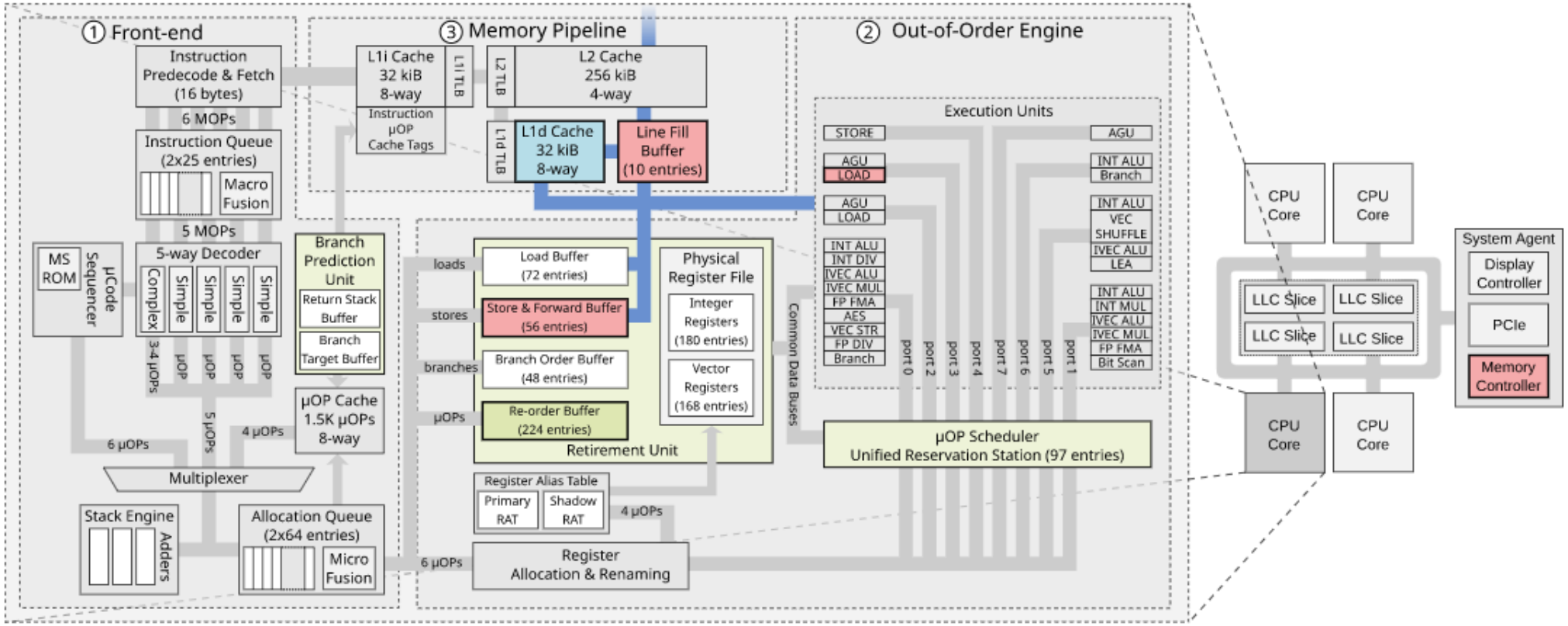
CSE COMPUTER SCIENCE AND ENGINEERING UNIVERSITY OF MICHIGAN | THE UNIVERSITY of ADELAIDE

DATA 61 | WPI | TU Graz | KU LEUVEN

Read the Fallout paper | Cite

As expected, there's more! - RIDL/Fallout

Interactive guide to speculative execution attacks



Click on the various components to interact with them. The full interactive version can be found here and the raw SVG can be found here. There is also a more vibrant colored version (the one used in our paper), which can be found here. These diagrams have been made by Stephan van Schaik (@themadstephan).

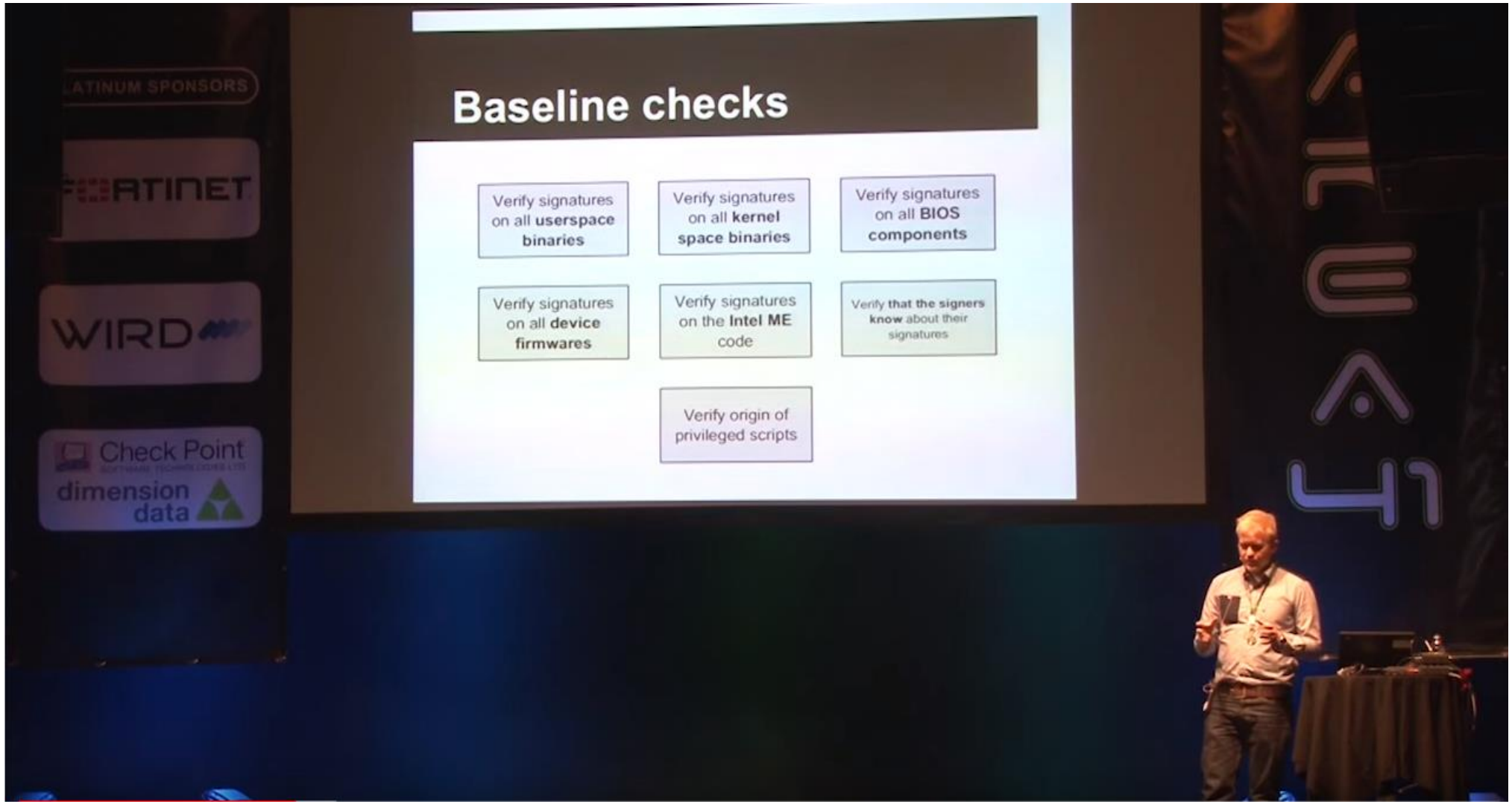
Conclusion

Hardware Attacks - Conclusion

- X86 hardware has layers we cannot control, and which are insecure
- Most embedded platforms are very insecure
- Our hardware itself is insecure
- Nothing can be trusted

Trusting our computers

Area41 2014: Halvar Flake: Keynote



Trusting our computers

Area41 2014: Halvar Flake: Keynote