Hardware Exploitation
Ring -1, -2, -3
Exploiting Ring <0

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

The operating systems

<table>
<thead>
<tr>
<th>Code you know about</th>
<th>Code you don't know about</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring 3 (User)</td>
<td>Ring -2 kernel and ½ kernel</td>
</tr>
<tr>
<td>Ring 0 (Linux)</td>
<td>Control all CPU resources.</td>
</tr>
<tr>
<td>Ring -1 (Xen etc.)</td>
<td>Invisible to Ring -1, 0, 3</td>
</tr>
<tr>
<td></td>
<td>SMM ½ kernel. Traps to 8086 16-bit mode.</td>
</tr>
<tr>
<td></td>
<td>UEFI kernel running in 64-bit paged mode.</td>
</tr>
<tr>
<td></td>
<td>Ring -3 kernels</td>
</tr>
<tr>
<td></td>
<td>Management Engine, ISH, IE.</td>
</tr>
<tr>
<td></td>
<td>Higher privilege than Ring -2.</td>
</tr>
<tr>
<td></td>
<td>Can turn on node and reimage disks invisibly.</td>
</tr>
<tr>
<td></td>
<td>Minix 3.</td>
</tr>
</tbody>
</table>

X86 CPU you know about | X86 CPU(s) you don’t know about
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

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

Embedded Systems

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.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.
What about ASA (Cisco)

https://www.youtube.com/watch?v=eDyxBgIUaR8
Attacking Hardware
Local Kernel Exploits - Hardware

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!
Local Kernel Exploits - Hardware

- Meltdown / Spectre
  - Read memory of kernel or other userspace processes
  - Confidentially of “protected” memory pages not guaranteed
    - Stealing keys
    - Bypass kASLR
  - Can be exploited via Browser (JavaScript)
  - 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.
Conclusion
Hardware Attacks - Conclusion

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

Area41 2014: Halvar Flake: Keynote

Baseline checks

- Verify signatures on all userspace binaries
- Verify signatures on all kernel space binaries
- Verify signatures on all BIOS components
- Verify signatures on all device firmwares
- Verify signatures on the Intel ME code
- Verify that the signers know about their signatures
- Verify origin of privileged scripts
Tusting our computers

Area41 2014: Halvar Flake: Keynote

Baseline checks

- Verify signatures on binaries
- Verify signatures on space binaries
- Verify signatures on components
- Verify signatures on firmwares
- Verify signatures on code
- Verify signatures on privilege scripts