Meltdown

→ exploits side effects of microarchitecture optimization to allow user processes to read arbitrary kernel memory (including physical memory mapped into the kernel)

→ microarchitecture optimization

→ CPU pipeline: fetch, decode, execute, memory subsystem

→ every instr. goes through each stage
→ an instr. may stall in execute (e.g. mem access)
→ or we can schedule subsequent instr. execute out of order execution!

→ results stored in temp buffer/ register
→ retired (visible) in order
→ intel processors enforce perm. check at retire time, not execute time
→ Cache attacks
  → CPU caches: L1, L2, L3
  → cache hit is noticeably faster
  → flush + reload
    → attacker/receiver flush every cache line, wait for a while
    → victim/sender access memory, causing certain cache lines to be filled
    → attacker/receiver reloads every cache line, measure the time for each access to learn the access pattern from victim

→ Kernel memory: physical memory mapped into kernel memory, kernel memory mapped into every process's VAS.
The attack:

1. Read 1 byte of kernel memory
   (should raise an exception)
2. Access [user_array [Kernel byte x 4096]]
   (if this is executed before the exception is raised, can use the temp result)
3. Install a custom SIGSEGV handler, upon an exception, reload cache line to figure out the value of kernel byte

Mitigation:

Kernel page-table isolation

- Only map a small part of kernel necessary for trap/interrupt entrance into every user
- Switch to a full kernel page table once in kernel mode

<table>
<thead>
<tr>
<th>Root Cause</th>
<th>Description</th>
<th>How it affects performance</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel page-table isolation (KPTI) (§4.1.1)</td>
<td>Removes kernel memory mappings from the page table upon entering userspace to mitigate Meltdown.</td>
<td>A kernel entry/exit now swaps the page table pointer, which further leads to subsequent TLB misses.</td>
<td></td>
</tr>
</tbody>
</table>

Security Enhancements: max combined slowdown: 146% poll

recv 63%, send1-read 60%