Recitation 22 — Meltdown

Overview

- Threat model: Attacker has “arbitrary unprivileged code execution on the attacked system”.
- Attacker’s goal: learn secret data (e.g., passwords, private keys)

The attack

- Step 1: load the secret into a register
  - “Listing 2” in the paper does this with the line mov al, byte [rcx] (puts the secret into rax)
  - CPU begins to transfer the virtual address into a physical one, while also checking permission bits of the virtual address.
  - The permission bits will cause an interrupt, but some of the additional lines of Listing 2 will have already (started to be) executed.
- Step 2: transmit the secret
  - Line 5 and Line 7 of Listing 2 multiply the secret value by the page size (Line 5), and add it to the base value of a “probe array” that the attacker has allocated (Line 7). The probe array is allocated such that none of its memory is cached.
  - At this point, the attacker has taken the value of the secret, and mapped that value to a particular memory address. For instance, if the value of the secret was “2”, the address is now base probe array address + 2*page size. What the attacker needs to know, now, is what that memory address is (base probe array address + 2*page size), *not* content that is stored at that memory address (i.e., not the content located at base probe array address + 2*page size).
  - When that address is read, it will be stored in the cache.
- Step 3: receive the secret
  - The attacker iterates over all of the 256 pages of the probe array (memory that it has access to). It measures the access time of each access; the fast one is the address that was cached, and that address - the base address of the probe array is the secret.

Ways to stop Meltdown

- Disable out-of-order execution. Downside: performance suffers.
- Serialize permission checks and register fetches: Downside: overhead (so, again, performance suffers).
- Hard split of user/kernel space. More realistic, but still involves new hardware.
- KAISER, a software solution. Still has limitations because some privileged memory locations are mapped into user space.