MELTDOWN Vulnerability

What is this vulnerability?

The MELTDOWN vulnerability is one of 2 CPU vulnerabilities found in 2018 that abuse speculative branching by CPUs. Usually if an instruction violates a security rule, the CPU cancels the instruction and returns an error to the user, but when a speculative execution has invalid memory access, the instruction would still execute at the time that the MELTDOWN vulnerability was discovered.

What is its root cause?

The root cause of this vulnerability was the fact that when a CPU was doing speculative executions, it would essentially bypass the privilege checking mechanism and access kernel level memory without the proper privilege level. This can lead to malicious users accessing sensitive information that they should not be able to access through timing analysis.

What is the extent of its impact?

The MELTDOWN vulnerability affected all Intel x86 processors, IBM Power Micro processors, and some ARM processors. All machines that ran IOS, MacOS, Linux, or Windows that used the aforementioned processors were vulnerable to this exploit.

How to patch it? (Or how it was patched by the vendor)

There were a few different steps taken in order to patch this vulnerability. First, the kernel page table was isolated from the system when user mode code was being executed. This ensured that attackers were not able to access kernel level memory because the CPU had to context switch in order to access kernel level memory. Next, CPU manufacturers released microcode updates in order to ensure that there were proper security checks when a speculative execution tried to access kernel level memory. Eventually, CPU manufacturers released entirely new CPUs that no longer allow speculative execution to access privileged memory.

How could it have been prevented?

This vulnerability could have been prevented in a couple different ways. First, this could be done by ensuring that speculative executions could not access kernel level memory without proper privileges. While this may add a bit of overhead in the form of more time intensive context switching, it ensures that user mode code can’t access kernel memory. Another option is changing the design of CPUs on a hardware level such that the privilege checks are not bypassed by speculative execution.