Graham Dungan
February 16, 2025
UIN: 332001764

CSCE 413: Software Security
Self-Study: MELTDOWN

What is the Vulnerability?

Meltdown is a vulnerability that abuses speculative execution and late permission checks to circumvent permission checks and read kernel memory addresses. While speculative execution will undo any attempted executed instructions to the kernel space, side-channel attacks can be performed by analyzing cache timings.

What was the Root Cause?

Meltdown is primarily possible due to speculative execution (and, by proxy, branch prediction). This was discussed extensively in my self-study about SPECTRE, and will only be covered briefly here for the sake of brevity. In order to optimize CPU wait times, the CPU attempts to "guess" the output of certain instructions based on previous operations. If the output was correct, then the CPU saved time. If the output was incorrect, it will calculate the value at no true expense of the process. The speculative execution paradigm is powerful but leaves a major security vulnerability in its wake. Since the CPU will "guess" outputs of certain variables and check them later (once cached) attackers can avoid immediate checks and run malicious instructions in the meantime. Suppose there is an instruction to read the kernel space of memory. Before reading, the machine will check to ensure the instruction has proper permissions. This check can be circumvented by a race condition, making it such that the speculative execution will check the permissions of the memory access later. While reads to memory are already powerful, this technique can also be used to perform similarly powerful side-channel attacks.

What is the Extent of its Impact?

Intel's hardware was most effected by the MELTDOWN vulnerability because their CPU's did not abandon instruction executions early if there is a permission mismatch, unlike AMD. Both AMD and Intel use speculative execution, but they have different measures for making permission checks. It followed that nearly all Intel CPU's had this flaw, while only a handful of AMD CPU's were recognized as having the MELTDOWN vulnerability. While MELTDOWN appeared at the same time as SPECTRE, there were no massively publicized instances of MELTDOWN being used in any attacks. The only specific uses of MELTDOWN attacks were in proof-of-concept demonstrations, as seen in this presentation.

How to Patch it?

One of the most prominent solutions to MELTDOWN was changing the mapping of the kernel space in the runtime stack. With this new method, direct kernel mappings are removed from the page table if a process is launched in user mode. While this adds some performance overhead, especially during context switching, it prevent speculative reads to kernel pages. Different OS manufactures implemented this memory protection strategy or forms of it.

How Could it Have Been Prevented?

Like SPECTRE, this vulnerability was partially founded from speculative execution being created without software security in mind (prioritizing efficiency over protection). Unlike SPECTRE, MELTDOWN had a unique software-oriented prevention- a continued protection of kernel-level memory. Using the philosophy of least privilege, it was never considered that there would be a need to strictly separate user space at the cost of performance.

References

1. Meltdown (security vulnerability) - Wikipedia
2. What are the differences between Meltdown and Spectre? - Stack Overflow
3. Meltdown Security Vulnerability - GeeksforGeeks
4. Meltdown Security Paper - USENIX Security '18
5. Kernel page-table isolation - Wikipedia