Meltdown and Spectre

Began Bajrami, Rahmi El Mechri

June 25, 2022

Abstract

1 Introdutcion

2 Introduction

2.1 Out of Order Execution

Out of order is a technique used by many CPUs nowadays, the main reason being the improvements on perfomance that it brings, allowing CPU to execute instructions in a different order than how the program was compiled, in order to avoid wasting computational power. In this paper we refer to out-of-order as 'out-of-order issue out-of-order completion'.

2.1.1 Tomasulo's algorithm

For a better understaindg of the Intel's CPU architecture, here is a brief introduction to Tomasulo's algorithm which first introduced to techniques like register renaming, reservation station and common data bus (CDB), which allowed out-of-order execution.

> In 1967, Tomasulo developed an algorithm that enabled dynamic scheduling of instructions to allow out-of-order execution.

[1] Tomasulo's reservation station allows instructions that operate on the same physical registers to rename registers (register renaming) and use the last logical one to solve read-after-write (True data dependeny, or RAW), write-after-read (Antidependency, or WAR) and write-after-write (WAW) hazards. Moreover, this lets the execution units use data values as soon as they are computed rather then reading value from a register, writing the result on the register and then, again, reading it. [1][2 wikipedia] All execution units are directly (and individually) connected to the reservation station via a common data bus (CDB), where operands of instructions are passed as soon as they're available. This is useful if an instruction is waiting for an operand that is not already on the register, so it can directly listen on the CDB to recive the operand as soon as it is available.

2.1.2 Intel Architecture

Meltdown researchers provide a simplified illustration of a single core of the Intel's Skylake microarchitecture which well illustrates the architecture. The pipeline of Intel's Skylake processors consists of the front-end, which fetches instructions from memory and decodes them into micro-operations (since intel's processors are CISC, while Superscalar/superpipelined processors suits better on RISC, the processor must decode complex operations into smaller, less complex micro-operations in order to make the most of out-of-order execution), the back-end (execution engine), which implements out-of-order execution, and the memory subsystem. The Reorder Buffer is responsible of register allocation, register renaming and retiring. It is also responsible of reordering instruction outputs as was intended by the program(mer). Micro-operations are directly forwarded to the Unified Reservation Station that queues the operations on exit ports that are connected to Execution Units. Of course, Intel's Skylake has it's branch predictor. Usally branch predictors are implemented with *taken/not taken* bits which tracks the history of a branch and indicates if previosly the branch was taken or not taken. This can be implemented with 1-bit or 2-bit counters.

2.2 Address Spaces

Usually, CPUs support virtual address spaces to isolate processes from each other and to let compilers use logical addresses instead of directly accessing physical memory addresses. Virtual addresses are then translated to physical addresses. For optimization of memory usage, paging is also used to reduce memory usage. Paging is also used to separate User Space addresses from Kernel Mode addresses, in order to let only privileged processes to access kernel address space. Translation tables are used in order to define virtual to phisical mappings and also protection properties such as readable, writeble, executable and whether the page is accessible by user or not (meaning that only kernel mode processes can access the page). Every proces has its own translation table which is held on a special CPU register, so "on each context switch the **operating system** updates this register with the next process' translation table address in order to implement per process virtual address spaces". Each virtual address space itself is split into a user and a kernel part.

2.2.1 Exploitation and mitigation

Attacks that are targeting memory corruption bugs often requires the knowledge of addresses of specific data. ASLR mitigation has been introduced to randomize address space layout in order to obfuscate memory mapping to attackers. KASLR (Kernel Address Layout Randomization) was introduced to protect the kernel, randomizing the offsets where drivers are located on every boot, making attacks harder as they now require to guess the location of kernel data structures.

2.2.2 Side channel attacks

WIKIPEDIA:

> In computer security, a side-channel attack is any attack based on extra information that can be gathered because of the fundamental way a computer protocol or algorithm is implemented, rather than flaws in the design of the protocol or algorithm itself or minor, but potentially devastating, mistakes or oversights in the implementation.

Side-channel attacks allow to detect the exact location of kernel data structures or derandomize ASLR. A combination of software bug and the knowledge of these addresses can lead to privileged code execution.

3 Meltdown

4 Spectre

5 Exploits

6 Impact

7 Mitigations

8 Conclusion