CSE 351 AB/BB Section 9

Meltdown and Wrap-Up

With Callum and Amy!

Meltdown

Meltdown and Spectre

- *Meltdown* is a critical security flaw disclosed in January 2018 affecting a huge variety of modern processors. It allows a process to read all memory, bypassing usual security checks!
- *Spectre* is a somewhat similar vulnerability, discovered around the same time, which exploits speculative execution to read private memory from other processes.
  - It is harder to exploit, but harder to mitigate
- We will be focusing on Meltdown today

Speculative Execution

We haven't quite been honest with you this quarter…

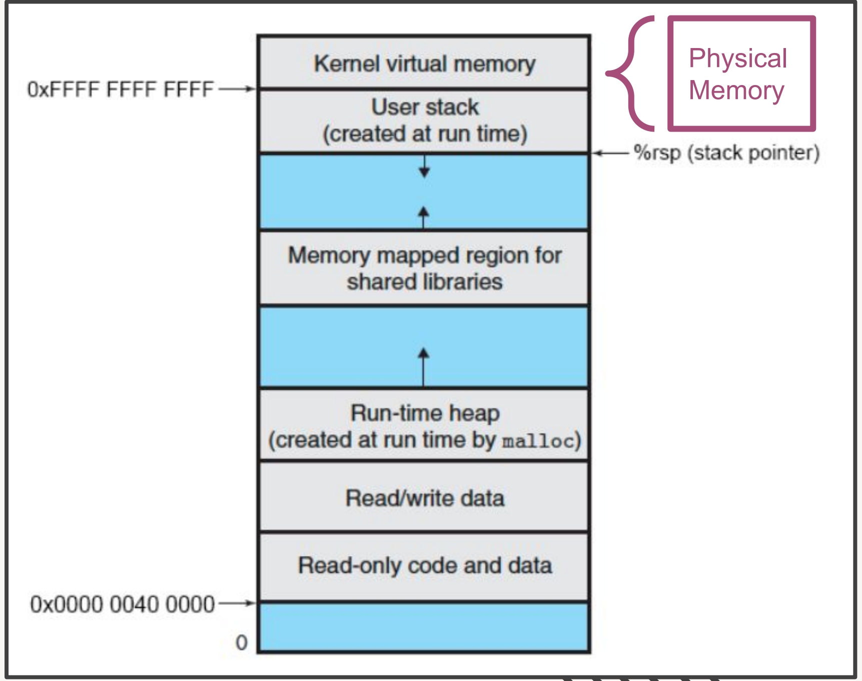
- Modern Intel (and other) CPUs actually work hundreds of instructions ahead of what your program is doing at any given time
  - Sometimes, instructions are not even executed sequentially; many modern processors support *out-of-order execution*, where the CPU can schedule future instructions while waiting on a previous slower one to finish (e.g. one which reads from memory)
- Branch prediction: CPU tries to predict which branch your program will take, and executes those instructions ahead of time ("speculatively")
  - Based on past program behavior, e.g. if a branch is known to be taken almost all of the time, the processor will work ahead on this branch and commit state changes if the guess was correct

Speculative Execution

- If the CPU predicts incorrectly, the results are discarded before the program knows they exist
  - Maintains correctness, just loses a little speed
- However…
  - Instructions executed speculatively **do not always trigger exceptions for access to privileged memory locations**
  - And, these instructions **affect the cache**

Virtual Memory Space

- Modern operating systems map kernel memory into each process's VA space.
  - Speeds up system calls
- Kernel memory often contains mapping for the computer's entire physical memory



Meltdown Assumptions

- **All of physical memory** is mapped to kernel addresses in user process
  - Start address (VA in user process) of physical memory is known, A_k
  - Physical memory is K bytes total, and mapped directly, $[A_k \ldots (A_k + K - 1)]$
- An exception (illegal memory access) can be handled/suppressed
- Kernel Address Space Layout Randomization not used
  - Similar to randomizing start address of stack, kernel data structure start address can be randomized

Meltdown Example

- Set up an array that is large enough to span 255 pages
  - So we can index into it using a byte
- Speculatively try to access a byte from kernel memory
- Use that as an index into the array, multiplied by your system’s page size to ensure that adjacent accesses are not cached together
- Try to access each array index (separated by the page size) and time how long it takes.
- The index that took much less time to access corresponds to the secret data!

Meltdown Example

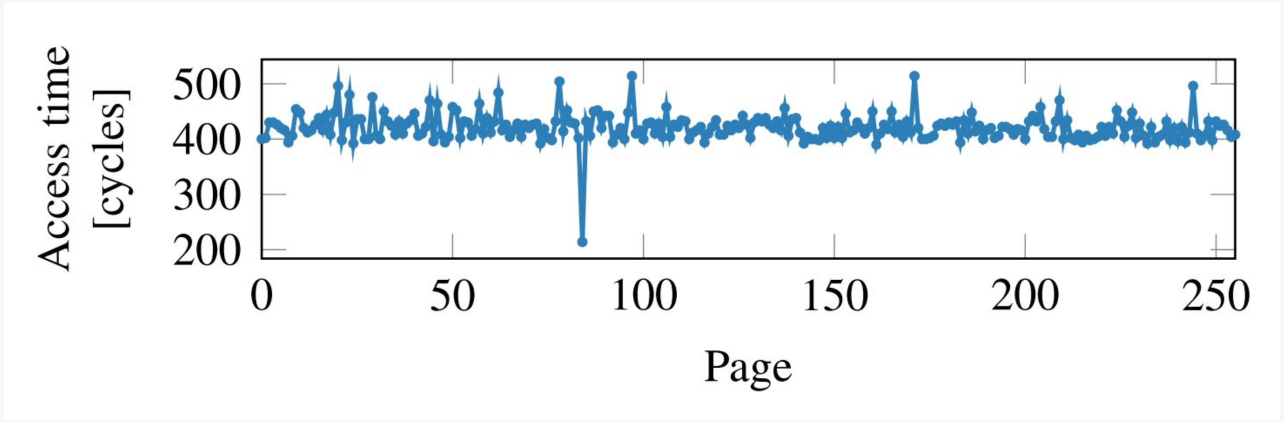
```
char array[256 * PAGE_SIZE];
flush_cache();

if (<condition that is always false>)
   char data = *(<kernel address>);
   char idx = array[data * PAGE_SIZE];

for (int i = 0; i < 255; i++)
  access(array[i * PAGE_SIZE]);
```

Executed speculatively!

Meltdown Example



- Index with unusually fast access corresponds to the secret data!

Meltdown Summary

- Allows a user process to read all of physical memory on the system, which is mapped in kernel addresses and by extension in user process address space

- Visit meltdownattack.com to read the original papers for Meltdown and Spectre if you’re interested in learning more!

Mitigation

- KAISER (patch by Gruss et al.) implements a stronger isolation between kernel and user space. It leaves physical memory unmapped in kernel address space.
    - https://lwn.net/Articles/738975/
- Use an AMD processor, which doesn’t bypass memory protection during speculative execution.

That’s All, Folks!

Thanks for attending section and making this quarter great.

The remainder of our section time will be office hours if you have any questions. Feel free to stick around if you would just like to talk as well :)

We’re in the home stretch now. Best of luck on everything!

Course Evaluations

https://uw.iasystem.org/survey/228930