Sp18 CS 61B Discussion 5

Welcome!

Wayne Li

wli2@berkeley.edu

https://wayne-li2.github.io/

Announcements

- Midterm Grading: We’re working on it :)
- Lab: Mandatory this week!

Quiz Instructions

- If you haven’t yet, please also **neatly** put your email address **outside the name box** if you want to be emailed!
- Bubble number **41**.

Aside

Spectre/Meltdown

What is Spectre/Meltdown?

- Flaw in **every** major chip in the last 20 years.
- Discovered by Google's Project Zero team.
  - "**Zero-day exploit**": Programs that exploit vulnerabilities before the vulnerabilities are public.

Speculative Execution (CS 61C)

- Let’s say we have the following piece of code:

```
If a == true:
    do x
else:
    do y
```

Speculative Execution (CS 61C)

- Chip will compute both branches first.
  - Then pick the correct branch later.

Caching (CS 61C)

- This was our “extra” in week 2!
- Recall our library analogy.

Combined?

- Data from speculative execution (both branches) stored on the cache.
- Still… seems okay.

Problem (CS 162)

- Computer memory is divided into two parts:
  - User memory (for you!)
  - Kernel memory (for the OS)
    - Contains all sorts of private/secure data.

Problem (CS 162)

- Accessing kernel memory:
  - 1. Perform privilege check.
  - 2. If secure, proceed. Else deny permission.

Problem (CS 162)

- With speculative execution, chip performs both branches.
- With caching, stores sensitive data in the cache.

One Variant of the Attack (CS 161)

- 1. Probe a piece of protected memory repeatedly.
- 2. Force the CPU to guess the permissions check.
- 3. Monitor cache, and detect differences in the cache.
- 4. Extract from cache / guess data via the address.

Moral of the Story

- **It’s not your fault.**
- **Low level stuff is cool!!**

References

- https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html
  - Original blog release by Google

Onto Discussion