CCSEP GROUP 7

Meltdown & Spectre

Anh Dinh

Software Eng. Student

Hoang Ngo

Cyber Sec. Student

Olimar Ramilo

Software Eng. Student

Introduction

Vulnerability

Discovered in 2018, Meltdown & Spectre are both hardware vulnerabilities that exist due to the modern Architecture of Processors.

Risks

The risks of a successful meltdown attack can result in attackers gaining access to confidential and sensitive information such as:

- Application Data
- Passwords
- Encryption keys
- Critical System Information

network security

Secure with strong passwords and multi-factor authentication

Background Knowledge

Cache vs RAM Timing

- Faster access to cache memory.
- Slower in RAM

Flush+Reload

- Side channel attack via Cache.
- Timed attack using cache memory to complete instructions faster

Kernel Space

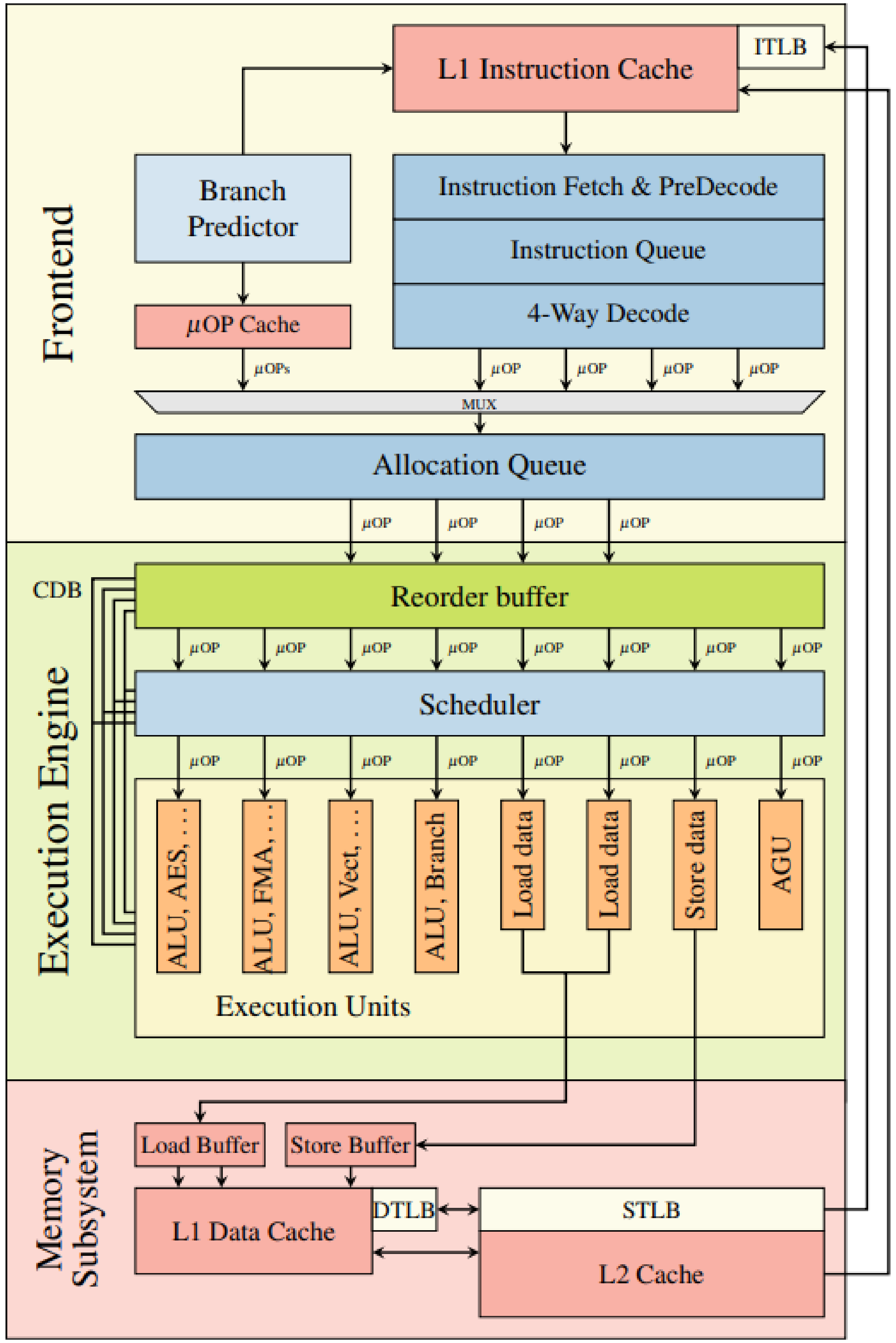
- Privileged part of the OS that performs critical tasks.
- Separated from User space programs which has less privileges and no direct access to hardware

Out-of-Order Execution

- A technique used by modern processors to enhance CPU performance.
- Instructions are executed out of order than they appear in the program.
- Enhances performance so that processors don't stall when waiting for available resources

Speculative Execution

- A processor makes educated guesses about which instructions to execute next, based on likely outcomes of branches in the code.
- It does this to keep itself busy and speed up operations.
- If the guess is wrong, it just goes back and corrects it without causing any harm.

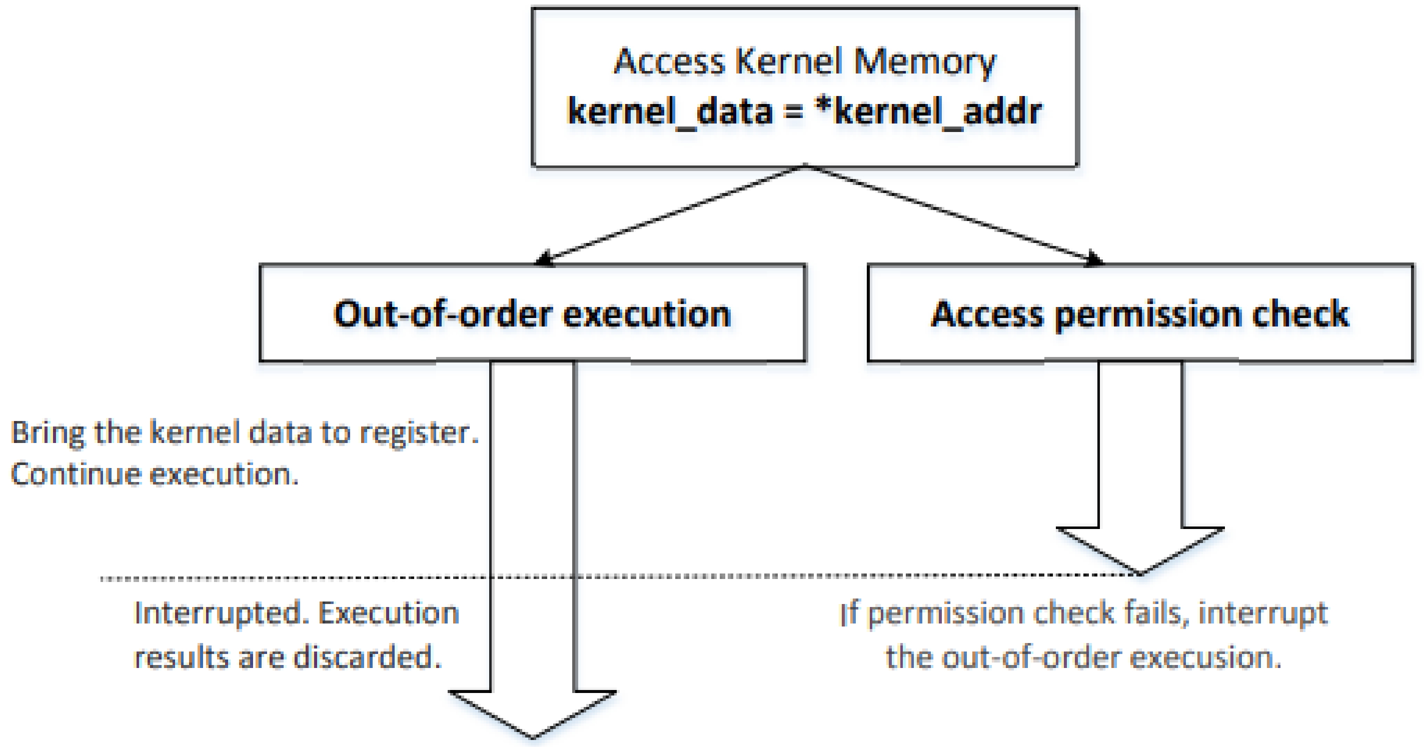


Meltdown Attack

Is a Hardware Vulnerability among Modern Processors. in Intel Architecture.

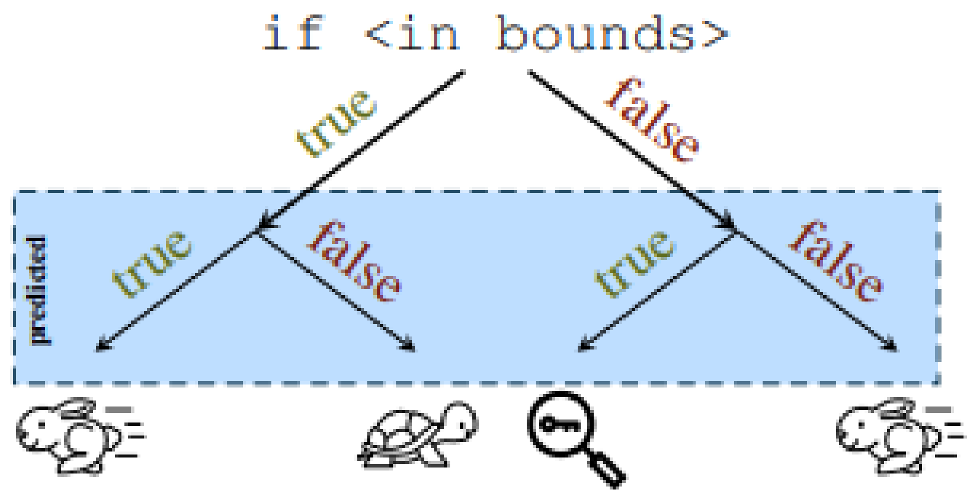
- Only works well with machines running on Intel CPU’s
- Due to race condition we can access Kernel Space data through User Space
- Kernel address must be known for this attack to work
  - Attackers must find a way to get a kernel address
  - Or guess
- The data must be in cache as well or meltdown will be difficult

Exploiting Out-Of-Order Execution



Exploiting Conditional Branch Prediction

- Is an optimisation tehcnique used to modern CPUs
- Predicts the outcome of conditional branches in code
- CPU uses past branch outcomes to make predictions
- Prediction aims to aim to minimize idle time and keep the CPU executing instructions efficiently.



Spectre Attack

- Select a conditional branch we want to exploit
- Train the CPU to always take the true branch inside restrictedAccess()
- Exploit Out-of-order execution technique on the branch
- Obtain the secret value from Cache memory

```
uint8_t restrictedAccess(size_t x)
{
  if (x < buffer_size) {
     return buffer[x];
  } else {
     return 0;
  }
}
```

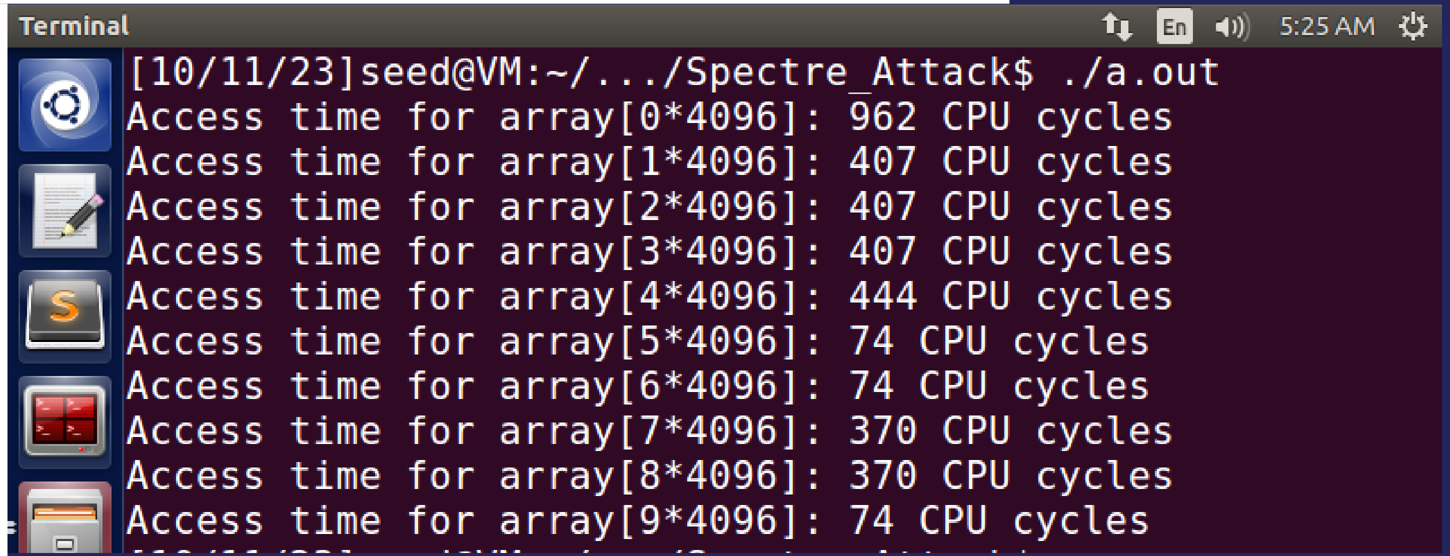
```
// Flush buffer_size and array[] from the cache.
_mm_clflush(&buffer_size);
for (i = 0; i < 256; i++)  { _mm_clflush(&array[i*4096 + DELTA]); }
```

Step 1: Identifying Cache vs Main Memory

To Compile:

- gcc -march=native CacheTime.c -o CacheTime
- ./CacheTime

Intended Result:



Observation

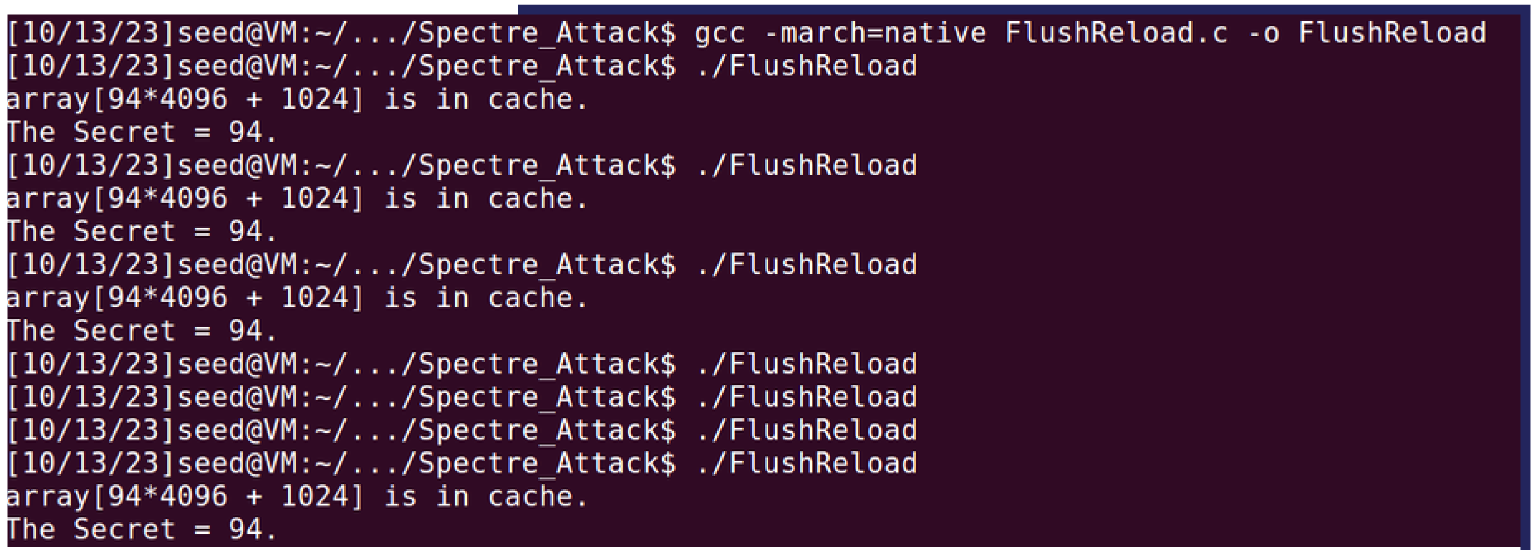
- Array 5, 6, 9 takes 74 cycles to access.
  - -> Accessing from Cache
- Other arrays show 100+ cycles to access
  - -> Accessing from RAM

Step 2: Flush & Reload

To Compile:

- gcc -march=native FlushReload.c -o FlushReload
- ./FlushReload

Intended Result:



Goal: Find one-byte secret value

1. Cache Purge (Flush):
2. Cache Access (Invoke “Victim” Function)
3. Cache Reconnaissance (Reload)

Meltdown

To Compile:

- make
- sudo insmod MeltdownKernel.ko
- dmesg | grep 'secret data address'
- gcc -march=native MeltdownAttackImproved.c -o MeltdownAttackImproved
- ./MeltdownAttackImproved

Intended Result:

```
[10/11/23]seed@VM:~/.../Meltdown_Attack$ gcc -march=native MeltdownImproved.c
[10/11/23]seed@VM:~/.../Meltdown_Attack$ a.out
The secret value is 83 S
The number of hits is 908
The secret value is 69 E
The number of hits is 934
The secret value is 69 E
The number of hits is 941
The secret value is 68 D
The number of hits is 953
The secret value is 76 L
The number of hits is 823
The secret value is 97 a
The number of hits is 914
The secret value is 98 b
The number of hits is 939
The secret value is 115 s
The number of hits is 890
```

Spectre Attack

To Compile:

- gcc -march=native SpectreAttackImproved.c -o SpectreAttackImproved
- ./SpectreAttackImproved

Intended Result:

```
[10/12/23]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreAttackImproved.
c -o SpectreAttackImproved
[10/12/23]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
The secret is:
S
o
m
e

S
e
c
r
e
t

V
a
l
u
e
```

Mitigation Strategy

Hardware patches are hard to implement

- Software Patches to OS (KAISER)
    - Reduced amount of Kernel memory that could be mapped to User level

No code based mitigation strategy

- Regular Software Updates
- Least privilege
- Virtualize environments
- Monitoring and Regular Auditing

Meltdown

- Won’t work outside Intel processors
- More difficult to implement on newer systems

Spectre

- Works on Most Processors
- Harder to detect due to speculative execution

References

Computer security: A hands-on approach | udemy. Available at: https://www.udemy.com/course/du-computer-security/ (Accessed: 12 October 2023).

Dingchang (No Date) Dingchang/Seedlab, GitHub. Available at: https://github.com/Dingchang/SeedLab/tree/master (Accessed: 13 October 2023).

Kocher, P. et al. (2018) Spectre attacks: Exploiting speculative execution, arXiv.org. Available at: https://arxiv.org/abs/1801.01203 (Accessed: 13 October 2023).

Lipp, M. et al. (2018) Meltdown, arXiv.org. Available at: https://arxiv.org/abs/1801.01207 (Accessed: 13 October 2023).

SamuelXing (No Date) Samuelxing/MeltdownDemo: Meltdown attack demo., GitHub. Available at: https://github.com/SamuelXing/MeltdownDemo/tree/master (Accessed: 13 October 2023).

SEEDLabs Meltdown Attack. Available at: https://seedsecuritylabs.org/Labs_16.04/System/Meltdown_Attack/ (Accessed: 13 October 2023).

SEEDLabs Spectre Attack . Available at: https://seedsecuritylabs.org/Labs_16.04/System/Spectre_Attack/ (Accessed: 13 October 2023).