Meltdown Attack Lab

Xinyi Li

February 25, 2020

Task 1

To compile `CacheTime.c` successfully, you should add 2 lines first to resolve the type alias:

```
#include <stdio.h>
#include <stdint.h>
```

Yes. Obviously, the accesses of `array[3*4096]` and `array[7*4096]` are extremely faster than that of the other elements, even though the access times of each elements seemed are randomly various among 10 attempts.

Task 2

Somehow it always finds the correct secret. So I modify the `CACHE_HIT_THRESHOLD` from 80 to 60, It begins to fail to find the secret with nothing output for a few times.

Task 3

```
[ 2184.475777] secret data address:f8922000
```

Task 4

No. I cannot access the kernel memory from user space. After executing the test program, the error message of *Segmentation fault* appears.

Task 5

It handles the exception and prints

```
Memory access violation!
Program continues to execute.
```

Task 6

I get the outputs

```
Memory access violation!
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[7*4096 + 1024] is in cache.
The Secret = 7.
```

It has a little difference with expected in the textbook. I tried several times but the outputs always had two candidates as above. I guess that because the cache line in my virtual machine is large enough to store two blocks of data, after putting `array[7*4096 + 1024]` into the cache, there is still some space for data when accessing all candiates to pick up the secret. Then the first elements during accessing, `array[7*4096 + 1024]`, is also put into the cache.

To verify my assumption, I replace all `4096` with `4096*2` in the code. So it remains no space for another data in the cache. As expected, the result is clear

```
Memory access violation!
array[7*4096*2 + 1024] is in cache.
The Secret = 7.
```

Therefore, I remain the modification to get expected results in the following tasks.

Task 7

Taks 7.1

It shows

```
Memory access violation!
array[0*4096*2 + 1024] is in cache.
The Secret = 0.
```

However, consider the situation in Task 6 above, 0 is not the real kernel data. Actually, I cannot get the kernel data via such a simple approach.

Task 7.2

Add the code in a place between `flushSideChannel()` and `sigsetjmp()`. Anyway, It doesn't work as well.

Task 7.3

Somehow, it still fails to steal the actual secret value. Even though I tried many times and modified the loop number.

Task 8

The attack fails.

I find that probably I cannot reproduce the attack according to the instruction. Because in the *lab environment* mentions:

> First, the Meltdown vulnerability is a flaw inside Intel CPUs, so if a student's machine is an AMD machine, the attack will not work. Second, Intel is working on fixing this problem in its CPUs, so if a student's computer uses new Intel CPUs, the attack may not work

The virtual machine still relies on the hardwares of my computer **physically**, which is equipped with the last 10th intel CPU:

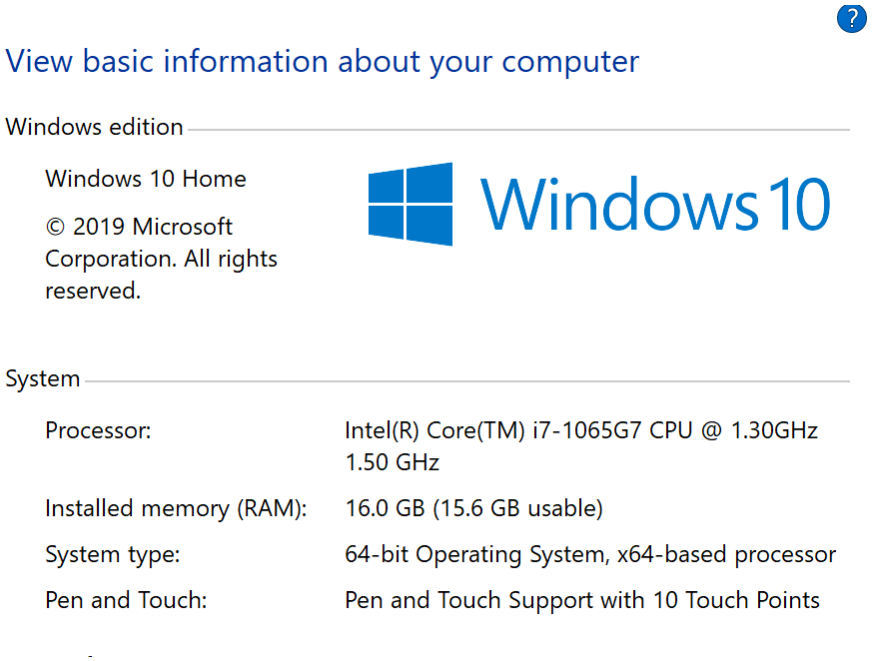

Figure 1: Basic info about my computer

So I need to do the lab on another computer.