Meltdown Attack Lab

Ethical Hacking 2022/23, University of Padua

Alessandro Brighente, Eleonora Losiouk, Gabriele Orazi, Francesco Marchiori

---

1 Task 1

In this task we have to find a good threshold to distinguish between cached and not cached data. After a cople of tries like the one presented in the following, we set the threshold to 200, but the value could vary for your machine.

```
seed@VM:~/.../scripts$ ./CacheTime
Access time for array[0*4096]: 1504 CPU cycles
Access time for array[1*4096]: 258 CPU cycles
Access time for array[2*4096]: 298 CPU cycles
Access time for array[3*4096]: 138 CPU cycles // cached
Access time for array[4*4096]: 420 CPU cycles
Access time for array[5*4096]: 454 CPU cycles
Access time for array[6*4096]: 456 CPU cycles
Access time for array[7*4096]: 134 CPU cycles // cached
Access time for array[8*4096]: 290 CPU cycles
Access time for array[9*4096]: 268 CPU cycles
```

It is important to notice that other software running in the same machine can have an impact on the access time: therefore, on different runs, there may be big differences. Try to find a good threshold but take into considerations that the following attacks could fail from time to time.

2 Task 2

If you did everything correctly, you have to see something like:

```
seed@VM:~/.../scripts$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
```

We find it in about 70% of the 20 run of the application. However, it is really variable, so values above 50% are great. Furthermore, it you have bad results, try to modify the threshold and rememeber to recompile your program.

3 Task 3

Follow the instructions and you wil get the position of the secret in the kernel space:

```
seed@VM:~/.../scripts$ dmesg | grep "secret"
[ 1870.562372] secret data address:f9e2c000
```

The `0x` may be omitted.

4 Task 4

You have to add your address in the code and run it. You will obtain something like:

```
seed@VM:~/.../Solutions$ ./task4
Segmentation fault
```

Even running the code with root priviledge will result in the same fault:

```
seed@VM:~/.../Solutions$ sudo ./task4
Segmentation fault
```

The problem is in line (2) where your code try to access an address which is in the kernel space. It is not part of the application and so the kernel prevent your program to read this part of the memory.

5 Task 5

As predicted, the code will detect a memory access violation and will then proceed without reaching the other part of the `if` statement:

```
seed@VM:~/.../scripts$ ./ExceptionHandling
Memory access violation!
Program continues to execute.
```

6 Task 6

To prove the execution of the Line (2) we can see that the only value of the arry which is accessed in less than the threshold is the element of index 7, which is the one accessed in Line (2). It means that the values was cached. Furthermore, the values is `7` as it have to be since the Line (1) triggers an exception and the results of Line (2) will be discarded by the CPU.

```
seed@VM:~/.../scripts$ ./MeltdownExperiment
Memory access violation!
array[7*4096 + 1024] is in cache.
The Secret = 7.
```

7 Task 7.1

For this task you simply have to change `7` with `kernal_value` in the `meltdown()` method. Then, we expect to find only one hit in the cache represented by the value of `kernel_value`. Unfortunately, this is not working: sometime there are no cache hit, sometime too many, or only a few. It is difficult because of the race condition in the out-of-order execution since, for the attack to be successful, we need that Line (2) have to be executed before Line (1).

8 Task 7.2

With this modification we can see a really small improve in the success rate. Still, the attack need better performances to be used in practice.

9 Task 7.3

With this modification the success rate is improved. Increasing the number of loops in the assembly code we can see a small improvment in the results.

10 Task 8

By setting the parameters (threshold and address) in the `MeltdownAttack.c` file we can finally get a complete version of the attack. This is the execution to steal the first letter of the secret:

```
seed@VM:~/.../scripts$ ./MeltdownAttack
The secret value is 83 S
The number of hits is 906
```

Which is `S` (ascii value 83).