# CS 5153/5053 Network Security, Spring 2023

## Project 5: Meltdown Attack

## Report

### Student: Austin Tyler Conn

Contents

[Link to Source Code 4](#_Toc132374939)

[Host Environment Used 4](#_Toc132374940)

[Additional Information 4](#_Toc132374941)

[Task 1 5](#_Toc132374942)

[How do you compile the code for this task 5](#_Toc132374943)

[Screenshots 5](#_Toc132374944)

[Task 2 6](#_Toc132374945)

[How do you compile the code for this task 6](#_Toc132374946)

[Screenshots 6](#_Toc132374947)

[Task 3 7](#_Toc132374948)

[How do you compile the code for this task 7](#_Toc132374949)

[Screenshots 7](#_Toc132374950)

[Task 4 9](#_Toc132374951)

[How do you compile the code for this task 9](#_Toc132374952)

[Screenshots 9](#_Toc132374953)

[Task 5 10](#_Toc132374954)

[How do you compile the code for this task 10](#_Toc132374955)

[Screenshots 10](#_Toc132374956)

[Task 6 11](#_Toc132374957)

[How do you compile the code for this task 11](#_Toc132374958)

[Screenshots 11](#_Toc132374959)

[Task 7.1 12](#_Toc132374960)

[How do you compile the code for this task 12](#_Toc132374961)

[Screenshots 12](#_Toc132374962)

[Task 7.3 13](#_Toc132374963)

[How do you compile the code for this task 13](#_Toc132374964)

[Screenshots 14](#_Toc132374965)

Link to Source Code <https://github.com/austinc3030/meltdown_m11809075>

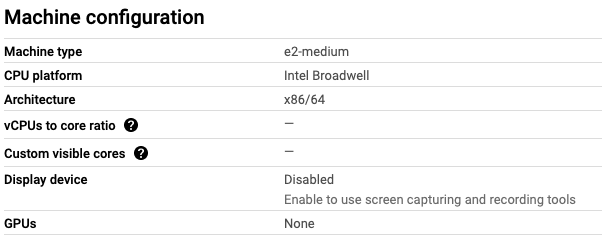
## Host Environment Used

Operating System: Ubuntu 20.04 LTS



Hardware: Google Cloud E2 Instance

Google Cloud Machine Configuration:



Links Used for Environment Setup:

* [seed-labs/seedvm-cloud.md at master · seed-labs/seed-labs (github.com)](https://github.com/seed-labs/seed-labs/blob/master/manuals/cloud/seedvm-cloud.md)
* [seed-labs/create\_vm\_gcp.md at master · seed-labs/seed-labs (github.com)](https://github.com/seed-labs/seed-labs/blob/master/manuals/cloud/create_vm_gcp.md)

## Additional Information

I noticed in the 20.04 version of the labs, the SEEDLabs pdf contains the following excerpt regarding the use of a 20.04 SEEDLabs VM:

**“This lab has been tested on our pre-built Ubuntu 16.04 VM, which can be downloaded from the SEED website. On the SEED Ubuntu 20.04 VM, Tasks 1 to 6 still work as expected, but Tasks 7 and 8 will not work due to the countermeasures implemented inside the OS.”**

I am using a SEEDLabs 20.04 VM and believe this may have had an influence on the results I was able to obtain. As noted in the assignment pdf Note 1, I have included all screenshots and necessary documentation of my attempts to complete tasks 7.1 and 7.3.

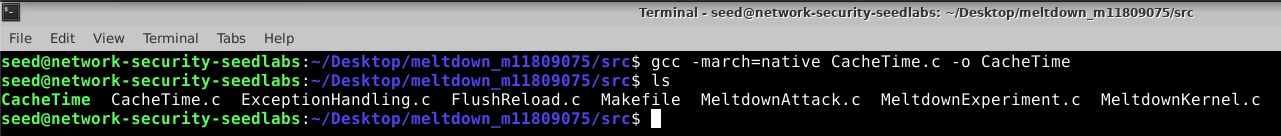
## Task 1

### How do you compile the code for this task

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

### Screenshots

Compilation



Execution

Graphical user interface

Description automatically generated

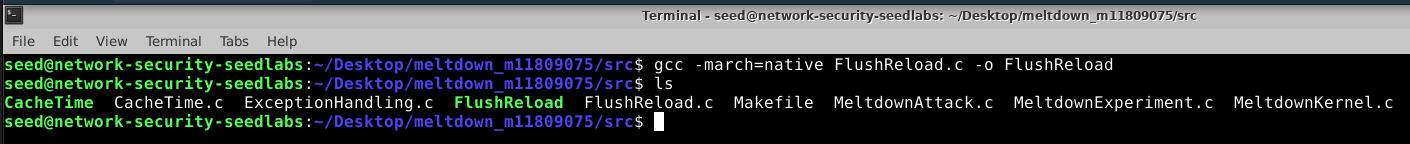
## Task 2

### How do you compile the code for this task

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

### Screenshots

Compilation



Execution

**Note:** To see some variation, I did have to alter CACHE\_HIT\_THRESHOLD to 203 to see instances where an incorrect secret value or multiple secret values were found. However, setting this much lower yielded very reliable results in getting the correct secret value. On the left is with CACHE\_HIT\_THRESHOLD set to 203, on the right is with CACHE\_HIT\_THRESHOLD set to 80. The value 80 also closely corresponds with the values seen in Task 1 and makes sense.

Text

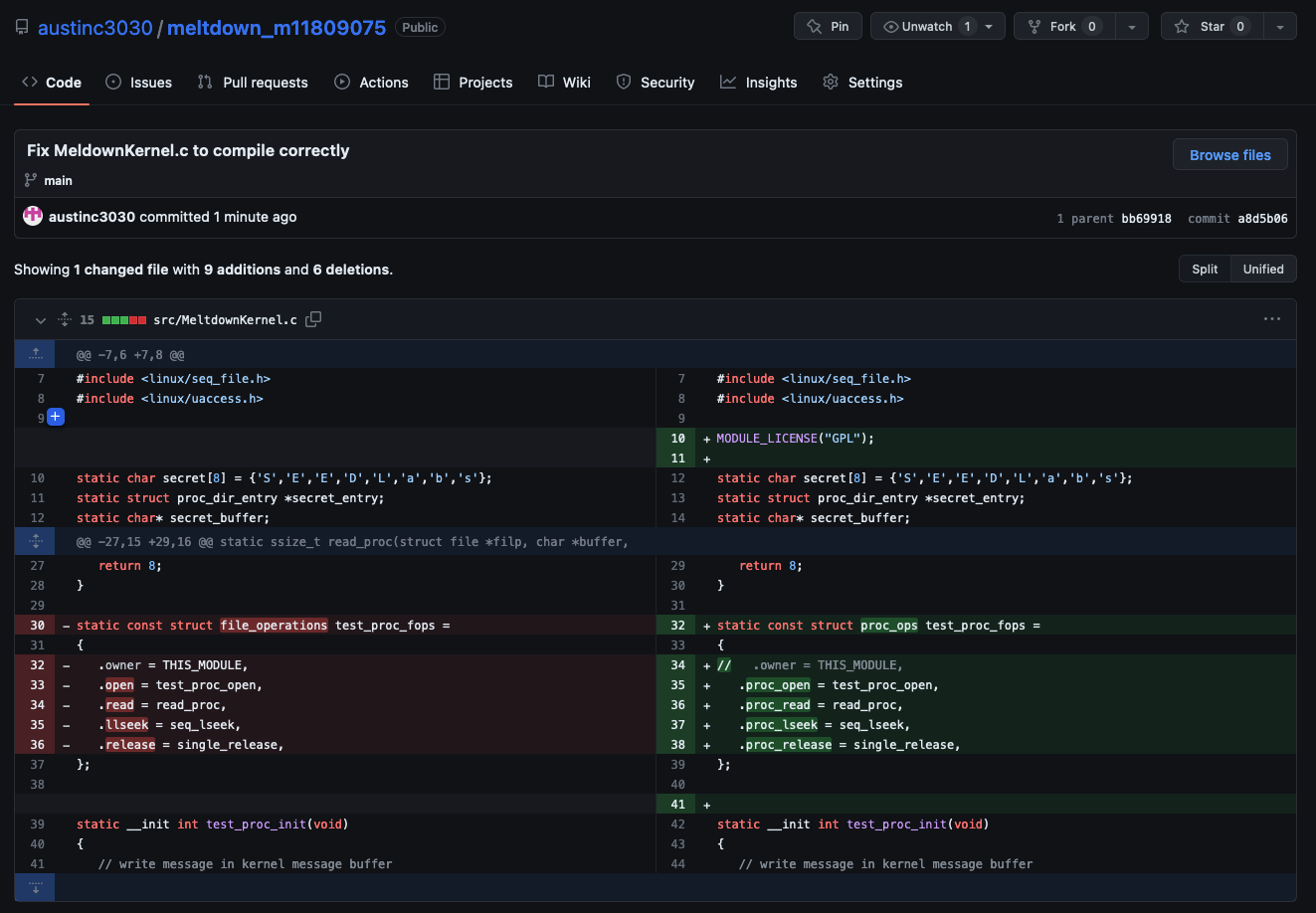
Description automatically generated Graphical user interface, text

Description automatically generated

## Task 3

### How do you compile the code for this task

To get *MeltdownKernel.c* to compile, the following changes (displayed as a diff from GitHub) are required.



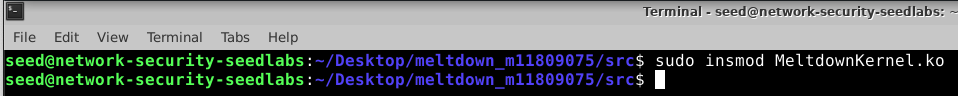
Upon completing the change, all that is required to compile is to run `make`

Graphical user interface, text, application

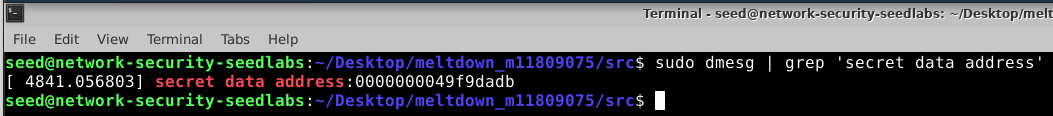
Description automatically generated

### Screenshots

To install the kernel module



Find the secret data’s address from the kernel message buffer (note the use of *`sudo`*)



## Task 4

### How do you compile the code for this task

Place the code given in the pdf into *KernelMemoryAccessTest.c*



Compile using `gcc -march=native KernelMemoryAccessTest.c -o KernelMemoryAccessTest

A screenshot of a computer

Description automatically generated with medium confidence

### Screenshots

Execution fails

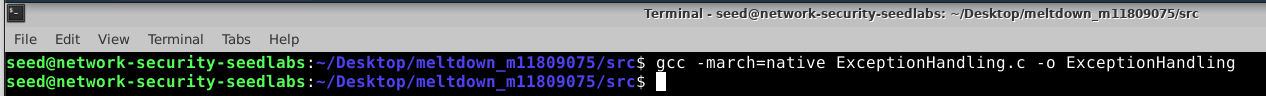


## Task 5

### How do you compile the code for this task

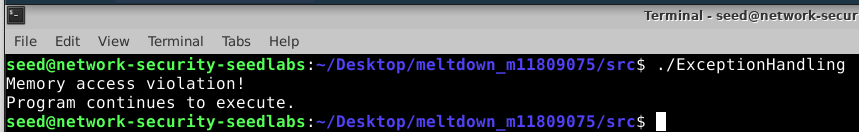
Compile ExceptionHandling.c by running

`gcc -march=native ExceptionHandling.c -o ExceptionHandling`



### Screenshots

Execution is successfully continued after the memory access violation

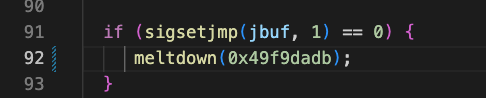


## Task 6

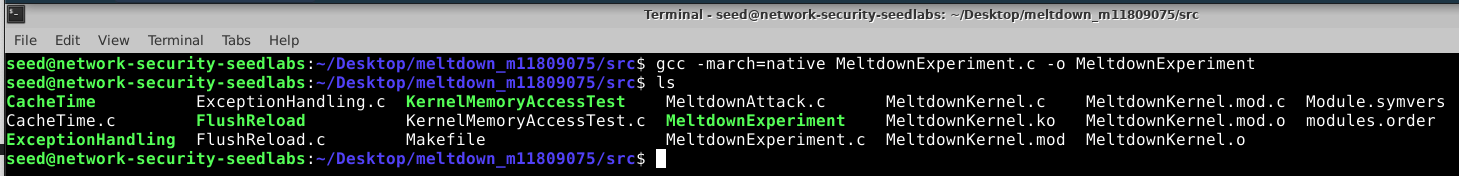
### How do you compile the code for this task

Update line 92 in *MeltdownExperiment.c* with the address found in Task 3

(0x49f9dadb in this case)



To compile, run `gcc -march=native MeltdownExperiment.c -o MeltdownExperiment`



### Screenshots

Execution is successful when CACHE\_HIT\_THRESHOLD is set to 80

A screenshot of a computer

Description automatically generated

When CACHE\_HIT\_THRESHOLD is set too low(10), we do not receive any results

Graphical user interface, text

Description automatically generated

When CACHE\_HIT\_THRESHOLD is set too high (203), we receive more than one result

A screenshot of a computer

Description automatically generated

## Task 7.1

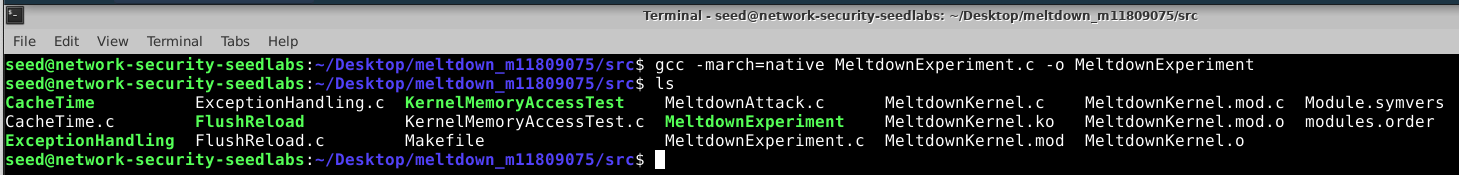
### How do you compile the code for this task

Make the change on line 53 of *MeltdownExperiment.c* as described in Task 7.1

Text

Description automatically generated

Compile using `gcc -march=native MeltdownExperiment.c -o MeltdownExperiment`



### Screenshots

Execution is unsuccessful despite multiple attempts

Graphical user interface, text

Description automatically generated

## Task 7.3

### How do you compile the code for this task

Update line 103 in *MeltdownExperiment.c* with the address found in Task 3

(0x49f9dadb in this case)

Graphical user interface, text

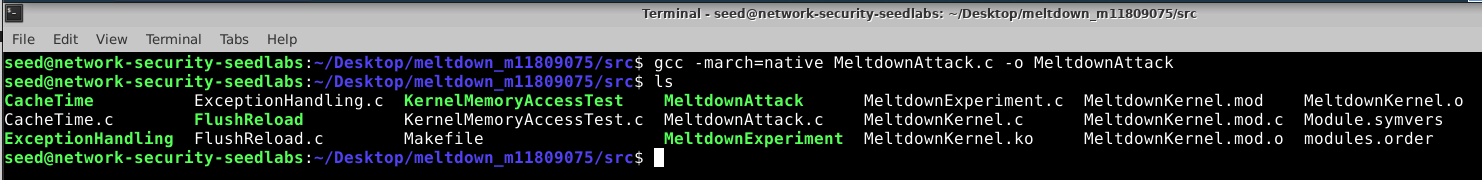
Description automatically generated

Make necessary changes shown in the following GitHub diff to try and steal more than 1 byte from kernel memory.

A screenshot of a computer screen

Description automatically generated

To compile, run `gcc -march=native MeltdownAttack.c -o MeltdownAttack`



### Screenshots

While the attack runs, it does not appear to succeed. I am not sure if this is due to my implementation being faulty, the countermeasure in the OS as described in the note from SEEDLabs in their updated version of this attack/document, or some sort of protection in Google Cloud’s VM infrastructure. When running their code unmodified, the secret value is reported as a 0, which does not correspond with the secret placed into memory in *MeltdownKernel.c.*

Text

Description automatically generated