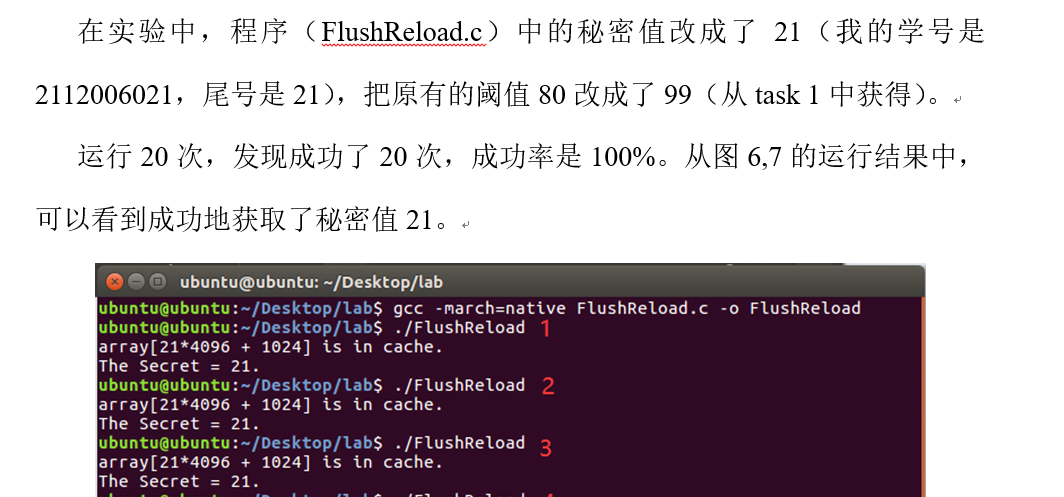
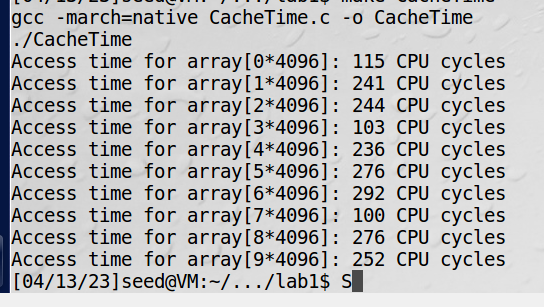
|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《网络与系统安全》 实验报告** |
|  |
| 实验一  Meltdown Attack 实验  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: | 梁鑫嵘 | | 学 号: | 200110619 | | 专 业: | 计算机专业 | | 日 期: | 2023年4月 | |

# 一、实验过程

每个实验步骤（共8个任务）要求有具体截图和说明，类似以下说明：

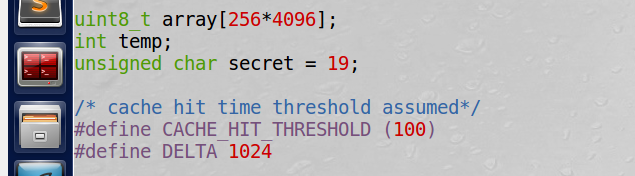


**任务1**

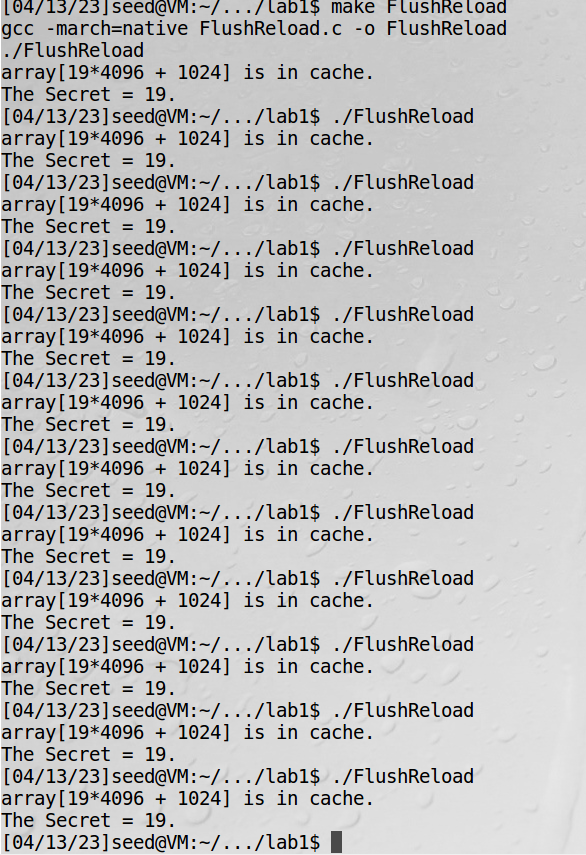
****

**任务2**

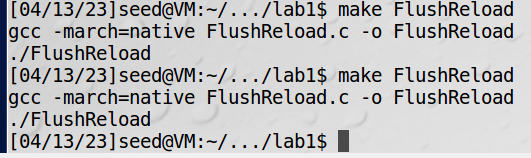
我的学号是200110619，尾号是19，所以令 secret=19。从Task1中得知，最小值为100，填入 CACHE\_HIT\_THRESHOLD。



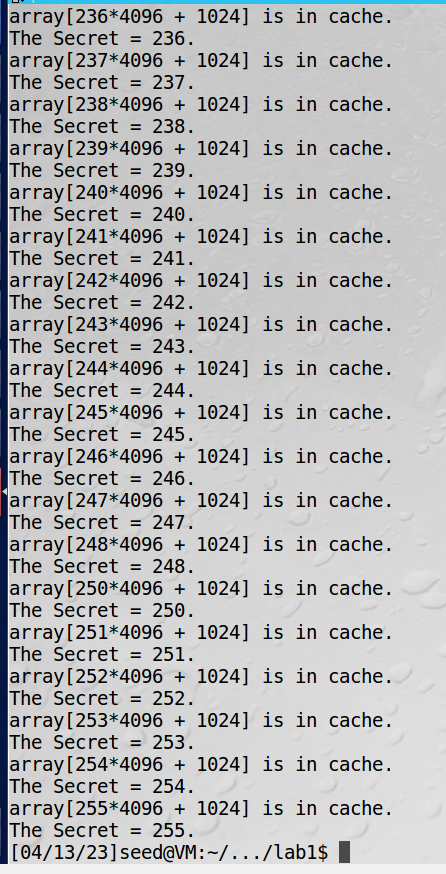
进行测信道攻击，得到 Secret=19，与代码中设置的相符。多次运行，这个Threshold可以稳定触发。



尝试将Threshold改小，例如20，则很可能不会有输出，即未检测到。

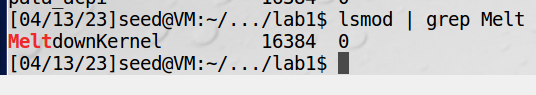


如果将Threshold改大，例如500，则会将过多的值算进来，如下：

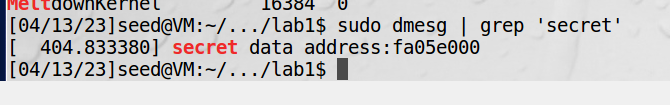


**任务3**

已经将MeltdownKernel.ko 加载入内核。

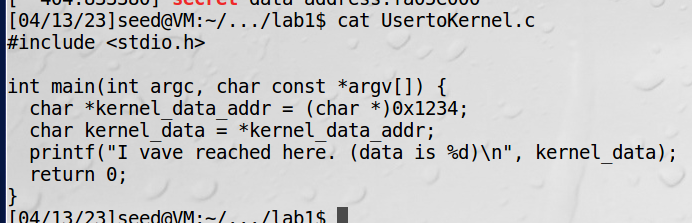


从dmesg中查找秘密数据所在内核地址：0xfa05e000

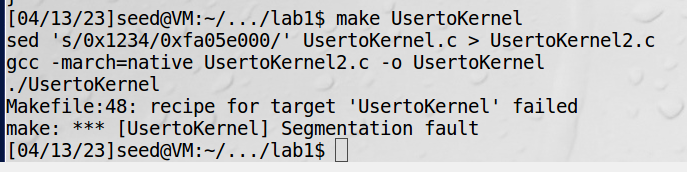


任务4

编写的UsertoKernel.c内容如下。其中0x1234会在Makefile中替换为实际内核地址。

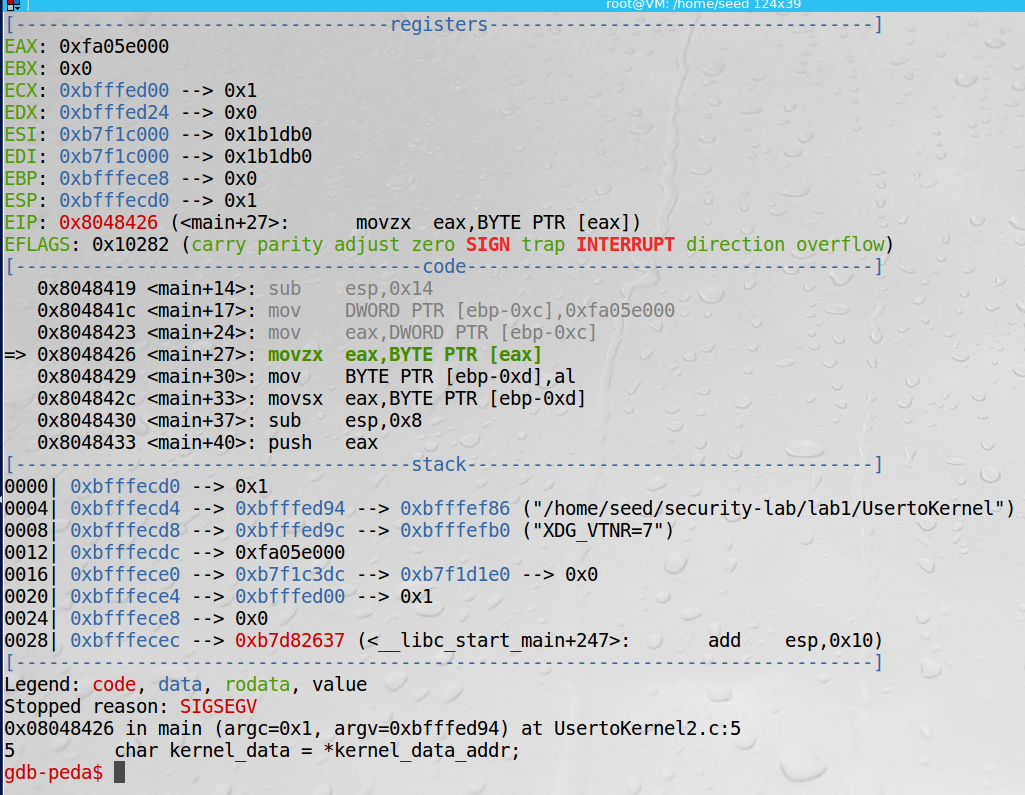


执行之，得到如下结果：



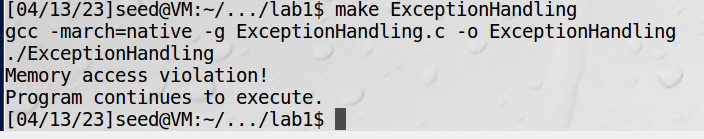
可以看到，在执行过程中程序收到 Segmentation fault 然后被系统杀死。

重新编译，在CFLAGS中添加-g来添加调试信息，使用GDB工具调试该程序。可以看到确实是在对一个内核地址解引用时（行5）收到Segmentation fault。



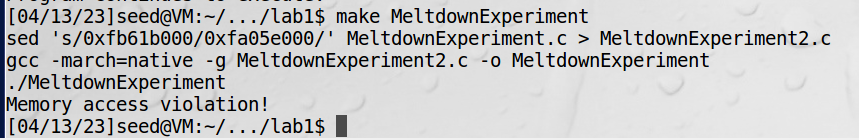
**任务5**

观察得知，程序在触发错误之后收到系统的Signal，设置了对应Signal的处理Handler之后，程序可以继续向下执行，也就是catch了Exception。



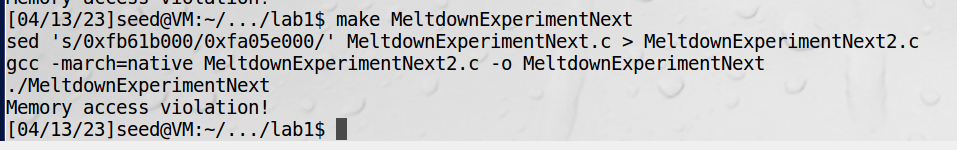
**任务6**

代码中的目标地址进行替换，然后编译执行。每一次执行的结果是基本一致的，都如下图所示，看起来没有访问到secret。



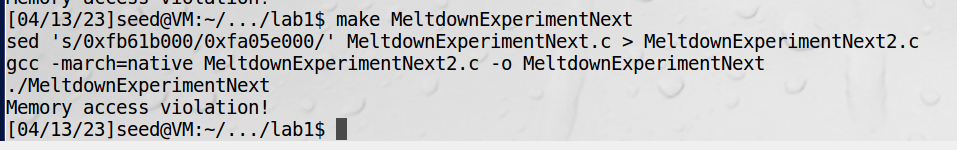
**任务7.1**

并不成功。



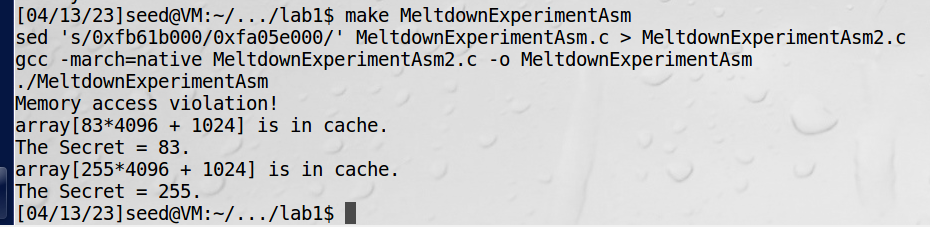
**任务7.2**

并不成功。

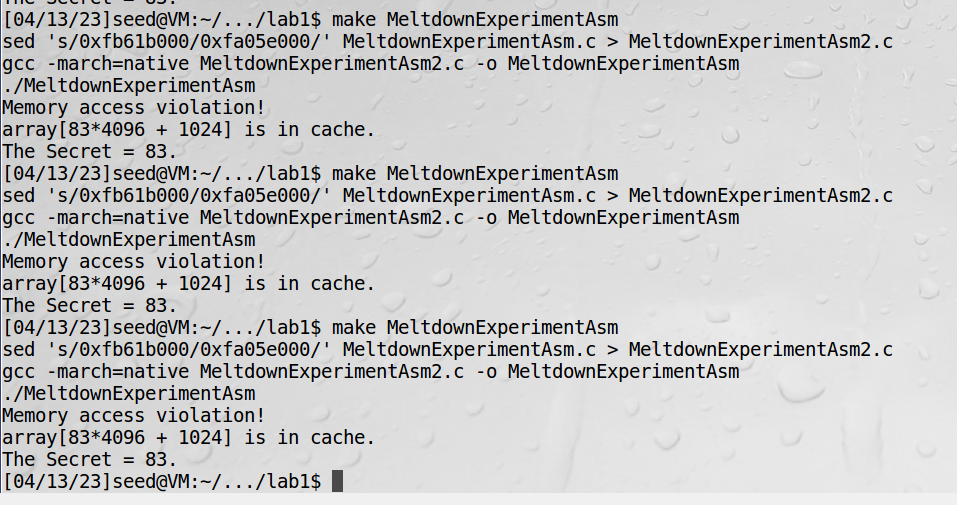


**任务7.3**

成功，这一次获取到了两个字节。



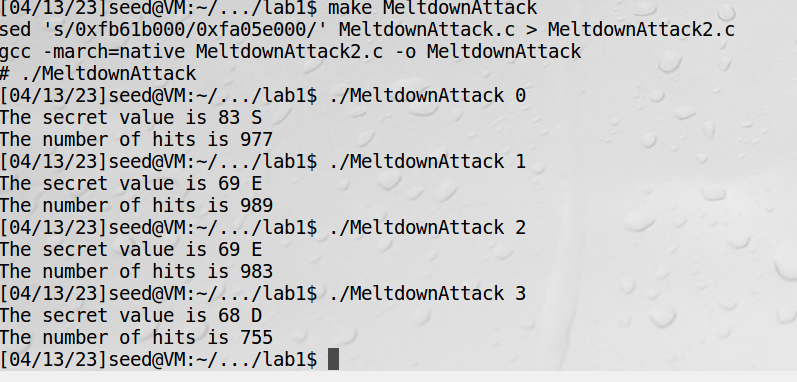
将循环次数从400增加到4000，得到的结果更加稳定了。



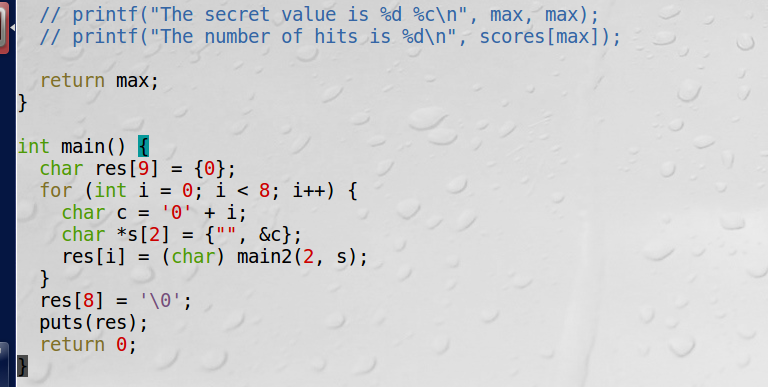
**任务8**

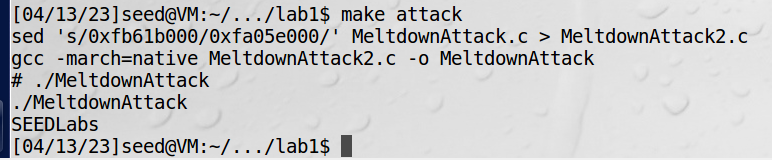
首先，修改asm部分的循环为4000，增强效果；然后给原来的MeltdownAttack.c添加一些参数，不断运行，测试是否能够稳定触发。





可以观察到对对应偏移数值稳定触发Meltdown攻击。然后再添加一个循环，将值记录为字符串，则得到结果。



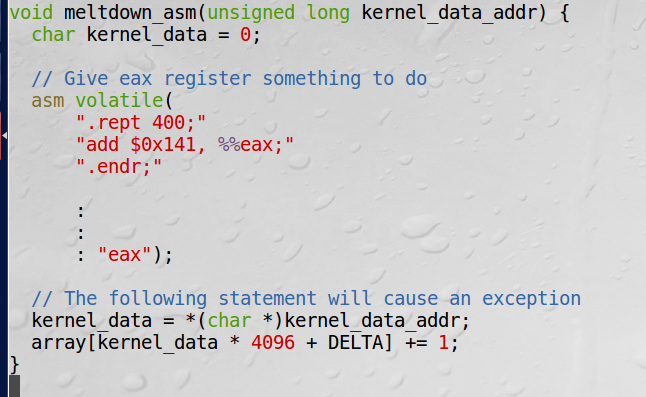


# 二、说明汇编代码在本次实验中的作用

即说明MeltdownExperiment.c 文件中下面函数的作用

void meltdown\_asm(unsigned long kernel\_data\_addr)

函数内容如下：



这一段ASM代码作用是不断地给EAX寄存器加0x141，循环400次。这一段代码能够让CPU的ALU在一段时间内一直保持忙，提升CPU的数据竞争状态，让后面的代码越过这段ASM先执行的概率更大，从而提高了Meltdown攻击的成功率。