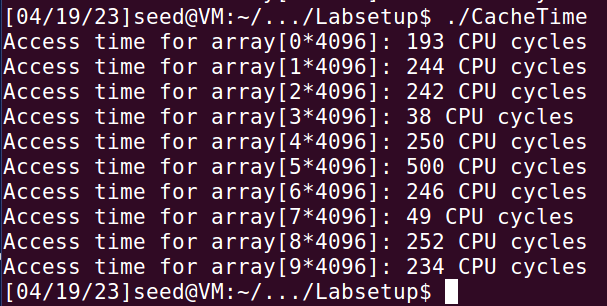
|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《网络与系统安全》 实验报告** |
|  |
| 实验一  Meltdown Attack 实验  学 院: 计算机科学与技术学院   |  |  | | --- | --- | | 姓 名: | 宗晴 | | 学 号: | 200110513 | | 专 业: | 计算机类 | | 日 期: | 2023年4月 | |

# 一、实验过程

任务1：分别测试从 Cache 读取和从内存读取数据的时间长度，对比时间差。

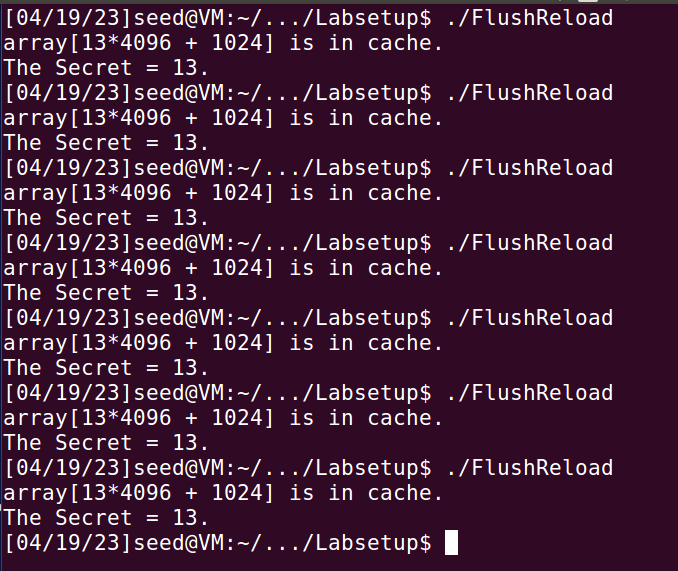
选取用于区分缓存和RAM这两种内存访问类型的时间阈值为107。

最后一次的运行结果截图：

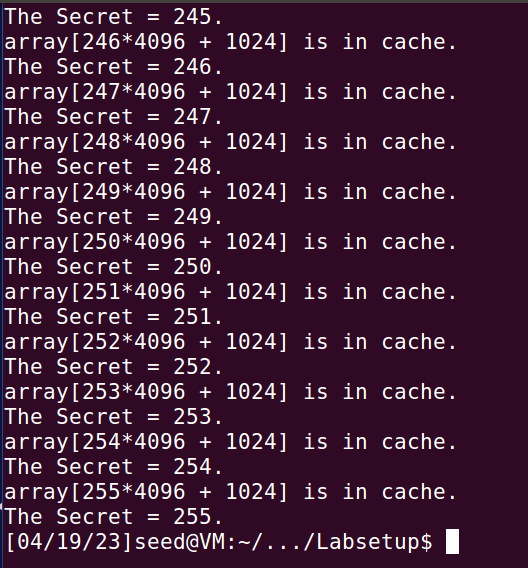


任务2：利用缓存进行侧信道攻击的示例

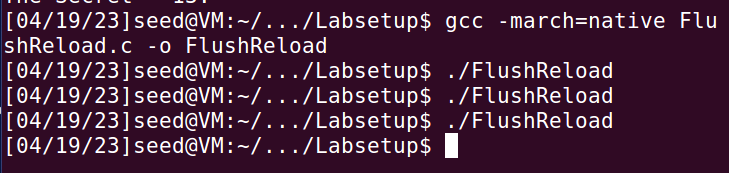
当阈值为任务1中选取的107时，成功的次数为20/20，能够获取到准确的秘密值。如下图所示：



将阈值修改为一个较大的值，如300，此时并不能获取到准确的秘密值，如下图所示：



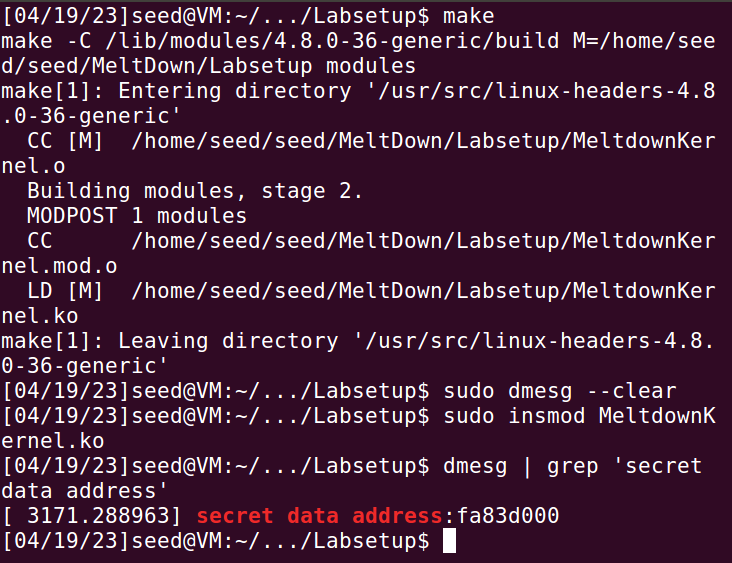
将阈值修改为一个较小的值，如20，此时也不能获取到准确的秘密值，如下图所示：



这是因为第二次和第三次所选取的阈值过大或过小，并不能起到区分缓存和RAM这两种内存访问类型的 。

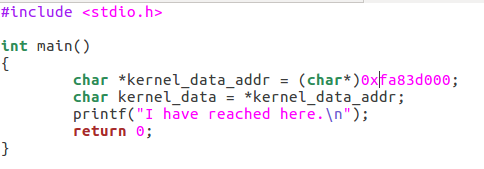
任务3：将秘密数据加载到内核空间

获取到秘密数据的地址如下图所示：

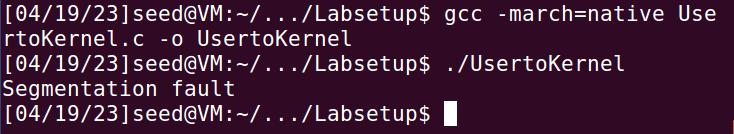


任务4：从用户空间访问内核内存

编写代码如下：



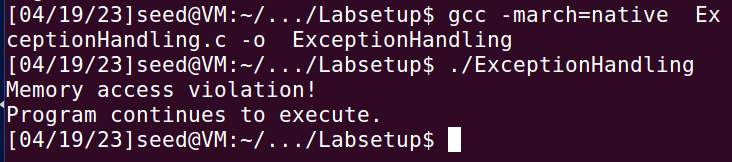
编译并运行这个程序：



程序运行后出现段错误。在line2程序不能成功执行，程序能执行到line2，但在line2处报错。

任务5：处理程序的错误/异常

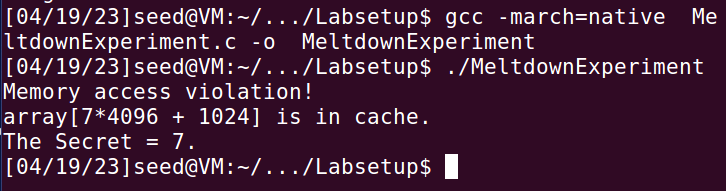
程序执行结果如下：



可以发现程序顺利完成了对 SIGSEGV 信号的处理并往下执行。

任务6：CPU的乱序执行

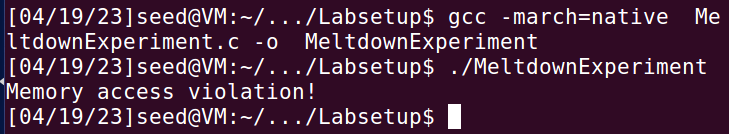
执行结果如下图所示：



说明程序成功地访问到了内核空间中的秘密数据。

任务7.1：观察Cache中是否加载到kernel\_data的值

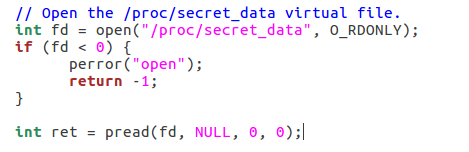
执行结果如下图所示：



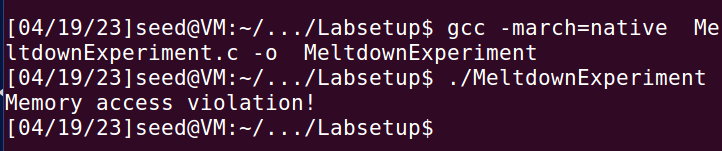
程序并不能访问到内核空间中的秘密数据。

任务7.2：通过加载内核缓存秘密数据实施攻击

在任务7.1的基础上添加如下代码：



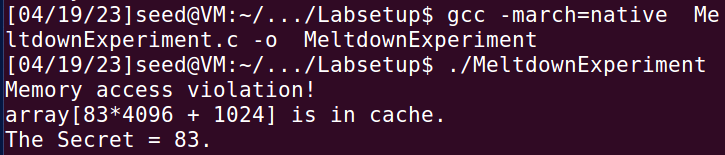
执行结果如下图所示：



程序仍不能访问到内核空间中的秘密数据。

任务7.3：使用汇编代码触发Meltdown攻击

调用meltdown\_asm（）函数，而不是原来的meltdown（）函数，执行结果如下图所示：

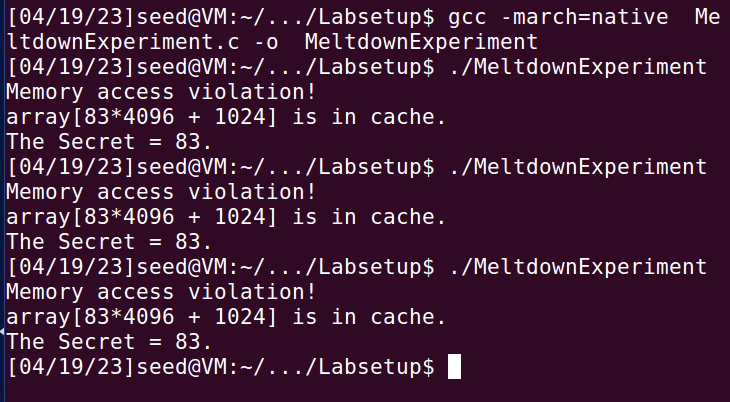


说明程序成功地访问到了内核空间中的秘密数据。

将循环次数改为10次后，成功概率降低：



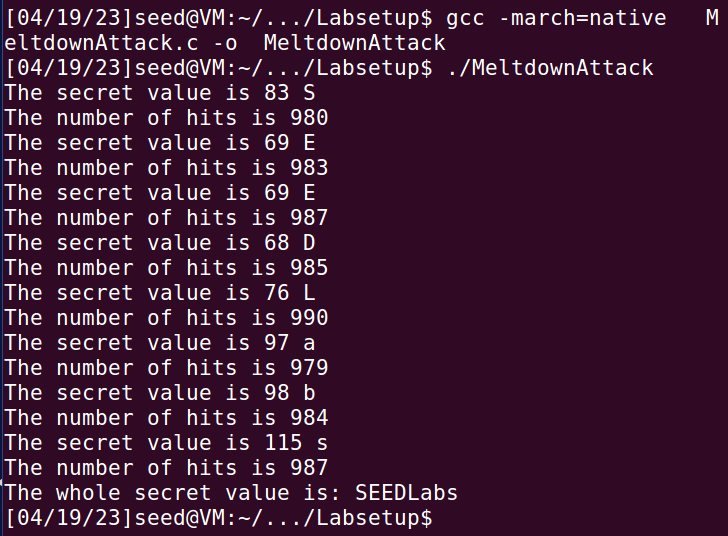
将循环次数改为1000次后，成功概率提高：



这是因为增加循环次数后，权限检查变慢，乱序后可以跑的代码更多。

任务8：优化完成更有效的 Meltdown 攻击

获取这8个秘密数据SEEDlabs，结果如下图所示：



# 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)

乱序后能跑多少后续的代码取决于权限检查的速度，权限检查越慢，乱序后可以跑的代码越多，这是一个竞态条件。而meltdown\_asm这个函数进行了大量的循环，虽然这段代码基本上在做无用的计算，但是这些额外的代码行“给了算法单元一些需要处理的内容，正在推测内存访问”，从而使得权限检查变慢，因而乱序后能够跑的代码就越多，因此更容易从内核中获取秘密数据。