Lab6 Report

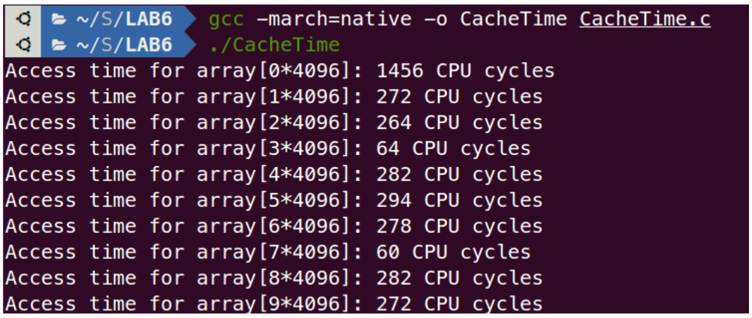
唐亚周 519021910804

1 Q1

> 完成上述实验，尝试输出secret中的第四个字符(D)；

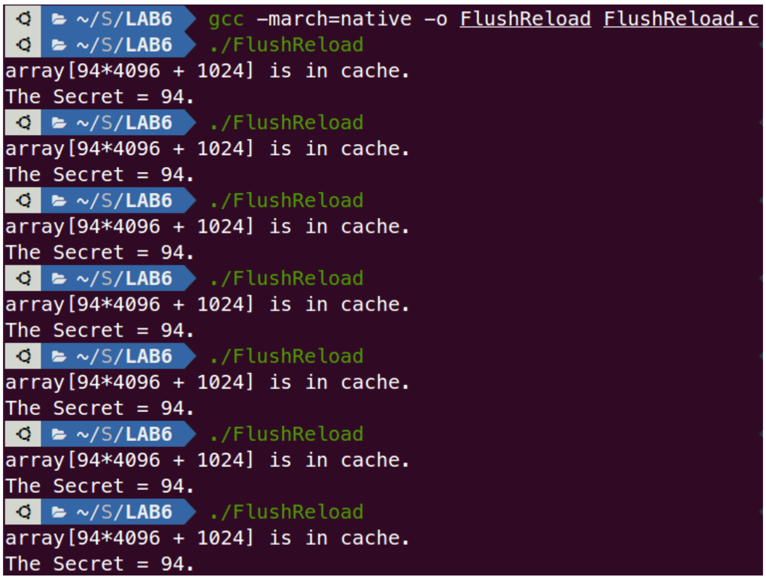
1.1 任务一：从内存和CPU缓存中读取数据

运行结果如图所示。多次测试后，我将任务二中的阈值设置为 200。



1.2 任务二：将CPU缓存作为SideChannel

运行结果如图所示。可以看到，几乎每次都成功判断到了 Cache Hit。



1.3 任务三：熔断攻击

1.3.1 step 1：将secret注入内核

运行结果如图所示。可以看到 secret 被成功注入到内核。

```
~/S/LAB6 make                                                    21:05:00
make -C /lib/modules/4.15.0-142-generic/build M=/mnt/hgfs/SJTU_Files/3-Junior-2
/IS308-Computer-System-Security/SysSec-labs/LAB6 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-142-generic'
  CC [M]  /mnt/hgfs/SJTU_Files/3-Junior-2/IS308-Computer-System-Security/SysSec
-labs/LAB6/MeltdownKernel.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /mnt/hgfs/SJTU_Files/3-Junior-2/I
S308-Computer-System-Security/SysSec-labs/LAB6/MeltdownKernel.o
see include/linux/module.h for more information
  CC      /mnt/hgfs/SJTU_Files/3-Junior-2/IS308-Computer-System-Security/SysSec
-labs/LAB6/MeltdownKernel.mod.o
  LD [M]  /mnt/hgfs/SJTU_Files/3-Junior-2/IS308-Computer-System-Security/SysSec
-labs/LAB6/MeltdownKernel.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-142-generic'
~/S/LAB6 sudo insmod MeltdownKernel.ko                           21:07:14
[sudo] password for student:
~/S/LAB6 sudo cat /proc/kallsyms | grep secret                   21:07:28
d73c9690 t dh_set_secret
d7845660 t addrconf_sysctl_stable_secret
d7d15884 d ts_secret
d7d15894 d net_secret
d7d16e80 d inet_ehash_secret.62499
d7d16ee4 d udp_ehash_secret.66569
d7d17000 d syncookie_secret
d7d17348 d udp_ipv6_hash_secret.65619
d7d1734c d udp6_ehash_secret.65618
d7d173a0 d syncookie6_secret
d7d17bf8 d ipv6_hash_secret.60312
d7d17bfc d inet6_ehash_secret.60311
d7f1d480 b ip4_frags_secret_interval_unused
d7f1fda4 b ip6_frags_secret_interval_unused
fd4bd0c0 r secret       [MeltdownKernel]
fd4be200 b secret_buffer        [MeltdownKernel]
f8a77c70 t compute_ecdh_secret  [bluetooth]
f8968980 t ecdh_set_secret      [ecdh_generic]
f8968470 t crypto_ecdh_shared_secret    [ecdh_generic]
```

之后尝试直接读取 secret，代码如下。[1]

```c
#include <stdio.h>

int main()
{
    char *kernel_data_addr = (char*)0xfb4bd0c0; // 替换为 secret 的地址
    char kernel_data = *kernel_data_addr;
    printf("I have reached here.\n");
    return 0;
}
```

运行结果如下，产生了 segmentation fault。

```
~/S/LAB6 gcc -o UserAccess UserAccess.c
~/S/LAB6 ./UserAccess
zsh: segmentation fault (core dumped)  ./UserAccess
```

1.3.2 step 2：实施攻击

运行结果如下，可以看到攻击失败了。

```
~/S/LAB6 gcc -march=native -o MeltdownAttack MeltdownAttack.c
~/S/LAB6 ./MeltdownAttack                                    21:27:02
The secret value is 0
The number of hits is 972
~/S/LAB6 ./MeltdownAttack                                    21:27:08
The secret value is 0
The number of hits is 995
~/S/LAB6 ./MeltdownAttack                                    21:27:21
The secret value is 0
The number of hits is 988
~/S/LAB6 ./MeltdownAttack                                    21:27:22
The secret value is 0
The number of hits is 994
```

检查我的电脑的 CPU 是否存在 Meltdown 的 bug[2]，发现并不存在。我的电脑的 CPU 为 11th Gen Intel i7-11800H，应该已经修复了该 bug。

```
~/S/LAB6 sudo cat /proc/cpuinfo | grep bugs                  21:42:34
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs itlb_multihit
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs itlb_multihit
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs itlb_multihit
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs itlb_multihit
```

2 Q2

> 解释使用UserAccess读取数据为何会产生Segmentation Fault；

答：因为 secret 在内核区域的内存上，不可以从用户空间直接访问。

3 Q3

> 解释实验中array的有效数字之间存在较大的间隔，为何在任务二和任务三中还需要增加一个偏移量Delta；

答[1]：因为 `array` 数组所在的内存的相邻区域上可能有其它的变量，当某个在 `array` 的“前面”的变量被缓存时，`array` 的前几个位置也可能因为处于同一个缓存块中，也一起被读入缓存中。因此我们不能使用 `array` 的前几个位置来进行测试。也就是说，如果我们不增加偏移量 Delta，那么测试 `array[0*4096]` 时，它可能已经被缓存了，这就造成了实验结果不准确。

而为了保证测试的一致性，我们就在每次取测试点时，都加上一个偏移量 Delta，因此我们使用 `array[k*4096+DELTA]` 作为测试点。

4 Q4

> 解释`meltdown_asm`中汇编指令的作用。（选做）

答[1]：这段汇编指令执行了 400 次将 `0x141` 这个值加到 `eax` 寄存器的行为。这实际上是无效的计算，但它的目的是通过占用 CPU 中的 ALU 来增加判断访问异常的时间，从而提高攻击的成功率。

---

1. https://seedsecuritylabs.org/Labs_16.04/PDF/Meltdown_Attack.pdf ↩ ↩ ↩

2. https://xuanxuanblingbling.github.io/ctf/pwn/2020/06/30/meltdown/ ↩