实验课程安排与考核标准

实验课程共8个学时，2个实验项目，总成绩为30分（30%）。

实验项目

项目编号	实验一	实验二
学时数	4	4
实验项目	Meltdown Attack	passwd实现细粒度访问控制及root能力安全使用
分数数	12	18

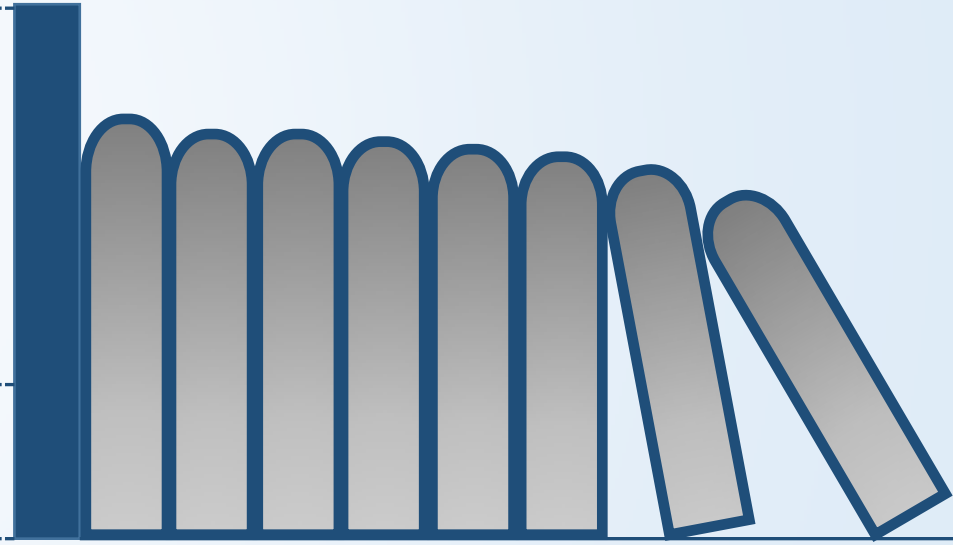
考核方式

- 课堂检查+ 实验报告
- 每次课程均需提交实验程序源代码，以及实验报告。
- 实验报告：参照提供的报告模板。

禁止抄袭，发现雷同，本次实验双方都是0分。

Meltdown 实验

2021年春



实验目的

- 了解Cache访问速度与RAM访问速度的差距
- 了解什么是FLUSH + RELOAD
- 掌握编译和安装内核模块的指令
- 学会如何处理SIGSEGV信号，使得程序能够继续执行
- 了解Intel CPU乱序执行的原理
- 掌握如何使用户级程序读取到存储在内核内存中的数据的方法

参考内容

- 实验内容地址

https://seedsecuritylabs.org/Labs_16.04/System/Meltdown_Attack/

- Meltdown and Spectre

https://meltdownattack.com/

- 15分钟读懂英特尔熔断幽灵漏洞

https://www.bilibili.com/video/av18144159?spm_id_from=333.788.b_765f64657363.1

- 侧信intel：spectre&Meltdown侧信道攻击（一）

https://www.cnblogs.com/theseventhson/p/13282921.html

- 一步一步理解CPU芯片漏洞：Meltdown与Spectre

https://www.freebuf.com/articles/system/159811.html

- Meltdown 是什么（对Meltdown论文的翻译，有些小错误，不影响阅读）

https://zhuanlan.zhihu.com/p/33621030

实验内容

本次实验分为8个步骤来完成Meltdown Attack的测试过程

- Step1: 测试分析从缓存读取和从内存读取的速度差异，找到一个临界点
- Step2: 根据Step1得到的临界值，使用Flush+Reload技术通过**缓存侧信道攻击**来提取受害者函数使用的秘密值
- Step3: 将机密数据放入内核空间
- Step4: 从用户空间访问内核内存
- Step5: 用C语言处理错误/异常
- Step6: 验证CPU的乱序执行
- Step7: 基本的Meltdown攻击（基础->缓存密码信息->汇编）
- Step8: 用更实际的方式完成Meltdown attack

实验原理

- 由于处理器的缓存（cache）机制，那些被预测执行或乱序执行的指令会被先加载到缓存中，但在处理器恢复状态时并不会恢复处理器缓存的内容。
- Meltdown会利用Intel处理器的**预测执行**设计，破坏应用程序和操作系统间的界限，攻击程序有机会访问到操作系统所使用的内存空间，也就有机会从中获取操作系统级别的数据。

乱序执行

侧信道攻击

实验原理

➢ 乱序执行

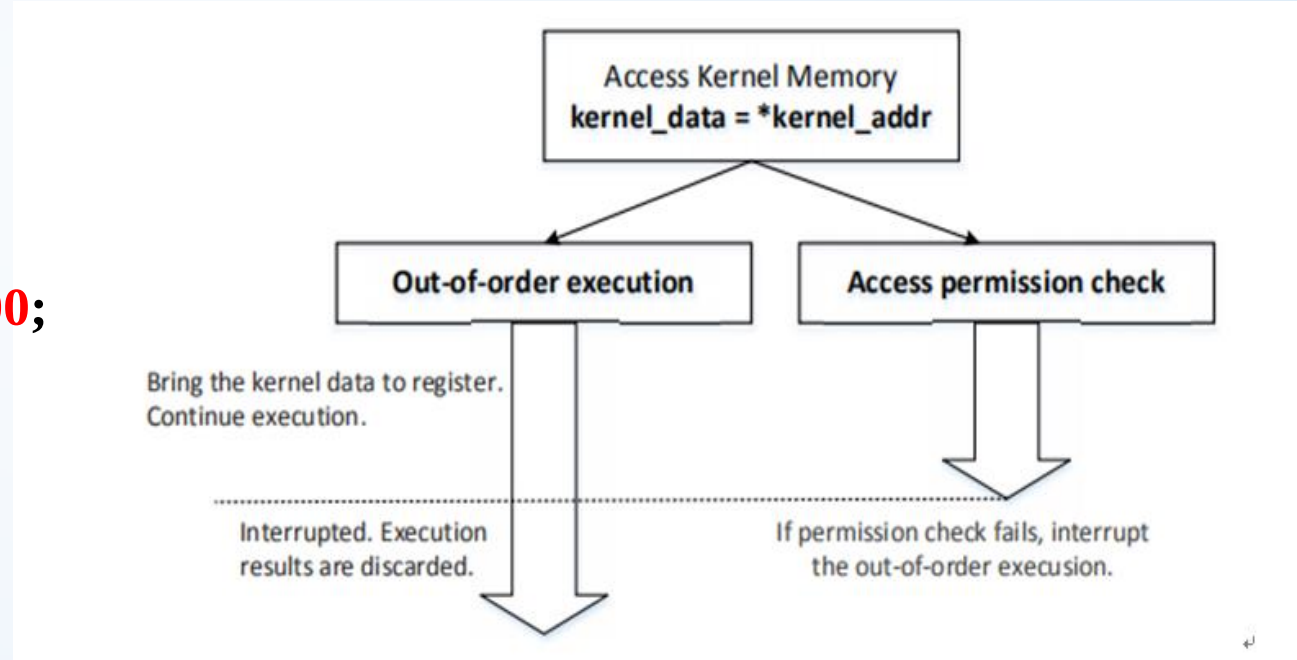
CPU遇到指令依赖的情况时，会转向下条不依赖的指令去执行。

乱序执行示例：

```
line1   number = 0;
line2   *kernel_address = (char*)0x fa59c000;
line3   kernel_data = *kernel_address;
line4   number = number + kernel_data;
```

注：0x fa59c000是内核地址

思考：line4在系统能执行了么？



实验原理

➢ 侧信道攻击

缓存通过数据共享来加快数据访问，也就是说缓存命中与失效对应的响应时间是有差别的，攻击者正是利用这种时间的差异性来推测缓存中的信息，从而获得隐私数据。

缓存侧信道攻击主要有Evict+Time、Prime+Probe与Flush+Reload等攻击方式，本次实验采用Flush+Reload技术。

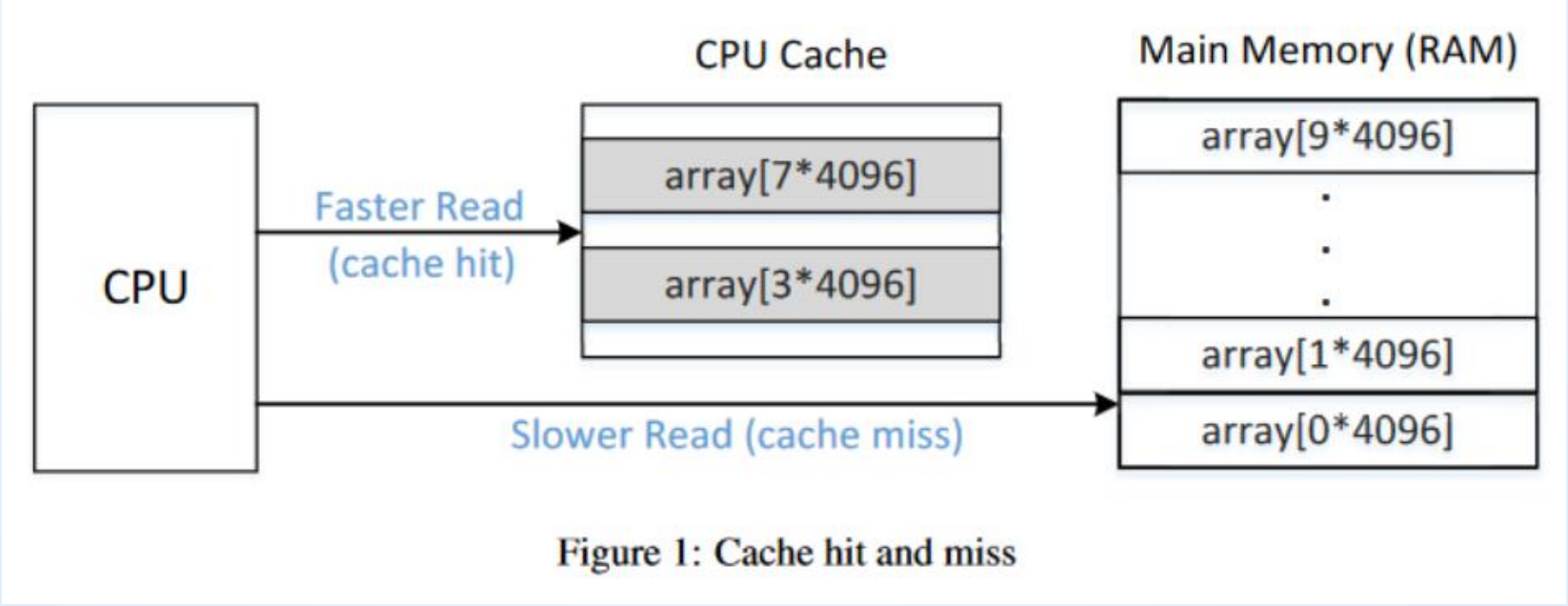


Figure 1: Cache hit and miss

实验原理

➢ FLUSH+RELOAD过程

- 从cache内存FLUSH所有数组，来保证数组没有被缓存到cache中。
- 唤醒缺陷函数。这个缺陷函数基于secret来访问数组的某个元素。这个行为导致相应的数组元素被缓存到cache。
- RELOAD整个数组，并且测量每个元素重载的时间。如果某个元素加载比较快，那么意味着这个元素之前就已经在cache中了。

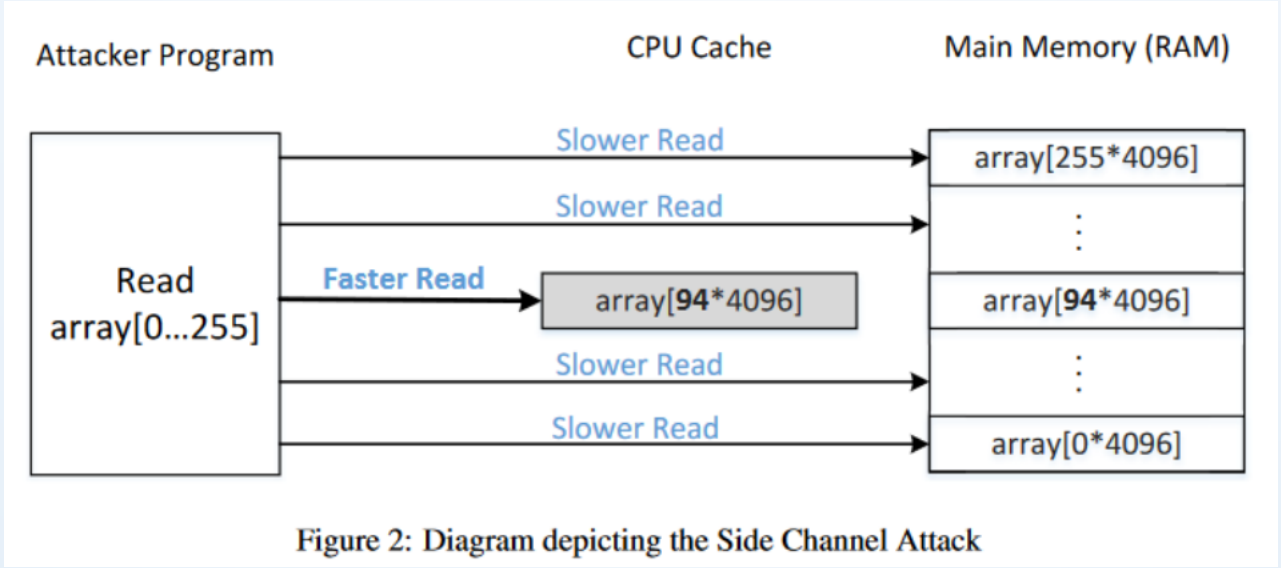


Figure 2: Diagram depicting the Side Channel Attack

实验原理

➢ 处理程序异常崩溃

如果一个程序试图读取内核内存，访问将失败并且将引发异常；需要程序中定义信号处理程序来捕获异常。C不提供对异常处理的直接支持，例如try/catch子句。通过模拟try/catch子句，使用sigsetjmp（）和siglongjmp（）来实现。

```
static sigjmp_buf jbuf;
static void catch_segv()
{
  // Roll back to the checkpoint set by sigsetjmp().
  siglongjmp(jbuf, 1);
}
int main()
{
  // The address of our secret data
  unsigned long kernel_data_addr = 0xfb61b000;
  // Register a signal handler
  signal(SIGSEGV, catch_segv);

  if (sigsetjmp(jbuf, 1) == 0) {
     // A SIGSEGV signal will be raised.
     char kernel_data = *(char*)kernel_data_addr;

     // The following statement will not be executed.
     printf("Kernel data at address %lu is: %c\n",
                          kernel_data_addr, kernel_data);
  }
  else {
     printf("Memory access violation!\n");
  }
```

注册信号

触发信号，回归到检查点

实验环境

Vmware 虚拟机Ubuntu16.04-seed

用户名密码:seed/dees

实验文件路径:

其中压缩文件件Meltdown_Attack为本次实验需要的部分源代码

实验内容

本次实验分为8个步骤来完成Meltdown Attack的测试过程

- Step1: 测试分析从缓存读取和从内存读取的速度差异，找到一个临界点
- Step2: 根据Step1得到的临界值，使用Flush+Reload技术通过**缓存侧信道攻击**来提取受害者函数使用的秘密值
- Step3: 将机密数据放入内核空间
- Step4: 从用户空间访问内核内存 ----需要自行编写代码
- Step5: 用C语言处理错误/异常
- Step6: 验证CPU的乱序执行
- Step7: 基本的Meltdown攻击（基础->缓存密码信息->汇编）------需要自行编写代码
- Step8: 用更实际的方式完成Meltdown attack

实验要求

➢ 课堂检查&课后提交

① 课堂现场检查完成情况（50%）

② 课后提交实验报告（50%）

➢ 截止时间

① 提交时间（2021-5-15 24:00）

② 平台链接 http://10.249.12.98:8000/#/login

用户名/密码：学号/学号

初次登录，请修改密码！

➢ 提交内容

① 将实验报告和代码打成zip包上传

② 以学号_姓名命名

谢谢