1. 漏洞介绍
2. 漏洞发现

英特尔已经披露了有关一类新的CPU投机执行漏洞的详细信息，这些漏洞统称为“ L1终端故障”，该漏洞可能发生在过去和当前的英特尔处理器上；诸如Meltdown，Rogue系统寄存器读取和“懒人FP状态还原”一样，当受影响的Intel微处理器推测超出未经许可的数据访问权限时，可能会出现“ L1终端故障”漏洞。通过在这些情况下继续推测，受影响的英特尔微处理器将暴露出新的攻击旁道。AMD 发表声明，表示它的处理器不受该漏洞影响，主要云服务供应商都已发表安全公告（AWS，Google Cloud，Azure 等）。

L1TF 是一种预测执行攻击，类似今年一月披露的 Spectre/Meltdown 漏洞，允许攻击者从 PC 或第三方云窃取敏感信息。英特尔的软件保护扩展(SGX)功能让程序可以在英特尔处理器上建立所谓的安全区域(secure enclave)。芯片的这部分区域被围了起来，运行计算机操作系统无法访问或更改的代码。即使恶意软件或另外的威胁危及主计算机，安全区域也为敏感数据提供了避风港。但是来自世界各地五个学术机构的一群研究人员发现，虽然 SGX 基本上可以击退 Spectre 和 Meltdown 攻击，但一种相关的攻击可以绕过它的防线。他们称之为 Foreshadow 漏洞。

这类攻击，虽然对个人电脑和服务器里的英特尔 CPU 都有效，但更多的是针对服务器进行。不过现在越来越多的安全服务在尝试把 SGX 用在消费级产品里，比如说密码存储软件 1Password、端到端加密的聊天软件 Signal 等。但安全研究员表示，黑客极低可能会利用该漏洞攻击某个特定的用户，一般做法是攻击整套基础设施。

1. 漏洞原理

预兆漏洞流程与熔毁漏洞相似，但 SGX 的安全机制使攻击流程多了两步：

一，即使利用乱序执行漏洞，SGX 飞地数据也无法从内存读取，必须预先加载到 L1 缓存才能绕过限制，这也是该漏洞被英特尔官方命名为 L1 Terminal Fault 的原因；

二，对指向 SGX 飞地的指针解引用会返回中止页（Abort Page）使得结果为 -1，而不像之前因访问内核空间而缺页（Page Fault）。为了绕开这个限制，需要调用 mprotect 函数将页表项的 Present 位设为无效，从而提前在传统页表检查时便抛出缺页。

1. 漏洞变体

Foreshadow 有两个变体版本，第一种设计用于提取英特尔 SGX 安全区保护的数据，第二种 Next Generation (NG) 影响虚拟机，虚拟机管理器程序，操作系统内核内存，系统管理模式内存，Foreshadow-NG 潜在可被用于读取 L1 缓存中的任何信息，读取运行在同一处理器其它线程上的虚拟机中储存的信息（因为超线程共享 L1 缓存）。最简单的解决方法显然是关闭超线程，但这无疑会带来巨大的性能开销。

1. 漏洞比较

Meltdown：乱序执行对异常的延时处理

Spectre：预测执行的误判

Foreshadow：预测执行攻击

2.

Meltdown：熔毁漏洞从处理器的微架构着手攻击，通过旁路攻击绕开对加密算法的理论分析，而利用其硬件实现泄露的信息来进行攻击。在熔断漏洞中，缓存充当了攻击的旁路，基于缓存的 Flush+Reload 攻击击溃了乱序执行的最后一道防线。Flush+Reload 攻击先清空缓存，再等待受害程序进行访存操作，然后通过数据访问的用时来判断某段数据在此期间是否被受害程序访问过。这样指令被称为暂态执行（Transient Execution），这也是幽灵系列漏洞的核心技术。通过暂态执行和缓存旁路攻击能够提取物理内存中的所有数据，危害性极强。

Spectre：幽灵漏洞的核心也是暂态执行，暂态执行除了前面叙述的乱序执行之外还有其他的触发方式，均与分支预测有关。因为分支指令可能涉及内存读取，需要上百个时钟周期才能完成，因此现代处理器都设计了分支预测器来预先推测执行。一个解耦的分支预测器通常包含两个部分：分支方向预测器和目标地址预测器。使其运行目标代码时会进行特定的预测执行；同时可以把条件判断所需的数据挤出缓存，以提高预测执行发生的概率。

Foreshadow：预兆漏洞的步骤与熔毁漏洞类似，但利用乱序执行漏洞，使得数据无法从内存读取，必须预先加载到缓存。对指向 SGX 飞地的指针解引用返回中止页（Abort Page）使得结果为 -1，调用 mprotect 函数将页表项的 Present 位设为无效，从而提前在传统页表检查时便抛出缺页。

1. 漏洞复现

这是CVE-2018-3646的PoC。 此漏洞使恶意/受损VM guest虚拟机能够读取主机物理内存。在支持VT-x和EPT（extended page tables）的大多数Intel CPU上，可以利用此漏洞。这包括所有Intel Core iX CPU。此PoC仅在64位x86-64系统（主机和客户机）上工作。

目前，对该漏洞的缓解应该由VMM完成。因此，如果VMM不是最新的，打过补丁的主机系统无法保护客户虚拟机免受损害。

实验环境：

|  |  |
| --- | --- |
| CPU | Intel Core i5-8250U |
| Host System | Ubuntu 18.04.1, kernel: Ubuntu kernel 4.15.0-34 from Aug 27 (has patch for L1TF) |
| VMM | VMware Workstation player 14.1.2 build-8497320 (one version prior to the newest patched version) |
| Guest system | Same as host |

Poc工作原理

授予对guest VM的root访问权限，PoC使用/ dev / mem来执行以下操作：

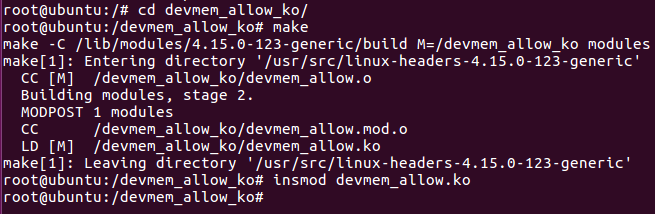
1. 将一个页面映射到一个dummy (magic) physical address
2. 彻底搜索guest VM的整个内存，以便找到映射magic address的PTE
3. 用标记为不存在的目标主机physical address的特殊(L1TF)条目重写magic address PTE

以上所有操作都是为了仅使用用户空间代码进行攻击。

那时，PoC将运行代码，以推测方式访问“不存在”页面，并通过cache timing side channel泄漏获取的信息。

该内核模块仅从运行的内核中修补“ devmem\_is\_allowed”功能，使/ dev / mem可以使用。

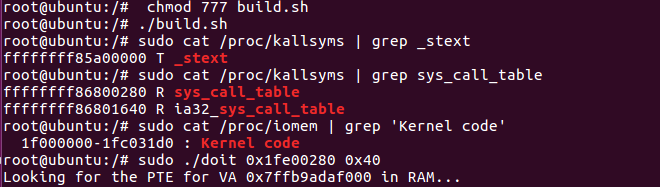
用法：



（这里mkdir了devmem\_allow\_ko目录，在里面创建了devmeme\_allow.c和Makefile文件

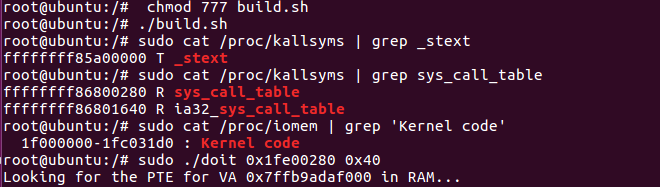
在上一级目录创建了doit.c，doit.S，phys.c，build.sh）

以root身份登录到客户机里，使用以下方法构建代码



假定数据被保存在L1 cache中，这将转储给定主机物理地址上的内容！ 如果VM和主机进程在同一CPU内核上运行，则L1缓存在VM和主机进程之间共享。

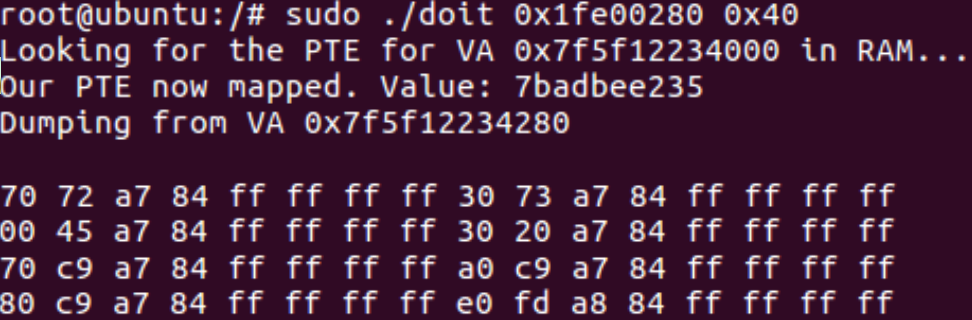
例如，在主机上，我们这样获取内核的物理地址



要获取物理地址，计算公式为：

ffffffff86800280-ffffffff85a00000+1f000000=1fe00280

有了物理地址，我们现在就可以在客户机上使用PoC读取它]JFEWK(NB`W3~NSZ4OXU@EO



攻击受到以下因素的限制：  
1. 数据需要在L1缓存中(只有32KB)，以便在采样一个字节时读取数据。

1. 每个CPU核都有一个单独的L1缓存(在两个超线程之间共享)。攻击者进程必须运行在正确的核上。
2. 物理地址随机化使得攻击者更难找到目标数据。

对于此攻击向量，主机物理地址随机化的局限性似乎并不十分重要，因为在主机内核物理地址的熵约为14位时，可以通过16k次读取尝试（基于此PoC大约10-20分钟）找到非常容易识别的sys\_call\_table。 从那里可以解析主机内核中的所有地址。

参考资料：

L1TF (Foreshadow) VM guest to host memory read PoC (https://github.com/gregvish/l1tf-poc)