



针对 KASLR 的 Linux 计时攻击方法

丛 眸¹, 张 平², 王 宁³

(1. 长春理工大学 计算机科学技术学院, 长春 130022; 2. 陆军装甲兵学院 北京 100072; 3. 公安部第三研究所, 北京 100142)

摘 要: 针对开启内核地址空间布局随机化(KASLR)防护的 Linux 系统, 提出一种基于 CPU 预取指令的 Cache 计时攻击方法。Intel CPU 的预取指令在预取未映射到物理地址的数据时会发生 Cache 失效, 导致消耗的 CPU 时钟周期比已映射到物理地址的数据要长。根据这一特点, 通过 rdtscp 指令获取 CPU 时钟周期消耗, 利用计时攻击绕过 KASLR 技术防护, 从而准确获取内核地址映射的 Offset。实验结果表明, 该攻击方法能够绕过 Linux 操作系统的 KASLR 防护, 获得准确的内核地址映射位置, 并且避免引起大量 Cache 失效。

关键词: 内核地址空间布局随机化; 预取指令; 计时攻击; 内核; Cache 失效

开放科学(资源服务)标志码(OSID):



中文引用格式: 丛眸, 张平, 王宁. 针对 KASLR 的 Linux 计时攻击方法[J]. 计算机工程, 2021, 47(8): 177-182.

英文引用格式: CONG M, ZHANG P, WANG N. Method of timing attack for Linux against KASLR [J]. Computer Engineering, 2021, 47(8): 177-182.

Method of Timing Attack for Linux Against KASLR

CONG Mou¹, ZHANG Ping², WANG NING³

(1. Institute of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China;

2. Army Academy of Armored Forces, Beijing 100072, China;

3. The Third Research Institute of Ministry of Public Security, Beijing 100142, China)

[Abstract] For Linux systems with Kernel Address Space Layout Randomization (KASLR) protection, this paper proposes a Cache instant attack method based on CPU prefetch instruction. When the prefetch instructions of Intel CPU prefetch data that is not mapped to physical address, a Cache failure will occur, resulting in the consumption of CPU clock cycles longer than the data mapped to physical address. According to this feature, the CPU clock cycle consumption is obtained by using the rdtscp instruction, and the protection of KASLR technology is bypassed by using timing attacks, so as to accurately obtain the Offset of kernel address mapping. Experimental results show that this attack method can bypass the KASLR protection of Linux operating system to obtain the accurate mapping location of kernel address, and avoid causing a large number of Cache failures.

[Key words] Kernel Address Space Layout Randomization (KASLR); prefetch instruction; timing attack; kernel; Cache miss
DOI: 10. 19678/j. issn. 1000-3428. 0058582

0 概述

计算机技术的迅速发展在提高人类生产力的同时也给个人信息安全带来挑战。在计算机硬件设计上, 由于硬件的自身设计缺陷而造成的安全问题颇受学术界关注, 并且此问题从软件层面难以彻底解决, 从而致使个人信息泄露。侧信道分析^[1]就是一种利用计算机物理缺陷的攻击方法, 其通过分析计算机元器件泄露的电磁、电流、声音、时间等信息来获取计算机的私密信息。这种攻击方式彻底颠覆了传

统攻击理念, 并且很难对其进行防御。

Cache 计时攻击是一种侧信道分析^[1]方法。根据计算机存储系统中多级层次存储器的结构, Cache 的读写速度远高于内存。设计 Cache 是为了减少 CPU 访问主存的次数, 从而加快计算机的运算速度。所有进程在运行时都会使用到 Cache, 同时 CPU 也会从 Cache 中读取指令和数据进行运算。虽然进程之间相互隔离, 但是从硬件的角度分析, 不同的进程使用同一个 Cache, 因此, Cache 存在泄漏进程信息的可能。此外, 在 CPU 中的两条进程, 其中一条进程

基金项目: 装备预研领域基金(61400010301)。

作者简介: 丛 眸(1999—), 女, 硕士研究生, 主研方向为网络安全、计算机视觉; 张 平, 副教授; 王 宁, 助理研究员。

收稿日期: 2020-06-19 修回日期: 2020-08-15 E-mail: 1477342726@qq.com

能够通过 Cache 泄漏的信息窥探另一条进程的数据。Cache 计时攻击^[2-3]就是利用 Cache 信息泄露,通过侧信道分析方式非法获取其他进程数据的一种攻击方式。

侧信道分析是目前信息安全领域最主要的安全威胁之一,如2018年爆发的处理器A级漏洞Meltdown(熔断)^[4]和Spectre(幽灵)^[5]均对个人信息安全带来了极大的挑战,利用这种攻击方式不仅对个人计算机造成威胁,同时还对云服务商的安全问题提出了新的要求。虽然能以软件的方式给系统打补丁,但是基于这2个漏洞的变种仍然可以实现攻击,以打补丁的方式不仅不能完全封堵漏洞,而且还会损失芯片5%~30%的性能作为代价。造成这两个漏洞的本质是利用Cache这个共享部件,通过使用Cache计时攻击来获取系统内核内容和其他进程信息。

在Linux操作系统中使用内核地址空间布局随机化(Kernel Address Space Layout Randomization, KASLR)防护技术是为了提高系统安全性。在没有使用KASLR技术时,系统会把低位的虚拟地址默认分配给内核的物理地址,在这种情况下,内核的存放位置几乎对攻击者是透明的。而使用KASLR技术将内核的物理地址随机映射到虚拟地址中,就能保护内核映射情况不被攻击者知晓,从而提高系统的安全性。在内核地址映射到系统虚拟地址的过程中,不可避免地要使用Cache,这就使Cache计时攻击获取内核地址的偏移位成为可能。文献[6]利用侧信道分析对分支目标缓冲区(Branch Target Buffer, BTB)进行攻击,突破了KASLR防护,获取了内核的地址和受害者进程信息。文献[7]利用侧信道分析对内存管理系统进行攻击,绕过KASLR并获取系统特权,证实了该方法在X86 CPU实现的可行性,并且适用于虚拟机,会给云服务带来安全威胁。文献[8]利用CPU预测执行和乱序执行的漏洞,使用Cache计时攻击突破KASLR防护并破解密码算法等私密信息。

本文根据CPU在处理数据时会预取指令的特点,将数据存放在Cache中,利用Cache计时攻击对内核地址偏移位进行探测,获取内核地址的虚拟地址偏移位,从而突破Linux系统的KASLR技术防护,将内核地址暴露在普通用户层面,致使系统处于高风险状态。

1 基础知识

1.1 Cache 计时攻击原理

计时攻击的原理主要是根据访问数据时间的长短,并利用计算机硬件设计的缺陷来实现攻击。Cache的设计目的是为了解决CPU运算速度和内存访问速度不对等的问题,但是Cache具有共享特性,

容易造成线程数据泄露^[9]。Cache计时信息可分为多种,采集部件容量越小,攻击准确度越高。传统的Cache计时攻击需要将整个Cache清空来采集计时信息,容易受到其他进程影响,采集的信息噪声大,导致在分析阶段非常困难,攻击效果较差。之后研究者对Cache计时攻击提出了多种改进方法,如将采集部件由整个Cache改进为一个Cache组,甚至可为一个Cache行,使计时攻击采集的计时信息更准确,其他进程影响逐渐降低,从而提高攻击准确度,并且降低分析阶段的难度。例如Evict+Time^[10]、Prime+Probe^[11]、Flush+Reload^[12-14]、Evict+Reload^[15-16]、Flush+Flush^[17]攻击模型,只需要清空一个具体的Cache组或者一个Cache行,而不再需要清空整个Cache来采集计时信息。2018年1月爆出的Intel CPU设计漏洞,主要原因仍在于Cache泄露的计时信息,攻击者通过改进的计时攻击模型对Cache进行攻击来获取用户私密信息。

Cache计时攻击可分为基于时序驱动、基于访问驱动和基于踪迹驱动^[18]的攻击,本文以基于访问驱动模型的攻击为研究对象。模型的实现需要以下3点假设:

- 1) 受害计算机中存在攻击者部署的间谍进程(Spy Process)。
- 2) 攻击者能够获取被攻击进程的数据在Cache中的映射关系,并且能够清除被攻击进程在Cache中的数据。
- 3) 攻击者能够通过计时手段来判断访问的数据是否在Cache中。

实现攻击的难点在于第3步,通过计时手段来判断数据是否加载到Cache中,在Intel处理器中,能够使用rdtsc或rdtscp指令来获取CPU时钟周期为精度的计时。图1所示为Intel i7-6700 CPU中访问一次Cache数据命中与失效的时间,从中能够明显地区分访问命中和失效,Cache计时攻击就是通过这种时间的微妙差异来实现攻击。

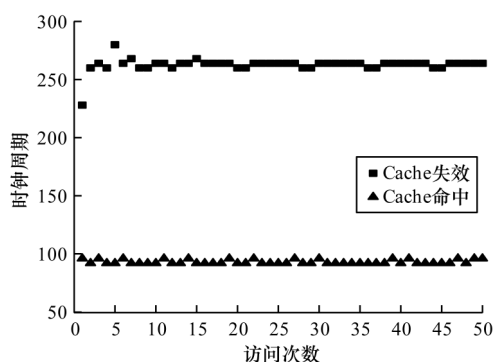


图1 访问一次数据所需要的时间

Fig.1 Time spent on accessing data once

由于操作系统的其他程序进程也需要访问 Cache,在探测内核地址的同时,难以阻止其他程序进程访问攻击地址,一旦攻击地址被其他进程访问,攻击就会受到干扰,容易对结果造成影响,因此 Cache 计时攻击在分析阶段时,就需要先排除噪声等影响才能获取正确的数据。

1.2 KASLR 技术

KASLR 是一种保持计算机内核地址的隐蔽性以防受到缓冲区溢出等攻击的计算机安全技术。如果内核地址映射到一个固定的地址,攻击者能够在内存中跳转到特定的函数位置以执行非法程序,获取受害者的私密数据。2001 年,KASLR 技术首次被设计实现,到 2002 年 10 月,KASLR 技术被应用于 Linux 内核地址随机化实现过程中,相比于其他实现方式,KASLR 技术还提供了更多的熵,能够增加攻击者预测目标地址的困难性并阻碍攻击。

如果没有使用 KASLR 技术,Kernel image 将会按照 vmlinux 链接脚本中的链接地址去映射虚拟地址,而使用 KASLR 技术,则会将 Kernel Image 再次映射到虚拟地址中,此次映射将会按照链接地址+offset 的新地址。由于 Offset 会随着计算机的每次开机而随机变化,因此 Kernel Image 映射到虚拟地址每次都会不同,从而实现内核地址随机化。

KASLR 技术能够有效防止系统内核受到攻击,内核负责进程、内存管理、硬件 I/O 等操作,没有 root 权限的用户不会影响到内核程序的正常运行,从而能够确保内核运行处于一个安全稳定的状态。内核空间是共享的,如果 KASLR 技术被突破,将直接暴露内核地址,使攻击更为简单,通过攻击能够使普通用户提权,监控内核中运行的程序,直接获取用户输入的账户、银行卡号、手机号等秘密信息,并且控制内核后还能直接破坏系统的正常运行,删除、更改、破坏用户重要数据,给社会安全带来严重的隐患。

1.3 CPU 数据预取

CPU 的发展给计算机带来了巨大的变革,但硬盘的存储速度与 CPU 处理速度存在巨大的差距,传统硬盘无法满足 CPU 的处理速度,从而催生 Cache 存储,以弥补存储速度之间的差距^[19]。虽然 Cache 能够弥补存储速度之间的差异,但是 CPU 访问数据发生一次 Cache 失效也会带来巨大的开销。为提高 CPU 的性能,数据预取技术被提出,其在可能会发生 Cache 失效之前提前将数据加载到 Cache 中,以避免发生 Cache 失效,从而提高 CPU 性能,并且预取指令只对数据有效,而对指令预取无效。

Cache 预取实现方式可分为硬件实现和软件实现^[20]。硬件实现是通过专门的硬件来监控程序正在执行的指令或处理的数据、预测程序下步需要的数据或指令并提前预取到处理器中;软件实现是利用编译器对代码进行分析,通过在程序编译时往代码段中插入 prefetch 指令,在执行过程中以触发数据或指令预取操作。软件实现的优点是不会阻塞内存操作,并且不会改变 CPU 状态或导致页错误,缺点是预取指令同样需要消耗 CPU 时钟周期。本文主要以预取数据软件实现为研究对象,在 Intel 处理器中,软件实现的预取指令主要有 prefetcht0、prefetcht1、prefetcht2、prefetchnta 和 prefetchw,这 5 种预取指令的特性如表 1 所示。

表 1 预取指令的特性

Table 1 The characteristics of prefetch instructions	
预取指令	特性
prefetcht0	预取数据到所有级别 Cache 中
prefetcht1	预取数据到 L2、L3 Cache 中,但不到 L1 Cache 中
prefetcht2	仅预取数据到 L3 Cache 中
prefetchnta	非临时预取数据,最小化 Cache 污染
prefetchw	预取数据到 Cache 中等待写入

一旦数据被预取到 Cache 中,就能够避免发生 Cache 失效,从而提高 CPU 的性能,但另一方面,内核地址映射预取到 Cache 中,能够通过计时攻击得到内核地址映射的 offset,从而可以更容易地触发其他类型的攻击。

2 本文攻击方法的原理与实现

2.1 攻击原理

与 Flush+Reload 攻击模型不同,本文使用的 Cache 计时攻击不需要重新加载数据和监控其他进程的信息,并且不需要任何特权,而只需要简单地从 Cache 中驱逐指定虚拟地址的数据,然后再次计算预取该地址数据的时间。如图 2(a)所示,普通程序分配的虚拟地址数据被加载到 Cache 中,同样地,随机化的内核地址数据也被加载到 Cache 中。由于没有 root 权限,使用简单的汇编指令没有权限将内核地址数据从 Cache 中驱逐。本文方法利用这一点对 KASLR 技术的内核起始位置进行探测,如图 2(b)所示。当能够明确无法驱逐内核地址数据时,表明内核地址数据依然存在于 Cache 中,如图 2(c)所示。再次预取指定数据时,通过使用 rdtsc 或 rdtscp 指令对预取数据进行计时,得到预取数据所消耗的时间,对所消耗的时间进行分析,则能得到内核地址的起始地址。

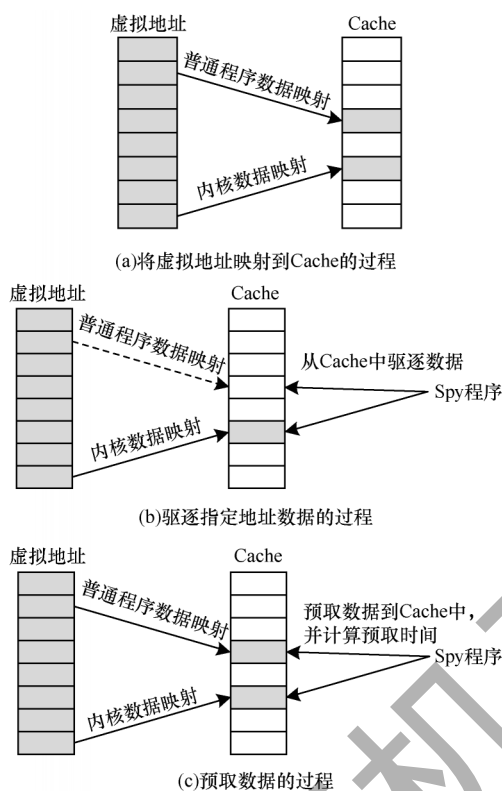


图2 本文Cache计时攻击模型

Fig.2 The proposed Cache timing attack model

2.2 方法实现

笔者通过对Intel手册进行分析发现,在Intel CPU中,预取“未映射到物理页面的地址”会导致不确定的性能损失。对此,可以利用预取指令时所需要的时间消耗来探测内核映射的起始地址,使得Cache计时攻击成为可能。

首先将指定地址数据从Cache中驱逐,保证再次预取普通程序数据时,该数据是未缓存到Cache中的,然后使用rdtscp指令对预取数据进行计时。本文方法中预取指令使用prefetcht2,因为在Intel CPU多级Cache存储结构中,只需要使用prefetcht2指令将数据预取到L3 Cache中就能达到攻击效果,不会对Cache造成污染,同时能够提高攻击效率。攻击Linux内核地址的步骤如下:

步骤1 将数据从Cache中驱逐。

步骤2 触发prefetcht2指令并计算指令消耗周期。

步骤3 使用rdtscp指令计算预取每一行地址数据的时间消耗。

对于开启KASLR防护的Linux操作系统,无法猜测内核的起始地址,只有通过遍历内存地址来探测每一个可能的起始地址。一个Cache行的大小为64 Byte,在攻击过程中,传入地址的递增梯度为64 Byte,然后依次循环探测地址,直到找到内核地址偏移位为止。攻击过程中需要探测每一个内存地址,如果按照64 Byte的递增梯度,攻击的精度能精

确到Cache行,但是这种方式将会消耗大量的时间,在分析prefetcht2预取指令时,预取指令预取数据时可以只加载一个Cache行,也可以设置加载一个内存页到Cache中,并且KASLR在映射内核地址时会4K对齐。在攻击实现时,本文将预取数据的大小设置为一个内存页,如果梯度仍设为64 Byte,在一个内存页的范围内预取数据消耗的时间都会很短,从而造成不必要的地址探测。鉴于此,攻击实现以一个内存页大小为递增梯度,以加快探测内核地址偏移位的速度,在Linux操作系统中,默认的内存页大小为4 KB,一个内存页的数据需要64个Cache行来存储,使用内存页作为递增梯度不仅不会影响计时攻击的准确率,而且能够大幅提高攻击的效率。

攻击实现在普通用户环境下实现,在遍历的一个地址时,由于普通用户权限受限问题,无法从Cache中驱逐Linux系统内核的数据,但是可以正常驱逐其他程序的数据,因此,攻击的实现不会受到用户权限的影响。在预取数据时,其他程序地址数据预取时消耗CPU周期较长,预取内核地址数据时较短,如果探测到内核地址的偏移位,预取数据消耗周期会出现跳跃并且会连续一段地址,然后通过分析消耗时间的差异以确定内核数据映射的起始位置。

获取内核地址后,能够更使缓存溢出、旁路攻击、提权等攻击方式更容易实现。在提供prefetcht2指令的Intel CPU中,这种攻击很容易实现,尽管攻击的效率不高,但是能够准确获取内核地址,并且受影响的范围广。针对这种攻击方式,可设计相应的对抗方法,如利用控制精确计时、使用权限、监控Cache驱逐率等方式来阻止攻击。

3 实验与结果分析

通过攻击实验来验证本文方法的有效性。利用计时攻击来获取内核地址的偏移位,为验证计时攻击的可行性,在同一硬件环境下对开启KASLR和未开启KASLR的Linux操作系统进行测试,并在不同硬件环境下的计算机中进行攻击测试,以验证计时攻击的普适性。攻击实验计算机硬件环境CPU为Intel i7-4720HQ和Intel i7-6700,两台计算机的CPU都是4核8线程,且L3 Cache大小分别为6 MB和8 MB;实验环境内存为4 GB,攻击目标为Ubuntu 16.04版本,计时攻击的实现普通用户权限下进行。

在实际环境中,无法确认内核地址偏移位的大概位置,攻击时采用暴力遍历法,通过遍历每个地址来判断内核地址的偏移位。攻击第一步需要将传入的探测地址中的数据清除掉,使用cflush指令将传入的地址从Cache中驱逐,由于递增地址为4 KB,因此每次驱逐数据时需要驱逐以传入地址为起始地址的一个内存页大小。

在未开启 KASLR 防护的 Ubuntu 操作系统中,内核映射的起始位置为 0x00000000,在 32 位操作系统中,前 1 GB 大小虚拟地址空间的内存预留给内核,剩下的地址空间用户使用。攻击起始地址为 0x00000000。如图 3 所示,每一行数据表示一个虚拟内存地址,每一列数据表示一次预取数据消耗的 CPU 时钟周期。每一个地址探测实验进行 12 次,结果如图 3 中各列所示。从地址 0xc0000000 开始,后续的地址数据预取时所消耗的周期明显增加,由此可以推测 0xc0000000 为内核映射的尾地址,与真实情况下的内核地址映射是一致的。实验结果表明,对于没有开启 KASLR 防护的 Linux 操作系统,计时攻击能够准确探测内核地址。

0xbf000000	25	42	23	26	27	24	26	28	29	21	22	24
0xbf000000	23	24	23	27	33	22	21	22	23	26	32	26
0xbf000000	47	26	26	26	25	24	26	27	25	26	27	23
0xc0000000	347	395	353	385	477	334	316	198	297	321	352	344
0xc0100000	321	373	469	411	402	395	373	388	326	357	369	455
0xc0200000	465	385	562	361	354	312	200	268	314	364	375	382
0xc0300000	353	321	387	425	299	622	328	552	458	364	414	332
0xc0400000	333	349	376	329	476	435	322	511	386	374	509	373

图3 针对未开启 KASLR 防护系统的攻击结果

Fig.3 Attack result for Linux system without KASLR protection

在开启 KASLR 防护的 Ubuntu 操作系统中,验证计时攻击探测内核地址的可能性。实验环境与在攻击未开启 KASLR 防护的环境保持不变,攻击起始地址为 0x00000000,递增梯度为 4 KB。图 4 为计时攻击发现内核地址映射偏移位的部分截图,在攻击地址 0x1c000000 时,数据预取时间周期明显缩短,并且预取连续地址消耗的时钟周期很短,因此,推测内核映射的 offset 为 0x1c000000。

0x1d000000	510	437	335	326	427	403	392	429	356	336	529	426
0x1d000000	395	387	333	411	433	452	385	364	371	368	420	392
0x1d000000	412	326	382	319	503	328	400	382	366	375	399	398
0x1c000000	47	35	43	28	27	24	26	28	27	21	23	23
0x1c100000	21	33	29	29	22	26	26	28	57	26	29	28
0x1c200000	29	25	25	21	22	23	20	29	34	34	35	32
0x1c300000	23	23	25	25	29	23	51	26	31	30	28	29
0x1c400000	26	29	28	22	26	26	28	102	24	29	26	30

图4 针对开启 KASLR 防护系统的攻击结果

Fig.4 Attack result for Linux system with KASLR protection

为验证计时攻击的可靠和普适性,在其他 CPU 中重复进行上述实验,探测出不同环境下的内核地址映射的偏移位。因为开启 KASLR 技术的操作系统在每一次开机都会重新映射内核,偏移位会发生变化,因此将同一台机器在两次开机情况下进行,比较内核映射偏移位的区别。如表 2 所示,对不同 CPU 中进行攻击,每台机器都进行两次实验,并且两次获取的内核映射偏移位都不同,经过多次实验获取内核地址偏移位,并在操作系统的 root 环境下获取内核地址偏移位进行比对。结果表明,计时攻击探测出 Ubuntu 16.04 系统内核地址映射偏移位的成功率为 100%。

表2 在不同 CPU 设备下的攻击结果

Table 2 Attack results under different CPU devices

CPU 型号	开机次数	内核探测地址	内核地址偏移位
Intel 6700	第 1 次开机	0x10300000	0x10300000
	第 2 次开机	0x15cb0000	0x15cb0000
Intel 4720HQ	第 1 次开机	0x27dc0000	0x27dc0000
	第 2 次开机	0x1ac00000	0x1ac00000

4 结束语

在使用 KASLR 防护技术的 Linux 系统中,虽然内核地址能够得到有效防护,但存在 CPU 预取指令会泄露计时信息的不足。本文利用这一漏洞,设计一种计时攻击方法来获取 Linux 内核映射地址。实验结果表明,本文攻击方法能够准确获取内核映射偏移位,并且可在不同 CPU 设备中实现。因为 Cache 作为所有进程共享的部件,所以计时攻击方式能够突破虚拟机之间的隔离,达到窥探其他主机内核数据的目的,对云服务商造成严重威胁。本文攻击实现需要遍历探测内存地址,效率较低,下一步将分析 CPU 乱序执行和预执行的特性,对计时攻击模型进行改进,以提高攻击效率。同时针对这种攻击方式提出相应的安全解决方案,以确保信息安全。

参考文献

- [1] GUILLAUME D, DOGET J, PROUFF E. A new second-order side channel attack based on linear regression[J]. IEEE Transactions on Computers, 2013, 62(8): 1629-1640.
- [2] BERNSTEIN D J. Cache-timing attacks on AES [C]// Proceedings of IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design. Washington D. C., USA: IEEE Press, 2005: 218-221.
- [3] ANDREOU A, BOGDANOV A, TISCHHAUSER E. Cache timing attacks on recent microarchitectures [C]// Proceedings of IEEE International Symposium on Hardware Oriented Security and Trust. Washington D. C., USA: IEEE Press, 2017: 155-155.
- [4] LIPP M, SCHWARZ M, GRUSS D, et al. Meltdown [EB/OL]. (2018-04-10) [2020-05-10]. <https://meltdownattack.com/meltdown.pdf>.
- [5] KOCHER P, HORN J, FOGH A, et al. Spectre attacks: exploiting speculative execution [J]. Communications of the ACM, 2020, 63(7): 93-101.
- [6] DMITRY E, PONOMAREV D, ABUGHAZALEH N B. Jump over ASLR: attacking branch predictors to bypass ASLR [C]// Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture. Washington D. C., USA: IEEE Press, 2016: 1-13.
- [7] RALF H, WILLEMS C, HOLZ T. Practical timing side channel attacks against kernel space ASLR [C]//

- Proceedings of IEEE Symposium on Security and Privacy. Washington D. C., USA: IEEE Press, 2013: 1-5.
- [8] CANELLA C, SUNAR B, van BULCK J, et al. Fallout: leaking data on meltdown-resistant CPUs[C]//Proceedings of ACM SIGSAC Conference. New York, USA: ACM Press, 2019: 769-784.
- [9] GAUR J, CHAUDHURI M, SUBRAMONEY S. Bypass and insertion algorithms for exclusive last-level caches[J]. ACM SIGARCH Computer Architecture News, 2011, 39(3): 81-92.
- [10] GREEN M, RODRIGUESLIMA L, ZANKL A, et al. AutoLock: why cache attacks on arm are harder than you think [C]//Proceedings of the 26th USENIX Security Symposium. Vancouver, Canada: USENIX Association, 2017: 1075-1091.
- [11] REINBRECHT C, SUSIN A, BOSSUET L, et al. Side channel attack on NoC-based MPSoCs are practical: NoC Prime+Probe attack [C]//Proceedings of Symposium on Integrated Circuits and Systems Design: Chip on the Mountains. Natal, Brazil: SBCCI, 2016: 1-6.
- [12] YAROM Y, FALKNER K. FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack [C]//Proceedings of USENIX Security Symposium. [S. l.]: USENIX Association, 2014: 1-5.
- [13] LU B G, INCI M S, IRAZOQUI G, et al. A faster and more realistic flush+reload attack on AES [C]//Proceedings of International Workshop on Constructive Side-Channel Analysis and Secure Design. Berlin, Germany: Springer, 2015: 1-5.
- [14] 周平, 王韬, 张帆, 等. SM2 签名算法 flush-reload cache 计时攻击[J]. 华中科技大学学报(自然科学版), 2018, 46(3): 24-29.
- ZHOU P, WANG T, ZHANG F, et al. Flush-reload cache timing attack on SM2 digital signature algorithm [J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2018, 46(3): 24-29. (in Chinese)
- [15] BRASSER F, MÜLLER U, DMITRIENKO A, et al. Software grand exposure: SGX cache attacks are practical [J]. USENIX Workshop on Offensive Technologies, 2017, 9(2): 54-70.
- [16] GRUSS D, SPREITZER R, MANGARD S. Cache template attacks: automating attacks on inclusive last-level caches [C]//Proceedings of USENIX Conference on Security Symposium. [S. l.]: USENIX, 2015: 897-912.
- [17] GRUSS D, MAURICE C, WAGNER K. Flush+Flush: a stealthier last-level cache attack [J]. Computer Science, 2015, 6(3): 279-299.
- [18] 赵新杰, 王韬, 郭世泽, 等. AES 访问驱动 Cache 计时攻击[J]. 软件学报, 2011, 22(3): 572-591.
- ZHAO X J, WANG T, GUO S T, et al. Access driven cache timing attack against AES [J]. Journal of Software, 2011, 22(3): 572-591. (in Chinese)
- [19] 孙玉强, 王文闻, 巢碧霞, 等. 基于预取的 Cache 替换策略[J]. 微电子学与计算机, 2017, 34(1): 85-89, 94.
- SUN Y Q, WANG W W, CHAO B X, et al. Cache replacement policy based on prefetch [J]. Microelectronics & Computer, 2017, 34(1): 85-89, 94. (in Chinese)
- [20] 苗新亮, 蒋烈辉, 常瑞. 访问驱动下的 Cache 侧信道攻击研究综述[J]. 计算机研究与发展, 2020, 57(4): 824-835.
- MIAO X L, JIANG L H, CHANG D. Survey of access-driven cache-based side channel attack [J]. Journal of Computer Research and Development, 2020, 57(4): 824-835. (in Chinese)

编辑 金胡考