计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202110774
60(1): 206−222, 2023

缓存侧信道攻击与防御

张伟娟[1,2] 白 璐[1,2] 凌雨卿[1,2] 兰 晓[3] 贾晓启[1,2]
[1](中国科学院信息工程研究所 北京 100093)
[2](中国科学院大学网络空间安全学院 北京 100049)
[3](四川大学网络空间安全研究院 成都 610207)
(zhangweijuan@iie.ac.cn)

Cache Side-Channel Attacks and Defenses

Zhang Weijuan[1,2], Bai Lu[1,2], Ling Yuqing[1,2], Lan Xiao[3], and Jia Xiaoqi[1,2]
[1](*Institute of Information Engineering*, *Chinese Academy of Sciences*, *Beijing* 100093)
[2](*School of Cyber Security*, *University of Chinese Academy of Sciences*, *Beijing* 100049)
[3](*Cyber Science Research Institute*, *Sichuan University*, *Chengdu* 610207)

Abstract In recent years, with the development of information technology, cache side-channel attack threats in information system has a rapid growth. It has taken more than 10 years for cache side channel attacks to evolve and develop since cache-timing analysis was proposed to speculate encryption keys. In this survey, we comb the cache side-channel attack threats in the information system by analyzing the vulnerabilities in the design characteristics of software and hardware. Then we summarize the attacks from attack scene, cache levels, attack targets and principles. Further more, we compare the attack conditions, advantages and disadvantages of 7 typical cache side-channel attacks in order to better understand their principles and applications. We also make a systematic analysis of the defense technology against cache side channel attack from detection stage and prevention stage, classify and analyze the defence technology based on different defense principles. Finally, we summarize the work of this paper, discuss the research hotspots and the development trend of cache side-channel attack and defense under the Internet ecosystem, and point out the future research direction of cache side-channel attack and defense, so as to provide reference for researchers who want to start research in this field.

Key words information system security; CPU cache; cache side-channel attack; attack detection; defense strategies

摘 要 近年来,随着信息技术的发展,信息系统中的缓存侧信道攻击层出不穷.从最早利用缓存计时分析推测密钥的想法提出至今,缓存侧信道攻击已经历了10余年的发展和演进.研究中梳理了信息系统中缓存侧信道攻击风险,并对缓存侧信道攻击的攻击场景、实现层次、攻击目标和攻击原理进行了总结. 系统分析了针对缓存侧信道攻击的防御技术,从缓存侧信道攻击防御的不同阶段出发,分析了攻击检测和防御实施2部分研究工作,并基于不同防御原理对防御方法进行分类和分析.最后,总结并讨论了互联网生态体系下缓存侧信道攻击与防御的研究热点,指出缓存侧信道攻击与防御未来的研究方向,为想要在这一领域开始研究工作的研究者提供参考.

收稿日期: 2021−07−19; **修回日期**: 2021−12−07
基金项目: 中国科学院战略性先导科技专项(C类)(XDC02010900); 北京市科学技术委员会项目(Z191100007119010); 国家自然科学基金面上项目(61772078); 中国科学院网络测评技术重点实验室项目; 网络安全防护技术北京市重点实验室项目
This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences(XDC02010900), the Project of Beijing Municipal Science & Technology Commission (Z191100007119010), the General Program of the National Natural Science Foundation of China (61772078), the Program of CAS Key Laboratory of Network Assessment Technology, and the Program of Beijing Key Laboratory of Network Security and Protection Technology.
通信作者: 白璐(bailu@iie.ac.cn)

关键词 信息系统安全；CPU缓存；缓存侧信道攻击；攻击检测；防御策略

中图法分类号 TP309

近年来，信息安全越来越引起人们的重视.传统的信息窃取技术通常从软件系统实现漏洞入手，结合特定攻击手段，对目标用户敏感信息（如密钥、口令等）进行窃取.然而，这类攻击方式随着软件系统的更新有着时效性差、易防御等特性.相比之下，从物理信道中提取信息的侧信道攻击（side-channel attack, SCA）技术[1]通常基于设备物理架构特点实现，具有攻击信道多样、难以防御等特点，逐渐成为信息安全领域的研究热点.

侧信道攻击技术最初针对加密设备提出，利用设备不同加密操作产生的时间差异、功率消耗差异或电磁辐射差异构造侧信道，对加密设备进行攻击，推测密钥信息.之后，研究者将这种基于加密操作执行差异构造侧信道，获取密钥信息的思想应用在处理器缓存，提出了缓存侧信道攻击技术.缓存作为处理器微结构的硬件组件，为不同进程所共享，给攻击创造了条件.利用缓存构造的侧信道攻击巧妙且难以防御，其安全风险广泛存在于普通用户个人电脑及运营商服务器，被工业界和学术界广泛关注和研究.

缓存侧信道攻击技术一般分为时序驱动（time-driven）缓存攻击、踪迹驱动（trace-driven）缓存攻击和访问驱动（access-driven）缓存攻击[2].时序驱动缓存攻击[1,3-5]依据执行加解密操作所消耗的时间差异来获取用户信息，实现密钥恢复.踪迹驱动缓存攻击[6]依据用户执行某个操作的能量消耗来推断用户信息，如攻击者通过度量电量获取用户进程的缓存命中与未命中的访问序列，从而获取用户敏感数据.访问驱动缓存攻击[7-11]依靠用户访问缓存时留下的访问位置痕迹获取用户信息，攻击者通过访问与用户共享的缓存来构造信道，通常为借助一个攻击进程探测用户进程在加解密操作执行前后对缓存特定位置访问情况，进而恢复密钥.在这3类攻击中，时序驱动缓存攻击的研究主要针对早期加密硬件；踪迹驱动缓存攻击通常需要一些物理接触来获取电量等数据，攻击条件严格受限，难以进行远程攻击；访问驱动缓存攻击基于用户进程对缓存的访问踪迹实现，攻击粒度较细，攻击条件较灵活，且近年来出现了很多新型攻击，是缓存侧信道攻击研究的主流和热点.

随着缓存侧信道攻击的发展，其逐渐呈现攻击场景复杂化、攻击层次细致化、攻击目标多样化和攻击原理精巧化的趋势.在攻击场景方面，从最初的处理器单核心攻击发展到跨核心，从单机环境延伸到云环境，从单纯针对缓存硬件的攻击到利用新的硬件特性及安全机制进行攻击；在攻击实现层次上，随着CPU的更新换代从开始的基于L1缓存的攻击发展到针对L1、L2、最后一级缓存（last level cache, LLC），甚至基于转址旁路缓存（translation lookaside buffer, TLB）等的攻击；在攻击目标上，从单纯的密钥提取扩展到对关键内存、链接、口令甚至随机数的提取；在攻击原理方面更是多种多样，目前出现的典型攻击及其变种多达数十种.本文对缓存侧信道攻击研究进展进行追踪，并在攻击场景、攻击层次、攻击目标和攻击原理方面做了系统分析.

在缓存侧信道攻击防御方面，本文从攻击防御的不同阶段出发，分析了攻击检测和防御实施2部分研究工作.在防御实施阶段，按照不同防御技术原理，分别从缓存隔离、缓存访问随机化、缓存计时破坏、漏洞分析与修复等方面进行了防御技术的归纳总结.

本文接下来首先对攻击中利用的CPU缓存等组件工作原理予以介绍，并对信息系统中的缓存侧信道攻击风险进行分析；其次对缓存侧信道攻击进行分类，从攻击场景、攻击层次、攻击目标和攻击原理方面对缓存侧信道攻击进行系统分析；再次根据防御原理对缓存侧信道攻击的防御方法进行介绍；最后提出缓存侧信道攻击与防御技术未来可能的研究方向，并对全文进行总结.

1 缓存侧信道攻击的前置问题分析

1.1 缓存的组成及特性分析

为充分理解缓存侧信道攻击原理，本部分首先对缓存的组成、特性及工作机制进行介绍.以x86架构为例[12]，最早的缓存只有L1缓存，L1缓存的访问速度与寄存器接近.随着CPU性能的进一步提升，在L1缓存与主存间逐渐引入了更大的L2和L3缓存，其中L3缓存又称为LLC.L1缓存根据缓存内容又分为数据缓存和指令缓存，数据和指令都具有的缓存则称为联合缓存.在现代计算机系统中缓存结构如图1所示.

缓存的映射方式为3种：直接映射缓存、全相联映射缓存和多路组相联映射缓存，这3种方式各有

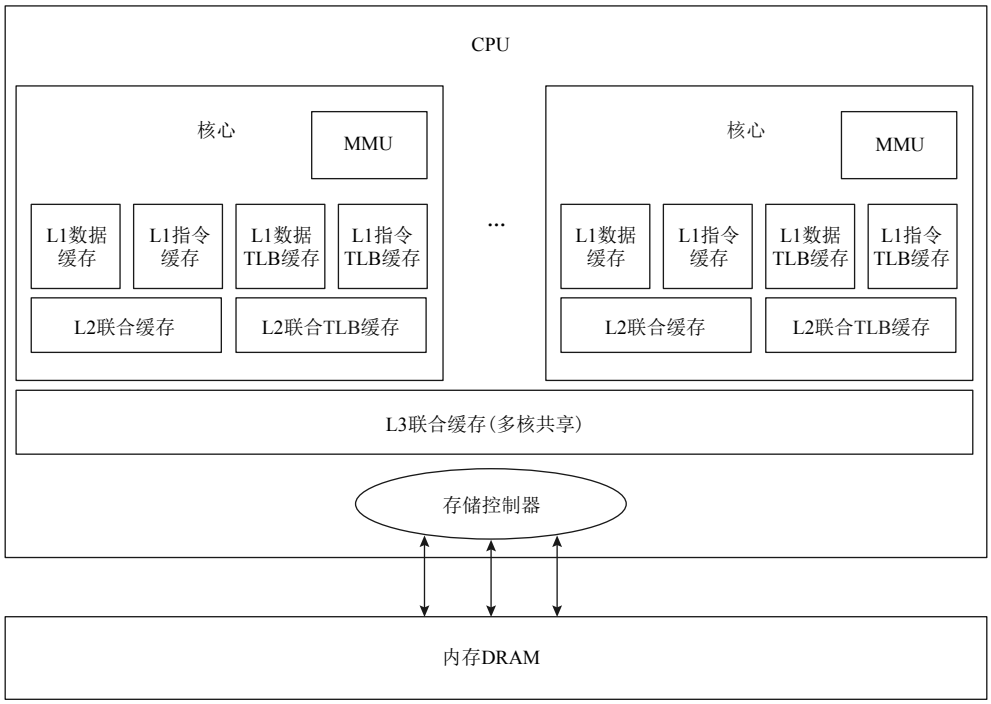

Fig. 1 Cache architectures

图 1 缓存架构

优势.现代系统出于功性能的综合考虑，一般采取多路组相联映射缓存设计.如图 2 所示，多路组相联映射缓存首先被分为 s 个组，每组包含 n 路缓存，也称作 n 个缓存行.内存数据以地址为索引分别映射到不同缓存组中，系统根据缓存组的使用情况选择具体映射在哪个缓存行.由图 2 可以看出，不同内存数据可能会映射在相同的缓存组中，当该缓存组所有缓存行都被占用时，下次 CPU 访问的数据要映射在此缓存组时，就会发生缓存行的换出.

此外，还有一类较为特殊的缓存——TLB，它用来缓存虚拟地址到物理地址的转换，由硬件内存管理单元(memory management unit，MMU)所管理.在 Intel 架构处理器中，TLB 也分为 2 级缓存.当攻击者与用户共享 TLB 时，攻击者同样可进行侧信道攻击[1,13].

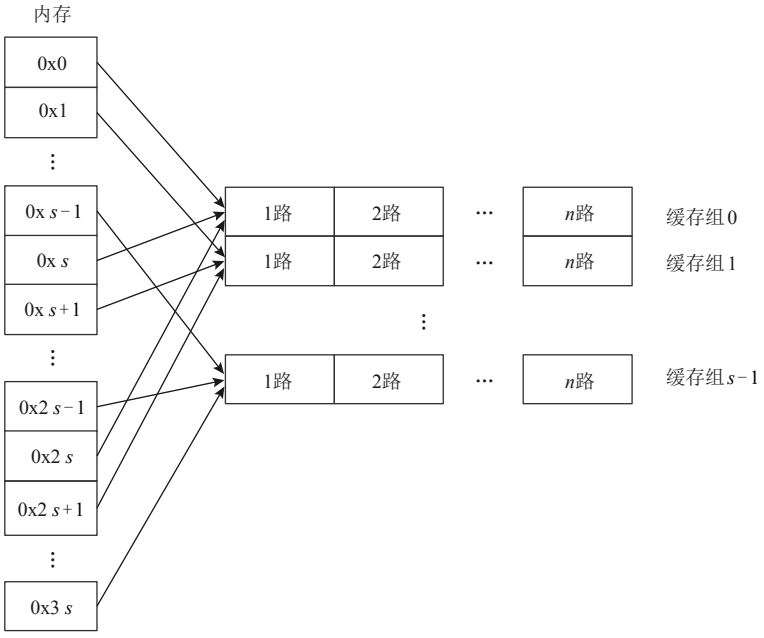

Fig. 2 Multi-way set-associative mapping cache structure

图 2 多路组相联映射缓存结构

1.2 缓存侧信道攻击风险分析

信息系统中存在的缓存侧信道攻击风险主要体现在3个方面：

1)缓存状态可探测

信息系统中攻击进程可以通过特定方式对缓存状态进行探测.首先,由于硬件设计上的特性,缓存在不同进程之间共享,且缓存的分配和映射有一定规律,攻击者可以利用这些规律设计实现有效的缓存监测方案.其次,程序在访问各级缓存及内存时访问时间存在一定差异,通过访问时间差异可推断缓存命中与否,从而监控缓存的使用情况.最后,还可以基于其他硬件特性,如事务同步扩展(transactional synchronization extensions, TSX)对缓存进行监测.TSX事务性内存的原子特性决定了当TSX事务中读取的缓存行被从缓存中驱逐时,会引发事务中止.一旦中止发生,就意味着有其他进程访问该缓存行[14].攻击者可以利用这一特性监控目标进程对特定缓存的使用状况,进而进一步提取信息.缓存状态可探测是构建缓存侧信道的根本.

2)可利用的硬件结构特性

缓存侧信道攻击的实施通常会借助硬件或软件设计的特性,结合构造的缓存侧信道来进行信息提取.在硬件特性方面最典型的为微体系结构特性中的CPU乱序执行和CPU分支预测执行,这2个执行的代表性攻击为Meltdown[15]和Spectre[16].CPU乱序执行时允许在错误指令的未授权结果上进行计算,Meltdown攻击利用这点来暂时绕过硬件安全策略,从而泄露敏感数据.Spectre攻击则利用CPU分支预测执行的特性,通过反复训练引导目标进程执行特定的路径,暂时绕过软件定义的安全策略(边界检查、内存存储等),进而泄露机密信息.这2类攻击后续衍生出大量的变种[17-21],均基于微体系结构特性构建.

3)可利用的程序设计特点

在信息系统软件特性利用方面,攻击者通常需要对密码算法、敏感程序在设计实现上具有的缓存访问特征进行分析,进而设计和构造侧信道攻击方法.以密码算法实现为例,针对非对称密码算法RSA的攻击可基于缓存状态对其函数调用序列进行监控.RSA算法的实现主要为模幂运算,加解密进程在执行时会基于密钥的比特位将模幂运算分解为“平方—乘”或“平方—乘—模—乘”操作.Yarom等人[22]提出的攻击中通过逆向OpenSSL二进制文件,找到了其平方/平方乘操作代码分别对应的内存地址,利用Flush-Reload[11]方法对加密算法的操作地址进行缓存探测,结合缓存命中与未命中的时间分析推断目标的具体操作.

综上,在缓存共享的前提下,攻击者可以结合软硬件特性进行巧妙地构思,实现多种场景下的缓存侧信道攻击.随着信息系统的不断发展,其引入的缓存侧信道攻击风险也在不断变化.

2 缓存侧信道攻击研究现状与分类

本节从缓存侧信道攻击的场景变化、实现层次、攻击目标和攻击原理4个方面对近年来的研究工作进行分类介绍.

2.1 缓存侧信道攻击的场景变化

缓存侧信道攻击是Page[23]在2002年提出,这一研究是对Kelsey等人[24]在1998年提出的利用加密算法实现特点构造侧信道获取用户密钥这一想法的扩展.在此之前,利用侧信道攻击获取密钥的想法被认为理论性强于可操作性,并且难以实际应用,可以说是处于萌芽阶段.

缓存攻击的早期研究工作主要集中于单机环境.在CPU单核心时代,同步多线程技术的出现使得缓存攻击具有了较强的可操作性[9].后续随着CPU架构的发展,CPU开始出现多核心和多级缓存,缓存侧信道攻击也从一开始的L1逐渐扩展到L2, L3,并出现跨核心的缓存攻击.随着计算机硬件架构的复杂化,基于TLB, MMU等模块的缓存攻击也逐渐出现.此外,缓存侧信道攻击还呈现多平台化,如针对AMD, ARM架构[4]的攻击.2016年,Irazoqui等人[25]提出了跨CPU的细粒度缓存攻击,在AMD Opteron处理器和ARM架构处理器上实现缓存侧信道攻击.同年,Lipp等人[26]提出针对移动端设备的缓存攻击ARMageddon.

随着虚拟化技术的发展,云计算应用领域的兴起,出现了针对虚拟化和云环境的缓存侧信道攻击研究.2009年,Ristenpart等人[10]提出了在云环境下跨虚拟机的侧信道攻击方法,并以Amazon EC2云平台为研究对象进行了实验.Ristenpart提出的跨虚拟机缓存侧信道攻击利用Prime-Probe攻击方法来探测EC2实例的缓存使用情况.在这项工作中,攻击者获取信息的粒度较粗,只能获取到目标虚拟机的缓存使用概况,推测出键盘按键信息,攻击效果还不足以从中获取密钥等信息.2012年Zhang等人[7]对跨虚拟机访问驱动型缓存侧信道攻击的构成及实现进行了详细的分析,且在攻击者与目标同驻条件下能够达到细粒度的信息获取,分析出目标的完整私钥.

此外,还有一些缓存侧信道攻击针对可信环境,如软件保护扩展(software guard extensions, SGX)[27]展开.SGX 的攻击面较小,这种攻击通常需要很强的攻击模型,如具有操作系统特权级的控制.一些传统缓存侧信道攻击方法在 SGX 环境同样有效[28-31],这些攻击需要的攻击条件非常苛刻,其假设为攻击者可以控制整个系统资源进行攻击.有研究者将 Prime-Probe 缓存攻击方法应用于 SGX 环境,在 CPU 配置开启超线程使攻击者与目标 Enclave①进程运行在同一核心并共享缓存,在此基础上结合操作系统提供的中断服务实现与目标进程的同步,攻击者结合以上攻击条件获取加密程序的密钥.另外还有利用 Page-fault[32] 中断构造侧信道,对运行在不可信操作系统但认为硬件及虚拟化层可信的内存隔离环境进行攻击,攻击可以获取目标环境的文本及图片信息.之后,Bulck 等人[33] 在此基础上又提出一种更加隐蔽的方法,能够不依赖于 Page-fault 获得 Enclave 所访问的页面信息.Lee 等人[34] 提出一个利用 CPU 分支预测单元构造侧信道攻击的方法 Branch Shadowing. Schwarz 等人[29] 则从另一个角度出发,认为若恶意软件利用 SGX 为 Enclave 提供的安全隔离机制,将攻击行为隐藏在恶意 Enclave 中,则可以构造更加隐蔽的 Enclave-Enclave 之间的攻击.多项研究工作表明,在 TLB 到动态随机存取存储器(dynamic RAM, DRAM)范围都存在对 SGX 的侧信道攻击的潜在攻击向量[35-40].攻击者可以利用 TLB、行填充缓冲区(line fill buffer, LFB)、页表项(page table entry, PTE)、page 属性位等机制构建侧信道,实现较为隐蔽的攻击.

研究者在 SGX 环境下对 CPU 指令特性也进行了深入的挖掘与利用,Foreshadow[41] 和 SgxPectre[42] 分别是在 SGX 环境下利用 Meltdown 和 Spectre 漏洞的攻击变种.这些研究工作从另一方面也说明,即使是在具有很强安全隔离的可信环境中依然无法避免缓存侧信道攻击.

2.2 缓存侧信道攻击的实现层次

从缓存侧信道攻击所利用的缓存层次来分,可以将攻击分为:针对 CPU 核心内部的 L1 数据缓存(D-cache)、L1 指令缓存(I-cache)和 L2 缓存的攻击;针对多核共享的 L3 缓存的攻击;针对其他硬件组件,如 MMU, TLB 等实现的缓存侧信道攻击.

对于 CPU 核心内部的 L1 和 L2 级缓存攻击而言,攻击者与目标运行于同一核心,这类攻击主要利用 Intel 在 2002 年提出的超线程技术(hyper threading)实现.超线程技术使得单核心处理器允许同时执行 2 个线程,提高了处理器的效率.2 个线程若共享某计算或存储资源时会产生资源争用,攻击者通过与目标进程争用 L1, L2 缓存而实现攻击.在能够利用 L3 缓存实现攻击之前,缓存侧信道攻击的研究主要集中于利用 L1 缓存(I-cache 和 D-cache),而 L2 缓存与 L1 缓存在容量和速度上差异不大,研究相对较少[10].

I-cache 是 L1 缓存中负责存储处理器最近所执行指令的组件,Aciiçmez[43] 表明其可利用 I-cache 恢复 RSA 算法的 1 024 b 密钥,实现对 OpenSSL (version 0.9.8d) 的攻击.Aciiçmez 使用的缓存探测方法是 Prime-Probe,先用空指令占满指令缓存继而探测缓存使用情况恢复密钥.此后,又相继出现了对 OpenSSL 不同实现方法的 I-cache 层面缓存攻击[44-45].

D-cache 在 L1 缓存中负责存储最近处理的数据,因而也常被攻击者利用进行敏感信息的恢复.如利用 D-cache 恢复同驻于同一核心的进程信息[9].2006 年,Osvik 等人[8] 提出利用 L1 D-cache 的 Prime-Probe 和 Evict-Time 攻击方法,实现了针对 AES 的密钥恢复.同年,Neve 等人[46] 通过操作系统调度在单线程处理器上实现 L1 D-cache 侧信道攻击.2011 年,Cache games[11] 针对 L1 缓存,利用中断在同核心实现攻击者与 OpenSSL AES 加密线程同步恢复密钥的缓存侧信道攻击.2012 年,Zhang 等人[7] 使用 L1 数据缓存实现了跨虚拟机的攻击,在实验环境跨虚拟机获取 ElGamal 解密密钥.

由于 L1 I-cache 或 D-cache 为单核心所有,攻击者与目标同驻于同一核心比较困难,一定程度上限制了攻击的实际应用效果.随着基于 L3 的缓存侧信道攻击的出现,攻击逐渐具有实际效果.2014 年,Yarom 等人[22] 提出 Flush-Reload 攻击方法,该方法利用了 LLC 和内存页的共享,攻击可跨进程、CPU 核心和虚拟机获取泄露的信息,且攻击粒度细,可以监控到单个内存行.随后,Flush-Reload 攻击方法被扩展应用到对其他加密算法的攻击[47-48] 和在云环境中跨虚拟机(VM)的攻击[49-51],更进一步还有研究者将该攻击方法扩展到 ARM 处理器[26,52] 和浏览器[53] 中.2015 年,Liu 等人[54] 提出了一种针对 L3 缓存的 Prime-Probe 侧信道攻击方法,这一方法能够获取较高的攻击分辨率,

① SGX 允许应用程序初始化一个 Enclave, Enclave 是一块硬件隔离的可信内存区域,可以为应用程序的敏感部分提供硬件增强的机密性和完整性保护,实现不同程序间的隔离运行。

且能够在跨核及跨虚拟机环境下进行攻击.Gruss 等人[55]提出缓存模板攻击，可自动分析和利用任何程序中基于缓存的信息泄露.2016 年，Kayaalp 等人[56]提出一种高分辨率的 LLC 侧信道攻击，与 Flush+Reload 相比，使用限制条件较少.

2017 年，Gras 等人[57]针对 MMU 执行的虚拟地址转换进行攻击，提出一个新的 Evict-Time 缓存攻击 AnC.这种攻击方法利用在虚拟地址翻译中会查找页表并将其缓存在共享缓存中的实现特点，在目标 MMU 查找页表后推断目标所访问的缓存组从而获取信息.AnC 是首个针对硬件组件 MMU 的攻击方法，其攻击场景是构造恶意 JavaScript 代码获取浏览器的地址空间.与之类似，Schaik 等人[58]在 2018 年提出了一种基于 MMU 进行间接缓存攻击的方法，攻击者利用 MMU 在进行地址转换操作时会将页表的部分信息缓存在固定缓存组中的特性，结合 Evict-Time、Prime-Probe 攻击原理设计了新型攻击，这种间接攻击方法可以绕过基于缓存隔离的防御方法.

2018 年，Gras 等人[13]提出了基于 TLBs 的攻击方法 TLBleed.这一方法通过逆向 Intel 处理器地址映射函数，并结合机器学习的方法实现.该方法表明即使用户采用一些防御缓存侧信道攻击的方法，如 CAT（cache allocation technology），但攻击者仍可以通过 TLBleed 攻击探测细粒度的目标信息.因为 TLB 缓存为每个核心所独有，攻击需攻击者与目标同驻于 1 个核心，当攻击者与目标共享 1 个核心时可利用共享的 TLB 构建侧信道进行攻击.2019 年，Canella 等人[59]发现了新的 TLB 侧信道——Store-to-Leak，并提出利用该侧信道的 Fallout 攻击.实验证明，Fallout 可破坏内核地址空间布局随机化，泄露由操作系统内核写入内存的敏感数据.2020 年，Schaik 等人提出 CacheOut[39]攻击，并针对 SGX 环境进行改进提出 SGAxe[40]攻击.CacheOut 攻击利用 TLB 与 L1 数据缓存之间的隐藏交互行为，克服了以往微体系结构数据采样（microarchitectural data sampling，MDS）攻击在控制和可用性上的限制.攻击者可利用此新漏洞来选择要泄露的数据，而不必等待数据可用，同时还可控制泄露地址.

随着计算机技术的发展，现代计算机系统单个处理器拥有核心的数量在增加，单个系统拥有的处理器数量也在增加，即使是移动平台也常常拥有至少 2 个单独的处理器.2016 年，Irazoqui[25]提出了跨 CPU 的细粒度缓存攻击，该方法基于现代处理器和多处理器系统缓存一致性协议来实现侧信道攻击，不依靠特定的缓存结构，可以达到恢复 AES 和 ElGamal 密钥的攻击效果.

综上，随着 CPU 和攻击场景的变化，缓存侧信道攻击的实现近年来有从基于传统的 L1/L2, L3 级缓存向 TLB, MMU 等更小的缓存单元甚至跨 CPU 发展的趋势，攻击层次逐渐多样化，图 3 对缓存侧信道攻击场景和攻击实现层次变化进行了总结.

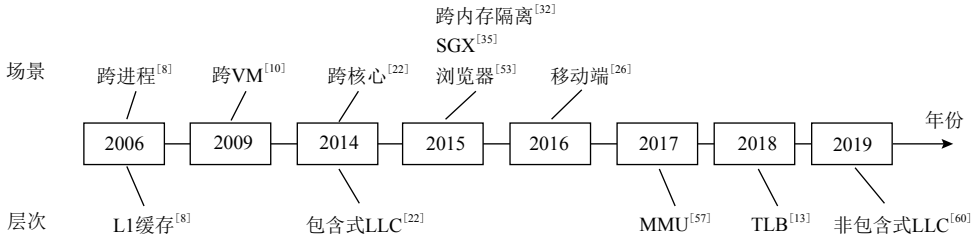

Fig. 3 The development of scenarios and levels of cache side-channel attacks

图 3 缓存侧信道攻击场景和层次的发展

2.3 缓存侧信道攻击的目标

缓存攻击的目的通常为探测和信息窃取，攻击的目标多为包含敏感信息的程序或系统模块，如含密码算法的程序或加密库、用户名/口令文件、关键内存、目标系统运行情况等.本节对缓存侧信道攻击关注的攻击目标做归纳总结，为针对缓存攻击的系统安全性提升提供参考.

1）含密码算法的程序或加密库

缓存侧信道攻击的最典型的攻击目的是提取加解密过程中的密钥，因此最常见的攻击目标为包含密码算法的程序或加密库，如 OpenSSL, GnuPG, GnuTLS, Libcrypt, CyaSSL, MatrixSSL, PolarSSL 等.密码算法通常可分为 2 类，即对称密码和非对称密码.对称密码算法的加密过程和解密过程均采用同一密钥，且加解密过程是“对称”的，典型的算法有 AES, DES 等.研究人员实现了针对 AES 的缓存侧信道攻击等[61]、针对 OpenSSL 中 AES 密钥的提取[8,62-63]攻击等.非对称密码算法又称公钥密码算法，采用公私钥对将加

密和解密功能分开, 主要用途为加密和签名, 典型的算法有 RSA、椭圆曲线等.研究人员实现了对 GnuPG 中 RSA 密钥的提取[22,64]、Libcrypt 中 ElGamal 解密密钥提取[4]、OpenSSL 中 ECDSA 签名破解[48] 等.

2)用户名/口令文件

文献 [49] 中提到借助 Flush-Reload 缓存侧信道攻击方法, 实现基于密码重置方式的用户账号窃取、购物车信息窃取等.RIDL[37] 是一类新的预测执行攻击, 可以跨地址空间和特权边界泄露任意数据, 其中包括跨虚拟机边界.攻击者通过反复尝试用 SSH 进行身份验证, 结合 Flush-Reload 攻击方法提取目标虚拟机/etc/shadow 文件, 进而提取用户口令.

3)关键内存数据

Meltdown[15] 攻击利用现代处理器乱序执行漏洞, 结合 Flush-Reload 缓存侧信道攻击方法实现任意内核内存的读取.Spectre[16] 攻击基于分支预测执行技术实现对应用程序间隔离性的破坏, 使攻击程序有机会访问到其他应用程序所使用的内存空间.研究人员还实现了绕过地址空间布局随机化机制对内存地址空间布局进行推测[59,65].如文献 [66] 设计的攻击方法, 可以在攻击者进程或是用户级目标进程的分支指令间利用分支目标缓冲区(branch target buffer, BTB)冲突来影响攻击者代码执行时间, 进而推断目标进程或内核地址空间分支指令的位置, 获取内核和用户级应用程序的内存地址分布.

4)目标系统运行情况

部分缓存侧信道攻击的目的是探测目标系统运行情况, 例如目标系统运行的操作系统、关键软件、浏览器页面等.目标系统可以是与攻击进程同驻的虚拟机, 也可以是远程连接的操作系统.Irazoqui 等人[67] 给出基于 Flush-Reload 攻击方法对同驻虚拟机的加密库进行探测的攻击, 该攻击可以检测并区分多种加密库, 包括 MatrixSSL, PolarSSL, GnuTLS 等, 还可以对加密库的版本进行区分.Shusterman 等人[68] 提出对浏览器页面的攻击, 该攻击探测同驻虚拟机内部浏览器页面信息, 浏览器类型包括 Chrome (Win, Linux), Firefox (Win, Linux), Safari MacOS, Tor Linux.

本文对 2002—2020 年共 66 篇缓存侧信道攻击文献①中攻击目标进行统计分类, 4 类攻击目标所占比例如图 4 所示, 可见含密码算法的程序或加密库为攻击者的首要攻击目标.

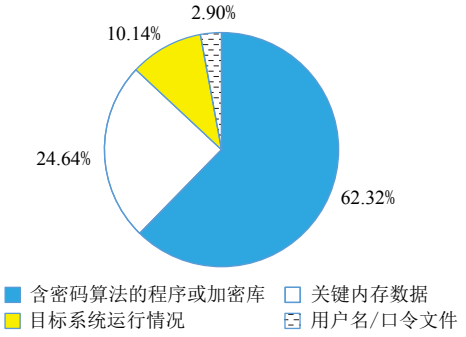

Fig. 4 Statistical analysis of targets of cache side-channel attacks

图 4 缓存侧信道攻击目标统计分析

2.4 缓存侧信道攻击原理

目前从攻击原理来分, 现有的缓存侧信道攻击主要有 Evict-Time, Prime-Probe, Flush-Reload, Flush-Flush, Evict-Reload, Prime-Abort, Reload-Refresh 这 7 种.表 1 对各类型缓存攻击的探测方法进行了对比总结.

2.4.1 Evict-Time

Evict-Time[8] 攻击方法在 2006 年被提出, 是比较早期的 Cache 计时攻击.该攻击在单机单核心环境下恢复出了完整的 AES 密钥, 其攻击步骤有 3 个:

1)目标进程执行加密操作并测量所需时间.

2)Evict.攻击者通过访问自身内存空间, 驱逐部分目标进程执行加密操作时占用的缓存.

3)Time.攻击者再次触发目标进程进行加密运算, 测量完成加密所需要的时间.

该攻击基于 AES 算法的查表操作实现.在执行步骤 1)之后, 目标进程在加密时所用到的全部 $\boldsymbol{T}$ 表② 内存块被加载入内存; 步骤 2)中攻击进程通过访问特定内存块, 驱逐目标进程部分缓存, 例如驱逐的恰好是 $\boldsymbol{T}$ 表某元素 p 映射到的缓存组, 则 p 将被换出缓存; 步骤 3)对同样的明文以预测的密钥 key' 再次执行加密操作, 若执行时间变长说明在此密钥下查找了此前被换出的 $\boldsymbol{T}$ 表元素 p, 以此推断密钥 key' 为密钥 key 的可能性, 并通过进一步分析推断完整密钥.该攻击对目标进程的 $\boldsymbol{T}$ 表加载空间做了大量先验测试, 并在明文已知的前提下通过多次测试来实现密钥恢复.

① 文献主要来自 dblp 数据库, 此外还有部分其他在线资源。

② 在实践中, $\boldsymbol{T}$ 表通常是提前算出然后作为一个常量数组编码在密码算法软件的实现代码中。这样, 就可将繁琐的运算过程变成对于计算机而言简单高效的查表和按位运算。

Table 1 Comprehensive Comparison of Cache Side-Channel Attacks

表 1 不同缓存侧信道攻击技术的综合比较

攻击方法	出现年份	攻击条件	优点	缺点
Evict-Time	2006	攻击者和目标运行在同一处理器，实现查表操作的加密算法攻击．攻击进程可以触发目标进程执行加密操作，并对目标进程加密操作时间进行测量．攻击为已知明文攻击，且通过先验测试获取目标进程 T 表或者 S 盒加载缓存空间	只需要测试加密操作整体时间	不适用于实际攻击场景，需要多次测试
Prime-Probe	2006	攻击进程需要精确把握“等待”间隔，才能准确监测目标进程缓存变化；攻击进程需要事先测好缓存命中的阈值，以便精确判断	细粒度的监测，可以基于缓存组粒度来监测	需提前对缓存状态进行建模，噪声较多，需要多次测试获取缓存状态
Flush-Reload	2014	攻击进程和目标进程共享内存，且支持 clflush 指令．需要提前对程序地址空间进行分析（反汇编等），获取目标函数地址	细粒度的监测，可以进行函数粒度的监控，干扰较少	需要在共享内存的前提下攻击，攻击条件较为苛刻
Evict-Reload	2015	属于 Flush-Reload 攻击变种，在限制 clflush 指令的场景中可以利用 Evict-Reload 方法实现攻击，需要了解缓存映射和替换策略	与 Flush-Reload 攻击相比，不受 clflush 指令的限制	需要在共享内存的前提下攻击，攻击条件较为苛刻
Flush-Flush	2016	属于 Flush-Reload 攻击变种，攻击进程和目标进程共享内存，需支持 clflush 指令	与 Flush-Reload 相比，具有更快速和更隐蔽的特点．细粒度的监测，可以进行函数粒度的监控，干扰较少	需要在共享内存的前提下攻击，攻击条件较为苛刻
Prime-Abort	2017	基于 Intel TSX 的非计时攻击，攻击者需要将要监控的内存空间利用 TSX 机制保护起来，当目标监控的缓存集中有缓存换出，触发 Abort 事件发生	不受高精度计数器的限制，在不具有计时分析条件的场景下也可以实现攻击，不依赖中断保持攻击者进程与目标进程同步	依赖于硬件 TSX 机制
Reload-Refresh	2020	需要知道处理器的缓存替换策略和要监控的目标地址在缓存中的位置，能够构造出该目标地址的驱逐集，并且攻击者和受害者在同一处理器上运行	不需要强制驱逐受害者缓存数据，攻击隐蔽性强	需要在共享内存和明确缓存替换策略的前提下攻击，攻击条件较为苛刻

2.4.2 Prime-Probe

Prime-Probe[8-9] 同样是在 2006 年提出的 Cache 计时攻击，它与 Evict-Time 方法的区别在于它度量的是攻击者访问自身内存块所需的时间，这一特点也使得 Prime-Probe 攻击方法更容易应用于实际场景中．具体攻击步骤有 3 个：

1）Prime.攻击者通过访问内存块填充缓存．

2）等待.攻击者等待一段时间，在这期间目标进程将执行自身代码，访问内存数据．

3）Probe.攻击者再次访问在步骤 1）用来 Prime 缓存的内存块，测试访问时间．

攻击者需要确定一个区分内存访问和缓存访问的时间阈值，还需设计一个驱逐集（映射到同一缓存组，同时分别驻留在不同缓存行的一组地址）.攻击者在步骤 1）中首先利用准备好的数据填充缓存集，等待目标进程运行之后，再次对先前准备的数据集进行访问，根据访问的时间差异判断目标进程对于缓存的访问.如果目标进程没有访问目标缓存集中的数据，则攻击者对该缓存集的访问会缓存命中，则访问时间短；反之，则访问时间长.通过监控目标进程对缓存的使用状况，攻击者可以推断目标进程的敏感信息，如密钥等.Prime-Probe 攻击以单个缓存集为目标，可以检测目标进程对指令或数据的访问，还可以跨虚拟机实现[7,10].Prime-Probe 攻击最开始基于 L1 缓存实现，之后扩展到 L3 缓存.Liu 等人[54] 的研究提出了新的算法使 Prime-Probe 缓存探测方法在 L3 缓存上的应用得以实现，在跨虚拟机的侧信道攻击中成功获取密钥．

2.4.3 Flush-Reload

Flush-Reload 攻击方法在 2011 年提出[11]，随后又发展出了跨核的攻击[22,67].Flush-Reload 方法是较为细粒度的攻击，攻击者可以比较准确地获取特定内存位置是否被缓存的情况，Flush-Reload 攻击方法条件比较苛刻，要求攻击者与目标共享某块使用到的物理内存，例如共享库文件.假设攻击者与目标共享内存块 b，具体攻击步骤有 3 个：

1）Flush.攻击者使用 clflush 指令清空内存块 b 所映射的缓存内容．

2）等待.攻击者等待目标运行一段时间．

3）Reload.重新加载指定内存块 b，测量并记录数据块的重载时间．

步骤 1）保证了下一次访问 b 时需要再次读取内存，同时 clflush 指令会将各级缓存中的 b 都清空.在步骤 2）中，攻击者等待目标执行代码片，期间目标可能会访问到在步骤 1）时换出缓存的数据块 b.步骤 3）攻击者利用 rdtsc 指令度量再次访问数据块 b 所需的时间，若所需时间超过某个阈值则认为在步骤 2）中目标并没有访问到这块数据，否则表示目标在运行

时访问了数据块 *b*.同样地,攻击者通过这种方式监控目标进程对缓存的使用状况,从而推断目标进程的敏感信息.

Flush-Reload 攻击方法将访问驱动型缓存侧信道攻击技术继续向前推进了一步,它不仅将攻击从单一核心扩展到了跨核心,同时还提高了信息获取粒度,将探测缓存的粒度从缓存组(在 x86 架构中一般为 4 路 256 B 或 8 路 512 B)层面提升到了缓存行(在 x86 架构中一般为 64 B).因为具备跨核心和细粒度这 2 点优势,Flush-Reload 方法与之前的侧信道攻击研究相比其攻击性更强也更加危险.

2.4.4 Flush-Flush

Flush-Flush[69] 攻击方法是 Flush-Reload 方法的变种,它的实现同样需要满足与目标共享内存的条件.它利用 clflush 指令在清空缓存时所用时间的差异探测信息,当缓存位置有缓存数据时执行时间长,反之则短.具体的利用方式为:

1)Flush.攻击者执行 clflush 指令清空与目标共享的内存块所映射的缓存行.

2)等待.攻击者等待目标运行一段时间间隔.

3)Flush.攻击者循环执行 clflush 指令,并使用 rdtsc 指令度量 clflush 指令的执行时间.

当被清除的数据为缓存命中时,clflush 指令的执行时间长,反之若为缓存未命中则执行时间短.攻击者通过测量 clflush 指令的执行时间推测内存数据是否被换入缓存,即目标进程是否访问了该内存块数据.Flush-Flush 攻击测量缓存是否命中的时间差异较小,因而准确率比 Flush-Reload 低,但其攻击相对隐蔽,且可以绕过部分基于硬件性能计数器的缓存侧信道攻击检测方法.

2.4.5 Evict-Reload

Evict-Reload[26,55,60] 攻击同样是 Flush-Reload 方法的变种,它将 clflush 指令用缓存驱逐(cache eviction)的方式代替,使得在无法执行 clflush 指令的环境或架构下仍然可以利用 Evict-Reload 方法进行缓存信息探测.具体攻击方式为:

1)Evict.通过访问大块内存,如较大的数组,将目标缓存组换出.

2)等待.等待受害者进程的运行.

3)Reload.重新访问目标地址,测量访问时间,做出判断.

这种攻击方式下要从缓存中驱逐特定的缓存组,需要结合缓存映射方式和替换策略进行设计,通过访问特定内存地址来填充缓存,驱逐目标地址.根据不同 CPU 缓存替换策略、所需的地址数量和访问模式,实现缓存驱逐的方式可能会有所不同.

2.4.6 Prime-Abort

Prime-Abort[14] 是一个不依赖于时间计数的缓存信息探测方法,它利用了 Intel TSX[70] 中事务性内存特性.当目标进程访问特定地址时会引起行为中止(Abort),因此可以依靠 TSX 这一特点监测对某些共享内存块的写操作,监测同一核心上的进程读写操作,或是监测同一处理器上的进程读写操作.具体的利用方式为:

1)Prime.攻击者开启 TSX 事务,用自己的内存块将缓存填满,即 Prime.

2)等待.攻击者等待一段时间,这个间歇目标进程将执行自身代码,访问内存数据等,目标进程的执行将会引起数据在缓存的换入/换出,由于 TSX 原子操作的缘故会导致事务 Abort 的发生.

3)Abort.攻击者检测事务 Abort 的发生,以此推断目标进程访问了特定缓存组.

与传统缓存攻击相比,它具有不依赖时间计数的优势,可以绕过基于计时干扰的攻击防御方法.另外,基于 TSX 机制,目标进程对特定缓存的访问行为会实时触发攻击者的监测,解决了攻击者在探测目标行为时与目标进程难以同步的问题.

2.4.7 Reload-Refresh

Reload-Refresh 攻击[71] 在 2020 年被提出,该攻击巧妙地利用缓存替换策略,在不强制驱逐目标进程数据的情况下,跟踪目标进程的缓存访问.这一攻击同样需要攻击进程和目标进程共享目标内存(即攻击的目标内存空间),攻击进程还要事先准备好一组替换目标内存的内存条目,称为驱逐集,并执行 4 个攻击步骤:

1)准备.攻击进程首先将目标内存读入缓存,然后逐条加载驱逐集,保留一个监控条目(目标地址),作为缓存策略的驱逐候选者.

2)等待.等待目标进程执行,若目标进程访问了目标地址,则目标地址不再是驱逐候选者;若没有访问,则目标地址仍然是驱逐候选者.

3)Reload.攻击进程加载之前保留的监控条目,使得缓存集中发生冲突,缓存要依据替换策略驱逐缓存条目.攻击进程重新读取目标地址,可根据读取时间判断该地址是否被替换,从而推断在上一阶段中是否被目标进程访问.

4)Refresh.将缓存的状态恢复到步骤 1)的状态,进行下一次迭代.

攻击成功实施的前提是要知道当前运行环境中缓存的替换策略.所以在攻击开始之前，需要对目标进程运行的处理器进行测试，以确定该处理器使用的是哪一种缓存替换策略，并根据策略构建目标地址的驱逐集.

3 缓存侧信道攻击防御研究现状与分类

随着缓存攻击的不断出现，针对缓存攻击的防御方法也在同步发展.本节将针对缓存攻击的防御分为2个阶段：1)针对缓存侧信道攻击的检测；2)防御技术的实施.针对缓存攻击的检测可在攻击发生之前或攻击的初始阶段发现攻击，及时告警或终止缓存攻击；针对缓存攻击的防御技术从攻击实现原理、攻击目标等方面入手，考虑通过缓存隔离、缓存访问随机化、缓存计时破坏、漏洞分析与修复等技术对缓存侧信道攻击进行主动防御.本节从攻击检测和攻击防御2个方面出发，对近年来缓存侧信道攻击防御技术的发展进行了系统分析.

3.1 攻击检测

在前期针对缓存侧信道攻击检测技术的研究中，研究人员基于缓存攻击原理做了一些探索.2011年，Zhang等人[72]设计出了命名为HomeAlone的软件工具，通过在自己的虚拟机中运行Prime-Probe攻击并检测操作时间的变化，来检测是否有攻击者的虚拟机和自己同驻.该文针对缓存攻击特征的收集和处理，给缓存攻击检测带来一定借鉴.之后Gruss等人[55]提出缓存模板攻击，开发人员可以针对特定选择的事件自动检测潜在的缓存侧信道漏洞，然后对其进行修复.该文通过触发特定事件，使用Flush-Reload攻击测试访问内存地址的cache-hit踪迹，形成缓存模板.将“缓存模板攻击”作为系统服务运行，则可以检测可能受到攻击的代码和数据.

之后，基于CPU硬件性能计数器(hardware performance counters, HPCs)对缓存攻击进行检测的方法出现.HPCs是一组内置在x86(如Intel, AMD)和ARM处理器中的特殊寄存器，它们与特定硬件事件的事件选择器一起工作，并在硬件事件发生后更新计数器.2013年，Demme等人[73]提出基于HPCs提取的特征检测Prime-Probe攻击.2016年，Chiappetta等人[74]提出将攻击进程硬件特征与机器学习相结合进行缓存攻击实时检测的方法，该方法通过quickhpc工具收集攻击进程运行过程中的硬件特性(CPU cycles, L2 cache hit, L3 cache miss等)，基于机器学习的方法将收集到的进程特征与已有攻击进程特征做匹配，实时检测缓存攻击.同年，Zhang等人[75]提出基于HPCs的缓存攻击检测架构CloudRadar. CloudRadar使用HPCs计数器读取到的硬件事件(cache hit/miss等)作为特征，结合特征检测和异常检测对缓存攻击进行实时检测.该架构可检测引起HPCs变化的缓存攻击如Flush-Reload, Prime-Probe等，根据应用环境的不同可在操作系统层或虚拟化层实现.

还有研究人员希望通过修改硬件架构来实现对缓存侧信道攻击的检测.2014年，Chen等人[76]提出Cache计时攻击检测架构CC-Hunter，该架构通过使用额外的硬件单元来动态跟踪共享CPU的缓存资源使用冲突情况，进而对Cache计时攻击进行检测.部分研究人员提出针对特定缓存攻击的检测.2020年，Guo等人[77]提出基于符号执行的方法来检测由预测执行引入的缓存计时侧信道漏洞.2021年，Kim等人[78]提出基于PMU(performance monitoring unit)，PCM(performance counter monitor)对Prime-Abort攻击进行实时检测的系统.

此外，针对一些新型缓存侧信道攻击，如SGX环境下的缓存侧信道攻击，也出现了一些检测方法.2018年，Chen等人[79]提出HyperRace架构，将虚拟化云环境同驻检测方法HomeAlone[72]的想法扩展到SGX领域，基于LLVM实现了基于超线程技术的缓存侧信道攻击检测方法.研究人员还提出了一类结合TSX技术对SGX缓存侧信道攻击进行检测的方法[80-82]，这类检测方法主要针对特殊环境下的缓存攻击，不具备普适性.

3.2 攻击防御

3.2.1 缓存隔离

缓存侧信道攻击的基本原理是基于进程间缓存共享，因此，基于缓存隔离的攻击防御技术一直是研究人员关注的热点.

Page[83]在2005年提出一个隔离缓存的硬件防御措施，将一部分缓存隔离给受保护的进程使用，以减少不同用户间缓存争用的情况发生.这种对缓存的隔离方法比较粗暴，对缓存使用的性能影响较大.2007年，Wang等人[84]提出一种新的缓存设计结构，采用缓存隔离的方式锁定部分缓存作为安全区域PLcache，这种缓存设计结构与之前的方法相比较为灵活，隔离粒度细.2009年，Kong等人[85]的研究使得防御方法在此前PLcache的基础上又有了新的改善，可以防止更强的攻击.

2016年Zhou等人[86]提出一个内存管理系统

CacheBar, 可以防御利用 L3 缓存共享导致的跨核缓存侧信道攻击.CacheBar 划分不同安全域并且对安全域之间共享的物理内存页进行自动管理, 防止 L3 缓存行的共享, 能够防御 Prime-Probe 和 Flush-Reload 攻击.2016 年, Liu 等人[87] 利用目前商用 CPU 的性能优化技术 CAT 设计实现了原型系统 CATalyst 来实现对 L3 缓存侧信道攻击的防御.CATalyst 通过将 L3 缓存划分成不同的管理域缓存, 从而实现不同虚拟机在 L3 缓存的隔离.这种方法不仅可以防御 Prime-Probe 攻击, 同时也在一定程度上防御了基于页复用技术的 Flush-Reload 攻击.

2017 年, Yan 等人[88] 提出一种针对共享缓存的缓存行替换算法 SHARP, 需要很小的硬件修改就可以有效防御现有的跨核共享缓存攻击.2019 年 Werner 等人[89] 提出了新的缓存设计方法 SCATTERCACHE 来防止缓存攻击, 它排除了特定缓存组的一致性, 使基于驱逐集的缓存攻击不再适用.

还有一类云环境中防御缓存侧信道攻击的典型技术 Page Coloring[90].在云环境中, 要对虚拟机之间的缓存进行隔离, 研究人员需要通过重新设计内存到缓存的映射方案来实现.这类方法因为涉及到物理内存地址到缓存的映射, 所以需要系统级别(系统内核或虚拟化层)的修改, 这种缓存隔离策略通常称为 Page Coloring.Chameleon[91] 提供了 1 个低消耗的自动化 Page Coloring 机制, 在安全域执行时提供缓存隔离.安全进程将被分配在具有特定安全色的缓存中.安全色缓存只提供给关键安全敏感操作使用, 与其同驻于相同硬件的其他虚拟机不可访问这块区域.STEALTHMEM[92] 也是基于 Page Coloring 实现的系统级缓存攻击防御, 其为每个核心管理一组锁定的缓存行, 对这些缓存行的使用不会发生换出的情况.虚拟机可以将敏感数据加载到锁定的缓存行, 锁定的缓存行为该虚拟机独占, 这样一来每个虚拟机可以使用专属的特殊页面来存储敏感数据, 其对敏感数据的访问也不会因为共享而泄露, 从而保障虚拟机之间的安全.

3.2.2 缓存访问随机化

缓存访问随机化的防御方法是通过将内存到缓存的映射随机化, 使其并不按照固定的映射策略进行, 破坏攻击者通过监测缓存特定位置获取信息.RPcache[84] 和 Newcache 就是使用内存到缓存映射随机化的方法来实现防御.RPcache 为需要保护的可信域提供单独的映射表, 可以将索引位映射到不同组, 防止了攻击者将受保护进程缓存行换出的情况发生, 可防御基于 Flush-Reload 进行目标信息探测的攻击.Newcache 通过引入重映射表, 利用逻辑上“直接映射缓存”架构将访问地址随机映射到某缓存行, 实现缓存访问的随机化.2014 年, Liu 等人[93] 提出一个基于硬件的解决方案, 将内存地址到缓存的映射变为运行时自动随机化, 这种缓存映射随机化的方式对性能的影响较小.

2015 年, Crane 等人[94] 提出了利用软件相异性实现缓存攻击防御的方式.该工作利用代码复制和控制流随机化实现了较细粒度的程序多样化来防御缓存攻击, 在保持了原始程序语义的同时确保指令集级别每一副本的不同.此方法通过动态变化程序运行时的控制流, 使攻击者无法从缓存中探测到目标程序的执行意图.

另外一些研究工作基于编译器实现缓存访问随机化, 使程序在执行内存/缓存访问时不易被攻击者追踪.Raccoon[95] 基于混淆程序在源码层制造出程序具有很多执行路径的假象, 使攻击者难以通过程序使用缓存的规律推断出分支的逻辑信息, 进而防御多种侧信道攻击, 如利用地址痕迹、缓存使用、数据大小等作为侧信道的攻击.

3.2.3 缓存计时破坏

大多数缓存侧信道攻击方法依靠不同状态下对特定缓存的访问时间差异来推测目标信息, 因此, 可以通过破坏系统中计时的精度来破坏依赖时间度量的缓存攻击, 进而达到防御的效果.

部分研究人员希望通过增大计时粒度实现缓存攻击的防御.Vattikonda 等人[96] 提出在虚拟化环境中通过去掉细粒度的时间计数来限制恶意虚拟机探测缓存的能力, 他们在 x86 架构的 Xen 虚拟化平台上通过修改 rdtcs 指令返回值来模糊时间度量, 进而使该恶意虚拟机获取的时间粒度变粗, 无法基于高精度的计时信息来分析缓存状况, 从而实现缓存攻击防御.

一些研究人员试图通过在计时器中添加噪声来实现缓存侧信道攻击的防御, 如 Martin 等人[97] 提出的防御措施 TimeWarp.TimeWarp 通过在攻击者进行时间度量时加入任意数值的时间噪声使其无法区分缓存命中与未命中的状态, 使攻击者无法从被噪声污染的数据中获取有效信息.TimeWarp 中的计时器分析较为全面, 它将时间计数分为内部时间计数、外部时间计数和指令时间计数, 这 3 种时间计数分别对应于硬件计时器、外部中断或其他设备传输网络包的传输时间, 以及执行时间周期确定的指令(如执行周期为 1 cycle 的 ADD 指令), 通过对这些时间计

数进行混淆来防御攻击.

还有研究人员研究实现恒定时间的防御方法，x86 处理器中浮点加法和乘法指令的运行时间根据操作数的不同差距很大，Andrysco 等人[98]设计了一个新的数学库 libfixedtimefixedpoint 来缓解浮点数据时间通道的问题，该库可用于非整数运算，且所有操作都在恒定时间内完成.也有一些研究人员提出通过禁用系统中的高精度计时器来实现缓存侧信道攻击防御，如 Gullasch 等人[11]实现的 Cache games 方法则提出通过禁用细粒度的时间计数指令 rdtsc 来缓解缓存攻击.但在信息系统中，很多应用程序基于高精度的计时指令 rdtsc 实现，所以这一类实现方式并不理想.

3.2.4 漏洞分析与修复

软件设计和硬件实现上的一些漏洞往往是攻击者利用的对象，例如密码算法实现的逻辑漏洞、CPU 乱序执行特性等.部分研究人员针对这些侧信道攻击风险展开研究工作，致力于软硬件漏洞的分析和修复，防范于未然.

Doychev 等人[99]实现了缓存侧信道静态分析工具 CacheAudit，该工具以二进制文件作为输入，通过静态分析自动预测程序的缓存使用状况，评估其对各类型缓存攻击的安全对抗情况，预测可能存在的各类型侧信道，进而给出改进建议.Wang 等人[100]提出针对程序的静态分析方法 CacheS.CacheS 基于 SAS（secret-augmented symbolic）域实现，SAS 通常用来在大规模敏感软件上执行抽象表示，完成对程序细粒度敏感信息（如密钥及密钥依赖）的跟踪.CacheS 可以做到对实际使用的加密系统进行完整的静态度量，覆盖率及准确率较高，能够在一定程度上防御缓存侧信道攻击.Hassan 等人[101]组合了 2 个独立开发的侧信道攻击（side-channel attack，SCA）安全框架来识别和测试安全漏洞，并实现了对 Mozilla 的 NSS 安全库的库一级 SCA 安全性评估.

Gras 等人[102]针对商业的 CPU 微体系架构，通过实现自动化的黑盒侧信道分析来发现 CPU 微体系结构中的侧信道攻击风险；还有研究者通过修补微体系结构上导致信息泄露的漏洞来防御相关的缓存侧信道攻击.Andrea 等人[103]提出一种用于研究微体系结构攻击及其缓解措施的新工具 Speculator，用于研究单个代码片段或更复杂的场景（例如分支目标注入攻击）中微体系结构行为.Yu 等人[104]提出数据无关的指令集架构（instruction set architecture，ISA）扩展，该扩展的设计在指令层面保证了现有数据无关的程序（不在共享资源上施加敏感的数据依赖）的安全执行，从而阻塞侧信道攻击.Koruyeh 等人[105]提出的 SpecCFI 防御机制，利用控制流完整性（control-flow integrity，CFI）信息约束投机执行过程中的非法控制流，以限制前向控制流路径（间接调用和分支）上的危险推测，从而防御 Spectre 类型攻击.

3.2.5 其他防御方法

随着新的硬件机制的出现，研究者开始尝试利用硬件特性实现针对缓存侧信道攻击的防御，如 TSX，SGX 等技术.

Gruss 等人[106]提出了针对缓存侧信道攻击的防御措施 Cloak，其核心思想是利用 TSX 技术防止敏感操作或数据访问时缓存未命中情况的发生，防止攻击者探测缓存访问信息.Cloak 利用 TSX 事务内存来执行可能泄露信息的敏感算法，保证所有敏感数据和代码在执行时受事务内存保护，以增加缓存攻击的难度，甚至阻止攻击的发生.

Weiser 等人[107]提出基于 SGX 可信执行环境的缓存侧信道攻击防御方法 SGXJail. SGXJail 针对以 Enclave 为宿主的恶意软件提出了实际的防御机制，其通过对 SGX 进行硬件扩展，利用 Intel 的内存保护密钥限制 Enclave 的执行，实现了对恶意 Enclave 更高效的防御，一定程度防止了 Enclave 间的缓存侧信道攻击.Ahmad 等人[108]提出在 SGX 商用硬件上的混淆引擎方法 OBFUSCURO，该方法可以用来防御缓存模式攻击和缓存计时攻击.Weichbrodt 等人[109]则给出一个 SGX Enclave 性能分析工具集 SGX-perf，可自动化地为 SGX 应用程序提供细粒度的性能关键事件分析，为 SXG 可信环境中缓存侧信道安全分析提供参考.

4 总结与展望

缓存侧信道攻击随着计算机技术的发展及应用环境的变化而不断创新，攻击者在将传统缓存侧信道攻击原理进行扩展的同时也在不断提出新的缓存利用方式，设计和构造令安全防御人员感到棘手的侧信道攻击方法.Meltdown 及 Spectre 漏洞的曝出将缓存侧信道攻击的研究又推向了新的高潮，Intel SGX 等安全机制的出现则使缓存侧信道攻击研究领域进一步扩大.

缓存侧信道攻击的设计往往很精巧，能够造成较强的攻击效果.但缓存侧信道攻击的实施条件也比较苛刻，从实际攻击场景中获取有效信息存在一定难度.从缓存侧信道攻击发展来看，攻击实现的设计思路与信道构造条件基本不变，但近年的研究工作

正尝试着朝攻破更强的安全隔离发展.一方面,随着缓存结构向 non-inclusive[110] 的模式发展,此前主要针对 inclusive 缓存攻击的可靠性可能受到动摇.AMD 处理器一直在使用独立 LLC[111],在 L1 缓存的数据不一定存在于 LLC,也就不能通过 LLC 探测目标信息,Intel 处理器也在向这个方向转变[32].针对 non-inclusive 缓存的攻击或许是未来值得研究的一个方向.另一方面,由于微体系架构中的漏洞相对彻底修补的复杂度和代价更高,近年来新提出的攻击也逐渐偏向于利用 TLB, LFB 等更低层次的缓存结构中的漏洞来构造侧信道,再结合 Flush-Reload 等攻击方法来提取泄露的信息.

在缓存侧信道攻击防御方面,研究者一方面针对典型攻击不断地提出新型防御方法,如攻击检测、缓存隔离、地址随机化、计时器混淆、软硬件漏洞分析与修复等方式;另一方面,针对新型场景下的缓存侧信道攻击,提出在特定条件下的防御措施,如在可信环境 SGX 中的缓存侧信道攻击的防御.目前针对缓存侧信道攻击的防御,软件层面的实现方式多引入较大性能开销,硬件防御方法多为设计思路,未能实际落地.未来,基于新型 CPU 体系结构和硬件设计的防御方案或许是从根本上杜绝缓存侧信道攻击的最终出路.

综上,对于现代处理器架构及操作系统来说,缓存侧信道攻击仍然是其重要的安全威胁.随着新型攻击方法和高效率攻击手段的不断出现,针对缓存侧信道攻击的研究是当下热点,设计实现有效的防御手段同样有着重要意义.

作者贡献声明:张伟娟提出论文主体研究思路和框架,撰写论文初稿;白璐参与论文主体研究思路的讨论,整理文献,以及审阅和修改论文;凌雨卿收集和梳理相关文献,制作图表,排版与校勘论文;兰晓负责部分内容的撰写与修改;贾晓启参与部分内容撰写.

参 考 文 献

[1] Kocher P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems[C] //Proc of the 16th Annual Int Cryptology Conf on Advances in Cryptology. Berlin: Springer, 1996: 104−113

[2] Zhang Yinqian. Cache side channels: State of the art and research opportunities[C] //Proc of the 24th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 2617−2619

[3] Acıiçmez O, Schindler W, Koç Ç K. Cache based remote timing attack on the AES[C] //Proc of the 7th Cryptographers' Track at the RSA Conf on Topics in Cryptology. Berlin: Springer, 2007: 271−286

[4] Bernstein D J. Cache-timing attacks on AES[EB/OL]. [2021-05-27]. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=01774A09A6B2A49EC3AAB74B4EC8705B?doi=10.1.1.67.1622&rep=rep1&type=pdf

[5] Weiß M, Heinz B, Stumpf F. A cache timing attack on AES in virtualization environments[C] //Proc of the 16th Int Conf on Financial Cryptography and Data Security. Berlin: Springer, 2012: 314−328

[6] Aciiçmez O, Koç Ç K. Trace-driven cache attacks on AES[C] //Proc of the 8th Int Conf on Information & Communications Security. Berlin: Springer, 2006: 112−121

[7] Zhang Yinqian, Juels A, Reiter M K, et al. Cross-VM side channels and their use to extract private keys[C] //Proc of the 19th ACM Conf on Computer and Communications Security. New York: ACM, 2012: 305−316

[8] Osvik D A, Shamir A, Tromer E. Cache attacks and counter-measures: The case of AES[C] //Proc of the 6th Topics in Cryptology—the Cryptographers' Track at the RSA Conf. Berlin: Springer, 2006: 1−20

[9] Percival C. Cache missing for fun and profit[C/OL]. [2021-05-27]. http://css.csail.mit.edu/6.858/2014/readings/ht-cache.pdf

[10] Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds[C] //Proc of the 16th ACM Conf on Computer and Communications Security. New York: ACM, 2009: 199−212

[11] Gullasch D, Bangerter E, Krenn S. Cache games: Bringing access-based cache attacks on AES to practice[C] //Proc of the 32nd IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2011: 490−505

[12] Bryant R E, O'Hallaron D R. Computer Systems: A Programmer's Perspective[M]. 2nd ed. New York: Addison-Wesley Educational Publishers Inc, 2010

[13] Gras B, Razavi K, Bos H, et al. Translation leak-aside buffer defeating cache side-channel protections with TLB attacks[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 955−972

[14] Disselkoen C, Kohlbrenner D, Porter L. Prime+abort: A timer-free high-precision L3 cache attack using Intel TSX[C] //Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 51−67

[15] Lipp M, l Schwarz M, Gruss D, et al. Meltdown: Reading kernel memory from user space[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 973−990

[16] Paul K, Daniel G, Daniel G, et al. Spectre attacks: Exploiting speculative execution[J]. arXiv preprint, arXiv: 1801.01203, 2018

[17] Lenovo. Reading privileged memory with a side-channel[EB/OL]. [2021-05-27]. https://support.lenovo.com/us/en/solutions/ps500151-reading-privileged-memory-with-a-side-channel

[18] Intel. Speculative store bypass[EB/OL]. [2021-05-27]. https://www.

intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/speculative-store-bypass.html

[19] Intel. Q3 2018 speculative execution side channel update[EB/OL]. [2021-05-27]. https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00161.html

[20] Kiriansky V, Waldspurger C. Speculative buffer overflows: Attacks and defenses[J]. arXiv preprint, arXiv: 1807.03757, 2018

[21] Koruyeh E M, Khasawneh K N, Song Chengyu, et al. Spectre returns! Speculation attacks using the return stack buffer[C/OL]. [2021-05-27]. https://www.usenix.org/system/files/conference/woot18/woot18-paper-koruyeh.pdf

[22] Yarom Y, Falkner K. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack[C] //Proc of the 23rd USENIX Security Symp. Berkeley, CA: USENIX Association, 2014: 719–732

[23] Page D. Theoretical use of cache memory as a cryptanalytic side-channel[J/OL]. [2021-05-25]. http://eprint.iacr.org/2002/169

[24] Kelsey J, Schneier B, Wagner D, et al. Side channel cryptanalysis of product ciphers[C] //Proc of the 5th European Symp on Research in Computer Security. Berlin: Springer, 1998: 97–110

[25] Irazoqui G, Eisenbarth T, Sunar B. Cross processor cache attacks[C] //Proc of the 11th ACM on Asia Conf on Computer and Communications Security. New York: ACM, 2016: 353–364

[26] Lipp M, Gruss D, Spreitzer R, et al. ARMageddon: Last-level cache attacks on mobile devices[C] //Proc of the 25th USENIX Security Symp. Berkeley, CA: USENIX Association, 2016: 549–564

[27] Intel. Intel software guard extensions[EB/OL]. [2021-05-27]. https://software.intel.com/content/www/cn/zh/develop/topics/software-guard-extensions.html

[28] Brasser F, Müller U, Dmitrienko A, et al. Software grand exposure: SGX cache attacks are practical[C/OL]. [2021-05-25]. https://arxiv.org/abs/1702.07521

[29] Schwarz M, Weiser S, Gruss D, et al. Malware guard extension: Using SGX to conceal cache attacks[C] //Proc of the 13th Int Conf on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin: Springer, 2017: 3–24

[30] Hähnel M, Cui Weidong, Peinado M. High-resolution side channels for untrusted operating systems[C] //Proc of the 26th USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2017: 299–312

[31] Moghimi A, Irazoqui G, Eisenbarth T. CacheZoom: How SGX amplifies the power of cache attacks[C] //Proc of the 19th Int Conf on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2017: 69–90

[32] Xu Yuanzhong, Cui Weidong, Peinado M. Controlled-channel attacks: Deterministic side channels for untrusted operating systems[C] //Proc of the 36th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2015: 640–656

[33] Bulck J V, Weichbrodt N, Kapitza R, et al. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution[C] //Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 1041–1056

[34] Lee S, Shih MW, Gera P, et al. Inferring fine-grained control flow inside SGX enclaves with branch shadowing[C] //Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 556–574

[35] Pessl P, Gruss D, Maurice C, et al. DRAMA: Exploiting DRAM addressing for cross-CPU attacks[C] //Proc of the 24th USENIX Security Symp. Berkeley, CA: USENIX Association, 2015: 564–581

[36] Wang Wenhao, Chen Guoxing, Pan Xiaorui, et al. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX[C] //Proc of the 24th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 2421–2434

[37] Schaik S V, Milburn A, Österlund S, et al. RIDL: Rogue in-flight data load[C] //Proc of the 40th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2019: 88–105

[38] Schwarz M, Lipp M, Moghimi D, et al. ZombieLoad: Cross-privilege-boundary data sampling[C] //Proc of the 26th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 753–768

[39] Schaik S V, Minkin M, Kwong A, et al. CacheOut: Leaking data on Intel CPUs via cache evictions[EB/OL]. [2021-10-05]. https://cacheoutattack.com/files/CacheOut.pdf

[40] Schaik S V, Kwong A, Genkin D, et al. SGAxe: How SGX fails in practice[EB/OL]. [2021-10-05]. https://cacheoutattack.com/files/-SGAxe.pdf

[41] Bulck J V, Minkin Ma, Weisse O, et al. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 991–1008

[42] Chen Guoxing, Chen Sanchuan, Xiao Yuan, et al. SgxPectre: Stealing Intel secrets from SGX enclaves via speculative execution[C] //Proc of the 4th IEEE European Symp on Security and Privacy. Piscataway, NJ: IEEE, 2019: 142–157

[43] Aciiçmez O. Yet another micro architectural attack: Exploiting I-Cache[C] //Proc of the 1st ACM Workshop on Computer Security Architecture. New York: ACM, 2007: 11–18

[44] Aciicmez O, Schindler W. A major vulnerability in RSA implementations due to microarchitectural analysis threat[J/OL]. [2021-05-27]. http://eprint.iacr.org/2007/336

[45] Acıiçmez O, Brumley B B, Grabher P. New results on instruction cache attacks[C] //Proc of the 12th Int Conf on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2010: 110–124

[46] Neve M, Seifert J P. Advances on access-driven cache attacks on AES[C] //Proc of the 13th Int Workshop on Selected Areas in Cryptography. Berlin: Springer, 2006: 147–162

[47] Benger N, Pol J, Smart N P, et al. “Ooh aah. . . Just a little bit”: A small amount of side channel can go a long way[C] //Proc of the 16th Int Workshop on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2014: 75–92

[48] Yarom Y, Benger N. Recovering OpenSSL ECDSA nonces using the FLUSH+RELOAD cache side-channel attack[J/OL]. [2021-05-27]. https://eprint.iacr.org/2014/140.pdf

[49] Zhang Yinqian, Juels A, Reiter M K, et al. Cross-tenant side-channel attacks in PaaS clouds[C] //Proc of the 21st ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2014: 990–1003

[50] Inci M S, Gulmezoglu B, Irazoqui Go, et al. Seriously, get off my cloud! Cross-VM RSA key recovery in a public cloud[J/OL]. [2021-05-27]. https://eprint.iacr.org/2015/898

[51] Apecechea G I, Eisenbarth T, Sunar B. S$A: A shared cache attack that works across cores and defies VM sandboxing and its application to AES[C] //Proc of the 36th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE 2015: 591–604

[52] Zhang Xiaokuan, Xiao Yuan, Zhang Yinqian. Return-oriented Flush-Reload side channels on ARM and their implications for Android devices[C] //Proc of the 23rd ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 858–870

[53] Oren Y, Kemerlis V P, Sethumadhavan S, et al. The spy in the sandbox: Practical cache attacks in JavaScript and their implications[C] //Proc of the 22nd ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2015: 1406–1418

[54] Liu Fangfei, Yarom Y, Ge Qian, et al. Last-level cache side-channel attacks are practical[C] //Proc of the 36th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2015: 605–622

[55] Gruss D, Spreitzer R, Mangard S. Cache template attacks: Automating attacks on inclusive last-level caches[C] //Proc of the 24th USENIX Security Symp. Berkeley, CA: USENIX Association, 2015: 897–912

[56] Kayaalp M, Abu-Ghazaleh N, Ponomarev D. A high-resolution side-channel attack on last-level cache[C/OL]. [2021-05-25]. https://dl.acm.org/doi/10.1145/2897937.2897962

[57] Gras B, Razavi K, Bosman E, et al. ASLR on the line: Practical cache attacks on the MMU[C/OL]. [2021-05-26]. https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_09-1_Gras_paper.pdf

[58] Schaik S V, Giuffrida C, Bos H, et al. Malicious management unit: Why stopping cache attacks in software is harder than you think[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 937–954

[59] Canella C, Genkin D, Giner L, et al. Fallout: Leaking data on meltdown-resistant CPUs[C] //Proc of the 26th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 769–784

[60] Yan Mengjia, Sprabery R, Gopireddy B, et al. Attack directories, not caches: Side channel attacks in a non-inclusive world[C] //Proc of the 40th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2019: 888–904

[61] Zhou Yongbin , Feng Dengguo. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing[J/OL]. [2021-05-27]. https://eprint.iacr.org/2005/388.pdf

[62] Irazoqui G, Inci M S, Eisenbarth T, et al. Wait a minute! A fast, cross-VM attack on AES[C] //Proc of the 17th Int Workshop on Recent Advances in Intrusion Detection. Berlin: Springer, 2014: 299–319

[63] Tromer E, Osvik D A, Shamir A. Efficient cache attacks on AES and countermeasures[J]. Journal of Cryptology, 2010, 23(1): 37−71

[64] Tóth R, Faigl Z, Szalay M, et al. An advanced timing attack scheme on RSA[C] //Proc of the 13th Int Telecommunications Network Strategy and Planning Symp. Piscataway, NJ: IEEE, 2008: 1–24

[65] Hund R, Willems Ca, Holz T. Practical timing side channel attacks against kernel space ASLR[C] //Proc of the 34th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2013: 191–205

[66] Evtyushkin D, Ponomarev D, Abu-Ghazaleh N. Jump over ASLR: Attacking branch predictors to bypass ASLR[C] //Proc of the 49th IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2016: 40:1–40:13

[67] Irazoqui G, Inci M S, Eisenbarth T, et al. Know thy neighbor: Crypto library detection in cloud[J]. Proceedings on Privacy Enhancing Technologies, 2015, 2015(1): 25−40

[68] Shusterman A, Kang L, Haskal Y, et al. Robust website fingerprinting through the cache occupancy channel[C] //Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 639–656

[69] Gruss D, Clémentine M, Wagner K, et al. Flush+Flush: A fast and stealthy cache attack[C] //Proc of the 13th Int Conf on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin: Springer, 2016: 279–299

[70] Intel. Intel development manual[EP/OL]. [2021-05-27]. https://software.intel.com/en-us/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4

[71] Briongos S, Malagón P, Moya J M, et al. RELOAD+REFRESH: Abusing cache replacement policies to perform stealthy cache attacks[C] //Proc of the 29th USENIX Security Symp. Berkeley, CA: USENIX Association, 2020: 1966–1984

[72] Zhang Yinqian, Juels A, Oprea A, et al. HomeAlone: Co-residency detection in the cloud via side-channel analysis[C] //Proc of the 32nd IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2011: 313–328

[73] Demme J, Maycock M, Schmitz J, et al. On the feasibility of online malware detection with performance counters[C] //Proc of the 40th Annual Int Symp on Computer Architecture. New York: ACM, 2013: 559–570

[74] Chiappetta M, Savas E, Yilmaz C. Real time detection of cache-based side-channel attacks using hardware performance counters[J]. Applied Soft Computing, 2016, 49(C): 1162−1174

[75] Zhang Tianwei, Zhang Yinqian, Lee R B. CloudRadar: A real-time side-channel attack detection system in clouds[C] //Proc of the 19th Int Symp on Research in Attacks, Intrusions, and Defenses. Berlin: Springer, 2016: 118–140

[76] Chen Jie, Venkataramani G. CC-Hunter: Uncovering covert timing channels on shared processor hardware[C] //Proc of the 47th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2014: 216–228

[77] Guo Shengjian, Chen Yueqi, Li Peng, et al. SpecuSym: Speculative symbolic execution for cache timing leak detection[C] //Proc of the 42nd ACM/IEEE Int Conf on Software Engineering. New York: ACM, 2020: 1235–1247

[78] Kim H, Hahn C, Hur J. Real-time detection of cache side-channel attack using non-cache hardware events[C] //Proc of the 35th Int Conf on Information Networking. Piscataway, NJ: IEEE. 2021: 28–31

[79] Chen Guoxing, Wang Wenhao, Chen Tianyu, et al. Racing in hyperspace: Closing hyper-threading side channels on SGX with

contrived data races[C] //Proc of the 39th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2018: 178–194

[80] Chen Sanchuan, Liu Fangfei, Mi Zeyu, et al. Leveraging hardware transactional memory for cache side-channel defenses[C] //Proc of the 13th on Asia Conf on Computer and Communications Security. New York: ACM, 2018: 601–608

[81] Chen Sanchuan, Zhang Xiaokuan, Reiter M K, et al. Detecting privileged side-channel attacks in shielded execution with Déjà Vu[C] //Proc of the 12th Asia Conf on Computer and Communications Security. New York: ACM, 2017: 7–18

[82] Shih M W, Lee S, Kim T, et al. T-SGX: Eradicating controlled-channel attacks against enclave programs[C/OL]. [2021-05-27]. https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_07-2_Shih_paper.pdf

[83] Page D. Partitioned cache architecture as a aide-channel defence mechanism[J/OL]. [2021-05-27]. https://eprint.iacr.org/2005/280

[84] Wang Zhenghong, Lee R B. New cache designs for thwarting software cache-based side channel attacks[C] //Proc of the 34th Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2007: 494–505

[85] Kong Jingfei, Aciiçmez O, Seifert J, et al. Hardware-software integrated approaches to defend against software cache-based side channel attacks[C] //Proc of the 15th Int Conf on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2009: 393–404

[86] Zhou Ziqiao, Reiter M K, Zhang Yinqian. A software approach to defeating side channels in last-level caches[C] //Proc of the 23rd ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 871–882

[87] Liu Fangfei, Ge Qian, Yarom Y, et al. CATalyst: Defeating last-level cache side channel attacks in cloud computing[C]//Proc of the 22nd IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2016: 406–418

[88] Yan Mengjia, Gopireddy B, Shull T, et al. Secure hierarchy-aware cache replacement policy (SHARP): Defending against cache-based side channel attacks[C] //Proc of the 44th Annual Int Symp on Computer Architecture. New York: ACM, 2017: 347–360

[89] Werner M, Unterluggauer T, Giner L, et al. SCATTERCACHE: Thwarting cache attacks via cache set randomization[C] //Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 675–692

[90] Raj H, Nathuji R, Singh A, et al. Resource management for isolation enhanced cloud services[C] //Proc of the 1st ACM Workshop on Cloud Computing Security. New York: ACM, 2009: 77–84

[91] Shi Jicheng, Song Xiang, Chen Haibo, et al. Limiting cache-based side channel in multi-tenant cloud using dynamic page coloring[C] //Proc of the 41st IEEE/IFIP Int Conf on Dependable Systems and Networks Workshops. Piscataway, NJ: IEEE, 2011: 194–199

[92] Kim T, Peinado M, Mainar-Ruiz G. STEALTHMEM: System-level protection against cache-based side channel attacks in the cloud[C] //Proc of the 21st USENIX Security Symp. Berkeley, CA: USENIX Association, 2012: 189–204

[93] Liu Fangfei, Lee R B. Random fill cache architecture[C] //Proc of the 47th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2014: 203–215

[94] Crane S, Homescu A, Brunthaler S, et al. Thwarting cache side-channel attacks through dynamic software diversity[C/OL]. [2021-05-27]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.714.3408&rep=rep1&type=pdf

[95] Rane A, Lin C, Tiwari M. Raccoon: Closing digital side-channels through obfuscated execution[C] //Proc of the 24th USENIX Security Symp. Berkeley, CA: USENIX Association, 2015: 431–446

[96] Vattikonda B C, Das S, Shacham H. Eliminating fine grained timers in Xen[C] //Proc of the 3rd ACM Cloud Computing Security Workshop. New York: ACM, 2011: 41–46

[97] Martin R, Demme J, Sethumadhavan S. TimeWarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks[C] //Proc of the 39th Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2012: 118–129

[98] Andrysco M, Kohlbrenner D, Mowery K, et al. On subnormal floating point and abnormal timing[C] //Proc of the 36th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2015: 623–639

[99] Doychev G, Köpf B, Mauborgne L, et al. CacheAudit: A tool for the static analysis of cache side channels[J]. ACM Transactions on Information & System Security, 2015, 18(1): 4:1−4:32

[100] Wang Shuai, Bao Yuyan, Liu Xiao, et al. Identifying cache-based side channels through secret-augmented abstract interpretation[C] //Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 657–674

[101] Hassan S U, Gridin I, Delgado-Lozano I M, et al. Déjà vu: Side-channel analysis of Mozilla's NSS[C] //Proc of the 27th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2020: 1887–1902

[102] Gras B, Giuffrida C, Kurth M, et al. ABSynthe: Automatic blackbox side-channel synthesis on commodity microarchitectures[C/OL]. [2021-05-28]. https://www.ndss-symposium.org/wp-content/uploads/2020/02/23018-paper.pdf

[103] Andrea M, Neugschwandtner M, Sorniotti A, et al. Speculator: A tool to analyze speculative execution attacks and mitigations[C] //Proc of the 35th Annual Computer Security Applications Conf. New York: ACM, 2019: 747–761

[104] Yu Jiyong, Hsiung L, Hajj M E, et al. Data oblivious ISA extensions for side channel-resistant and high performance computing[C/OL]. [2021-05-28]. https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_05B-4_Yu_paper.pdf

[105] Koruyeh E M, Shirazi S, Khasawneh K N, et al. SpecCFI: Mitigating spectre attacks using CFI informed speculation[C] //Proc of the 41st IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2020: 39–53

[106] Gruss D, Lettner J, Schuster F, et al. Strong and efficient cache side-channel protection using hardware transactional memory[C] //Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 217–233

[107] Weiser S, Mayr L, Schwarz M, et al. SGXJail: Defeating enclave malware via confinement[C] //Proc of the 22nd Int Symp on Research in Attacks, Intrusions, and Defenses. Berkeley, CA: USENIX Association, 2019: 353–366

[108] Ahmad A, Joe B, Xiao Yuan, et al. OBFUSCURO: A commodity obfuscation engine on Intel SGX[C/OL]. [2021-05-27]. https://

www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_10-1_Ahmad_paper.pdf

[109] Weichbrodt N, Aublin PL, Kapitza R. SGX-perf: A performance analysis tool for Intel SGX enclaves[C] //Proc of the 19th Int Middleware Conf. New York: ACM, 2018: 201–213

[110] Green M, Rodrigues-Lima L, Zankl A, et al. AutoLock: Why cache attacks on ARM are harder than you think[C] //Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 1075–1091

[111] Intel. 6th Gen Intel Core X-Series Processor Family Datasheet [EP/OL]. [2021-05-26]. https://www.intel.com/content/www/us/en/products/processors/core/6thgen-x-series-datasheet-vol-1.html

Zhang Weijuan, born in 1989. PhD, assistant professor. Her main research interests include cloud computing security, cache side-channel attack and defense.

张伟娟, 1989 年生.博士，助理研究员.主要研究方向为云计算安全、缓存侧信道攻击与防御.

Bai Lu, born in 1988. Master, assistant professor. Her main research interest is cache side-channel attack.

白　璐, 1988 年生.硕士，助理研究员.主要研究方向为缓存侧信道攻击.

Ling Yuqing, born in 1997. Master. Her main research interest is computer side-channel attacks and defenses.

凌雨卿, 1997 年生.硕士.主要研究方向为计算机侧信道攻击与防御.

Lan Xiao, born in 1990. PhD, assistant professor. Her main research interest is applied cryptography, including multi-party computation, provable security research of authenticated key establishment protocols, and blockchain security.

兰　晓, 1990 年生.博士，助理研究员.主要研究方向为应用密码学，包括安全多方计算、认证与密钥协商协议的可证明安全研究、区块链安全.

Jia Xiaoqi, born in 1982. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include cloud computing security, OS security.

贾晓启, 1982 年生.博士，研究员，博士生导师.CCF 高级会员.主要研究方向为云计算安全、操作系统安全.