软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(1):381−403 [doi: 10.13328/j.cnki.jos.006501]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

可信执行环境软件侧信道攻击研究综述*

杨 帆[1], 张倩颖[1,2,3], 施智平[1,4], 关 永[1,5]

[1](首都师范大学 信息工程学院, 北京 100048)

[2](高可靠嵌入式系统北京市工程研究中心 (首都师范大学), 北京 100048)

[3](计算机体系结构国家重点实验室 (中国科学院 计算技术研究所), 北京 100190)

[4](电子系统可靠性技术北京市重点实验室 (首都师范大学), 北京 100048)

[5](北京成像理论与技术高精尖创新中心 (首都师范大学), 北京 100048)

通信作者: 张倩颖, E-mail: qyzhang@cnu.edu.cn

摘 要: 为保护计算设备中安全敏感程序运行环境的安全, 研究人员提出了可信执行环境 (TEE) 技术, 通过对硬件和软件进行隔离为安全敏感程序提供一个与通用计算环境隔离的安全运行环境. 侧信道攻击从传统的需要昂贵设备发展到现在仅基于微体系结构状态就能通过软件方式获取机密信息的访问模式, 从而进一步推测出机密信息. TEE 架构仅提供隔离机制, 无法抵抗这类新出现的软件侧信道攻击. 深入调研了 ARM TrustZone、Intel SGX 和 AMD SEV 这 3 种 TEE 架构的软件侧信道攻击及相应防御措施, 并探讨其攻击和防御机制的发展趋势. 首先, 介绍了 ARM TrustZone、Intel SGX 和 AMD SEV 的基本原理, 并详细阐述了软件侧信道攻击的定义以及缓存侧信道攻击的分类、方法和步骤; 之后从处理器指令执行的角度, 提出一种 TEE 攻击面分类方法, 利用该方法对 TEE 软件侧信道攻击进行分类, 并阐述了软件侧信道攻击与其他攻击相结合的组合攻击; 然后详细讨论 TEE 软件侧信道攻击的威胁模型; 最后全面总结业界对 TEE 软件侧信道攻击的防御措施, 并从攻击和防御两方面探讨 TEE 软件侧信道攻击未来的研究趋势.

关键词: 可信执行环境 (TEE); 隔离架构; ARM TrustZone; Intel SGX; AMD SEV; 软件侧信道攻击

中图法分类号: TP309

中文引用格式: 杨帆, 张倩颖, 施智平, 关永. 可信执行环境软件侧信道攻击研究综述. 软件学报, 2023, 34(1): 381–403. http://www.jos.org.cn/1000-9825/6501.htm

英文引用格式: Yang F, Zhang QY, Shi ZP, Guan Y. Survey on Software Side-channel Attacks in Trusted Execution Environment. Ruan Jian Xue Bao/Journal of Software, 2023, 34(1): 381–403 (in Chinese). http://www.jos.org.cn/1000-9825/6501.htm

Survey on Software Side-channel Attacks in Trusted Execution Environment

YANG Fan[1], ZHANG Qian-Ying[1,2,3], SHI Zhi-Ping[1,4], GUAN Yong[1,5]

[1](College of Information Engineering, Capital Normal University, Beijing 100048, China)

[2](Beijing Engineering Research Center of High Reliable Embedded System (Capital Normal University), Beijing 100048, China)

[3](State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences), Beijing 100190, China)

[4](Beijing Key Laboratory of Electronic System Reliability Technology (Capital Normal University), Beijing 100048, China)

[5](Beijing Advanced Innovation Center for Imaging Theory and Technology (Capital Normal University), Beijing 100048, China)

Abstract: In order to protect the security of the execution environment of security-sensitive programs in computing devices, researchers have proposed the trusted execution environment (TEE) technology, which provides security-sensitive programs with a secure execution

* 基金项目: 国家自然科学基金 (61802375, 61602325, 61876111, 61877040); 北京市教委科技计划一般项目 (KM201910028005); 中国科学院计算技术研究所计算机体系结构国家重点实验室开放课题 (CARCH201920); 交叉科学研究院项目 (19530012005)
收稿时间: 2021-03-07; 修改时间: 2021-05-31; 采用时间: 2021-10-03; jos 在线出版时间: 2021-10-20
CNKI 网络首发时间: 2022-11-15

environment isolated from the rich computing environment by hardware and software isolations. Side-channel attacks have evolved from traditional attacks requiring expensive equipment to now attacks using software to infer confidential information from its access mode obtained through microarchitecture states. The TEE architecture only provides an isolation mechanism and cannot resist this kind of emerging software side-channel attacks. This study thoroughly investigates the software side-channel attacks and corresponding defense mechanisms of three TEE architectures: ARM TrustZone, Intel SGX, and AMD SEV, and discusses the development trends of the attacks and defense mechanisms. First, this study introduces the basic principles of ARM TrustZone, Intel SGX, and AMD SEV, and then elaborates on the definition of software side-channel attacks and the classification, methods, and steps of cache side-channel attacks. Second, from the perspective of processor instruction execution, a TEE attack surface classification method is proposed to classify TEE software side-channel attacks, and the attacks combining software side-channel attacks and other attacks are explained. Third, the threat model of TEE software side-channel attacks is discussed in detail. Finally, the industry's defense mechanisms against TEE software side-channel attacks are comprehensively summarized, and some future research trends of TEE software side-channel attacks are discussed from two aspects: attack and defense.

Key words: trusted execution environment (TEE); isolation architecture; ARM TrustZone; Intel SGX; AMD SEV; software side-channel attack

随着计算设备在各种领域的应用, 计算设备中存储着越来越多的个人信息和秘密数据. 例如, 移动和嵌入式设备不仅绑定了用户身份证和银行卡, 而且增加了移动支付功能. 然而, 这些安全敏感的信息缺乏专用的保护机制, 容易被攻击者窃取, 从而给用户造成极其严重的安全隐患. 为了给用户应用程序提供一个安全的执行环境, 学术界提出了可信执行环境 (trusted execution environment, TEE) 的概念, 它通过隔离硬件和软件来保证应用程序执行环境的安全.

与传统的通用执行环境 (rich execution environment, REE) 不同, TEE 位于主处理器内部并且与通用操作系统 (rich operating system, Rich OS) 并行运行, 在 TEE 中运行的应用程序称为可信应用 (trusted application, TA), 在 REE 中运行的应用程序称为客户应用 (client application, CA). 为了保证可信应用程序的机密性和完整性, TEE 通过隔离计算设备的软硬件资源为安全敏感的应用程序提供了安全的执行环境. 目前, 产业界已经提出很多 TEE 隔离架构, 其中典型的有 AMD SEV (secure encrypted virtualization)、ARM TrustZone 和 Intel SGX (software guard extensions). AMD SEV 是一种加密虚拟机数据的安全技术, 可以实现虚拟机监视程序 (hypervisor) 与虚拟机之间的隔离, 但它缺乏对主内存页面的完整性保护; ARM TrustZone 和 Intel SGX 技术实现了计算机硬件层面的隔离: ARM TrustZone 技术在 CPU、总线、MMU、cache、DRAM 以及外设等系统资源上都实现了安全扩展, 从而保护计算设备中安全敏感的应用程序和数据不被攻击者非法窃取; Intel SGX 通过增加一组与安全相关的指令集来创建被称为飞地 (enclave) 的可信环境, 飞地代码和数据存储在被硬件加密的飞地页面缓存 (enclave page cache, EPC) 中, 使得不同应用程序之间实现运行隔离. ARM TrustZone 技术和 Intel SGX 技术通过隔离机制给应用程序提供了安全的运行环境. TEE 虽然可以通过隔离机制来保障用户运行环境的安全, 但是攻击者仍可利用软件侧信道攻击来获取 TEE 内部的机密信息.

侧信道攻击 (side-channel attacks, SCA)[1–3]的主要思路是通过一定的信道 (channel) 获取系统运行时产生的副作用信息, 然后利用副作用信息推理系统内部的机密信息[4]. 传统的侧信道攻击主要利用系统产生的物理方面的副作用信息, 包括音量、电磁辐射量、功耗大小、时间长短等, 然后通过声音[5]、电磁辐射[6]、功耗[7]、时间[8]等这些物理信道获取副作用信息, 最终推测系统内部的信息. 而软件侧信道主要将软件运行对 CPU 微体系结构的影响作为副作用信息, 通过 cache 等微体系结构构造软件信道获取微体系结构的副作用信息, 最终推测软件内部的机密信息. 因此软件侧信道与传统侧信道的不同主要体现在两个方面: 副作用信息的来源及其传输信道, 传统的侧信道攻击依赖系统产生的物理层面的副作用信息, 它获取这些副作用信息的方式是一些物理信道. 软件侧信道攻击依赖的副作用信息是软件运行对 CPU 微体系结构的影响, 其获得这些副作用信息的方式主要是基于 cache 等构建的软件信道.

为什么软件侧信道攻击能绕过当前隔离架构的保护而窃取到应用程序的机密信息呢? 原因是当前的隔离架构中微体系结构层面的优化组件会导致 CPU 运行时出现错误 (例如, 预测错误) 或发生系统事件 (例如, page fault), 这些错误和系统事件会导致 TEE 在微体系结构层面留下与机密信息相关的痕迹. 当前的隔离架构沿用了通

用执行环境中微体系结构层面的优化组件, 这些具有优化功能的微体系结构组件被用于改善 CPU 计算性能或提高计算设备的执行效率, 例如, 重排序缓冲区 (reorder buffer) 或 CPU cache 等. 如果优化组件发生错误, 系统会根据组件的设计逻辑处理该错误. 优化组件发生错误以及系统处理错误均在体系结构层面上不可见, 但会改变系统的微体系结构状态, 从而给攻击者留下侧信道漏洞. 在 TEE 隔离环境中, CPU 缓存、MMU 和 DRAM 等与内存访问相关的部件以及重排序缓冲区、预留站 (reservation station) 和退休缓冲区 (retirement buffer) 等与瞬态执行相关的部件可能出现侧信道漏洞, 与之对应的攻击有 cache 侧信道攻击、TLB 侧信道攻击、page table 侧信道攻击、DRAMA、瞬态执行攻击 (transient-execution attack) 等, 将它们统称为基于微体系结构的攻击.

本文首先介绍 3 种隔离架构的基本原理, 并总结现有的软件侧信道攻击的方法; 之后, 通过分析 ARM 体系结构中 CPU 指令流水线的执行过程, 提出一种基于 CPU 指令流水线的 TEE 攻击面分类方法, 利用这种分类方法, 可以将目前已知的所有 TEE 软件侧信道攻击进行归类; 然后, 讨论 TEE 软件侧信道攻击的威胁模型, 并按照攻击者能力的大小将其分为 3 类; 最后, 针对基于微体系结构攻击和组合攻击, 总结已有的防御措施, 并从攻击和防御两方面对 TEE 软件侧信道攻击研究的发展趋势进行展望.

1 背　景

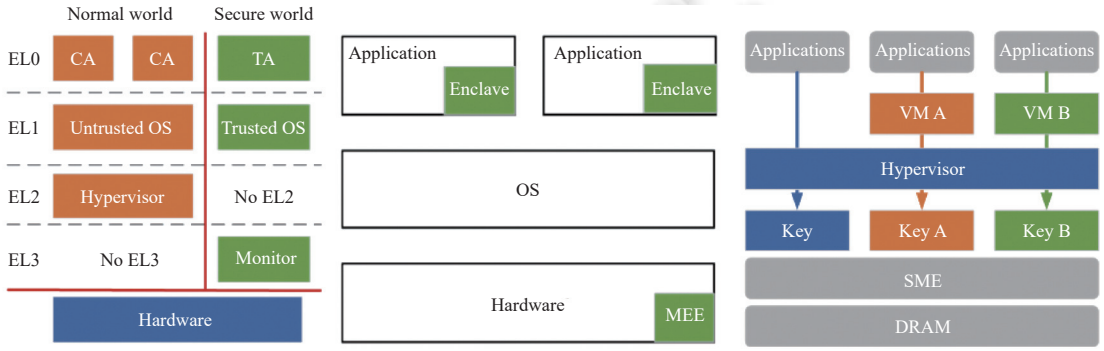
本节首先分别介绍 ARM TrustZone、Intel SGX 和 AMD SEV 这 3 种典型的隔离架构, 然后阐述了软件侧信道攻击的定义、基本原理以及与传统侧信道攻击的区别, 并对缓存侧信道攻击的方法和步骤进行详细分类.

1.1 ARM TrustZone

ARM TrustZone 是 ARM 处理器架构的硬件安全扩展[9–11]. ARM TrustZone 技术将 CPU 逻辑上划分为普通世界 (normal world) 和安全世界 (secure world), 每个世界都有自己的操作系统来管理该世界的应用程序, 监视器模式 (monitor mode) 负责普通世界和安全世界的上下文切换. 同时, 总线、MMU 和 cache 都扩展了一个新的 NS 位, 用于标记当前 CPU 所处于的状态, TrustZone 还增加了 TZASC (TrustZone address space controller)、TZPC (TrustZone protection controller) 和 TZMA (TrustZone memory adapter) 组件来增强外围设备的安全. 图 1(a) 描述了 ARM TrustZone 的架构图.

1.2 Intel SGX

Intel SGX 是 x86 体系结构的扩展[10]. 为了保护应用程序中关键代码和数据不被恶意软件破坏, SGX 通过增加一组与安全相关的指令集来创建一种被称 enclave 的可信执行环境. 飞地代码和数据存储在保留内存区域 (preserved random memory, PRM) 的 EPC 中, 并使用内存加密引擎 (memory encryption engine, MEE) 对 EPC 进行加密, 用于确保应用程序的机密性和完整性; EPC 的访问控制由 EPCM (enclave page cache map) 负责, 它被用于存储每个页面的状态 (如配置、权限、类型等), 任何 non-enclave 的应用程序都不能访问 EPC 的内容, 这使得恶意应用程序无法访问这些敏感数据, 使得不同应用程序之间实现运行隔离. 图 1(b) 描述了 Intel SGX 的架构图.



(a) ARM TrustZone 构架　(b) Intel SGX 构架　(c) AMD SEV 构架

图 1　3 种典型的 TEE 隔离架构图

1.3 AMD SEV

安全加密虚拟化 SEV 技术是 AMD 虚拟化体系结构的安全扩展[12], 它将内存加密技术与现有的 AMD-V 虚拟化体系结构相结合, 保护虚拟机免受物理攻击, 实现不同虚拟机之间以及 hypervisor 与虚拟机之间的隔离. AMD SEV 采用的内存加密技术称为安全内存加密 (secure memory encryption, SME), SME 由嵌入式高级加密标准 (advanced encryption standard, AES) 引擎执行, 当数据写入 DRAM 或从 DRAM 读取数据时, 该 AES 引擎会根据不同 VM 的加密密钥 (VM encryption keys, VEK) 对其进行加密或解密操作. AMD SEV 允许每个 VM 都拥有自己的 VEK, VEK 每次系统启动时随机生成并存储在 SoC 上的寄存器中, 由安全处理器 AMD-SP 管理. 图 1(c) 描述了 AMD SEV 的架构图.

由于 AMD SEV 技术存在设计缺陷, 目前针对 AMD SEV 的攻击都利用这种缺陷破坏 SEV 内存加密技术提供的机密性. 典型的设计缺陷包括: 基于 ASID 的不正确的内存隔离和访问控制缺陷、虚拟机控制块 (VM control blocks, VMCB) 未加密、nested page table 缺乏保护、内存加密缺乏身份验证、I/O 操作未受保护等. 为了解决这些安全问题, 以进一步增强 AMD SEV 的安全, AMD 在 SEV 的基础上增加了 SEV-ES 和 SEV-SNP 两种安全扩展, 它们很好地抵御了先前被发现的针对 AMD SEV 的所有攻击.

1.4 软件侧信道攻击

系统运行会对计算设备的物理实现造成影响, 这种影响将通过不同的 channel 传输出来. 传统的侧信道攻击将目标设备在运行期间产生的时间、声音、电磁辐射、功耗等物理信号作为 channel, 通过收集这些物理信号分析机密信息. 这类传统的侧信道攻击不仅要求攻击者物理接触目标设备, 而且还需要花费大量时间对信号进行分析和噪声处理, 攻击成本高, 效率低下. 随着攻击技术的发展, 软件侧信道攻击成为目前主流的攻击技术之一, 该攻击依赖微体系结构泄露的副作用信息, 再通过软件方式将机密信息从 channel 中提取出来, 它不需要物理探测设备以及对物理信号进行解析. 一般来说, 传统的侧信道攻击大致可以分为两个阶段: (1) 信号探测阶段, 攻击者利用物理器件探测目标加密设备在运行时产生的物理信号, 该物理信号可作为信息泄露的 channel; (2) 信号分析阶段, 由于系统执行不同的运算会导致 channel 信号发生变化, 因此攻击者可以通过分析收集到的 channel 信息恢复机密信息. 软件侧信道攻击可以分为 3 个阶段: (1) 硬件分析阶段, 分析处理器硬件层面存在的安全缺陷, 找到能泄露受害者机密信息的微体系结构组件; (2) 攻击构建阶段, 利用软件方式构建攻击, 将硬件安全缺陷泄露的机密信息提取到不同的微体系结构组件中; (3) 提取机密信息阶段, 根据不同的微体系结构组件, 构建对应的侧信道攻击, 以微体系结构状态的形式 (例如, cache 侧信道攻击中的时间信息) 提取机密信息.

在软件侧信道攻击中, 由于 cache 侧信道攻击技术[13−37]十分成熟、空间和时间分辨率高、噪声低, 因此攻击者更倾向于构建 cache 侧信道攻击提取机密信息. 根据攻击者对受害者访问能力的差异, 将 cache 侧信道攻击分为时间驱动攻击 (time-driven attack)[38]、访问驱动攻击 (access-driven attack)[39]以及踪迹驱动攻击 (trace-driven attack)[40]. 在时间驱动攻击中, 攻击者仅了解受害者执行的总时间即可获取攻击者想要的信息. 另外, 根据攻击者的位置, 时间驱动攻击又可分为主动攻击和被动攻击; 在访问驱动攻击中, 攻击者监视受害者在运行期间是否与 cache 组发生交互行为, 通过测量 cache hit 和 cache miss 发生的时间来推断出受害者访问过的数据, 访问驱动攻击要求攻击者和受害者彼此共享缓存; 在踪迹驱动攻击中, 攻击者通过操纵 cache 来观测受害者进程在运行期间是否与 cache 行发生交互行为 (例如, cache hit 或 cache miss), 基于此交互信息来获取机密信息, 踪迹驱动攻击比时间驱动攻击更高效、更复杂. 常见的 cache 侧信道攻击方法原理如后文表 1 所示.

2 TEE 侧信道攻击面

计算设备中常见的攻击可以分为 3 类, 分别是软件攻击、电路板级物理攻击和芯片级物理攻击. 软件攻击指只执行纯软件的攻击, 实施这类攻击所需要的条件不高, 因此软件攻击最常见. 电路板级物理攻击指采用低成本硬件 (例如, 探针) 而发起的攻击, 它通过在电路板上挂载攻击设备以窃听或篡改计算设备上的信息[41]. 芯片级物理攻击指对芯片内部的机密信息进行攻击, 比如 CPU 根密钥, 这类攻击依赖的攻击设备昂贵, 并且对攻击者具有很

高的要求, 攻击成本很高. 为了增强计算机系统的安全性, 学术界利用 TEE 技术给应用程序提供一个隔离的执行环境. TEE 通常可以抵抗软件攻击和电路板级物理攻击, ARM TrustZone 技术通过对系统资源进行隔离从而保证位于普通世界的恶意软件无法攻击安全世界的应用程序, 因此 ARM TrustZone 技术可以抵御软件攻击; Intel SGX 技术和 AMD SEV 技术都可以抵御电路板级物理攻击, 因为 Intel SGX 利用内存加密技术保证飞地代码和数据加密存储在内存上, AMD SEV/SNP 通过内存加密技术保护虚拟机数据在片外内存的安全性, 它们都将机密代码和数据以密文的形式存储在 CPU 外部, 因此可以抵御第 2 类攻击. 目前所有的 TEE 技术都没有对 CPU 芯片进行物理防护, 因此都不能抵抗芯片级物理攻击.

表 1　常见的 cache 侧信道攻击方法

方法	步骤
Prime+Probe[15,17−22,27,31,32]	1. 攻击者用自己的数据填充特定的cache组 (Prime) 2. 等待目标受害者执行, 更新缓存 3. 重新读取Prime阶段填充的数据, 测量并记录各个cache组的读取时间 (Probe)
Flush+Reload[13,14,17,18,20,22,23,28,30,31,33]	1. 将共享内存中特定位置的缓存数据逐出 (Flush) 2. 等待目标受害者执行, 更新缓存 3. 重新加载Flush阶段逐出的内存块, 测量并记录cache组的重载时间 (Reload)
Flush+Flush[17,18,20,22,29]	1. 通过Flush清空缓存的原始数据 2. 等待目标受害者运行, 更新缓存, 并刷新共享缓存行, 测量刷新时间 3. 根据测量时间判断原始数据是否被重新加载进缓存
Evict+Time[13,15,17−20,22]	1. 受害者执行, 并记录其执行时间 2. 使用Evict方法覆盖cache组中的数据 3. 再次执行受害者进程, 并第2次记录执行时间 (Time), 如果时间不一致且执行时间变长则说明程序运行时读取了缓存中的数据
Prime+Count[16]	1. 攻击者使用自身地址空间中的内容填充缓存 (Prime) 2. 等待受害者执行, 更新缓存 3. 攻击者通过PMU (performance monitor unit, PMU) 检测发生缓存未命中事件的个数 (Count)
Prime+Abort[20,34]	1. 打开TSX (transactional synchronization extensions)事务 2. 攻击者用自己的数据填充特定的cache组 (Prime) 3. 等待受害者执行 4. 如果发生Abort, 则表示某些程序已经访问了目标cache组 (Abort)
Reload+Refresh[36]	1. 攻击者和受害者需要共享内存, 受害者要用到的数据是w, 假设insertion age为2 2. 攻击者将目标地址与w−1个数据(0–ev_{w-2})放入cache, 置cache为insertion age 3. 等待受害者访问cache 1) 如果受害者访问了目标, 则目标块的age减1, 其余age不变 2) 如果受害者没有访问目标, 则所有的age均不变 4. 检索信息(强制miss+Reload): 攻击者访问ev_{w-1}来强制产生一个cache miss, 逐出组的位置由受害者是否访问了该目标决定 1) 如果受害者访问了目标, 按照逐出策略, 逐出第1个age为3的块并将其置为insertion age 2) 如果受害者没有访问目标, 按照逐出策略, 逐出目标块并将其置为insertion age 5. 刷新 (Refresh)状态: 刷新cache, 重载 (Reload) 目标并将其置为insertion age, 然后再Reload被ev_{w-1}逐出的数据

虽然现在主流的 TEE 技术可以通过硬件层面的隔离技术来保护应用程序免受软件攻击或电路板级物理攻击, 但它们未在处理器微体系结构层面进行隔离. 现代处理器从成本和效率角度出发, 在微体系结构层面使用了许多共享和优化机制, 系统运行会在微体系结构中遗留下一些副作用信息. 软件侧信道攻击者通过构建微体系结构的侧信道攻击, 就能将微体系结构层面的副作用信息提取出来并进一步推测得到软件运行时的机密信息.

软件侧信道攻击严重威胁着 TEE 隔离架构的安全, 但是当前学术界对 TEE 软件侧信道攻击面缺乏系统分析, 这不利于研究人员进一步挖掘 TEE 隔离架构的侧信道漏洞. 为了使 TEE 中软件侧信道攻击面的研究系统化, 本文提出一种基于 CPU 指令流水线的 TEE 攻击面分类方法.

CPU 指令流水线的执行过程几乎涉及了计算设备全部的微体系结构组件, 通过分析 ARM 体系结构中 CPU

的指令流水线, 找出此过程可能出现侧信道攻击面的组件, 本文提出一种通用的 TEE 攻击面分类方法. 现代 CPU 执行指令的过程一般分为 5 个阶段: 取指令 (instruction fetch, IF)、指令译码 (instruction decode, ID)、指令执行 (execution, EX)、内存访问 (memory access, MEM) 和结果写回 (writeback, WB). IF 阶段 CPU 将一条指令从内存中取到指令寄存器, 此过程中可能涉及到的组件有 cache、分支预测单元 (branch prediction unit, BPU). 取出指令之后, 计算机进入 ID 阶段, 在该阶段, 指令译码器对取出的指令进行翻译, 识别并区分出不同的指令, 此阶段尚未发现侧信道漏洞. 当指令译码完成之后, 接着进入 EX 阶段, 此阶段会完成指令的具体任务, 此过程可能涉及到的组件有瞬态执行相关部件 (如, reorder buffer, reservation station 以及 retirement buffer). 根据指令需要, 有可能要访问内存, 读取操作数, 这样就进入了内存访问 (MEM) 阶段, 此阶段可能涉及到的组件有 TLB、page table、cache、DRAM 等. WB 阶段会把 EX 阶段的运行结果写回到 CPU 的内部寄存器或内存中. 图 2 描述了 CPU 指令流水线过程中可能用到的组件.

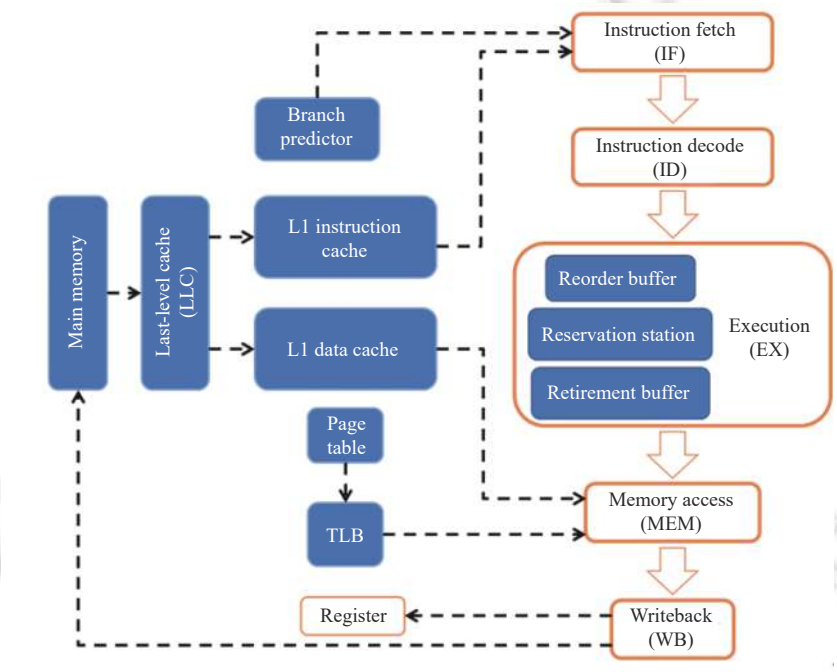


图 2 ARM 体系结构中 CPU 指令流水线过程所需组件

利用上述 TEE 攻击面分类方法, 可以将目前已知的所有 TEE 软件侧信道攻击方法进行系统归类. 目前已经发现的 TEE 软件侧信道攻击方法包括: 瞬态执行攻击、分支预测攻击、基于 TLB 的攻击、cache 侧信道攻击、基于 page fault 的攻击和基于 DRAM 的攻击等, 与这些攻击对应的组件有: 重排序缓冲区、预留站、退休缓冲区、分支预测器、TLB、cache、page table 和 DRAM. 表 2 按照通用的 TEE 攻击面分类方法对目前已知的软件侧信道攻击进行系统分类.

按照上述对软件侧信道攻击的分类标准, 本节剩余部分将分别展示不同的 TEE 软件侧信道攻击. 首先, 根据软件侧信道攻击的 3 个通用步骤 (参考第 1.4 节) 对不同的 TEE 软件侧信道攻击技术进行归纳总结. 具体来说, 将微体系结构组件划分为 CPU 内部的优化组件、分支预测器、存储结构和操作系统功能组件这 4 类, 那么软件侧信道攻击技术与这 4 类微体系结构组件之间的关系如下: 瞬态执行攻击依赖 CPU 内部的优化组件, 分支预测攻击利用分支预测器泄露机密信息, TLB 侧信道攻击、cache 侧信道攻击和基于 DRAM 的侧信道攻击与存储结构相关, 基于 page table 的侧信道攻击通过操作系统功能组件获取机密信息, 组合攻击实质上是物理攻击与软件侧信道攻击相结合形成的攻击手段. 然后分别阐述不同的 TEE 软件侧信道攻击的技术原理和攻击实例.

(1) 瞬态执行攻击[42−54]: 现代处理器中微体系结构层面的优化组件 (例如, 重排序缓冲区) 可以泄露体系结构层面无法访问的信息; 攻击者通过构建数据提取或数据注入两种软件侧信道攻击方法, 将机密信息提取到微体系结构组件中; 最后, 根据不同的微体系结构组件, 构建基于微体系结构的侧信道攻击提取机密信息. 常见的瞬态执行攻击有 Meltdown-type 的数据提取攻击、Spectre-type 的分支诱导攻击以及基于逆向推测性执行 (reversely speculative execution) 的数据注入攻击.

(2) 分支预测攻击[52,55−62]: 在上下文切换期间, 隔离架构不会刷新分支预测器 BPU, 因此 BPU 中遗留了与受害者相关的机密信息; 攻击者以软件形式将 BPU 中受害者的分支信息显式地缓存在其他攻击者能够访问的微体系结构组件中; 通过构建基于微体系结构的侧信道攻击提取受害者的分支信息, 将分支信息与受害者源码中的各种分支结构相互对比, 攻击者可以推断出 TEE 中受害者细粒度的控制流. 常见的分支预测攻击有基于分支目标缓冲区的“Branch Shadowing”攻击和基于定向分支预测器的 BranchScope 攻击.

(3) TLB 侧信道攻击[63,64]: 由于 TLB 组件能够在攻击者和受害者之间共享, 因此 TLB 在理论上存在 TEE 软件侧信道漏洞. 攻击者通过监视隔离架构上下文切换期间 TLB 条目的变化, 可以推断出受害者访问的 TLB 条目.

(4) cache 侧信道攻击[65−74]: 隔离架构中的 cache 软件侧信道攻击类似于在普通处理器 (即, 通用执行环境) 中的 cache 侧信道攻击, 当受害者执行完毕之后, 会将数据遗留在 cache 中; 攻击者通过测量系统事件发生的时间 (例如, cache hit/cache miss 的时间) 或监视受害者在执行期间的访问轨迹等方法来获取隔离架构的机密信息. 常见的 cache 侧信道攻击有 Prime+Count 攻击、缓存定时攻击 (cache-timing attack)、CacheZoom 攻击等.

(5) 基于 page table 的侧信道攻击[75,76]: 攻击者通过操纵受害者的页表条目 (page table entry, PTE) 的某些标志位可以泄露其控制流和数据访问模式, 这类攻击的应用场景一般是 Intel SGX. 首先, 根攻击者操纵受害者飞地的 PTE 的某些标志位 (例如, “present”标志位), 使得受害者飞地再次访问该页时发生 page fault; 然后, 受害者飞地执行完后, 攻击者通过是否发生 page fault 来判断受害者飞地是否访问过上述被操纵的页, 以此推断受害者飞地与机密信息相关的数据流访问模式. 常见的基于 page table 的攻击有 Controlled-Channel 攻击、SPM 攻击等.

(6) 基于 DRAM 的侧信道攻击[77−79]: DRAM 中 row buffer 的作用类似于 cache, 攻击者利用 row buffer 造成的内存访问的时间差推断受害者的内存访问信息. 在 Intel SGX 中, 飞地的所有信息被存储在处理器保留内存 PRM 中的安全页面缓存 EPC 中, 飞地外部的应用程序无法访问 EPC. 因此, 要在隔离架构中构建基于 DRAM 的侧信道攻击, 攻击者和受害者必须满足共享同一个 DRAM row 的条件. 常见的基于 DRAM 的侧信道攻击有 cache-DRAM 攻击.

(7) 组合攻击[80]: 组合攻击是微体系结构攻击与其他攻击相结合而形成的攻击手段, 例如, 缓存侧信道攻击与总线监听技术相结合. 常见的组合攻击有 Membuster 攻击.

表 2　指令流水线过程中所需组件以及对应的 TEE 软件侧信道攻击

指令执行阶段	所需组件	TEE侧信道攻击
IF	BPU	分支预测侧信道[52,55−62]
	cache	cache侧信道[65−74]
	page table	基于page table的攻击[75,76]
ID	译码器	—
	寄存器	
EX	Reorder buffer	瞬态执行攻击[42−54]
	Reservation station	
	Retirement buffer	
MEM	TLB	基于TLB的攻击[63,64]
	page table	基于page table的攻击[75,76]
	cache	cache侧信道[65−74]
	DRAM	DRAMA[77−79]
WB	寄存器	DRAMA[77−79]
	DRAM	

2.1 瞬态执行攻击

现代处理器之所以拥有很高的计算性能, 与其内部复杂的微体系结构优化密切相关. 微体系结构级别的优化会预测控制流甚至重排序指令流, 使得指令流水线时刻保持满载状态, 从而提高了处理器的执行效率. 然而, 这些为了提升 CPU 计算性能而在微体系结构上做出的优化有时可能会导致 CPU 运行时发生错误, 例如, 预测错误. 发生错误之后, CPU 会刷新指令流水线并丢弃在发生错误期间 CPU 处理的指令的所有体系结构级别的状态信息. 换句话说, 从发生错误到 CPU 发现这种错误期间执行的任何指令, 都是暂时执行的, 以后还会被丢弃掉, 所以称这段时期执行的指令叫瞬态执行[42−46]. 瞬态执行的结果永远不会提交到体系结构层面, 但可能会在微体系结构状态中留下痕迹, 攻击者通过微体系结构状态信息就能提取到机密信息, 这就是瞬态执行攻击.

原始的瞬态执行攻击有 Meltdown 攻击和 Spectre 攻击, 它们的攻击目标是普通的处理器 (即, 通用执行环境), 并没有涉及到可信执行环境. 研究人员发现, 可信执行环境中处理器内部微体系结构的优化组件会给 TEE 隔离架构带来安全隐患, 攻击者通过微体系结构的状态信息可以提取 TEE 隔离架构的机密信息. TEE 隔离架构中的瞬态执行攻击可以分为 Meltdown-type 的数据提取攻击、Spectre-type 的分支诱导攻击以及基于逆向推测性执行的数据注入攻击: Meltdown-type 攻击利用了处理器乱序执行 (out-of-order execution, OoOE) 的能力提取体系结构层面无法访问的数据, Spectre-type 攻击利用了处理器推测执行 (speculation execution) 的能力诱导受害者分支偏离正确的执行路径, 逆向推测性执行攻击将攻击者控制的数据注入到微体系结构数据缓冲区或预测器中进而提取机密信息.

● Meltdown-type 攻击

基于 Meltdown-type 的数据提取攻击有 Foreshadow 攻击、ZombieLoad 攻击[48]和 RIDL 攻击[49], 它们利用处理器在约束条件下 (例如, 发生页面错误或微码辅助) 会瞬态读取 L1D cache 或行填充缓冲区 (line-fill buffer, LFB) 中的旧值 (stale value) 的特点, 从而提取受害者机密信息. 下面以 Foreshadow 攻击为例进行介绍.

比利时荷语鲁汶大学的 imec-DistriNet 研究团队提出了一种 Foreshadow 攻击[47], 该攻击利用了 Intel 处理器中乱序执行导致的错误而从 CPU 缓存中窃取飞地的机密信息. Foreshadow 攻击是一种 Meltdown-type 攻击, 基本的 Foreshadow 攻击分为缓存飞地机密信息、瞬态执行和接收机密信息这 3 个阶段. 在 Intel SGX 中, 只有当数据驻留在 L1D cache 中时, 处理器由于乱序执行才会将数据转发给后续的瞬态执行操作. 因此 Foreshadow 攻击首先运行受害者飞地, 在飞地机密信息缓存到 L1D cache 中时利用 SGX-Step 中断技术或超线程技术异步中断飞地将机密信息暂时保留在 L1D cache 中; 在 Foreshadow 攻击的第 2 个阶段, 无特权的攻击者需要通过执行错误指令以触发处理器进入瞬态执行状态 (例如, 普通应用程序通过指针访问受害者飞地的内存). 但在 SGX 中, 普通应用程序访问飞地内存不会产生 page fault 且读取的数据会被伪值取代. 因此攻击者利用“mprotect”系统调用撤销对目标受害者飞地的访问权限, 这样对飞地页面的任何访问都会产生 page fault, 从而为后续指令的执行创造一个瞬态执行窗口. 另外, 与 Meltdown 攻击不同, Foreshadow 攻击在瞬态指令流执行期间, 机密信息会被转换成 oracle 缓冲区中的 slot entry; 第 3 阶段, 处理器发生 page fault, 将调用攻击者的异常处理程序. 该程序会 Reload 所有的 oracle slot, 并计算每个 slot 的加载时间, 其中与机密信息相关的 slot 的加载时间最短. 通过实验测试, 无特权攻击者的成功率为 96.82% (使用 TSX); 通过对 Intel Launch Enclave 进行实际攻击, 经过 13 次 page fault 可以无噪声地提取 128 位密钥; 在对 Intel Quoting Enclave 进行的攻击中, 经过 14 次 page fault 可以提取 128 位密钥, 攻击成功率达 100%.

● Spectre-type 攻击

基于 Spectre-type 的分支诱导攻击[50−53]包括 SgxPectre 攻击和 SpectreRSB 攻击[51]: SgxPectre 攻击利用了处理器中分支预测单元在不同安全域上共享的特点, 使得攻击者容易诱导受害者分支的执行路径, 从而导致推测性执行攻击; SpectreRSB 则利用返回堆栈缓冲区 (return stack buffer, RSB) 在不同安全域上共享的特点, 通过不受信任的攻击者操纵 RSB 使得执行飞地时发生错误推测从而导致推测性执行攻击. 下面以 SgxPectre 攻击为例进行介绍.

俄亥俄州立大学的研究团队提出了一种 SgxPectre 攻击[53], 它利用分支指令解析 (resolution) 的延迟与推测性

执行内存引用之间存在的瞬态执行攻击窗口来生成侧信道痕迹从而读取内存内容. SgxPectre 攻击是一种 Spectre-type 的攻击, 它可以分为 5 个步骤: 毒化分支目标缓冲区 (branch target buffer, BTB)、扩大瞬态攻击窗口、设置飞地的寄存器、飞地小工具执行以及侧信道攻击提取机密信息. 第 1 步, 攻击者构造一个间接跳转指令, 该指令的源地址和目标地址的低 32 位与受害者飞地中的目标分支指令地址相对应. 利用此跳转指令训练 BTB, 使得当受害者飞地在目标分支指令处执行时, 会瞬态更改飞地代码的控制流到攻击者提前选择好的目标地址处; 第 2 步, 攻击者利用具体的方法扩大瞬态执行攻击的窗口, 例如, 通过刷新受害者的分支目标地址以延长分支目标指令退休的时间等. 通过这些方法可以最大限度地延长分支指令退休时间并减少代码小工具瞬态执行的时间, 从而扩大瞬态执行窗口; 第 3 步, 在 EENTER 进入飞地之前, 攻击者设置小工具代码中使用的飞地寄存器, 使它们能够读取目标飞地内存的机密信息. 这些寄存器在进入飞地后会一直由攻击者控制, 直到被飞地代码修改为止; 第 4 步, 飞地代码开始执行, 由于 BTB 已经被攻击者毒化, 因此在解析源地址处的 ret 指令时, 飞地会根据 BTB 中的 entry 推测性地执行目标地址. 在目标地址处的代码利用第 3 步设置的寄存器值对目标地址的机密信息进行编码, 这些编码值与受攻击者控制的数组 entry 相对应, 这样目标地址处的数据就等于数组 entry; 第 5 步, 使用 Flush+Reload 侧信道攻击提取机密信息. 实验表明, SgxPectre 攻击可以从飞地中窃取封装密钥和证明密钥, 从而严重破坏 Intel SGX 的机密性.

● 注入式攻击

基于逆向推测性执行的数据注入攻击有加载值注入 (load value injection, LVI) 攻击, 它利用处理器在约束条件下 (例如, 页面偏移量相同) 会导致瞬态转发 L1D cache、LFB、存储缓冲区 (store buffer, SB) 中 stale value 的特点, 从而开发了逆向推测性的数据注入攻击.

比利时荷语鲁汶大学的研究团队提出了基于 LVI 的新型攻击[54], 它采用基于注入的方法来逆向研究推测性执行导致的微体系结构数据泄露. LVI 攻击是一种 Meltdown-type 的攻击, 完整的 LVI 攻击分为 4 个阶段: 准备微体系结构缓冲区、错误的瞬态转发、小工具编码机密信息和侧信道分析接收机密信息. 在第 1 阶段, LVI 攻击者以微体系结构组件作为瞬态数据注入源, 将攻击者控制的数据填充到相应的微体系结构组件中; 第 2 阶段, 由于 Intel CPU 中的优化组件 (例如 L1D cache 或存储缓冲区等) 在满足一定的约束条件 (例如, 对于存储缓冲区, 其约束条件为: 当可信加载导致页面错误时, 如果该加载操作的页面偏移量与最近的存储操作的页面偏移量一致, 那么该加载会从存储缓冲区中获取攻击者的数据) 时可以导致错误的瞬态转发, 因此攻击者在受害者飞地之外通过修改飞地的堆栈 PTE 等操作, 使飞地在下一次执行与堆栈相关的操作时产生页面错误或微码辅助. 当受害者飞地再次访问相应的堆栈或页时会导致错误的瞬态转发, 从而将攻击者的数据注入到受害者飞地的瞬态执行流中; 第 3 阶段, 攻击者必须在受害者飞地代码中找到满足 LVI 攻击条件的全部小工具, 它们能将第 2 阶段注入的攻击者的数据进行编码或重定向控制流, 并将计算结果保留在微体系结构组件中; 第 4 阶段, 根据微体系结构组件进行相应的侧信道分析, 从而恢复受害者飞地的机密信息. LVI 攻击很难被防御, 能抵御 Meltdown 攻击的处理器也无法防御 LVI 攻击 (因为零值也可以被 LVI 攻击者利用), 该团队展示了基于编译器的防御措施会造成 2–19 倍的运行开销.

2.2 分支预测攻击

现代计算机按照流水线方式执行指令. 处理器在处理分支指令时, ID 阶段才能知道该指令是否为条件分支指令, 待分支指令通过了 EX 阶段才会把下一条指令送入流水线的 IF 阶段, 这大大降低了处理器的处理效率. 分支预测是现代处理器在程序分支指令执行前预测其结构的一种机制, 采用分支预测, 处理器会提前预测分支的走向, 并且基于预测结果进行后续的取指和译码工作. 然而, 在上下文切换期间, 分支预测器 BPU 并不会被处理器刷新. 也就是说, BPU 中可能保留有隔离架构的机密信息, 这些信息会被攻击者以软件侧信道方式窃取, 从而导致严重的安全隐患[55−62].

在现代处理器中, BPU 一般由两部分组成, 分别是 BTB 和定向分支预测器 (directional branch predictor): BTB 预测分支的目标地址, 定向分支预测器预测分支的方向 (taken 或 not taken). BTB 和定向分支预测器都能泄露隔离架构的机密信息.

美国佐治亚理工学院的研究团队提出了“Branch Shadowing”攻击[56], 它利用 BTB 中遗留的受害者飞地的分支

目标信息来泄露飞地机密信息. 该攻击首先通过 Branch Shadowing 代码引入组冲突, 然后探查共享的分支目标缓冲区 BTB 条目, 并根据执行时间确定目标分支指令的历史记录是否存储在 CPU 内部的 BTB 中, 最后通过最近分支记录 (last branch record, LBR) 间接地显示 enclave 的分支信息. “Branch Shadowing”攻击可以分为 4 步: 第 1 步, 代码分析阶段, 攻击者通过详细分析受害者飞地的源代码, 找到所有类型的分支及其目标地址; 第 2 步, 构建“Branch Shadow”, 根据源代码中的分支类型 (条件分支、无条件分支以及间接分支), 为每种分支编写与之对应的影子代码, 不同的分支对应的影子代码不同; 第 3 步, 探测分支历史, 在 Intel SGX 中, 由于 BPU 被不同安全域的应用程序共享, 并且在上下文切换期间没有清除分支历史, 使得 BPU 中遗留了飞地的机密信息. 为了从 BPU 中获得受害者飞地的分支历史信息, 攻击者首先执行飞地代码. 当执行完某种分支指令后, 与该指令相关的信息被缓存在 BPU 中. 然后攻击者中断飞地执行, 转而执行与该分支对应的影子代码. 影子代码中分支的执行结果与 BPU 中的分支历史密切相关: 如果飞地代码的分支指令发生跳转 (taken), 那么处理器会根据此分支历史预测性地执行影子代码, 影子代码的分支指令也会发生跳转. 预测执行的结果会通过 LBR 显式地记录下来, 根据 LBR 的记录结果可以知道受害者飞地细粒度的分支执行流; 第 4 步, 通过分析这些分支信息推断飞地执行的全部控制流. 实验结果表明, 该攻击能以 99.8% 的准确率提取 RSA 密钥.

美国威廉与玛丽学院的研究团队开发的 BranchScope 攻击[52]是另一类基于定向分支预测器的侧信道攻击, 它与基于 BTB 的攻击类似. 攻击者通过操纵定向分支预测器来强制攻击者与受害者分支之间发生冲突, 从而推断受害者条件分支指令的方向. 此外, NCC Group 的研究人员[55]将 cache 侧信道攻击与分支预测侧信道攻击相结合, 开发出一种针对 ARM TrustZone 的低噪声、高时间分辨率和高空间分辨率的攻击, 并成功从高通公司硬件支持的密钥库中恢复了 256 位密钥.

2.3 TLB 侧信道攻击

现代处理器利用转换后备缓冲区 (translation lookaside buffer, TLB) 来缓存虚拟地址与物理地址之间的转换关系. 在某些设计中, TLB 是竞争性共享的微体系结构资源, 而任何竞争性共享的资源都能产生侧信道. 目前, 学术界没有出现实际的针对 TEE 的 TLB 侧信道攻击案例, 但 TLB 在理论上是存在侧信道漏洞的.

荷兰阿姆斯特丹自由大学的研究团队首先提出了一种针对 Intel 处理器 (非 SGX) 的 TLBleed 攻击[64]. 在此基础上, 中国科学院信息工程研究所的研究团队[63]系统地分析了 Intel SGX 的侧信道攻击面. 其中 TLB 可能存在的侧信道攻击面有两类: 第 1 类攻击面是由于处理器启用超线程时 TLB 能够在受害者进程与攻击者进程之间共享. 这样, 与受害者进程相关的 TLB entry 会受到攻击者进程的影响从而产生侧信道漏洞; 第 2 类攻击面是由于 Intel 处理器通过启用 TLB 条目中的“进程上下文标识符 (process-context identifier, PCID)”字段来允许在上下文切换期间有选择地刷新 TLB 条目. 这样, 处于飞地之外的攻击者可以在上下文切换期间根据已刷新的 TLB 条目推断飞地进程的执行. 然而, 该研究团队没有针对该 TLB 攻击面构建基于 TLB 的侧信道攻击, 只是将 TLB 攻击面作为刷新 TLB 的手段应用到该团队开发的基于 page table 的 SPM (sneaky page monitoring) 攻击中.

2.4 cache 侧信道攻击

在现代的 CPU 架构中, 处理器的处理能力和内存访问之间的速度不匹配, 这个问题严重制约了计算机处理能力的提高. 缓存的出现就是为了匹配 CPU 和内存之间的速度问题, 即计算机把处理器最近用过的数据存在 cache 中, 当处理器再次访问该数据时可以在 cache 中直接获取. 基于 cache 的攻击大都利用了传统 cache 侧信道攻击技术, 通过测量受害者的执行时间或监视受害者在执行期间的访问轨迹等方法来获取机密信息[16,65−74].

美国亚利桑那州立大学的研究团队提出 Prime+Count 缓存侧信道攻击技术[16], 该技术能在 ARM TrustZone 中构建跨世界的隐蔽信道. 该攻击分为 3 步: 第 1 步, Prime 阶段, 攻击者通过连续访问自身的地址空间从而使攻击者的数据完全填充 cache; 第 2 步, 等待受害者执行, 在受害者执行过程中如果 TA 发生内存访问, 将会逐出攻击者 cache 行中的数据并将 TA 的数据填充进去; 第 3 步, Count 阶段, 攻击者再次访问其自身的地址空间, 并根据 PMU 对发生变化的 cache 行进行计数. 这种攻击技术空间分辨率较差但噪声低. 德国埃朗根-纽伦堡大学的研究团队证明了 Intel SGX 容易受到缓存定时攻击[66]. 该团队在 Intel 平台上启用了超线程技术, 攻击者和受害者是位于

同一个进程的不同线程. 受害者线程在飞地中运行加密程序, 产生加密密钥. 攻击者线程是飞地之外的普通应用程序, 它通过读取性能监视计数器 (performance monitoring counters, PMC) 来进行 Prime+Probe 缓存侧信道攻击. 实验表明, 此缓存定时攻击能以不到 10 s 的时间获得 Intel SGX 飞地的 AES 密钥.

另外, 还有一些团队也做出了与根级缓存定时攻击 (root-level cache-timing attack) 类似的成果, 例如, 美国伍斯特理工学院的研究团队提出的用于分析 SGX 飞地内存访问的攻击 CacheZoom[67], 德国达姆施塔特工业大学的研究团队提出的针对 Intel SGX 的缓存侧信道攻击[74]等.

2.5 基于 page table 的侧信道攻击

当应用程序访问物理内存时, CPU 会发出逻辑地址 (虚拟地址) 交由 MMU 进行内存寻址, 找到物理内存上的内容. 当要访问的页面不在内存中时, 处理器发生 page fault, 由 OS 负责将缺失的部分装入物理内存. 基于 page table 的攻击大都针对 Intel SGX 隔离架构. 在 SGX 中, 内存管理留给了不可信的 OS, OS 可以在 enclave 执行的任意时刻强制设置飞地页面为不可访问. 此时, enclave 对该页面的任何访问都会触发 page fault, 攻击者根据发生 page fault 的飞地页面可以获取与机密信息有关的 enclave 控制流和数据访问模式.

中国科学院信息工程研究所的研究团队[63]按照飞地页面 PTE 中不同的标志位, 将 Intel SGX 中基于 page table 的侧信道漏洞系统地分为 4 类: 第 1 种漏洞是根攻击者通过清除受害者飞地 PTE 中的“present”标志位来收集飞地页面的访问踪迹, 从而推断出依赖于机密信息的控制流或数据访问模式; 第 2 种漏洞是攻击者通过监视 PTE 中的“accessed”标志位的更新检测受害者飞地访问的页面; 第 3 种漏洞是监视 PTE 中的“dirty”标志位, 根据该标志位是否发生变化推断受害者飞地对页面的内存写操作; 第 4 种漏洞是利用 PTE 中的其他位, 例如一些保留位, 攻击者利用这些位来触发 page fault 以探测飞地页面的机密信息. 美国德克萨斯大学奥斯汀分校的研究团队提出了 Controlled-Channel 侧信道攻击[75,76], 它利用第 1 种漏洞造成的 page fault 来探测飞地页面的数据访问模式. 另外, 该团队开发了一种利用 PTE 中“accessed”标志位的基于 page table 的侧信道攻击, 并将该侧信道攻击称为 SPM 攻击. SPM 攻击者是飞地外运行的系统级进程, 它可以检查每个 page table 条目的“accessed”标志位. 当飞地访问该页面时, “accessed”标志位被置 1, 攻击者会将 PTE 的置位信息记录下来, 然后将其重置为 0, 从而达到反复检测飞地页面访问踪迹的目的. 然而, 飞地在第 1 次页表遍历之后, 相应的 PTE 信息会被缓存在 TLB 中. 下次飞地再访问该页面时, PTE 中的“accessed”标志位不会发生变化, 攻击者也无法探测飞地对同一页面的第 2 次页面访问. 为了解决此问题, 攻击者在不同的 CPU 核中生成处理器间中断 (inter-processor interrupt, IPI) 以导致飞地进入异步飞地退出 (asynchronous enclave exits, AEX), 从而刷新与当前 PCID 关联的 TLB 条目. 上述 SPM 攻击称为基本的 SPM (basic SPM, B-SPM) 攻击. 由于 B-SPM 存在时间分辨率不高以及对同一页面的重复访问会触发较高的中断频率等问题, 该团队提出了相应的优化方案: 时间增强型 SPM (timing-enhanced SPM, T-SPM) 攻击通过测量飞地中不同页面上两条指令之间的执行时间来推断页面中分支的执行方向; 超线程 SPM (hyper threading SPM, HT-SPM) 攻击通过打开超线程功能即可清除 TLB 而不触发任何中断.

2.6 基于 DRAM 的侧信道攻击

基于 DRAM 的攻击与 DRAM 本身的结构有关. DRAM 通常由多个 channel 组成, 一个 channel 被分为许多个 DIMM, 每个 DIMM 通常有两个 rank, 不同 rank 上有许多 DRAM chip, 每个 chip 可以分为多个 bank, 一个 bank 可以看成一个矩阵的形式, 每个 bank 中都有一个 row buffer 用来缓存最近访问过的某个 row. 其中每个 row 通常为 8 KB, 可以由两个 4 KB 的页共享. 同一个 bank 中的 row 是否位于 row buffer 中会给内存访问造成时间差异, 根据时间差异极容易判断出当前受害者的内存访问情况[63,77−79].

奥地利格拉茨工业大学的研究团队首次提出了基于 DRAM 的侧信道攻击: DRAMA 攻击[77]. 基于此攻击, 中国科学院信息工程研究所的研究团队[63]在 Intel SGX 隔离环境中成功实施了针对 Intel SGX 的跨飞地 DRAMA 攻击和 cache-DRAM 攻击. 跨飞地 DRAMA 攻击可以捕获飞地进程是否访问了某个物理地址的信息. 在实施攻击之前, 攻击者通过逆向工程 DRAM 的寻址算法, 在同一个 DRAM bank 中分别找到受害者飞地的目标地址 a、与 a 共享同一个 row 的地址 b 以及不与 a 共享同一个 row 的地址 c. 实施 DRAMA 攻击具体可分为 3 步: 首先, 攻击者

多次访问物理地址 c, 使得 row buffer 中缓存了被地址 c 占据的 row; 其次, 等待受害者执行; 最后, 攻击者访问物理地址 b 并测量其访问时间. 如果 b 的访问时间较短, 说明此时 row buffer 中缓存的是被地址 a 和 b 占据的 row, 因此能够判断受害者飞地访问了目标地址 a. 然而, 跨飞地 DRAMA 攻击存在很多诸如空间粒度粗糙和噪声高等限制条件. 为了解决这些问题, 该团队在上述攻击的基础上提出了 cache-DRAM 攻击. cache-DRAM 攻击的攻击者是两个线程, 其中一个线程在飞地之外, 执行 Prime+Probe 侧信道攻击, 该攻击的目的是将缓存在 cache 中的目标地址 a 清除出去, 以便飞地每次都能在 DRAM 中访问地址 a; 另一个线程是恶意飞地程序, 与受害者飞地程序在同一个飞地中,用于执行跨飞地 DRAMA 攻击.这样,通过测量地址 b 的访问时间即可确定受害者飞地程序是否访问了目标地址 a

2.7 组合攻击

除了基于微体系结构的侧信道之外, 研究人员将其他类型的攻击与微体系结构攻击相结合, 构成一种新颖的组合攻击. 这类攻击吸收了微体系结构攻击的优势, 足以攻破当前 TEE 隔离架构而泄露大量敏感信息. 组合攻击最典型的例子就是将物理攻击与 cache 侧信道攻击相结合从而破坏了 Intel SGX 的 Membuster 攻击.

美国加利福尼亚大学伯克利分校的研究团队通过片外监听内存总线、片上制造内存访问发生 cache miss 的方式构造了一种 Membuster 攻击[80]. Membuster 攻击引入了关键页面白名单、缓存压缩、基于 oracle 的模糊匹配算法这 3 种技术, 从而使攻击具有无干扰、细粒度、隐蔽的特性. Membuster 能以很高精度窃取 enclave 中应用程序的机密信息. 首先, 在受害者计算机系统 boot 之前, 攻击者需要在 DIMM 插槽上安装插入器并等待受害者执行, 然后根据受害者的执行情况收集 DRAM 信号并通过插入器将收集到的信号存储起来; 其次, 利用从 DRAM 插入器收集到的信息, 对 Intel CPU 的寻址算法进行逆向工程, 得到物理地址与 DRAM 之间的映射关系, 之后, 通过修改 SGX 驱动程序, 获取物理地址与虚拟地址之间的映射关系; 然后, 利用关键页面白名单、跨核 Prime 和缓存压缩技术来增加缓存未命中率, 从而导致飞地访问 DRAM 的频率增加; 最后, 通过基于 oracle 的模糊匹配算法降低噪声, 从而获取飞地细粒度的机密信息.

3 威胁模型

可信执行环境通过减少可信计算基 (trusted computing base, TCB) 限制安全敏感的应用程序与恶意应用程序之间的交互, 从而保证计算设备在隔离的可信执行环境中存储和处理机密信息. TEE 中的 TCB 通常仅包含 CPU 或少量代码. 在 ARM TrustZone 中, CPU、可信操作系统和监视器 Monitor 是可信的组件, 除上述组件之外, 其余组件均被视为受攻击者控制而被排除在 TCB 之外. Intel SGX 的可信计算基更小, 仅包括 CPU, 除 CPU 之外的其他所有组件都不可信. 尽管通过减小可信执行环境的 TCB 可以减少 TEE 中的软件漏洞, 但无法抵御 TEE 软件侧信道攻击.

当前针对 TEE 的软件侧信道攻击中, 包含 3 类不同能力的攻击者, 分别是用户级攻击者、内核级攻击者以及物理攻击者. 用户级攻击者不需要具备任何特权级权限, 仅通过用户层的恶意应用程序就能发起攻击; 内核级攻击者要求获得系统内核层的权限以执行特权攻击; 物理攻击者则需要将攻击设备物理接触目标设备.

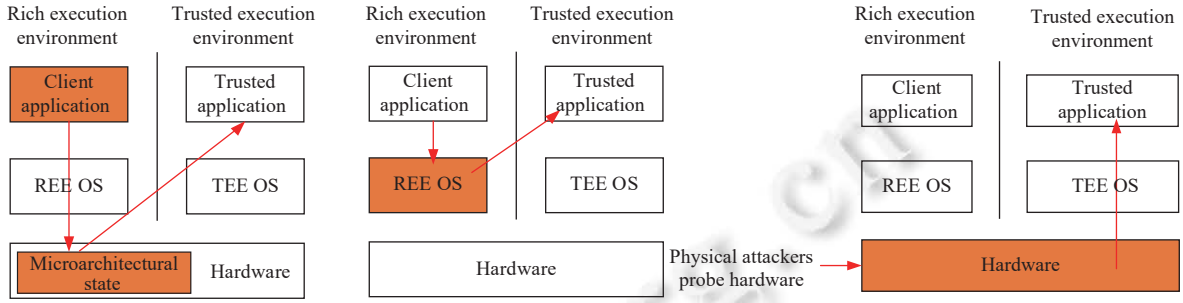
(1) 用户级攻击者

用户级攻击者通常是一个无特权的应用程序, 它不必具备任何特权级的权限, 通过执行一段恶意代码或执行系统调用就能将受害者机密信息泄露到微体系结构中, 用户级攻击者模型如图 3(a) 所示. 用户级攻击者的一个显著特点是: 攻击者只需运行非特权代码, 就能破坏 TEE 中不同特权级和地址空间之间的内存隔离. Foreshadow 攻击、ZombieLoad 攻击、RIDL 攻击和 DRAMA 攻击都是用户级攻击者. 以 Foreshadow 攻击为例, 飞地外的恶意应用程序利用“mprotect”系统调用清除飞地 PTE 中的当前位, 从而使对该页的任何访问都导致 page fault, 操作系统捕获到该 page fault 之后转向执行异常处理函数, 系统执行异常处理函数的这段时间给攻击者创造了一个乱序执行的窗口.

(2) 内核级攻击者

内核级攻击者通常会控制目标设备的特权级软件, 通过调用操作系统提供的特权指令和内核功能对可信应用

程序发起攻击, 内核级攻击者模型如图 3(b) 所示. 内核级攻击者的特点是需要控制操作系统内核. 常见的内核级攻击者有 SgxPectre 攻击、LVI 攻击、Branch Shadowing 攻击、BranchScope 攻击、Prime+Count 缓存侧信道攻击、CacheZoom 攻击、Controlled-Channel 攻击、SPM 攻击等. 以 Branch Shadowing 攻击为例, 攻击者是恶意的操作系统, 它通过操纵飞地进程的虚拟地址空间, 可以创建与目标飞地中分支指令相冲突的分支影子指令. 在受害者飞地运行期间, 攻击者通过频繁中断目标飞地的执行以运行分支影子代码, 从而推测出受害者内部细粒度的控制流.



(a) 用户级攻击者 (b) 内核级攻击者 (c) 物理攻击者

图 3 TEE 中 3 种不同能力的攻击者模型

(3) 物理攻击者

物理攻击者需要攻击设备能够物理探测被攻击设备的硬件, 然后利用收集到的信息推断受害者的机密信息, 物理攻击者模型如图 3(c) 所示. 物理攻击者与上述两类攻击者的不同之处在于: 用户级攻击者和内核级攻击者都是软件攻击, 而物理攻击者需要利用物理器件探测设备的信息. 典型的物理攻击者是 Membuster 攻击, 它通过收集受害者执行过程中 DRAM 产生的信号变化, 对 Intel SGX 的寻址算法进行逆向工程, 再利用 cache 侧信道攻击获取飞地的细粒度机密信息.

4 TEE 侧信道攻击

针对 TEE 软件侧信道攻击的防御措施可以从硬件层面和软件层面两方面考虑. TEE 软件侧信道攻击大都利用了微体系结构组件的优化功能, 因此重新设计硬件结构能从根本上解决 TEE 软件侧信道攻击带来的问题. 然而, 硬件层面的防御措施短期内无法部署到计算设备中, 因此安全研究人员提出了一些软件层面的防御措施, 用于短期内部署到计算设备中来缓解 TEE 侧信道攻击.

本节按照第 2 节对 TEE 侧信道攻击面的分类, 分别介绍不同 TEE 软件侧信道攻击的防御措施. 表 3 总结了不同的 TEE 软件侧信道攻击对应的防御措施, 与硬件层面的对策相比, 软件层面的防御措施往往会引入较高性能开销. 下面将从硬件层面和软件层面两部分分别探讨软件侧信道攻击的防御措施.

4.1 瞬态执行侧信道防御措施

由于瞬态执行攻击滥用了现代处理器中用于提升计算效率的微体系结构组件, 因此软件层面的对策 (例如在瞬态执行指令之后插入 lfence 屏障) 或硬件层面的微码更新无法彻底抵御瞬态执行攻击, 硬件层面通过更改处理器硅片结构等措施能从根本上抵御瞬态执行侧信道攻击.

● 硬件层面

(1) 更新微码: 通过发布微码更新, 可以对具有瞬态执行漏洞的硬件打补丁. 例如, 通过在飞地退出时刷新 L1 cache 可以抵御 Foreshadow 攻击[47]、通过刷新行填充缓冲区并在刷新操作完成之后增加一个 lfence 屏障可以抵御 ZombieLoad 攻击[48]和 RIDL 攻击[49]、带有微码补丁的 Intel 处理器会在进入或恢复飞地时刷新分支目标缓冲区从而抵御 SgxPectre 攻击[51].

(2) 硅片级更改: 在硅片级别重新设计处理器[47,49,50,52−54]以禁止瞬态执行指令进行瞬态计算, 从而使得错误加载不会瞬态转发任何数据. 该方案可以从根本上解决当前处理器中一些优化技术带来的侧信道, 从而保护 TEE 隔

离架构免受瞬态执行的影响. 这是一种长期解决方案, 其特点在于能从根本上解决瞬态执行侧信道, 缺点是短期内无法部署.

(3) 修改分支预测器: SpecCFI[81]通过在 BTB 条目中增加一个与目标地址相关的 CFI (control-flow integrity) 标签或划分出一块内存区域充当影子调用堆栈 (shadow call stack, SCS), 从而抵御基于 BTB 和基于 RSB 的 Spectre-type 攻击.

表 3 不同的 TEE 软件侧信道攻击对应的防御措施

TEE侧信道攻击类型	硬件层面		软件层面	
	防御技术	方法概述	防御技术	方法概述
瞬态执行攻击	更新微码[49,52−54]	通过微码更新来发布硬件补丁	修改源码[47]	在源码中每条瞬态执行指令之后插入lfence屏障
	修改分支预测器结构[81]	在BTB中增加一个标签		
	硅片级更改[47,49,50,52−54]	硅片级别修改处理器结构以禁止瞬态执行指令进行瞬态计算		
分支预测侧信道攻击	更新微码[51,56,60]	通过微码更新来发布硬件补丁	Zigzagger技术[56]	利用CMOV指令将源码中的条件分支转换为无条件分支
	修改分支预测器结构[51]	对BPU分区	if-conversion技术[51]	利用CMOV指令消除条件分支
		静态预测安全敏感的分支	禁用性能计数器[60]	禁用高精度性能计数器以模糊时间
TLB侧信道攻击	修改TLB硬件结构[63,64,82]	按照安全域对TLB进行分区	禁用超线程技术[63]	禁用处理器中的超线程功能
cache侧信道攻击	cache分区[13,83−86,97,98]	对cache分区使得受害者和攻击者无法共享同一个cache行	时间差消除技术[17,88−94]	阻止攻击者精确测量受害者内存访问的时间
	随机化技术[85,87,96]	随机化DRAM到cache之间的映射关系	基于检测的技术[17,18]	利用性能计数器检测系统事件
			基于隔离的技术[95]	利用软件缓存锁定技术对L1缓存进行隔离
基于page table的攻击	硬件隔离技术[63,99−103]	重新设计硬件使得安全敏感的页面访问模式对恶意OS不可见	软件隔离技术[63,101]	利用编译器技术将依赖于机密信息的执行进行隔离保护
			基于检测的技术[79,102]	检测AEX发生的频率以判断是否出现侧信道攻击
			重写源码[68,75,76]	修改依赖于机密信息的内存访问指令
			Heisenberg技术[103]	飞地执行前利用TLB预加载飞地将要使用的页面转换关系
基于DRAM的攻击	更改DRAM和处理器架构[63]	重新设计DRAM和CPU架构	时间差消除技术[63,77−79]	阻止攻击者精确测量受害者内存访问的时间
			基于检测的技术[63,77]	利用性能计数器检测系统事件
组合攻击	更改DRAM和处理器架构[80]	重新设计DRAM和CPU架构	ORAM技术[80]	利用ORAM技术隐藏实际的内存访问模式

● 软件层面: 基于编译器的软件防御措施可以缓解 LVI 攻击[47]. 通过分析飞地源码, 在每条瞬态执行指令之后插入 lfence 屏障, 可以序列化处理器流水线. 这能确保瞬态执行指令加载的值不是攻击者控制的注入值, 而是体系结构上正确的值.

4.2 分支预测侧信道防御措施

分支预测侧信道攻击滥用了现代处理器中的分支预测单元, 硬件层面通过修改 BPU 结构或更新微码以使安

全域上下文切换期间刷新 BPU 能从根本上抵御此攻击, 而软件层面通过消除受害者源码中的条件分支指令能缓解此攻击.

● 硬件层面

(1) 更新微码: 通过更新微码[51,56,60]使得受害者飞地在上下文切换期间刷新 BPU 中与受害者机密信息相关的分支信息.

(2) 修改 BPU 结构: 通过修改 BPU 的结构[51]缓解分支预测攻击可以从两方面考虑: 第一, 对 BPU 进行分区. 分区之后, 攻击者无法访问与受害者有关的 BPU 条目, 从而无法探测受害者的分支信息; 第二, 静态预测安全敏感的分支. 将受害者源码中的所有安全敏感的分支标记出来, 这些分支只能被静态预测且每次预测之后不能更新 BPU 条目, 这样 BPU 中永远不会出现与安全敏感的分支相关的信息, 从而抵御分支预测侧信道攻击.

● 软件层面

(1) Zigzagger 技术: 利用 Zigzagger 技术[56]可以缓解 Branch Shadowing 攻击. Zigzagger 技术利用 CMOV 指令将受害者源码中的条件分支转换为 Zigzagger 蹦床 (trampoline) 中的无条件分支, 经过蹦床多次跳转才能到达目标地址. 与条件分支相比, 无条件分支的执行难以被攻击者分辨.

(2) if-conversion 技术: 基于编译器的 if-conversion 技术[51]通过在受害者程序中消除条件分支以抵御分支预测攻击. 与 Zigzagger 类似, if-conversion 也使用 CMOV 指令将条件分支指令转换为顺序执行的代码, 通过消除条件分支结果对机密信息的依赖性来保护敏感数据. 利用 if-conversion 删除条件分支指令可以缓解 BranchScope 攻击.

(3) 禁用性能计数器: 通过禁用高精度的性能计数器[60]等方法可以使攻击者难以准确读取受害者的执行时间, 从而无法判断分支执行的方向.

4.3 TLB 侧信道防御措施

中国科学院信息工程研究所的研究团队在基于 page table 的 SPM 攻击中利用了 TLB 刷新技术, 在受害者飞地每次内存访问之后利用超线程技术刷新 TLB, 从而提高 SPM 攻击的成功率和准确性[63]. 因此, 硬件层面可以通过重新设计 TLB 结构抵御此侧信道攻击, 软件层面通过禁用超线程技术可以防止攻击者刷新 TLB.

● 硬件层面: 通过更改 TLB 硬件结构[82]能够抵御大多数 TLB 侧信道攻击, 例如, 根据不同安全域对 TLB 进行分区.

● 软件层面: 攻击者利用超线程技术[63]可以监视受害者飞地已访问的 TLB 条目或通过刷新 TLB 为其他类型的攻击 (SPM 攻击) 创造条件, 因此可以禁用超线程技术以防止攻击者对 TLB 进行恶意操作, 从而缓解 TLB 侧信道漏洞.

4.4 cache 侧信道防御措施

TEE 隔离架构中 cache 侧信道攻击十分普遍, 也有许多针对该攻击的防御措施[83−98], 其中硬件层面的防御措施能从根本上抵御 cache 侧信道漏洞, 例如分区技术和随机化技术等; 软件层面的对策包括时间差消除技术、隔离技术以及检测技术.

● 硬件层面

(1) cache 分区: 对缓存分区将使得攻击者与受害者无法共享 cache 行, 从而提供强大的隔离缓存机制来抵抗缓存侧信道攻击, 这类防御措施有 STEALTHMEM 的 cache 行锁定技术[84]、基于 cache 组分区[13]的页面着色技术 (page coloring)、基于缓存 way 分区[13]的缓存分配技术 (cache allocation technology, CAT)、Catalyst 技术[86]、DAWG (dynamically allocated way guard) 技术[83]和缓存分区锁定 (partition-locked cache, PLcache) 技术[85]等.

(2) 随机化 (randomization) 技术: 将内存到缓存之间的映射关系随机化可以增加攻击者找到与受害者地址相冲突的 cache 组的难度, 这类防御措施有随机排列缓存 (random-permutation cache, RPcache) 技术[85]和 ScatterCache 技术[87]等.

● 软件层面

(1) 时间差消除技术: cache 侧信道攻击会根据受害者访问内存的时间差异来推断机密信息, 因此可以利用时

间差消除技术 (例如, 注入噪声) 阻止攻击者测量受害者内存访问的时间. 这类防御措施包括噪声注入技术[89]、禁用高精度计时器[88−94]、恒定时间 (constant-time) 技术[87]等.

(2) 基于检测的技术: cache 侧信道攻击可以导致大量的缓存未命中等系统事件发生, 利用性能计数器[17,18]检测系统事件 (例如, cache hit 或 cache miss) 能够判断 cache 侧信道攻击是否发生.

(3) 基于隔离的技术: SmokeBomb 技术[95]是一种典型的基于隔离的技术, 它利用软件缓存锁定技术将 L1 缓存作为每个进程的私有空间, 只用于安全敏感的操作, 从而防止攻击者通过测量受害者访问时间获取机密信息.

4.5 基于 page table 的侧信道防御措施

基于 page table 的侧信道攻击利用页表 PTE 的标志位泄露受害者的内存访问模式. 硬件层面通过硬件隔离技术抵御此攻击, 软件层面的防御措施主要包括软件隔离技术、检测技术、重写源码以及 Heisenberg 技术等.

● 硬件层面: 硬件层面的防御措施主要是硬件隔离技术, 硬件隔离技术通过重新设计硬件使得安全敏感的页面访问模式对恶意的操作系统不可见, 从而抵御基于 page table 的侧信道攻击. 这类防御措施包括自分页 (self-paging) 技术[100]和 Sanctum 技术[99], 这两种技术都可以为 SGX 飞地提供独立的 page table.

● 软件层面

(1) 软件隔离技术: 利用编译器技术将所有依赖于机密信息的执行隔离保护, 使得恶意应用程序无法探测受害者飞地的内存访问模式. 这类防御措施主要有 T-SGX 技术[101]和确定性复用 (deterministic multiplexing) 技术: T-SGX 技术利用硬件事务内存 (transactional memory) 隔离保护受害者飞地; 确定性复用技术将依赖于机密信息的代码和数据放在一个代码页中顺序执行, 从而导致攻击者检测到的 page fault 序列全部相同.

(2) 基于检测的技术: 基于 page table 的侧信道攻击会出现大量的 AEX, 因此可以通过检测 AEX 发生的频率来抵御该攻击. 这类防御措施主要有 Déjà Vu 技术[102].

(3) 重写源码: 受害者源码中存在依赖于机密信息的内存访问指令, 通过重写可信应用程序代码可以消除或隐藏内存访问模式对敏感数据的依赖性.

(4) Heisenberg 技术: Heisenberg 技术[103]可以全面抵御基于 page table 的侧信道攻击, 该技术要求 TLB 在恢复飞地执行时预加载目标飞地的页面转换关系, 使得飞地在内存访问时不会出现页表遍历, 从而防止发生 page fault 以抵御基于 page table 的侧信道攻击.

4.6 基于 DRAM 的侧信道防御措施

由于计算机 DRAM 中的 row buffer 会对内存访问造成时间差异, 因此只能从硬件层面消除基于 DRAM 的侧信道漏洞. 软件层面通过检测系统事件或注入噪声等方法可以检测或缓解该攻击[63].

● 硬件层面

通过更改 DRAM 和处理器架构可以从根本上解决基于 DRAM 的侧信道攻击.

● 软件层面

(1) 基于检测的技术: 基于 DRAM 的侧信道攻击会产生大量的缓存未命中, 因此通过性能计数器检测系统事件可以检测该攻击.

(2) 时间差消除技术: 与 cache 侧信道攻击类似, 基于 DRAM 的侧信道攻击也需要测量不同内存访问之间的时间差异才能推断受害者的机密信息, 因此, 可以通过禁用高精度的计时器或向攻击过程中注入噪声使测量结果精度下降, 从而缓解该攻击.

4.7 组合攻击的防御措施

Membuster 攻击[80]是物理攻击与软件侧信道攻击的结合, 因此其防御措施可以从两个方面着手: 一方面通过修改硬件等对策使得物理攻击难以窃取受害者的物理地址, 另一方面利用 ORAM 技术隐藏受害者的内存访问模式.

● 硬件层面: 硬件层面必须要更改 DRAM 和 CPU 架构以实现对数据总线和地址总线的加密.

● 软件层面: 利用 ORAM (oblivious RAM) 技术来隐藏 CPU 的实际内存访问模式.

5 研究展望

软件侧信道攻击的出现促使研究人员持续在 TEE 侧信道攻击和防御两方面进行研究. 结合最新的研究趋势, 本节从攻击和防御两方面对未来可能出现的研究点进行探讨, 以供相关研究人员参考.

从攻击方面, 侧信道攻击从最开始需要大量昂贵设备探测物理信号以推断机密信息发展到现在以软件方式捕获微体系结构状态信息来提取机密信息, 这使得越来越多的研究人员开始深入研究计算机微体系结构可能存在的侧信道漏洞. 在 TEE 隔离架构中, Intel SGX 极容易遭受基于微体系结构攻击的威胁, ARM TrustZone 也发现了基于微体系结构的侧信道. 未来针对可信执行环境的侧信道攻击, 可以考虑以下几个研究方向.

(1) 挖掘微体系结构侧信道漏洞: 根据第 2 节对 TEE 侧信道攻击面的描述, 当前针对可信执行环境的 TEE 软件侧信道攻击几乎都利用了处理器微体系结构组件的优化功能. 也就是说, 现代处理器中的微体系结构优化组件在提升计算效率的同时牺牲了计算设备的安全性, 利用这些组件可能会造成严重的安全问题. 因此, 未来需要继续挖掘 TEE 隔离架构中微体系结构优化组件的侧信道漏洞, 通过详细分析 Intel SGX 和 ARM TrustZone 等可信执行环境在硬件层面的处理流程、安全检查机制以及隔离机制, 寻找可能会影响 TEE 隔离机制的侧信道漏洞.

(2) 探索 TLB 在上下文切换期间的处理逻辑: 当前, TEE 隔离架构中没有出现基于 TLB 的侧信道攻击案例, 但是 TLB 理论上存在侧信道漏洞 (参考第 2.3 节). 例如, 上下文切换期间有选择地刷新 TLB 条目, 攻击者通过监视 TLB 即可了解受害者进程的内存访问模式. 因此, 需要继续探索 TEE 隔离架构中的 TLB 漏洞, 并且构建出基于 TLB 的侧信道攻击实例.

从防御方面, 针对 TEE 侧信道攻击的防御措施主要分为软件层面和硬件层面两个方向, 整体呈现“软件为主, 硬件为辅”的趋势. 硬件层面可以通过重新设计硬件结构、更新微码等方法从根本上解决 TEE 软件侧信道攻击问题. 由于硬件层面的对策短期内不太容易部署且成本很高, 故研究人员针对具体攻击提出了相应的软件层面的防御措施. 与硬件层面的对策相比, 软件层面的防御措施更容易实现但无法从根本上防御 TEE 软件侧信道攻击, 有时甚至会引入大量性能开销. 因此, 未来对于 TEE 软件侧信道攻击的防御措施, 可能存在以下几个研究方向:

(1) 探索“硬件为主, 软件为辅”的防御措施: TEE 软件侧信道攻击利用了现代 CPU 内部微体系结构的优化组件来提取受害者的机密信息, 因此基于软件的方法只能起到缓解作用而不能一劳永逸, 彻底解决 TEE 侧信道攻击问题需要朝着“硬件为主, 软件为辅”的方向探索防御措施. 这类防御措施的技术原理如下: 针对当前 TEE 侧信道防御措施中软件方案开销较大以及难以抵御的攻击 (例如, 瞬态执行攻击或组合攻击等), 硬件层面, 硬件开发人员需要尽快重新设计 DRAM 和 CPU 结构, 消除 DRAM 和 CPU 中优化技术带来的侧信道漏洞; 软件层面, 软件开发人员应该继续探索减小性能开销的技术 (例如, 静态分析技术). 针对当前 TEE 侧信道防御措施中软件方案开销较小或易于抵御的攻击 (例如, 分支预测攻击), 软件开发人员可以通过注入噪声等方法进行缓解.

(2) 寻找通用的缓解措施: 目前, 在针对基于 Flush 的 cache 侧信道攻击[104,105]或针对基于 page table 的侧信道攻击的研究中, 研究人员已经找到了抵御这些攻击的通用方法. 除此之外, 大多数防御措施都只针对具体的攻击, 没有尝试将该防御措施扩展到其他类似攻击的防御中, 这不利于软件层面的防御措施的应用与发展. 因此未来针对防御措施的研究需要在原有对策的基础上找到能够抵御同类攻击的通用缓解措施.

6 结束语

本文针对 TEE 隔离架构中的软件侧信道攻击, 详细调研其威胁模型、相关技术和实际的攻击案例, 重点阐述基于处理器指令流水线的 TEE 攻击面分类方法, 并按照上述分类方法对实际的攻击技术进行系统归类, 同时, 从硬件层面和软件层面两个角度分类归纳不同软件侧信道攻击对应的防御技术. 最后, 对可信执行环境软件侧信道攻击技术和其防御措施进行更深层次的探讨, 展望未来的发展趋势并给出一些研究方向, 以期对 TEE 隔离架构的安全研究提供参考.

References:

[1] Kocher P, Jaffe J, Jun B, Rohatgi P. Introduction to differential power analysis. Journal of Cryptographic Engineering, 2011, 1(1): 5–27.

[doi: 10.1007/s13389-011-0006-y]

[2] Mangard S, Oswald E, Popp T. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Berlin: Springer, 2007. 5–6.

[3] Standaert FX. Introduction to side-channel attacks. In: Verbauwhede IMR, ed. Secure Integrated Circuits and Systems. Boston: Springer, 2010. 27–42. [doi: 10.1007/978-0-387-71829-3_2]

[4] Joy PG, Prabhu M, Shanmugalakshmi R. Side channel attack-survey. Int'l Journal of Advanced Scientific Research and Review, 2011, 1(4): 54–57.

[5] Pongaliur K, Abraham Z, Liu AX, Xiao L, Kempel L. Securing sensor nodes against side channel attacks. In: Proc. of the 11th IEEE High Assurance Systems Engineering Symp. Nanjing: IEEE, 2008. 353–361. [doi: 10.1109/HASE.2008.26]

[6] Agrawal D, Archambeault B, Rao JR, Rohatgi P. The EM side-channel(s). In: Proc. of the 4th Int'l Workshop on Cryptographic Hardware and Embedded Systems. California: Springer, 2002. 29–45. [doi: 10.1007/3-540-36400-5_4]

[7] Kocher P, Jaffe J, Jun B. Differential power analysis. In: Proc. of the 19th Annual Int'l Cryptology Conf. California: Springer, 1999. 388–397. [doi: 10.1007/3-540-48405-1_25]

[8] Kocher PC. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Proc. of the 16th Annual Int'l Cryptology Conf. California: Springer, 1996. 104–113. [doi: 10.1007/3-540-68697-5_9]

[9] ARM Limited. ARM security technology—Building a secure system using TrustZone technology. Technical Report, Cambridge: ARM, 2009.

[10] Mukhtar MA, Bhatti MK, Gogniat G. Architectures for security: A comparative analysis of hardware security features in Intel SGX and ARM TrustZone. In: Proc. of the 2nd Int'l Conf. on Communication, Computing and Digital systems (C-CODE). Islamabad: IEEE, 2019. 299–304. [doi: 10.1109/C-CODE.2019.8680982]

[11] Pinto S, Santos N. Demystifying ARM TrustZone: A comprehensive survey. ACM Computing Surveys, 2019, 51(6): 130. [doi: 10.1145/3291047]

[12] Kaplan D, Powell J, Woller T. AMD memory encryption. White Paper, Santa Clara: AMD, 2016.

[13] Ge Q, Yarom Y, Cock D, Heiser G. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. Journal of Cryptographic Engineering, 2018, 8(1): 1–27. [doi: 10.1007/s13389-016-0141-6]

[14] Irazoqui G, Inci MS, Eisenbarth T, Sunar B. Wait a minute! A fast, cross-VM attack on AES. In: Proc. of the 17th Int'l Symp. on Research in Attacks, Intrusions and Defenses. Gothenburg: Springer, 2014. 299–319. [doi: 10.1007/978-3-319-11379-1_15]

[15] Green M, Rodrigues-Lima L, Zankl A, Irazoqui G, Heyszl J, Eisenbarth T. AutoLock: Why cache attacks on ARM are harder than you think. In: Proc. of the 26th USENIX Security Symp. Vancouver: USENIX Association, 2017. 1075–1091.

[16] Cho H, Zhang PH, Kim D, Park J, Lee CH, Zhao ZM, Doupé A, Ahn GJ. Prime+Count: Novel cross-world covert channels on ARM TrustZone. In: Proc. of the 34th Annual Computer Security Applications Conf. San Juan: ACM, 2018. 441–452. [doi: 10.1145/3274694.3274704]

[17] Lyu Y, Mishra P. A survey of side-channel attacks on caches and countermeasures. Journal of Hardware and Systems Security, 2018, 2(1): 33–50. [doi: 10.1007/s41635-017-0025-y]

[18] Lipp M. Cache attacks on ARM [MS. Thesis]. Graz: University of Technology, 2016.

[19] Tromer E, Osvik DA, Shamir A. Efficient cache attacks on AES, and countermeasures. Journal of Cryptology, 2010, 23(1): 37–71. [doi: 10.1007/s00145-009-9049-y]

[20] van Schaik S, Giuffrida C, Bos H, Razavi K. Malicious management unit: Why stopping cache attacks in software is harder than you think. In: Proc. of the 27th USENIX Security Symp. Baltimore: USENIX Association, 2018. 937–954.

[21] Zhang N, Sun K, Shands D, Lou WJ, Hou YT. TruSpy: Cache side-channel information leakage from the secure world on ARM devices. Cryptology ePrint Archive, 2016. 980.

[22] Lipp M, Gruss D, Spreitzer R, Maurice C, Mangard S. ARMageddon: Cache attacks on mobile devices. In: Proc. of the 25th USENIX Security Symp. Austin: USENIX Association, 2016. 549–564.

[23] Yarom Y, Benger N. Recovering OpenSSL ECDSA nonces using the Flush+Reload cache side-channel attack. Cryptology ePrint Archive, 2014. 140.

[24] Ristenpart T, Tromer E, Shacham H, Savage S. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In: Proc. of the 16th ACM Conf. on Computer and Communications Security. Chicago: ACM, 2009. 199–212. [doi: 10.1145/1653662.1653687]

[25] Neve M, Seifert JP. Advances on access-driven cache attacks on AES. In: Proc. of the 13th Int'l Workshop on Selected Areas in Cryptography. Montreal: Springer, 2006. 147–162. [doi: 10.1007/978-3-540-74462-7_11]

[26] Gullasch D, Bangerter E, Krenn S. Cache games—Bringing access-based cache attacks on AES to practice. In: Proc. of the 32nd IEEE

Symp. on Security and Privacy. Oakland: IEEE, 2011. 490–505. [doi: 10.1109/SP.2011.22]

[27] Liu FF, Yarom Y, Ge Q, Heiser G, Lee RB. Last-level cache side-channel attacks are practical. In: Proc. of the 36th IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 605–622. [doi: 10.1109/SP.2015.43]

[28] Yarom Y, Falkner K. Flush+Reload: A high resolution, low noise, L3 cache side-channel attack. In: Proc. of the 23rd USENIX Security Symp. San Diego: USENIX, 2014. 719–732.

[29] Gruss D, Maurice C, Wagner K, Mangard S. Flush+Flush: A fast and stealthy cache attack. In: Proc. of the 13th Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. San Sebastián: Springer, 2016. 279–299. [doi: 10.1007/978-3-319-40667-1_14]

[30] Gülmezoğlu B, Inci MS, Irazoqui G, Eisenbarth T, Sunar B. A faster and more realistic Flush+Reload attack on AES. In: Proc. of the 6th Int'l Workshop on Constructive Side-channel Analysis and Secure Design. Berlin: Springer, 2015. 111–126. [doi: 10.1007/978-3-319-21476-4_8]

[31] Miao XL, Jiang LH, Chang R. Survey of access-driven cache-based side channel attack. Journal of Computer Research and Development, 2020, 57(4): 824–835 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2020.20190581]

[32] Schwarz M, Weiser S, Gruss D, Maurice C, Mangard S. Malware guard extension: Using SGX to conceal cache attacks. In: Proc. of the 14th Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. Bonn: Springer, 2017. 3–24. [doi: 10.1007/978-3-319-60876-1_1]

[33] Bruinderink LG, Hülsing A, Lange T, Yarom Y. Flush, Gauss, and Reload—A cache attack on the bliss lattice-based signature scheme. In: Proc. of the 18th Int'l Conf. on Cryptographic Hardware and Embedded Systems. Santa Barbara: Springer, 2016. 323–345. [doi: 10.1007/978-3-662-53140-2_16]

[34] Disselkoen C, Kohlbrenner D, Porter L, Tullsen D. Prime+Abort: A timer-free high-precision L3 cache attack using Intel TSX. In: Proc. of the 26th USENIX Security Symp. Vancouver: USENIX Association, 2017. 51–67.

[35] Ge JQ, Gao N, Tu CY, Xiang J, Liu ZY. More secure collaborative APIs resistant to Flush+Reload and Flush+Flush attacks on ARMv8-A. In: Proc. of the 26th Asia-Pacific Software Engineering Conf. (APSEC). Putrajaya: IEEE, 2019. 410–417. [doi: 10.1109/APSEC48747.2019.00062]

[36] Briongos S, Malagón P, Moya JM, Eisenbarth T. Reload+Refresh: Abusing cache replacement policies to perform stealthy cache attacks. In: Proc. of the 29th USENIX Security Symp. Berkeley: USENIX Association, 2020. 1967–1984.

[37] Korpershoek JJ. Profiling encryption algorithms using ARM-based cache eviction attacks [MS. Thesis]. Enschede: University of Twente, 2020.

[38] Brumley BB, Hakala RM. Cache-timing template attacks. In: Proc. of the 15th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Tokyo: Springer, 2009. 667–684. [doi: 10.1007/978-3-642-10366-7_39]

[39] Zhang YQ, Juels A, Reiter MK, Ristenpart T. Cross-VM side channels and their use to extract private keys. In: Proc. of the 19th ACM Conf. on Computer and Communications Security. Raleigh: ACM, 2012. 305–316. [doi: 10.1145/2382196.2382230]

[40] Acıiçmez O, Koç ÇK. Trace-driven cache attacks on AES (short paper). In: Proc. of the 8th Int 'l Conf. on Information and Communications Security. Raleigh: Springer, 2006. 112–121. [doi: 10.1007/11935308_9]

[41] Zhang QY, Zhao SJ. Survey of research on protection mechanisms of operating system against board level physical attacks. Ruan Jian Xue Bao/Journal of Software, 2020, 31(10): 3120–3146 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6067.htm [doi: 10.13328/j.cnki.jos.006067]

[42] Schwarz M, Gruss D. How trusted execution environments fuel research on microarchitectural attacks. IEEE Security & Privacy, 2020, 18(5): 18–27. [doi: 10.1109/MSEC.2020.2993896]

[43] Canella C, van Bulck J, Schwarz M, Lipp M, von Berg B, Ortner P, Piessens F, Evtyushkin D, Gruss D. A systematic evaluation of transient execution attacks and defenses. In: Proc. of the 28th USENIX Security Symp. Santa Clara: USENIX Association, 2019. 249–266.

[44] Lipp M, Schwarz M, Gruss D, Prescher T, Haas W, Fogh A, Horn J, Mangard S, Kocher P, Genkin D, Yarom Y, Hamburg M. Meltdown: Reading kernel memory from user space. In: Proc. of the 27th USENIX Security Symp. Baltimore: USENIX Association, 2018. 973–990.

[45] Kocher P, Horn J, Fogh A, Genkin D, Gruss D, Hass W, Hamburg M, Lipp M, Mangard S, Prescher T, Schwarz M, Yarom Y. Spectre attacks: Exploiting speculative execution. In: Proc. of the 40th IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2019. 1–19. [doi: 10.1109/SP.2019.00002]

[46] Wu XH, He YP, Ma HT, Zhou QM, Lin SF. Microarchitectural transient execution attacks and defense methods. Ruan Jian Xue Bao/Journal of Software, 2020, 31(2): 544−563 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5979.htm [doi: 10.

13328/j.cnki.jos.005979]

[47] van Bulck J, Minkin M, Weisse O, Genkin D, Kasikci B, Piessens F, Silberstein M, Wenisch TF, Yarom Y, Strackx R. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In: Proc. of the 27th USENIX Security Symp. Baltimore: USENIX Association, 2018. 991–1008.

[48] Schwarz M, Lipp M, Moghimi D, van Bulck J, Stecklina J, Prescher T, Gruss D. ZombieLoad: Cross-privilege-boundary data sampling. In: Proc. of the 26th ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 753–768. [doi: 10.1145/3319535.3354252]

[49] van Schaik S, Milburn A, Österlund S, Frigo P, Maisuradze G, Razavi K, Bos H, Giuffrida C. RIDL: Rogue in-flight data load. In: Proc. of the 40th IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2019. 88–105. [doi: 10.1109/SP.2019.00087]

[50] van Schaik S, Kwong A, Genkin D, Yarom Y. SGAxe: How SGX fails in practice. 2020. https://sgaxeattack.com/

[51] Koruyeh EM, Khasawneh KN, Song CY, Abu-Ghazaleh N. Spectre returns! Speculation attacks using the return stack buffer. In: Proc. of the 12th USENIX Workshop on Offensive Technologies. Baltimore: USENIX Association, 2018. 1–12.

[52] Evtyushkin D, Riley R, Abu-Ghazaleh N, Ponomarev D. BranchScope: A new side-channel attack on directional branch predictor. In: Proc. of the 23rd Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. Williamsburg: ACM, 2018. 693–707. [doi: 10.1145/3173162.3173204]

[53] Chen GX, Chen SC, Xiao Y, Zhang YQ, Lin ZQ, Lai TH. SgxPectre: Stealing Intel secrets from SGX enclaves via speculative execution. In: Proc. of the 4th IEEE European Symp. on Security and Privacy (EuroS&P). Stockholm: IEEE, 2019. 142–157. [doi: 10.1109/EuroSP.2019.00020]

[54] van Bulck J, Moghimi D, Schwarz M, Lippi M, Minkin M, Genkin D, Yarom Y, Sunar B, Gruss D, Piessens F. LVI: Hijacking transient execution through microarchitectural load value injection. In: Proc. of the 41st IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2020. 54–72. [doi: 10.1109/SP40000.2020.00089]

[55] Ryan K. Hardware-backed heist: Extracting ECDSA keys from Qualcomm's TrustZone. In: Proc. of the 26th ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 181–194. [doi: 10.1145/3319535.3354197]

[56] Lee S, Shih MW, Gera P, Kim T, Kim H, Peinado M. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In: Proc. of the 26th USENIX Security Symp. Vancouver: USENIX Association, 2017. 557–574.

[57] Nilsson A, Bideh PN, Brorsson J. A survey of published attacks on Intel SGX. arXiv:2006.13598, 2020.

[58] Bhattacharya S, Maurice C, Bhasin S, Mukhopadhyay D. Branch prediction attack on blinded scalar multiplication. IEEE Trans. on Computers, 2019, 69(5): 633–648. [doi: 10.1109/TC.2019.2958611]

[59] Tan Y, Wei JZ, Guo W. The micro-architectural support countermeasures against the branch prediction analysis attack. In: Proc. of the 13th IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications. Beijing: IEEE, 2014. 276–283. [doi: 10.1109/TrustCom.2014.38]

[60] Evtyushkin D, Ponomarev D, Abu-Ghazaleh N. Jump over ASLR: Attacking branch predictors to bypass ASLR. In: Proc. of the 49th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). Taipei: IEEE, 2016. 1–13. [doi: 10.1109/MICRO.2016.7783743]

[61] Acıiçmez O, Gueron S, Seifert JP. New branch prediction vulnerabilities in OpenSSL and necessary software countermeasures. In: Proc. of the 11th IMA Int'l Conf. on Cryptography and Coding. Cirencester: Springer, 2007. 185–203. [doi: 10.1007/978-3-540-77272-9_12]

[62] Agosta G, Breveglieri L, Pelosi G, Koren I. Countermeasures against branch target buffer attacks. In: Proc. of the 4th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). Vienna: IEEE, 2007. 75–79. [doi: 10.1109/FDTC.2007.10]

[63] Wang WH, Chen GX, Pan XR, Zhang YQ, Wang XF, Bindschaedler V, Tang HX, Gunter CA. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In: Proc. of the 24th ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 2421–2434. [doi: 10.1145/3133956.3134038]

[64] Gras B, Razavi K, Bos H, Giuffrida C. Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks. In: Proc. of the 27th USENIX Security Symp. Baltimore: USENIX Association, 2018. 955–972.

[65] Wang J, Fan CY, Cheng YQ, Zhao B, Wei T, Yan F, Zhang HG, Ma J. Analysis and research on SGX technology. Ruan Jian Xue Bao/Journal of Software, 2018, 29(9): 2778−2798 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5594.htm [doi: 10.13328/j.cnki.jos.005594]

[66] Götzfried J, Eckert M, Schinzel S, Müller T. Cache attacks on Intel SGX. In: Proc. of the 10th European Workshop on Systems Security. Belgrade: ACM, 2017. 1–6. [doi: 10.1145/3065913.3065915]

[67] Moghimi A, Irazoqui G, Eisenbarth T. Cachezoom: How SGX amplifies the power of cache attacks. In: Proc. of the 19th Int'l Conf. on Cryptographic Hardware and Embedded Systems. Taipei: Springer, 2017. 69–90. [doi: 10.1007/978-3-319-66787-4_4]

[68] Hähnel M, Cui WD, Peinado M. High-resolution side channels for untrusted operating systems. In: Proc. of the 2017 USENIX Annual

Technical Conf. Santa Clara: USENIX Association, 2017. 299–312.

[69] Irazoqui G, Eisenbarth T, Sunar B. S$A: A shared cache attack that works across cores and defies VM sandboxing and its application to AES. In: Proc. of the 36th IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 591–604. [doi: 10.1109/SP.2015.42]

[70] Spreitzer R, Moonsamy V, Korak T, Mangard S. Systematic classification of side-channel attacks: A case study for mobile devices. IEEE Communications Surveys & Tutorials, 2018, 20(1): 465–488. [doi: 10.1109/COMST.2017.2779824]

[71] Crane S, Homescu A, Brunthaler S, Larsen P, Franz M. Thwarting cache side-channel attacks through dynamic software diversity. In: Proc. of the 22nd Annual Network and Distributed System Security Symp. San Diego: NDSS, 2015. 1–14. [doi: 10.14722/ndss.2015.23264]

[72] Szefer J. Survey of microarchitectural side and covert channels, attacks, and defenses. Journal of Hardware and Systems Security, 2019, 3(3): 219–234. [doi: 10.1007/s41635-018-0046-1]

[73] van Schaik S, Minkin M, Kwong A, Genkin D, Yarom Y. CacheOut: Leaking data on Intel CPUs via cache evictions. In: Proc. of the 42nd IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2021. 339–354. [doi: 10.1109/SP40001.2021.00064]

[74] Brasser F, Müller U, Dmitrienko A, Kostiainen K, Capkun S, Sadeghi AR. Software grand exposure: SGX cache attacks are practical. In: Proc. of the 11th USENIX Workshop on Offensive Technologies. Vancouver: USENIX Association, 2017. 1–12.

[75] Xu YZ, Cui WD, Peinado M. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: Proc. of the 36th IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 640–656. [doi: 10.1109/SP.2015.45]

[76] Shinde S, Chua ZL, Narayanan V, Saxena P. Preventing page faults from telling your secrets. In: Proc. of the 11th ACM on Asia Conf. on Computer and Communications Security. Xi'an: ACM, 2016. 317–328. [doi: 10.1145/2897845.2897885]

[77] Pessl P, Gruss D, Maurice C, Schwarz M, Mangard S. DRAMA: Exploiting DRAM addressing for cross-CPU attacks. In: Proc. of the 25th USENIX Security Symp. Washington: USENIX Association, 2016. 565–581.

[78] Wang MH, Zhang Z, Cheng YQ, Nepal S. DRAMDig: A knowledge-assisted tool to uncover DRAM address mapping. In: Proc. of the 57th ACM/IEEE Design Automation Conf. (DAC). San Francisco: IEEE, 2020. 1–6. [doi: 10.1109/DAC18072.2020.9218599]

[79] Xiao Y, Zhang XK, Zhang YQ, Teodorescu R. One bit flips, one cloud flops: Cross-VM row hammer attacks and privilege escalation. In: Proc. of the 25th USENIX Security Symp. Washington: USENIX Association, 2016. 19–35.

[80] Lee D, Jung D, Fang IT, Tsai CC, Popa RA. An off-chip attack on hardware enclaves via the memory bus. In: Proc. of the 29th USENIX Security Symp. Berkeley: USENIX Association, 2020. 487–504.

[81] Koruyeh EM, Shirazi SHA, Khasawneh KN, Song CY, Abu-Ghazaleh N. SpecCFI: Mitigating spectre attacks using CFI informed speculation. In: Proc. of the 41st IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2020. 39–53. [doi: 10.1109/SP40000.2020.00033]

[82] Deng SW, Xiong WJ, Szefer J. Secure TLBs. In: Proc. of the 46th ACM/IEEE Annual Int'l Symp. on Computer Architecture (ISCA). Phoenix: IEEE, 2019. 346–359. [doi: 10.1145/3307650.3322238]

[83] Kiriansky V, Lebedev I, Amarasinghe S, Devadas S, Emer J. DAWG: A defense against cache timing attacks in speculative execution processors. In: Proc. of the 51st Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). Fukuoka: IEEE, 2018. 974–987. [doi: 10.1109/MICRO.2018.00083]

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

[85] Wang ZH, Lee RB. New cache designs for thwarting software cache-based side channel attacks. ACM SIGARCH Computer Architecture News, 2007, 35(2): 494–505. [doi: 10.1145/1273440.1250723]

[86] Liu FF, Ge Q, Yarom Y, Mckeen F, Rozas C, Heiser G, Lee RB. CATalyst: Defeating last-level cache side channel attacks in cloud computing. In: Proc. of the 22nd IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). Barcelona: IEEE, 2016. 406–418. [doi: 10.1109/HPCA.2016.7446082]

[87] Werner M, Unterluggauer T, Giner L, Schwarz M, Gruss D, Mangard S. Scattercache: Thwarting cache attacks via cache set randomization. In: Proc. of the 28th USENIX Security Symp. Santa Clara: USENIX Association, 2019. 675–692.

[88] Coppens B, Verbauwhede I, de Bosschere K, de Sutter B. Practical mitigations for timing-based side-channel attacks on modern x86 processors. In: Proc. of the 30th IEEE Symp. on Security and Privacy. Oakland: IEEE, 2009. 45–60. [doi: 10.1109/SP.2009.19]

[89] Hu WM. Reducing timing channels with fuzzy time. Journal of Computer Security, 1992, 1(3–4): 233–254. [doi: 10.3233/JCS-1992-13-404]

[90] Martin R, Demme J, Sethumadhavan S. Timewarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks. In: Proc. of the 39th Annual Int'l Symp. on Computer Architecture (ISCA). Portland: IEEE, 2012. 118–129. [doi: 10.1109/ISCA.2012.6237011]

[91] Zhou ZQ, Reiter MK, Zhang YQ. A software approach to defeating side channels in last-level caches. In: Proc. of the 23rd ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 871–882. [doi: 10.1145/2976749.2978324]

[92] Chandra S, Karande V, Lin ZQ, Khan L, Kantarcioglu M, Thuraisingham B. Securing data analytics on SGX with randomization. In: Proc. of the 22nd European Symp. on Research in Computer Security. Oslo: Springer, 2017. 352–369. [doi: 10.1007/978-3-319-66402-6_21]

[93] Seo J, Lee B, Kim S, Shih MW, Shin I, Han DS, Kim T. SGX-shield: Enabling address space layout randomization for SGX programs. In: Proc. of the 24th Annual Network and Distributed System Security Symp. San Diego: NDSS. 2017. 1–15.

[94] Gruss D, Lettner J, Schuster F, Ohrimenko O, Haller I, Costa M. Strong and efficient cache side-channel protection using hardware transactional memory. In: Proc. of the 26th USENIX Security Symp. Vancouver: USENIX Association, 2017. 217–233.

[95] Cho H, Park J, Kim D, Zhao ZM, Shoshitaishvili Y, Doupé A, Ahn GJ. SmokeBomb: Effective mitigation against cache side-channel attacks on the ARM architecture. In: Proc. of the 18th Int'l Conf. on Mobile Systems, Applications, and Services. Toronto: ACM, 2020. 107–120. [doi: 10.1145/3386901.3388888]

[96] Liu FF, Lee RB. Random fill cache architecture. In: Proc. of the 47th Annual IEEE/ACM Int'l Symp. on Microarchitecture. Cambridge: IEEE, 2014. 203–215. [doi: 10.1109/MICRO.2014.28]

[97] Wang Y, Ferraiuolo A, Zhang DF, Myers AC, Suh GE. SecDCP: Secure dynamic cache partitioning for efficient timing channel protection. In: Proc. of the 53rd Annual Design Automation Conf. Austin: ACM, 2016. 1–6. [doi: 10.1145/2897937.2898086]

[98] Dessouky G, Frassetto T, Sadeghi AR. HybCache: Hybrid side-channel-resilient caches for trusted execution environments. In: Proc. of the 29th USENIX Security Symp. Berkeley: USENIX Association, 2020. 451–468.

[99] Costan V, Lebedev I, Devadas S. Sanctum: Minimal hardware extensions for strong software isolation. In: Proc. of the 25th USENIX Security Symp. Washington: USENIX Association, 2016. 857–874.

[100] Fu YC, Bauman E, Quinonez R, Lin ZQ. Sgx-Lapd: Thwarting controlled side channel attacks via enclave verifiable page faults. In: Proc. of the 20th Int'l Symp. on Research in Attacks, Intrusions, and Defenses. Atlanta: Springer, 2017. 357–380. [doi: 10.1007/978-3-319-66332-6_16]

[101] Shih MW, Lee S, Kim T, Peinado M. T-SGX: Eradicating controlled-channel attacks against enclave programs. In: Proc. of the 24th Annual Network and Distributed System Security Symp. San Diego: NDSS, 2017. 1–15. [doi: 10.14722/ndss.2017.23193]

[102] Chen SC, Zhang XK, Reiter MK, Zhang YQ. Detecting privileged side-channel attacks in shielded execution with Déjá Vu. In: Proc. of the 12th ACM on Asia Conf. on Computer and Communications Security. Abu Dhabi: ACM, 2017. 7–18. [doi: 10.1145/3052973.3053007]

[103] Dall F, de Micheli G, Eisenbarth T, Genkin D, Heninger N, Moghimi A, Yarom Y. Cachequote: Efficiently recovering long-term secrets of SGX EPID via cache attacks. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2018, (2): 171–191. [doi: 10.13154/tches.v2018.i2.171-191]

[104] Tang CR, Liu ZB, Ma CQ, Ge JQ, Tu CY. SecFlush: A hardware/software collaborative design for real-time detection and defense against Flush-based cache attacks. In: Proc. of the 21st Int'l Conf. on Information and Communications Security. Beijing: Springer, 2019. 251–268. [doi: 10.1007/978-3-030-41579-2_15]

[105] Mukhtar MA, Bhatti MK, Gogniat G. IE-Cache: Counteracting eviction-based cache side-channel attacks through indirect eviction. In: Proc. of the 35th IFIP TC 11 Int'l Conf. on ICT Systems Security and Privacy Protection. Maribor: Springer, 2020. 32–45. [doi: 10.1007/978-3-030-58201-2_3]

附中文参考文献:

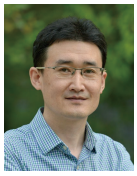
[31] 苗新亮, 蒋烈辉, 常瑞. 访问驱动下的cache侧信道攻击研究综述. 计算机研究与发展, 2020, 57(4): 824–835. [doi: 10.7544/issn1000-1239.2020.20190581]

[41] 张倩颖, 赵世军. 抗电路板级物理攻击的操作系统防御技术研究. 软件学报, 2020, 31(10): 3120–3146. http://www.jos.org.cn/1000-9825/6067.htm [doi: 10.13328/j.cnki.jos.006067]

[46] 吴晓慧, 贺也平, 马恒太, 周启明, 林少锋. 微架构瞬态执行攻击与防御方法. 软件学报, 2020, 31(2): 544–563. http://www.jos.org.cn/1000-9825/5979.htm [doi: 10.13328/j.cnki.jos.005979]

[65] 王鹃, 樊成阳, 程越强, 赵波, 韦韬, 严飞, 张焕国, 马婧. SGX技术的分析和研究. 软件学报, 2018, 29(9): 2778−2798. http://www.jos.org.cn/1000-9825/5594.htm [doi: 10.13328/j.cnki.jos.005594]

杨帆(1998－), 男, 硕士生, 主要研究领域为嵌入式操作系统安全, 侧信道攻击.



施智平(1974－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为形式化验证, 视觉信息处理.

张倩颖(1986－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为操作系统安全, 形式化验证.

关永(1966－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为高可靠嵌入式系统, 形式化验证.