软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(9):2778−2798 [doi: 10.13328/j.cnki.jos.005594]

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

SGX 技术的分析和研究*

王 鹃[1,2], 樊成阳[1,2], 程越强[3], 赵 波[1,2], 韦 韬[3], 严 飞[1,2], 张焕国[1,2], 马 婧[4]

[1](武汉大学 国家网络安全学院,湖北 武汉 430072)

[2](空天信息安全与可信计算教育部重点实验室,湖北 武汉 430072)

[3](Baidu Information Technology Co., Ltd, Bordeaux Dr, Sunnyvale, CA 94089, USA)

[4](信息保障技术重点实验室,北京 100072)

通讯作者: 王鹃, E-mail:jwang@whu.edu.cn

摘 要: 安全性是云计算中一项极为重要的需求,然而如何保护云计算中关键应用程序和数据的安全、防止云平台管理员泄露用户隐私,仍然是目前没有解决的难题.2013 年,Intel 公司提出了新的处理器安全技术 SGX,能够在计算平台上提供一个可信的隔离空间,保障用户关键代码和数据的机密性和完整性.作为系统安全领域的重大研究进展,SGX 对系统安全,尤其是云计算安全保护方面具有非常重要的意义.该文介绍了 SGX 的原理和特性,分析了 SGX 的关键技术以及针对 SGX 的侧信道攻击及防御方法.同时,总结和归纳了该技术的研究成果,分析了 SGX 技术与其他可信计算技术的异同,并指出了 SGX 技术的未来研究挑战和应用需求.

关键词: 云计算;SGX;Enclave;可信计算;侧信道;云安全

中图法分类号: TP311

中文引用格式: 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

英文引用格式: 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). http://www.jos.org.cn/1000-9825/5594.htm

Analysis and Research on SGX Technology

WANG Juan[1,2], FAN Cheng-Yang[1,2], CHENG Yue-Qiang[3], ZHAO Bo[1,2], WEI Tao[3], YAN Fei[1,2], ZHANG Huan-Guo[1,2], MA Jing[4]

[1](School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China)

[2](Key Laboratory of Aerospace Information Security and Trusted Computing Ministry of Education, Wuhan 430072, China)

[3](Baidu Information Technology co., Ltd, Bordeaux Dr, Sunnyvale, CA 94089, USA)

[4](Science and Technology on Information Assurance Laboratory, Beijing 100072, China)

Abstract: Security is an essential requirement for cloud computing. However, how to protect critical applications and data in cloud computing and prevent platform administrators from violating user privacy is still an unsolved problem. In 2013, Intel proposed SGX, a new processor security technology which can provide trust zones on a computing platform to ensure the confidentiality and integrity of key user code and data. As a major research progress in the field of system security, SGX has a very important significance for system security, especially the security protection of cloud computing. In this paper, the mechanisms and properties of SGX are introduced, the

* 基金项目: 国家自然科学基金(61402342, 61173138, 61103628, 61772384); 国家重点基础研究发展计划(973)(2014CB340600); 信息保障技术重点实验室开放基金(KJ-17-103)

Foundation item: National Natural Science Foundation of China (61402342, 61173138, 61103628, 61772384); National Basic Research Program of China (973) (2014CB340600); Foundation of Science and Technology on Information Assurance Laboratory (KJ-17-103)

收稿时间: 2017-10-20; 修改时间: 2018-01-29; 采用时间: 2018-04-21

key principle and technology are analyzed, and the side-channel attack and defense against the SGX technology are presented. Meanwhile, the paper surveys the state of the art of SGX and compares it with other trusted computing technologies. Finally, the research challenges and the future application requirements of SGX are suggested.

Key words: cloud computing; SGX; Enclave; trusted computing; side-channel; cloud security

随着信息技术的迅速发展与广泛应用,人类社会已经进入了一个崭新的互联网时代.一方面,人们享受着互联网科技带来的便利;另一方面,由网络和信息系统构成的网络空间也面临着日益严峻的安全问题.信任是网络空间中安全交互的基础,但随着软件复杂度和攻击水平的提高,移动环境和云平台的安全对硬件和平台安全机制的需要更加迫切,基于硬件的可信执行环境必不可少.以处理器安全为核心的硬件安全技术竞相发展,应用广泛的主流技术包括虚拟化技术如 Intel VT(Intel virtualization technology)与 AMD SVM(AMD secure virtual machine)技术、基于可信平台模块(trusted platform module,简称 TPM)的可信计算技术如 Intel TXT(Intel trusted execution technology)、嵌入式平台 ARM TrustZone 安全扩展等.其中,虚拟化技术基于特权软件 hypervisor 对系统资源进行分配与监控,极大提升了资源利用率,但 hypervisor 潜在的软件漏洞可能威胁到整个系统;基于 TPM 的可信架构在程序加载时进行完整性度量,却难以保障程序运行时的可信执行;TrustZone 为程序提供了两种隔离的执行环境,但需要硬件厂商的签名验证才能运行在安全执行环境,这一特性使得多数软件开发者望而却步.

2013 年,Intel 推出 SGX(software guard extensions)指令集扩展,旨在以硬件安全为强制性保障,不依赖于固件和软件的安全状态,提供用户空间的可信执行环境,通过一组新的指令集扩展与访问控制机制,实现不同程序间的隔离运行,保障用户关键代码和数据的机密性与完整性不受恶意软件的破坏.不同于其他安全技术,SGX 的可信计算基(trusted computing base,简称 TCB)仅包括硬件,避免了基于软件的 TCB 自身存在软件安全漏洞与威胁的缺陷,极大地提升了系统安全保障;此外,SGX 可保障运行时的可信执行环境,恶意代码无法访问与篡改其他程序运行时的保护内容,进一步增强了系统的安全性;基于指令集的扩展与独立的认证方式,使得应用程序可以灵活调用这一安全功能并进行验证.作为系统安全领域的重大研究进展,Intel SGX 是基于 CPU 的新一代硬件安全机制,其健壮、可信、灵活的安全功能与硬件扩展的性能保证,使得这项技术具有广阔的应用空间与发展前景.目前,学术界和工业界已经对 SGX 技术展开了广泛的研究,Intel 也在其最新的第六代 CPU 中加入了对 SGX 的支持.

本文重点分析了 Intel SGX 技术的基本原理及关键技术,并深入讨论了针对 SGX 的侧信道攻击和防御;同时,将该技术与其他可信计算技术进行了对比分析,也着重分析了其优势和不足,并进一步讨论了该技术的相关学术研究工作和商业应用模型;最后,结合当前可信计算领域存在的安全问题和该技术在安全方面的优势,展望了该技术的未来发展方向和应用需求.

1 SGX 架构概述

1.1 SGX技术概述

Intel SGX[1,2]是 Intel 架构新的扩展,在原有架构上增加了一组新的指令集和内存访问机制[3].这些扩展允许应用程序实现一个被称为 enclave 的容器,在应用程序的地址空间中划分出一块被保护的区域,为容器内的代码和数据提供机密性和完整性的保护,免受拥有特殊权限的恶意软件的破坏.SGX 整体架构如图 1 所示.

SGX 的实现需要处理器、内存管理部件、BIOS、驱动程序、运行时环境等软硬件协同完成.除了提供内存隔离与保护安全属性,SGX 架构还支持远程认证和密封的功能,可用于安全软件应用和交互协议的设计.

1.2 SGX的设计目标

(1) 允许应用程序开发者保护敏感数据不被未授权访问或者更高特权级别软件的修改[4−6];

(2) 使得应用程序能够拥有保护敏感代码和数据的机密性与完整性的能力,而不会中断这些资源被合法程序和系统调度、使用和管理的能力;

(3) 使得计算设备的消费者能够控制自己平台,并且具有自由安装和卸载应用与服务的能力;

(4) 使得平台能够度量应用程序的可信代码,生成签名验证,并且度量和认证过程的代码都能够在可信赖的环境中正确的初始化;
(5) 使得可信应用程序的开发过程中能够使用原来的工具和流程;
(6) 允许可信应用程序的性能能够随着处理器的能力增强而得到扩展;
(7) 使得软件代理商能够使用它们选择的分发通道来分发、更新可信应用程序;
(8) 使得应用程序可以定义一个安全代码和数据区域,这一区域可以维护其机密性,即使攻击者能够物理上控制这个平台以及产生对内存的直接攻击,也能够有效加以抵御.

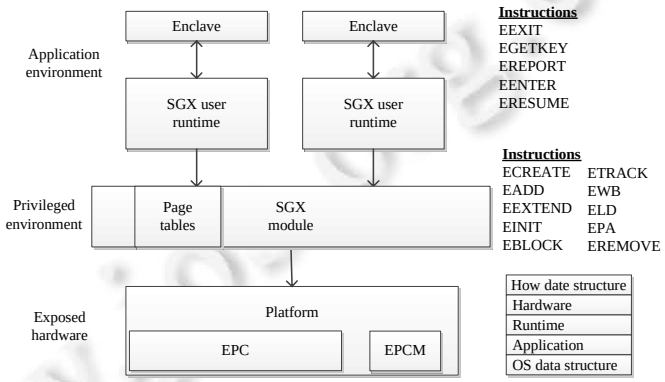


Fig.1 Architecture description of SGX
图 1 SGX 整体架构

2 SGX 关键技术

2.1 Enclave安全容器

Enclave 是一个被保护的内容容器,用于存放应用程序敏感数据和代码[7].SGX 允许应用程序指定需要保护的代码和数据部分,在创建 enclave 之前,不必对这些代码和数据进行检查或分析,但加载到 enclave 中去的代码和数据必须被度量.当应用程序需要保护的部分加载到 enclave 后,SGX 保护它们不被外部软件所访问.Enclave 可以向远程认证者证明自己的身份,并提供必需的功能结构用于安全地提供密钥.用户也可以请求独有的密钥,这个密钥通过结合 enclave 身份和平台的身份做到独一无二,可以用来保护存储在 enclave 之外的密钥或数据.

所有的 enclave 都驻留在 EPC(enclave page cache)中,这是系统内一块被保护的物理内存区域,用来存放 enclave 和 SGX 数据结构[8].EPC 布局由平台具体实现决定,如果 CPU 支持 SGX 架构并在加密保护的 DRAM (dynamic random access memory)中实现 EPC,那么它也支持 BIOS 保留一段叫 PRM(processor reserved memory)的内存范围.BIOS 通过配置一组范围寄存器分配 PRM.具体的 PRM 和 EPC 布局和平台有关,并取决于 BIOS 设置,下图 2 是一个 PRM 和 EPC 布局的例子.

Enclave 具有如下特征.

(1) 具有自己的代码和数据;
(2) 提供机密性保护;
(3) 提供完整性保护;
(4) 具有可控的入口点;
(5) 支持多线程;
(6) 对应用程序内存具有最高访问权限.

Enclave 的结构如图 3 所示,其中,TCS(thread control structure)保存着进入或退出 enclave 时恢复 enclave 线程的特殊信息.每一个 enclave 中的执行线程都和一个 TCS 相关联,它需要 4K 字节对齐,由多个部分组成,例如

保留位(RESERVED)、标志位(FLAGS)、状态保存区偏移量(state save area offset,简称 OSSA)等.

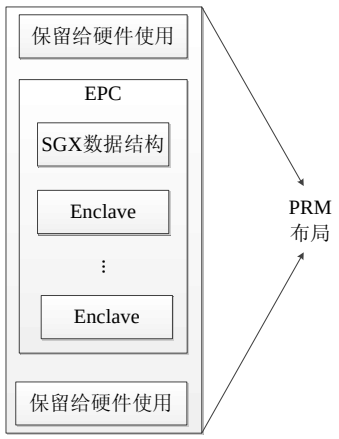


Fig.2 Sample layout of PRM

图 2 PRM 布局示例

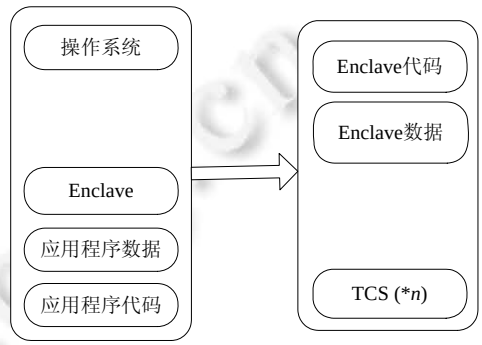


Fig.3 Schematic diagram of enclave

图 3 Enclave 示意图

2.2 Enclave保护机制

针对 enclave 的保护机制主要包括两个部分:一是 enclave 内存访问语义的变化,二是应用程序地址映射关系的保护,这两项功能共同完成对 enclave 的机密性和完整性的保护.

2.2.1 内存访问语义

在系统内分配一块被保护的物理内存区域EPC,用来存放 enclave 和 SGX 数据结构[9].必须保证内存保护机制在物理上锁住 EPC 内存区域,将外部的访问请求视为引用了不存在的内存,使得外部的实体(直接存储器访问、图像引擎等)无法访问.对于使用 MOV 等指令访问 enclave 内部的页面的情况,硬件将执行下列的检查.

(1) 处理器当前运行在 enclave mode 中;

(2) 访问地址在 enclave 地址空间;

(3) 物理地址在 EPC 内存中;

(4) EPCM(enclave page cache map)检查,请求访问的页属于正在运行的 enclave(只有 enclave 内的代码才能访问该 enclave 的内容).

系统在 SGX 调用前,必须处于保护模式,且需要支持分页.SGX 所提供的内存保护机制,在保护模式所提供的段保护、页保护机制基础上进行进一步的内存保护,访问地址由虚拟地址转换为物理地址进行访问.对内存的访问可分为如下 5 种类型,如图 4 所示.

(1) 运行于非 enclave 模式的处理器访问 PRM 之外的内存,按照保护模式下的机制进行访问;

(2) 运行于非 enclave 模式的处理器访问 PRM 内部内存,将被视为引用了不存在的内存;

(3) 处理器运行于 enclave 模式,访问的页面不在 enclave 的虚拟地址空间,但是处于 EPC 的区域范围内,则 CPU 将这次访问视为引用了不存在的内存;

(4) 处理器运行于 enclave 模式,硬件允许 enclave 代码访问处理器保留内存(PRM)外部的地址;

(5) 如果页面在 enclave 的虚拟地址空间外,且指向 PRM 页面,硬件将阻止访问并且发出异常.

简而言之,enclave 外部的应用程序不能访问 enclave 内存;enclave 内部的代码在 EPC 范围内只能访问属于自己的内存区域,不能访问别的 enclave 内存;对于 PRM 以外的内存,则按照系统中其他的保护机制进行访问.这样的内存保护机制,防止了 enclave 内部运行的程序被其他恶意软件盗取隐私信息和篡改.

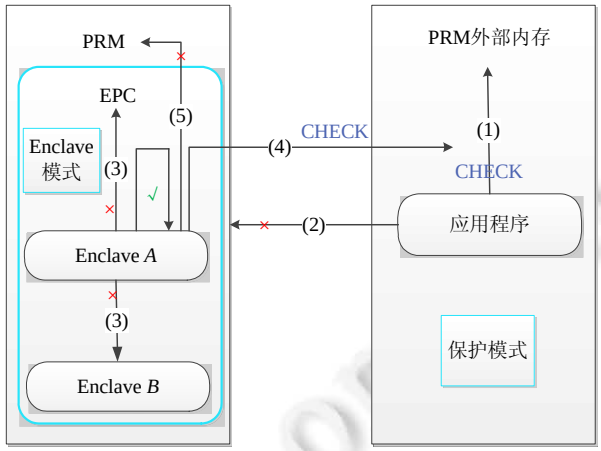


Fig.4 Memory access control of SGX

图 4 SGX 的内存访问控制

2.2.2 地址映射保护

EPC 内存以页为单位进行管理,页的控制信息保存在硬件结构 EPCM 里,一个页面对应一个 EPCM 表项,类似于操作系统内的页表,管理着 EPC 页面的基本信息,包括页面是否已被使用、该页的拥有者、页面类型、地址映射和权限属性等[10].EPCM 结构在 CPU 地址映射过程中用于执行 enclave 页面的访问控制,逻辑上而言,它在保护模式的段保护和页保护机制的基础上增加了一层安全的访问控制.EPCM 结构由 PMH(page miss handler)硬件模块访问,这个模块通过查询页表(系统软件维护的)、范围寄存器、EPCM 来进行内存访问.EPCM 逻辑结构图如图 5 所示.

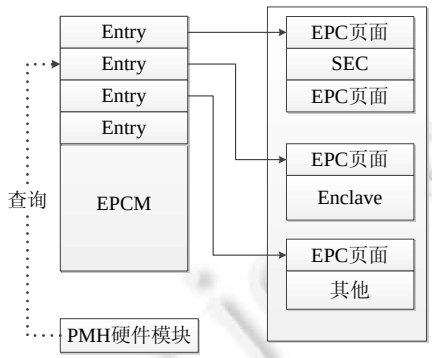


Fig.5 Logical structure of EPCM

图 5 EPCM 逻辑结构

2.2.3 Enclave 机密性和完整性保护

应用程序在申请创建一个 enclave 时,需要进行页面分配、复制程序代码与数据和度量操作,创建过程的最后一步需要对 enclave 的完整性进行验证,判断特权软件在创建过程中是否篡改了程序数据,如分配了多余的页、将恶意代码复制进来,或是篡改了复制的数据等.通过对每个添加的页面内容进行度量,最终得到一个创建序列的度量结果,保存在 enclave 的控制结构中.然后,SGX 通过一条初始化指令将这个结果与 enclave 所有者签名的证书中的完整性值进行比较:如果匹配,则将证书中的所有者公钥进行哈希,作为密封身份保存在 enclave 控制结构中;如果不匹配,则说明创建过程存在问题,指令返回失败结果.

成功进行了初始化指令之后,才能进入 enclave 执行程序,此后 SGX 提供的内存保护和地址映射保护使得

外界无法访问 enclave 内存,从而保证了 enclave 的机密性和完整性,远程的认证者可以通过 enclave 的完整性度量值和其密封身份,确保其正确地创建.图 6 对这个过程进行了描述,其中,较粗箭头表示请求的操作,细箭头表示具体步骤.

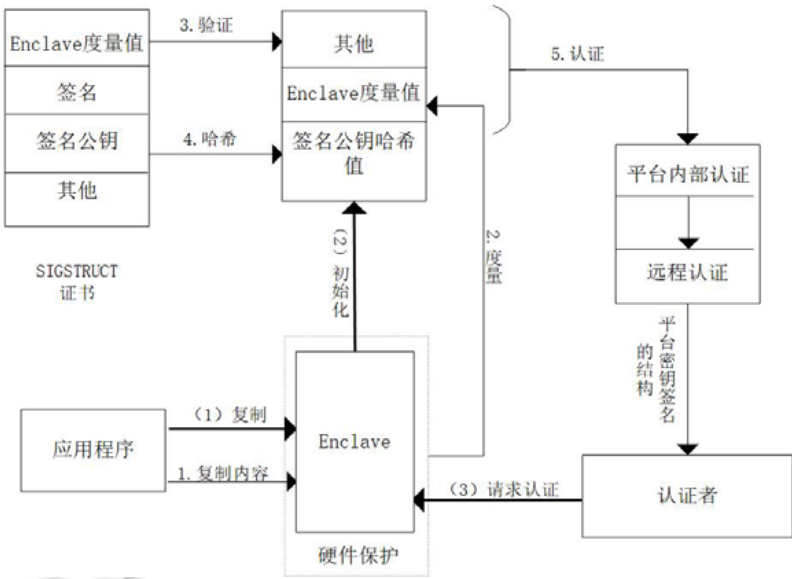


Fig.6 Process of establishing protection

图 6 Enclave 建立保护的过程

2.3 SGX认证

SGX 提出了两种类型的身份认证方式:一种是平台内部 enclave 间的认证,用来认证进行报告的 enclave 和自己是否运行在同一个平台上;另一种是平台间的远程认证,用于远程的认证者认证 enclave 的身份信息.

当 enclave 向平台上其他 enclave 报告身份时,先获取当前的 enclave 的身份信息和属性、平台硬件 TCB 信息,附加上用户希望交互的数据,生成报告结构;然后获取目标 enclave 的报告密钥,对报告结构生成一个 MAC 标签,形成最终的报告结构,传递给目标 enclave,由目标 enclave 验证请求报告身份的 enclave 跟自己是否运行于同一平台.

为了实现远程认证,需要引入一个特殊的引用(quoting)enclave.同一平台 enclave 之间的验证使用的是对称密钥,不适用于远程认证,因此,平台间的认证采用非对称密钥机制.由引用 enclave 创建平台认证的签名密钥 EPID(enhanced privacy identification),这个密钥不仅代表平台,还代表着底层硬件的可信度,并且绑定处理器固件的版本,当 enclave 系统运行时,只有引用 enclave 才能访问到 EPID 密钥.

远程认证的过程中,假设远程认证方 *B* 要认证 enclave *A*,*A* 先执行 EREPORT 指令,将 *A* 的身份和附加信息组合生成 REPORT 结构,利用引用 enclave(称其为 *Q*)的报告密钥生成一个 MAC,连同报告结构一起发给 *Q*,*Q* 通过该结构验证 *A* 是否运行于同一平台,然后将它封装为一个引用结构体 QUOTE,并使用 EPID 进行签名,将 QUOTE 和签名一同发给远程认证者.报告结构还需提供额外的用户数据域,可用来传递用户自定义的信息,以支持更复杂的交互方式.

3 SGX 在云安全中的应用进展

3.1 基于SGX构建云端应用安全隔离执行环境

当前,云计算环境需要可信计算的支持,云环境的构建通常使用传统分层的安全模型来保护特权程序免受

不可信的用户程序的攻击,但是却不能保护用户程序的数据被特权软件所访问和篡改.这导致云环境下的用户只能被动地相信云服务供应商的硬件和软件的可靠性,以及管理人员不会去窃取自己的私密数据.

目前,保护云计算环境安全的方法主要有 3 种.

- 第 1 种是基于特定的硬件保护关键的秘密信息,如密钥的安全.该方法难以保证整个应用程序的安全,且密钥通常会以明文的形式在不可信节点上使用;
- 第 2 种是基于可信的 VMM(virtual machine manager)来保护应用程序,该方法需要整个 VMM 可信,并且无法防止特权用户窃取用户隐私数据;
- 第 3 种方法是基于密文数据的计算,如密文检索,但该方法在性能方面存在局限性.

文献[11]利用 Intel SGX 技术为用户程序提供一个安全的隔离执行环境,从而防止用户数据被特权软件访问,或是受到基于硬件的攻击(如内存扫描).方案基于 Drawbridge[12]沙箱机制,为用户程序的运行提供了一个 Picoprocess 容器[13],从而保证运行在里面的用户程序无法对外界系统造成破坏;再在容器中创建一个 enclave,将用户程序、System Library 和 Shield module 放进 enclave 中,以防止这些数据和代码被外界的特权软件或恶意程序访问和篡改.System Library 通过 Downcalls 和 Upcalls 的方式与 Drawbridge 主机进行交互,用来完成用户程序需要的系统功能,Shield module 模块配合检查函数的参数和返回的结果,进而保护用户程序的安全执行[14,15].

由于操作系统自身可能是不可信的,因此方案中设计了一个 System Library 库,用来将操作系统的系统调用进行封装,并在应用程序运行时将其一起放到 enclave 中供应用程序使用,System Library 自身实现了全部的系统调用.

为了保护用户程序和 System Library 的代码和数据不被 enclave 外的恶意软件攻击,设计了 Shield module,该模块通过仔细地检查参数和函数调用的返回值来进行保护.Shield module 自身包含了一些典型的内核函数:内存管理、进程调度、文件系统操作等.

3.2 基于SGX技术构建安全容器

基于容器的虚拟化现在越来越流行,已经有不少的多租户环境开始使用容器来实现应用程序的安全隔离.Docker[16]或者 Kubernete[17]提供的容器只需要占用较少的资源就能提供快速的启动和比由 hypervisor 监控的虚拟机提供更高的 I/O 性能.但是,现有的容器隔离机制专注于保护其免受不可信容器的访问,然而租户需要保护应用程序数据的机密性和完整性,以防止未经授权的其他容器,或更高级的系统软件(如操作系统内核和管理程序等)访问.这便需要有硬件机制能够保护用户级软件免受特权级系统软件的影响.Intel SGX 的出现恰好满足这一需求.使用 SGX 来构建安全容器面临着两个挑战:(1) 尽量减少 enclave 中可信计算基的大小;(2) 尽量减少性能开销.

因此,文献[18]提出了 SCONE,一种用于 Docker 的安全容器环境,其利用 Inter CPU 提供的 SGX 机制来保护 Docker 容器内进程免受外部攻击.SCONE 的设计主要实现了:

(1) 一个较小的可信计算基;

(2) 更低的性能消耗:SCONE 提供了一个安全 C 语言静态库接口用于透明的加解密 I/O 数据;降低了因线程同步和系统调用导致的高性能消耗;支持用户级线程和异步系统调用.

通过实验评估表明:SCONE 能够通过 SGX 保护未被修改的应用程序,并实现 0.6~1.2 倍的吞吐量.

3.3 基于SGX构建云端大数据安全可信计算环境

MapReduce[19]的出现有力推动了大数据技术和应用的发展,使其成为目前大数据处理最成功的计算技术.由于大数据计算通常会租用公共计算设施,如公有云,因此,如何在 MapReduce 计算过程中确保用户数据不被泄露,解决大数据计算中数据的安全和隐私,是目前亟需要解决的问题[20,21].

目前,对于大数据安全通常采用的是基于密码的保护机制,如全同态加密机制、安全多方计算或零知识证明的机制,然而这些方式目前都因受到性能的制约而没有大规模实用.另外一些方法,如数据库加密机制,如

CtrptDB[22]和 Cipherbase[23],只能对数据库进行保护,却不能保护计算中的代码和数据.基于此,文献[19]提出了基于 SGX 技术构建大数据安全可信计算环境,确保数据在计算和存储中的安全.

该方案主要需解决以下 3 个关键问题.

(1) 利用 SGX 构建最小可信计算基

为增加方案的实用性,本方法需要运行在未修改过的 Hadoop 上,因此系统的可信计算基不包括 Hadoop,OS 和 hypervisor.用户编写 map 和 reduce 代码,并且将它们进行加密,之后上传到云端.在每一个工作节点上,云操作系统将这些代码加载进一个隔离的 enclave 中之后,enclave 内的代码会执行密钥交换协议,解密出 map 和 reduce 函数,从而运行分布式计算处理用户数据.

(2) 保证整个分布式计算的完整性

SGX 只能在本地计算节点上为程序和数据构建安全执行环境,如何在分布式大数据处理过程中确保代码和数据的安全可信是需要解决的关键问题.本方案提出了一个高效的分布式作业执行协议来保证 MapReduce 作业的正确性和机密性.每个计算节点为正在运行的程序产生一个安全的摘要信息,之后再将这些摘要进行收集整合,通过验证最后结果中的最终摘要信息,用户可以检查云服务提供商是否干扰了计算的执行.

(3) 保护用户程序免受非法内存访问攻击

SGX 技术允许用户程序访问系统的全部地址空间,因此,不安全的内存访问可能会泄露数据或者带来其他的安全威胁.如何限制 enclave 内部程序的内存访问,减轻由于应用程序本身的缺陷而导致其遭受非法内存访问攻击,是需要解决的一个问题.该项目基于 GCC 开发了安全增强的编译器[24],在代码编译过程中增加额外参数,将其地址空间限定在有效范围内,从而有效地将需要保证完整性的代码放到一个独立的区域中,并且对该区域中变量的读写访问都将进行检查.只有通过检查,才能真正访问到用户数据.

3.4 基于SGX技术实现NFV的状态保护

网络功能虚拟化(network function virtualization,简称 NFV)通过软硬件解耦及功能抽象,使网络设备功能不再依赖于专用硬件,资源可以充分灵活共享,实现新业务的快速开发和部署,并基于实际业务需求进行自动部署、弹性伸缩、故障隔离和自愈等.NFV 网元是有状态的[25],例如,内容分发网络从远程服务器缓存浏览器内容并且把它们发送至客户端.类似地,入侵检测系统和入侵防御系统都有逐流或者多流(共享)状态来应对入侵.在现今 NFV 的部署方式中,攻击者可以通过访问共享的物理资源来窃取网络应用的状态信息.文献[25]中,作者提出一种保护方案(S-NFV),即是利用 SGX 对 NFV 产生的状态进行安全隔离保护.但该方案较简单,仅在模拟环境 OpenSGX 中验证了保护 Snort[26]应用程序流状态安全的方案.

4 SGX 的漏洞及防御

4.1 SGX侧信道攻击

4.1.1 威胁模型

侧信道攻击主要目标是攻击 enclave 数据的机密性.攻击者来自 non-enclave 部分,包括应用程序和系统软件.系统软件包括 OS,hypervisor,SMM(system management mode),BIOS 等特权级软件.

侧信道攻击一般假设攻击者知道 enclave 初始化时候的代码和数据,并且知道内存布局.内存布局包括虚拟地址、物理地址以及他们之间的映射关系.有些侧信道攻击假设攻击者知道 enclave 的输入数据,并且可以反复触发 enclave,进行多次观察记录.侧信道攻击还假设攻击者知道运行 enclave 平台的硬件配置、特性和性能,比如 CPU、TLB(translation lookaside buffer)、Cache、DRAM、页表、中断以及异常等各种系统底层机制.

4.1.2 侧信道的攻击面

Enclave 和 non-enclave 共享大量的系统资源,这就给侧信道攻击留下了非常大的攻击面.经过对现有资料的总结和系统结构的分析,我们把 SGX 的攻击面总结在图 7 里面.

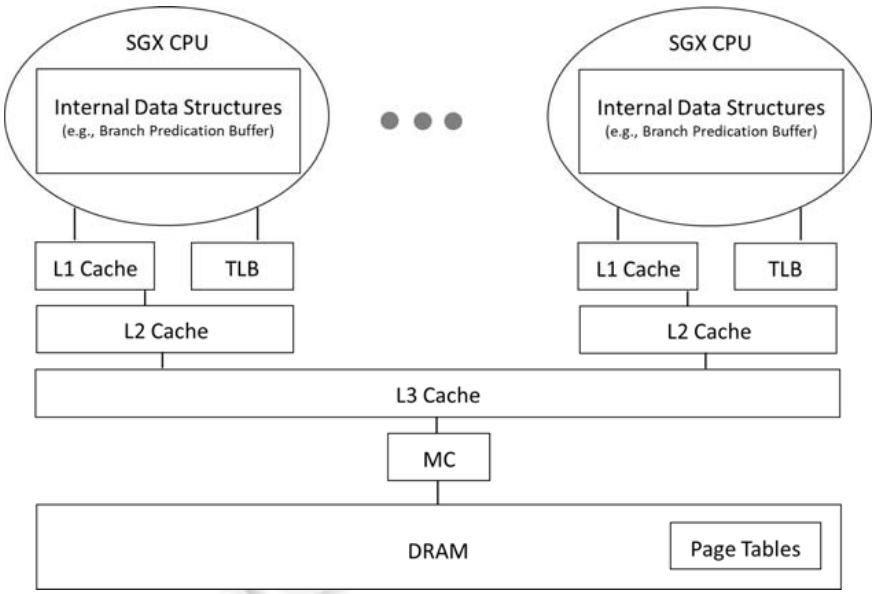


Fig.7 Attack surface of SGX side channel

图 7 Intel SGX 侧信道的攻击面

如图 7 所示,enclave 的运行过程中会用到:

(1) CPU 内部结构.比如 pipeline,BPB(branch prediction buffer)等等.这些结构不能够直接访问,但是如果可以间接利用[27],仍然可能泄露 enclave 的控制流或数据流;

(2) TLB.TLB 包括 iTLB,dTLB 和 L2 TLB.如果 Hyper-Threading 打开,两个逻辑核共享一个物理核,这个时候会大大增加侧信道攻击的可能;

(3) Cache.Cache 包括 L1 instruction Cache,L1 data Cache,L2 Cache 和 L3 Cache(又叫 LLC Cache);

(4) DRAM.DRAM 包括 channels,DIMMs(dual inline memory module),ranks,banks.每个 banks 又包括 rows, columns 和 row buffer;

(5) 页表(page table).页表可以通过权限控制来触发缺页异常,也可以通过页表的状态位来表明 CPU 的某些操作.

对于不同的攻击面,攻击者需要了解具体的细节和工作原理,其中比较重要的参考文档就是 Intel 的手册[28,29].目前,SGX 已经部署在 SkyLake 的机器上面.因此,我们需要对 SkyLake 的一些硬件和性能细节重点掌握.文档[30]对 SkyLakei7-6700 的一些 CPU 细节和性能做了一个比较全面的介绍和测量.

目前,文献[31]发现了 Intel 文档的一处描述和实际实验有冲突,在这里需要重点说明一下.具体来讲,Intel CPU 开发手册指出,SkyLake 机器在打开 Hyper-Threading 之后的 TLB 的配置信息如图 8 所示,然而根据文献[31]的实验,这个 Intel 文档对于 TLB partition 的描述是不准确的.实验结果表明,iTLB 是 fixed partition,其他(dTLB 和 L2 TLB)都是 dynamic partition.

4.1.3 侧信道攻击

侧信道攻击的主要手段是通过攻击面获取数据,推导获得控制流和数据流信息,最终获取 enclave 的代码和数据的信息,比如加密密钥、隐私数据等等.我们这里不一一列举具体的工作,而是试图从攻击面的角度,全面地介绍侧信道攻击.本节下面的内容就从典型的攻击面,包括页表、TLB、Cache、DRAM 以及 CPU 内部结构,描述目前已知的侧信道攻击.

Level	Page Size	Entries	Associativity	Partition
Instruction	4KB	128	8 ways	dynamic
Instruction	2MB/4MB	8 per thread		fixed
First Level Data	4KB	64	4	fixed
First Level Data	2MB/4MB	32	4	fixed
First Level Data	1GB	4	4	fixed
Second Level	Shared by 4KB and 2/4MB pages	1536	12	fixed
Second Level	1GB	16	4	fixed

Fig.8 Configuration of TLB in SkyLake

图 8 SkyLake 的 TLB 配置

4.1.3.1 基于页表的攻击

最早的 SGX 侧信道攻击就是基于页表的攻击[32,33],这类利用页表对 enclave 页面的访问控制权,设置 enclave 页面为不可访问.这个时候任何访问都会触发缺页异常,从而能够区分 enclave 访问了哪些页面.按照时间顺序把这些信息组合,就能够反推出 enclave 的某些状态和保护的数据.该类典型的攻击包括 controlled-channel 攻击[32]和 pigeonhole 攻击[33].这类攻击的缺点就是精度只能达到页粒度,无法区分更细粒度的信息.但是在某些场景下,这类攻击已经能够获得大量有用信息.例如图 9 所示,这类基于页表的侧信道攻击可以获得 libjpeg 处理的图片信息.经过还原,基本上达到人眼识别的程度.pigeonhole 攻击也展示了大量对现有的安全库的攻击,如图 10 所示.

后来,基于页表的攻击有了新的变种.这些侧信道攻击主要利用页表的状态位[31].如图 11 所示,一个页表项有很多位,有些是用来做访问控制,比如 P,RW,US,XD;有些则标识状态,比如 D(dirty bit)和 A(access bit).如果 Abit 被设置,则表明该页表指向的页面已经被访问;如果 D bit 被设置,则表明该页表指向的页面发生了写操作.通过监控观察这些状态位,攻击者就可以获取和 controlled-channel/pigeonhole 攻击类似的信息.

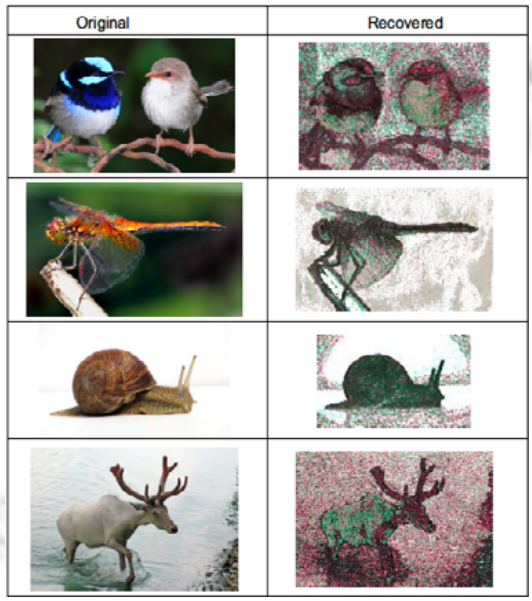


Fig.9 Attack effect of Controlled-channel attack to libjepg

图 9 Controlled-channel 攻击对 libjpeg 的攻击效果

Library	Algo	Secret Entity	Vulnerable Routine	Vulnerable Portion (gcc)	Vulnerable Portion (llvm)	Input Bits	Leakage (gcc)	%	Leakage (llvm)	%
Libgcrypt (v1.6.3)	AES	Symmetric key	Encryption	2 T-Boxes [11:89]	2 T-Boxes [50:50]	128, 192, 256	25	14.01	8	4.51
	CAST5		Key Generation	1 S-Box [38:62]	1 S-Box [48:52]	128	3	2.34	2	1.56
	SEED			1 SS-Box [88:12]	1 SS-Box [27:73]	128	*6	4.69	*4	3.13
	Stribog	Password used in PBKDF2	Key Derivation	4 S-Boxes [51:49]	4 S-Boxes [51:49]	512	32	6.25	32	6.25
	Tiger			2 S-Boxes [53:47]	2 S-Boxes [58:42]	512	4	0.78	4	0.78
	Whrilpool			4 S-Boxes [45:55]	4 S-Boxes [52:48]	512	32	6.25	32	6.25
	EdDSA	Session key (hence Private key)	Signing	ec_mul	ec_mul	512	512	100	512	100
	DSA	Private key	Key generation	powm	powm	256	*160	62.50	*160	62.50
	Elgamal		Modular exponentiation			400	*238	59.50	*238	59.50
	RSA	Private key mod (p-1)				2048	*1245	60.79	*1245	60.79
		Private key mod (q-1)				2048	*1247	60.89	*1247	60.89
OpenSSL (v1.0.2)	CAST5	Symmetric key	Key generation	1 S-Box [55:45]	1 S-Box [84:16]	128	2	1.56	*6	4.69
	SEED			1 SS-Box [47:53]	1 SS-Box [67:33]	128	16	12.50	*6	4.69
							Average	28.02		25.64

Fig.10 Attack effect of Pigeonhole attack to the security library

图 10 Pigeonhole 攻击对安全库的攻击效果

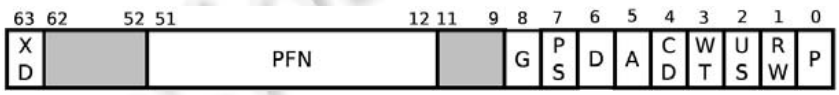


Fig.11 Typical format of PTE

图 11 典型的页表项的格式(x64 系统)

4.1.3.2 基于 TLB 的攻击

目前还没有完全基于 TLB 的攻击,但是已经出现 TLB 作为辅助手段的侧信道攻击,我们会在混合侧信道攻击的章节(第 4.1.3.6 节)里面介绍.关于 TLB 的两点重要信息,我们需要了解,希望对提出新的基于 TLB 的侧信道攻击和防御有所帮助.

(1) TLB 的层次结构.目前,SkyLake 的机器分为 L1 和 L2 两层,不同层次出现 TLB miss 的时间代价不同;

(2) TLB 对代码和数据的区分.L1 区分代码(iTLB)和数据(dTLB),两者直接有 Cache coherence 的保证.L2 不区分代码和数据.

4.1.3.3 基于 Cache 的攻击

传统侧信道有很多基于 Cache 的攻击[34−38].在 SGX 的环境里面,大部分侧信道技术仍然适用,而且可以做得更好.原因在于:在 SGX 环境里面仅依赖 CPU,因此当操作系统,甚至是 BIOS 都是恶意的情况下,攻击者可以控制整个系统的资源.文献[39]证明,SGX 易受 Cache-timing 攻击.他们实现了一种基于 Cache 的 Prime&Probe 算法,能够识别 Cache 行粒度上的 enclave 代码访问的内存位置,并在相同的 Hyper-threading 核心上运行 enclave 和攻击线程,使得攻击线程和 enclave 共享内存.通过这种方法,能够在不到 10 秒的时间获得加密程序的 AES 密钥.但是另一方面,SGX 能很好地防御利用 Flush+Reload 的 Cache 攻击.因为 EPC 页面一次只属于一个 enclave,这就导致攻击者和 enclave 程序不能共享代码,也就使得 Flush+Reload 变得不可能.

在攻击者能控制整个系统资源的情况下,可以有针对地调度资源,减少侧信道的噪音,增加侧信道的成功率.降低噪音的策略大体可以有以下几种[39−42].

(1) 核隔离(core Isolation).这个方法的主要目标就是让 enclave 独自占有一个核(不允许其他程序运行在该核上面);

(2) 缓存隔离(cache Isolation).尽量使用 L1 或者 L2 级别的 Cache 进行侧信道攻击.L3 的 Cache 被所有的核共用,会引入不必要的噪音;

(3) 不间断运行(uninterrupted execution).也就是不触发或尽量少触发 AEX(asynchronous enclave exit),因

为 AEX 和后续的 ISR(interrupt service routines)都会使用 Cache,从而引入不必要噪音.少触发 AEX 就是要使用中断绑定(interrupt affinity)和时钟频率;不触发 AEX 基本上就是让系统软件(比如 OS)屏蔽所有中断.

除了降低噪音,攻击者还可以提高攻击的精度,大体策略有:

(1) 高精度时钟.可以采用 APIC(advanced programmable interrupt controller)提供的高精度时钟和硬件 TSC(time stamp counter);

(2) 放大时间差异.比如,攻击者可以配置侧信道攻击代码所在的 CPU 以最高频率运行,而对 enclave 所在的 CPU 进行降频处理.

基于 Cache 的侧信道攻击可以进行细粒度的监控.最小粒度可以做到一个 Cache line,即 64 个字节.由于粒度更小,基于 Cache 的侧信道可以比基于页表的侧信道以及后面介绍的基于 DRAM 的侧信道获得更多的信息.

4.1.3.4 基于 DRAM 的攻击

在讲解基于 DRAM 的侧信道之前,我们首先了解一些 DRAM 的基本知识.DRAM 一般由 channel,DIMM,rank,bank 等部分构成,如图 12 所示.每个 bank 又由 columns 和 rows 组成,并且还有一个 row buffer 用来缓存最近访问过的一个 row.在访问 DRAM 的时候,如果访问地址已经被缓存在 row buffer 当中(情况 *A*),就直接从 buffer 里面读取;否则,需要把访问地址对应的整个 row 都加载到 row buffer 当中(情况 *B*).当然,如果 row buffer 之前缓存了其他 row 的内容,还需要先换出 row buffer 的内容再加载新的 row(情况 *C*).*A*,*B*,*C* 对应的 3 种情况,访问速度依次递减(情况 *A* 最快,情况 *C* 最慢).这样,通过时间上的差异,攻击者就可以了解当前访问的内存地址是否在 row buffer 里面以及是否有被换出.文献[42]在侧信道攻击过程中用到了基于 DRAM 的侧信道信息.另外,文献[43]介绍了更多基于 DRAM 的攻击细节,不过,该文献不是在 SGX 环境下的攻击.

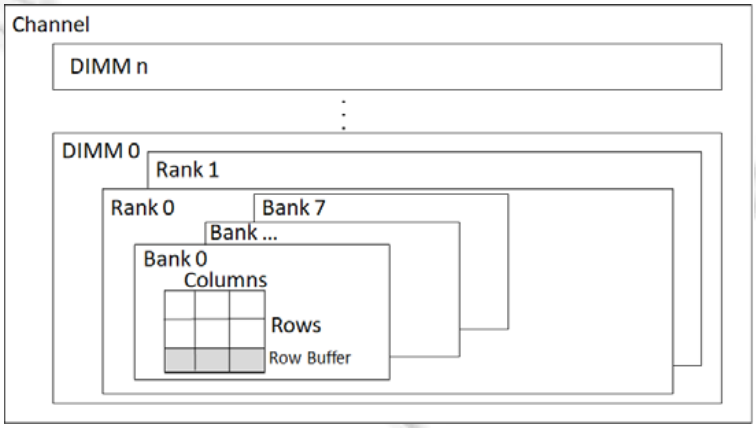


Fig.12 Typical format of DRAM

图 12 典型的 DRAM 的格式

基于 DRAM 的侧信道攻击有一些不足[31].

- enclave 使用的内存通常都在缓存里面,只有少部分需要从 DRAM 里面去取;
- DRAM 的精度不够.例如,一个页面(4KB)通常分布在 4 个 DRAM row 上面,这样,基于 DRAM 的侧信道攻击的精度就是 1KB,仅比基于页表的侧信道攻击好一些,远远不及基于 Cache 的侧信道攻击的精度;
- DRAM 里面存在很难避免的噪音干扰.因为一个 DRAM row 被很多页面使用,同时,同一个 bank 不同 row 的数据读取也会对时间测量造成干扰,使得误报时常发生.

4.1.3.5 基于 CPU 内部结构的攻击

CPU 内部有大量的结构是在 enclave 和 non-enclave 之间共用的.这就给侧信道攻击提供了大量的攻击面素材.比如,文献[27]里面提出使用 BPB 来实现侧信道攻击.具体来讲,在 enclave 和 non-enclave 切换的时候,BPB 里

面存留的跳转预测记录并没有被清除.这样使得 non-enclave 可以构造一个程序,测试这些跳转预测记录.如果预测成功,则执行时间较短;反之,如果预测失败,则执行时间较长.通过时间上的差异,攻击者就可以推测 enclave 之前运行的跳转分支,进而获得 enclave 运行的控制流图.通过控制流图,攻击者又可以进一步推测隐私数据,比如加密密钥等.这个攻击的强大之处在于,它几乎可以还原整个控制流.这样细粒度的信息使得该攻击可以泄露很多信息.该文献也进行了大量实验,充分展示了这个攻击的强大.实验表明,这个攻击可以泄露字符串信息、RSA 私钥以及网络数据等等.

目前,对以 CPU 内部结构为攻击面的工作才刚刚开始,仅仅有一个工作发表.相信通过进一步研究,还会有其他的攻击面被陆续发掘.

从设计上来讲,SGX 可以避免这类侧信道攻击.具体来讲,在 enclave 到 non-enclave 的切换过程中,CPU 清除这些共用的内部结构体.这样,non-enclave 就不会得到任何残留的记录.但在具体实现的时候还要注意一些细节,比如清除的时间也必须是稳定不变的.如果 enclave 运行的差异会导致清除操作的时间差异,攻击者很可能据此推导出 enclave 的某些运行状态.

4.1.3.6 混合侧信道攻击

混合侧信道攻击是同时采集多个侧信道攻击面的信息,或通过多个攻击面共同作用放大差异增加准确度.比较典型的做法包括:

(1) TLB 和页表混合攻击.比如 TLB miss 的时候会加载页表,这个时候 CPU 会设置页表的 Access bit.文献[31]就在 Hyper-Threading 的情况下触发大量的 TLB miss,再通过观察页表的 A bit 进行侧信道攻击;

(2) Cache 和 DRAM 混合攻击.基于 DRAM 的攻击只能精确到 row(一个 row 通常 8KB)的粒度.为了增强这类攻击的效果,文献[31]提出了 Cache-DRAM 攻击来增加空间精度,把精度提高到了一个 Cache line(64B).

除了结合两个攻击面的侧信道攻击,还可以采用多个攻击面相结合的侧信道攻击.这类混合攻击目前在 SGX 的环境下还没有看到相关工作.不过,之前 CCS'16 的文献[44]使用了 3 个攻击面(i.e.,TLB、页表和 Cache)的混合.我们相信,类似的攻击也可以在 SGX 的环境下使用.

4.1.4 未来可能的侧信道攻击

未来,新的侧信道攻击可能来自两个方面.

- 发掘新的混合侧信道攻击.前面列出的经典的混合侧信道攻击,他们往往是用两种攻击面信息.因此,我们可以考虑多个攻击面结合的侧信道攻击.以往的混合侧信道攻击往往专注于内存管理和地址转换等方面,新的侧信道攻击可以结合其他方面的信息,进行一些新的尝试;
- Enclave 所有和 non-enclave 共享的资源都可能成为潜在的侧信道攻击面.因此,发掘新的侧信道攻击的第 2 个途径就是发现新的共享资源,比如未被发掘的 CPU 内部共享结构.这些新的共享资源可能来自一些新的硬件特性,如 Intel PT(processor trace),Intel TSX(transactional synchronization extensions), Intel MPX(memory protection extensions),Intel CAT(cache allocation technology)等等.

4.2 SGX侧信道防御

目前,已经有很多文献给出了防御 SGX 侧信道攻击的方案,有些只是大体的思路,有些则已经有成型的设计和实现.我们在这里主要介绍防御的思路和方法,不涉及设计和实现的细节.首先,我们把这些方法涉及的层次分为源码级别、系统级别和硬件级别.

4.2.1 源码层次的解决方案

这类方法的主要思想就是通过修改源码,编写出能够防御侧信道的代码实现.这里的核心思想就是隐藏控制流和数据流.

这类方法的探索已经在一些密码算法中有所涉及,比如利用 exponent blinding[45]来增强 RSA 算法,利用 bit slicing 增强 DES 和 AES 算法[46,47].在机器学习领域,也有人做了尝试.Oblivious ML[48]修改了机器学习的算法,使用 oblivious assignments and comparisons 来隐藏控制流,使用 oblivious array access(即 *k*-anonymity)来隐藏数

据流.文献 Raccoon[49]也采用了类似的方法,使用 oblivious store 隐藏 if-else 控制流,使用 ORAM(oblivious random access memory)来隐藏数据流,从而抵御侧信道攻击.但是目前,这些技术还很难在一个通用的计算环境下实现,比如 loop tripcount,long jump 以及 break 等问题.

4.2.2 系统层次的解决方案

系统层次的解决方案主要是利用一些系统特性来防御或检测 SGX 侧信道攻击.这里有几个思路可以参考.

(1) 随机化技术(randomization).随机化技术可以应用在控制流和数据流上面,这将大大增加侧信道攻击的代价.防御效果与随机化粒度以及随机化频率有关;

(2) 检测可疑异常和中断.T-SGX[50]利用 TSX 技术来检测中断和缺页异常,从而抵御最原始的 controlled-channel 攻击[32].但是现在已经出现不需要触发 AEX 的侧信道攻击;

(3) 检测时间异常.目前,Déjá Vu 系统[51]也使用了 TSX 技术来保护 enclave 自己的时钟.如果攻击者中断或减缓 enclave 的运行,enclave 就可以通过自己的时钟检测出时间上的异常.目前,绝大多数侧信道攻击都会引起 enclave 的性能显著下降.因此,检测时间异常还是一个比较有效的方案;

(4) Cache 隔离.目前,Intel 推出了 CAT 技术,允许对 Cache 进行粗粒度的隔离.这个技术已经被使用在云计算平台上面[52]防御侧信道攻击,但是还没有看到在 SGX 环境里面的应用.把 CAT 应用到 SGX 的一个很大的障碍是 enclave 在用户空间无法有效地检测或验证 CAT 的配置.

4.2.3 硬件层次的解决方案

硬件层次的解决方案还处于探索阶段.加入侧信道防御,将会显著增加硬件复杂度,影响功耗和性能.这也可能是 Intel 在最初推出 SGX 的时候没有加入侧信道防御的一个原因.硬件解决方案可能有以下两种.

(1) 硬件分割(partition).类似于 ARM 里面的 TrustZone,有自己的 Cache,memory 等一系列硬件资源,物理上与 non-enclave 分离;

(2) 硬件隔离(isolation).类似于 Intel CAT 技术,可以单独为每一个 enclave 提供一个动态隔离出的 Cache.当 enclave 销毁的时候,隔离出的 Cache 可以被收回.这里一个很重要的要求就是,enclave 必须可以验证这个功能的有效性.

Sanctum[53]已经做了一些尝试,但是还不够彻底,还会遭受攻击[41].

4.3 SGX程序机密性问题

在文献[54]中,作者基于自动定理证明和信息流分析,提出了一套 SGX 的使用规范,设计了 Moat 这一检测工具,通过在汇编语言层面对程序进行分析,从而检测应用程序是否存在泄露 SGX 区域中秘密信息的可能.

4.4 SGX多线程同步漏洞

在文献[55]中,展示了在使用 SGX 后以往被视为无害的同步漏洞可能会变为严重的安全漏洞.通过在 enclave 代码中利用 UAF(use-after-free)和 TOCTTOU(time-of-check-to-time-of-use)漏洞,一个攻击者可以劫持它的控制流或者绕过访问控制[56,57].

文献[55]提出 AsyncShock,一个利用运行于 SGX 的多线程代码的同步漏洞的工具.AsyncShock 只能通过操作用于执行 enclave 代码的线程调度来达到这一的目标.它允许一个攻击者通过在 enclave 页强制分割错误来中断线程.

5 SGX 技术的挑战与展望

5.1 SGX与其他相关技术对比分析

5.1.1 TPM/TCM

TPM/TCM 提供度量、签名、加解密、密封等功能,其主要用于系统和应用程序的静态完整性度量,无法确保运行态的安全.SGX 除了提供程序加载时的完整性验证外,还保证程序运行时安全,缓解针对程序运行时的攻击[58−60].

5.1.2 Intel TXT

Intel TXT[61]基于 TPM 动态可信度量根,通过一组安全扩展,为系统创建一个被保护的环境,其在设计时缺乏对 SMM 的考虑导致可被绕过,而 SMM 恶意代码无法绕过 SGX 硬件保护机制去访问受保护的模块.

5.1.3 硬件虚拟化

Intel VT/AMD SVM 硬件虚拟化技术提供操作系统级别的隔离环境,SGX 则针对用户空间提供程序粒度的隔离.虚拟化技术通过特权分级制度保证隔离,其安全性依赖于特权软件的安全性,若 VMM 出现安全漏洞,平台的安全性将难以保证.而 SGX 提供硬件隔离的执行环境,不依赖于固件和特权软件的安全性,特权软件也无法访问普通程序被保护的内存[62,63].

5.1.4 TrustZone 技术

TrustZone 技术是由 ARM 公司提出的嵌入式平台安全技术,在尽量不影响原有处理器设计的情况下,通过物理隔离保护安全内存、加密块、键盘和显示器等外设.其将系统的硬件和软件资源划分为两个执行环境——安全环境(secure world)和普通环境(normal world)[64].每个执行环境都有自己的系统软件和应用软件、内存区及外围设备.通过 TrustZone 的硬件逻辑,建立一个隔离的 TEE 为安全敏感应用提供安全服务,使得安全环境的资源不能被普通环境的组件访问,将其与普通环境隔离开来.TrustZone 技术已经得到广泛的研究与应用:文献[65]介绍了利用 TrustZone 构建移动平台 TEE 的方法;文献[66]利用 TrustZone 技术提供的代码隔离技术可以实现可信白名单,确保目标设备只能执行符合白名单策略的授权代码;文献[67]设计了对移动嵌入式内核完整性保护方案,保障内核的可信性.TrustZone 与可信计算技术相结合方面也已经有不少的研究成果:文献[68,69]基于 TrustZone 中实现了符合 TPM 规范的应用于移动平台的 DAA(direct anonymous attestation)协议,为移动平台的安全应用提供快速有效的匿名身份认证服务;文献[70]提出一套可信操作的 TEEM 方案,利用 TrustZone 技术实现了 TPM1.2 模拟器的安全隔离,保障了可信操作的安全性.在实际应用中,苹果、华为和三星等公司也基于 TrustZone 开发了指纹识别和安全支付等应用.

TrustZone 和 SGX 的不同之处在于:TrustZone 中,通过 CPU 将系统划分为两个隔离环境,两者之间通过 SMC 指令通信,一旦 Secure world 中存在恶意程序,那么将危害整个系统的安全;而 SGX 中,一个 CPU 可以运行多个安全 enclaves,并发执行,即使某个 enclave 中存在恶意程序,它也不能访问和危害其他 enclave 的被保护内容.

5.2 SGX技术的优势与不足

5.2.1 优 势

与现有的几种硬件安全技术相比,Intel SGX 主要有 3 大优势:一是通过内存加密技术保护程序运行态的安全,使得通过内存泄漏攻击获取关键信息的难度增大;二是将系统的可信计算基缩小到 CPU,相比以往将整个操作系统或特权软件(如 hypervisor 等)视为可信计算基,可以避免更多的系统攻击带来的危害;三是支持虚拟化技术、容器技术,可用性更强.

5.2.2 SGX 的不足之处

(1) 由于 enclave 处于用户态,其自身无法执行系统调用,需要与不可信区域进行交互(运行库的支持有限,接口的安全性).在执行系统调用前需要退出 enclave,执行完成后将结果返回到 enclave 中,增大了安全风险和系统开销;

(2) Enclave 中的数据和处理过程,如果依赖于外部数据,则存在一定的安全隐患.例如:通过一些不合法输入,可以发起对可信区的缓冲区溢出攻击,这些攻击可能会改变可信区中程序的执行流程、获取可信区中的敏感信息;

(3) SGX 本身无法抵御侧信道攻击;

(4) SGX 提供的 enclave 可使用内存太小,当程序数量和规模增大时,需要换进换出页面.为了保证安全性,需要对页面进行完整性和机密性保障,导致系统开销大;

(5) 使用 SGX 提供的 enclave 时需要对程序进行改造,当程序规模大时,带来的编程成本高.

5.3 SGX研究和应用需求展望

自 2013 年 Intel 推出 SGX 技术以来,SGX 技术便受到了学术界和工业界的极大关注.目前,除 SGX 自身安全问题研究外,SGX 技术已经被学术界和工业界应用到了很多领域.第 3 节中我们介绍了目前 SGX 在云计算安全相关领域的研究工作,此外,SGX 已经被用于构建可信的身份认证环境[71]、可信的网络通信通道[72]、可信的系统审计[73]、高效安全的密文计算机制[74]、保护 AI 程序和数据安全[75]等方面.基于以上研究进展,我们提出了 SGX 在云计算相关研究领域的进一步应用需求和研究方向.

5.3.1 可信计算与 SGX 技术的结合

目前,可信云主要依赖于在服务器中部署 TPM[76,77],利用它的信任链以及完整性度量技术,保障平台的可信启动、操作系统可信、虚拟机可信等.但这种保护是静态的,TPM 无法动态保护系统内存中的密钥等隐私数据的安全.SGX 技术可以对软件运行时的环境进行硬件隔离,安全系数高,将其应用于可信环境构建中,与可信计算技术[78]相结合,可以保障可信应用运行时的安全性.然而,如何将 TPM 与 SGX 技术进行结合,从而构建更安全的可信计算环境,是目前需要进一步研究的问题.

5.3.2 利用 SGX 技术构建可信云安全环境

随着云计算的兴起与发展,虚拟化技术得到了广泛的研究与应用.虚拟化技术为云平台实现资源抽象、隔离以及资源的按需分配提供技术支撑.在云虚拟化架构中,虚拟机是系统资源虚拟化的直观体现,也与云用户密切相关.

目前,一些研究工作将可信计算和虚拟化技术相结合,提出了虚拟可信平台模块的概念,它通过模拟硬件 TPM 的功能为每个虚拟机分配一个 vTPM 设备[79,80],进而为虚拟机提供可信计算服务[81−83].然而这种安全更多关注的是虚拟机的静态安全,对于虚拟机内应用程序的动态安全却无法通过这种方案得到根本保证;同时,虚拟机动态迁移是构建可靠的云平台的重要需求之一[84−88].通过动态迁移技术,可以将虚拟机从负载过高的主机迁移到负载较小的主机,以实现负载均衡;当某一主机出现问题时,可以通过迁移将该主机的虚拟机迁移至其他主机,保障虚拟化平台的安全可靠.以前对于虚拟机动态迁移的研究大都集中在提升性能方面,主流的虚拟化平台包括 Xen[89],KVM[90]都有适用于自身平台的完善的动态迁移技术,但是随着网络空间安全的日益严峻[91],虚拟机迁移的安全问题也开始受到人们的重视.

因此,如何结合 Intel SGX 技术构建可信云执行环境,实现虚拟机的可信启动、可信执行、虚拟可信根的安全保护以及虚拟机及其虚拟可信根的安全迁移,是未来一个值得研究的方向.

5.3.3 利用 SGX 技术构建虚拟网元可信执行环境

在云虚拟网络中,网络功能主要不再仅以硬件的方式出现,取而代之的是数量众多并且可复用的虚拟网元(virtual network function,简称 VNF).然而,如何保护这些虚拟网元自身的安全,是云虚拟网络安全面临的一个难题.SGX 为这一问题的解决提供了硬件基础.

基于上述 SGX 技术在云环境中应用的构想,如图 13,可以构建基于 SGX 的虚拟网元可信执行环境构建方案.通过 SGX 技术为虚拟网络功能构建安全可信的执行环境.主要的思路是:将虚拟网络功能中涉及安全隐私的主要代码和数据,如密码操作、安全策略等放到 enclave 中执行和加密保护;同时,基于 SGX 对虚拟网元进行身份认证、授权和网络通信监管,构建达到与单独隔离硬件接近的安全执行环境.

5.3.4 利用 SGX 技术构建面向云的可信外包计算

外包计算的安全与隐私,是近几年来的研究热点问题.通过将计算外包到云中,一个计算能力有限的用户可以利用云计算强大的计算能力实现更高效更快捷的计算服务.然而,如何保护外包数据和计算过程的安全与隐私,是目前需要解决的问题.传统的方式是利用密文检索等方法,在保护数据隐私的同时实现安全的计算.然而密文检索目前只能在密文数据上实现简单的计算,此外,性能目前仍然是制约密文检索技术被应用的瓶颈.SGX 为保护外包计算的安全与隐私提供了一条新的途径.目前,该方面的研究工作刚刚起步[75],如何基于 SGX 构建可信的外包计算环境,解决多方授权的外包计算安全,以及针对具体应用构建可实用的安全外包计算服务,实现外包计算中数据安全隐私保护,是目前需要进一步研究的问题.

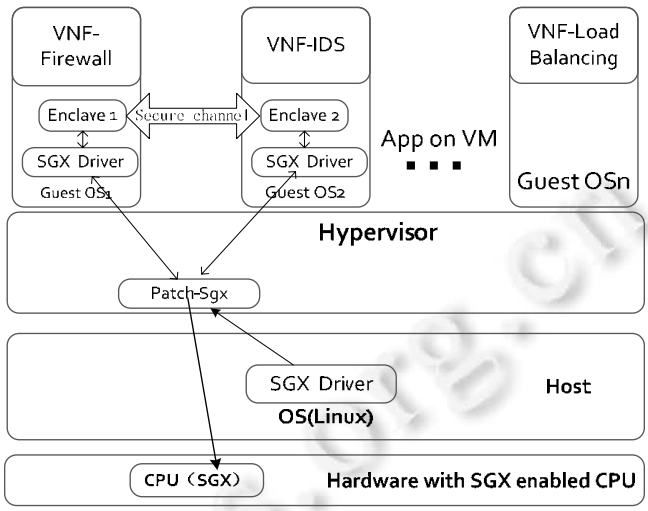


Fig.13 Protected cloud architectures of NFV based on the SGX

图 13 基于 SGX 的 NFV 保护云环境架构

5.3.5 SGX 与 ORAM 的结合

云计算中,用户的数据访问模式可以泄露用户的隐私.为解决这一问题,目前,研究者利用不经意随机存取(ORAM)保护云平台中用户数据访问模式的隐私.然而,目前ORAM方案效率较低,无法实际应用.SGX拥有较小的可信基,同时具有较高的性能和安全性.如何结合 SGX 和各种 ORAM 机制[92],例如 Path ORAM 和 Circuit ORAM,实现更安全的用户数据访问模式隐私保护,是值得关注和研究的问题.

6 结束语

本文对 SGX 的技术原理、侧信道攻击以及在云安全领域的应用和未来研究方向进行了深入分析和总结.首先介绍了该技术的基础架构,并且对其安全框架进行了详细分析;接着,深入剖析了其技术原理,同时,针对 SGX 自身安全,深入讨论了 SGX 侧信道攻击及其防御;此外,文中也将该技术与其他可信计算技术进行了分析对比,在此基础上,指出了 SGX 技术自身的优势和不足,并进一步介绍了 SGX 在云计算安全相关领域的应用;最后,本文提出 SGX 在云安全相关领域的未来应用需求和研究方向.

References:

[1] Wang JW, Jiang Y, Li Q, Yang Y. Survey of research on SGX technology application. Journal of Network New Media, 2017,6(5): 3–9 (in Chinese with English abstract).

[2] Intel Corporation. Intel® software guard extensions programming reference. Rev.2. 2014. Ref. #329298-002.

[3] Shaun D, Richard F. SGX: The good, the bad and the downright ugly. 2014.

[4] Intel Corporation. Intel® SGX for dummies (Intel® SGX design objectives). 2013. https://software.intel.com/en-us/blogs/2013/09/26/protecting-application-secrets-with-intel-sgx

[5] Intel Corporation. Intel® SGX for dummies—Part 2. 2014. https://software.intel.com/en-us/blogs/2014/06/02/intel-sgx-for-dummies-part-2

[6] Intel Corporation. Intel® SGX for dummies—Part 3. 2014. https://software.intel.com/en-us/blogs/2014/09/01/intel-sgx-for-dummies-part-3

[7] Prof. Dr. -Ing. Sadeghi AR. Trusted Execution Environments Intel SGX. Germany: Technische Universität Darmstadt (CASED).

[8] Intel Corporation. Intel® software guard extensions (Intel® SGX). Intel Labs., 2013. https://software.intel.com/sgx

[9] Intel Corporation. Intel® software guard extensions (Intel® SGX) SDK for Linux* OS., 2017.

[10] Rich M. Intel software guard extensions (SGX) is mighty interesting. 2013. https://securosis.com/blog/intel-software-guard-extensions-sgx-is-mighty-interesting

[11] Baumann A, Peinado M, Hunt G. Shielding applications from an untrusted cloud with haven. ACM Trans. on Computer Systems, 2015,33(3):1−26.

[12] Porter DE, Boyd-Wickizer S, Howell J, Olinsky R, Hunt GC. Rethinking the library OS from the top down. ACM Sigplan Notices, 2011,46(3):291−304.

[13] Howell J, Parno B, Douceur JR. How to run POSIX apps in a minimal picoprocess. In: Proc. of the USENIX Conf. on Technical Conf. 2013. 321−332.

[14] Baumann A, Lee D, Fonseca P, Glendenning L, Lorch JR, Bond B, Olinsky R, Hunt GC. Composing OS extensions safely and efficiently with bascule. In: Proc. of the 8th ACM European Conf. on Computer Systems. 2013. 239−252.

[15] Tsai CC, Arora KS, Bandi N, Jain B, Jannen W, John J, Kalodner HA, Kulkarni V, Oliveira D, Porter DE. Cooperation and security isolation of library OSes for multi-process applications. In: Proc. of the 9th European Conf. on Computer Systems. 2014. 1−14.

[16] Merkel D. Docker: Lightweight Linux containers for consistent development and deployment. Linux Journal, 2014(239).

[17] Bernstein D. Containers and cloud: From LXC to docker to kubernetes. IEEE Cloud Computing, 2014,1(3):81−84.

[18] Arnautov S, Trach B, Gregor F, Knauth T, Martin A, Priebe C, Lind J, Muthukumuran D, O'Keeffe D, Stillwell M, Goltzche D, Eyers D, Kapitza R, Pietzuch P, Fetzer C. SCONE: Secure Linux containers with Intel SGX. In: Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI). 2016

[19] Schuster F, Costa M, Fournet C, Gkantsidis C. VC3: Trustworthy data analytics in the cloud using SGX. In: Proc. of the Security and Privacy. IEEE, 2015. 38−54.

[20] Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. Communications of the ACM, 2008,51(1):107−113.

[21] Dill S, Kumar R, Mccurley KS, Rajagopalan S, Sivakumar D, Tomkins A. Self-Similarity in the Web. ACM Trans. on Internet Technology (TOIT), 2002,2(3):205−223.

[22] Popa RA, Redfield CMS, Zeldovich N, Balakrishnan H. CryptDB: Protecting confidentiality with encrypted query processing. In: Proc. of the ACM Symp. on Operating Systems Principles 2011 (SOSP 2011). Cascais, 2011. 85−100.

[23] Arasu A, Blanas S, Eguro K, Kaushik R, Kossmann D, Ramamurthy R, Venkatesan R. Orthogonal security with cipherbase. In: Proc. of the Cidr. Asilomar, 2015.

[24] Nagarakatte S, Zhao JZ, Martin MMK, Zdancewic S. CETS: Compiler-enforced temporal safety for C. ACM SIGPLAN Notices, 2010,45(8):31−40.

[25] Shih MW, Kumar M, Kim T, Gavrilovska A. S-NFV: Securing NFV states by using SGX. In: Proc. of the 2016 ACM Int'l Workshop on Security in Software Defined Networks & Network Function Virtualization. 2016. 45−48.

[26] Roesch M. Snort-lightweight intrusion detection for networks. In: Proc. of the 13th USENIX Conf. on System Administration (LISA'99). 1999. 229−238.

[27] 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. 2016.

[28] Intel Corporation. Intel x64 and ia-32 architectures optimization reference manual. 2016.

[29] Intel Corporation. Intel 64 and IA-32 architectures software developer's manual combined volumes: 1, 2a, 2b, 2c, 2d, 3a, 3b, 3c, 3d and 4. 2017.

[30] 7-Zip LZMA benchmark. Intel skylake benchmark. https://www.7-cpu.com

[31] 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 2017 ACM SIGSAC Conf. on Computer and Communications Security (CCS 2017). 2017. 2421−2434.

[32] Xu YZ, Cui WD, Peinado M. Controlled-Channel attacks: Deterministic side channels for untrusted operating systems. In: Proc. of the 2015 IEEE Symp. on Security and Privacy (SP 2015). 2015. 640−656.

[33] 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 (ASIA CCS 2016). 2016. 317−328.

[34] Liu FF, Yarom Y, Ge Q, Heiser G, Lee RB. Last-Level cache side-channel attacks are practical. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE Computer Society, 2015. 605−622.

[35] Yarom Y, Falkner K. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In: Proc. of the 23rd USENIX Conf. on Security Symp. 2014. 719−732.

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

[37] Gruss D, Maurice C, Wagner K, Mangard S. Flush+Flush: A fast and stealthy cache attack. In: Proc. of the 13th Conf. on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA). 2016.

[38] Gruss D, Spreitzer R, Mangard S. Cache template attacks: Automating attacks on inclusive last-level caches. In: Proc. of the 24th USENIX Conf. on Security Symp. 2015. 897−912.

[39] Götzfried J, Eckert M, Schinzel S, Müller T. Cache attacks on Intel SGX. In: Proc. of the 10th European Workshop on Systems Security (EuroSec 2017). 2017. 1−6.

[40] Moghimi A, Irazoqui G, Eisenbarth T. Cachezoom: How SGX amplifies the power of cache attacks. In: Proc. of the Conf. on Cryptographic Hadware and Embedded Systems (CHES 2017). 2017. 69–90.

[41] Brasser F, Müller U, Dmitrienko A, Kostianinen K, Capkun S, Sadeghi AR. Software grand exposure: SGX cache attacks are practical. In: Proc. of the 11th USENIX Workshop on Offensive Technologies (WOOT 2017). 2017.

[42] Schwarz M, Weiser S, Gruss D, Maurice C, Mangard S. Malware guard extension: Using SGX to conceal cache attacks. In: Proc. of the Detection of Instrusions and Malware, and Vulnerability Assessment. 2017. 3–24.

[43] 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. 2016.

[44] Gruss D, Maurice C, Fogh A, Lipp M, Mangard S. Prefetch side-channel attacks: Bypassing SMAP and kernel ASLR. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. 2016. 368−379.

[45] Kocher PC. Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Proc. of the 16th Annual Int'l Cryptology Conf. on Advances in Cryptology. 1996. 104−113.

[46] Biham E. A fast new DES Implementation in software. In: Proc. of the Int'l Workshop on Fast Software Encryption. 1997. 260−272.

[47] Matsui M. How far can we go on the x64 processors? In: Proc. of the Int'l Workshop on Fast Software Encryption. 2006. 341−358.

[48] Ohrimenko O, Schuster F, Fournet C, Mehta A, Nowozin S, Vaswani K, Costa M. Oblivious multi-party machine learning on trusted processors. In: Proc. of the 25th USENIX Security Symp. (USENIX Security 2016). 2016. 619−636.

[49] Rane A, Lin C, Tiwari M. Raccoon: Closing digital side-channels through obfuscated execution. In: Proc. of the 24th USENIX Security Symp. (USENIX Security 2015). 2015. 431−446.

[50] Shih MW, Lee S, Kim T, Peinado M. T-SGX: Eradicating controlled-channel attacks against enclave programs. In: Proc. of the 2017 Annual Network and Distributed System Security Symp. (NDSS). 2017.

[51] Chen S, Zhang XK, Reiter MK, Zhang YQ. Detecting privileged side-channel attacks in shielded execution with Déjá Vu. In: Proc. of the 2017 ACM on Asia Conf. on Computer and Communications Security. 2017. 7−18.

[52] 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 2016 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). 2016. 406−418.

[53] Costan V, Lebedev I, Devadas S. Sanctum: Minimal hardware extensions for strong software isolation. In: Proc. of the 25th USENIX Security Symp. (USENIX Security 2016). 2016. 857−874.

[54] Sinha R, Rajamani S, Seshia SA, Vaswani K. Moat: Verifying confidentiality of enclave programs. In: Proc. of the ACM SIGSAC Conf. on Computer and Communications Security. 2015. 1169−1184.

[55] Weichbrodt N, Kurmus A, Pietzuch P, Kapitza R. AsyncShock: Exploiting synchronisation bugs in Intel SGX enclaves. In: Proc. of the Computer Security (ESORICS 2016). 2016. 440−457.

[56] Savage S, Burrows M, Nelson G, Sobalvarro P, Anderson T. Eraser: A dynamic data race detector for multi-threaded programs. In: Proc. of the 16th ACM Symp. on Operating Systems Principles. 1997. 27−37.

[57] Checkoway S, Shacham H. Iago attacks: Why the system call API is a bad untrusted RPC interface. In: Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. 2013. 253−264.

[58] GB/T 29827-2013. Information security technology-Trusted computing specification-motherboard function and interface of trusted platform. 2013 (in Chinese with English abstract).

[59] GB/T 29828-2013. Information security technology-Trusted computing specification-trusted connect architecture. 2013 (in Chinese with English abstract).

[60] GB/T 29829-2013. Information security techniques-Functionality and interface specification of cryptographic support platform for trusted computing. 2013 (in Chinese with English abstract).

[61] Intel Corporation. Intel Trusted Execution Technology: Hardware-Based Technology for Enhancing Server Platform Security. White Paper, 2012.

[62] Shi L, Zou DQ, Jin H. Virtualization of Xen. Wuhan: Huazhong University of Science and Technology Press, 2009 (in Chinese).

[63] Sheng QN. Technology status and development trend of virtual trusted platform. Netinfo Security, 2010,(4):34–36 (in Chinese with English abstract).

[64] ARM. ARM security technology: Building a secure system using TrustZone technology. ARM Limited, 2009. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[65] Verma A. Get into the Zone: Building Secure Systems with ARM TrustZone Technology. White Paper, Texas Instruments, 2013.

[66] Zhang YJ, Feng DG, Qin Y, Yang B. A trustzone-based trusted code execution with strong security requirements. Journal of Computer Research & Development, 2015,52(10):2224−2238 (in Chinese with English abstract).

[67] Ge XY, Vijayakumar H, Jaeger T. Sprobes: Enforcing kernel code integrity on the TrustZone architecture. In: Proc. of the Computer Science. 2014. 1793−1795.

[68] Yang B, Feng DG, Qin Y, Zhang QY, Xi L, Zheng CW. Research on direct anonymous attestation scheme based on trusted mobile platform. Journal of Computer Research & Development, 2014,51(7):1436−1445 (in Chinese with English abstract).

[69] Chen LQ, Dietrich K, Löhr H, Sadeghi AR, Wachsmann C, Winter J. Lightweight anonymous authentication with TLS and DAA for embedded mobile devices. In: Proc. of the Int'l Conf. on Information Security. 2011. 84−98.

[70] Feng W, Feng DG, Wei G, Qin Y, Zhang QY, Chang DX. TEEM: A user-oriented trusted mobile device for multi-platform security applications. In: Proc. of the Int'l Conf. on Trust and Trustworthy Computing. 2013. 133−141.

[71] Gu JJ. Intel® Hardware-based Security Technologies Bring Differentiation to Biometrics Recognition Applications Part 1. 2015.

[72] Jain P, Desai S, Kim S, Shih MW, Lee JH, Choi C, Shin Y, Kim T, Kang BB, Han D. OpenSGX: An open platform for SGX research. In: Proc. of the Network and Distributed System Security Symp. 2016.

[73] Karande V, Bauman E, Lin Z, Khan L. SGX-Log: Securing system logs with SGX. In: Proc. of the 2017 ACM on Asia Conf. on Computer and Communications Security. 2017.

[74] Fisch BA, Vinayagamurthy D, Boneh D, Gorbunov S. Iron: Functional encryption using Intel SGX. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. 2017. 765−782.

[75] Ding Y, Duan R, Li L, Cheng YQ, Zhang YL, Chen TH, Wei T, Wang H. POSTER: Rust SGX SDK: Towards memory safety in Intel SGX enclave. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. 2017. 2491−2493.

[76] Trusted Computing Group. Trusted platform module (TPM). 2010. https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

[77] Wang J, Shi Y, Peng GJ, Zhang HG, Zhao B, Yan F, Yu FJ, Zhang LQ. Survey on key technology development and application in trusted computing. China Communications, 2016,13(11):70−90.

[78] Shen CX, Zhang HG, Wang HM, Wang J, Zhao B, Yan F, Yu FJ, Zhang LQ, Xu MD. Research and development of trusted computing. Science China Press: Information Science, 2010,40(2):139−166 (in Chinese with English abstract).

[79] Berger S, Cáceres R, Goldman KA, Perez R, Sailer R, Doom LV. vTPM: Virtualizing the trusted platform module. In: Proc. of the Conf. on USENIX Security Symp. 2006. 305−320.

[80] Shi Y, Zhao B, Zhang HG. A security-improved scheme for virtual TPM based on KVM. Wuhan University Journal of Natural Sciences, 2015,20(6):505−511.

[81] Chen XQ, Gao XP, Wan H, Wang SM, Long X. Application-Transparent live migration for virtual machine on network security enhanced hypervisor. China Communications, 2011,8(3):32−42.

[82] Shetty J, Anala MR, Shobha G. A survey on techniques of secure live migration of virtual machine. Int'l Journal of Computer Applications, 2012,39(12):34−39.

[83] Nagin K, Hadas D, Dubitzky Z, Glikson A, Loy I, Rochwerger B, Schour L. Inter-Cloud mobility of virtual machines. In: Proc. of the Int'l Conf. on Systems and Storage. 2011. 1−12.

[84] Patil VP, Patil GA. Migrating process and virtual machine in the cloud: Load balancing and security perspectives. Int'l Journal of Advanced Computer Science & Information Technology, 2012,1(1).

[85] Bin Sulaiman NA, Masuda H. Evaluation of a secure live migration of virtual machines using ipsec implementation. In: Proc. of the Int'l Conf. on Advanced Applied Informatics. Iiai: IEEE, 2014. 687−693.

[86] Fan W, Kong B, Zhang ZJ, Wang TT, Zhang J, Huang WQ. Security protection model on live migration for KVM virtualization. Ruan Jian Xue Bao/Journal of Software, 2016,27(6):1402−1416 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5009.htm [doi: 10.13328/j.cnki.jos.005009]

[87] Wang W, Zhang Y, Lin B, Wu XX, Miao K. Secured and reliable VM migration in personal cloud. In: Proc. of the Int'l Conf. on Computer Engineering and Technology. 2010. V1-705−V1-709.

[88] Aslam M, Gehrmann C, Bjorkman M. Security and trust preserving VM migrations in public clouds. IEEE, In: Proc. of the Int'l Conf. on Trust, Security and Privacy in Computing and Communications. 2012. 869−876.

[89] Xen Open Source Community. Xen project. http://www.xenproject.org/

[90] KVM (kernel-based virtual machine). https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine

[91] Zhang HG, Han WB, Lai XJ, Lin DD, Ma JF, Li JH. Survey on cyberspace security. Science China Information Sciences, 2015, 58(11):1−43.

[92] Sasy S, Gorbunov S, Fletcher CW. ZeroTrace: Oblivious memory primitives from Intel SGX. In: Proc. of the Symp. on Network and Distributed System Security (NDSS). 2018.

附中文参考文献:

[1] 王进文,江勇,李琦,杨芫.SGX 技术应用研究综述.网络新媒体技术,2017,6(5):3–9.

[58] GB/T 29827-2013.信息安全技术可信计算规范可信平台主板功能接口.2013.

[59] GB/T 29828-2013.信息安全技术可信计算规范可信连接架构.2013.

[60] GB/T 29829-2013.信息安全技术可信计算密码支撑平台功能与接口规范.2013.

[62] 石磊,邹德清,金海.Xen 虚拟化技术.武汉:华中科技大学出版社,2009.

[63] 沈晴霓.虚拟可信平台技术现状与发展趋势.信息网络安全,2010,(4):34–36.

[66] 张英骏,冯登国,秦宇,杨波.基于 Trustzone 的强安全需求环境下可信代码执行方案.计算机研究与发展,2015,52(10):2224–2238.

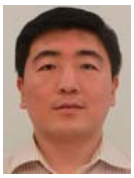
[68] 杨波,冯登国,秦宇,张倩颖,奚璨,郑昌文.基于可信移动平台的直接匿名证明方案研究.计算机研究与发展,2014,51(7):1436–1445.

[78] 沈昌祥,张焕国,王怀民,王戟,赵波,严飞,余发江,张立强,徐明迪.可信计算的研究与发展.中国科学:信息科学,2010,40(2):139–166.

[86] 范伟,孔斌,张珠君,王婷婷,张杰,黄伟庆.KVM 虚拟化动态迁移技术的安全防护模型.软件学报,2016,27(6):1402–1416. http://www.jos.org.cn/1000-9825/5009.htm [doi: 10.13328/j.cnki.jos.005009]

王鹃(1976－),女,博士,副教授, CCF 专业会员,主要研究领域为系统和网络安全,可信计算,云计算,物联网安全.

樊成阳(1994－),男,硕士生,主要研究领域为可信计算,云计算.



程越强(1984－),男,博士,研究员,主要研究领域为系统安全,云安全,可信计算,软件安全.

赵波(1972－),男,博士,教授,博士生导师, CCF 高级会员,主要研究领域为可信计算,系统安全.

韦韬(1975－),男,博士,主要研究领域为安全架构,编程语言,人工智能.

严飞(1980－),男,博士,副教授,CCF 专业会员,主要研究领域为系统安全.

张焕国(1945－),男,教授,博士生导师,CCF 高级会员,主要研究领域为信息安全,可信计算,密码学.

马婧(1982－),女,工程师,主要研究领域为信息安全.