科学通报
进 展
《中国科学》杂志社
SCIENCE CHINA PRESS

软硬件协同的操作系统安全能力创新与应用

古金宇[1,2], 华志超[1,2], 李明煜[1,2], 陈海波[1,2*]

1. 上海交通大学电子信息与电气工程学院, 上海 200240;
2. 领域操作系统教育部工程研究中心(筹), 上海 200240
* 联系人, E-mail: haibochen@sjtu.edu.cn

2022-05-15 收稿, 2022-06-24 修回, 2022-06-28 接受, 2022-06-30 网络版发表

摘要 无论是在移动平台、云平台, 还是在新兴的人机物融合领域, 系统安全都至关重要. 操作系统是现代计算平台的基础与核心支撑技术, 且其内涵和外延随着应用与硬件的发展而不断扩大, 也是构建系统安全能力的核心所在. 面向多维度安全威胁和漏洞, 增强操作系统安全能力的需求迫切, 这也要求操作系统的设计需要统筹兼顾芯片可信执行环境(trusted execution environment, TEE)安全、虚拟化安全、系统内核安全、应用系统安全等多个层次. 本文介绍陈海波团队采用软件与硬件协同的研究思路, 从上述4个层次出发, 开展的操作系统安全能力创新与应用工作. 同时, 也对上述各层次中具有代表性的学术工作进行综述.

关键词 操作系统安全, 可信执行环境安全, 虚拟化安全, 内核安全, 应用系统安全

在个人电脑领域, 我们熟悉的操作系统有Windows、Linux、macOS等; 在手机平台, Android、iOS也是操作系统. 随着计算平台的变化, 还出现了很多新的操作系统概念, 例如城市操作系统、云操作系统、数据中心操作系统、智能汽车操作系统等. 这些不同的操作系统有哪些共性? 总体而言, 操作系统有两个职责: 对硬件进行管理和抽象, 为应用提供服务并进行管理.

首先, 从硬件的角度来看, 操作系统主要包含两类共性功能: 一是管理硬件, 二是对硬件进行抽象. 操作系统将复杂、具备不同功能的硬件资源纳入统一的管理. 其次, 从应用的角度来看, 操作系统主要包含两类共性功能: 一是服务应用, 二是管理应用. 一方面, 操作系统为应用提供了各种不同层次、不同功能的接口以提供不同类型的服务, 将应用从繁杂的资源管理等系统工作中解放出来. 另一方面, 操作系统也负责对应用的生命周期进行管理, 包括应用的加载、启动、切换、调度、销毁等, 从全局角度进行资源分配, 保证应用间的公平性、性能与安全的隔离性, 防止和限制少数恶意应用对系统整体产生的影响[1].

由于操作系统的以上两个主要职责分别面向硬件和应用, 因此当硬件和应用发生改变时, 操作系统通常也需要随之进行演进——这正是当前的现状. 操作系统是现代计算平台的基础与核心支撑技术, 其内涵和外延随着应用与硬件的发展而不断扩大.

正因为操作系统的重要地位, 操作系统的安全能力对整个系统的安全至关重要. 然而, 操作系统面临着来自多个层面的安全威胁. 硬件层面, 诸如幽灵攻击、熔断攻击、冷启动攻击等硬件攻击手段层出不穷, 操作系统需要在依托硬件功能支撑的同时抵御硬件层面的安全威胁. 操作系统层面, 虚拟机监视器与操作系统等核心系统软件越发庞大, 漏洞数量快速增长, 操作系统需要抵御自身漏洞所引入的安全威胁. 应用层面, 大量新兴应用场景对操作系统提出了更高的安全需求,

引用格式: 古金宇, 华志超, 李明煜, 等. 软硬件协同的操作系统安全能力创新与应用. 科学通报, 2022, 67: 3861–3871
Gu J Y, Hua Z C, Li M Y, et al. Innovations and applications of operating system security with a hardware-software co-design (in Chinese). Chin Sci Bull, 2022, 67: 3861–3871, doi: 10.1360/TB-2022-0557

操作系统需要提供更为灵活、高效、全面的应用安全支撑. 面对来自上述3个层面的安全威胁, 操作系统的安全能力构建需要综合考虑硬件体系结构层与用户应用层的安全特性和架构设计. 同时, 操作系统的安全能力构建也离不开硬件体系结构层的安全特性支撑, 软硬协同的方法能够更好地综合利用体系结构与操作系统的各自优势, 构建更为高效的操作系统安全能力. 如图1所示, 本文将操作系统的安全能力构架划分为自底向上的4个层次, 具体包括如下内容.

(1) 芯片可信执行环境(trusted execution environment, TEE)安全层次. 近年来, 芯片层提供的可信执行环境技术支撑了操作系统中高可信隔离域的构建. 除了安全隔离能力本身, 保护粒度是否灵活、安全资源是否充分、隔离效率是否高效等均影响了TEE技术的实际应用. 为此, 需要将芯片级安全特性的设计与操作系统层的安全架构进行有机结合, 从而构建兼具效率与安全性的TEE层安全能力.

(2) 虚拟化安全层次. 系统虚拟化技术是提升资源利用率, 增强运行环境隔离性的重要技术手段. 然而, 虚拟机监视器的庞大代码量为虚拟化层引入了大量安全漏洞, 直接威胁了操作系统与上层应用的安全性. 为此, 需要通过系统软件架构的重设计进行虚拟化层的安全能力构建, 抵御系统软件自身漏洞所引入的安全威胁.

(3) 系统内核安全层次. 操作系统内核是操作系统架构的核心, 一方面需要管理各类软硬件资源、承载数量庞大的功能服务; 另一方面需要为应用提供最为关键的安全支撑. 如何在保证自身功能可扩展的同时, 兼具安全性与可靠性, 是操作系统内核的主要安全挑战. 为此, 需要将安全特性融入操作系统的架构设计, 在内核架构层面实现系统隔离性的提升.

(4) 应用系统安全层次. 随着人工智能、智联网汽车等场景的发展, 应用安全性的要求快速提升, 不仅关系数据与财产安全, 还可能关系生产和生命安全. 另一方面, 大量的系统软件漏洞、复杂的系统与网络架构等均使特权级攻击者成为可能. 应用安全不仅需要考虑应用层安全威胁, 更需要考虑来自特权级软件的恶意行为. 为此, 需要融合硬件层与操作系统层的设计, 使上层应用构件能够抵御特权攻击者的强安全保护能力.

针对上述4个层次, 本文分别介绍本团队在对应层次操作系统安全能力构建方面的代表性创新研究, 并对相关学术研究进行综述.

1 芯片TEE与操作系统安全

可信执行环境(TEE)是由安全硬件与固件构建, 保证运行其中的安全程序(enclave)能够免受恶意操作系统与其他用户程序的攻击, 同时, TEE也提供了远程验证的机制, 保证了运行其中的代码与数据完整性. TEE系统由于高安全性、强隔离性以及可验证性而被广泛地应用在当前主流架构之中. 例如, 移动设备会使用TEE保障用户的支付安全; 云厂商会提供TEE服务来保

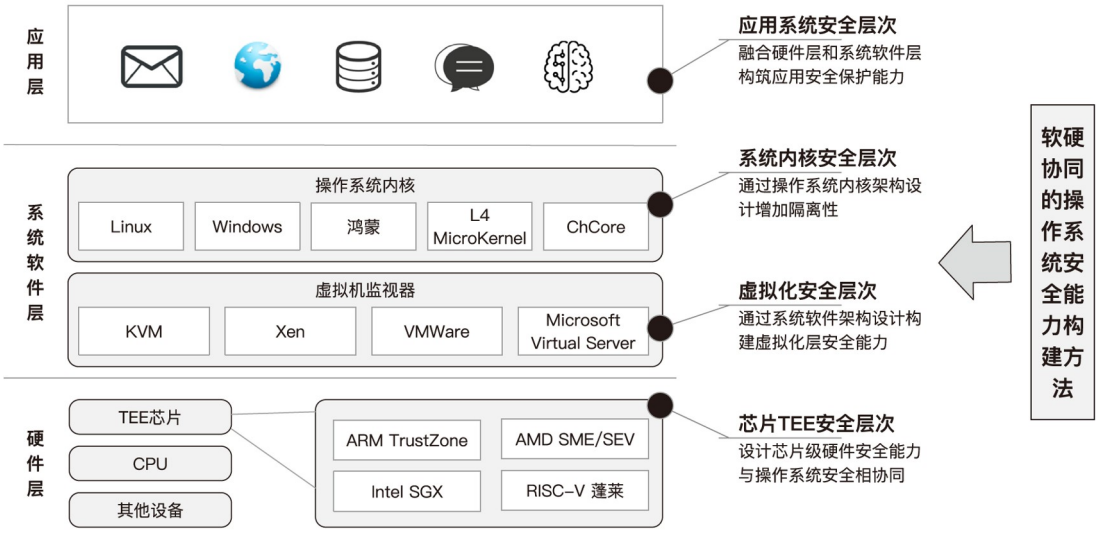

图 1 操作系统安全的4个层次
Figure 1 Four layers of operating system security

护用户的隐私数据以及进行机密计算. 从终端到云端, TEE已经成为当下计算机系统与架构中最为重要的安全基石之一.

本团队提出的蓬莱TEE[2]能够提供一种高效且细粒度的内存保护方案, 并以软硬协同的方式构建可信执行环境, 成为RISC-V官方推荐的三大主流TEE架构之一. 如图2所示, 蓬莱TEE提供了一系列安全硬件扩展以及可信固件设计, 解决了传统TEE架构的主要痛点——安全内存面临的诸多限制: (1) 在安全内存的粒度上, 蓬莱TEE首次实现了细粒度与高效的内存强隔离机制, 保证了在页粒度的内存隔离下, 运行时几乎不产生任何额外开销. (2) 在安全内存的大小上, 蓬莱TEE通过可扩展的内存完整性保护引擎, 将加密与完整性保护的内存大小从256M扩展到了512G, 使得蓬莱TEE能够支持运行大型应用. (3) 在完全内存初始化上, 蓬莱TEE提出了影子enclave, 极大地减少enclave的启动时延, 使TEE能够和新型的云范式(serverless)有机结合. 蓬莱TEE与开源社区有深度合作, 其sPMP硬件扩展已经被RISC-V官方采纳, 增加到下一版本的RISC-V指令集规范中. 同时, 蓬莱TEE与openEuler、openHarmony有深度合作, 使其成为国产操作系统中机密计算的重要基石.

芯片TEE安全层作为现代计算机架构中重要的安全基石, 被广泛地运用在各大主流架构中. Intel利用CPU硬件扩展以及微码的方式, 提出了“软件保护扩展”(software guard extension, SGX)[3]的安全扩展方案. SGX是业界第一个考虑内存隔离、加密与完整性保护, 同时提供远程验证机制的TEE方案. SGX只信任CPU, 其他软硬件包括操作系统以及片外设备均在SGX的可信基之外. 正因为SGX的高安全性, 其在工业界与学术界中被广泛地研究和运用. 然而, SGX也存在诸多缺陷, 例如, 安全内存大小只有256M, 同时需要针对SGX的场景重新开发程序, 极大地增加了SGX应用的开发成本. Intel为了解决这些痛点, 推出了Scalable-SGX以及TDX(https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html). 在牺牲部分安全性的前提下, 扩展了安全内存的大小, 同时为上层应用提供更好的抽象(例如安全虚拟机抽象), 使其能够较好地兼容已有的程序. AMD利用独立的安全芯片, 提出了“安全加密虚拟化”(secure encrypted virtualization, SEV)架构(https://developer.amd.com/sev). 不同于SGX的设计, SEV利用一个独立的安全芯片, 对内存进行加密保护, 同时直接提供安全虚拟机的抽象, 使其获得了更为广泛的应用场景. 然后, SEV的隔离域受限于硬件中密钥的个数, 并且存在诸多其他安全隐患(逆向解密安全内存、物理攻击等). Arm通过软硬协同的方式, 构建移动设备上的可信执行环境: TrustZone[4]. TrustZone将所有的硬件资源分为可信世界和不可信世界, 并且保证了可信世界中的数据只能由可信世界中的模块访问. 同时, TrustZone使用最高特权级管理所有硬件资源. 然而, TrustZone也存在硬件资源静态划分, 难以动态扩展等问题. 为了解决这个问题, Arm在V9中提出了新一代CCA架构(https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture), 能够支持更细粒度的内存保护与动态迁移, 为云场景提供安全基石. Komodo[5]使用形式

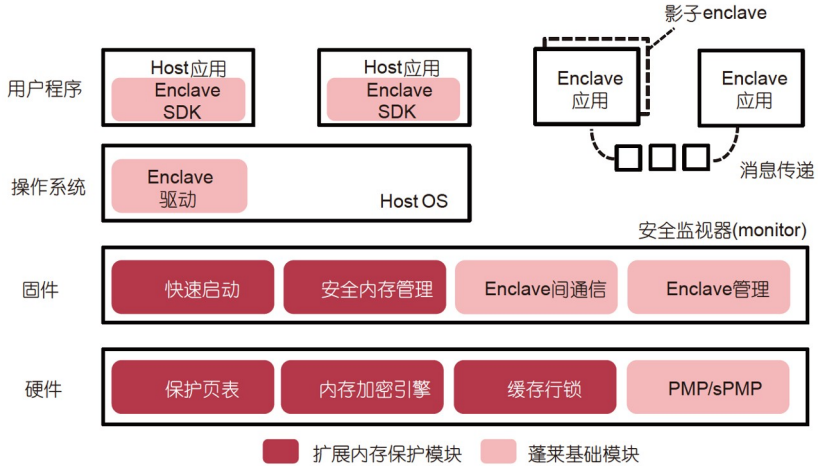

图 2 蓬莱芯片TEE安全设计
Figure 2 Penglai TEE design for the CPU chip

化验证的方式，保证了运行在TrustZone中最高特权级软件不会违反定义的安全规范．形式化验证通过证明的方式，彻底杜绝了软件中存在漏洞与攻击的可能，提供了最强的软件安全保障．因为RISC-V架构开源开放的原因，不同研究团队提出了各自的TEE方案[2,6~8]．除了蓬莱TEE，Keystone是RISC-V上另一个主流的TEE架构．然而，Keystone采用了PMP保护机制，对内存做静态划分，导致同时运行的enclave个数最多为16个．

2 虚拟化与操作系统安全

针对云计算场景中关键的虚拟化安全诉求，本团队提出CloudVisor[9]，研究如何为虚拟机提供安全可用的数据保护方案．由于传统虚拟化方案中VMM软件栈非常庞大复杂，导致威胁数据安全的漏洞层出不穷，容易被潜在攻击者破坏或滥用．如图3所示，位于主机模式中的VMM运行在比虚拟机更高的特权级，能够直接读写虚拟机所使用的内存，一旦VMM被攻破，虚拟机中的敏感数据将暴露给攻击者．面对上述安全威胁，CloudVisor希望即使在VMM被攻破的情况下也能够保护虚拟机的数据隐私和完整性，同时提供一种透明、向后兼容的安全虚拟化方案，以便应用于真实场景．前述相关工作通常使用加固VMM安全性的方法来减少安全漏洞、降低自身被攻破的可能性．然而，这类方案需要对VMM进行大量侵入式修改，难以应用于真实场景．此外，VMM中依旧存在大量代码以支持各类虚拟化功能，导致其安全漏洞难以根除．

CloudVisor创新性地基于嵌套虚拟化概念将资源管理功能与安全保护模块解耦，近乎透明地将VMM放入客户模式中，同时在主机模式中引入一个微小的安全模块为用户虚拟机提供保护．如图3右图所示，位于主机模式中的CloudVisor安全模块保证了VMM与用户虚拟机间的安全隔离，在允许VMM为虚拟机提供资源管理等复杂功能的同时，对二者间的交互进行安全检查，因此即使VMM被攻破，也不会对用户虚拟机中的数据安全造成威胁．测试结果表明，CloudVisor在确保虚拟机数据安全性的同时，仅引入了极小的代码修改以及一定的额外性能开销，能够低成本地应用到现有虚拟化场景中，对后续虚拟化安全工作的研究思路与设计具有指导意义．

后续很多相关工作也延续了CloudVisor思路，将虚拟机与VMM相互隔离以保证数据安全性．TwinVisor[10]为现有云端ARM平台提供了安全虚拟机支持．由于现有ARM平台缺少对安全虚拟机的专用硬件支持，TwinVisor将安全虚拟机放入ARM TrustZone中，与其他外部软件隔离，并复用了传统VMM资源管理功能，同时在VMM与虚拟机之间插入一个轻量级安全模块，以确保虚拟机数据安全性．SSC[11]保护虚拟机免受云服务商攻击，同时为云租户提供了灵活的虚拟机管理能力．SSC将传统VMM划分为相互隔离的系统域与用户域，其中系统域仅允许云服务商进行必要的管理，但无法查看用户数据，而用户域允许租户为虚拟机自定义VMM功能．

如何降低虚拟化安全机制带来的性能开销也是一个重要的研究问题．CloudVisor-D[12]尽可能降低了

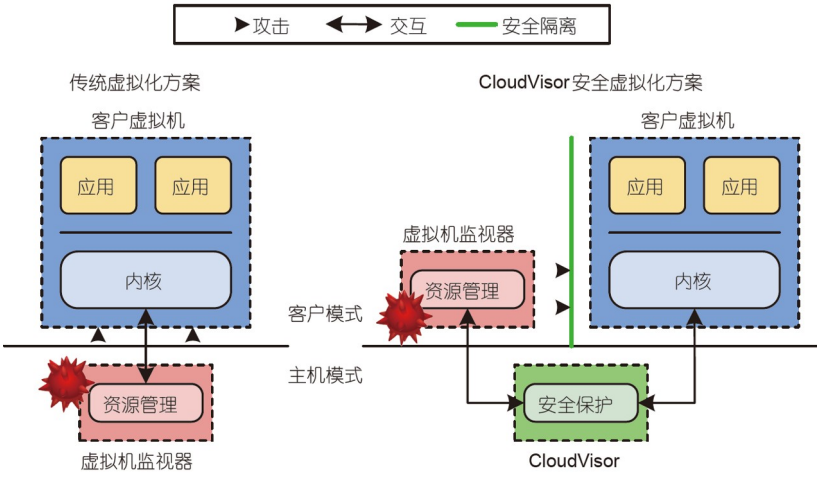

图 3 传统虚拟化方案与CloudVisor安全虚拟化方案对比
Figure 3 Comparison between traditional virtualization systems and the CloudVisor system

CloudVisor架构带来的额外性能开销. VMM与虚拟机间交互必须经过CloudVisor安全模块进行检查, 导致过多的客户/主机模式间切换. 因此, CloudVisor-D利用VMFUNC硬件配合客户模式中的安全模块, 允许VMM与虚拟机安全高效地直接交互, 大幅优化了系统性能. CrossOver[13]则向虚拟化系统提供了安全通用的跨世界(即跨特权级或模式)调用. CrossOver通过扩展VMFUNC硬件, 将跨世界调用的安全检查交给硬件并将调用的授权交给被调用者, 实现了在任意世界间安全高效地切换, 显著提升了系统性能, 也大幅降低了代码复杂度.

除了安全隔离机制以及性能优化方法以外, 为虚拟化系统提供可靠的资源审计也是一大热点, 能够帮助云服务商和用户及时发现安全性与可用性问题. OS-Sommelier[14]提供了精确且高度可靠的虚拟机系统指纹识别(fingerprinting)方法. OS-Sommelier依赖物理内存内容, 针对系统核心代码计算哈希值作为标识符, 消除数据段或设备驱动的噪声, 能够完成精确的指纹识别. HRA[15]则向虚拟机提供安全可靠的资源审计. HRA利用了系统管理模式(system management mode, SMM)的硬件, 运行在系统管理模式中的审计软件拥有比VMM更高的特权级, 因此即使VMM被攻破也不会影响资源审计的可靠性. ATOM[16]为云计算场景提供了高效的资源审计框架来持续自动追踪、协调和监控虚拟化系统中的资源使用情况. ATOM基于主成分分析(principal component analysis, PCA)的方法持续追踪虚拟机资源使用, 不断根据实时监测结果对审计模型进行修正, 以提供更精确的资源使用报告以及安全问题识别.

另有大量工作关注如何提升VMM本身的安全性, 以及如何在发现安全漏洞后及时应对以最小化损失. Min-V[17]禁用了VMM中所有对云上虚拟机不关键的设备模拟, 并对剩余的设备进行了深度裁剪, 移除了不必要的功能, 加固了VMM自身安全性. SCOS[18]对已有安全虚拟化系统采用的3种强制访问控制(mandatory access control, MAC)方法进行了详细分析, 评估了这些方法在云计算场景下的实际安全效果, 针对性地为云场景下的MAC提出了一套额外的安全策略标准以全面控制可能受到损害的云服务威胁. HyperTP[19]基于不同VMM通常不会同时出现相同安全漏洞这一观察, 为用户虚拟机提供了一套透明的跨VMM迁移框架, 允许云服务商在运行中的VMM出现安全漏洞时, 快速将虚拟机迁移到另一种不受影响的VMM, 以减轻该漏洞造成的危害.

3 内核架构与操作系统安全

本团队从安全可靠的角度出发, 选择微内核操作系统架构, 构建了ChCore操作系统[20]. 尽管宏内核操作系统架构的应用范围广泛, 但在安全性方面, 由于所有系统组件都运行在特权级, 具有最高特权, 只要有一个组件被攻击者攻破, 那么攻击者就可能控制整个操作系统. 在可靠性方面, 包括驱动在内的所有操作系统组件都运行在同一个地址空间中, 任意组件的错误都可能导致整个系统的崩溃. 而软件系统很难避免存在缺陷, Herder等人[21]指出, 一般的工业界系统中每千行代码大约会有6~16个缺陷. 以Linux操作系统为例, 其中每年都会被发现约100~200个缺陷和漏洞. 微内核架构将非必须放在内核态的操作系统功能全部以单独系统服务的形式运行在用户态, 从而使得在内核态运行的代码很少, 同时也就意味着潜在的漏洞比宏内核要少很多. 而运行在用户态、互相解耦的微内核操作系统服务具有清晰的隔离边界, 能够极大地限制安全漏洞的影响.

因此, 如图4所示, 本团队自研ChCore操作系统内核采用“微内核+系统服务+系统库”的微内核架构. 完全自研的微内核运行在具有特权的内核态, 它的职责清晰简明, 主要包括3个方面: 资源管理、通信机制、线程调度. 微内核为上层运行的系统服务和系统库提供基础抽象, 但不实现具体的提供给应用程序使用的功能. 诸如读写文件、收发网络包这些功能主要由运行在不具有特权的用户态中系统服务实现. 系统服务层已包括进程管理服务、文件系统服务、网络协议栈

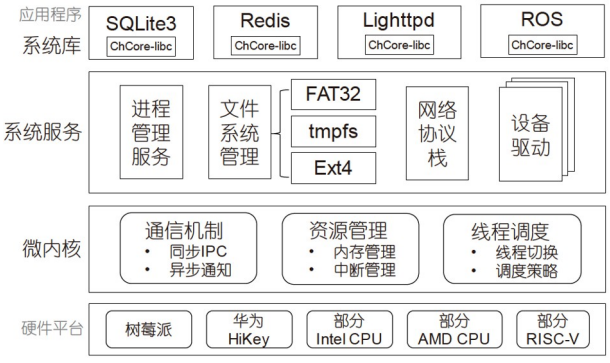

图 4 在系统内核安全层构建ChCore微内核

Figure 4 Building ChCore microkernel as the secure OS kernel

和设备驱动等. 每个系统服务作为一个用户态进程运行在微内核上, 不同系统服务之间互相解耦合, 通过微内核提供的通信机制进行彼此交互以及向应用程序提供服务接口. 系统库直接向应用程序提供编程接口, 既能够透明地兼容现有接口, 比如ChCore-libc支持广泛使用的POSIX接口, 又能提供面向新型硬件或新场景需要的接口, 比如面向enclave的安全编程库. 在微内核架构之上, ChCore操作系统利用软硬件协同的思路进行创新研究, 构建事务性进程间通信抽象, 赋予系统高可靠能力[22], 并结合硬件安全特征, 为应用提供高安全抽象[23].

在内核层安全研究方面, 基于宏内核引入隔离机制也是重要的研究方向. 具体又分为两个子方向: 一是根据内核组件的边界对组件进行隔离, 从而限制其对内核其他部分的访问; 二是使用同级保护的思路, 在内核中隔离出具有某种特权的运行环境负责关键操作. 前者的典型代表是隔离设备驱动的相关工作, 系列工作包括Microdriver[24]、SUD[25]、LXFI[26]、LXD[27]和LVD[28]. 这些工作将不可靠的设备驱动运行在隔离环境中, 避免其中漏洞破坏内核安全. 后者的代表工作是Nested Kernel[29](x86)和TZRKP[30](ARM+TrustZone). 这两个工作从架构上类似于在内核中构建一个更小(可信)的内核, 其中仅包含少量的关键代码, 例如修改页表的代码, 从而保证即便内核被攻破, 受保护的关键代码也不会被劫持, 从而限制攻击能够造成的后果. 内核内部隔离的研究往往会利用到硬件机制, 比如现有的页表、虚拟化等, 或者提出一些新的硬件扩展. 在研究中往往需要利用“hardware for software”的研究思路——如何有效利用或设计硬件机制以提升隔离的安全保证和降低隔离的性能开销. 同级保护机制旨在从内核中隔离出具有某种特权的运行环境负责关键操作, 即使攻击者攻破了内核的剩余部分, 依然无法掌控这些关键能力.

4 应用层与操作系统安全

面向大数据检索、联邦学习等分布式应用场景, 本团队提出PiXiu系统[31], 研究如何为分布式应用提供高可扩展和快速验证的机制. 如图5所示, PiXiu将安全计算的边界拓展到广域网, 实现云边端的数据可信融合. 广域网应用安全层的关键挑战有两点: (1) 节点数量众多, 对可扩展性要求极高; (2) 节点间复杂依赖难以验证. 针对第一项挑战, PiXiu使用“存算分离”思路,

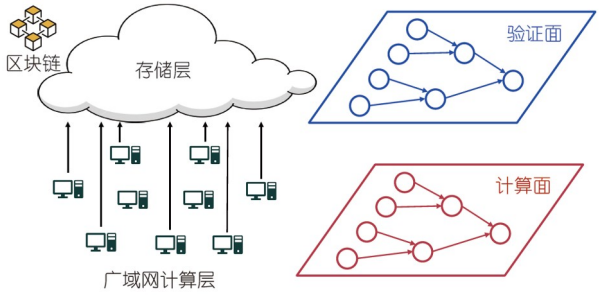

图 5 面向分布式应用层安全的PiXiu系统
Figure 5 PiXiu System: Targeting the security of distributed application scenarios

将状态分离到可用性较强的云存储侧, 而将计算分摊到广域网计算节点侧, 计算节点间组成工作流网络, 定期将计算结果同步到云端. 数据无需在不同计算节点间同步, 达到了较好的可扩展性; 由于状态最终都同步到云端, 数据丢失的概率也大大下降, 达到了较好的容错性. 针对第二项挑战, PiXiu在存储端创新性地将云存储和区块链进行有机结合, 综合了云存储的成本低、写入快以及区块链的只可追加、不可篡改的优势. 计算端则引入基于因果性的数据流追踪机制, 方便任何人查看计算进展同时按需溯源. PiXiu创新性地提出“验算分离”思路, 避免在数据链路上重复验证; 在验证性能上, PiXiu允许不同用户复用已验证的某段过程, 进而提供快速验证能力, 将小时级验证缩短到秒级. PiXiu的应用层安全架构复用了云存储、区块链、可信硬件等现有基础设施, 在可部署性、安全性和性能、成本上取得了较好的权衡.

在应用层大数据方面, 通常使用云计算节省部署成本和管理成本并提供强大算力. 考虑到云上存在隐私泄露的风险, VC3[32]提出基于可信硬件的可验证MapReduce平台, 并提供两方面安全保证: (1) 输入与输出的完整性和隐私性; (2) 分布式计算过程的正确性. 在大数据实际使用过程中, 外部观察者仍可根据访问模式逆向数据特征, Opaque[33]为Spark分析平台设计了针对聚合、过滤、连接等操作提供访存无关的混淆算子.

除了数据中心外, 数据处理应用还可能由多方平台共同协作完成. Ryoan[34]设计了分布式沙盒. 首先, 通过浏览器沙盒保证数据不能在处理过程中泄露; 其次, 通过可信硬件保证了用户可以远程认证沙盒身份; 最后, 通过分布式消息流控制, 使得数据可以在不同沙盒之间流转并始终在用户主导的控制之下. 现代应用通常使用Java、Go等高级语言开发, 安全应用也不例

外. Civet[35]结合静态分析和动态污点技术对JVM进行了划分，将大型代码库中涉及敏感数据流相关的服务划分到硬件隔离的可信边界内，从而抽取出代码量相对较小的JAR包，不仅能有效降低JVM运行过程中所使用的信任组件，也能进一步提高整体系统的可审计性. GOTEE[36]对Go运行时扩展并提供安全协程实现. 基于GOTEE，开发人员给可信应用提供强类型安全保护以及轻量级协程切换的高性能等优势，实现支持TLS的高性能web服务器、保护用户私钥的以太坊钱包.

对于数据库、机器学习等经典场景，同样存在机密性和完整性的双重需求. EnclaveDB[37]将内存数据库保护在可信硬件提供的安全内存里. 为了避免SQL注入等风险，EnclaveDB将受信任的客户端SQL提前编译为存储过程并签名发送给服务器，这样可以避免不授权的用户访问数据内容. 为了在可信硬件上支持深度神经网络的推理，Slalom[38]将网络划分为两部分，非线性层保留在可信边界内完成，而线性层则外包给不可信的GPU进行加速. 线性层在外包前通过可信边界内的随机噪声进行混淆，从而保证外包机器学习的数据安全性.

新型应用场景也存在隐私保护的强烈需求. 针对区块链应用缺乏隐私保护的问题，Ekiden[39]将智能合约从链上转移到链下执行，同时使用可信硬件保护链下智能合约的机密性、完整性和可验证性. 在架构上，Ekiden将计算和共识相分离. 计算节点在链下完成智能合约的执行，共识节点负责区块链账本的维护，对计算结果进行记录：Ekiden智能合约可将计算吞吐提高两个数量级. 针对函数计算应用在可信硬件保护下冷启动性能差的问题，PIE[41]通过分析发现启动瓶颈主要在哈希验证上. PIE将函数计算状态分为机密段和非机密段，非机密段以插件形式对内容和哈希进行跨实例复用，将冷启动时延从十多秒降低到亚秒级别. 在移动端应用方面，恶意手机应用可以伪造用户请求，VButton[41]通过移动端TrustZone引入可信UI组件，允许云服务商提供方对用户行为进行认证，区分用户真实行为和恶意软件模拟的行为.

5 总结与展望

在人-机-物融合的时代，系统安全愈发重要，而构建高安全且高效的系统需要同时考虑构建硬件安全能力和软件安全能力. 因此，软硬件协同的操作系统安全能力是构建系统安全底座的关键支撑，能够在富生态、全场景下提高计算系统的安全性.

参考文献

1 Chen H B, Xia Y B. Modern Operating System Principle and Implementation (in Chinese). Beijing: China Machine Press, 2020 [陈海波, 夏虞斌. 现代操作系统原理与实现. 北京: 机械工业出版社, 2020]

2 Feng E, Lu X, Du D, et al. Scalable memory protection in the Penglai enclave. In: 15th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: The USENIX Association, 2021. 275–294

3 Hoekstra M, Lal R, Pappachan P, et al. Using innovative instructions to create trustworthy software solutions. HASP@ISCA, 2013, 11: 2487726–2488370

4 Pinto S, Santos N. Demystifying arm trustzone. ACM Comput Surv, 2019, 51: 1–36

5 Ferraiuolo A, Baumann A, Hawblitzel C, et al. Komodo: Using verification to disentangle secure-enclave hardware from software. In: Proceedings of the 26th Symposium on Operating Systems Principles. New York: The Association for Computing Machinery, 2017. 287–305

6 Costan V, Lebedev I, Devadas S. Sanctum: Minimal hardware extensions for strong software isolation. In: 25th USENIX Security Symposium (USENIX Security 16). Berkeley: The USENIX Association, 2016. 857–874

7 Lee D, Kohlbrenner D, Shinde S, et al. Keystone: An open framework for architecting trusted execution environments. In: Proceedings of the Fifteenth European Conference on Computer Systems. New York: The Association for Computing Machinery, 2020. 1–16

8 Bahmani R, Brasser F, Dessouky G, et al. CURE: A security architecture with CUstomizable and resilient enclaves. In: 30th USENIX Security Symposium (USENIX Security 21). Berkeley: The USENIX Association, 2021. 1073–1090

9 Zhang F, Chen J, Chen H, et al. Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In: Proceedings of the Twenty-third ACM Symposium on Operating Systems Principles. New York: The Association for Computing Machinery, 2011. 203–216

10 Li D, Mi Z, Xia Y, et al. TwinVisor: Hardware-isolated confidential virtual machines for ARM. In: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. New York: The Association for Computing Machinery, 2021. 638–654

11 Butt S, Lagar-Cavilla H A, Srivastava A, et al. Self-service cloud computing. In: Proceedings of the 2012 ACM Conference on Computer and

Communications Security. New York: The Association for Computing Machinery, 2012. 253–264
12 Mi Z, Li D, Chen H, et al. (Mostly) Exitless VM protection from untrusted hypervisor through disaggregated nested virtualization. In: 29th USENIX Security Symposium (USENIX Security 20). Berkeley: The USENIX Association, 2020. 1695–1712
13 Li W, Xia Y, Chen H, et al. Reducing world switches in virtualized environment with flexible cross-world calls. ACM SIGARCH Comput Archit News, 2015, 43: 375–387
14 Gu Y, Fu Y, Prakash A, et al. Os-sommelier: Memory-only operating system fingerprinting in the cloud. In: Proceedings of the Third ACM Symposium on Cloud Computing. New York: The Association for Computing Machinery, 2012. 1–13
15 Jin S, Seol J, Huh J, et al. Hardware-assisted secure resource accounting under a vulnerable hypervisor. SIGPLAN Not, 2015, 50: 201–213
16 Du M, Li F. ATOM. IEEE Trans Parallel Distrib Syst, 2017, 28: 2172–2189
17 Nguyen A, Raj H, Rayanchu S, et al. Delusional boot: Securing hypervisors without massive re-engineering. In: Proceedings of the 7th ACM European Conference on Computer Systems. New York: The Association for Computing Machinery, 2012. 141–154
18 Sun Y, Petracca G, Jaeger T. Inevitable failure: The flawed trust assumption in the cloud. In: Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security. New York: The Association for Computing Machinery, 2014. 141–150
19 Ngoc T D, Teabe B, Tchana A, et al. Mitigating vulnerability windows with hypervisor transplant. In: Proceedings of the Sixteenth European Conference on Computer Systems. New York: The Association for Computing Machinery, 2021. 162–177
20 Gu J, Wu X, Li W, et al. Harmonizing performance and isolation in microkernels with efficient intra-kernel isolation and communication. In: 2020 USENIX Annual Technical Conference (USENIX ATC 20). Berkeley: The USENIX Association, 2020. 401–417
21 Herder J N, Bos H, Gras B, et al. Construction of a highly dependable operating system. In: 2006 Sixth European Dependable Computing Conference. New York: IEEE, 2006. 3–12
22 Li W, Gu J, Liu N, et al. Efficiently Recovering Stateful System Components of Multi-server Microkernels. In: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS). New York: IEEE, 2021. 494–505
23 Gu J Y, Li H, Xia Y B, et al. Unified enclave abstraction and secure enclave migration on heterogeneous security architectures. J Comput Sci Technol, 2022, 37: 468–486
24 Ganapathy V, Renzelmann M J, Balakrishnan A, et al. The design and implementation of microdrivers. SIGPLAN Not, 2008, 43: 168–178
25 Boyd-Wickizer S, Zeldovich N. Tolerating malicious device drivers in Linux. In: 2010 USENIX Annual Technical Conference (USENIX ATC 10). Berkeley: The USENIX Association, 2010
26 Mao Y, Chen H, Zhou D, et al. Software fault isolation with API integrity and multi-principal modules. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. New York: The Association for Computing Machinery, 2011. 115–128
27 Narayanan V, Balasubramanian A, Jacobsen C, et al. LXDs: Towards isolation of kernel subsystems. In: 2019 USENIX Annual Technical Conference (USENIX ATC 19). Berkeley: The USENIX Association, 2019. 269–284
28 Narayanan V, Huang Y, Tan G, et al. Lightweight kernel isolation with virtualization and VM functions. In: Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. New York: The Association for Computing Machinery, 2020. 157–171
29 Dautenhahn N, Kasampalis T, Dietz W, et al. Nested kernel: An operating system architecture for intra-kernel privilege separation. In: Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems. New York: The Association for Computing Machinery, 2015. 191–206
30 Azab A M, Ning P, Shah J, et al. Hypervision across worlds: Real-time kernel protection from the arm trustzone secure world. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. New York: The Association for Computing Machinery, 2014. 90–102
31 Li M, Zhu J, Zhang T, et al. Bringing Decentralized Search to Decentralized Services. In: 15th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: The USENIX Association, 2021. 331–347
32 Schuster F, Costa M, Fournet C, et al. VC3: Trustworthy data analytics in the cloud using SGX. In: 2015 IEEE Symposium on Security and Privacy. New York: IEEE, 2015. 38–54
33 Zheng W, Dave A, Beekman J G, et al. Opaque: An oblivious and encrypted distributed analytics platform. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17). Berkeley: The USENIX Association, 2017. 283–298
34 Hunt T, Zhu Z, Xu Y, et al. Ryoan. ACM Trans Comput Syst, 2018, 35: 1–32
35 Tsai C C, Son J, Jain B, et al. Civet: An efficient Java partitioning framework for hardware enclaves. In: 29th USENIX Security Symposium (USENIX Security 20). Berkeley: The USENIX Association, 2020. 505–522
36 Ghosn A, Larus J R, Bugnion E. Secured routines: Language-based construction of trusted execution environments. In: 2019 USENIX Annual Technical Conference (USENIX ATC 19). Berkeley: The USENIX Association, 2019. 571–586
37 Priebe C, Vaswani K, Costa M. EnclaveDB: A secure database using SGX. In: 2018 IEEE Symposium on Security and Privacy (SP). New York:

IEEE, 2018. 264–278
38 Tramer F, Boneh D. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. 2018, arXiv: 1806.03287
39 Cheng R, Zhang F, Kos J, et al. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P). New York: IEEE, 2019. 185–200
40 Li M, Xia Y, Chen H. Confidential serverless made efficient with plug-in enclaves. In: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). New York: IEEE, 2021. 306–318
41 Li W, Luo S, Sun Z, et al. VButton: Practical attestation of user-driven operations in mobile apps. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. New York: The Association for Computing Machinery, 2018. 28–40

Summary for “软硬件协同的操作系统安全能力创新与应用”

Innovations and applications of operating system security with a hardware-software co-design

Jinyu Gu[1,2], Zhichao Hua[1,2], Mingyu Li[1,2] & Haibo Chen[1,2*]

[1] *School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China;*
[2] *Engineering Research Center for Domain-specific Operating Systems, Ministry of Education, Shanghai 200240, China*
* Corresponding author, E-mail: haibochen@sjtu.edu.cn

The operating system is the foundation and core support technology of modern computing platforms, responsible for managing hardware resources, controlling the operation of programs, improving the human-machine interface and providing support for application software. Its connotation and extension are constantly expanding with the development of applications and hardware. The scientific aspects of operating systems fall into two categories: The first is the efficient abstraction and management of physical resources; the second is the provision of an efficient operating environment for applications. In the last decade and the period ahead, the specific connotation of the scientific problem is how to provide efficient abstraction and management of physical resources such as heterogeneous cores and data centers, to create efficient operating environments to support application scenarios such as cloud computing, big data and the Internet of Things. Because of the importance of the operating system, the security capability of the operating system is critical to the security of the entire system. The security of the operating system is the security pillar in mobile platforms or cloud platforms. Similarly, in many emerging scenarios, such as the industrial internet, smart networked cars and serverless computing, computer systems' security is related to data and property security, and possibly production and life safety. Therefore, the need to enhance the security capability of the operating system against software and hardware attacks remains urgent in the face of various security threats and multi-dimensional security vulnerabilities. It requires the design of system software to take into account chip TEE security, virtualization security, system kernel security, and application system security. This paper presents our team's work on the innovation and application of operating system security from the above aspects. Specifically, in the aspect of TEE on-chip, we propose Penglai that can offer trusted execution environments for security-sensitive computation. Penglai is built over the emerging RISC-V architecture and its core feature is scalability in both memory capacity and performance. In the aspect of virtualization security, we design CloudVisor which introduces a new system architecture for the virtualization software stack. CloudVisor can offer the abstraction of secure VM against curious or malicious hypervisor and cloud providers. In the aspect of kernel security, we build ChCore which uses the microkernel design and introduces some new mechanisms for reliability. As a microkernel OS, ChCore can greatly mitigate the consequences of security vulnerabilities and it can also work as an TEE OS to cooperate with the TEE hardware technologies. In the aspect of application system security, we propose PiXiu which can provide security guarantee for distributed applications. PiXiu targets on securing sensitive computation, especially for the distributed computation, and it assumes a powerful attack model. Overall, all of the above-introduced research adopt the hardware-software co-designs, and they can fuse with each other to offer a full-stack security system. Meanwhile, the paper will also provide an overview of the representative academic work in each aspect, which includes the comparison of different technical contributations.

operating system security, trusted execution environment security, virtualization security, kernel security, application system security

doi: 10.1360/TB-2022-0557