一种抵御内部人员攻击的云租户密钥保护方法

何 运[1,2], 贾晓启[1,2], 刘 鹏[3], 张伟娟[1]

[1] 中国科学院信息工程研究所 北京 中国 100093
[2] 中国科学院大学网络空间安全学院 北京 中国 100049
[3] 宾夕法尼亚州立大学 宾夕法尼亚 美国 16802

摘要 云计算作为一种新兴计算模式, 近几年来对传统 IT 架构产生了巨大影响。然而, 云计算也面临着新的安全挑战, 例如, 存储在云虚拟机内存中口令、密钥等易受到云平台内部人员发起的攻击。恶意云运维人员可通过简单命令获取云虚拟机的内存快照, 再从内存快照中提取敏感数据(称作内存快照攻击)。本文为保护虚拟机内的加密密钥免受内存快照攻击, 提出 HCoper 方案, HCoper 在 CPU 内部完成所有加密计算, 保证密钥不被加载到 RAM 中。HCoper 采用 key-encryption-key 结构实现密钥动态调度, 以支持多应用多密钥场景。主密钥存储在 CPU 寄存器中, 数据加密密钥由主密钥加密后存储在 RAM 中。HCoper 执行加密计算时, 数据加密密钥将被解密并直接加载到 CPU 寄存器进行加密计算。HCoper 作为 Xen 的内核模块, 可防止其他进程访问持有密钥的 CPU 寄存器。HCoper 旨在为租户提供加密计算服务, 同时保证密钥(即主密钥, 数据加密密钥)不受内部恶意人员的攻击。实验结果表明, HCoper 可有效地防御内部人员发起的内存快照攻击, 其带来的性能开销不影响实用性。

关键词 内存快照攻击; 内部人员攻击; 密钥保护; 云计算
中图法分类号 TP393.08 TP309.2 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2021.05.12

A Method of Protecting Tenants' Secret Keys against Insider Attacks

HE Yun[1,2], JIA Xiaoqi[1,2], LIU Peng[3], ZHANG Weijuan[1]

[1] Institute of Information Engineering, Chinese Academy of Science, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Science, Beijing 100049, China
[3]Pennsylvania State University, Pennsylvania 16802, U.S.

Abstract Cloud computing has been seen as the next innovative computing model and has made a tremendous impact on the traditional Information Technology (IT) architecture over the past years. However, cloud computing also faces new security challenges. For example, the cryptographic keys or passwords in the guest VM's memory are vulnerable to memory-based attacks (e.g., memory dump attacks) launched by malicious insiders. A rogue cloud operator can take a memory dump of the guest VMs by executing simple commands, then extracts sensitive data (e.g., plaintext of secret keys) from the memory dump files. In this paper, to protect the customer's secret keys against memory dump attacks, we proposed an approach named HCoper, which implements all cryptographic computations entirely within the CPU, without any secret keys loaded into the RAM. HCoper is a key-encryption-key architecture performing dynamic scheduling of secret keys to support multiple keys for multiple applications. The master key is stored in CPU registers, the data-encryption keys are encrypted by the master key and then stored as cipher-text in the RAM. When HCoper is working, the data-encryption keys will be decrypted and then directly loaded into CPU registers for encryption computation. We implement HCoper as a kernel module of Xen to prevent other malicious processes from accessing the CPU registers that hold the master key or data-encryption keys. HCoper provides the tenants with cryptographic computation services that are secure against memory dump attacks launched by malicious insiders. Meanwhile, experiments demonstrate that our implementation of HCoper defends against insider threats effectively and it only introduces reasonable performance overhead.

Key words memory dump attacks; insider attacks; key protection; cloud computing

通讯作者: 贾晓启, 博士, 研究员, Email:jiaxiaoqi@iie.ac.cn。
本课题得到中国科学院网络测评技术重点实验室资助项目, 网络安全防护技术北京市重点实验室资助项目, 北京市科技计划课题(No.Z191100007119010), 国家自然科学基金(No.61772078)资助。
收稿日期: 2019-05-31; 修改日期: 2019-09-22; 定稿日期: 2021-03-05

1 引言

云计算作为一种新兴计算模式, 由于其低成本、灵活性、高度自动化等特点而备受关注。越来越多的企业将其应用程序和数据迁移到云平台, 文献[1]调查显示几乎有一半的公司表示其 IT 系统中有 31%~60%是基于云平台进行部署的。此外, 预计到 2024 年全球云计算市场将突破 1 万亿美元[2]。然而, 云计算迅速发展的过程中也面临着诸多安全挑战。近期的调查报告[3-5]指出云计算安全问题已成为企业部署或使用云平台时首要考虑的因素。云安全联盟(Cloud Security Alliance, 简称 CSA)发布的一份报告[6]归纳了云计算中的 12 大安全威胁, 其中恶意内部人员攻击(Insider Attacks)排名第六。此类攻击通常由某个组织中(如云服务提供商)的恶意内部人员发起。例如, DuPont 公司的商业机密被一位前工程师偷走并泄露给竞争对手[7]。2016 年美国网络犯罪状况调查发现, 27%的网络犯罪是由内部人士造成的[8], 该调查还显示, 30%的受访者认为内部人员攻击造成的损害比外部攻击造成的损害更加严重。针对云平台环境, 云服务提供商的恶意内部人员很容易窃取客户虚拟机的敏感数据[9-10]。考虑一个常见场景: 当客户虚拟机中的某个进程正在执行加密(或解密)操作, 此时云服务提供商的某些恶意内部人员, 如云平台运维人员(Cloud Operators), 可能会对客户虚拟机执行内存快照(Memory Dump)操作, 然后可以轻松地从内存快照文件中恢复密钥, 本文将此类攻击称作内存快照攻击(Memory Dump Attacks)。

针对恶意内部攻击, 文献[11-14]通过分析内部人员的行为记录来检测未经授权或非法访问云租户数据的异常事件。然而, 这些方法并不总是奏效, 因为客户虚拟机中的敏感数据已经被窃取, 即便后续发现了此类攻击行为。为了防止恶意内部人员对云租户数据进行窥探或篡改, TCCP[15]为客户虚拟机提供一个封闭的执行环境, 将客户虚拟机运行于该执行环境中。但 TCCP 尚未实现其方案的原型系统, 并且该方案需要依赖第三方信任代理。Overshadow[16]设计了一种 mutli-shadowing 机制, 该机制为物理内存的提供不同视图(view)。具体而言, Overshadow 为应用程序提供其内存页的明文视图, 而为客户操作系统提供加密视图, 其目的在于保护客户操作系统内运行的应用程序的机密性和完整性。然而, Overshadow 机制中的加密计算过程都是在内存中完成, 用于加密的密钥也存储在内存中。换句话说, Overshadow 也不能抵御内存快照攻击。

Intel SGX[17]是 Intel CPU 的一套指令集扩展, 它提供硬件支持的可信执行环境(被命名为 Enclave)来运行受信任的代码。受信任的代码和数据驻留在受 CPU 保护内存区域中, 该内存区域为 Enclave Page Cache(EPC)。SGX 仅允许在 Enclave 中执行的受信任代码访问 EPC。EPC 内存区域受硬件加密引擎(Memory Encryption Engine)保护。即使是不受信任的操作系统或虚拟机管理程序(Virtual Machine Monitor, 也称作 Hypervisor)也不能破坏受硬件保护的 Enclave。理论上, 内存快照攻击对 SGX 无效, 但将 SGX 应用到云平台时会存在一些限制, 主要有以下几个方面。a) 内存资源限制。为 Intel SGX 保留的物理内存最多为 128MB, 而实际可用的为 93MB。虽然可通过在 EPC 和非 EPC 之间执行内存换入/换出来支持 Enclave 访问超过 128MB 的虚拟地址空间。但这将导致代价高昂的上下文切换(即 Enclave 进入/退出操作), 并导致性能显著下降[18]。b) 侧信道攻击威胁。Intel SGX 无法抵御侧信道攻击(side-channel)。例如, 在基于页面错误(page fault)的侧信道攻击[19]中, 不受信任的操作系统可以通过页面错误信息推测 Enclave 访问内存页面的规律进而窃取敏感数据。此外, 最近的研究[20-21]表明, Intel SGX 也容易受到基于缓存的侧信道攻击。c) 实际部署限制。在客户操作系统中运行 SGX 应用程序需要虚拟机管理程序(Hypervisor)的支持。然而, 支持 SGX 的开源虚拟机管理程序, 如 kvm-sgx[22]和 xen-sgx[23]还处在研发阶段尚不能用于实际部署支持 SGX 的云虚拟机。

针对上述由恶意内部人员发起的内存快照攻击(Memory Dump Attacks), 本文提出并实现了一个名为 HCoper(Hypervisor Cryptographic operations outside RAM)的解决方案, 该方案在内存外部执行加密算法, 整个加密运算过程仅在 CPU 内部完成, 只涉及 CPU 寄存器和缓存(Cache)的使用, 进而使得整个加密计算过程中不会将任何机密数据(如数据加密密钥)或中间结果导出到系统内存中。从而使得内存快照攻击无法从内存镜像文件中恢复出数据加密密钥。此外, HCoper 不受侧信道攻击的影响(将在第 6.2 节中讨论), HCoper 可方便地部署, 而无任何内存限制。具体地, HCoper 采用一种通用的 key-encryption-key 结构, 主密钥存储在寄存器中, 数据加密密钥由主密钥加密后以密文状态存储在 RAM 中。本文中的主密钥特指用于加密其他密钥的密钥, 主密钥不用于数据加密用途。相应地, 数据加密密钥(也称加密密钥)特指用于数据加密用途的密钥, 数据加密密钥是主密钥加密的对象。执行加密(或解密)计算任务时,

密文状态的数据加密密钥会被动态解密后直接加载到寄存器中, 使得明文态的密钥永远不会被加载到RAM 中。存储主密钥或数据加密密钥的寄存器属于特权资源, 只允许 HCoper 对其进行访问。对这些寄存器的任何读写操作都将被 HCoper 捕获并拦截(见4.2 节)。此外, HCoper 提供动态调度密钥的机制, 以支持多应用多密钥场景。

本文利用 Intel AES-NI[24]机制实现了 HCoper 的原型系统, Intel AES-NI 支持在 CPU 内完成加密算法(即 AES)。特别地, HCoper 将主密钥存储在 MSR 寄存器(即 x86 Model Specific Registers)中。当 HCoper 运行时, 数据加密密钥临时加载到调试寄存器(Debug Registers)中完成加密运算。HCoper 以一个内核模块的形式集成到开源虚拟机管理程序 Xen[25]内核中, 并为客户虚拟机提供加密接口。理论上, 其他虚拟机管理程序(如 kvm[26]也适用于 HCoper。此外, 本文已经将 HCoper 提供的加密接口集成到 OpenSSL 库中, 以便于开发第三方应用程序。本文在 Intel Core i7 4790 CPU 上运行并评估 HCoper 原型系统。实验结果表明, HCoper 能够有效防御内存快照攻击。与原始的加密实现方案(OpenSSL)相比, HCoper 具有良好的加密运算性能。同时, HCoper 对并发型应用程序性能的影响很小(不影响实用性)。

2 相关工作

2.1 云环境中的内部人员攻击

内部人员攻击通常指由内部人员滥用组织给予的信任和权力实施的非法活动[9], 如窃取、窥探用户隐私数据等。例如, 恶意的云平台运维人员(Cloud Operator)可能会实施精心策划的攻击, 以窃取客户的敏感数据。这种攻击通常比较隐秘且难以察觉, 但极具破坏性, 因为运维人员通常对云平台中不同的系统有着不同级别的访问权限, 并且他们对云平台的基础设施部署情况有比较详细的了解, 这使得恶意运维人员可以更加容易地实施窃密攻击。相对而言, 外部攻击者则需要花费很长时间才能探查出云平台基础设施的部署情况。

正如文献[10]所述, 恶意内部人员可以很轻易地破坏客户虚拟机内部数据的机密性而不需要实施任何高超的渗透手段。其中, 比较简单却有效的攻击手段通常是从客户虚拟机内存快照或内存镜像文件中提取敏感数据。实施此类攻击时, 恶意内部人员通常通过执行一些 Hypervisor 提供的管理指令(如 Xen 提供的“xl save”指令)先获取目标虚拟机的内存快照(Memory Dump)。接着, 可以使用简单的文本分析工具从内存快照文件中提取隐私数据(如明文口令等)。更进一步, 攻击者可以使用类似冷启动攻击[27]中介绍的攻击手段或工具从客户虚拟机内存快照文件中恢复出加密密钥, 如 AES 的加密密钥。文献[27]已证实可从内存快照文件中恢复出 AES 加密密钥。

在虚拟机克隆攻击[9](IaaS VM Cloning Attacks)中, 客户虚拟机本质上也是存储在磁盘上的文件, 可能会被备份、复制或转移到其他服务器上。相应地, 恶意内部人员可能会复制客户虚拟机对应的镜像文件, 并在外部私有的 Hypervisor 上重新加载和启动该虚拟机。利用暴力破解工具获得虚拟机登录口令后, 进入克隆虚拟机中窥探用户隐私数据。类似的, 针对 DaaS(Data Storage as a Service)场景, 恶意内部人员可能会滥用权限非法访问客户文件, 再借助一些取证技术或工具从文件中盗取隐私数据(如口令, 密钥等), 此类攻击被称作 DaaS File Copying Attacks[9]。本文将重点关注内存快照攻击(Memory Dump Attacks), 虚拟机克隆攻击(VM Cloning Attacks)和 DaaS File Copying Attacks 这三类攻击。

2.2 云环境中的内部威胁缓解方案

文献[28]分析了不同级别管理人员及其所对应的潜在安全威胁等级。文献[9]讨论了不同类型的内部攻击向量以及不同类型的恶意内部人员。针对恶意内部人员带来的安全威胁, 一些研究者提出基于内部人员行为分析来检测和识别内部威胁。如基于图的分析(Graph-based analysis)方案[12], 主动分析(Proactive analysis)方案[13]和蜜罐(Honeypot)分析方案[11]等。文献[14]通过记录系统调用(System call analysis)活动来分析用户行为特性进而检测出恶意内部人员实施的异常行为。在文献[29-30]中, 类似的度量和策略, 如心理分析(Psychological Profiling)、用户分类(User Taxonomy)被用于预测和检测潜在的恶意内部人员。在文献[31]中, 击键动态技术(keystroke dynamic technology)被用来检测恶意的内部人员。

TCCP[15]设计一个可信的云计算平台(trusted cloud computing platform, 简称 TCCP), 使 IaaS 服务提供商能够为客户的虚拟机提供一个封闭的执行环境。即使 IaaS 服务提供商的管理人员也无权访问客户虚拟机内部的机密执行环境。然而, 该方案仅从理论层面提出 TCCP 构想, 并未实现原型系统。此外, TCCP 需依赖于第三方可信机构。文献[32]得出结论: 将技术与有效的管理策略相结合才是缓解内部威胁的最好的解决方案。文章建议管理员在执行任何操作之前必须向参与管理的其他 k 个管理员提交申请许可。只有其他 k 个管理员都许可该操作时才能继续

往下执行。但该方案需要较长的时间来响应客户的请求。

Fog Computing [33]利用诱骗信息(decoy information)技术对恶意内部人员进行虚假信息攻击(disinformation attacks), 使得攻击者无法区分出真实的敏感数据和虚假无用的数据。MyCloud[34]提出一种新的虚拟化架构削减云服务提供商的管理特权, 使控制域虚拟机(类似于 Dom0)没有访问其他客户虚拟机资源(如内存)的特权。此外, MyCloud 还提供了一个访问控制矩阵, 供客户根据需要配置隐私策略, 该矩阵定义了能够访问虚拟机的实体。理论上, MyCloud 可以抵御内部威胁, 但 MyCloud 不具备和 Xen 一样的实用性。

表 1 将 HCoper 与其他现有的针对云环境内部威胁的缓解方案进行了比较。本文主要关注第 2.1 节中描述的三种类型的内部攻击(Memory Dump Attacks, VM Cloning Attacks, DaaS File Copying Attacks)。如表 1 所示, HCoper 可以有效防御此三种内部攻击。此外, HCoper 是一种适用于云平台的实用性解决方案。虽然 TCCP[15]可以抵御这三类内部攻击, 但其无原型系统并需依赖第三方机构。MyCloud[34]实用性较弱。其他基于策略的[29-30,32]方法、行为分析方法[12-14,31]或基于蜜罐的[11]方法均专注于检测恶意内部人员, 但它们无法抵御内存快照攻击。

表 1 三种典型攻击向量下的现有缓解方案对比

Table 1 Comparison of Mitigation under Three Attack Vectors

类型	代表方案	攻击向量		
		Memory Dump Attacks	VM Cloning Attacks	DaaS File-Copying
Behavior Analysis	Graph based analysis[12]	X	X	X
	Keystroke dynamic [31]	X	X	X
	Proactive analysis [13]	X	X	X
	System call analysis [14]	X	X	X
Policy & Detection	User Taxonomy [29,30]	X	X	X
	Mutual authorization [32]	X	X	X
Decoys & Honeypots	Fog Computing [33]	X	X	X
	Honeypot [11]	X	X	X
Isolated Execution Environments	TCCP [15]	√*	√*	√*
New Hypervisor	MyCloud [34]	√*	√*	√*
CPU-bound	HCoper	√	√	√

(注: X 表示该方案不能阻止相应的攻击窃取密钥; √*指表示该方案虽能保护密钥免遭对应的攻击, 但具有局限性(如只是理论方案, 不具有实用性); √代表该方案能保护密钥安全, 使得相应的攻击失效。)

2.3 保护加密密钥

为了防御冷启动攻击[27], TRESOR [35]、AESSE[36]和 Amnesia[37]通过将 AES 密钥存储在寄存器中来增强全盘加密的安全性。值得注意的是, 冷启动攻击本质上也是内存快照攻击。Copker[38]使用受 TRESOR [35]保护的 AES 密钥作为主密钥来加密 RSA 的私钥, 并在 CPU 缓存中实现 RSA 算法。PRIME [39]与 Copker 类似, 它在 AVX[40]寄存器中完成 RSA 中安全敏感(Security-sensitive)的操作, 将不影响安全性的中间值保存在 RAM 中, 实现一种 memory-less 的 RSA 方案。Mimosa[41]通过使用 Intel TSX[42]机制进一步增强 Copker 的安全性, 以确保一旦检测到攻击者试图读取 Cache 中的密钥, 敏感数据将被自动清除。Mimosa 的目的是阻止恶意线程窃取 CPU 缓存中的私钥。

基于寄存器的 AES 实现方案容易受到基于 DMA 的高级攻击[43], 这些攻击可以将恶意代码注入操作系统内核中, 然后访问存放密钥的寄存器。但是, BARM[44]提供了一种检测基于 DMA 的攻击的方法。另外, 还可以通过配置 IOMMU[45]和 SMM[46]来防御高级 DMA 攻击。

PixeValut[47]在 GPU 中执行整个加密计算过程, 所有敏感数据都存储在 GPU 的缓存和寄存器中。保证 CPU 上运行的恶意进程无法访问 GPU 上的数据, 即使恶意的或被攻陷 OS 内核也无法窃取任何敏感数据。CaSE[48]利用 ARM 处理器的 TrustZone[49]技术创建了一个基于缓存的独立执行环境。将加密算法(例如, AES、RSA、SHA1)运行在该独立的执行环境中。CaSE 能有效地防御软件级和硬件级内存泄漏攻击(如冷启动攻击)。

上述现有工作均主要集中在个人计算机或移动设备上实现抵御内存攻击的加密运算, 均需要进一步的扩展和改进, 因此无法直接适用于云计算平台。

3 系统设计

3.1 系统模型

如文献[50]所述, 云平台中的管理任务主要由三个不同组别的成员完成:

(1)云平台管理员(Cloud Administrator): 云管理员通常是云服务提供商指定的极少数高级别员工(人数少, 易受监控)。云服务提供商授予其配置、初始化和启动云平台的权限。云管理员负责创建云平台服务条款, 并监管、审查服务条款的执行情况。

(2)云平台运维人员(Cloud Operator): 云运维人员负责完成来自用户的配置请求(例如, 创建虚拟机等)。云运维人员负责管理和维护云平台的日常运营工作, 例如, 解决在资源调配或虚拟机运行过程中发生的错误。更重要的是, 它们通常代表客户执行一些敏感操作, 例如, 当执行云平台故障修复任务时, 云运维人员可能会暂停, 终止, 重启虚拟机; 该过程中会伴有获取和恢复虚拟机快照等操作。通常云运维人员的数量远超过云管理员的数量, 这使得云服务提供商难以监督所有云运维人员的日常操作行为。

(3)云用户(Cloud User): 或云租户, 他们向云服务提供商申请虚拟机资源, 并使用云服务提供商提供的用户管理接口来管理分配给他们的任何虚拟机。

显然, 我们可以区分受信任的云管理员和不受信任的云运维人员。云服务提供商通常只会委派少数且可信云管理员进行初始化云平台。云管理员最能代表云服务提供商的利益。通常云管理员(即代表云服务提供商)有责任保护客户的隐私, 以维护企业良好的形象和声誉。本文假设云平台管理员为可信的, 不会主动实施内存快照攻击。然而, 云平台启动结束后云平台运维工作交给其他运维人员, 云运维人员不可信且数量通常较多, 甚至运维工作可能会被外包给第三方人员。因此, 数目众多的(难监控的)云运维人员可能会恶意实施内存快照攻击, 窃取客户虚拟机中的敏感数据。本文主要目标是防御恶意云运维人员发起的内存快照攻击。

3.2 威胁模型

云运维人员通常有机会访问云平台设施(如Dom0), 其可能会有意滥用职权执行违规操作从而对云平台的保密性和完整性产生负面影响。例如, 借助专业工具, 如 aeskeyfind[27], 从客户虚拟机的内存快照文件中提取隐私数据(如口令, 密钥等)。此外, 远程攻击者也有可能对客户虚拟机发起类似的内存攻击, 他们通常利用虚拟机管理程序的软件漏洞来攻击客户虚拟机, 然后对整个客户虚拟机的内存数据或者某一个受害进程作内存快照, 以从内存快照文件中提取敏感数据。此外, 硬件平台(如 CPU)的漏洞也可能为攻击者提供一种可行的方法来转储(Dump)整个系统内存。如针对 Intel CPU 的 meltdown[51]漏洞, 攻击者可以利用该漏洞转储系统内存以窃取隐私数据。

本文假设 Hypervisor 具备代码完整性, 且无任何漏洞可供攻击者利用以执行恶意代码来破坏 Hypervisor 的内核。本文假设恶意的云运维人员(其职责如第 3.1 节中所述)能够执行常规的维护管理 shell 命令(例如“xl save”)。此外, 本文还假设在密钥初始化期间(很短的时间), 整个 Hypervisor 系统都是安全的。在此期间, 获得云服务提供商授权的云管理员会初始化 HCoper 的主密钥和密钥。一旦密钥初始化完成, 即使在特权域(Dom0)内安装 rootkit 也无法窃取密钥。这是因为 Dom0 内核的 CPU 权限级别比 Hypervisor 低。因此, Dom0 内核中的 rootkit 不可能篡改 Hypervisor 的内核内存(例如, hook 超级调用表)。据本文所知, 在不利用 Hypervisor 内核漏洞的情况下无法直接将 rootkit 安装到 Hypervisor(如 Xen)中。而如何挖掘和修复 Hypervisor 的漏洞超出了本文的研究范围。

3.3 设计目标与原则

云平台场景中, 攻击者对云虚拟机作内存快照, 内存快照通常是无语义信息的二进制数据, 攻击者需要还原出二进制数据的语义信息才可获取到有价值的信息(如加密密钥等)。再次, 攻击者通常不会无休止地对云虚拟机作内存快照, 这一方面增大攻击成本(时间开销和存储开销), 另一方面, 频繁的内存快照操作容易暴露攻击行为。相反, 加密计算过程(如 AES 算法)通常表现出固定的规律, 加密算法使用的参数、密钥等会以某种特定的格式存储在内存中, 这使得攻击者能以可接受的攻击成本恢复出加密密钥的语义信息, 如文献[27]中根据 AES 加密密钥在内存中存储的特定规律和格式成功从内存快照中恢复出 AES 密钥。因此, 窃取加密密钥成为攻击者的首选目标。此外, 从云用户的角度来说, 通常采用加密处理(如加密传输)的数据都是少量但重要的隐私数据, 一旦攻击者获取到加密密钥, 再辅以其他攻击手段可获取到更多的隐私数据(如网站的登录凭证等)。因此, 针对云平台环境, 密钥泄露对云用户造成的破坏性更大。

为保护客户虚拟机在执行加密运算时, 其使用

的加密密钥不被窃取。首要的设计目标是确保主密钥和加密密钥在任何时候都不会被加载到系统内存(RAM)中。此外, 加密计算过程中的中间结果也不能出现在 RAM 中。本文提出的方案中, 将主密钥永久存储在 CPU 寄存器中, 而加密密钥和计算过程的中间状态都只是临时暂存在寄存器中。然而, 虚拟化平台(如 Xen)通常会允许虚拟机之间共享硬件资源, 目前 Xen 的实现方案默认允许虚拟机直接访问 CPU 的寄存器(如 Debug 寄存器)。因此, 另一个设计目标需要防止客户虚拟机(包括特权域 Dom0)访问存储着密钥的寄存器以窃取密钥。总体而言, HCoper 的目标是为云租户提供一个通用、高效的加密计算服务, 同时保证该加密运算服务使用的密钥不会被攻击者(特别是恶意内部人员)窃取。基于上述目标, HCoper 的设计需要满足以下几个要求(准则):

(1)在执行数据块加密运算(通常是最小的加密运算执行单位)时, 该运算过程不能被中断。否则, 很可能在执行进程调度的过程中, 在将进程状态(包括寄存器状态)导出(swap)到 RAM 中时也将密钥或运算的中间结果导出到 RAM 中, 从而导致密钥被窃取。

(2)针对多核(multi-core)CPU, 同一个加密进程可能会被调度到不同的核(core)上去执行。而不同的 CPU 核内部维护着不同的寄存器状态。因此, 需要确保当同一个加密进程不同时刻运行在不同的 CPU 核上时, 其加密运算使用的加密密钥仍保持一致。

(3)客户虚拟机可能会有意访问存储着密钥的寄存器。因此, 需要确保客户虚拟机访问保存有密钥的寄存器的行为能被捕获并拦截。

为了满足原则(1), HCoper 的块加密计算必须在一个原子域中运行, 在该原子域中将暂时禁用所有中断, 并暂时禁用 CPU 抢占。同样, 加密密钥的加载和调度也需要放置在原子域中执行, 以确保数据加密密钥不被泄露, 同时保证在执行块加密之前数据加密密钥正确无误地被载入到寄存器中。需注意的是, 块加密运算(最小加密运算执行单位)通常很小, 如 AES 为 16 字节(128bit), 块加密计算的原子域是在极短暂的时间内完成, 完成块加密计算后会立即恢复中断和 CPU 抢占, 而不是在运行整个加密进程的过程中都禁用 CPU 抢占和中断。块加密(原子操作)计算完成后恢复中断和 CPU 抢占, 使得加密进程和其他进程一样处于同等的地位, 系统执行进程调度算法正常调度进程(包括加密进程)获取 CPU 执行权限, 对系统内其他进程的影响很小。在下一节(第 3.4 节)中, 将介绍满足原则(2)的设计细节。满足原则(3)所使用的技术实现细节将在第 4.2 节中给出。

3.4 HCoper 设计细节

如图 1 所示, HCoper 主要由五个组件构成。同时, HCoper 是一个三层模型结构: 用户层(User Space), 虚拟机内核层(Guest Kernel)和虚拟化平台层(Hypervisor)。位于用户层的应用程序通过 ioctl 接口将数据加密请求提交给虚拟机内核中的 gCoper(guest Coper)模块, gCoper 模块负责将接收到的加密请求通过调用 Coper 提供的超级调用(hypercall)接口提交给 Hypervisor 内核中的加密模块 Coper。Coper 模块在 CPU 内部完成相应的加密计算任务。HCoper 的执行流程主要分为两个阶段: 初始化阶段(initialization phase)和数据加密阶段(data-encryption phase)。

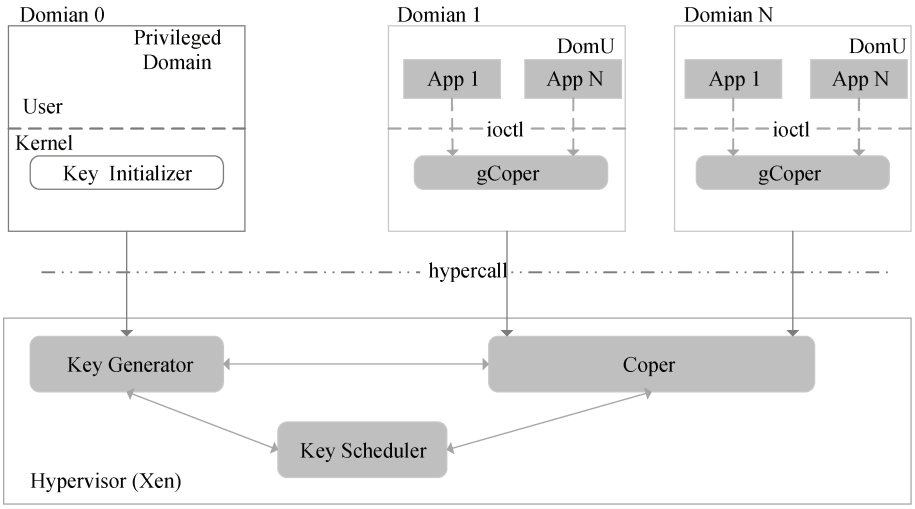


图 1 HCoper 系统架构图

Figure 1 The Architecture of HCoper

在初始化阶段, 获得云服务提供商授权的云平台管理员在特权域(Dom0)中临时安装 Key Initializer 模块, 该模块的主要功能是负责接收管理员输入的长字符串(如 password)并将该 password 提交该 Hypervisor 内核中的 Key Generator 模块。Key Generator 模块利用收到的 password 生成主密钥和数据加密密钥。具体地, Key Generator 首先生成主密钥, 并立即将主密钥加载到 MSR 寄存器中。HCoper 中只有一份主密钥用于加密解密其他数据加密密钥, HCoper 中维护着多个加密密钥(理论上加密密钥个数不受限制)。接着利用再次收到的 password 生成加密密钥, 加密密钥将会被主密钥加密后以密文的形式存储在系统内存(RAM)中。HCoper 支持多应用多密钥场景, 因此在初始化阶段云管理员会输入多次 password 并相应的生成多个数据加密密钥。值得注意的是, 密钥初始化工作是在一个很短的时间内完成, 本文假设在此短暂的初始化期间, 整个 Hypervisor 平台是安全的。另外密钥的初始化工作通常在系统启动期间完成(即 Boot 阶段), 在此期间云平台管理员通过外设输入 password, 在此短暂期间内 password 会暂留 RAM, 一旦根据 password 生成主密钥或数据加密密钥后, RAM 中的 password 会被彻底清除。密钥初始化阶段通常是 Hypervisor 平台的启动阶段, 此时平台尚未启动完成, 云虚拟机尚未启动, 通过云虚拟机的攻击在此阶段无效。此外, 系统启动工作尚未结束使得无法实施内存快照攻击, 因为内存快照执行过程中需要调用 Hypervisor 向外提供的接口。

Key Initializer 模块是云平台管理员临时安装到 Dom0 中的, 只有云管理员有权限使用该模块。该模块调用 Key Generator 提供的隐式 hypercall 完成密钥的初始化工作, 该hypercall不对外公开, 即云运维人员不知道该 hypercall 的任何细节。需注意的是, 该 hypercall 每次被调用时都会先执行合法性验证, 验证调用者是否运行在特权域(Dom0)中, 如果是非特权域的调用则直接拒绝, 若是来自特权域的调用还需要进一步判断调用者是否是 Key Initializer 模块, 具体地当 Key Initializer 模块调用隐式 hypercall 时, HCoper 能捕获到调用者 EIP 寄存器信息, 进而可以获取调用者执行 hypercall 时, 控制流上下文信息, HCoper 预先保存了一份 Key Initializer 调用该隐式 hypercall 时的控制流上下文信息, 每次都以同样的方式获取调用者执行 hypercall 时的控制流上下文信息, 然后再进行对比(匹配)进而判断调用者是否是 Key Initializer 模块。云管理员后续可以随时在重新安装 Key Initializer 模块更新主密钥和数据加密密钥。另外, 在完成密钥初始化工作之后, Key Initializer 模块会被从 Dom0 中卸载并移除。Key Generator 模块生成密钥时, 对收到的 password 执行一定次数(在本方案中至少计算 2000 次)的 SHA-256 哈希计算后获得的摘要值(digest)作为最终的密钥。HCoper 将主密钥存储在原本用于硬件性能计数[52](hardware performance counter)的 MSR 寄存器(涉及 4 个 MSR 寄存器)中。本文选择该组 MSR 寄存器时主要考虑到硬件性能计数特性并不是大多数软件必需依赖的特性。存储主密钥仅占用 4 个用于硬件性能计数功能的 MSR 寄存器而不是所有的 MSR 寄存器。另外, 部署在云场景中的业务很少使用该组 MSR 寄存器, 在云虚拟机内部进行性能评测(如, 评测虚拟机内程序运行时间开销)是不推荐的。这是由于云虚拟机运行在 Hypervisor 之上, 云虚拟机占用的资源均由 Hypervisor 调度, 云虚拟机性能受云平台整体负载情况的影响。因此, 即使 HCoper 关闭了硬件计数功能也不会对大多数软件的正常运行造成影响。数据加密密钥的初始化过程和主密钥基本一致, 不同点在于生成主密钥后直接加载到 MSR 寄存器中, 而数据加密密钥生成后先被主密钥加密后以密文形式存储在系统内存中。只有在执行加密运算时才会被解密后加载到 Debug 寄存器内(该加载过程是寄存器之间的数据拷贝行为, 不涉及内存的读写)。数据加密密钥的解密过程也是在 CPU 内部完成, 因此, 不会造成密钥泄露的风险。HCoper 同样独占了 Debug 寄存器, 使得硬件调试功能被禁用, 软件调试功能不受影响。对于大多数软件来言, 硬件调试也不是必需的功能, 因此, HCoper 并不会破坏大多数软件的兼容性。总体而言, HCoper 的密钥初始化工作可以简要的归纳为三个步骤, 如图 2 所示: (1)Key Generator 利

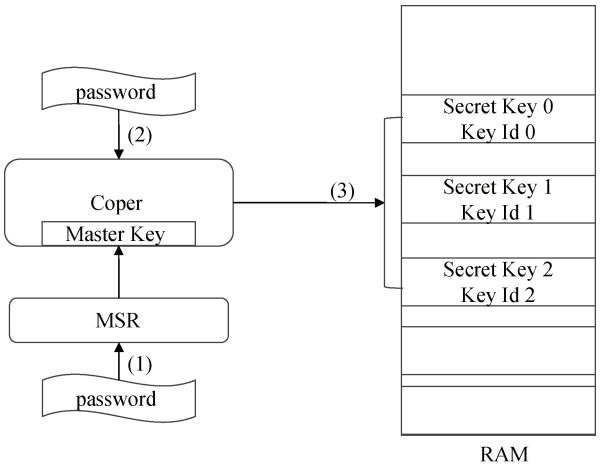


图 2 密钥初始化

Figure 2 Initialization of Keys

用 password 生成主密钥(master key)并立即将其载入 MSR 寄存器; (2)Key Generator 利用再次收到的 password 生成数据加密密钥, 接着使用 MSR 寄存器中的主密钥对数据加密密钥进行加密处理; (3)密文态的数据加密密钥被存储到系统内存中。

数据加密阶段, 用户层应用程序通过 ioctl 接口提交加密请求, 该请求中包含待加密明文的内存地址, 密文存储地址以及所使用的加密密钥标识符。HCoper 对每个加密密钥都用唯一的 *key-id* 进行标识。gCoper 将加密请求提交该 Coper 模块后, Coper 根据 *key-id* 载入对应的数据加密密钥并完成相应的加密运算。数据加密密钥的载入与调度工作实际上是由 Key Scheduler 模块完成。

加密密钥调度, 为了支持多应用多密钥场景, Key Scheduler 模块维护了一个加密密钥使用记录列表。该列表记录了最近被使用过的加密密钥, 列表记录的只是加密密钥对应的 *key-id*, 而列表的索引值为 CPU 核序号(*core-id*)。例如, 某一时刻, CPU 核 2 上运行的加密进程使用的加密密钥的 *key-id* 为 3, 则该列表中索引值(或下标)为 2 的元素值为 3。当在处理某个加密请求时, Key Scheduler 首先需要获取当前加密进程所在的 CPU 核的 *core-id*, 然后以此 *core-id* 为索引到列表中进行查询获得最近被载入到该 CPU 核上的加密密钥的 *key-id*(记为 *L-key-id*)。再将 *L-key-id* 与用户提交的加密请求中指定的加密密钥的 *key-id*(记为 *R-key-id*)进行对比。若 *L-key-id* 与 *R-key-id* 一致, 则直接使用该 CPU 核中存留在 Debug 寄存器中的加密密钥, 需注意的是, 为了减少从系统内存解密并载入加密密钥的次数, 存储在 Debug 寄存器中的加密密钥在加密运算任务结束后依旧存留而不会被清除。若 *L-key-id* 与 *R-key-id* 不一致, 则需要使用 MSR 寄存器中主密钥解密系统内存中的密文态的加密密钥, 并将解密后的加密密钥直接载入 Debug 寄存器。由于整个解密(或加密)运算过程都是在 CPU 内部完成, 具体而言, 均使用 CPU 内存的寄存器来暂存密钥、中间状态值以及最后的解密结果都是暂存在寄存器中。因此, 解密后的加密密钥也是暂存在寄存器中, 在将其载入 Debug 寄存器时也仅仅是寄存器之间的数据拷贝而不会涉及任何内存读写操作。因此, 解密后的加密密钥被直接拷贝到 Debug 寄存器而不会泄漏到系统内存中。

图 3 简要地归纳了数据加密阶段中的密钥调度过程: 1)当 *L-key-id* 和 *R-key-id* 不一致时, Key Scheduler 模块使用 MSR 寄存器中的主密钥解密加密密钥。加密密钥的密文存储在 RAM 中, 本质上是一个密文列表, 使用 *key-id*(如 *R-key-id*)为索引载入对应加密密钥的密文。2)解密后的加密密钥被直接载入 Debug(简记为 db)寄存器, 该过程不涉及任何内存读写操作。3)Coper 完成最后的加密计算任务, 从内存中载入待加密数据(Input), 将加密结果写回内存中(Output)。在完成加密计算后, Coper 会将计算过程中使用到的寄存器(如 Streaming SIMD Extensions[53], 简称 SSE)的状态值都清零, 只保留 Debug 寄存器中加密密钥以备下一次使用。

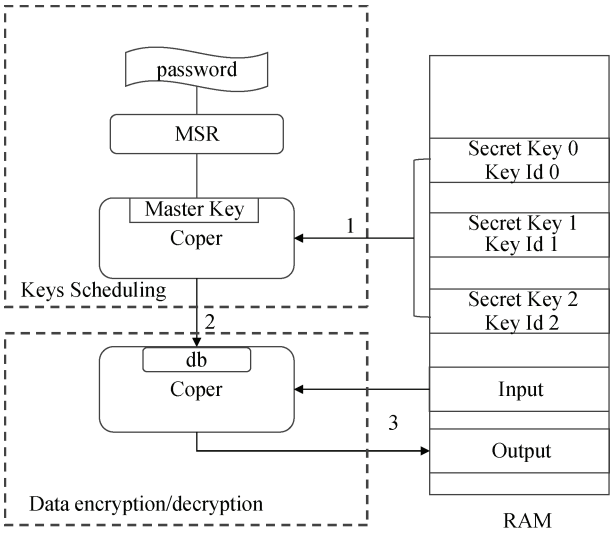


图 3 动态密钥调度

Figure 3 Dynamic Scheduling of Keys

4 系统实现

4.1 加密接口

本文基于开源虚拟化平台 Xen 4.4.0 实现 HCoper 原型系统。HCoper 利用 Intel AES-NI 指令集在 CPU 内实现加密计算, 利用 x86 架构的调试寄存器(Debug Registers)存储数据加密密钥, 利用 MSR 寄存器存储主密钥。Intel AES-NI 利用 SSE[53]寄存器存储加密密钥、中间状态和 AES 轮次密钥, 从而确保不会将任何中间状态或密钥加载到 RAM 中。HCoper 通过超级调用(hypercall)机制向客户虚拟机的内核提供加密计算服务。客户虚拟机的用户空间进程通过 ioctl 接口调用加密计算服务。本文已将 ioctl 接口集成到 openssl-1.1.0g[54]和 polarssl-2.6.0[55]中, 作为通用 API 为其他第三方应用程序提供开发接口。

超级调用(hypercall)是从虚拟机到 Hypervisor 的软件级陷入(software trap)操作。虚拟机通过超级调用请求特权操作(例如, 更新页表等)。类似地, HCoper 为虚拟机内核提供了两个超级调用, 用于调用加密

和解密服务。在 Xen 的实现中每个超级调用都有全局唯一的超级调用号与之对应。相应的, 在 HCoper 的实现中, 在Xen内核中扩展了两个超级调用, 其超级调用号分别为 41、42。此外, HCoper 还定义了用来封装加解密请求的结构体, 如下图所示:

```
typedef struct {
    unsigned char *dst;
    unsigned char *src;
    unsigned long msglen;
    unsigned long key-id;
}crypto-arg-t;
```

图 4　加密请求结构体

Figure 4　Structure of Encryption Request

用户层应用程序需要将加密请求封装为该结构体, 再提交到 HCoper 内部。其中 *src* 表示待加密(或解密)的数据, *dst* 用于存储加密(或解密)的结果, *msglen* 指示待加密(或解密)的数据大小, *key-id* 指明本次加密计算所使用的加密密钥。

4.2　密钥保护

HCoper 作为 Hypervisor 的一个内核模块运行, 因此, 位于上层的客户虚拟机访问 MSR 寄存器或 Debug 寄存器的行为均能被 HCoper 捕获并拦截, 以保证主密钥和加密密钥不会被客户虚拟机窃取。Intel 虚拟化机制(Intel VT)允许在 Hypervisor 层定义敏感操作, 当上层客户虚拟机执行敏感操作时会陷入(trap)到 Hypervisor 内核中, 即执行权限被转交给 Hypervisor。例如, 客户虚拟机读写 CR3 寄存器以更新页表为典型的敏感操作。在虚拟化场景下客户虚拟机也需要 Hypervisor 的辅助才能访问物理内存。因此, 更新 CR3 须陷入到 Hypervisor 中进行处理。具体地, Intel VT 虚拟化技术提供虚拟机执行控制域字段[56](VM execution control filed)来定义敏感操作。HCoper 通过该字段将 MSR 寄存器和 Debug 寄存器的访问操作定义为敏感操作, 当客户虚拟机访问这两组寄存器时被 HCoper 截获。具体过程如下:

(a)对于 Debug 寄存器, 将 VM execution control 字段上 MOV-DR 标志位设置为 1, 使得虚拟机执行 mov 指令读写 Debug 寄存器时, 触发 VM-exit 事件陷入到 Hypervisor 内核, HCoper 处理该事件并阻止访问 Debug 寄存器。

(b)类似的, 在 VM execution control 字段上将 RDMSR 和 WRMSR 标志位设置为 1, 截获上层虚拟机访问 MSR 的行为。

(c)针对半虚拟化场景, 不能使用上述步骤中的方法。本文修改(patch)了 Hypervisor(即 Xen)中原本用于访问 MSR 寄存器和 Debug 寄存器的超级调用, 将对 MSR 寄存器和 Debug 寄存器的读写操作重定向到一块固定内存区域, 使得半虚拟化虚拟机不能读写真实的硬件寄存器, 从而保证密钥的机密性。

5　实验评估

本节首先进行有效性验证实验, 以证明恶意的云运维人员(Cloud operator)在 HCoper 执行加密计算期间无法从内存快照文件中恢复出加密密钥和主密钥。然后, 本节通过比较 OpenSSL-1.1.0g 中的原始加密函数与 HCoper 实现的加密函数之间的性能差异来评估 HCoper 的实用性。本次实验环境部署在 Intel Core i7 4790 CPU(8 核)的工作站上, 其运行的 Hypervisor 为 Xen 4.4.0 版本。

5.1　安全功能验证

HCoper 的加密接口被集成到三个常见的加密库中: GnuPG 1.4.22[57]、OpenSSL-1.1.0g[54] 和 polarssl-2.6.0[58]。安全功能验证实验主要分为 3 个步骤: (1)首先, 在实验虚拟机内部执行这三个加密工具中的 AES 加密函数对测试文件进行加密(解密)操作, 在执行加密函数的过程中, 在特权虚拟机(Dom0)中执行“xl save”命令获取实验虚拟机的内存快照; (2)使用文献[27]中的 aeskeyfind 工具从内存快照中提取 AES 加密密钥; (3)在实验虚拟机内执行 HCoper 加密计算进程, 同时在此期间获取虚拟机快照, 再次使用 aeskeyfind 工具从内存快照中尝试提取 AES 加密密钥。

此外, 本节中还模拟了外部攻击者获取客户虚拟机内存快照并从内存快照中恢复出 AES 加密密钥。本次模拟外部攻击者实验中, 在虚拟机中运行加密计算进程时, 通过虚拟机内部的/dev/mem 设备接口来获取虚拟机的内存镜像, 再使用同样的工具尝试从内存快照中提取 AES 加密密钥。两个实验的结果均一致表明攻击者(内部或外部攻击者)无法从内存快照中获取 HCoper 使用的主密钥和加密密钥, 而运行常规加密库中的原始加密函数时能成功从内存快照中恢复出加密密钥, 实验结果如表 2 所示。另外,

表 2　密钥恢复结果对比

Table 2　Comparison of Recovering Keys

加密库	AES-128	AES-256	HCoper-128
GnuPG 1.4.22	Y	Y	N
OpenSSL-1.1.0g	Y	Y	N
polarssl-2.6.0	Y	Y	N

(注: Y 表示从内存快照中成功恢复出密钥, N 代表失败)

由于 HCoper 使用的密钥不会存在于内存中, 也不会出现在虚拟机镜像文件和普通的文件中, 进一步表明了 HCoper 不受冷启动攻击(Cold-boot)、VM 克隆攻击(VM Cloning)和 DaaS 文件复制攻击(DaaS File Copying)的威胁。

5.2 性能评估

5.2.1 加密运算性能评估

本节通过对比 HCoper 与 OpenSSL 加密运算的时间开销来评估 HCoper 性能与实用性。HCoper 的加密接口被集成到 OpenSSL 库中, 本文将其命名为 openssl-xen-aes-ni。具体地, 本节使用 Linux 平台中的 time[59]测量工具来统计执行 openssl-xen-aes-ni 函数以及 openssl-ecb-aes 函数时的时间开销。主要包括总时间开销(real time), 系统时间开销(sys time)和用户态时间开销(user time)。在实验过程中, 分别调用这两个函数加密不同大小的文件(从 100MB 到 1GB), 然后统计每次加密运算的时间开销。针对每个测试文件进行 10 次加密运算, 接着计算出 10 次加密运算的平均时间开销。

图 5 对比了 openssl-xen-aes-ni 函数和 openssl-ecb-aes 函数在加密不同大小的文件时总的时间开销(real time)。如图 5 所示, openssl-xen-aes-ni 的总时间开销甚至比原加密函数 openssl-ecb-aes 的要低。本次实验针对每个不同大小的测试文件总共执行 10 轮加密运算, 然后计算出平均吞吐量。例如, openssl-ecb-aes 用时 3.5199s 加密 300MB 文件, 因此吞吐量为 85.2MB/s(即 300MB/3.5199s)。最后经过计算得出 openssl-xen-aes-ni 加密运算的平均吞吐量(96.7MB/s)是 openssl-ecb-aes(89.6MB/s)的 1.08 倍。这可能是因为 HCoper 在执行加密运算操作时不会被系统任务调度或系统中断等事件中断。另外, HCoper 还充分利用 Intel AES-NI 指令集特性, 从硬件级别提高加密计算的性能。

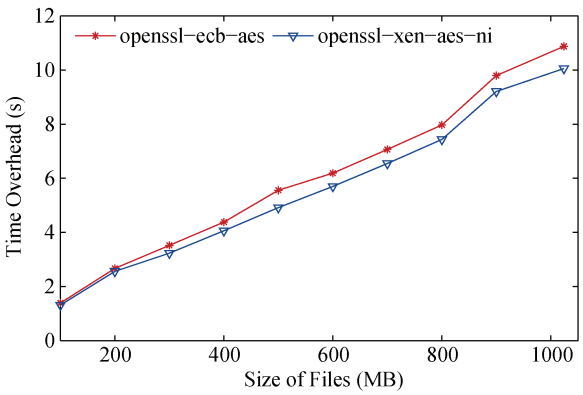


图 5 总时间开销

Figure 5 Real Time Overhead

系统时间开销(sys time)是指某个进程在内核态(内核模式)中执行的时间。如图 6 所示, openssl-xen-aes-ni 的内核态执行时间开销明显高于 openssl-ecb-aes。当加密同一个文件(例如 900MB)时, openssl-ecb-aes(2.1248s)在内核模式下的平均开销仅为 openssl-xen-aes-ni(6.9544s)的 30.6%(2.1248/ 6.9544)。这可以归因于 HCoper 大部分加密计算操作是在 Hypervisor 的内核中执行的。仅有少部分的操作是在用户模式下完成, 比如用户层应用程序简单地构造加密请求, 然后将加密请求提交给客户操作系统内核。此外, 从图 6 中还可以看到, 随着文件大小的持续增长, openssl-xen-aes-ni 的系统时间开销相比于 openssl-ecb-aes 增长速度要较快些。

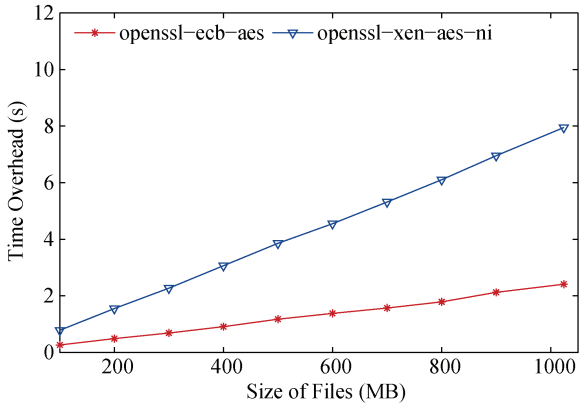


图 6 系统时间开销

Figure 6 Sys Time Overhead

相应地, 实验结果还表明 openssl-xen-aes-ni 的用户态时间开销(即在用户模式下的开销)比 openssl-ecb-aes 低很多, 并且 openssl-xen-aes-ni 的用户态时间开销几乎不随文件大小的增加而增长, 如图 7 所示。对于相同大小的文件(例如, 100MB), openssl-xen-aes-ni(0.042s)在用户模式下的平均开销

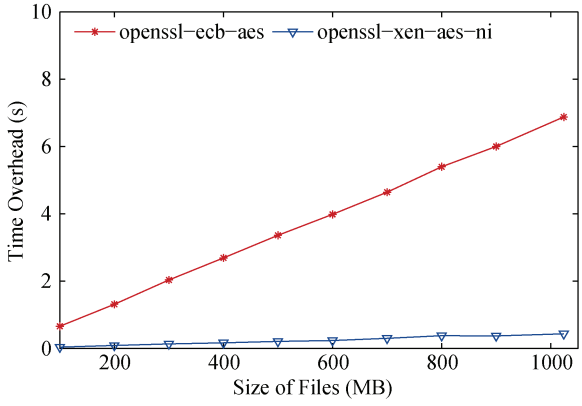


图 7 用户态时间开销

Figure 7 User Time Overhead

仅为 openssl-ecb-aes(0.6588s)的 6.4%(0.042/0.6588)。这与我们所期望的一致, 内核模式时间开销越高, 用户态时间开销越低, 两者的总和即是总时间开销(real time)。总之, 尽管 openssl xen-aes-ni(或 HCoper)的平均内核模式时间开销要高于 openssl-ecb-aes, 但 openssl-xen-aes-ni 的整体加密运算性能甚至比 openssl-ecb-aes 要快一点。换言之, HCoper 的加密运算效率与 OpenSSL 库中的加密实现方案一样具备实用性。

5.2.2 并发性能影响评估

由于 HCoper 在执行块加密计算期间, HCoper 会短暂地禁用 CPU 抢占, 因此被分配到同一 CPU 核上运行的其他任务的性能可能会受到影响。本次实验使用 SysBench[60]工具来衡量 HCoper 短时间独占 CPU 核对其他并发型应用程序造成的影响, SysBench 是一个通用的系统性能(如内存访问速度, CPU 运算性能)度量工具。本节实验部署时先将 HCoper 集成到 Apache Web 服务器中, 具体地, 在 PHP 文件中调用 HCoper 执行加密运算。在客户机端, 本次实验启动了多个客户线程来并发地提交加密 8KB 数据的请求。在服务器端(HCoper 和 SysBench 都运行在服务器端, 即同一台机器上), 使用 SysBench 启动 4 个线程向本地 CPU 核发出 10K 个请求, 每个请求都涉及高达 40K 的素数计算, 即求出 40K 个素数。我们收集了当客户端线程以不同的速率(requests/second)发出加密请求时(即此时 HCoper 需要响应客户的加密请求), 记录 SysBench(在服务器端运行)所发起的每个素数计算任务的平均运行时间。

如图 8 所示, 基准值(baseline)是在没有执行任何加密进程的干净环境中测量的。对于相同的加密速率(requests/sec), 相对于 openssl-xen-aes-ni 而言, 原来的 openssl-ecb-aes 对 Sysbench 启动的素数计算进程性能的影响较低一些, 但实验结果表明, HCoper 对其他并发型应用程序性能造成的影响是可以接受的。例如, 当加密速率为 400(requsets/sec)时, openssl-xen-aes-ni 导致执行素数计算任务的并发进程的性能下降了 1.8%, 而 openss-ecb-aes 导致素数计算性能下降了 1.3%。相比之下, openssl-xen-aes-ni 带来的并发性能影响和 openssl-ecb-aes 造成的并发性能影响还处于同一个数量级(仅高了 0.5%)。这主要是因为当 HCoper 执行块加密运算时, 它将暂时地禁止 CPU 抢占, 使得 CPU 密集型的运算任务的性能会受到影响。然而实验结果表明, 与 openssl-ecb-aes 相比, HCoper 引起附加开销是可以接受的, 即不影响实用性。

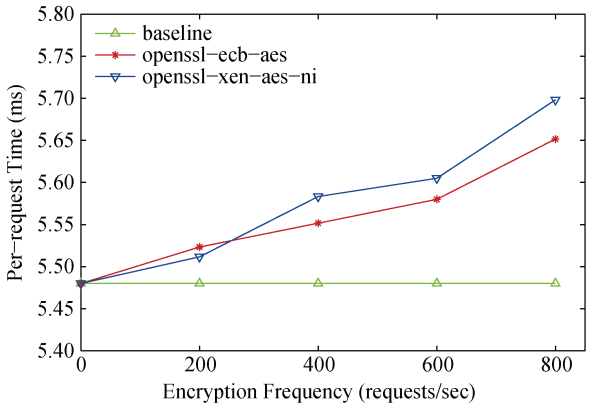


图 8 对并发型程序的性能影响

Figure 8 Impact on Concurrent Processes

6 潜在安全威胁分析

如第 5 节所示, 实验结果已验证了 HCoper 的功能正确并且在执行加密运算时不会将加密密钥泄漏到 RAM 中。在本节中, 本节将分析 HCoper 针对 Hypervisor 级攻击的防御能力。然后讨论 HCoper 可能会面临的来自硬件层面的攻击方式以及相关防御措施。

6.1 Hypervisor 层攻击

HCoper 的安全性基于 Hypervisor 安全可靠的假设之上。然而在真实的云环境中, Hypervisor 不可能没有任何安全缺陷。不可避免地, 攻击者可能会利用 Hypervisor 的漏洞窃取密钥。例如, 在虚拟机逃逸攻击[61]中, 攻击者可以使用虚拟机逃逸漏洞向 Hypervisor 的内核中注入恶意代码, 然后设计精妙的攻击手段(如 ROP)来执行恶意代码。由于恶意代码与 Hypervisor 内核的执行权限级别相同, 因此攻击者可以轻松读取或写入寄存器(即 Debug 或 MSR)以获取密钥。类似的, 利用 XSA-148[62]漏洞利用也能实现任意读取和写入 Xen 的物理内存。此外, 恶意的半虚拟化客户虚拟机中可以通过此漏洞篡改(Hook)Xen 内核中的超级调用表(hypercall table), 进而实现在 Hypervisor 内核级别执行任意代码[63]。本文使用最新的安全补丁来提高 HCoper 对抗 Hypervisor 内核级别攻击的能力。如何增 Hypervisor 内核的安全性以及发现 Hypervisor 的漏洞超出了本文的研究范围。

6.2 硬件层攻击

云服务提供商内部的恶意运维人员(Cloud operator)可能会有机会物理接触到部署云计算平台的物理机器。考虑一个简单的场景: 恶意运维人员有可能使用恶意启动设备(例如, 外部 USB 启动设备)重新启动计算机, 以读取 CPU 内部寄存器(例如, DR 和

MSR)中残存的数据内容。幸运的是, 这种攻击是无效的, 因为在重新启动物理机器时, CPU 内部的所有寄存器都将被重置为零[35]。

基于 CPU 缓存(Cache)的侧信道(side-channel)攻击[64-65]对很多加密系统而言的是一大安全威胁。但是这种攻击对 HCoper 中是无效的, 因为 Intel AES-NI 提供的硬件加密指令集对 timing-attacks[24]免疫。即 Intel AES-NI 指令执行加密运算时没有表现出规律性的时间差异, 攻击者不能根据执行时间差异性逆推出加密运算执行的细节。另外, HCoper 的执行控制流中没有输入依赖性(input dependent)的执行分支(branch), 使得攻击者不能使用侧信道攻击的手段(通过监控、窥探目标代码内跳转分支的执行情况, 逆推出执行分支代码时的输入值)来推测加密密钥以及加密计算得中间状态值等。因此 HCoper 能防御侧信道攻击。

攻击者还可能会发起基于 DMA 的高级攻击[43], 将恶意代码注入 Hypervisor 内核, 然后访问调 Debug 存器和 MSR 寄存器中的密钥。幸运的是, 我们可以通过配置 IOMMU[45]或使用 BARM[44]来监视系统总线活动以检测、揭露从外围设备(peripherals)访问内存的恶意行为, 从而抵御这种攻击。最后, 通过物理访问(接触)云平台中的真实机器, 恶意的内部人员(例如高级电气工程师)可能会通过使用示波器等专业设备测量 CPU 周围的电磁信号来变向读取正在运行的 CPU 的寄存器。但是这种攻击实施条件异常严苛并且实施过程高度复杂, 据本文所知, 目前暂时还没公开过这种高难度攻击手段成功的案例。如何防御此种攻击已不在 HCoper 研究的范围。

7 结论

本文提出了一种在 CPU 内执行加密运算(CPU-bound)的解决方案 HCoper, 目的在于保护云租户的密钥免受恶意运维人员发起的内存快照攻击。HCoper 适用于真实云计算平台。同时, 所有的密码计算都在 CPU 寄存器内完成, 因此 HCoper 也不受冷启动攻击(Cold-boot)、VM 克隆攻击(VM Cloning)和 DaaS 文件复制攻击(DaaS File Copying)的威胁。通过执行动态密钥调度, HCoper 支持多个应用程序使用多个密钥。此外, HCoper 已经集成到其他第三方密码库中(如 OpenSSL-1.1.0g、polarssl-2.6.0、GnuPG 1.4.22), 使得开发新应用程序变得容易。HCoper 原型系统作为 Xen4.4.0 的核心模块实现, 体现其具备实际部署价值。安全功能验证实验表明, 恶意运维人员无法从客户虚拟机的内存快照文件中提取数据加密密钥或主密钥。此外, 性能评估表明, HCoper 的效率与传统的加密计算实现方案相当, 对其他并发型应用程序性能的影响在可接受范围(即 HCoper 带来的性能损失不足以影响其实用性)。

目前 HCoper 只针对 Intel CPU 实现了对称加密算法(AES)。理论上, HCoper 的架构思想可以部署在其他类型处理器上, 只要这些处理器也提供指令集支持实现加密算法(如 AES、RSA)的加密操作。将来, HCoper 可以扩展到非对称加密计算(如 RSA)。然而, 完全在 CPU 内实现非对称加密计算极具挑战性, 因为 RSA 计算需要更大的存储空间, 而寄存器(如 SSE)暂时不支持 RSA 所需要的存储空间。HCoper 方案比较适用于加密计算的中间变量或参数占用空间较小的算法(如 AES 算法等)。而其他算法(如 RSA)计算过程中使用到的参数占用空间较大远超过 SSE 寄存器的空间, 针对此类算法, 可将一部分加密过程交给 HCoper 来完成或者使用 HCoper 来加密此类算法运行过程中重要参数达到加密密钥保护的目标, 这将是 HCoper 未来的优化工作。另外, HCoper 基于这样一个假设: Hypervisor 是安全的, 没有任何安全缺陷。另一个未来的工作可在硬件支持的可行执行环境(如 Intel SGX 提供的 Enclave)中实现加密计算, SGX 对不受信任或遭受攻击的 Hypervisor 发动的攻击免疫。然而, Intel SGX 只为 Enclave 保留 128M 物理内存, 如何使用有限的 RAM 加密大文件是一个挑战。

参考文献

[1] 2018 trends in cloud computing, CompTIA, https://www. comptia.org/resources/cloud-computing-trends-research#section2, May. 2018.

[2] Global Cloud Computing Market Forecast 2019-2024, https://www.marketresearchmedia.com/?p=839, Jan. 2018.

[3] Verma A, Kaushal S. Cloud Computing Security Issues and Challenges: A Survey[M]. Advances in Computing and Communications. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011: 445-454.

[4] Rohit Bhadauria, Rituparna Chaki, Nabendu Chaki, et al. A survey on security issues in cloud computing[J]. *International Journal of Engineering & Technology* (0975-4024), 5(2):1912–1920, 2011.

[5] Sengupta S, Kaulgud V, Sharma V S. Cloud Computing Security——Trends and Research Directions[C]. *2011 IEEE World Congress on Services*, 2011: 524-531.

[6] Cloud Security Alliance. Treacherous 12 top threats to cloud computing plus: Industry insights, CSA, https://downloads. cloudsecurityalliance.org/assets/research/top-threats/treacherous-12-top-threats.pdf, 2017.

[7] Former dupont employee sentenced for stealing trade secrets, https://raysemko.com/2015/08/17/former-dupont-employee-sentenced-for-stealing-trade-secrets/, Aug. 2015.

[8] 2016 state of cybercrime survey, https://resources.sei.cmu.edu/ library/asset-view.cfm?assetID=499782, Carnegie Mellon University, May. 2016.

[9] Duncan A J, Creese S, Goldsmith M. Insider Attacks in Cloud Computing[C]. *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012: 857-862.

[10] Rocha F, Correia M. Lucy in the Sky without Diamonds: Stealing Confidential Data in the Cloud[C]. *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops*, 2011: 129-134.

[11] Spitzner L. Honeypots: Catching the Insider Threat[C]. *19th Annual Computer Security Applications Conference*, 2003: 170-179.

[12] Eberle W, Graves J, Holder L. Insider Threat Detection Using a Graph-Based Approach[J]. *Journal of Applied Security Research*, 2010, 6(1): 32-81.

[13] Bradford, P., and Ning Hu. A layered approach to insider threat detection and proactive forensics[C]. *The Twenty-First Annual Computer Security Applications Conference*. 2005.

[14] Nguyen N, Reiher P, Kuenning G H. Detecting Insider Threats by Monitoring System Call Activity[C]. *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop*, 2003: 45-52.

[15] Nuno Santos, Krishna P Gummadi, and Rodrigo Rodrigues. Towards trusted cloud computing[C]. *In Conference on Hot Topics in Cloud Computing*, 2009:3.

[16] Chen X X, Garfinkel T, Lewis E C, et al. Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems[C]. *The 13th international conference on Architectural support for programming languages and operating systems - ASPLOS XIII*, 2008: 2-13.

[17] Intel software guard extensions programming reference, Intel, https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf, Oct. 2014.

[18] Orenbach M, Lifshits P, Minkin M, et al. Eleos: ExitLess OS Services for SGX Enclaves[C]. *The Twelfth European Conference on Computer Systems*, 2017: 238– 253.

[19] Xu Y Z, Cui W D, Peinado M. Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems[C]. *2015 IEEE Symposium on Security and Privacy,* 2015: 640-656.

[20] Michael Schwarz, Samuel Weiser, Daniel Gruss, et al. Malware guard extension: Using sgx to conceal cache attacks[C]. *International conference on detection of intrusions and malware, and vulnerability assessment*, 2017:3–24.

[21] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, et al. Software Grand Exposure: SGX Cache Attacks Are Practical[C]. *In* 11th *USENIX Workshop on Offensive Technologies*, 2017

[22] kvm-sgx, Intel, https://github.com/intel/kvm-sgx, Mar. 2018.

[23] xen-sgx, Intel, https://github.com/intel/xen-sgx, Nov. 2017.

[24] Jeffrey Rott (Intel). Intel advanced encryption standard instructions(aes-ni), https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni, Feb. 2012.

[25] Barham P, Dragovic B, Fraser K, et al. Xen and the Art of Virtualization[C]. *The nineteenth ACM symposium on Operating systems principles - SOSP '03*, 2003: 164–177.

[26] KVM, Main page, KVM contributors, https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792, Nov. 2016.

[27] Halderman J A, Schoen S D, Heninger N, et al. Lest we Remember[J]. *Communications of the ACM*, 2009, 52(5): 91-98.

[28] Claycomb W R, Nicoll A. Insider Threats to Cloud Computing: Directions for New Research Challenges[C]. *2012 IEEE 36th Annual Computer Software and Applications Conference*, 2012: 387-394.

[29] Kandias M, Mylonas A, Virvilis N, et al. An Insider Threat Prediction Model[M]. Trust, Privacy and Security in Digital Business. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010: 26-37.

[30] Magklaras G B, Furnell S M. Insider Threat Prediction Tool: Evaluating the Probability of IT Misuse[J]. *Computers & Security*, 2001, 21(1): 62-73.

[31] Bondada M B, Bhanu S M S. Analyzing User Behavior Using Keystroke Dynamics to Protect Cloud from Malicious Insiders[C]. *2014 IEEE International Conference on Cloud Computing in Emerging Markets*, 2014: 1-8.

[32] KumarMandal K, Chatterjee D. Insider Threat Mitigation in Cloud Computing[J]. *International Journal of Computer Applications*, 2015, 120(20): 7-11.

[33] Stolfo S J, Salem M B, Keromytis A D. Fog Computing: Mitigating Insider Data Theft Attacks in the Cloud[C]. *2012 IEEE Symposium on Security and Privacy Workshops*, 2012: 125-128.

[34] Li M, Zang W Y, Bai K, et al. MyCloud: Supporting User-Configured Privacy Protection in Cloud Computing[C]. *The 29th Annual Computer Security Applications Conference*, 2013: 59–68.

[35] Felix C. Freiling and Andreas Dewald. Tresor runs encryption securely outside ram[C]. *In Usenix Conference on Security*, 2011:17.

[36] Müller T, Dewald A, Freiling F C. AESSE: A Cold-Boot Resistant Implementation of AES[C]. *The Third European Workshop on System Security - EUROSEC '10*, 2010: 42-47.

[37] Simmons P. Security through Amnesia: A Software-Based Solution to the Cold Boot Attack on Disk Encryption[C]. *The 27th Annual Computer Security Applications Conference on - ACSAC '11*, 2011: 73-82.

[38] Guan L, Lin J Q, Ma Z Q, et al. Copker: A Cryptographic Engine Against Cold-Boot Attacks[J]. *IEEE Transactions on Dependable and Secure Computing*, 2018, 15(5): 742-754.

[39] Garmany B, Müller T. PRIME: Private RSA Infrastructure for Memory-less Encryption[C]. *The 29th Annual Computer Security Applications Conference*, 2013: 149-158.

[40] Introduction to Intel® Advanced Vector Extensions, Intel, https://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions, Jun. 2011.

[41] Le Guan, Jingqiang Lin, Bo Luo, et al. Protecting private keys against memory disclosure attacks using hardware transactional memory[C]. *IEEE Symposium on Security and Privacy*, 3–19, 2015.

[42] Chapter 8: Intel architecture instruction set extensions programming reference, Intel, https://www.naic.edu/~phil/software/intel/319433-014.pdf, Aug. 2012.

[43] Blass E O, Robertson W. TRESOR-HUNT: Attacking CPU-Bound Encryption[C]. *The 28th Annual Computer Security Applications Conference on - ACSAC '12*, 2012: 71–78.

[44] Patrick Stewin. A primitive for revealing stealthy peripheral-based attacks on the computing platforms main memory[C]. In *International Workshop on Recent Advances in Intrusion Detection*, 2013:1–20.

[45] Stewin P, Bystrov I. Understanding DMA Malware[M]. Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013: 21-41.

[46] Zhang F W. IOCheck: A Framework to Enhance the Security of I/O Devices at Runtime[C]. *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop*, 2013: 1-4.

[47] Vasiliadis G, Athanasopoulos E, Polychronakis M, et al. PixelVault: Using GPUs for Securing Cryptographic Operations[C]. *The 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014: 1131-1142.

[48] Zhang N, Sun K, Lou W J, et al. CaSE: Cache-Assisted Secure Execution on ARM Processors[C]. *2016 IEEE Symposium on Security and Privacy*, 2016: 72-90.

[49] ARM Security Technology Building a Secure System using TrustZone Technology, ARM, http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, 2009.

[50] How cloud management works, servicenow, https://docs. servicenow.com/bundle/istanbul-it-operations-management/page/product/cloud-provisioning/reference/r_HowCloudProvisioningWorks.html, Ari. 2017.

[51] Lipp M, Schwarz M, Gruss D, et al. Meltdown: Reading kernel memory from user space [C]. *27th USENIX Security Symposium*. 2018: 973-990.

[52] Hardware performance counter, https://en.wikipedia.org/wiki/Hardware_performance_counter, 2017.

[53] X86 Assembly/SSE, WikiBooks, https://en.wikibooks.org/wiki/X86_Assembly/SSE, Apr. 2019.

[54] OpenSSL, Inc. OpenSSL Foundation, https://www.openssl.org/, May. 2019.

[55] Polarssl, ARM Limited, https://tls.mbed.org/, May. 2019.

[56] Inc. Intel. Intel 64 and ia-32 architectures software developer's manual combined volumes: 1, 2a, 2b, 3a and 3b. 2011.

[57] The GNU Privacy Guard, The GnuPG Project, https://gnupg.org/, Mar. 2019.

[58] Polarssl-2.6.0, ARM Limited, https://tls.mbed.org/download/start/mbedtls-2.6.0-gpl.tgz. May. 2019.

[59] time(1) - linux man page, https://linux.die.net/man/1/time, May. 2019.

[60] sysbench, https://launchpad.net/sysbench, May. 2019.

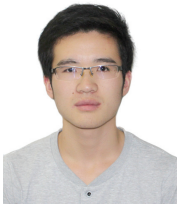
[61] Virtual-machine-scape, https://en.wikipedia.org/wiki/Virtual-machine-scape, May. 2019.

[62] CVE-2015-7835, https://xenbits.xen.org/xsa/advisory-148.html, Oct. 2015.

[63] Exploit Two Xen Hypervisor Vulnerabilities, Cloud Platform Security Team of Alibaba Cloud, https://www.blackhat.com/docs/us-16/materials/us-16-Luan-Ouroboros-Tearing-Xen-Hypervisor-With-The-Snake-wp.pdf, 2016.

[64] Osvik D A, Shamir A, Tromer E. Cache attacks and countermeasures: The case of aes [C]. *Topics in Cryptology-CT-RSA 2006*, 2006: 1-20.

[65] Liu F F, Yarom Y, Ge Q, et al. Last-Level Cache Side-Channel Attacks are Practical[C]. *2015 IEEE Symposium on Security and Privacy*, 2015: 605-622.



何运 于 2016 年在北京工业大学信息安全专业获得学士学位。现在中国科学院大学网络空间安全专业攻读博士学位。研究领域为系统安全。研究兴趣包括: SGX、虚拟化。Email: heyun@iie.ac.cn

贾晓启 于 2010 年在中国科学院研究生院信息安全专业获得博士学位。现任中国科学院信息工程研究所研究员。研究领域为系统安全。Email: jiaxiaoqi@iie.ac.cn

刘鹏 于 1999 年在乔治梅森大学获得博士学位。现任美国宾夕法尼亚州立大学信息科学与技术系教授、网络安全、信息隐私与信任中心主任、网络安全实验室主任。研究领域为系统安全。Email: pliu@ist.psu.edu

张伟娟 于2018年在中国科学院信息工程研究所信息安全专业获得博士学位, 现任信息工程研究所助理研究员。研究领域为系统安全。Email: zhangweijuan@iie.ac.cn