软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(10):3120−3146 [doi: 10.13328/j.cnki.jos.006067]

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

抗电路板级物理攻击的操作系统防御技术研究*

张倩颖[1,4,5], 赵世军[2,3]

[1](首都师范大学 信息工程学院,北京 100048)

[2](中国科学院 软件研究所,北京 100190)

[3](华为技术有限公司,北京 100195)

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

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

通讯作者: 赵世军, E-mail: zqyzsj@gmail.com

摘 要: 计算设备处理和存储日益增多的敏感信息,如口令和指纹信息等,对安全性提出更高要求.物理攻击技术的发展催生了一种通过攻击电路板级硬件组件来获取操作系统机密信息的攻击方法:电路板级物理攻击.该类攻击具有工具简单、成本低、易流程化等特点,极容易被攻击者利用形成黑色产业,是操作系统面临的新安全威胁和挑战.在处理器上扩展内存加密引擎可抵抗该类攻击,但是目前大部分计算设备并未配备该硬件安全机制.学术界和产业界提出软件方式抗电路板级物理攻击的操作系统防御技术,该类技术已成为近年来的研究热点.深入分析了该类技术的研究进展,总结其技术优势和不足,并探讨其发展趋势.首先,介绍了电路板级物理攻击的定义、威胁模型、现实攻击实例.之后,介绍软件方式抗电路板级物理攻击的操作系统防御技术所依赖的一些基础技术.然后,对该类防御技术的研究进展按照保护范围进行分类总结和归纳.最后,分析了该类防御技术的优势与不足,给出工程实现建议,并探讨该类防御技术未来的研究趋势.

关键词: 内存保护;物理攻击;内存加密;内存完整性

中图法分类号: TP311

中文引用格式: 张倩颖,赵世军.抗电路板级物理攻击的操作系统防御技术研究.软件学报,2020,31(10):3120−3146. http://www.jos.org.cn/1000-9825/6067.htm

英文引用格式: Zhang QY, Zhao SJ. Survey of research on protection mechanisms of operating system against board level physical attacks. Ruan Jian Xue Bao/Journal of Software, 2020,31(10):3120−3146 (in Chinese). http://www.jos.org.cn/1000-9825/6067.htm

Survey of Research on Protection Mechanisms of Operating System against Board Level Physical Attacks

ZHANG Qian-Ying[1,4,5], ZHAO Shi-Jun[2,3]

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

[2](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[3](Huawei Technologies Co., Ltd., Beijing 100195, China)

* 基金项目: 国家自然科学基金(61802375, 61602325, 61876111, 61877040); 北京市教委科技计划一般项目(KM20190028005); 中国科学院计算技术研究所计算机体系结构国家重点实验室开放课题(CARCH201920)

Foundation item: National Natural Science Foundation of China (61802375, 61602325, 61876111, 61877040); Project of Beijing Municipal Education Commission (KM20190028005); Open Research Fund of State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences) (CARCH201920)

本文由“系统软件前沿进展”专题特约编辑武延军研究员、陈海波教授、包云岗研究员、李玲研究员推荐.

收稿时间: 2020-02-10; 修改时间: 2020-04-04; 采用时间: 2020-05-09; jos 在线出版时间: 2020-06-10

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

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

Abstract: Computing devices are processing and storing more and more sensitive information, such as passwords and personal fingerprints, so higher security requirements are required for them. With the development of physical attacks, a new kind of attack called board level physical attacks is developed, and this kind of attack can obtain secrets in the operating system by attacking hardware components at the printed circuit board (PCB) level. This newly proposed attack only uses simple tools, its cost is inexpensive, and it can be streamlined simply, so it can be leveraged by attackers to form new underground industry easily. Therefore it is a new security threat and challenge for operating systems. A common defense against this kind of attack is to extend a specialized memory encryption engine to the CPU, but most current computing devices are not equipped with such hardware security mechanisms. Thus, the academic fields and industrial fields propose software-based techniques to defend board level physical attacks, and these techniques have been becoming a research hotspot in recent years. This paper deeply analyzes the development of these techniques, summarizes their advantages and disadvantages, and discusses their development trends. First, the paper introduces the definition, threat model and some real-world attack cases of the board level physical attacks. Second, the paper describes the building blocks relied by the software-based techniques to defense the board level physical attacks. Third, the paper makes a survey of and categorizes the related work on the software-based defense technology according to their protection domains. At last, the paper analyzes the advantages and disadvantages of the technology, gives suggestions on how to implement it in practice, and discusses some development trends of this technology.

Key words: memory protcetion; physical attack; memory encryption; memory integrity

随着信息技术的发展,移动互联网、云计算、工业 4.0 等应用场景不断出现,以实现消费者、企业以及工业等领域的信息化目标.与此同时,越来越多的安全敏感业务也迁移到智能终端和服务器平台等各种计算设备上:智能终端等个人终端设备提供身份认证、电子钱包、企业办公等安全敏感功能;企业数据中心和智能工厂等服务器平台存储并处理企业数据以及工业数据,这些数据涉及企业的核心机密以及工厂的隐私信息,一旦泄露后果极为严重.

计算设备中信息价值的提高吸引了越来越多的攻击者甚至黑色产业者的注意,并开始设计各种先进的攻击方法.除了传统的软件攻击,最新出现了一种低成本的物理攻击:电路板级物理攻击(以下简称为板级物理攻击).这类攻击典型的案例包括冷启动攻击[1−5]、总线窃听攻击[6−10]和 DMA 攻击[11−14].板级物理攻击与传统的物理攻击相比(譬如侵入式攻击[15]和半侵入式攻击[16]),所需攻击工具成本低,攻击流程简单,具有成本低、易复制、可流程化等特点,容易被攻击者利用形成窃取个人机密信息的黑色产业,已经成为目前计算机系统严重的安全威胁之一.

冷启动攻击等板级物理攻击出现之后,处理器体系结构领域的研究人员提出通过保护片外 RAM 来抵抗该攻击的方法,其主要思路是在处理器芯片与片外 RAM 之间的传输路径上部署硬件加密和完整性保护引擎[17−39]来保护片外 RAM 的机密性和完整性.这种基于硬件的方法将片外 RAM 的安全性规约为处理器芯片的安全性,从而彻底杜绝只能实施在处理器芯片外部的硬件组件的板级物理攻击.加拿大滑铁卢大学的 Elbaz 等人和美国达特茅斯学院的 Henson 等人分别就当前学术界硬件方式的内存完整性保护方案[40]和内存加密方案[41]进行了调研,总结了各种方案的优缺点以及所能达到的安全保护能力.学术界的先进方法也促进了产业界的技术发展,以 XOM[17]和 AEGIS[18]为代表的学术界技术思路被产业界所采纳,形成了 Intel SGX(software guard extensions)[42,43]和 AMD SME(secure memory encryption)[44]等产业界内存保护方案.

但是实际上,只有部分 Intel 和 AMD 的服务器处理器配备了 SGX 和 SME 等内存保护机制,智能终端、绝大多数个人电脑以及部分服务器都不具备抵抗板级物理攻击的能力.而板级物理攻击是这些设备所部署的应用场景的一个严重威胁,因此这些设备同样需要具备抗板级物理攻击的安全能力.针对该问题,学术界提出了基于软件方式的抗板级物理攻击的操作系统防御技术,这类技术的思路是在操作系统等系统软件上扩展安全增强机制,将这些安全机制运行在设备上通用的安全存储资源中并提供对片外 RAM 的安全保护.中科院信工所的林璟锵等人调研了利用缓存和寄存器这两类通用安全存储资源构建的内存保护方案[45],分析了各种方案的优点和局限性以指导硬件辅助安全性方面的研究.软件方式的防护技术无需修改处理器硬件就能为安全关键程序、通用应用程序乃至整个系统提供较强的物理防护能力,对已经出厂但没有配备硬件内存保护引擎的设备

来说具有重要的实际意义,因此该方法自板级物理攻击出现以后就一直受到系统安全领域的重视,是近 10 年来系统安全的一个热点研究课题.

本文首先介绍板级物理攻击的基本原理、威胁模型和实际案例,然后描述基于软件方式的抗板级物理攻击方法的相关技术,之后按照保护范围分析现有主流方案,对各种方案的优缺点、安全能力、技术局限等进行全方面的分析总结,其后讨论该类技术的优势与不足,并为相关技术人员部署该类技术提供参考建议,最后对该研究方向的发展趋势进行分析展望.

1 物理攻击概述

根据攻击层次的不同,物理攻击分为芯片级物理攻击和电路板级物理攻击[46]两类.芯片级物理攻击的目标是位于芯片内部的密钥等机密信息,实施该攻击需要电子显微镜工作台、聚焦离子束工作台和激光切割系统等企业级实验环境,攻击成本非常高.同时该类攻击对攻击人员的专业知识的广度和深度都有很高的要求,一般要求攻击人员掌握高级芯片架构、探针技术、聚焦离子束操作等多个领域的专业知识.此外,每次攻击只能针对一块芯片,无法一次性对批量的芯片造成威胁,因此对于价值一般的设备来说攻击意义不大.

板级物理攻击的主要目标是设备片外 RAM 中的数据和代码.攻击者借助示波器、逻辑分析仪、探针等简易工具就能通过电路板上片外 RAM 与外部的连接通道(譬如 CPU 芯片与片外 RAM 之间的总线、片外 RAM 的接口、外设 DMA 接口等等)实施对内存数据的窃听、篡改和重放攻击.板级物理攻击的低成本、易流程化等特点导致其极容易被攻击者利用形成窃取个人和企业机密信息的黑色产业,是当前智能终端、个人电脑和服务器等设备面临的一个严重安全威胁.国际标准组织 GlobalPlatform 在智能终端可信执行环境保护轮廓规范[47]中明确声明板级物理攻击是智能终端所面临的除软件攻击之外的最大安全威胁.

1.1 板级物理攻击威胁模型

在板级物理攻击敌手能力下,只有 CPU 芯片能够抵抗板级物理攻击,片外 RAM 和系统总线等其他组件都能够被攻击者控制.因此在该模型下,只有 CPU 芯片属于可信区域,是整个系统的可信计算基(trusted computing base,简称 TCB),其他硬件组件都允许被敌手控制.敌手能够主动控制片外 RAM 以及系统总线传输的内容.在该威胁模型下,敌手能够任意控制 CPU 从片外 RAM 读入的内容,譬如篡改和窃听 CPU 读入的代码和数据、篡改片外 RAM 的内容等等.因此,所有从片外 RAM 读入的数据都不可信,必须经过机密性和完整性保护.

学术界将攻击者划分为被动敌手和主动敌手两类.被动敌手不能对片外 RAM 进行篡改,其主要通过总线监听或者直接读取片外 RAM 的内容实施攻击,因此该类敌手的主要能力是读取片外 RAM 的内容.主动敌手可以通过总线注入等手段篡改片外 RAM 的内容,根据板级物理攻击者对系统总线和片外 RAM 的操控能力,学术界设计了通用的板级物理攻击威胁模型(如图 1 所示)[32,40].在该模型中,敌手控制芯片外的总线并在总线上挂载一个恶意片外 RAM,敌手可以自由控制该恶意片外 RAM 的内容.该片外 RAM 地址总线与正常的片外 RAM 重用,敌手可以将数据总线在正常片外 RAM 和恶意片外 RAM 之间进行切换,从而控制 CPU 获取的数据来自正常片外 RAM 或者恶意片外 RAM.在该模型基础上,所有的板级物理攻击被抽象为 3 种类型的攻击:欺骗攻击、替换攻击和重放攻击.

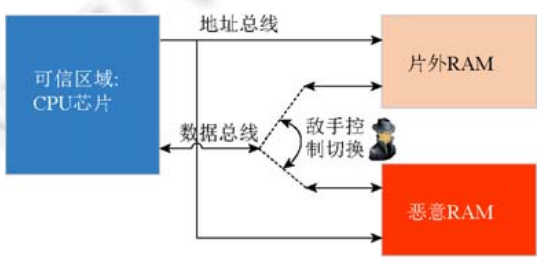

Fig.1 Threat model for the board level physical attacks

图 1 板级物理攻击威胁模型

• 欺骗攻击:敌手使用任意的内容修改某地址处内存块的内容.敌手将自己想让 CPU 获取的内容存储到恶意片外 RAM 中.每当 CPU 从该地址读取内容时,敌手将数据总线切换到恶意片外 RAM,这样 CPU 读取的就是敌手设置的内容.

• 替换攻击:敌手将地址 *A* 处的内容替换为地址 *B* 处的内容.敌手将正常片外 RAM 地址 *B* 处的内容存放在恶意片外 RAM 地址 *A* 处.每当 CPU 读取地址 *A* 处的内容时,敌手将数据总线切换到恶意片外 RAM,这样 CPU 读取的就是正常片外 RAM 地址 *B* 处的内容.

• 重放攻击:敌手将地址 *A* 处的内容替换为地址 *A* 以前某一时刻的内容.敌手在某一时刻将正常片外 RAM 地址 *A* 处的内容记录在恶意片外 RAM 地址 *A* 处.在之后的某一时刻 CPU 读取地址 *A* 处的内容时,敌手将数据总线切换到恶意片外 RAM,这样 CPU 实际读取的是地址 *A* 以前的内容.

1.2 实际板级物理攻击

现实世界中已经出现了众多板级物理攻击实例,攻破了 DRAM 接口、系统总线和 DMA 外设接口等硬件组件(如图 2 所示).

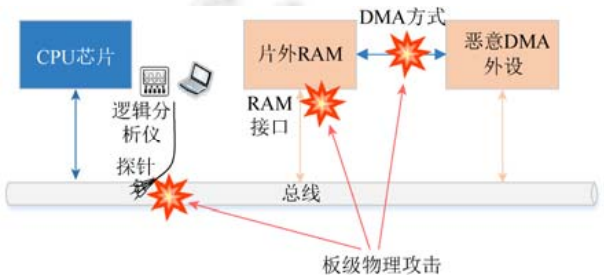

Fig.2 Illustration of real-world board level physical attacks

图 2 实际板级物理攻击示意图

具体攻击方法主要包含冷启动攻击、总线攻击以及 DMA 攻击.冷启动攻击[1−5]利用 DRAM 的断电存留特点[48](DRAM 内存在断电 1~2s 内,大部分数据仍残留在内存上,并且通过给 DRAM 降温可以延长数据存留时间),通过物理断电重启设备并重刷固件的方式在设备上加载恶意攻击程序,利用该恶意程序获取 DRAM 内存的镜像,进一步分析该镜像即可获得内存中的密钥等机密信息.普林斯顿大学的研究团队[1]实际测量了低温下 DRAM 随温度变化数据的损失率,测量结果发现在–50℃温度下,掉电 DRAM 每隔 10 分钟仅损失 1%的数据;在–196℃温度下,掉电 DRAM 每隔 60 分钟仅损失 0.17%的数据.该团队成功实施了几种类型的攻击:低温下直接重启或断电重启后加载恶意内核来读取 DRAM 镜像,或直接将 DRAM 内存条移植到一个攻击计算机上来读取镜像,最终从 DRAM 镜像中获得了全盘加密系统的 AES 密钥以及 RSA 密钥等机密数据.德国埃尔朗根-纽伦堡大学的研究团队[2]利用冷启动攻击成功攻破了谷歌 Nexus、三星 Galaxy 等系列品牌手机,能够从 DRAM 镜像中直接获取磁盘加密密钥、通讯录、网页历史记录、照片等信息.该团队开发并公开了他们的攻击工具 FROST,同时公开的还包括攻击步骤和攻击源代码,因此黑产从业者可直接利用该工具实施攻击.智能终端通常使用 PoP(package on package)技术将 CPU 和内存封装在一个芯片内,因此无法直接将 DRAM 组件移植到其他攻击设备上,只能通过重启加载攻击程序来读取 DRAM 镜像.为了减少攻击程序对 DRAM 镜像内容的覆盖,攻击程序所占用的内存越少越好.德国帕绍大学的研究团队[3]设计了一个轻量级的裸机攻击程序,仅占用 5KB 内存,大大减少了对 DRAM 镜像的覆盖.德国埃尔朗根-纽伦堡大学的研究人员基于 12 种计算机系统以及不同的内存配置对冷启动攻击进行了系统性的研究[4],证明冷启动攻击在 DDR1 和 DDR2 类型的内存上具有很高的可重现性,并证明每降低 10℃就能够极大增强 DRAM 芯片的断电存留特性.另外,该团队通过针对 5 种计算机系统以及多种内存配置的攻击实验发现:热启动方式的攻击对 DDR3 类型的内存能够成功,但是冷启动方式的攻击则对 DDR3 芯片无效.为了抵抗冷启动攻击等内存镜像取证工具对内存的获取,内存芯片厂商在 DDR3 内存

芯片中增加了内存加扰(memory scrambling)技术,该技术的有效性已经被相关研究人员的实验所证实[5].德国埃尔朗根-纽伦堡大学的研究人员通过实验分析证明了内存加扰机制一般实现为线性反馈移位寄存器(linear-feedback shift registers,简称 LFSR),然后设计了一种已知明文攻击攻破了内存加扰技术.该攻击最多需要 128B 的内存明文,并且在双通道内存系统中利用 LFSR 密钥流的数学关系可将攻击提升到只需要 50B 内存明文.操作系统大量内存数据被初始化为 0,因此这种攻击是实际可行的,该团队利用这种方法成功攻击了 4 种 Intel 计算机设备.

总线攻击[6−10]通过在系统总线上挂载探针,并借助逻辑分析仪等数字信号解析工具对 CPU 与片外 RAM 之间的传输数据进行监听和注入,从而达到窃取密钥等机密信息以及篡改运行代码的攻击目标.MIT 研究人员通过在微软 Xbox 电子设备的南北桥总线上挂载总线探针,使用 Xilinx Virtex-E 高速 FPGA 设备分析探针截取的总线信号,成功破解了 Xbox 南北桥信息传输所采用的高速总线协议 HyperTransport[7],最终获取了 Xbox 用于数字版权保护的 RC4 密钥,使得攻击者可以随意安装非授权软件,而整个攻击成本仅需几十美元.文献[8]证明即使设备配备了总线加密机制也可能遭受总线攻击.该文献提出的密文指令搜索攻击通过总线向 CPU 发送密文指令并观察 CPU 的响应来推测实际 CPU 指令,在推测出足够多的密文指令后利用这些密文指令构建攻击程序,最终成功将受保护的内存内容发送给攻击者.该攻击方法成功应用在安全芯片 DS5002FP 微控制器上,破解了其内部的加密密钥.英国信息安全公司 NCC Group 成功对可信计算技术的硬件安全基础 TPM(trusted platform module)芯片实施了总线攻击,并开发了配套的攻击工具 TPM Genie[9].基于该攻击工具可实施 3 种类型的攻击:篡改 TPM 芯片内部的完整性度量值,影响计算机系统的信任链、远程证明和数据封装等可信计算功能;修改 TPM 返回结果以减弱 TPM 芯片内部的随机数发生器的安全性,该攻击可影响主机上基于 TPM 随机数的密码学操作的安全性;通过在总线上伪造 TPM 返回结果影响主机上 TPM 软件栈的安全性,基于该攻击找到了 30 多个 TPM 驱动的内存安全隐患,影响到 Linux 内核、U-boot 和 tboot 等多个重要计算机系统软件.总线攻击由于实施容易、成本低,因而很容易与其他攻击结合形成新的攻击,加州大学伯克利分校的研究团队结合总线攻击与内存侧信道攻击提出了针对 Intel SGX 的新型侧信道攻击方法[10],他们称其为片外侧信道.Intel SGX 虽然对处理器芯片外部的 DRAM 内存进行了加密和完整性保护,即数据总线上的数据都是密文,但是未对地址总线进行加密,因此攻击者可以通过总线探针获得地址总线的信号,利用后台的信号分析仪获得要访问的内存地址,从而得知内存的访问模式,即可进一步实施侧信道攻击.该攻击方法比较适合攻击数据量大的程序,因此对大数据平台以及云平台上的人工智能算法具有较大的安全威胁.

DMA 攻击是一种利用 DMA 传输机制绕过内存管理单元(memory management unit,简称 MMU)和 CPU 而直接访问物理内存的攻击方法.法国国家网络安全局基于网卡控制器的一个漏洞提出了一种远程攻击方法[11]:通过远程向网卡发送一些定制攻击包就能远程控制网卡.基于该方法攻击者可通过网卡的 DMA 访问能力进一步访问计算机的全部内存并控制整个计算机系统.随后该团队提出了对应的安全解决方案[12].瑞士研究人员基于网卡和声卡的 DMA 能力提出了一种内存探测方法[13],该攻击具体实现为一个安全 shell 工具,通过远程固件刷新将该 shell 安装到网卡或声卡上.德国柏林高等工业学校的研究团队提出了一种基于 DMA 的恶意软件 DAGGER[14].DAGGER 实现在 Intel 的管理引擎(manageability engine,简称 ME)上,该攻击利用 ME 的 DMA 硬件获得对主机内存的访问权限,通过搜索主机内存空间获得键盘缓存区地址,然后通过监控该缓存区获取用户的键盘输入信息.该团队同时在 Windows 和 Linux 操作系统上实现了原型系统,实验结果表明该攻击能够快速获得用户的键盘输入信息,并且在平台启动早期就能够获取键盘输入信息,譬如 Linux 系统的硬盘加密口令.

2 抗板级物理攻击相关技术

2.1 片上内存

片上内存(on-chip memory,简称 OCM)已经是嵌入式领域处理器芯片(又称为片上系统)的一个基础组件,基本上所有的嵌入式处理器芯片都配备了片上内存.文献[49]调研了智能终端领域十几款主流的片上系统,统计结果表明大部分芯片都配备有 100KB 以上的片上内存.

片上内存相比片外内存(一般为 DRAM 芯片)在性能和安全性两方面都具有一定的优势.在性能方面,片上内存因为通过内部高速总线与 CPU 连接,因此 CPU 访问片上内存的速度一般比片外内存快.在安全性方面,片上内存具有更高的物理安全性,具有抵抗板级物理攻击的能力:片上内存没有向芯片外部暴露物理引脚(physical pin),因此其数据信息和地址信息不会从芯片外引脚或总线上泄露出去.但是,物理攻击者可以通过冷启动攻击加载恶意攻击代码或恶意 DMA 设备来获取片上内存的内容.对于冷启动攻击方式,英国哥伦比亚大学研究团队[50]以及国内中科院软件所研究团队[51]在实际物理开发板上的实验表明智能终端处理器芯片的 BootROM 代码在重启后一般会对片上内存进行清空.而冷启动攻击的攻击程序只能在重启设备后才能加载,因此冷启动攻击无法获取片上内存的内容.对于 DMA 方式的攻击,硬件系统或软件系统设计者需要将片上内存与恶意 DMA 外设进行隔离以防止它们通过 DMA 方式获取片上内存的内容.国内中科院软件所的研究团队[49,52]使用 ARM TrustZone 的隔离能力来抵抗这种攻击,其他可采用的隔离方式还包括嵌套内核隔离机制[53−56],该隔离机制允许系统软件设计者在处理器没有提供类似 ARM TrustZone 这种硬件隔离机制的情况下逻辑隔离出一块地址空间.

2.2 基于Merkle Tree的内存完整性保护技术

Merkle Tree 是保护内存完整性的主要机制.传统的 Merkle Tree 内存保护方案如下(如图 3 左所示):首先将内存划分为一个个内存块并对每个内存块进行消息验证码(message authentication code,简称 MAC)计算,得到的 MAC 值作为 Merkle Tree 的叶子节点,然后自叶子节点向上一层一层进行 MAC 计算,最终计算到根节点.该树根节点可以看作整个内存的完整性值,需要存储在芯片内部.每次从片外内存读取数据时,都需要从对应叶子节点计算到根节点,用计算得到的根节点值与芯片内部的标准完整性值对比,以检测读取的片外内存是否被篡改;每次将芯片内的数据(譬如缓存或片上内存的数据)写回片外内存上时,需要更新对应叶子节点以及该节点到树根路径上所有节点的值.

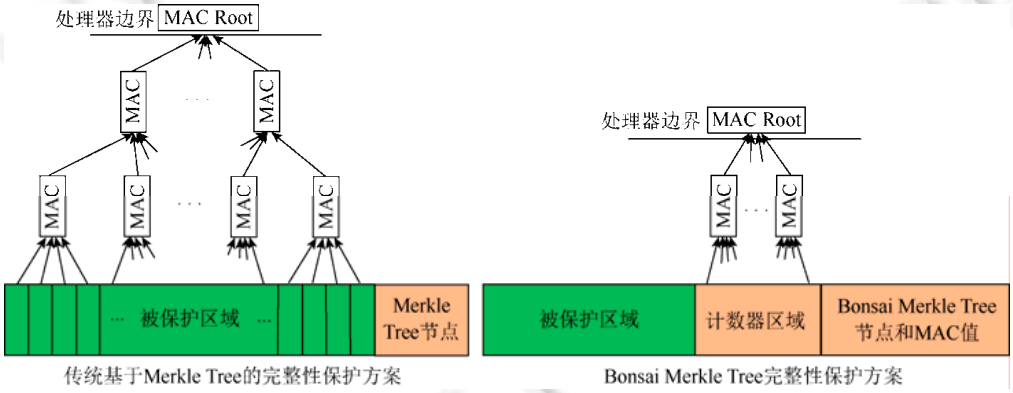

Fig.3 Comparison between Merkle Tree and Bonsai Merkle Tree

图 3 Merkle Tree 与 Bonsai Merkle Tree 对比

基于 Merkle Tree 的内存完整性保护技术每次验证或更新数据都需要计算从叶子节点到树根路径上所有节点的值,给系统带来较重的计算负载.为了降低计算负载,MIT CSAIL 实验室提出了将树节点存储在缓存上的优化方法[57],一旦某一节点被缓存在了芯片内部,该节点可以看作树根,完整性校验和更新只需要从叶子节点计算到该缓存节点即可.但是该优化方法只有在 Merkle Tree 比较小时才能提高完整性验证的效率,如果 Merkle Tree 较大,树节点会占用大量缓存,极端情况下会占用 50%的缓存,导致普通应用程序经常发生缓存缺失,降低应用程序运行速度.

因此,降低 Merkle Tree 大小成为优化内存完整性保护的一个重要研究方向.北卡罗莱纳州立大学的研究团队提出了 Bonsai Merkle Tree 的概念[29].Bonsai Merkle Tree 为每个内存块分配一个计数器,然后对每个内存块和对应计数器计算 MAC 值,MAC 值就是该内存块的完整性值.因为 MAC 值已经对内存进行了完整性保护,所以

实际上只需对内存的计数器进行完整性保护即可.图 3(右)描述了 Bonsai Merkle Tree 方法,其中被保护区域是普通的需要保护的内存,计数器区域用于存放计算内存块 MAC 值时用到的计数器,最后一个区域存放对计数器区域进行完整性保护的树节点和所有内存块的MAC值.为防止频繁更新内存导致计数器溢出,Bonsai Merkle Tree 除了为每个内存块设置一个局部计数器,还为较大的内存区域(譬如一个内存页)设置一个全局计数器,局部计数器达到最大值后将全局计数器加 1,然后局部计数器就可以归零重新计数.这种计数器设置方式避免为每个内存块设置大的计数器,降低计数器占用的内存.但是这种方式的一个缺点是一旦全局计数器更新,就需要将其对应的所有内存块(譬如一个内存页)重新加密.Bonsai Merkle Tree 方式极大地降低了需要完整性保护的内存大小.以内存块为 64B 和局部计数器为 8 bit 为例,计数器所占用内存与普通数据内存比例仅为 1:64.该研究团队的实验结果表明,Bonsai Merkle Tree 将完整性校验带来的负载从 12.1%降低到 1.8%,并且将完整性保护所占用的内存(包括树节点和计数器区域等)从 33.5%降低到 21.5%.

3 软件方式的抗板级物理攻击防御技术

软件方式的抗板级物理攻击的操作系统防御技术借助处理器芯片内部的通用存储资源,如 CPU 专用寄存器、CPU 缓存、片内 RAM 等,无需修改处理器硬件即可提供较高的物理安全性,成为系统安全领域的一个研究热点.这方面的研究工作按照保护范围可以分为 3 类:第 1 类主要关注密钥等对安全性要求非常高的数据,保障其存储和相应密码算法实现的安全性;第 2 类从保护安全攸关应用程序出发,借助片内存储资源保护安全应用的物理安全性;第 3 类从保护整个系统出发,着重考虑运用有限的片内存储空间为包含操作系统和应用程序在内的完整系统提供安全执行环境.本节按上述分类对已有研究工作进行全面分析和系统总结.

3.1 密钥及密码算法安全保障研究

密钥的安全性是许多系统安全机制的基础,为防止攻击者通过板级物理攻击从片外RAM中获得密钥[1,2,14,58,59],研究人员提出利用片内存储保护密钥和相应密码算法实现以抵抗板级物理攻击的解决方案,从而保证系统中密码原语的安全性.这类工作的基本思路是:在系统运行期间,不再将密钥存储在片外 RAM 中,而将其存储在片内存储中,并确保任何时刻密钥和有助于推测密钥的中间计算结果都不会以明文形式出现在片外 RAM 中.用于存储密钥和进行加解密计算的片内存储包括 CPU 内部寄存器和缓存、GPU 内部寄存器和缓存,利用这些存储空间将密钥和加解密过程限制在处理器芯片内部,避免与密钥相关的敏感数据受到板级物理攻击.保护密钥和密码算法的方案按其实现层次可以分为 3 类:应用层方案、内核层方案和 hypervisor 层方案,本节分别对这 3 类方案进行分析和总结.

3.1.1 应用层密钥和密码算法保护方案

这类方案在用户层实现,利用片内存储资源进行密钥存储和密码学计算,供其他应用程序调用,以保护应用程序密钥的安全性.基于片内存储的保护方案需要防止普通应用程序访问敏感数据的存储位置读取或者覆盖密钥,以及防止上下文切换等过程将敏感数据保存到片外 RAM 中,这些限制通常需要 root 权限才能实现,因此这类方案通常在内核层实现,在应用层实现的密钥和密码算法保护方案较少,目前已知的仅有 Parker 等提出的方案[60]和 Peapods[61]方案.

Parker 等的方案[60]在 x86 架构中利用 SSE(streaming SIMD extension)扩展指令集[62]的 XMM 寄存器组实现 RSA 算法.该方案不将完整的 RSA 私钥明文存储在片外 RAM 中,而是先将其混淆处理后再存储在片外 RAM 中,以防止内存泄露攻击;在进行 RSA 运算时,片外 RAM 中混淆后的私钥先经过重组得到原始私钥再加载到 SSE 寄存器进行密码学计算,以确保私钥明文仅出现在 CPU 寄存器中.该方案通过一个可加载内核模块(loadable kernel module,简称 LKM)在 RSA 运算期间为不具有 root 权限的应用程序禁用中断,以防止 SSE 寄存器的内容因上下文切换泄露到片外 RAM 中.该方案基于 OpenSSL 实现,因 SSE 寄存器容量有限,不能使用中国剩余定理(Chinese remainder theorem,简称 CRT)加速 RSA 的模幂运算,其 RSA-1024 签名操作的性能比 OpenSSL 中常规 RSA 实现低 4.4 倍.该方案的一个安全问题是:如果攻击者得到用于重组私钥的代码,就可以获知私钥混淆算法,攻击者实施冷启动攻击获得片外 RAM 中经混淆的私钥后,可以通过重组得到 RSA 私钥.

Peapods[61]是一种基于 LLVM 的编译器增强工具,旨在利用事务内存为软件加密引擎中的密钥提供用户模式的保护.Peapods 在编译阶段解析程序开发人员指定的要保护的敏感变量(密钥)和代码片断(密码学计算),将其作为 peapod 封装在事务中,并为程序添加初始化代码,用于生成 AES 主密钥,用主密钥加密敏感变量,以及将主密钥加载到 XMM 寄存器中.在执行阶段,Peapods 采用事务内存保护密钥,将密码学计算作为事务在 CPU L1 数据缓存中执行,以保证其原子性,事务结束前会重新加密密钥和中间计算结果,以确保敏感变量仅以密文形式出现在 RAM 中.Peapods 支持将耗时的密码学计算拆分为多个事务以提高性能,采用 peapods 保护开源加密引擎 PolarSSL 中的 RSA-2048 私钥计算时,引入的性能开销约为 10%.

3.1.2 内核层密钥和密码算法保护方案

这类方案在内核层实现,通常作为操作系统补丁.这类方案用于保护密钥的片内存储资源通常具有以下特点:(1) 能够特权占用.为防止用户空间进程访问密钥,密钥存储资源应能够设置为仅特权级别才可以访问;(2) 能够不将数据泄露到片外 RAM.密钥存储资源应能够设置为不将数据存储到片外 RAM 中,如不在上下文之中或不写回到片外 RAM;(3) 被占用不影响系统运行.密钥存储资源在系统运行期间由密钥保护机制独占使用,不能用于预期用途,缺少该资源应仍能保持大部分应用程序的二进制兼容性;(4) 具有足够保护密钥的容量.密钥存储资源应具备密钥存储和密码学计算所需的存储空间.基于以上特点,许多保护方案选择调试寄存器、SSE 扩展的 XMM 寄存器组等作为密钥存储资源.为防止密码学计算过程中上下文切换将敏感数据泄漏到片外 RAM,或其他内核模块访问敏感数据,密钥保护机制通常以原子方式执行.

内核层密钥和密码算法保护方案有 x86 架构中保护对称密码算法的方案 AESSE[63]、TRESOR[64]、Loop-Amnesia[65]、FrozenCache[66],ARM 架构中保护对称密码算法的方案 ARMORED[67,68]、Key-hiding[69]、Anti-forensics[70],x86 架构中保护非对称密码算法的方案 PRIME[71]、文献[72]中的方案、RegRSA[73]、Copker[74]、Mimosa[75],以及利用 GPU 保护对称和非对称密码算法的方案 PixelVault[76].

在 x86 架构对称密码算法保护方面,AESSE[63]是针对 x86 架构上 Linux 操作系统的 AES 抗冷启动攻击实现.与 Parker 等提出的方案[60]类似,AESSE 不将密钥存储在片外 RAM 中,而是将一个密钥和与密钥有关的轮密钥和中间状态等中间计算结果直接存储在 SSE 寄存器中,并在 SSE 寄存器中实现 AES 算法,确保密钥和任何中间计算结果不会离开 CPU.AESSE 在内核模式原子执行,SSE 仅在原子代码段中启用,以防止 SSE 寄存器中的敏感数据被读取或覆盖以及在上下文切换时被保存到片外 RAM 中.受限于 SSE 寄存器的容量,AESSE 对每个 128 bit 输入块的加解密运算都需要重新计算 10 个轮密钥以节约轮密钥的存储空间,造成较高的性能开销,其速度比内核标准 AES 慢 6 倍.AESSE 的另外一个问题是兼容性:用 SSE 寄存器长期存储 AES 密钥,影响使用 SSE 扩展的多媒体、数学和 3D 等应用.TRESOR[64]是 AESSE 作者提出的后继方案,旨在解决 AESSE 在性能和兼容性两方面的问题,其基本思路与 AESSE 相同,但是不需要长期占用 SSE 寄存器,并大幅提高了 AES 加解密性能.为尽量避免引起兼容性问题,TRESOR 没有使用 SSE 寄存器长期存储密钥,而是将 x86 架构中的调试寄存器用于长期存储 AES 密钥,仅在每次加解密时将 SSE 寄存器用于加载密钥及存储轮密钥和中间状态,因而无需长期占用 SSE 寄存器.为提高算法性能,TRESOR 采用了用于硬件加速 AES 实现的 Intel AES-NI(Intel AES new instructions)扩展指令集[77,78].尽管在每次加解密时仍需重新计算轮密钥,但是通过使用 AES-NI 指令,TRESOR 可达到与标准 AES 相同或更好的性能.该方案的一个问题是:长期占用调试寄存器的全部断点寄存器,运行 TRESOR 的系统无法再设置硬件断点,会对 GDB 等调试程序造成影响.Loop-Amnesia[65]是一个与 TRESOR 类似的方案,其与 TRESOR 的不同点在于:(1) 使用 MSRs(model-specific registers)中的性能计数寄存器而不是调试寄存器长期存储 AES 密钥;(2) 将主密钥存储在性能计数寄存器中,将其他密钥经主密钥加密后存储在片外 RAM 中,以这种方式用存储一个密钥的能力支持对多密钥的保护;(3) 基于 Linux 的开源磁盘加密软件包 loop-AES[79]实现,不使用 AES-NI 指令,导致其性能低于 TRESOR 和标准 AES;(4) 使用 x86-64 架构中全部通用寄存器而不是 SSE 寄存器存储轮密钥.由于占用了 MSRs 的性能计数寄存器,运行 Loop-Amnesia 的系统不能使用硬件辅助的性能分析器,会影响硬件性能分析工具的正常运行.上述方案仅能抵抗冷启动攻击这一种板级物理攻击,不能抵抗 DMA 攻击.Blass 等提出对类似 TRESOR 和 Loop-Amnesia 的方案的一种 DMA 攻击 TRESOR-

Hunt[80].TRESOR-Hunt 利用 DMA 传输的写操作向内核内存中注入攻击代码进行特权提升,攻击代码以 root 权限执行将受保护的密钥从寄存器提取到片外 RAM 的指定位置,之后即可利用标准 DMA 传输获得片外 RAM 中存储的密钥.FrozenCache[66]提出将密钥存储在 CPU 缓存中以进行保护的思路:将内存映射到 CPU 的 L1 缓存,使用缓存存储 AES 密钥和轮密钥,并清除片外 RAM 中的密钥;通过让所有处理器核心进入 no-fill 模式,也称 CAR(cache-as-RAM)模式,来冻结缓存中的密钥,以防止敏感数据被刷新到片外 RAM 中.FrozenCache 使 CPU 缓存不能发挥正常作用,每次内存访问都会导致缓存未命中.此外,因安全高效实现 FrozenCache 涉及很多细节问题,作者未实现该方案的原型系统.

上述密钥和密码算法保护方案都是针对 x86 架构平台,然而 ARM 架构平台的安全性也受到板级物理攻击的威胁,如 FROST 攻击[2].在 ARM 架构对称密码算法保护方面,ARMORED[67,68]是针对 ARM 架构和安卓操作系统的 TRESOR 修改版,利用 ARM-32 架构的调试寄存器和多媒体寄存器组 NEON[81]实现 AES 算法.ARMORED 将 AES 密钥长期存储在调试寄存器的断点寄存器和监视点寄存器中,受 ARM 架构调试寄存器数量的限制,仅支持 AES-128.每次加解密时,AES 密钥被加载到 NEON 寄存器,轮密钥和中间状态也都存储在 NEON 寄存器中.为提高性能,ARMORED 将原子代码中加解密范围从 1 个 AES 输入块增加到 16 个,从而减少轮密钥计算次数,即轮密钥计算频率由每输入 128 bit 计算 1 次减少为每输入 2K bit 计算 1 次.ARMORED 实现为安卓的 LKM,并通过集成到 Crypto-API 支持基于 ARM 的 Linux 内核,其性能比通用 AES 低 2.3 倍.该方案的一个安全问题是:ARM 架构平台上的 JTAG 设备可以通过 JTAG 接口破坏内核完整性,因而 ARMORED 不能抵抗通过 JTAG 接口进行的攻击.Key-hiding[69]在 ARMORED 基础上进行了改进,将保护密钥的长度扩大到 256 bit.ARMORED 无法存储超过 128 bit 密钥的原因是:ARM 架构要求所有指令都开始于能被 4 整除的地址,因此每个值寄存器的最后 2 位必须为 0,导致 ARMORED 用 4 个 32 bit 断点寄存器和 4 个 32 bit 监视点寄存器不能存储 256 bit 密钥.Key-hiding 的解决方案是利用与每个断点或监视点寄存器对应的控制寄存器,将不能在断点或监视点寄存器最后 2 位存储的数据存储到相应控制寄存器的第 1 位和第 2 位中,从而利用 4 个断点寄存器和 4 个监视点寄存器,以及相应的控制寄存器,在 ARM-32 架构中保护 256 bit AES 密钥.因为缺少与原型系统兼容的 AES-256 算法,Key-hiding 未进行性能测试.Anti-forensics[70]是一种针对安卓设备抗 FROST 攻击[2]的 AES 密钥保护方案.与前述方案不同,Anti-forensics 不是将密钥和加解密算法限制在 CPU 内部,而是通过更改密钥在内存中的存储位置来防止 FROST 攻击.Anti-forensics 将 AES 密钥存放在存储安卓启动命令行参数的内存地址,系统重新启动时密钥将被命令行参数覆盖,使 FROST 攻击无法获得密钥.该方案不需要占用 CPU 寄存器等其他存储资源,不会影响系统正常运行,且对 AES 算法性能影响较小,仅 1.03 倍性能降低,其性能比 ARMORED 高 2 倍,但是由于密钥存储空间的限制,Anti-forensics 只能保护一个密钥.

上述内核层对称密钥和密码算法保护方法被扩展到非对称密码算法保护中.与对称密码算法相比,非对称密码算法的存储或运算通常需要占用更多存储空间.以 RSA 算法为例,RSA 实验室[82]和 NIST[83]建议 RSA 私钥长度最少为 2048 bit,并且与 RSA 私钥相关的解密/签名运算也需要大量存储空间.在 x86 架构非对称密码算法保护方面,PRIME[71]、RegRSA[73]、Copker[74]、Mimosa[75]的基本思路类似:将受 TRESOR 保护的 AES 密钥用作主密钥加密 RSA 私钥,只需长期占用调试寄存器存储 AES 密钥,而将加密后的 RSA 私钥存储在片外 RAM 中,以解决私钥较长存储空间不足的问题,然后选择其他 CPU 内部存储保护 AES 算法和 RSA 算法,仅在进行 RSA 解密/签名的原子代码中独占这些存储资源,以降低对系统正常运行的影响.PRIME[71]利用 Intel AVX(advanced vector extensions)[84]的多媒体寄存器组实现了 RSA-2048 算法,其在 AVX 寄存器中解密 RSA 私钥以及进行 RSA 运算.PRIME 采用蒙哥马利(Montgomery)方法在 AVX 寄存器中实现了 RSA 的模幂运算,但是需要将一些不会影响 RSA 安全性的中间结果短暂存储在片外 RAM 中.由于存储空间限制,PRIME 不能使用 CRT 加速运算,导致其运行速度比常规 RSA 实现慢约 9 倍.与 PRIME 思路类似,Yang 等人[72]利用 x86-64 架构的调试寄存器和 AVX 寄存器实现椭圆曲线密码算法 ECDH(elliptic curve Diffie-Hellman),将上述寄存器分别用于长期存储 163 bit 私钥和计算 ECDH 共享密钥.RegRSA[73]提出寄存器缓冲区的概念,将所有可用 CPU 寄存器用作安全数据缓冲区,敏感数据仅存储在寄存器缓冲区中,并利用寄存器缓冲区实现了使用 CRT 加速的 RSA-2048 算法.该方案

使用标量指令和寄存器执行计算,在向量寄存器中维护中间结果,利用 704B 的寄存器缓冲区(16 个 64 bit 通用寄存器,8 个 64 bit MM 寄存器[85]和 16 个 256 bit YMM 寄存器[84])实现完全在寄存器缓冲区中计算的 1 024bit 蒙哥马利乘法,且允许多个实例同时在多核 CPU 上运行.由于寄存器缓冲区的容量有限,一些中间结果需要经 AES 主密钥加密后存储到片外 RAM 中.RegRSA 解决了 PRIME 不能使用 CRT 加速的问题,其性能是 OpenSSL 中常规 RSA 实现的 74%,且其性能超过了 PRIME.Copker[74]利用 CPU 缓存实现 RSA 算法,缓存的容量优势使其支持更长的私钥和更高效的算法,AES 解密和 RSA 解密/签名操作都在缓存中进行,且能够使用 CRT 加速运算,从而获得更好的性能.但是,Copker 在密码学计算过程中强制所有共享 LLC(last-level cache)的其他处理器核心进入 no-fill 模式,以防止这些核心将 L1 缓存刷新到片外 RAM 中,急剧降低了这些核心的内存访问性能,且 Copker 不支持共享 LLC 的核心上的多个实例.Copker 原型系统采用 128bit AES 主密钥保护 2 048bit RSA 私钥,其运行速度比常规 RSA 实现慢约 3 倍.Mimosa[75]利用 Intel TSX 机制在 CPU L1 数据缓存中为 RSA 提供安全计算环境.Intel TSX[86]是 x86 架构的指令集扩展,增加了对硬件事务内存(hardware transactional memory,简称 HTM)的支持.在密码学计算期间,Mimosa 使用 HTM 保证私钥和中间结果等敏感数据只出现在 CPU 绑定缓存中,不会泄露到片外 RAM,并且 HTM 的原子性保证任何非授权的私钥访问会导致密码学计算终止,所有敏感数据通过硬件机制自动清除,以防止软件形式的密钥窃取攻击.Mimosa 的性能可以与常规 RSA 实现媲美,并且借助 Intel TSX 的硬件支持,Mimosa 的性能优于 PRIME 和 Copker.

在利用 GPU 保护密码算法方面,PixelVault[76]基于 GPU 实现 AES 和 RSA 算法保护,其基本思想是在 GPU 中执行完整的密码算法,不使用主机和主机可访问的内存.PixelVault 将密钥和敏感代码分别存储在 GPU 的寄存器和指令缓存中,利用 GPU 与主机的隔离特性防止特权主机代码访问 GPU 上的敏感数据和代码,以保证敏感数据和代码的安全性.GPU 的非抢占执行特点使 PixelVault 独占执行,避免因上下文切换而导致敏感数据泄露.PixelVault 将计算密集型的加解密操作在 GPU 中执行,获得了较好的性能.Zhu 等人[87]分析了 PixelVault 关于 GPU 硬件特性的假设,指出该方案依赖的一些安全假设存在漏洞,利用控制寄存器和调试器的更新功能能够绕过 PixelVault 的安全机制破坏其安全性,从而指出 PixelVault 无法在其威胁模型下保证主密钥的机密性.之后,ZeroKernel[88]的作者指出对 PixelVault 的另一种攻击,攻击者通过操纵 GPU 上下文可以将寄存器中存储的主密钥泄露到设备内存的特定位置.

内核层密钥和密码算法保护方案可以抵抗冷启动攻击,但通常难以抵抗对系统空间具有写访问权限的攻击者的攻击.例如,具有 root 权限的攻击者可以轻松加载内核模块,利用该模块将密钥从 CPU 寄存器提取到片外 RAM 中.Blass 和 Robertson 将不能在内核特权级运行任意代码定义为内核完整性属性[80].即使是 PixelVault 在其初始化期间也需要保证操作系统内核的完整性.事实上这样的属性很难满足.在实际应用中,恶意软件通常可以获取 root 权限,且 DMA 攻击可以通过物理访问将恶意代码写入系统空间,而很多方案的威胁模型中并没有考虑抵抗 DMA 攻击.此外,大部分基于寄存器的方案只能保护物理平台密钥的安全性,无法保护虚拟机中的密钥,这是因为客户机的寄存器在虚拟机切换时会存储到主机的片外 RAM 中,存在被物理攻击的风险.

3.1.3 Hypervisor 层密钥和密码算法保护方案

内核层保护方案依赖于操作系统内核的完整性,并且大部分方案不能抵抗 DMA 攻击,虽然 DMA 攻击可以利用 IOMMU 防御,但是一些设备没有配备该机制.为保护不同操作系统中的密钥,即实现与操作系统无关的密钥和密码算法保护,研究人员提出 hypervisor 层保护方案.Hypervisor 层保护方案具有以下优势:(1) hypervisor 与操作系统隔离,即使是 root 权限用户和本地特权提升也不会破坏密码学计算过程或恢复出密钥;(2) 与操作系统类型无关,可以保护不同类型的操作系统;(3) 一些与安全性相关的硬件设置可以在 hypervisor 层统一设置,如通过设置 IOMMU 集中防御 DMA 攻击;(4) 这类 hypervisor 可具有较小的 TCB,降低了系统存在漏洞的风险,也便于通过形式化方法验证其正确性.

在 hypervisor 层实现的密钥和密码算法保护方案较少,目前已知的仅有 TRESOR 的作者提出的 TreVisor 方案[89].该方案将 TRESOR 功能从内核层下移到 hypervisor,可以透明的保护客户机操作系统的密钥.TreVisor 建立在 BitVisor 基础上,BitVisor[90]是一种轻量级虚拟机监视器,为单个客户机实现各种安全功能,利用虚拟化技

术增强 I/O 设备安全性,其 IOMMU 设置可抵抗 DMA 攻击,但是 BitVisor 不能防御冷启动攻击.TreVisor 结合了 TRESOR 和 BitVisor 这两个项目,借助前者抗冷启动攻击,而后者抗 DMA 攻击的特性,形成了一个 hyptervisor 层抗冷启动攻击和 DMA 攻击的、对操作系统透明的安全加解密方案.作者认为将密钥存储在用主密钥加密的片外 RAM 中会带来明显的性能缺陷,因此为保证性能,TreVisor 仅保护一个密钥,其性能比 AES-NI 下降约 33%.由于 TreVisor 占用了调试寄存器且使用了 Intel VT-x,造成客户机操作系统不完全支持调试器,无法设置硬件断点,也不完全支持 VirtualBox 和 VMware 等虚拟化软件.该方案存在的安全问题是:(1) 依赖于 hypervisor 的安全性,hypervisor 运行在片外 RAM 中,如果被攻击者通过板级物理攻击攻破,将破坏整个系统的安全性;(2) 在多核架构中,TreVisor 需要在所有处理器核心间分配密钥,以将密钥写入不同的核心,方法是将密钥从 BSP(boot strap processor)复制到片外 RAM,再复制到 AP(application processors),该过程对于冷启动攻击是不安全的.

3.1.4 小 结

密钥和密码算法保护方案利用片内存储资源将密钥存储和密码学计算限制在处理器芯片内部,能够抵抗板级物理攻击,特别是冷启动攻击.AESSE 是保护 AES 密钥及相应密码算法安全性的开创性工作,其后续工作 TRESOR 解决了 AESSE 的性能问题,已经成为该研究方向的基础技术.譬如,ARMORED 将 TRESOR 移植到 ARM CPU 硬件平台上的安卓操作系统中;Key-hiding 将 TRESOR 保护的 AES 密钥长度从 128bit 扩展到 256bit;TreVisor 系统软件利用 TRESOR 提供的高安全密码服务保护上层虚拟机内部的密钥.此外,大部分 RSA 密钥和密码算法保护方案都采用 TRESOR 的 AES 算法加密保护 RSA 私钥.在非对称密钥和密码算法保护方面,各项工作主要保护 RSA 密钥和算法的安全性,这些工作利用 CPU 架构提供的各种安全硬件机制来实施保护,其中 RegRSA 和 Mimosa 方案分别通过实现 CRT 加速和采用 Intel HTM 机制提高了算法实现的性能,是这些工作中性能比较好的方案.

但是这类方案在安全性和实用性方面仍存在一些问题.在安全性方面,这类方案可用的安全存储空间有限,只能保护密钥和中间计算结果等很小部分敏感数据,并不能保护其他敏感信息,更不能保护整个片外 RAM.此外,这类方案依赖于包括操作系统或 hypervisor 在内的庞大 TCB.在实用性方面,这类方案占用了 CPU/GPU 内部专用存储资源,使该存储资源无法用于其预期用途,会影响系统特定功能.此外,某些资源的独占使用会影响系统性能.例如,在禁用中断以原子执行密码学计算的情况下,操作系统对交互事件的响应可能会受到影响,鼠标和键盘等事件引起的中断被延迟到加解密/签名操作结束之后才能响应;运行基于缓存的保护方案时需要利用缓存控制指令防止共享缓存的其他核心访问为密码学计算保留的缓存,会降低其他核心上并发任务的性能,同时大部分方案需要将缓存设置到 no-fill 模式来保护密钥,而缓存在该模式与正常运行模式之间切换比较耗时.

3.2 应用程序抗物理攻击防御技术研究

第 3.1 节的方案只对密钥和密码算法进行了保护,而实际上应用程序的其他内存同样也可能存放着安全敏感的数据,譬如口令、个人生物信息和图像信息等,保护这些信息的安全性同样重要.为此,学术界提出了保护应用程序整体或部分关键数据免受物理攻击的方案:其中一部分方案重点针对冷启动攻击,考虑到系统休眠时是冷启动攻击的主要时机,所以这些方案重点保护系统休眠时的内存内容;其余方案考虑的比较完善,能够在系统运行时为应用程序提供物理防护.值得注意的是,很多方案使用第 3.1 节中的抗物理攻击密码算法作为保护应用程序的密码原语.

3.2.1 系统休眠时应用程序保护方案

研究人员为常见的开源操作系统提出了各种防护方案来抵抗休眠或锁屏时的冷启动攻击.加拿大卡尔顿大学为安卓系统设计了一种抵抗冷启动攻击的安全运行模式 Deadbolt[91].该模式启动前会清空安卓的全盘加密密钥和卸载全盘加密机制所保护的用户分区,从而保障用户数据分区免受冷启动攻击.该模式还加载一个仅包含基本用户应用的临时用户分区,为用户提供一些基本的手机功能.该方法的安全弱点是只能保障 Deadbolt 模式下设备的安全性,设备正常使用时仍存在被攻击的风险,因此只能缓解物理攻击.Sentry[50]是一种抵抗 FROST[2]等针对移动设备的物理攻击的方案,该方案利用 ARM SoC 架构中的片上内存或二级缓存实现了可抵抗物理攻击的 AES 算法,用于在安卓系统锁屏时对片外 RAM 上的机密数据进行加密保护,并基于 ARM 内存管

理机制实现了在保持 RAM 内存页为密文状态的情况下将明文写入二级缓存的技术,使得在安卓系统锁屏后内存数据为密文的情况下后台程序仍能正常运行,该方案采用禁止中断的方式防止 AES 中间计算结果被切换到片外 RAM.

为增强 Linux 操作系统休眠时的物理安全性,德国 Fraunhofer 协会的应用集成信息安全研究所将 Linux 的全盘加密机制从磁盘加密扩展到了内存加密[92].设备休眠和启动时 Linux 操作系统会转换所有进程状态,该方案在进程转换状态之前调用全盘加密机制的加解密接口对进程的虚拟内存地址空间进行加解密,之后清空磁盘加密密钥以及相关密码算法的数据结构以防止攻击者从内存镜像中获取该密钥.该团队后续又基于 Linux CGroup 的 freezer 子系统实现了对进程组的内存加密方案 Freeze&Crypt[93].该方案的技术思路是在 freezer 子系统停止和启动进程组的流程中加入进程加密和解密步骤,从而达到同时加密保护一个进程组的目标.该方案适用于移动设备等活动进程和休眠进程共存的场景.该团队在安卓容器场景中展示了方案的可行性,只有活动的安卓容器在内存中处于明文状态,其他后台容器均被加密以抵抗物理攻击.

另外也有研究人员提出使用底层 hypervisor 来对上层操作系统进行内存加密的思路,从而使内存保护与操作系统无关.加拿大康考迪亚大学提出了一个与操作系统无关的计算机内存加密方案 Hypnoguard[94].该方案利用 Intel 的动态信任链技术 TXT 构建了一个动态可信执行环境 Hypnoguard 对休眠前的内存进行加密.该方案的加密密钥由安全芯片 TPM 保护,以此抵抗物理攻击.同时该方案使用了可信计算的封装机制和用户的口令认证机制,保证只有未被篡改的 Hypnoguard 和合法用户才能解密内存唤醒计算机.为提高加密速度,该方案使用了 AES-NI 硬件加密指令和多核计算,保证在 1s 之内实现对 8G 内存的保护,降低用户等待时间.

3.2.2 系统运行时应用程序保护方案

在系统休眠时提供物理防护只能解决一部分场景的安全问题,实际上并没有为设备提供全面的物理防护:攻击者在设备运行时仍可以实施物理攻击.为解决该问题,研究人员提出能够在设备运行时对安全敏感应用进行保护的方案.这些方案中,一部分是通过内存滑动窗口技术来减缓物理攻击的威胁;一部分是通过片上内存、缓存和寄存器等处理器片内存储来保护应用的关键数据或代码,从而更加安全的抵抗物理攻击;还有一部分是利用处理器其他硬件机制来保护应用程序免受物理攻击.本节按照方案采用的内存滑动窗口、片上内存、缓存、寄存器以及其他硬件机制这 5 类进行划分对相关方案进行分类总结.

(1) 基于内存滑动窗口的内存防护技术研究

减缓针对内存的物理攻击的一个方法是减少内存中明文内存页的比例,基于该思路研究人员提出了滑动窗口技术.该技术的主要思路是只保持一部分内存页为明文状态,为应用提供工作内存,其余内存页均处于加密状态.CryptKeeper[95]实现为一个具有加解密功能的虚拟内存管理器,保证任何时刻只有一小块固定大小的内存处于明文状态用以支撑系统运行,其他片外 RAM 上的所有数据都处于加密状态,以缓解物理攻击.由于部分数据仍在片外 RAM 中以明文状态存储,因此该方案并没有彻底解决板级物理攻击问题.德国埃尔朗根-纽伦堡大学提出了 RamCrypt 方案[96],目标是保护进程私有数据在运行时的安全性.该方案借助 TRESOR[64]等方案的抗物理攻击密码算法,对进程地址空间中的 BSS 以及堆栈等私有数据空间进行加密保护,并基于操作系统的请求分页机制保证这些地址空间中的内存页只有被操作系统访问时才被解密,从而大大降低了数据在内存中以明文形式存在的时间.为降低性能损失,该方案实现了一个滑动窗口,保证进程在任意时刻只有滑动窗口内的内存页处于明文状态,是安全与性能的折衷方案.实验结果表明,一些密钥等机密数据只有 3%的时间处于明文状态,大大减少了物理攻击者的攻击时机.

(2) 基于片上内存的防护技术研究

CryptMe 方案[97]将通用操作系统的进程迁移到 ARM TrustZone 的安全世界来运行,并设计了一个特权软件来维护进程在安全世界的执行环境.该特权软件首先通过同步进程在两个世界间的进程结构体和页表等关键数据结构体以及转发系统调用来维持进程在安全世界的执行环境;其次,将受保护进程的数据段映射到片上内存以抵抗物理攻击,并在片内 RAM 上设计滑动窗口机制以解决片上内存无法同时加载进程全部数据的问题.MemVault 是针对安卓设备的细粒度片外 RAM 加密方案[98].该方案利用安卓的动态污点跟踪和分析工具

TaintDroid[99]跟踪标记敏感数据传播,将片外 RAM 中受污染的敏感数据对象加密,访问时将其解密存储在片上内存中.为避免频繁的加密操作,MemVault 使用 LRU(least recently used)算法将最常使用的对象保留在片上内存中.该方案需要借助 ARM TrustZone 或 SMMU[100]才能抵抗 DMA 攻击,并且因无法污染由本机代码处理的敏感数据,而无法为这类数据提供保护.Oath 是一种针对 TrustZone 平台的敏感数据保护方案[101],其思路是仅在 TrustZone 安全世界能够访问的片上内存中存储和处理敏感数据.在该方案中,内存拆分机制将可信应用划分为敏感部分和非敏感部分,仅将敏感部分加载到片上内存中存储和处理,从而支持多个可信应用执行;动态片上内存分配机制按需动态调整分配给安全世界的安全片上内存大小,将安全世界不使用的片上内存交给普通世界使用,以减轻对通用操作系统的影响.Oath 将敏感 I/O 数据存储在安全片上内存中,从而为用户提供可信用户界面.ZeroKernel 是在 GPU 片内存储隔离的上下文中执行轻量级任务的方案[88].在该方案中,ZeroKernel 的代码、数据和页表分别加载到 GPU 片内存储 I-Cache、C-Cache 和 TLB 中,以形成驻留缓存的上下文,而 ZeroKernel 的代码地址转换和代码 PTE 则分别从 TLB 和设备内存中删除,以防止攻击者访问其代码页.为防止攻击者重构 GPU 上下文,ZeroKernel 将 GPU 页表内存位置随机化并持续检测页表基地址是否被修改.该方案的局限是:ZeroKernel 代码大小不能超出 I-Cache 容量,并且由于 GPU 上下文信息已在设备内存中删除,ZeroKernel 不支持多个 GPU 内核执行.一些嵌入式系统利用片上内存来抵抗物理攻击(数据移出片上内存时需要加密保护),同时也利用片上内存的高速访问特性来存储高频访问数据以提高程序运行速度.韩国国民大学提出一种片上内存高效分配方法 DynaPoMP[102],通过优化片上内存专用于敏感数据的内存比例来提高嵌入式应用程序运行速度.DynaPoMP 使用经验方式来确定安全区域占用片上内存的比例:在各种比例情况下运行嵌入式应用程序,统计程序运行时间,程序运行时间最短的即为最佳比例.这种片上内存划分方法只适用于程序比较固定的嵌入式设备,对于运行程序种类繁多的设备则无效.

(3) 基于缓存的防护技术研究

CaSE 是弗吉尼亚理工学院暨州立大学为智能终端设计的抗物理攻击并且与移动操作系统隔离的安全解决方案[103],其利用 Cache-as-RAM 技术在 TEE(trusted execution environment)安全缓存内设计了包含数据段、代码段以及堆栈的安全运行环境,利用 TEE 提供软件隔离,利用加密和完整性校验机制保证片外 RAM 抵抗物理攻击;该团体同时发现具有 TEE 扩展的 ARM CPU 在 Cache 层面存在不一致现象,并利用该不一致现象成功在 Cache 中为 RootKit 构建执行环境 CacheKit,可躲避软硬件安全工具的检测.美国西北大学为嵌入式系统提出了一种基于软件的总线加密方案[104].该方案借助操作系统的分页机制实现了对应用程序的保护.应用程序的内存页被调用时,操作系统内核的页错误处理函数将该页从磁盘中读入片外 RAM,对该页解密,并利用缓存锁定技术将其锁定在缓存中.当该页需要被踢出内存时,内核先将其加密,然后再放入磁盘.EncExec 方案[105]预留一块专用缓存作为安全环境,然后制定如下规则来保护进程的机密数据:首先,预留所有映射到该缓存的物理内存页,修改内核请求分页系统将这些内存页只映射给进程机密数据;其次,同一时刻使用专用内存页的总量不得超过专用缓存,防止处理器因为缓存冲突将机密数据从缓存写回内存;最后,预留缓存必须被配置为 write-back 模式,保证处理器不会将修改的数据写回内存.上述规则保证在任意时刻进程的机密数据只在缓存中以明文形式存在,在片外 RAM 中则以密文形式存在.

(4) 基于寄存器的防护技术研究

美国莱斯大学提出使用寄存器在不可信操作系统中保护应用程序中敏感数据的方案 Ginseng[106].该方案的安全目标是敏感数据在运行时只能出现在寄存器中,并且使用它的函数一旦被切换,则函数切换上下文只会以密文形式存在于内存中,从而不给物理攻击者任何攻击时机.Ginseng 设计静态保护和运行时保护机制实现上述安全目标.静态保护实施在编译阶段,通过扩展编译器保证用户标记的敏感数据在编译后只存在于寄存器中,不会出现在内存中.运行时保护机制实施在敏感数据所在的函数,在函数调用或上下文切换需要将寄存器存储到内存中时,通过 ARM TrustZone 安全世界的一个安全服务将机密数据加密存储在安全堆栈上.希腊 FORTH 计算机科学研究所基于静、动态插桩技术对内存操作指令进行监控[107],然后以 XMM 寄存器作为中间存储对 CPU 加载/写回的内存进行解密/加密,从而实现对片外内存数据的机密性保护.该方案的缺点在于假设整个操

作系统内核以明文存储在片外内存中并且不被攻击,而物理攻击者完全可以攻破该假设.德国埃尔朗根-纽伦堡大学设计了一个可抵抗板级物理攻击的字节码解释器[108].该解释器被实现为一个简易栈图灵机,所有的指令操作对象都存储在栈上,同时该团队还设计了一个安全类C语言SCLL.在编译阶段,SCLL编译器调用抗物理攻击的密码算法TRESOR[64]对二进制程序进行加密;在运行阶段,解释器将内存中的指令、操作数解密到x86架构的AVX寄存器中,之后在寄存器中执行指令,最后将运行结果加密后存储到内存中,从而保证程序的指令和数据运行时只存储于AVX寄存器内,一旦离开CPU都以密文形式存在.

(5) 基于其他硬件机制的防护技术研究

上海交通大学的研究团队提出软硬件协同的VM安全保护方法Fidelius[109].Fidelius利用AMD SEV的硬件内存加密机制保护VM内存页,使其免受冷启动和总线监听等物理攻击.为防止恶意hypervisor通过操纵关键资源绕过内存保护机制,Fidelius通过不可旁路的内存隔离机制撤销了hypervisor对关键资源的访问权限,并提供3种不同的门机制来确保隔离的地址空间之间的转换,以对关键资源进行保护.此外,Fidelius通过复用SEV API实现对VM从启动到关闭的寄存器状态、运行时内存、I/O读写以及内存共享的全生命周期保护.SEVGuard[110]是一个基于AMD SEV构建的最小化的虚拟执行环境,旨在保护用户模式应用程序的机密性.该方法将加密的应用程序迁移到轻量级VM中,使用SEV加密其内存和寄存器状态,从而确保其代码和数据的机密性.SEVGuard引入一个主机调用接口,该接口将执行流重定向到主机,以使加密的应用程序可以使用不可信的主机功能,如系统调用和库调用.由于重定向机制需要进行上下文切换,SEVGuard对I/O密集型任务的性能影响大于对CPU密集型任务的性能影响.

3.2.3 小 结

应用程序抗物理攻击防御方案主要针对系统休眠时和系统运行时这两种时刻的攻击.系统休眠时的防御方案主要目的是解决各种流行的操作系统缺乏内存保护的缺陷:Deadbolt和Sentry方案针对的是安卓系统,德国Fraunhofer协会应用集成信息安全研究所的方案[92,93]针对的是Linux操作系统,Hypnoguard在虚拟层解决的是虚拟机的休眠时内存安全.在系统运行时防御方面,最初的方案是基于滑动窗口的思路,只有部分内存页以明文状态存在,其他内存页被加密保护以减少攻击者可获得的信息.但是这种思路只能缓解板级物理攻击,无法全面抵抗这种攻击.为此,研究人员进一步提出利用芯片内部的存储资源,包括OCM、Cache和寄存器,构建安全执行环境的思路,基于该思路出现了各种方案:CryptMe、MemVault、ZeroKernel、CaSE和Ginseng等.但是这些方案的缺点在于大部分受限于片内存储的容量,只能运行有限大小的程序.值得注意的是,CryptMe方案采用了分页调度机制,可以将应用程序暂时不使用的内存页调度到片外RAM中,从而能够运行体积较大的程序.该方案出现了全盘加密的影子,但是没有对OS内核层进行加密保护,因此在理论上仍存在被攻击的风险.基于其他处理器硬件机制保护应用程序的方案虽然不受片内存储容量的限制,但是实际上其采用的硬件机制不属于处理器通用资源,因此这类方案只能应用在特定处理器架构中.

3.3 全系统加密技术研究

3.3.1 密码算法及应用程序保护方案的安全隐患

虽然第3.1节和第3.2节描述的研究工作能够抵抗冷启动攻击等特定类型的攻击,但是理论上他们无法全面抵抗板级物理攻击.密钥及密码算法安全保障方面的工作仅仅保护了密钥及密码算法这类高安全要求的数据和代码,但是没有保护整个内存空间,物理攻击者仍然可以对片外RAM实施攻击,包括获取应用程序的机密数据、篡改代码等.应用程序保护方面的工作仅保护了应用层,没有考虑内核遭受物理攻击的情况.另外,这类工作局限于片上内存容量,只能保护有限大小的程序,而实际的操作系统通常占用几兆甚至上百兆的地址空间,远远超出片上内存的容量,所以这类工作无法直接应用到整个操作系统的保护.

综上所述,前文介绍的方案均无法对整个操作系统进行保护,所以必然有一部分操作系统内核数据和代码以明文形式存放在片外RAM中,攻击者完全可以通过板级物理攻击篡改这些数据和代码,CPU一旦运行被篡改的代码即可被攻击者获取内核的控制权,并且如果页表、中断向量表、内核栈、中断处理函数等CPU运行所必需的基本组件没有被保护,攻击者可利用这些组件加速攻击,一个典型的攻击实例如图4所示:CPU读取片

上内存中的指令时,需要读取页表获得指令虚拟地址对应的物理地址,攻击者可以篡改页表中映射该虚拟地址的 Entry,将该虚拟地址映射到已经植入恶意代码的内存,从而获得系统的控制权.虽然 CaSE 方案[103]提出将被保护程序的地址映射锁定在 TLB 中以防止攻击者篡改的思路,但是一般来说操作系统的地址空间远远大于 TLB 所能保护的地址空间,因此同样存在由于部分地址空间没有被保护而受到攻击的缺陷,譬如以下攻击:(1) 一旦 CPU 执行片外 RAM 中的指令,攻击者即通过总线向 CPU 注入清空 TLB 指令,使得锁定的 TLB 失效,而 CPU 再次建立地址映射时页表已经被攻击者篡改;(2) 如果中断处理代码没有受到完整的保护,攻击者可以向 CPU 发起硬件中断,将未被保护的中断处理代码(即位于片外 RAM 中的代码)篡改为清空 TLB 指令或其他恶意代码来控制系统.

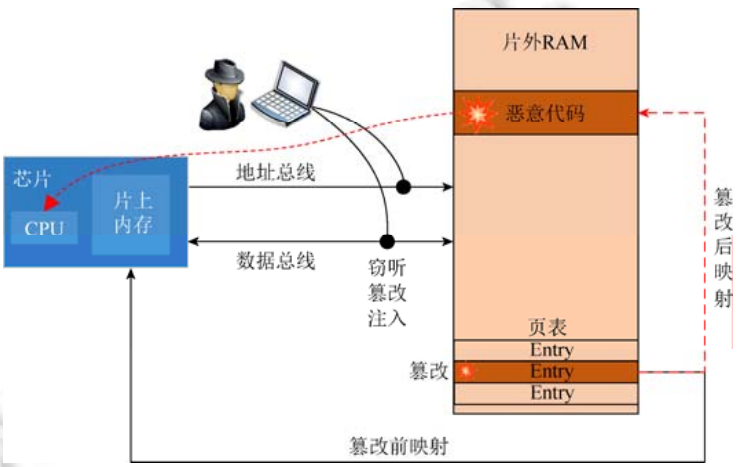

Fig.4 An example of board level physical attacks

图 4 板级物理攻击实例

第 3.1 节和第 3.2 节的密码算法保障和应用程序保护类工作无法从根本上抵抗板级物理攻击的主要原因在于防御组件无法对其所在的地址空间进行整体保护(如果防御组件实现为内核模块,那么其所在地址空间就是内核地址空间).虽然部分方案提出将所有的代码特别是管理代码全部放入片内存储的思路,但是图 4 描述的攻击表明这种思路对实际的操作系统内核是不可行的:由于操作系统内核体积较大无法全部放入片内存储,攻击者可以攻击未受保护的那些代码和数据,并且如果防御组件缺乏对页表、中断处理等关键数据和代码的保护则会加速攻击过程,而这恰恰是当前众多方案所缺乏的.针对该安全问题,学术界提出了全系统加密技术,该技术的思路是对防御组件所在的地址空间进行整体防护.一般来说防御组件位于内核层,因此这类工作基本上对操作系统内核进行了整体防护,并向上对应用层内存也进行了加密和完整性保护.根据技术实现思路的不同,全系统加密技术可分为轻量级操作系统全系统加密和基于请求分页系统的全系统加密两类.

3.3.2 轻量级操作系统全系统加密

这类方案面向的是实时操作系统、嵌入式操作系统等轻量级操作系统.这类操作系统运行所需工作内存较小,完全可以整体放入片上内存中来抵抗物理攻击.在此基础上,内核为应用程序提供物理防护机制.

美国达特茅斯学院的研究团队设计并实现了一个运行在片上内存的轻量级操作系统 Bear OS[111]来抵抗板级物理攻击.该操作系统内核使用微内核架构来减少自身 TCB 以便运行在容量有限的片上内存中.Bear OS 限定进程只能运行在片上内存中,并设计了进程静态加密和动态加密两种进程保护方式.进程静态加密的技术思路是只将进程的二进制镜像加密,进程的数据段,包括 heap、stack、data 等,直接在片上内存中分配,内核在加载进程时直接从磁盘中读取加密镜像,解密后放入片上内存.静态加密方式在进程初次运行时整体加载至片上内存,对进程性能影响较小,但是对片上内存容量要求较大,需要容纳整个进程空间.动态加密方式将进程的不同段分别进行加密,包括 code、data、heap、stack 等段都分别加密,在进程运行过程中需要哪个段就将该段解密

加载到片上内存,如果片上内存耗尽,那么就将一些暂不需要的段加密暂时备份到片外 RAM 中.这种加载方式会降低进程运行速度,但是适用于进程较多并且对工作内存需求较大的场景.

国内首都师范大学的研究团队针对实时操作系统对软件隔离和物理安全的要求,基于 ARM TrustZone 和内存加密技术提出实时任务的物理防护方法 SoftME[112,113].该方法在 TrustZone 的安全世界运行实时操作系统 FMP[114]来处理实时任务,并将该实时操作系统运行在片上内存中以抵抗物理攻击.SoftME 在实时操作系统内核层增加了一个任务调度器和一个内存保护引擎,其中任务调度器负责安全实时任务的调度,内存加密引擎负责安全实时任务在片上内存和片外 RAM 之间的调度:当某个任务需要从片上内存切换到片外 RAM 时,内存保护引擎对该任务进行加密并保护数据的完整性;当某个任务需要从片外 RAM 加载到片上内存运行时,内存保护引擎会对其进行解密并执行完整性检查.为保证普通世界与安全世界之间的通信,SoftME 设计了共享内存机制来传递信息.

3.3.3 基于请求分页系统的全系统加密

除了一些简单的实时操作系统和嵌入式操作系统,大部分成熟的操作系统运行所需的内存都比片上内存大,因此无法像轻量级操作系统那样直接加载到片上内存中来抵抗物理攻击.针对该问题,学术界提出了基于请求分页系统的全系统加密技术.

德国埃尔朗根-纽伦堡大学基于 TreVisor[89]系统实现了对虚拟机操作系统及上层应用的全系统加密方案 HyperCrypt[115].该方案利用 x86 架构的二级页表(second layer address translation,简称 SLAT)在虚拟机物理内存地址和实际物理内存地址转换过程中实现了请求分页系统,然后利用 TreVisor 提供的能够抵抗物理攻击的加密引擎,在请求分页系统中实现了对虚拟机物理内存页的加密保护:虚拟机访问一个内存页时,HyperCrypt 对该页进行解密放入滑动窗口,如果滑动窗口已满就加密一个不用的内存页以将其踢出滑动窗口.虽然该方案能够保证虚拟机的绝大部分内存都处于加密状态,但是滑动窗口仍然暴露给物理攻击者,因此不能算是严格的全系统加密方案.德国 Fraunhofer 协会的应用集成信息安全研究所几乎同时提出了与 HyperCrypt 类似的方案 TransCrypt[116].TransCrypt 面向 ARM 架构设计了一个轻量级 hypervisor,基于硬件虚拟化机制实现了请求分页系统,同样保证只有虚拟机的部分内存页处于明文状态,其他大部分内存被加密保护.与 HyperCrypt 不同的是,TransCrypt 实现了一个动态的 DMA 内存页检测方法以抵抗 DMA 攻击,另外该方案还提供多核支持和动态的滑动窗口支持.

中国科学院软件研究所针对 ARM 平台的 TEE 操作系统不能抵抗物理攻击的安全弱点,设计了一种新型的 TEE 操作系统架构 Minimal Kernel[49].该架构针对大部分 TEE 操作系统镜像较大而片上内存太小的难题,将 TEE 操作系统划分为两部分:一部分只包含维持 CPU 运行软件所必须的核心组件,称为最小核;其余操作系统组件则由最小核调度运行.为减小最小核的代码量,该方案提出一种构建最小化请求分页系统的原则,基于该原则确定了最小化请求分页系统所必须包含的内核组件(页表、异常向量表、低层异常处理函数、栈以及这些组件所依赖的组件).为了抵抗板级物理攻击,最小核运行在片上内存中,其余组件运行在片外 RAM 中,并由最小核提供机密性和完整性保护.实验表明该方案构建的最小核只需要 100KB,可运行在大部分 ARM 平台的片上内存中.该团队后续为 ARM CPU 架构提出了高安全 Enclave 架构 SecTEE[52],为 ARM 平台提供了抗板级物理攻击和内存侧信道攻击等现代高威胁性攻击手段的安全能力.SecTEE 在 Minimal Kernel 的抗板级物理攻击基础之上,扩展了抗内存侧信道的安全能力和可信计算特性:基于 Page coloring 机制和缓存锁定机制将安全敏感应用固定在不受外部攻击程序影响的缓存上,抵抗基于页表的侧信道和缓存侧信道攻击;并实现可信度量、远程证明、数据绑定等核心可信计算机制,为用户敏感应用提供高安全并且可证明的可信执行环境.SecTEE 以软件方式为 ARM CPU 提供了一种与 Intel SGX 方案同等安全水平的安全解决方案,适用于当前主流的 ARM 设备,具有很好的实际意义和应用前景.Minimal Kernel 和 SecTEE 实现了完整的全系统加密,具有很高的安全性,但是其带来的系统负载也很高:一般来说 SecTEE 带来约 2 倍以上的性能负载,而对于一些数据密集型程序带来的负载可能达几十倍.

3.3.4 小 结

从严格意义上来说,密钥及密码算法保障方案和应用程序保护方案仅仅保护了整个软件系统中很少部分的软件,主要是关键密码算法以及部分应用程序,没有能力提供对系统软件(譬如操作系统内核)的保护.而系统软件是所有应用程序的 TCB,如果 TCB 得不到保护,那么攻击者在攻破 TCB 之后就可以完全控制整个系统,这也是全系统加密技术产生的动机.但是目前全系统加密技术要么只能保护轻量级操作系统,要么会对系统造成极大的性能负载.以 Bear OS 和 SoftME 方案为例,这些方案都只能保护最多几百 KB 大小的轻量级操作系统,无法应用于成熟的大型操作系统.HyperCrypt、TransCrypt、Minimal Kernel 和 SecTEE 等方案虽然能够对成熟的操作系统进行加密保护,但这些方案给系统带来了相当大的负载,譬如 Minimal Kernel 方案对一些内存操作密集型的应用带来的负载能够达到 50 倍.如此高的性能负载极大的限制了这些方案的实际应用,一个比较好的解决方法是结合硬件密码加速器降低内存保护带来的计算负载.另外,全系统加密技术离不开密钥的支持,因此密钥的安全性是这类方案的安全基础之一.密钥的安全性问题可以通过两种方法来解决.第 1 种方法是利用第 3.1 节所述的密钥及密码算法安全保障技术来保护密钥的安全性,这种解决思路已经在 TreVisor 和 RamCrypt 等方案中有所体现.第 2 种方法是利用设备的硬件安全密钥,譬如移动设备普遍配备了设备根密钥,可以基于该密钥派生或保护内存加密所需要的密钥,譬如 Minimal Kernel 和 SecTEE 等全系统加密方案使用设备硬件密钥作为信任根,派生或保护内存加密、身份认证和数据加密等功能所需要的密钥.

4 技术讨论与工程应用建议

本节讨论当前软件方式的抗板级物理攻击技术的优势与不足,并给出在实际工程中部署该技术的建议.

4.1 技术优势

软件方式的抗板级物理攻击技术主要有如下几方面的技术优势:首先,不需要专有的硬件安全资源;其次,其部署不受硬件平台类型限制,可广泛部署在服务器、智能终端和嵌入式设备等各种平台上;最后,能够达到安全性很高的物理攻击防御能力,甚至能够达到专有硬件保护引擎的安全水平.

不需要专有的硬件安全资源.基于硬件的物理防护方案一般在处理器上扩展硬件安全组件对片外内存进行机密性和完整性保护,譬如 Intel SGX 的内存加密引擎[117].但是该类硬件安全组件目前主要应用在服务器设备,智能终端、嵌入式等设备均没有该类技术.而软件方式的抗板级物理攻击技术只依赖处理器的寄存器和片上内存等处理器普遍具备的硬件资源,然后通过软件内存加密技术实现对片外内存的保护,为不具备硬件加密引擎但是对物理防护有需求的应用场景提供了一种可行的解决方法.此外,部署这些方案无需更换硬件,只需改造系统软件即可,极大的降低了硬件成本.另外,软件的可扩展性和灵活性均高于硬件,一旦出现安全问题,能够比较容易的通过打补丁的方式进行更新.

不受平台限制.软件方式的抗板级物理攻击方法因为仅使用处理器芯片内普遍配备的资源,所以可以灵活的部署在各种平台上.TreVisor、HyperCrypt 等方案可以应用在云平台上,为虚拟机提供保护;AESSE、TRESOR、Loop-Amnesia 等方案可以应用于个人电脑;Minimal Kernel、SecTEE 等方案可以应用于智能终端;SoftME、Bear OS 等方案可以应用于低端嵌入式设备.因此,软件方式的抗物理攻击防护方案几乎涵盖了所有类型的计算平台,能够为绝大多数的设备提供物理防护能力.

能达到很高的物理安全水平.虽然部分方案的威胁模型只涵盖某种类型的板级物理攻击(譬如只考虑冷启动攻击)或者从效率方面出发,没有考虑全面的板级物理攻击能力,只对密钥、密码算法或应用程序等部分代码和数据提供了物理防护,但是实际上软件方式的抗板级物理攻击防御技术完全能够全面抵抗冷启动攻击、总线攻击和 DMA 攻击,达到与硬件方式相同的安全能力.中科院软件所研究团队的 Minimal Kernel 和 SecTEE 方案达到了与 Intel SGX 相同的安全水平.此外,SecTEE 方案还增加了内存侧信道防护机制,而 Intel SGX 的内存侧信道防护问题至今仍未完全解决,从这方面也可以看出软件方式的灵活性.

4.2 技术不足之处

大部分方案只保护部分关键代码,不能提供全系统加密.目前绝大多数方案主要是保护应用程序的关键数据,没有考虑对操作系统内核的保护.实际上,本文第 3.3.1 节已经指出,如果攻击者具备全方面的电路板物理操控能力,在没有对内核进行保护的情况下,其完全有能力从总线获取内核的内存镜像甚至让处理器运行内核态的恶意代码.攻击者一旦攻破内核,也就具备控制整个系统的能力,从而使得应用层的保护没有意义.这也是为什么大部分方案的威胁模型只考虑部分板级物理攻击,主要是冷启动攻击,而不考虑总线攻击,或者仅考虑被动式总线窃听攻击.然而总线攻击是板级物理攻击的一个典型案例,以 MIT 团队攻击 Xbox[7]为例,其所用到的硬件成本仅几十美元,是一种非常严重并且必须考虑的实际威胁.

对系统性能影响较大.软件方式的内存保护需要使用软件加密引擎来对片外内存进行加密,而软件加密引擎会占用处理器资源,从而影响普通程序的运行效率.另外,很多方案基于寄存器实现,中间计算结果存储在寄存器中,为避免进行上下文切换将中间计算结果切换到不安全的内存中,每次调用必须是原子操作并且占用时间不能太长以免影响用户输入响应,这种方式会造成频繁的关中断和上下文切换操作,从而降低系统性能.通过统计各种方案的实验结果,发现大部分方案都给系统带来了 2 倍以上的性能负载,而 Minimal Kernel 等全系统加密方案甚至给部分应用带来几十倍的性能负载.软件加密带来的负载可以通过一些硬件密码加速器来缓解,譬如 TRESOR 方案使用 x86 架构提供的硬件加密指令 AES-NI 来降低内存加密带来的性能负载.

影响部分专有功能的运行.一些方案占用了处理器上一些具有专门用途的硬件资源,譬如 AESSE 方案占用了用于多媒体计算的 SSE 寄存器,TRESOR 方案占用了用于程序调试的断点寄存器,CaSE 方案占用了用于提高内存读写速度的缓存等等.占用这些硬件资源必然会对相关功能的正常运行造成影响.

缺乏对隔离机制的支持.大部分方案将板级物理攻击防御组件实现为操作系统的一个内核模块,但是并没有与其他内核组件进行隔离,因此一旦攻击者攻破内核就完全有能力禁用防御组件.不但 Linux、Windows 这类大型操作系统经常被暴露出安全漏洞,像 TEE OS 这种专有的小型操作系统也被暴露出多个高危漏洞(CVE-2013-3051,CVE-2014-4322,CVE-2015-4422 等).如果软件实现的抗物理攻击组件没有与操作系统隔离,那么攻击者可以利用软件漏洞禁用抗物理攻击机制,这也是大部分方案都假设内核安全的原因.实际上,学术界已经提出在内核内部建立逻辑隔离地址空间的技术[53−56],并且该技术仅依赖处理器的内存管理等基本机制,可广泛应用在各种平台上.因此,增加隔离机制是增强当前方案安全性的一个可行思路.

4.3 工程应用建议

首先,对于像 Windows、Linux 这样的大型操作系统,建议将防御组件以内核模块的形式实现在内核层,然后向上层应用提供相应的安全服务.并且不建议这些操作系统使用全系统加密方案,因为以 Minimal Kernel 和 SecTEE 等全系统加密方案的性能评估结果来看,全系统加密带来的性能负载可达 2~50 多倍(主要与应用对内存的访问频率有关,对内存访问越频繁性能影响越大);而密码算法保障以及应用程序保护等非全系统加密方案带来的性能负载一般在 2 倍以内,最差情况也在 10 倍以内.大型操作系统对性能要求较高,而大部分应用的安全要求不需要达到防御物理攻击的水平,因此将防御组件实现为内核模块并只保障安全密切相关的应用即可.但是为了防止内核软件漏洞导致攻击者使用软件方式即破坏防御组件的安全问题,建议参考内核嵌套隔离地址空间技术[53−56]在内核层构建一个受保护的隔离地址空间,该技术通过独立页表、安全上下文切换、内核态代码去特权指令等安全机制保证其他内核组件无法访问隔离地址空间内部的代码和数据,并且只能通过预先定义好的接口调用隔离地址空间内部的安全服务,从而保证隔离地址空间与其他内核组件的逻辑隔离;然后在该隔离地址空间内运行可抵抗板级物理攻击的防御组件并向外部提供调用接口,应用层可调用该接口实现对关键数据的保护.

其次,对于安全性要求较高的小型操作系统,譬如 TEE OS 或安全密切相关的嵌入式 OS,建议使用全系统加密方案.因为 TEE OS 这类操作系统处理的主要是安全敏感业务,譬如密钥管理、身份认证、移动支付、数字版权保护和个人生物信息存储等,一旦信息泄露造成的损失较大.在这种高安全要求的背景下,为系统部署安全性

较高的全系统加密方案是必要的.对于体积特别小的嵌入式 OS,譬如只有几十 KB 或 100KB 左右的轻量级操作系统内核,可以将操作系统内核整体运行在片内存储中,然后根据需要将应用调度到片内存储中运行,一旦不需要运行即可将应用加密存储到片外内存中.对于体积较大的操作系统内核,可以借鉴 Minimal Kernel 方案,构建一个可调度内核以及应用的最小化请求分页系统,将该请求分页系统运行在片内存储中,对内核其他组件以及应用进行按需调度并实施加密和完整性保护.基于请求分页机制的全系统加密方案由于对内核等系统软件也进行了内存加密保护,给系统带来的性能负载较重,特别是对内存访问较频繁的应用带来的性能负载可能达 50 多倍.

最后,对于配备了硬件密码加速器的处理器,防御组件应尽可能使用这些加速器来进行密码操作,从而减少对主处理器的使用,降低方案带来的负载.但是,并不是所有的硬件密码加速都能提高软件加密的速度.法国帕莱索的研究所对飞思卡尔的密码加速模块 CAAM(cryptographic acceleration and assurance module)进行了性能测试[118],测试结果表明在保护数据较小时,譬如 1KB,ARM i.MX6Q 处理器进行完整性校验只需要 23μs,而 CAAM 需要 1 075μs;只有在保护较大数据(超过 100KB)时,CAAM 密码加速器才会比普通软件加密速度快:保护数据为 1MB 时,主处理器进行完整性校验需要 21 696μs,而 CAAM 只需要 13 008μs.这是因为 CAAM 使用 DMA 方式进行数据传输,在输入数据较小时,DMA 数据传输带来的延迟大大超过了其本身带来的加速.因此,最好采用类似 Intel AES-NI 这种 CPU 扩展指令方式的硬件密码加速机制.

5 研究展望

本节根据软件方式抗板级物理攻击防御技术的演进,对该技术的发展趋势和可能出现的研究点进行总结和讨论.

5.1 全系统加密或逻辑隔离的防御技术研究

随着物理攻击方法的提高,用户对计算机系统的安全要求也越来越高.而安全本身是一个木桶效应非常明显的领域,攻击者一般会从安全性最弱的组件实施攻击.对于计算机系统来说,如果某一部分组件没有被防护,那么攻击者会先攻击这一部分组件,获得系统权限后再去攻击防御组件.因此,为了保证安全方案实际可用,需要防御组件能够保护其所在的整个地址空间.

如果防御组件位于内核层,那么就需要对整个内核进行加密防御.但是从性能角度来看,对整个内核进行全系统加密只适用于小型、安全密切相关并且对性能要求不太高的操作系统,不适用于对性能要求较高的大型操作系统.对于大型操作系统,可以将嵌套内核隔离技术与软件加密相结合,设计与其他组件隔离的抗板级物理攻击防御技术,避免对整个内核进行加密带来的计算负载.这方面的工作在学术界还没有展开,但是能够解决软件方式抗板级物理攻击的短板,是一种实际可用的技术思路.

5.2 软硬件协同的安全防御技术

软件方式的抗板级物理攻击方法的一个技术劣势是性能低,这是软件内存加密造成的.解决该问题的常规思路是设计专有的硬件加密引擎或硬件密码加速器,将内存加密部分交给专门的硬件组件来处理,软件专门负责安全内存管理、中断管理、任务调度等系统软件功能.这种设计思路也符合现代计算机的发展趋势:图灵奖获得者 Hennessy 和 Patterson 共同提出未来计算机体系结构的一大发展趋势[119]就是为专有领域设计专有的处理器(domain-specific architecture).

软硬件协同设计在抗板级物理攻击方面具有优势的另外一个实例是现代 Secure Enclave 技术.现代计算机系统已经将抗板级物理攻击作为一项必备安全属性,而 Secure Enclave 技术的一个重要安全能力就是抵抗板级物理攻击.主流处理器都已经配备或开始研究 Secure Enclave 方案,譬如 Intel SGX、AMD SME、RISC-V Keystone[120],还有为 ARM 平台设计的学术界方案 Komodo[121]和 SecTEE 等.Komodo 方案设计者提出了一种软硬件协同设计方法:硬件只负责基本的硬件安全机制,包括内存加密和地址空间隔离等,软件利用这些基本的硬件安全机制实现 Secure Encalve 功能,包括内存管理、中断处理和可信计算等.这种设计方法的主要出发点是硬

件的可塑性较低,一旦出现问题难以修复,而软件具有很高的灵活性,可以比较容易的修复或增添特性.SecTEE方案的实验结果也证实了软件内存加密会增加很高的负载.类似于人工智能领域的专用深度学习处理器TPU[122]和寒武纪AI芯片[123−126],为Secure Enclave技术设计专用的安全处理器既符合体系结构领域的发展趋势,也是该技术的实际需求.

5.3 与侧信道结合的新攻击方法

以前内存侧信道攻击[127−137]主要基于页错误或缓存获取另外一个地址空间或隔离执行环境所运行程序的内存地址访问模式,然后基于内存地址访问模式推测被攻击程序的机密信息,譬如密钥或图像信息.由于内存的数据和代码都是经系统总线传输给处理器,其中地址总线传输地址信息,数据总线传输数据和代码,因此板级物理攻击完全有能力从总线上窃听处理器正在访问的数据或代码的地址,从而获得程序的内存地址访问模式.也就是说,攻击者完全有能力利用板级物理攻击实施侧信道攻击.该攻击方法由加州大学伯克利分校的研究团队提出并实现[10],已经成功对Intel SGX的Encalve实施了攻击.但是目前该攻击方法还处于初级阶段,只能对数据加载类型的、数据量大的程序有效,与依赖代码分支的侧信道(譬如RSA算法的侧信道攻击[138])结合的攻击效果还不明显,因为处理器会将经常执行的代码保留在缓存中,从总线上无法检测到该代码的地址信息.同样对于数据量小的程序,内存访问分布不均匀的程序,以及受缓存回写机制影响的程序,该方法的攻击能力还无法提供精确的内存地址访问模式,有待后续研究.由于该类攻击刚刚被提出,并且攻击能力还不完善,是一个非常值得关注并深入探索的研究点.

6 结束语

本文针对威胁日益严重的板级物理攻击,调研其威胁模型、相关技术和现实存在的攻击案例,重点对软件方式的、不需要修改处理器体系结构的抗板级物理攻击防御技术进行了详细深入的归纳总结,包括各种方案的技术原理、安全能力和保护范围等.本文还讨论了这种防御技术的优势及其目前存在的问题,并给出了在实际工程中应用该技术的建议,以便工程人员能够合理、正确的使用该技术.最后本文探讨了该技术今后的发展趋势和一些研究点,以供相关领域的研究人员参考.

References:

[1] Halderman JA, Schoen SD, Heninger N, Clarkson W, Paul W, Calandrino JA, Feldman AJ, Appelbaum J, Felten EW. Lest we remember: Cold-boot attacks on encryption keys. In: Proc. of the 17th USENIX Security Symp. (USENIX Security). USENIX Association, 2008. 45–60. [doi: 10.1145/1506409.1506429]

[2] Tilo M, Michael S, Freiling FC. FROST: Forensic recovery of scrambled telephones. In: Proc. of the 11th Int'l Conf. on Applied Cryptography and Network Security (ACNS). Berlin, Heidelberg: Springer-Verlag, 2013. 373–388. [doi: 10.1007/978-3-642-38980-1_23]

[3] Taubmann B, Huber M, Wessel S, Heim L, Reiser HP, Sigl G. A lightweight framework for cold boot based forensics on mobile devices. In: Proc. of the 10th Int'l Conf. on Availability, Reliability and Security (ARES). IEEE, 2015. 120–128. [doi: 10.1109/ARES.2015.47]

[4] Gruhn M, Müller T. On the practicability of cold boot attacks. In: Proc. of the 8th Int'l Conf. on Availability, Reliability and Security (ARES). IEEE, 2013. 390–397. [doi: 10.1109/ARES.2013.52]

[5] Bauer J, Gruhn M, Freiling FC. Lest we forget: Cold-boot attacks on scrambled DDR3 memory. Digital Investigation, 2016,16:S65–S74. [doi: 10.1016/j.diin.2016.01.009]

[6] Lawson N. How the PS3 hypervisor was hacked. 2010. https://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked

[7] Huang A. Keeping secrets in hardware: the Microsoft Xbox™ case study. In: Proc. of the 4th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES). Berlin, Heidelberg: Springer-Verlag, 2002. 213–227. [doi: 10.1007/3-540-36400-5_17]

[8] Kuhn MG. Cipher instruction search attack on the bus-encryption security microcontroller DS5002FP. IEEE Trans. on Computers, 1998,47(10):1153–1157. [doi: 10.1109/12.729797]

[9] NCC Group. TPM Genie: interposer attacks against the trusted platform module serial bus. 2018. https://www.nccgroup.trust/us/our-research/tpm-genie-interposer-attacks-against-the-trusted-platform-module-serial-bus

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

[11] Duflot L, Perez YA, Valadon G, Levillain A. Can you still trust your network card?. In: Proc. of the 2010 CanSecWest Conf. 2010. 24–26.

[12] Duflot L, Perez YA, Morin B. What if you can't trust your network card?. In: Proc. of the 14th Int'l Workshop on Recent Advances in Intrusion Detection (RAID). Berlin, Heidelberg: Springer-Verlag, 2011. 378–397. [doi: 10.1007/978-3-642-23644-0_20]

[13] Triulzi A. The Jedi Packet Trick takes over the Deathstar. In: Proc. of the 2010 CanSecWest Conf. 2010.

[14] Stewin P, Bystrov I. Understanding DMA malware. In: Proc. of the 9th Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA). Berlin, Heidelberg: Springer-Verlag, 2012. 21–41. [doi: 10.1007/978-3-642-37300-8_2]

[15] Tria A, Choukri H. Invasive attacks. In: Encyclopedia of Cryptography and Security. 2011. 623–629. [doi: 10.1007/978-1-4419-5906-5_511]

[16] Skorobogatov SP. Semi-invasive attacks: A new approach to hardware security analysis. Technical Report, UCAM-CL-TR-630, University of Cambridge, 2005.

[17] Lie D, Thekkath C, Mitchell M, Lincoln P, Boneh D, Mitchell J, Horowitz M. Architectural support for copy and tamper resistant software. In: Proc. of the 9th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, 2000. 168–177. [doi: 10.1145/356989.357005]

[18] Suh GE, Clarke D, Gassend B, Van Dijk M, Devadas S. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In: Proc. of the 17th Annual Int'l Conf. on Supercomputing (ICS). ACM, 2003. 160–171. [doi: 10.1145/782814.782838]

[19] Suh GE, Clarke D, Gassend B, Van Dijk M, Devadas S. Efficient memory integrity verification and encryption for secure processors. In: Proc. of the 36th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). IEEE, 2003. 339–350. [doi: 10.5555/956417.956575]

[20] Clarke D, Devadas S, Van Dijk M, Gassend B, Suh GE. Incremental multiset hash functions and their application to memory integrity checking. In: Proc. of the 9th Int'l Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT). Berlin, Heidelberg: Springer-Verlag, 2003. 188–207. [doi: 10.1007/978-3-540-40061-5_12]

[21] Lee RB, Kwan PCS, McGregor JP, Dwoskin J, Wang ZH. Architecture for protecting critical secrets in microprocessors. In: Proc. of the 32nd Int'l Symp. on Computer Architecture (ISCA). IEEE, 2005. 2–13. [doi: 10.1109/ISCA.2005.14]

[22] Shi W, Lee HHS, Ghosh M, Lu C. Architectural support for high speed protection of memory integrity and confidentiality in multiprocessor systems. In: Proc. of the 13th Int'l Conf. on Parallel Architecture and Compilation Techniques (PACT). IEEE, 2004. 123–134. [doi: 10.1109/PACT.2004.1342547]

[23] Zhang YT, Gao L, Yang J, Zhang XY, Gupta R. SENSS: Security enhancement to symmetric shared memory multiprocessors. In: Proc. of the 11th Int'l Symp. on High-performance Computer Architecture (HPCA). IEEE, 2005. 352–362. [doi:10.1109/HPCA.2005.31]

[24] Rogers B, Solihin Y, Prvulovic M. Memory predecryption: hiding the latency overhead of memory encryption. ACM SIGARCH Computer Architecture News, 2005,33(1):27–33. [doi: 10.1145/1055626.1055631]

[25] Yang J, Gao L, Zhang YT. Improving memory encryption performance in secure processors. IEEE Trans. on Computers, 2005,54(5):630–640. [doi: 10.1109/TC.2005.80]

[26] Yan CY, Englender D, Prvulovic M, Rogers B, Sokihin Y. Improving cost, performance, and security of memory encryption and authentication. In: Proc. of the 33rd Annual Int'l Symp. on Computer Architecture (ISCA). IEEE, 2006. 179–190. [doi:10.1109/ISCA.2006.22]

[27] Duc G, Keryell R. CryptoPage: An efficient secure architecture with memory encryption, integrity and information leakage protection. In: Proc. of the 22nd Annual Computer Security Applications Conf. (ACSAC). IEEE, 2006. 483–492. [doi:10.1109/ACSAC.2006.21]

[28] Rogers B, Prvulovic M, Solihin Y. Efficient data protection for distributed shared memory multiprocessors. In: Proc. of the 15th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT). ACM, 2006. 84–94. [doi: 10.1145/1152154.1152170]

[29] Rogers B, Chhabra S, Prvulovic M, Solihin Y. Using address independent seed encryption and bonsai merkle trees to make secure processors OS- and performance-friendly. In: Proc. of the 40th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). IEEE, 2007. 183–196. [doi: 10.1109/MICRO.2007.16]

[30] Vaslin R, Gogniat G, Diguet JP, Tessier R, Burleson W. Low latency solution for confidentiality and integrity checking in embedded systems with off-chip memory. In: Proc. of the 3rd Int'l Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC). 2007. 146–153.

[31] Lee M, Ahn M, Kim EJ. I2SEMS: Interconnects-independent security enhanced shared memory multiprocessor systems. In: Proc. of the 16th Int'l Conf. on Parallel Architecture and Compilation Techniques (PACT). IEEE, 2007. 94–103. [doi: 10.1109/PACT.2007.4336203]

[32] Elbaz R, Champagne D, Lee RB, Torres L, Sassatelli G, Guillemin P. TEC-Tree: A low-cost, parallelizable tree for efficient defense against memory replay attacks. In: Proc. of the 9th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES). 2007. 289–302. [doi: 10.1007/978-3-540-74735-2_20]

[33] Su LF, Courcambeck S, Guillemin P, Schwarz C, Pacalet R. SecBus: Operating system controlled hierarchical page-based memory bus protection. In: Proc. of the 12th Design, Automation and Test in Europe. IEEE, 2009. 570–573. [doi: 10.1109/DATE.2009.5090729]

[34] Enck W, Butler K, Richardson T, McDaniel P, Smith A. Defending against attacks on main memory persistence. In: Proc. of the 24th Annual Computer Security Applications Conf. (ACSAC). IEEE, 2008. 65–74. [doi: 10.1109/ACSAC.2008.45]

[35] Rogers B, Yan CY, Chhabra S, Prvulovic M, Solihin Y. Single-level integrity and confidentiality protection for distributed shared memory multiprocessors. In: Proc. of the 14th Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2008. 161–172. [doi: 10.1109/HPCA.2008.4658636]

[36] Champagne D, Elbaz R, Lee RB. The reduced address space (RAS) for application memory authentication. In: Proc. of the 11th Int'l Conf. on Information Security (ISC). Berlin, Heidelberg: Springer-Verlag, 2008. 47–63. [doi: 10.1007/978-3-540-85886-7_4]

[37] Vig S, Juneja R, Jiang GY, Lam SK, Ou CH. Framework for fast memory authentication using dynamically skewed integrity tree. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, 2019,27(10):2331–2343. [doi: 10.1109/TVLSI.2019.2923004]

[38] Chhabra S, Rogers B, Solihin Y, Prvulovic M. SecureME: A hardware-software approach to full system security. In: Proc. of the 25th Int'l Conf. on Supercomputing (ICS). ACM, 2011. 108–119. [doi: 10.1145/1995896.1995914]

[39] Chhabra S, Solihin Y. i-NVMM: A secure non-volatile main memory system with incremental encryption. In: Proc. of the 38th Annual Int'l Symp. on Computer Architecture (ISCA). IEEE, 2011. 177–188. [doi: 10.1145/2000064.2000086]

[40] Elbaz R, Champagne D, Gebotys C, Lee RB, Potlapally N, Torres L. Hardware mechanisms for memory authentication: A survey of existing techniques and engines. Trans. on Computational Science IV. Berlin, Heidelberg: Springer-Verlag, 2009. 1–22. [doi: 10.1007/978-3-642-01004-0_1]

[41] Henson M, Taylor S. Memory encryption: A survey of existing techniques. ACM Computing Surveys, 2014,46(4):53:1–53:26. [doi: 10.1145/2566673]

[42] McKeen F, Alexandrovich I, Berenzon A, Rozas CV, Shafi H, Shanbhogue V, Savagaonkar UR. Innovative instructions and software model for isolated execution. In: Proc. of the 2nd Int'l Workshop on Hardware and Architectural Support for Security and Privacy (HASP). ACM, 2013. 10. [doi: 10.1145/2487726.2488368]

[43] Anati I, Gueron S, Johnson SP, Scarlata VR. Innovative technology for CPU based attestation and sealing. In: Proc. of the 2nd Int'l Workshop on Hardware and Architectural Support for Security and Privacy (HASP). ACM, 2013. 13.

[44] Kaplan D, Powell J, Woller T. AMD memory encryption. White paper. Advanced Micro Devices, Inc., 2016.

[45] Lin JQ, Luo B, Guan L, Jing JW. Secure computing using registers and caches: The problem, challenges, and solutions. IEEE Security & Privacy, 2016,14(6):63–70. [doi: 10.1109/MSP.2016.130]

[46] Huo WJ. Research and design of secure run-time mechanism for embedded processor [Ph.D. Thesis]. Wuhan: Huazhong University of Science and Technology, 2010 (in Chinese with English abstract).

[47] GlobalPlatform Device Committee. TEE protection profile, version 1.2.1. 2016. https://globalplatform.org/specs-library/tee-protection-profile-v1-2-1

[48] Gutmann P. Data remanence in semiconductor devices. In: Proc. of the 10th USENIX Security Symp. (USENIX Security). USENIX Association, 2001. 39–54. [doi: 10.5555/1251327.1251331]

[49] Zhao SJ, Zhang QY, Qin Y, Feng W, Feng DG. Minimal kernel: An operating system architecture for TEE to resist board level physical attacks. In: Proc. of the 22nd Int'l Symp. on Research in Attacks, Intrusions and Defenses (RAID). USENIX Association, 2019. 105–120.

[50] Colp P, Zhang JW, Gleeson J, Suneja S, De Lara E, Raj H, Saroiu S, Wolman A. Protecting data on smartphones and tablets from memory attacks. In: Proc. of the 20th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, 2015. 177–189. [doi: 10.1145/2694344.2694380]

[51] Zhao SJ, Zhang QY, Hu GY, Qin Y, Feng DG. Providing root of trust for ARM TrustZone using on-chip SRAM. In: Proc. of the 4th Int'l Workshop on Trustworthy Embedded Devices (TrustED). ACM, 2014. 25–36. [doi: 10.1145/2666141.2666145]

[52] Zhao SJ, Zhang QY, Qin Y, Feng W, Feng DG. SecTEE: A software-based approach to secure enclave architecture using TEE. In: Proc. of the 26th ACM SIGSAC Conf. on Computer and Communications Security (CCS). ACM, 2019. 1723–1740. [doi: 10.1145/3319535.3363205]

[53] Dautenhahn N, Kasampalis T, Dietz W, Criswell J, Adve V. Nested kernel: An operating system architecture for intra-kernel privilege separation. In: Proc. of the 20th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, 2015. 191–206. [doi: 10.1145/2775054.2694386]

[54] Azab AM, Swidowski K, Bhutkar R, Ma J, Shen WB, Wang RW, Ning P. SKEE: A lightweight secure kernel-level execution environment for ARM. In: Proc. of the 23rd Annual Network and Distributed System Security Symp. (NDSS). Internet Society, 2016. [doi: 10.14722/ndss.2016.23009]

[55] Cho Y, Kwon D, Yi H, Peak Y. Dynamic virtual address range adjustment for intra-level privilege separation on ARM. In: Proc. of the 24th Annual Network and Distributed System Security Symp. (NDSS). Internet Society, 2017. [doi: 10.14722/ndss.2017.23024]

[56] Li WH, Xia YB, Lu L, Chen HB, Zang BY. TEEv: Virtualizing trusted execution environments on mobile platforms. In: Proc. of the 15th ACM SIGPLAN/SIGOPS Int'l Conf. on Virtual Execution Environments (VEE). ACM, 2019. 2–16. [doi: 10.1145/3313808.3313810]

[57] Gassend B, Suh GE, Clarke D, Van Dijk M, Devadas S. Caches and hash trees for efficient memory integrity verification. In: Proc. of the 9th Int'l Symp. on High-performance Computer Architecture (HPCA). IEEE, 2003. 295–306. [doi: 10.1109/HPCA.2003.1183547]

[58] Heninger N, Shacham H. Reconstructing RSA private keys from random key bits. In: Proc. of the 29th Annual Int'l Cryptology Conf. (CRYPTO). Berlin, Heidelberg: Springer-Verlag, 2009. 1–17. [doi: 10.1007/978-3-642-03356-8_1]

[59] Piegdon DR, Pimenidis L. Targeting physically addressable memory. In: Proc. of the 4th Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA). Berlin, Heidelberg: Springer-Verlag, 2007. 193–212. [doi: 10.1007/978-3-540-73614-1_12]

[60] Parker TP, Xu SH. A method for safekeeping cryptographic keys from memory disclosure attacks. In: Proc. of the 1st Int'l Conf. on Trusted Systems (INTRUST). Berlin, Heidelberg: Springer-Verlag, 2009. 39–59. [doi: 10.1007/978-3-642-14597-1_3]

[61] Li CW, Lin JQ, Cai QW, Luo B. Peapods: OS-independent memory confidentiality for cryptographic engines. In: Proc. of the 16th IEEE Int'l Conf. on Parallel and Distributed Processing with Applications (ISPA). IEEE, 2018. 862–869. [doi: 10.1109/BDCloud.2018.00128]

[62] Intel Corporation. Intel® 64 and IA-32 architectures software developer's manual—Volume 1: Basic architecture. 2019. https://software.intel.com/sites/default/files/managed/a4/60/253665-sdm-vol-1.pdf

[63] Müller T, Dewald A, Freiling FC. AESSE: A cold-boot resistant implementation of AES. In: Proc. of the 3rd European Workshop on System Security (EUROSEC). ACM, 2010. 42–47. [doi: 10.1145/1752046.1752053]

[64] Müller T, Freiling FC, Dewald A. TRESOR runs encryption securely outside RAM. In: Proc. of the 20th USENIX Security Symp. (USENIX Security). USENIX Association, 2011. 251–266. [doi: 10.5555/2028067.2028084]

[65] Simmons P. Security through Amnesia: a software-based solution to the cold boot attack on disk encryption. In: Proc. of the 27th Annual Computer Security Applications Conf. (ACSAC). ACM, 2011. 73–82. [doi: 10.1145/2076732.2076743]

[66] Pabel J. FrozenCache: Mitigating cold-boot attacks for full-disk-encryption software. In: Proc. of the 27th Chaos Communication Congress (CCC). 2010.

[67] Götzfried J, Müller T. ARMORED: CPU-bound encryption for Android-driven ARM devices. In: Proc. of the 8th Int'l Conf. on Availability, Reliability and Security (ARES). IEEE, 2013. 161–168. [doi: 10.1109/ARES.2013.23]

[68] Götzfried J, Müller T. Analysing Android's full disk encryption feature. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 2014,5(1):84–100. [doi: 10.22667/JOWUA.2014.03.31.084]

[69] Nilsson A, Andersson M, Axelsson S. Key-hiding on the ARM platform. Digital Investigation, 2014,11:S63–S67. [doi: 10.1016/j.diin.2014.03.008]

[70] Zhang XS, Tan YA, Xue Y, Zhang QX, Li YZ, Zhang C, Zheng J. Cryptographic key protection against FROST for mobile devices. Cluster Computing, 2017,20(3):2393–2402. [doi: 10.1007/s10586-016-0721-3]

[71] Garmany B, Müller T. PRIME: Private RSA infrastructure for memory-less encryption. In: Proc. of the 29th Annual Computer Security Applications Conf. (ACSAC). ACM, 2013. 149–158. [doi: 10.1145/2523649.2523656]

[72] Yang Y, Guan Z, Liu Z, Chen Z. Protecting elliptic curve cryptography against memory disclosure attacks. In: Proc. of the 16th Int'l Conf. on Information and Communications Security (ICICS). Cham: Springer-Verlag, 2014. 49–60. [doi: 10.1007/978-3-319-21966-0_4]

[73] Zhao Y, Lin JQ, Pan WQ, Xue C, Zheng FY, Ma ZQ. RegRSA: Using registers as buffers to resist memory disclosure attacks. In: Proc. of the 31st IFIP Int'l Conf. on ICT Systems Security and Privacy Protection (SEC). Cham: Springer-Verlag, 2016. 293–307. [doi: 10.1007/978-3-319-33630-5_20]

[74] Guan L, Lin JQ, Luo B, Jing JW. Copker: Computing with private keys without RAM. In: Proc. of the 21st Annual Network and Distributed System Security Symp. (NDSS). Internet Society, 2014. [doi: 10.14722/ndss.2014.23125]

[75] Guan L, Lin JQ, Luo B, Jing JW, Wang J. Protecting private keys against memory disclosure attacks using hardware transactional memory. In: Proc. of the 36th IEEE Symp. on Security and Privacy (SP). IEEE, 2015. 3–19. [doi: 10.1109/SP.2015.8]

[76] Vasiliadis G, Athanasopoulos E, Polychronakis M, Ioannidis S. PixelVault: Using GPUs for securing cryptographic operations. In: Proc. of the 21st ACM SIGSAC Conf. on Computer and Communications Security (CCS). ACM, 2014. 1131–1142. [doi: 10.1145/2660267.2660316]

[77] Gueron S. Intel® Advanced Encryption Standard (AES) new instructions set. Intel Corporation. 2010. https://www.intel.com.bo/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

[78] Gueron S. Intel's new AES instructions for enhanced performance and security. In: Proc. of the 16th Int'l Workshop on Fast Software Encryption (FSE). Berlin, Heidelberg: Springer-Verlag, 2009. 51–66. [doi: 10.1007/978-3-642-03317-9_4]

[79] Ruusu J. loop-AES—file system and swap encryption package. LinuxLinks. 2018. https://www.linuxlinks.com/loop-aes

[80] Blass EO, Robertson W. TRESOR-HUNT: attacking CPU-bound encryption. In: Proc. of the 28th Annual Computer Security Applications Conf. (ACSAC). ACM, 2012. 71–78. [doi: 10.1145/2420950.2420961]

[81] ARM Ltd. Neon. 2020. https://developer.arm.com/architectures/instruction-sets/simd-isas/neon

[82] Kaliski B. TWIRL and RSA key size. RSA Laboratories. 2003. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.4447&rep=rep1&type=pdf

[83] Barker EB, Barker WC, Burr WE, Polk WT, Smid M. NIST special publication 800-57 recommendation for key management - Part 1: General (revised). National Institute of Standards and Technology. 2006. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r2006.pdf

[84] Intel Corporation. Intel® advanced vector extensions programming reference. 2011. https://software.intel.com/sites/default/files/4f/5b/36945

[85] Mittal M, Peleg A, Weiser U. MMX™ technology architecture overview. Intel Corporation. 1996. https://software.intel.com/sites/default/files/m/d/3/0/MMX_Manual_Tech_Overview.pdf

[86] Intel Corporation. Intel® architecture instruction set extensions programming reference—Chapter 8: Intel® transactional synchronization extensions. 2012. https://software.intel.com/sites/default/files/m/9/2/3/41604

[87] Zhu ZT, Kim S, Rozhanski Y, Hu YG, Witchel E, Silberstein M. Understanding the security of discrete GPUs. In: Proc. of the 10th Workshop on General Purpose Processing Using GPUs (GPGPU). ACM, 2017. 1–11. [doi: 10.1145/3038228.3038233]

[88] Kwon O, Kim Y, Huh J, Yoon H. ZeroKernel: Secure context-isolated execution on commodity GPUs. IEEE Trans. on Dependable and Secure Computing, 2019. [doi: 10.1109/TDSC.2019.2946250]

[89] Müller T, Taubmann B, Freiling FC. TreVisor: OS-independent software-based full disk encryption secure against main memory attacks. In: Proc. of the 10th Int'l Conf. on Applied Cryptography and Network Security (ACNS). Berlin, Heidelberg: Springer-Verlag, 2012. 66–83. [doi: 10.1007/978-3-642-31284-7_5]

[90] Shinagawa T, Eiraku H, Tanimoto K, Omote K, Hasegawa S, Horie T, Hirano M, Kourai K, Oyama Y, Kawai E, Kono K, Chiba S, Shinjo Y, Kato K. Bitvisor: A thin hypervisor for enforcing I/O device security. In: Proc. of the 5th ACM SIGPLAN/SIGOPS Int'l Conf. on Virtual Execution Environments (VEE). ACM, 2009. 121–130. [doi: 10.1145/1508293.1508311]

[91] Skillen A, Barrera D, Van Oorschot PC. Deadbolt: Locking down Android disk encryption. In: Proc. of the 3rd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM). ACM, 2013. 3–14. [doi: 10.1145/2516760.2516771]

[92] Huber M, Horsch J, Wessel S. Protecting suspended devices from memory attacks. In: Proc. of the 10th European Workshop on Systems Security (EuroSec). ACM, 2017,10:1–6. [doi: 10.1145/3065913.3065914]

[93] Huber M, Horsch J, Ali J, Wessel S. Freeze and crypt: Linux kernel support for main memory encryption. Computers & Security, 2019,86:420–436. [doi: 10.1016/j.cose.2018.08.011]

[94] Zhao LY, Mannan M. Hypnoguard: protecting secrets across sleep-wake cycles. In: Proc. of the 23rd ACM SIGSAC Conf. on Computer and Communications Security (CCS). ACM, 2016. 945–957. [doi: 10.1145/2976749.2978372]

[95] Peterson PAH. Cryptkeeper: Improving security with encrypted RAM. In: Proc. of the 9th IEEE Int'l Conf. on Technologies for Homeland Security (HST). IEEE, 2010. 120–126. [doi: 10.1109/THS.2010.5655081]

[96] Götzfried J, Müller T, Drescher G, Nürnberger S, Backes M. RamCrypt: Kernel-based address space encryption for user-mode processes. In: Proc. of the 11th ACM Asia Conf. on Computer and Communications Security (ASIACCS). ACM, 2016. 919–924. [doi: 10.1145/2897845.2897924]

[97] Cao C, Guan L, Zhang N, Gao N, Lin JQ, Luo B, Liu P, Xiang J, Lou WJ. CryptMe: Data leakage prevention for unmodified programs on ARM devices. In: Proc. of the 21st Int'l Symp. on Research in Attacks, Intrusions, and Defenses (RAID). Cham: Springer-Verlag, 2018. 380–400. [doi: 10.1007/978-3-030-00470-5_18]

[98] Guan L, Cao C, Zhu SC, Lin JQ, Liu P, Xia YB, Luo B. Protecting mobile devices from physical memory attacks with targeted encryption. In: Proc. of the 12th Conf. on Security and Privacy in Wireless and Mobile Networks (WiSec). ACM, 2019. 34–44. [doi: 10.1145/3317549.3319721]

[99] Enck W, Gilbert P, Chun BG, Cox LP, Jung J, McDaniel P, Sheth AN. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In: Proc. of the 9th USENIX Symp. on Operating Systems Design and Implementation (OSDI). USENIX Association, 2010. 393–407. [doi: 10.5555/1924943.1924971]

[100] ARM Ltd. The ARM system memory management units. 2019. https://developer.arm.com/ip-products/system-ip/system-controllers/system-memory-management-unit

[101] Chu DW, Wang YW, Lei LG, Li YC, Jing JW, Sun K. OCRAM-assisted sensitive data protection on ARM-based platform. In: Proc. of the 24th European Symp. on Research in Computer Security (ESORICS). Cham: Springer-Verlag, 2019. 412–438. [doi: 10.1007/978-3-030-29962-0_20]

[102] Hong D, Bathen LAD, Lim SS, Dutt N. DynaPoMP: Dynamic policy-driven memory protection for SPM-based embedded systems. In: Proc. of the 6th Workshop on Embedded Systems Security (WESS). ACM, 2011,5:1–10. [doi: 10.1145/2072274.2072279]

[103] Zhang N, Sun K, Lou WJ, Hou YT. Case: Cache-assisted secure execution on ARM processors. In: Proc. of the 37th IEEE Symp. on Security and Privacy (SP). IEEE, 2016. 72–90. [doi: 10.1109/SP.2016.13]

[104] Chen X, Dick RP, Choudhary A. Operating system controlled processor-memory bus encryption. In: Proc. of the 11th Design, Automation and Test in Europe (DATE). IEEE, 2008. 1154–1159. [doi: 10.1109/DATE.2008.4484834]

[105] Chen Y, Khandaker M, Wang Z. Secure in-cache execution. In: Proc. of the 20th Int'l Symp. on Research in Attacks, Intrusions, and Defenses (RAID). Cham: Springer-Verlag, 2017. 381–402. [doi: 10.1007/978-3-319-66332-6_17]

[106] Yun MH, Zhong L. Ginseng: Keeping secrets in registers when you distrust the operating system. In: Proc. of the 26th Annual Network and Distributed System Security Symp. (NDSS). Internet Society, 2019. [doi: 10.14722/ndss.2019.23327]

[107] Papadopoulos P, Vasiliadis G, Christou G, Markatos E, Ioannidis S. No sugar but all the taste! Memory encryption without architectural support. In: Proc. of the 22nd European Symp. on Research in Computer Security (ESORICS). Cham: Springer-Verlag, 2017. 362–380. [doi: 10.1007/978-3-319-66399-9_20]

[108] Seitzer M, Gruhn M, Müller T. A bytecode interpreter for secure program execution in untrusted main memory. In: Proc. of the 20th European Symp. on Research in Computer Security (ESORICS). Cham: Springer-Verlag, 2015. 376–395. [doi: 10.1007/978-3-319-24177-7_19]

[109] Wu YM, Liu YT, Liu RF, Chen HB, Zang BY, Guan HB. Comprehensive VM protection against untrusted hypervisor through retrofitted AMD memory encryption. In: Proc. of the IEEE 24th Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2018. 441–453. [doi: 10.1109/HPCA.2018.00045]

[110] Palutke R, Neubaum A, Götzfried J. SEVGuard: Protecting user mode applications using secure encrypted virtualization. In: Proc. of the 15th Int'l Conf' on Security and Privacy in Communication Systems (SecureComm). Cham: Springer-Verlag, 2019. 224–242. [doi: 10.1007/978-3-030-37231-6_12]

[111] Henson M, Taylor S. Beyond full disk encryption: protection on security-enhanced commodity processors. In: Proc. of the 11th Int'l Conf. on Applied Cryptography and Network Security (ACNS). Berlin, Heidelberg: Springer-Verlag, 2013. 307–321. [doi: 10.1007/978-3-642-38980-1_19]

[112] Zhang MY, Zhang QY, Zhao SJ, Shi ZP, Guan Y. SoftME: A software-based memory protection approach for TEE system to resist physical attacks. Security and Communication Networks, 2019,8690853:1–12. [doi: 10.1155/2019/8690853]

[113] Zhang MY. Research on defending physical attacks for trusted execution environment based on on-chip memory [MS. Thesis]. Beijing: Capital Normal University, 2019 (in Chinese with English abstract).

[114] Ishida R, Honda S, Takada H, Fukui A, Ogawa T, Tawara Y. TOPPERS/FMP kernel: RTOS for embedded multiprocessor systems with real-time tasks and throughput-demanding tasks. Computer Software, 2012,19(4):219–243. [doi: 10.11309/jssst.29.4_219]

[115] Götzfried J, Dörr N, Palutke R, Müller T. Hypercrypt: Hypervisor-based encryption of kernel and user space. In: Proc. of the 11th Int'l Conf. on Availability, Reliability and Security (ARES). IEEE, 2016. 79–87. [doi: 10.1109/ARES.2016.13]

[116] Horsch J, Huber M, Wessel S. TransCrypt: Transparent main memory encryption using a minimal ARM hypervisor. In: Proc. of the 16th IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2017. 152–161. [doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.232]

[117] Gueron S. Memory encryption for general-purpose processors. IEEE Security & Privacy, 2016,14(6):54–62.

[118] Boudguiga A, Klaudel W, Wesolowski JD. On the performance of freescale i.MX6 cryptographic acceleration and assurance module. In: Proc. of the 7th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO). 2015. 1–8. [doi: 10.1145/2693433.2693441]

[119] Hennessy JL, Patterson DA. A new golden age for computer architecture. Communications of the ACM, 2019,62(2):48–60. [doi: 10.1145/3282307]

[120] Lee D, Kohlbrenner D, Shinde S, Asanović K, Song D. Keystone: An open framework for architecting trusted execution environments. In: Proc. of the 15th European Conf. on Computer Systems (EuroSys). ACM, 2020. 38:1–38:16. [doi: 10.1145/3342195.3387532]

[121] Ferraiuolo A, Baumann A, Hawblitzel C, Parno B. Komodo: Using verification to disentangle secure-enclave hardware from software. In: Proc. of the 26th Symp. on Operating Systems Principles (SOSP). ACM, 2017. 287–305. [doi: 10.1145/3132747.3132782]

[122] Jouppi NP, Young C, Patil N, *et al*. In-datacenter performance analysis of a tensor processing unit. In: Proc. of the 44th Annual Int'l Symp. on Computer Architecture (ISCA). ACM, 2017. 1–12. [doi: 10.1145/3079856.3080246]

[123] Chen TS, Du ZD, Sun NH, Wang J, Wu CY, Chen YJ, Temam O. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In: Proc. of the 19th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, 2014. 269–284. [doi: 10.1145/2541940.2541967]

[124] Chen YJ, Luo T, Liu SL, Zhang SJ, He LQ, Wang J, Li L, Chen TS, Xu ZW, Sun NH, Temam O. DaDianNao: A machine-learning supercomputer. In: Proc. of the 47th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). IEEE, 2014. 609–622. [doi: 10.1109/MICRO.2014.58]

[125] Du ZD, Fasthuber R, Chen TS, Ienne P, Li L, Luo T, Feng XB, Cheng YJ, Temam O. ShiDianNao: Shifting vision processing closer to the sensor. In: Proc. of the 42nd Annual Int'l Symp. on Computer Architecture (ISCA). ACM, 2015. 92–104. [doi: 10.1145/2749469.2750389]

[126] Liu DF, Chen TS, Liu SL, Zhou JH, Zhou SY, Temam O, Feng XB, Zhou XH, Cheng YJ. PuDianNao: A polyvalent machine learning accelerator. In: Proc. of the 20th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, 2015. 369–381. [doi: 10.1145/2694344.2694358]

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

[128] Brasser F, Müller U, Dmitrienko A, Kostiainen K, Capkun S, Sadeghi AR. Software grand exposure: SGX cache attacks are practical. In: Proc. of the 11th USENIX Workshop on Offensive Technologies (WOOT). USENIX Association, 2017. [doi: 10.5555/3154768.3154779]

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

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

[131] Götzfried J, Eckert M, Schinzel S, Müller T. Cache attacks on Intel SGX. In: Proc. of the 10th European Workshop on Systems Security (EuroSec). ACM, 2017. 2:1–2:6. [doi: 10.1145/3065913.3065915]

[132] Hähnel M, Cui WD, Peinado M. High-resolution side channels for untrusted operating systems. In: Proc. of the 2017 USENIX Annual Technical Conf. (USENIX ATC). USENIX Association, 2017. 299–312. [doi: 10.5555/3154690.3154719]

[133] Moghimi A, Irazoqui G, Eisenbarth T. CacheZoom: how SGX amplifies the power of cache attacks. In: Proc. of the 19th Int'l Conf. on Cryptographic Hardware and Embedded Systems (CHES). Cham: Springer-Verlag, 2017. 69–90. [doi: 10.1007/978-3-319-66787-4_4]

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

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

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

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

[138] Yarom Y, Falkner K. FLUSH+ RELOAD: A high resolution, low noise, L3 cache side-channel attack. In: Proc. of the 23rd USENIX Security Symp. (USENIX Security). USENIX Association, 2014. 719–732. [doi: 10.5555/2671225.2671271]

附中文参考文献:

[46] 霍文捷.嵌入式处理器安全运行机制的研究与设计[博士学位论文].武汉:华中科技大学,2010.

[113] 张美玉.基于片上内存的可信执行环境抗物理攻击研究[硕士学位论文].北京:首都师范大学,2019.

张倩颖(1986－),女,博士,讲师,CCF 专业会员,主要研究领域为嵌入式操作系统,系统安全,形式化验证.

赵世军(1985－),男,博士,华为技术有限公司技术专家,主要研究领域为信息安全,系统安全,可信计算.