◎网络、通信、安全◎

可信计算环境构建机制研究进展

程 戈,李 聪
CHENG Ge, LI Cong

湘潭大学 数学与计算科学学院,湖南 湘潭 411105
School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan 411105, China

CHENG Ge, LI Cong. Research progress of trusted computing environment. Computer Engineering and Applications, 2013, 49(13):59-64.

Abstract: Trusted computing environment provides a new arena to address the challenges in computer security by combining software and hardware to meet the definition of trusted computing. The authenticity, confidentiality, controllability and other properties that it provides can make up the deficiencies of traditional security methods. This paper describes the hardware basis of trusted computing, summarizes the recent trusted computing environment which is based on the DRTM(Dynamic Root of Trust for Measurement) and SRTM(Static Root of Trust for Measurement), analyzes the advantages and disadvantages of existed trusted computing environment, and indicates the direction of future research by analyzing the trust chain.
Key words: trusted computing environment; chain of trust; root of trust measurement

摘 要: 可信计算环境构建是通过软硬件结合的方式构建满足可信计算定义的系统,使其上进行的计算具有真实性、机密性、可控性等特性,并利用这些特性来弥补仅依靠传统安全防护方式的不足,从而更好地解决计算机安全面临的挑战和问题。介绍了可信计算环境构建的硬件基础,归纳了近年来基于静态可信度量根、动态可信度量根以及轻量虚拟机监控器的可信计算环境的构建机制,分析了现有可信计算环境构建机制的优势和不足;通过对可信计算环境中信任链的分析,指明了今后的研究方向。
关键词: 可信计算环境;信任链;可信度量根
文献标志码: A **中图分类号:** TP309 doi:10.3778/j.issn.1002-8331.1211-0049

1 可信计算环境

随着信息技术的发展,现代社会越来越依赖计算机系统。特别是近年来,在互联网技术的推动下,计算机越来越多地应用到社会政治、经济、教育和军事等领域中,使计算平台的安全性变得愈发的重要。然而自从计算机问世以来,计算机安全问题就一直伴随着计算机的发展而存在,近三十年来,其造成的损失也越来越严重。造成这种情况的一个重要原因,是传统的安全防护方式和软件的固有缺陷不足以防御日益增多的计算机安全问题。

(1)防火墙、入侵监测和病毒防范是构成传统信息安全系统的主要技术手段,这些技术手段是一种事后响应方式,即在攻击发生后或是进行中,通过对已发生过的滞后信息进行分析来判定是否存在攻击,从而进行相应的响应或是防护。面对当今日趋复杂和变化多端的恶意攻击手段,这些事后响应的传统防护手段往往无力应对新的攻击方式。

(2)现有平台架构是开放式的,计算机资源可任意被使用,尤其是执行代码可被任意修改。因此,在现有软件架构下,恶意程序很容易植入软件系统中。如果缺乏相关硬件的支持,仅仅依靠软件本身是不能完全检测出恶意代码的,因为所有试图通过软件检测恶意代码的方法都无法证明检测软件自身是安全的。

为克服这些问题,可信计算提供一个新的思路:从接入端增强系统的安全性,使系统每个接入端的计算平台都具有一定的物理保护,并在这样的平台上通过软硬件结合的方式构建可信的计算环境。可信计算环境可以确保其上进行的计算具有某些特性,例如,保证可信计算环境中运行程序和数据的真实性、机密性、可控性等。利用可信

基金项目: 国家自然科学青年基金(No.61202397);湘潭大学科研启动费项目(No.11QDZ42)。
作者简介: 程戈(1977—),男,博士,副教授,硕导,主要研究领域为可信计算,云计算;李聪(1988—),男,硕士研究生,主要研究领域为可信计算。E-mail:chengge@xtu.edu.cn
收稿日期: 2012-11-05 **修回日期:** 2013-02-05 **文章编号:** 1002-8331(2013)13-0059-06

计算环境提供的这些特性可以弥补仅依靠软件安全防护方式带来的不足,从而更好地解决计算机安全面临的问题和挑战。

本文介绍了可信计算环境构建的硬件基础,分析和归纳了近年来可信计算环境的主要的构建方式,指出了现有可信计算环境构建机制的不足,并对今后的研究方向进行了讨论。

2 可信计算平台

可信计算环境构建是通过软硬件结合的方式构建满足可信计算定义的系统,其目的是提升系统的安全性,其关注内容是服务器、网络和终端的行为及其相互间的影响。可信计算平台是可信计算环境构建的硬件基础。构建可信计算环境需要可信计算平台及软件相互协作。可信计算平台是具有一定物理防护的计算平台,该平台可以提供一定级别硬件安全来确保运行于该平台物理保护边界内的代码及数据具有某些特性,例如机密性、完整性、真实性等。

2.1 早期的可信平台

IBM4758安全协处理器[1]是早期的可信硬件,它是一个与主处理器相独立的处理器单元,用于负责和安全相关的计算。IBM4758设计目的是即使在存在本地敌手的情况下也能满足某些计算与存储性能。这些本地敌手可以是计算系统的操作者或是主处理器上正在运行的进程。安全协处理器可以提供一个隔离的运行环境确保该环境中计算的完整性和机密性,并能对外证明在这个环境中进行运算的真实性。

XOM[2]是斯坦福大学提出的一种基于只执行内存的安全增强型处理器结构,应用程序可以只信任处理器而不必信任操作系统。XOM体系可加密进程执行空间,只有处理器处于可信模式下才能用密匙解密程序。AEGIS[3]是是麻省理工学院提出的一种安全增强性处理器结构。类似IBM4758,该结构以处理器为信任根并假定处理器拥有一个秘密的密匙,AEGIS架构的处理器新增一种类似XOM的“安全执行模式”,并增加了新的指令用于进入/退出“全执行模式”和使用受保护的秘密密匙。AEGIS可用于认证执行和数字版权保护。Ceriums[4]是麻省理工学院提出的另一种可信处理器。Cerium结合上述增强型处理器的优点,它可以像XOM及AEGIS一样,通过加密被保护进程的地址空间来实现类似IBM4758中的认证执行。然而不同的是,Gerium采用软件的方式来实现加密保护进程,Gerium将一个可信微内核放入处理器内部,对被保护进程地址空间的所有操作都会触发这段微内核代码,由它来处理加密地址空间。

2.2 主流的商用可信平台

基于TCG[5]规范的TPM是商品化可信计算平台中的核心组件。TPM提供一组平台配置寄存器(Platform Configuration Registers,PCR)进行平台的配置证明。拥有TPM的可信计算平台可以通过PCR来记录从系统加电启动后进行的操作顺序和参与的软件。在“递归信任”过程中,前一阶段的代码在将控制权传递给下一阶段前,先对其进行度量,并把度量值保存到PCR中。这种递归信任必须有一个被称为可信度量根实体作为度量的起点,这个实体本身被假设为可信的,无法再被其他实体所度量。TPM提供可信报告根(Root of Trust for Reporting,RTR),RTR用签名密钥签名来证实PCR中数据的真实性。TPM还提供可信的存储根(Root of Trust of Storage,RTS),用于保护所有委托给TPM的密钥和数据。

商用CPU中,Intel和AMD分别推出了自己的增强型安全处理器技术:Intel的TXT[6]和AMD的SVM[7]。不同于前面介绍的其他的增强型安全处理器,TXT和SVM不提供加密进程空间的功能,仅在处理器中增加了新的指令作为动态可信度量根(Dynamic Root of Trust Measurement,DRTM),这条指令可以实现进程的保护执行,但是TXT或是SVM需要和TPM(1.2版本)相结合,才能够提供封装存储和对外认证的功能。

早期的可信硬件平台依赖于可信硬件在其物理保护边界内的可信计算环境。物理保护代价是比较昂贵的,这也注定了单纯依赖硬件保护的早期可信计算平台其计算能力是有限的,不能满足大部分的应用需求,只能应用于一些特定的场景(例如用于保存个人密钥的个人令牌)。而现有的商用可信平台是作为可信计算平台附属设备,可信硬件为构建可信计算环境提供了硬件基础,然而只有通过软件的协助才能构建满足多种应用需求的可信计算环境。

本文将单独依赖硬件保护的计算平台和把可信硬件作为附属设备的计算平台统一称为可信计算平台。

3 基于静态可信根的可信计算环境

构建基于静态可信度量根的可信计算环境,主要是构建静态信任链,通常由固化在主板BIOS中一段称为可信度量根(Root of Trust for Measurement,RTM)的代码开始,随着平台的启动,通过一种“递归信任”的过程将信任由RTM扩展到整个平台。

3.1 构建机制

安全硬件加强的MyProxy(SHEMP)[8]是Dartmouth大学2004年的一个项目,主要研究使用IBM4758协处理器来管理终端,通过控制安全边界来加强安全性,可用于MyProxy服务器。

TrustedGRUB[9]是Sourceforge支持的开源项目,它扩展了原始的GRUB引导程序来支持TPM提供的递归信任。在TrustedGRUB的引导过程中,TrustedGRUB扩展了引导加载程序GRUB(Grand Unified Boot Loader)的Stage1,可以度量GRUB Stage2的第1个扇区,包括Stage2的输入参数(内核、启动模块和相关配置信息)。TrustedGRUB将信任链扩展到操作系统层。

BEAR[10-11]是Dartmouth大学PKI实验室的研究项目,主要研究在具有TCG/TPM的商用可信平台上使用Linux构建可信桌面计算环境,该可信计算环境需要修改Linux

操作系统,但保持对操作系统上的应用透明,即传统软件和应用程序可以不加修改地在这个平台上运行。BEAR将信任链扩展到文件层,操作系统下任意一个目录第一次被打开时,BEAR都会检查该目录下文件的完整性。同时BEAR可以提供远程认证和安全存储功能。

IMA[12]是IBM提出的一种可信计算环境构建架构,类似BEAR项目,IMA同样是在拥有TCG/TPM的商用可信计算平台上使用Linux构建可信计算环境,并提供远程认证和安全存储的功能。不同的是,IMA采用在一些系统调用中加钩子的方式,当可加载内核模块和程序载入内存时,对其进行完整性校检,以使信任链可以扩展到应用程序层。

在20世纪70年代,IBM公司开发了安全内核的虚拟机监控模块KVM/370[13];在20世纪80年代,在VAX机上也开发了基于VMM的安全内核[14];在2000年4月,IBM又推出Z系列大型安全服务器系统[15]。这些基于虚拟机的可信计算环境在大型机上的长期使用,证明了基于虚拟机架构构建可信计算环境对增强系统安全的有效性。

斯坦福大学在2003年研发的Terra系统[16],是一个基于虚拟机架构的可信计算平台。Terra通过可信的虚拟机监视器(TVMM),在具有防篡改功能(temper-resistant)的可信硬件平台上(例如IBM4758协处理器)实现多个相互独立的虚拟机(VM)。这些虚拟机可以是具有通用性而对安全性没有特殊要求的Open Box,也可以是有特殊用途的、高安全性的Close Box。可信虚拟机的引入也为原有的专属平台上的应用,例如ATM,提供了向普通商用平台移植的可能。IBM实现的VTPM[17]为单一硬件平台的多个虚拟机提供可信计算支持。通过虚拟化可信平台模块(TPM),TPM的安全存储和加解密功能能够应用于虚拟机上的操作系统和应用程序。原有的基于TPM的可信计算环境,在引入虚拟层后,可以将TPM的信任链向上扩展建立可信虚拟机,建立可信虚拟机的方式采用传统方式,如IBM IMA架构。

3.2　主要问题

可信硬件作为构建可信计算环境的硬件基础,通过逐级度量的方式扩展信任环境,解决了早期单纯依赖硬件的物理保护造成的成本过高问题,使可信计算进入商业应用成为可能。然而,逐级度量也存在诸多问题。

IBM的安全协处理器和TCG的可信度量从系统加电开始,首先将控制权交给BIOS中固化的一段代码(信任根),信任根通过计算哈希值的方式度量BIOS中其他部分,并把度量值存储到TPM的PCR中,然后将控制权移交给被度量的代码,这个过程一直进行下去直到操作系统启动完毕。MyProxy(SHEMP)和TrustedGRUB[9]都通过这种方式构建可信计算环境。然而,IBM 的安全协处理器和TCG的可信启动只定义了度量加载的固定启动序列,而没有定义如何将信任链扩展到应用程序层。实际上当操作系统启动后,还有大量可执行的代码被加载(内核模块、二进制的共享库、脚本、插件、进程等),这些被加载的代码依据实际的需求而没有固定的序列。

Dartmouth大学PKI实验室的研究项目BEAR[10-11]和IBM研究中心Sailer等人的IMA系统[12]通过修改操作系统内核的方式(对某些系统调用加钩子)来度量操作系统启动后载入的可执行代码,如图1所示。但是,在操作系统中增加钩子函数的方式同样存在诸多问题。首先,修改操作系统的内核意味着这些方法对操作系统是不透明的,这使现有的操作系统无法支持应用程序级的完整性度量和认证;其次,该方式以操作系统为信任基(TCB),操作系统代码庞大,系统漏洞数量多且很难发现。

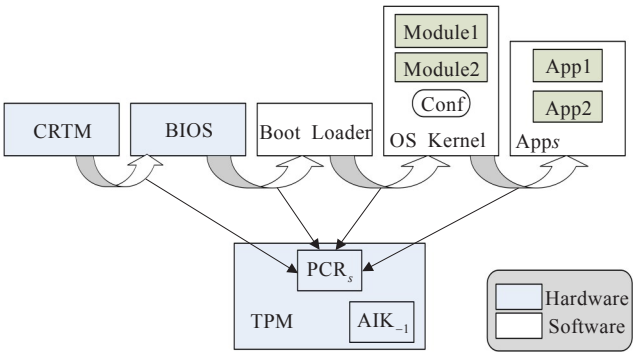

图1　IMA的完整性度量方式

基于虚拟机架构的可信计算以虚拟机监控器作为信任链中的一环,采用TCG信任链的扩展方式从可信根构建到应用层的信任链。这种方式使原本已经过长的静态信任链问题变得更加严重。还有一些研究采用粗粒度的信任链构建方式。例如,Terra以虚拟机镜像为单位进行度量。一方面,这种方式会产生较大的度量值,以4 GB的镜像为例,Terra大约要产生一个20 MB的度量值;另一方面,这种度量值不能用于解释度量内容,很难应用于远程认证的场景。

静态可信度量根自身的缺陷是造成上述问题的主要原因。因为在基于静态可信度量根的可信计算环境中,上层应用是否可信不仅取决于其自身的完整性,还包括其下层软件栈中每一个实体的完整性。同时,可信计算环境提供的隔离性可能比较弱,除非是在具有强制访问控制的系统中,否则,所有的进程都可能相互影响。因此,所有的软件实体,不管它是固化到BIOS中的代码还是由操作系统加载、动态加载器或是程序代码加载的都需要被度量。这就不可避免地造成信任链过长的问题。

基于静态可信度量根的可信计算系统存在着运行时刻和验证时刻不一致的问题。例如,在远程证明中,通过向质询的一方报告平台上已经运行程序的散列值(PCRs),以提供平台可信的证据,可是这些证据仅仅代表了程序在完整性度量时刻的状态,并不能表明度量后程序在执行时的状态,静态的散列值并不能充分表明程序当前的动态行为也可信。攻击者完全可能恰好在远程证明结束后,在平台中植入恶意程序。此外,由于现有软件的复杂性,出现漏洞的几率很高,经过度量后的程序在运行时很可能产生非预期的结果。例如,遭受病毒发起的缓冲区溢出攻击,程序行为发生改变,平台进入非预期的状态。

4 基于动态可信根的可信计算环境

动态可信度量根(Dynamic Root of Trust for Measurement,DRTM)不依赖于系统加电重启的过程,而是允许系统从任何一个不被信任的状态作为测量的起点来建立信任链。基于动态可信根的可信计算环境构建是近年来的研究重点。

4.1 构建机制

OSLO[18](Open Secure Loader)是基于AMD64位处理器的动态可信度量根的开源引导加载器,它通过AMD动态可信度量根技术中新增加的SKINIT指令来替代基于BIOS的静态可信度量根。OSLO启动加载器作为操作系统内核的一部分,它初始化TPM,通过TPM设备驱动来实现与TPM通信,调用SKINIT指令,度量所有被加载的文件,并将度量值扩展到TPM的动态PCR中。TBoot[19](Trusted Boot)类似于OSLO,但是TBoot使用的是Intel TXT(Trusted eXecution Technology)技术提供的动态可信度量根来度量操作系统内核/虚拟机监控器(Virtual Machine Monitor,VMM)。TBoot可以实现可信启动,撤销度量环境,重置数据保护,保护TXT内存范围等功能。

Flicker[20]利用AMD SVM中新增的动态可信度量根扩展指令暂停操作系统和其上运行的其他程序,为敏感代码创建一个完全隔离于操作系统的运行环境,并可以实现远程证明和安全存储。Flicker直接以动态可信度量根作为可信计算基,力图实现可信计算基的最小化。Flicker给出了创建可信执行环境——执行可信计算基——保存运行结果——撤销可信执行环境的具体实现。

Intel TXT安全架构是Intel在其增强型安全处理器上提出的一种可信计算环境构建架构。TXT架构为被保护软件提供安全的执行环境,防止其他软件访问到被保护软件的资源。Intel TXT并不像其他的增强型安全处理器一样具有安全模式,并当处理器离开安全模式时,由处理器来加密被保护软件的资源,将其同其他的软件实体隔离开。Intex TXT并不像其他的增强型安全处理器一样具有安全模式(当处理器离开安全模式时,由处理器来加密被保护软件的资源,将其同其他的软件实体隔离开),而是通过动态可信度量根技术暂停操作系统和其上正在运行的其他软件来实现安全执行环境。如果要在Intel TXT的可信平台上同时运行不安全的代码与安全的代码,必须依赖软件的方法。虚拟化技术提供的隔离性正好可以满足这个需求,在支持TXT处理器中几乎都同时支持Intel的硬件辅助虚拟化技术。在TXT安全架构中除了要求平台的处理器支持TXT技术外,还要求平台拥有特定的芯片组,及可信的I/O外设(键盘、鼠标和显卡)和TPM1.2。TXT安全架构的设计目标提供以下的安全特性:

(1)安全运行环境。为程序提供隔离的安全运行环境,防止其他未被授权的平台软件实体获取或是损害在这个保护运行环境中的程序。

(2)封印存储。为密钥、数据和其他敏感内容提供基于平台可信硬件(依赖于TPM)的封印存储:将这些数据同他们被存储时的系统环境绑定在一起,只有在同样的运行环境中才能被解密。

(3)可信输入和显示。提供输入设备,例如,键盘/鼠标到安全运行环境中应用程序以及安全运行环境中应用程序到显卡缓冲区间的安全路径,防止平台上运行的其他没有被授权的软件实体窃取或是篡改输入和显示的信息(需要特定的可信键盘、鼠标和显卡)。

(4)对外认证。提供安全执行环境和运行于其中程序的完整性证明。

图2显示了Intel TXT安全架构。不同安全需求的操作环境被虚拟机监控器隔离到不同的虚拟域,这个架构同Terra类似。但不同的是,依据TXT的设计,这个虚拟机监控器可以依据保护需要动态加载而不需要重启系统,并通过TPM1.2可以提供从TXT动态可信度量根到虚拟机监控器的信任链证明。TXT安全架构建议的隔离环境分为以下几种:

标准隔离域。该域提供一个标准的IA-32平台,在这个域中用户可以运行任何能在物理IA-32 PC平台上运行的操作系统和软件。在TXT架构设计中,虚拟机监控器可以依据保护需要在运行的操作系统中启动,并将原有的操作系统和其上的应用软件迁移到标准的隔离域中。

保护域。该域同标准隔离域共同运行于虚拟机监控器之上,运行于该域的软件实体同标准域中的软件实体都是完全隔离的,不会被标准域的软件实体侵害。当保护域停止运行时,安全域中的敏感数据将被保护到TPM中。

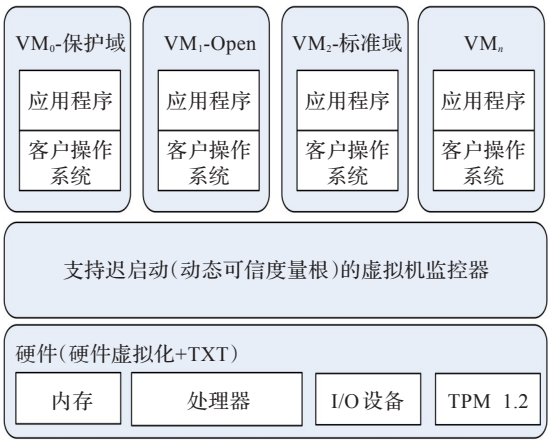

图2 Intel TXT安全架构图

微软的下一代安全计算基础(Next-Generation Secure Computing Base,NGSCB)[21]是TXT安全架构的一个特例化实现。虚拟机监控器将硬件资源分割成两个部分:左侧运行原生的遗留操作系统;右侧是一个微型的安全内核称为节点(nexus),其上运行的应用程序称为代理(agent),代理运行于一个隔离的地址空间并使用认证过的操作原语。虚拟机监控器和微安全内核一起为代理提供与TXT安全架构一致的安全特性:安全运行环境、封印存储、可信输入和显示及对外认证。

4.2　优势和问题

动态可信度量允许系统从任何一个不被信任的状态作为测量的起点来建立信任链,因此可以建立比静态度量更短的信任链。OSLO和TBoot使用动态可信度量根可以将BIOS和GRUB排除在信任链之外,然而它们延续的依然是传统的信任链构建方式,操作系统还必须包括进信任链中。这使其不可避免地具有和基于静态信任度量根的信任链相同的缺点。

Flicker直接以动态可信度量根作为信任基,其拥有最短的信任链。然而这是以暂停操作系统和其上正在运行的其他软件来实现的。因此,Flicker架构提供给用户的是一个近似裸机的编程环境,其架构下代码无法调用操作系统的系统调用,无法调用库函数,甚至无法调用程序本身不属于本次会话的代码,这也极大地限制了Flicker的应用范围。

如果要在动态可信度量根加载后,同时在平台上运行不安全的代码与安全的代码,必须保证可信代码同不可信代码之间的隔离性,这需要依赖软件的方法。虚拟化技术提供的隔离性正好可以满足这个需求,在支持动态可信度量根的处理器中几乎都同时支持硬件辅助虚拟化技术。Intel的TXT架构和微软的NGSCB架构结合动态可信根和虚拟化技术,基本上克服基于静态可信根构建可信计算环境的缺点。

然而,Intel TXT和微软的NGSCB的可信计算环境依然是建立在隔离域的基础上的,这不可避免地使虚拟机监控器因为资源管理和提供有效的虚拟域隔离而变得庞大(以Xen为例,不包括管理域,其Hypervisor的代码大约是50 000行)。同时,Intel TXT和微软NGSCB架构的可信基复杂度类似于操作系统。虽然可以利用动态可信度量根和硬件辅助虚拟化技术将宿主操作系统排除在信任基外,而其隔离域内的信任链构建却依然类似传统的静态信任链的构建,无法解决维护数量巨大的散列值的问题。因此,上述研究随着微软放弃了NGSCB计划而停留在构想阶段,缺乏可实施的设计。

5　基于轻量虚拟化架构的可信计算环境

依赖于可信硬件构建可信计算环境,无论采用静态可信度量根或是动态可信度量根都存在着信任链构建的性能损失问题,同时通用目的的虚拟机监控软件是为服务器端的资源聚合和管理而设计的,虚拟化带来较大的性能开销。近期的一些研究,尝试采用经过裁剪的轻量级虚拟机监控器作为可信根来构建可信计算环境。

SecVisor[22]使用轻量虚拟机监控器防止在内核模式下执行未经授权的代码访问内核页表,来维护内核代码的完整性。SecVisor还使用IOMMU来防止通过DMA写操作访问内核。BitVisor[23]通过轻量虚拟机管理器拦截访问I/O指令的一个子集,并通过解析和控制I/O指令来提升I/O操作的安全性。TrustVisor[24]通过轻量虚拟机监控器提供为进程特定的代码和数据提供完整性保护。Tsunghan Lin[25]在多核嵌入式平台上使用轻量虚拟机,提供更好的系统可靠性和安全性。NOVA[26]提出一种虚拟机监控器的裁剪方案,可以提供更小的可信基,也能提供传统虚拟机监控器支持的多个隔离虚拟域,可为多个原生的操作系统提供隔离的运行环境。HyperSafe[27]通过内存锁和限制控制流,提出了一种能够保护自身完整性的轻量虚拟化方案。CloudVisor[28]采用嵌套的虚拟化技术,使用轻量虚拟机监控器,在云计算场景下,确保上层用户的虚拟机即使在云提供商的虚拟机监控器不可信的情况下,依然能够保护它们的数据隐私和完整性。

这些基于轻量虚拟化的安全机制探讨了采用轻量虚拟化架构为高安全需求的应用提供更好的隔离性,提升I/O操作安全性以及如何确保上层应用的完整性的问题,并部分解决了虚拟机管理器代码庞大及自身的完整性问题,以及依赖硬件构建信任链的性能问题。然而,它却忽略了可信计算环境的构建必须解决可信基的自适应性、兼容性以及信任架构和可信服务等方面的问题。

6　可信计算环境构建的研究趋势

从以上相关工作来看,操作系统的固有缺陷和虚拟化架构的优势使现有的研究主要关注于基于虚拟机架构构建可信计算环境。裁剪虚拟机监控器提供更小的可信基,是基于虚拟化技术构建可信计算环境的研究趋势[22-28]。结合动态可信度量根和轻量虚拟化技术构建的可信计算环境能够解决传统信任链过长,运行时刻和验证时刻不一致等问题。同时,虚拟机化架构提供的隔离性使这些实体在操作系统恢复后运行于一个独立的环境中,而不依赖于操作系统和其他平台实体[6-7,13-16,20-21]。

可信基、信任链和可信执行环境的实现机制是解决可信计算环境构建的三个关键要素,这三者又是相互关联和制约的。但现有的解决方案还存在如下问题:

(1)如何依据定制的安全策略和软件状态变化动态加载和卸载轻量虚拟化可信基,如何实现可信基的动态可扩展性以及可信基及其组件的通信机制?

(2)由于操作系统自身的诸多缺陷以及轻量虚拟化可信基的动态性,如何确保可信基自身的可信性?动态可信度量如何处理轻量虚拟化可信基的自适应问题和可信基及软件组件间的信任传递?

(3)如何基于轻量虚拟化可信基对应用软件的敏感操作及信息进行标记,并进行有效跟踪?由于应用需求的多样性,标记策略需要解决这种针对多样性带来的复杂性问题,另外跟踪模型需要适应资源变化的动态性。

(4)如何在操作系统中依据用户制定的安全策略进行

细粒度的内存隔离？如何基于轻量虚拟化可信基为应用提供可信服务?由于可信基的轻量化设计,需要解决轻量化设计与遗留代码保护带来的兼容性问题。

针对这些问题,本文认为构建可信执行环境所涉及的诸多要素进包括可信基的剪裁,自适应和完整性机制,动态信任架构,行为监控机制,细粒度内存隔离和可信服务调用等,这些将是未来可信计算研究的方向。

(1)轻量虚拟化可信基的构建机制:研究解决轻量虚拟化可信基的动态拆装和自适应问题;

(2)基于轻量虚拟化可信基的动态可信架构:研究解决基于轻量虚拟化可信基的动态信任链构建和完整性保护等问题;

(3)基于轻量虚拟化可信基的软件行为监控机制:研究解决基于轻量级虚拟化可信基的软件行为监控问题;

(4)基于轻量级虚拟化的软件可信执行机制:研究解决软件执行过程中的内存隔离和可信服务调用等问题。

这些问题的解决将为有高安全性需求的应用提供可信保障,推动虚拟化技术在构建可信计算环境方面的商业化应用。

7 结论

本文总结和归纳了现有可信计算环境的构建机制,分析了基于静态可信度量根及及动态可信度量根的可信计算环境存在的问题,并指出了可信计算环境的研究趋势。

参考文献:

[1] Dyer J G,Lindemann M,Perez R,et al.Building the IBM4758 secure coprocessor[J].IEEE Computer,2001,34(10):57-66.

[2] Lie D,Thekkath C,Mitchell M,et al.Architectural support for copy and tamper resistant software[C]//Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems,Cambridge,MA,USA,2000:168-177.

[3] Suh G,Clarke D,Gassend B,et al.ARGIS:architecture for tamper-evident and tamper resistant processing[C]//Proceedings of the 17th International Conference on Supercomputing,San Francisco,CA,USA,2003:160-171.

[4] Chen B,Morris T.Certifying program execution with secure processors[C]//Proceedings of the 9th Hot Topics in Operating Systems,Lihue,Hawaii,USA,2003,9:23-29.

[5] Trusted Computing Group.TCG 1_4 architecture overview[EB/OL].[2012-10-01].http://www.Trustedcomputinggroup.org/files/resource_files/AC652DE1109-3519-ADA026A0C05CFAC2/TCG_1_4 Architecture_Overview.pdf.

[6] Intel Corporation.Intel trusted execution technology software development guide[EB/OL].[2012-10-01].http://www.intel.com/technology/security/down-loads/315168.htm.

[7] Advanced Micro Devices.AMD64 virtualization:secure virtual machine architecture reference manual[M].[S.l.]:AMD Publication,2005-05.

[8] Marchesini J,Smith S.SHEMP:secure hardware enhanced MyProxy[C]//Proceedings of 3rd Annual Conference on Privacy,Security and Trust,New Brunswick,Canada,2005.

[9] Trustedgrub[EB/OL].[2012-10-01].http://sourceforge.net/projects/trustedgrub.

[10] MacDonald R,Smith S W,Marchesini J,et al.Bear:an open-source virtual secure coprocessor based on TCPA,Technical Report TR2003-4-71[R].Hanover,New Hampshire:Dartmouth College,2003.

[11] Marchesini J,Smith S W,Wild O,et al.Experi menting with TCPA/TCG hardware,or:How I learned to stop worrying and love the bear,Technical Report TR2003-476[R].Hanover,New Hampshire:Dartmouth College,2003.

[12] Sailer R,Zhang X,Jaeger T,et al.Design and implementation of a TCG-based integrity measurement architecture[C]//Proceedings of the 13th Conference on USENIX Security Symposium,Berkeley,CA,USA,2004:223-238.

[13] Goldberg R.Survey of virtual machine research[J].IEEE Computer Magazine,1974,7:34-45.

[14] Karger P A,Zurko M E,Bonin D W,et al.A retrospective on the VAX VMM security kernel[J].IEEE Transactions on Software Engineering,1991,17(11):1147-1165.

[15] Security:IBM zSeries partitioning achieves highest certification[EB/OL].[2012-10-01].http://www.ibm.com/servers/eserver/zseries/security/certification.html.

[16] Garfinkel T,Pfaff B,Chow J,et al.Terra:a virtual machine-based platform for trusted computing[C]//Proceedings of the 19th ACM Symposium on Operating Systems Principles,Bolton Landing,NY,USA,2003:193-206.

[17] Berger S,Cáceres R,Goldman K,et al.vTPM:virtualizing the trusted platform module[C]//Proceedings of the 15th Conference on USENIX Security Symposium,2006.

[18] Open Secure Loader[EB/OL].[2012-10-01].http://os.inf.tudresden.de/~kauer/oslo/.

[19] Trusted Boot[EB/OL].[2012-10-01].http://sourceforge.net/projects/tboot/.

[20] McCune J M,Parno B J,Perrig A,et al.Flicker:an execution infrastructure for TCB minimization[C]//Proceedings of the 3rd SIGOPS/EuroSys Conference on Computer Systems,Glasgow Scotland UK,2008:315-328.

[21] Microsoft Corp.Next generation secure computing base[EB/OL].[2012-10-01].http://www.Microsoft.com/resources/ngscb/default.mspx.

[22] Seshadri A,Luk M,Qu N,et al.SecVisor:a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes[C]//Proceedings of the 21st ACM Symposium on Operating System Principles,Stevenson,Washington,USA,October,2007:335-350.

(下转197页)

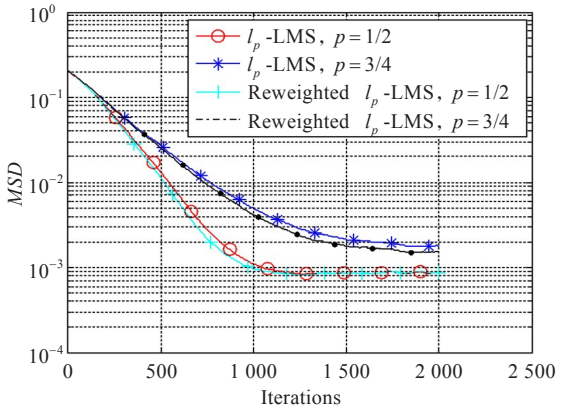

图7　稀疏系统和白信号条件下不同 p 值的 l_p -LMS算法收敛曲线

5　结论

针对稀疏的系统辨识问题，本文提出了一种改进的稀疏系统辨识方法——加权的 $l_p(0<p\leqslant 1)$ 范数惩罚LMS算法。仿真实验结果表明：对于稀疏的系统，改进算法的收敛性和稳态性有明显提高；在[1/2，1]范围内 p 取值越小，自适应滤波器的性能也就相对较好。 p 在(0，1/2)区间内步长参数的选择，将是下一阶段的重点研究问题。

参考文献：

[1] Bershad N J，Bermudez J C M，Toumere J Y.Stochastic analysis of the LMS algorithm for system identification with subspace input[J].IEEE Transation on Signal processing，2008，56(3)：1018-1027.

[2] 刘艳玲，邱丙益，樊长江.基于LMS算法的卫星通信回波抵消方法[J].船舶电子工程，2007，27(6)：100-102.

[3] 孙永国.稀疏路径回波对消自适应并法的研究[D].成都：四川大学，2006.

[4] 陈立峰.自适应声回波抵消算法的研究与实现[D].厦门：厦门大学，2006.

[5] 赵亮，朱维庆，朱敏.一种用于水声相干通信系统的自适应均衡算法[J].电子与信息报，2008，30(3)：648-651.

[6] Haykin S.Adaptive filter theory[M].4th ed.Upper Saddle River，NJ：Prentice Hall，2002.

[7] Diniz P S R.自适应滤波算法与实现[M].2版.刘郁林，景晓军，谭刚兵，译.北京：北京电子工业出版社，2004.

[8] Widrow B，Hoff M E.Adaptive switching circuits[C]//Proceedings of the IRE WESCON Conference，1960：96-104.

[9] 孙锦华，金力军.几种改进型RLS算法在自适应滤波系统中的应用[J].重庆邮电学院学报，2003，15(3)：14-15.

[10] 翁玉麟，邓长虹.自适应神经网络模糊推理系统最优参数的研究[J].计算机仿真，2005，22(8)：140-243.

[11] Fu W.Penalized regressions：the bridge versus the lasso[J].Journal of Computational and Graphical Statistics，1998，7(3)：397-416.

[12] 金坚，谷源涛，梅顺良.用于稀疏系统辨识的零吸引最小均方算法[J].清华大学学报：自然科学版，2010，50(9)：1312-1315.

[13] Chen Y，Gu Y，Hero A O.Sparse LMS for system identification[C]//Proc of IEEE ICASSP，Taipei，Taiwan，China，Apr，2009：3125-3128.

[14] Gu Y，Jin J，Mei S. l_0 norm constraint LMS algorithm for sparse system identification[J].IEEE Signal Processing Letters，2009，16(9)：774-777.

[15] Candes E J，Wakin M B，Boyd S P.Enhancing sparsity by reweighted l_1 minimization[J].Journal of Fourier Analysis and Applications，2008，14(5)：877-905.

（上接64页）

[23] Shinagawa T，Eiraku H，Tanimoto K，et al.BitVisor：a thin hypervisor for enforcing I/O device security[C]//Proceeding of the ACM SIGPLAN/SIOPS International Conference on Virtual Execution Environments，Washington DC，USA，March 2009：121-130.

[24] McCune J M，Li Y，Qu N，et al.TrustVisor：efficient TCB reduction and attestation[J]//Proceedings of IEEE Symposium on Security and Privacy，Washington DC，USA，May 2010.

[25] Lin T，Kinebuchi Y，Courbot A，et al.Hardware-assisted reliability enhancement for embedded multi-core virtualization design[J]//Proceedings of 14th IEEE International Symposium on Object/Component /Service Oriented Real-Time Distributed Computing(ISORC)，Newport Beach，CA，April 2011：241-249.

[26] Steinberg U，Kauer B.NOVA：a microhypervisor-based secure virtualization architecture[C]//Proceedings of the 5th European Conference on Computer Systems，New York，NY，USA，March 2010：209-222.

[27] Wang Z，Jiang X.HyperSafe：a lightweight approach to provide lifetime hypervisor control flow integrity[C]//Proceedings of the the IEEE Symposium on Security and Privacy(SP)，Oakland，CA，USA，July 2010：380-395.

[28] Zhang F，Chen J，Chen H，et al.CloudVisor：retrofitting protection of virtual machines in multitenant cloud with nested virtualization[C]//Proceedings of the 23rd ACM Symposium on Operating Systems Principles(SOSP-2011)，Cassias，Portugal，October 2011：203-216.