软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(3):662−674 [doi: 10.13328/j.cnki.jos.004447]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

面向云计算模式运行环境可信性动态验证机制*

刘川意[1,3], 林 杰[2,3], 唐 博[4]

[1](北京邮电大学 软件学院,北京 100876)

[2](北京邮电大学 计算机学院,北京 100876)

[3](可信分布式计算与服务教育部重点实验室(北京邮电大学),北京 100876)

[4](中国邮政储蓄银行 信息科技建设部,北京 100808)

通讯作者: 刘川意, E-mail: cy-liu04@mails.tsinghua.edu.cn

摘 要: 如何为用户提供一个可证明、可验证的可信运行环境,是云计算模式面临的重要问题.提出一种动态的用户运行环境可信性验证机制 TCEE(trusted cloud execution environment).通过扩展现有可信链,将可信传递到用户虚拟机内部,并周期性地对用户运行环境的内存和文件系统进行完整性验证.TCEE 引入可信第三方 TTP(trusted third party),针对用户虚拟机运行环境的可信性进行远程验证和审计,避免了由用户维护可信验证的相关信息和机制,同时也能够避免云平台敏感信息的泄露.实现了基于 TCEE 的原型系统,对 TCEE 的有效性和性能代价进行定量测试和评价.实验结果表明,该机制可以有效检测针对内存和文件系统的典型威胁,且对用户运行环境引入的性能代价较小.

关键词: 云计算;可信性验证;可信计算;TPM

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 刘川意,林杰,唐博.面向云计算模式运行环境可信性动态验证机制.软件学报,2014,25(3):662−674. http://www.jos.org.cn/1000-9825/4447.htm

英文引用格式: Liu CY, Lin J, Tang B. Dynamic trustworthiness verification mechanism for trusted cloud execution environment. Ruan Jian Xue Bao/Journal of Software, 2014,25(3):662−674 (in Chinese). http://www.jos.org.cn/1000-9825/4447.htm

Dynamic Trustworthiness Verification Mechanism for Trusted Cloud Execution Environment

LIU Chuan-Yi[1,3], LIN Jie[2,3], TANG Bo[4]

[1](Software School, Beijing University of Posts and Telecommunications, Beijing 100876, China)

[2](School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

[3](Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing 100876, China)

[4](Department of IT Construction, Postal Savings Bank of China, Beijing 100808, China)

Corresponding author: LIU Chuan-Yi, E-mail: cy-liu04@mails.tsinghua.edu.cn

Abstract: Providing a provable and verifiable execution environment for the tenants is a very important problem in the cloud computing mode. This paper proposes a dynamic trustworthiness verification mechanism for the tenants' virtual execution environment, named TCEE (trusted cloud execution environment), which extends the current trusted chain into virtual machine's architecture stack. It cyclically verifies the trustworthiness of the memory and file systems within the virtual execution environments. TCEE introduces a TTP (trusted third party) to perform the verification and audit action against tenants' virtual machines to avoid heavy involvement of end tenants and unnecessary information leakage of the cloud providers. A prove-of-concept prototype is implemented according to TCEE to evaluate the

* 基金项目: 国家自然科学基金(61202081)
收稿时间: 2013-02-07; 修改时间: 2013-03-29; 定稿时间: 2013-06-21; jos 在线出版时间: 2013-07-25
CNKI 网络优先出版: 2013-07-25 14:06, http://www.cnki.net/kcms/detail/11.2560.TP.20130725.1406.004.html

© 中国科学院软件研究所 http://www.jos.org.cn

effectiveness and the performance overhead incurred. Experimental results show that TCEE is effective and its performance overhead is minor.

Key words: cloud computing; trustworthiness verification; trusted computing; trusted platform module

云服务的推广和有效使用,很大程度上取决于云计算的可信性[1,2].云计算可信问题的根源在于用户失去了对托管在云端的数据和运行环境的直接控制能力.这些资源可能会受到两方面的威胁:一方面会遭受到来自于云提供商内部的窃取和破坏[3];另一方面由于云计算资源的共享使用,用户的资源也可能受到来自其他云用户的威胁[4].

具体来说,云计算模式提供给用户(tenant)(云提供商的用户既可以是服务提供商,也可以是最终用户.为方便以下简称为用户)的运行环境是以虚拟机为基础和核心的[5].如何在云计算模式下给用户提供可信的运行环境?总结起来,其面临如下的技术挑战:

- 云提供商单方面证明自己可信很难让用户信服.从体系结构栈来看,用户运行环境中的程序或任务是运行在云提供商的平台之上的,而用户仅仅是远程连接云提供商管理的虚拟机,能够获得的云平台信息是有限的,更无法了解云平台内部实现的细节;
- 云提供商为了吸引潜在用户,是倾向于证明自己是可信的.然而证明自身可信性的同时,势必会暴露敏感信息,这些信息可能会被利用来发起对云平台的攻击.因此,如何收集云提供商的可信性证据,使其具有不可抵赖性且不会给云提供商增加更多潜在攻击,是一个两难问题;
- 不同云提供商可能使用不同的物理机器、操作系统、虚拟机管理器等[6],而且其各自的云平台架构和部署也不尽相同.不能因为要验证其可信性,就要求不同的云提供商具有相同的配置,云提供商应有自主选择软、硬件的权利;
- 传统的可信计算(trusted computing)技术可以保证服务器在启动过程的可信性.而用户的运行环境是以虚拟机为载体的,如何将可信任链(trusted chain)扩展到虚拟机体系结构栈,是云计算环境面临的新问题.更进一步的,云节点的状态(如服务器硬件变化、安装新软件、软件升级等)是动态变化的[6],因此用户运行环境的可信性需要需要在运行时保证.

针对以上挑战,本文提出面向云计算模式的用户运行环境可信性动态验证机制 TCEE(trusted cloud execution environment),通过引入可信第三方 TTP(trusted third party),针对用户虚拟机运行环境的可信性进行远程验证和审计,只有通过 TTP 验证的虚拟机才能为用户提供可信的运行环境,从而避免由用户安装、维护可信性验证的相关设施和信息,并保证其完整性遭到破坏时能够及时被检测出来.同时,云平台的配置、运行信息只发送给 TTP,亦可避免云平台敏感信息的泄露.

本文的主要贡献在于:

- 通过引入可信第三方 TTP,针对云计算模式下用户运行环境的可信性进行验证和评价,不需要用户维护相关信息或由用户自行安装搭建相应的设施对其云计算运行环境的可信性进行验证和评价;
- 提出一种周期性验证方案,并对文件系统和内存进行完整性验证,来保证运行环境的文件系统和运行程序在运行时的可信性.而现有大多数相关工作只能保证程序和系统在启动或迁移时的可信性;
- 基于 TCEE 机制构建原型系统,并通过实验的方法对其性能和引入的额外代价进行定量评价,可为实际应用及部署做参考.

1 相关工作

目前,构建可信运行环境的方法可以分成以下两类:

(1) 第 1 类是将 VMM 作为可信任基 TCB(trusted code base),即设计一个可信的虚拟机管理器 TVMM (trusted virtual machine monitor)[5],然后在此基础上构建用户可信机制.

Garfinkel 等人设计和实现的 Terra 系统[12]使通用平台支持多种系统而为应用程序提供多样的运行环境,但

© 中国科学院软件研究所 http://www.jos.org.cn

其原型系统是在非开源的商用产品 VMware GSX 上实现的,不方便对 VMM 本身进行改造.

邹德清等人[13]基于 Xen 设计和实现了一种可信的 VMM,其假定云服务提供商是可信的,而 SaaS 云的服务提供商是不可信的.用户可以按照完整性要求保存敏感数据,使得用户信任的程序才能访问敏感数据,并且这种方案还提供了内存保护机制.该方案包括跟踪模块(tracer module,简称 TAM)、系统调用跟踪器(system call tracer,简称 STC)和决策引擎(decision-making engine,简称 DME).但是由于需要修改 VMM,这带来了很高的复杂度,且不具有通用性.

Khan I 等人[14]将可信计算集成到云基础设施 Eucalyptus 中,用户通过远程验证虚拟机的完整性检查其可信性.尤其是通过验证存储控制器(storage controller,简称 SC)的完整性,可以保证 VM 绑定的虚拟存储环境也是可信的.但是由于数据的动态变化,这种完整性检测无法真正保护数据的完整性、隐私性等,因此对用户数据的保护不在其考虑范围,用户可以采取加密等方式对敏感信息进行保护.

Krautheim[15]提出一种针对云计算的管理和安全模型 PVI(private virtual infrastructure),并给出该云计算安全模型的原则以及一些思路,通过划分云服务提供商和用户各自在云计算安全性中的职责,以保证云的安全与可信.PVI 云安全架构主要有两层:下层是 IaaS 基础设施层,主要提供计算资源,由云提供商保证它的安全性;上层是 PVI 层,主要提供一个虚拟的数据中心,由用户自己管理.

(2) 另外一类工作在现有可信链的基础之上,通过扩展可信链,将可信传递到虚拟机内部.这类方法一般通过为虚拟机提供虚拟可信模块 vTPM(virtual trusted platform module)[9]作为虚拟机的可信任根.但是,绝大多数相关工作没有考虑云提供商本身给用户运行环境带来的可信性威胁,同时也需要用户自己保存程序及应用的完整性信息.

Berger 等人[9]针对 Xen 设计和实现了一种对 TPM 虚拟化,他们为每个虚拟机创建一个 vTPM 实例,通过 dom0 中的 vTPM manager 管理 vTPM 实例,并响应虚拟机发出的 TPM 请求.然而,这种软件保护加密密钥的方式提供像硬件 TPM 一样的安全能力.

England 等人[16]通过虚拟机管理器,采用虚拟机共享 TPM 的技术.但这种方式的不足是:可信链(从物理 TPM 到虚拟机中的应用)很长,且在验证时部分可信链需要重复验证.Kursawe 等人[17]指出,当前 TPM 的实现过于复杂,这与设计 TPM 的初衷背道而驰.他们对可信的边界重新定义,并实现了基于硬件的 uTPM,使用更加灵活,实现和认证也更加简单,而且 uTPM 支持多个执行环境(uTPM 进程).但是,基于 HEK(hardware endorsement key)对远程认证的数据做签名,容易暴露 uTPM 的硬件信息.

Stumpf 等人[18]提出另外一种复用硬件 TPM 的方式,通过 TPM 控制模块,在硬件 TPM 中为每个虚拟机构建各自的环境,使得每个虚拟机对硬件 TPM 的使用不会对其他虚拟机的 TPM 状态造成影响,从而达到 TPM 复用的目的.这种方式同样使可信链非常长,而且虚拟机的可信链关系复杂.

此外,刘孜文等人[19]指出,静态完整性度量无法保证运行时的完整性,并基于虚拟化技术和 IMA(integrity measurement architecture)提出一种基于可信计算的动态度量架构. Bertholon 等人[20]针对 IaaS 云平台中的可信和完整性问题,提出一种基于 TPM 模块的解决办法.主要包括两个协议:TCRR(TPM-based certification of a remote resource)和 VerifyMyVM.TCRR 协议基于硬件 TPM,为用户提供验证远程物理系统完整性和分发对称密钥的方式;VerifyMyVM 协议基于虚拟 TPM,在 TCRR 保证虚拟机管理器完整性的前提下,允许用户以请求的形式验证虚拟机的完整性.

2 TCEE:用户运行环境可信性动态验证机制

在云计算模式下,用户运行环境是以虚拟机为核心的,TCEE 的最终目标是给用户构建可信的运行环境.用户虚拟机典型不可信因素,例如虚拟机镜像,可能在传输过程中遭受中间人攻击,运行的虚拟机实例也可能被替换或者迁移到恶意的服务器中等.但是本文的威胁模型也考虑到云提供商的物理资源可能遭受的攻击或错误配置,如物理服务器的组件(如硬件、BIOS、Bootloader、操作系统、虚拟机管理器等)在启动过程或运行时,可能被错误配置或恶意修改;甚至存在恶意的云提供商管理员,他们通过在物理服务器中安装木马、后门程序,以

© 中国科学院软件研究所 http://www.jos.org.cn

获取云系统或用户的数据.

然而,对于拥有物理服务器的物理权限而采取的攻击手段,例如冷启动攻击(cold boot attack),甚至是篡改硬件,不是本文考虑的威胁范围.这是因为云平台在实际运营时,云提供商不会分配这种具备所有权限的管理员.同时,云提供商也会采取严格的访问控制策略保护物理硬件的完整性.所以,我们假设云提供商自己可以防止需要物理访问服务器的攻击手段.

2.1 体系结构

TCEE 通过 vTPM[9]把传统的可信任链扩展到虚拟机管理器 VMM(virtual machine monitor)以及用户的虚拟机(guest virtual machine)体系结构栈,如图 1 所示.由配置信息收集模块负责收集用户虚拟机环境中运行程序、文件和网络的运行信息,并通过远程验证服务模块将配置信息收集模块记录的日志发送到 TTP,最后由 TTP 基于完整性验证方法,把用户虚拟机的当前配置状态和运行状态与事先登记或记录在 TTP 中正常的状态数据进行对比,根据对比结果即可判断用户运行环境的可信性.

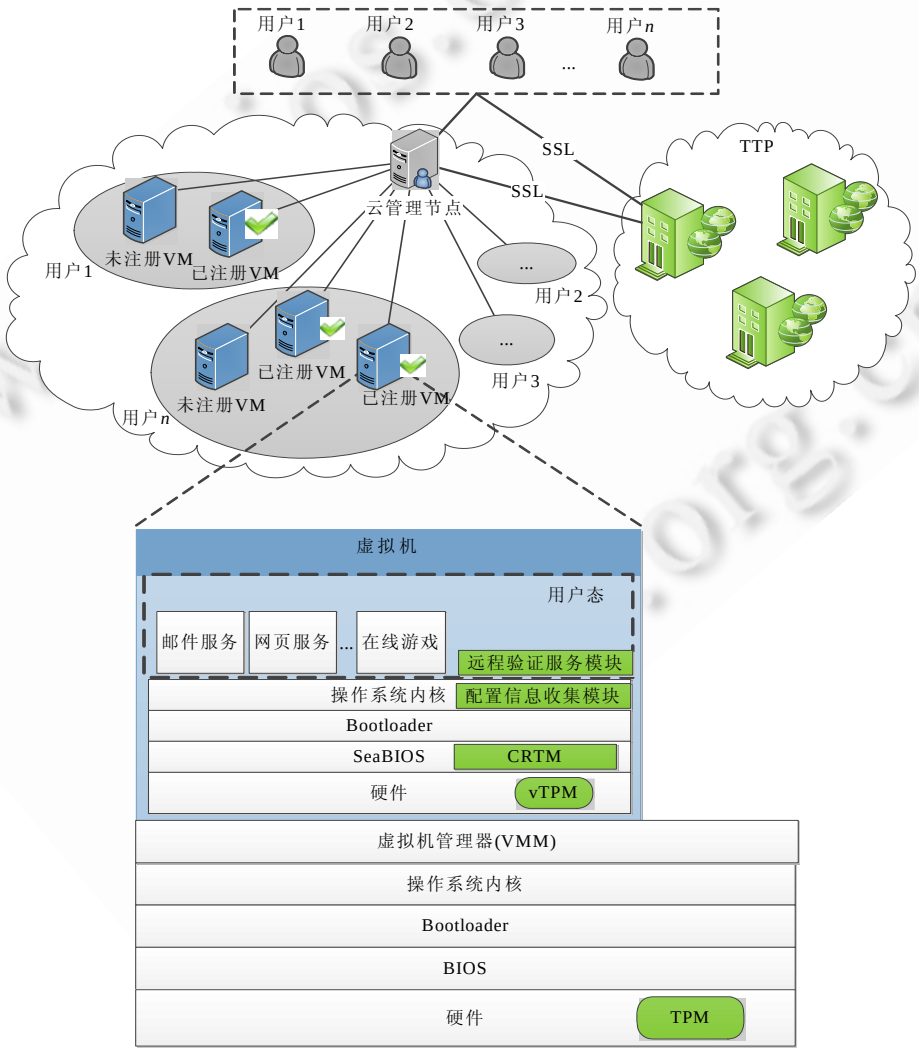


Fig.1 Dynamic verification model for trusted execution environment of cloud tenants

图 1 云用户运行环境可信性动态验证模型

© 中国科学院软件研究所 http://www.jos.org.cn

为解决 TTP 的单点瓶颈和单点故障问题,可以将 TTP 建设成一个专门用于用户运行环境可信性审计与验证的私有云平台.TTP 的验证服务器、可信软件仓库服务器、通讯服务器等可以根据当前待验证虚拟机的规模、负载等快速、动态的资源伸缩,用尽可能少的物理资源审计和验证尽可能多的云提供商平台,并支持各类服务器的迁移、容灾等高可靠性保证.我们把这种方式称为“小云审大云”.如何构建一个专有云平台,已有较多成熟的经验和案例[7,8],这部分内容不是本文考虑的重点.

2.2 可信性验证机制

TCEE 可信性验证机制主要包含两个过程,即:可信证据收集过程和远程验证过程.证据收集过程可以进一步细分成:用户虚拟环境启动过程的可信控制和运行时可信证据收集.

使用 TCG(trusted computing group)[29]规范的可信计算技术,如图 1 所示,保证云平台物理服务器的可信启动.其过程从物理服务器加电自检开始,到操作系统内核镜像载入内存、VMM 启动完成为止.对于以虚拟机为载体的用户运行环境,使用 vTPM[9]作为虚拟机的可信任根,通过接力式的度量、执行机制对涉及的所有可执行代码进行可信度量保存,并将 hash 值扩展(extend)到 vTPM 实例的相应 PCR(platform configuration register)寄存器中.具体过程如图 2 所示,当虚拟机完成加电自检后,系统控制权交给核心可信度量根 CRTM(core root of trust measurement),CRTM 首先计算自身代码的 hash 值并 extend 到 PCR 0 中,然后计算操作系统启动程序(boot loader)的 hash 值并 extend 到 PCR 4.系统控制权转交给 Boot Loader 后,由其计算整个操作系统内核镜像的 hash 值并 extend 到 PCR 5 中,然后加载操作系统内核镜像,进入运行态.至此,也就保证了配置信息收集模块和远程验证服务模块的可信启动.

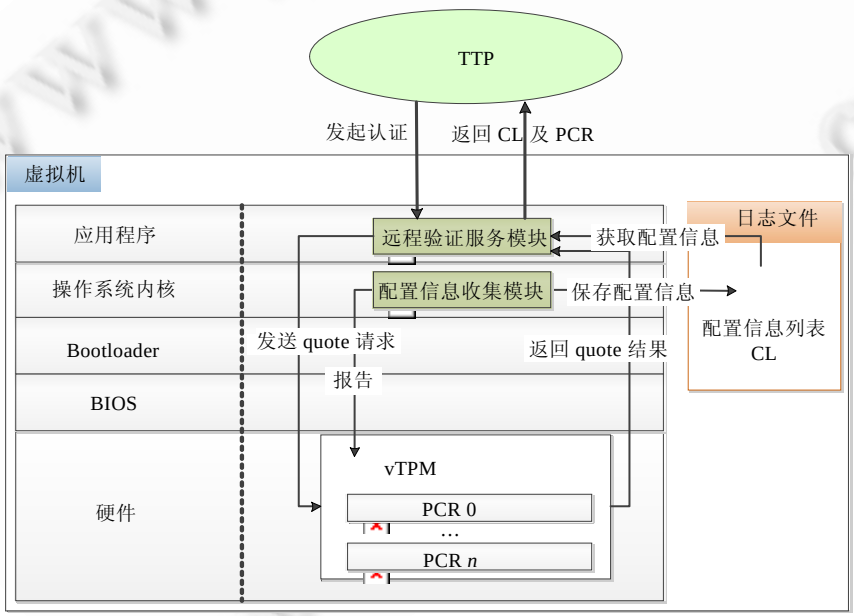


Fig.2 Trusted control for user virtual environments

图 2 用户虚拟环境启动过程的可信控制

运行时证据收集过程从操作系统内核运行开始,直到关机为止.如图 3 所示,在该阶段,利用配置信息收集模块作为可信度量代理,通过接力式的度量、执行机制,对涉及的所有可执行代码、打开的文件、载入的内核模块进行可信度量保存,并将其 hash 值追加(extend)到预设的 vTPM 静态 PCR 中,同时将上述被度量对象的描述信息、hash 值写入配置信息列表 CL(configuration list),并将配置信息列表保存到系统的安全位置.由于 PCR 的 extend 操作不可逆,所以它真实记录了所有运行程序可信任链,保证了远程验证时 CL 的完整性.以 Linux 操作系

© 中国科学院软件研究所 http://www.jos.org.cn

统内核为例,配置信息收集模块通过截获 mmap,bprm 以及 load-kernel 等系统调用函数来截获正在启动的程序、打开的文件、载入的内核模块(mmap 的截获函数用来计算系统中可执行程序和库文件的 hash 值,bprm 的 Hook 函数用来计算脚本文件以及一些重要配置文件的 hash 值,load-kernel 的 Hook 函数用来计算操作系统内核模块的 hash 值),然后再将 hash 值追加 extend 到 vTPM 的 PCR 10 中;同时把 hash 追加到 CL 文件中,并将 CL 保存在系统安全的位置,比如/sys/kernel/security 目录下.

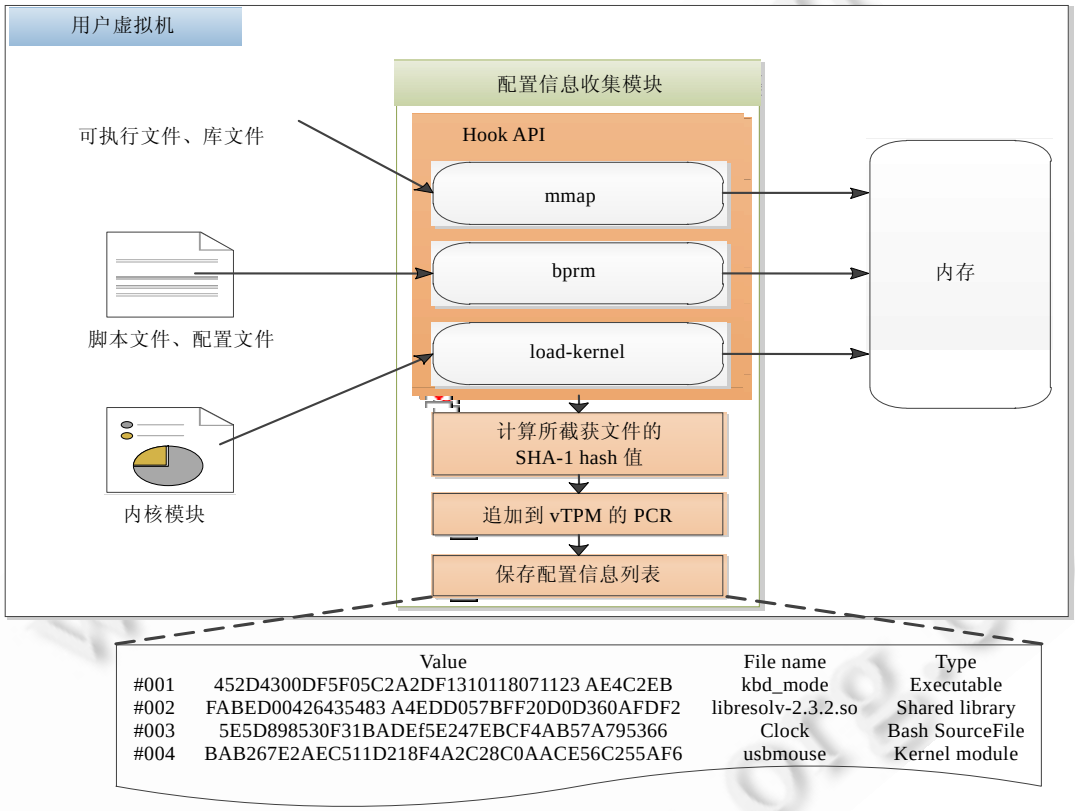


Fig.3 Trusted evidence collection mechanism at run time

图 3 运行时可信证据收集机制

2.3 可信性远程验证

利用事件(VM 启动迁移等)或计时器驱动 TTP 发起远程验证过程,其流程如图 4 所示:可信第三方 TTP 从用户运行环境的配置信息收集模块获取包括表现服务器启动运行过程的预定义行为模型、表现服务器当前状态的配置信息列表 CL 以及保证 CL 完整性的 vTPM 会话签名(每个用户的会话签名需要事先向 TTP 登记).利用 CL 驱动预定义的行为模型来验证物理机的启动顺序是否可信;根据 CL 检索匹配特征数据库,验证物理运行的程序、打开的文件和载入的内核模块等是否可信;根据用户的安全需求,诊断和审计用户环境存在的弱点和漏洞情况,判断其可信度.

具体来说,可信第三方 TTP 生成随机数,并将其发送给用户 VM 的远程验证服务模块;远程验证服务模块将接收到的随机数传入 vTPM,使用 vTPM 对当前 PCR 的值以及该随机数进行签名;远程验证服务模块将 vTPM 签名后的结果、会话签名 AIK 证书以及通过证据收集机制得到的配置信息列表 CL 返回给 TTP;TTP 接收到证据信息后,首先会验证 vTPM 的签名,确保所接收 PCR 的值和随机数的完整性,并验证该随机数与第 1 步中发送的随机数是否一致:如果不一致,表明接收的信息与本次验证无关;如果一致,再根据配置信息列表模拟 extend

© 中国科学院软件研究所 http://www.jos.org.cn

操作计算 PCR 的值,如果该值与返回的 PCR 的值一致,则表明配置信息列表是完整的、没有被修改的;最后,根据配置信息列表 hash 摘要,TTP 查询由其维护的可信软件数据仓库,以确定用户 VM 的当前状态,例如载入的内核模块、执行过的程序、打开过的文件等;如果所有这些内核模块、程序或文件是经过 TTP 认证的,那么表明 VM 是可信的;如果在可信软件数据仓库中没有相应记录,则是不可信的,或者云提供商需要向 TTP 注册登记这些程序或文件;如果在可信软件数据仓库中发现是恶意程序,则说明 VM 被攻破,正在运行上述恶意程序,不能被信任.

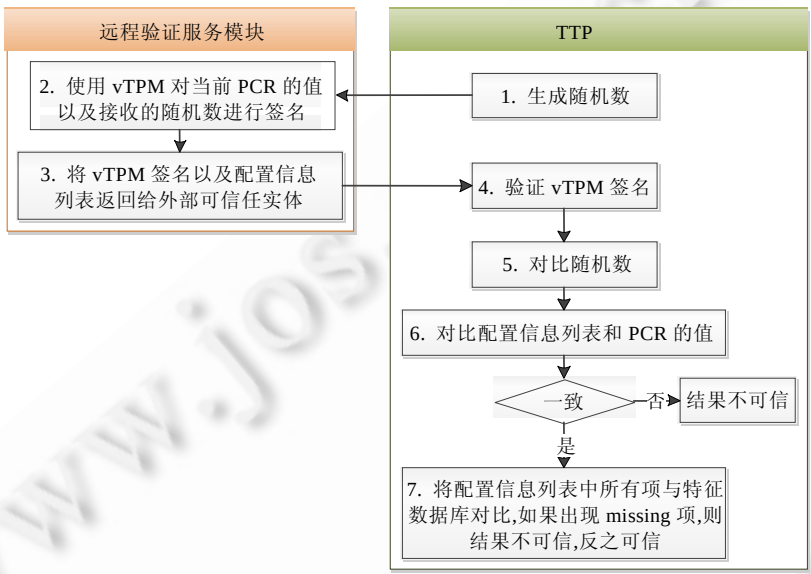


Fig.4 Remote verification mechanism on trust evidence

图 4 可信证据远程验证机制

TCEE 提出的远程验证机制具有以下特点:

(1) 将系统行为模型、完整性信息列表和系统安全性质结合在一起对系统进行可信验证,提高验证准确性.使用有限状态自动机(FSM)预定义系统的行为模型,行为模型中包括了一些系统需要满足的安全性质的先决条件以及可以产生的安全性质.首先验证云节点必须满足预定义行为模型和安全性质,然后使用 TCG 的可信计算标准收集云节点完整性信息,由可信第三方 TTP 进行验证.

(2) TTP 维护一个统一的动态可信软件数据仓库,解决了可信验证中用户软件升级问题,其中记录了各种软件的详细信息和完整性信息.在远程验证时,接收到云的完整性信息列表,经过可信计算验证数据可信后,与可信数据仓库进行对比,从而发现恶意程序.同时,云进行升级时,通知 TTP 与软件商合作验证升级软件确实可信之后,更新可信软件数据仓库,保证云端平滑升级.

(3) 现有的完整性验证方法只能做静态验证,即在操作系统启动时对文件系统做一次性的检查,很难保证整个运行环境的可信性.TCEE 提出一种周期性验证方法,它不仅可以检测出系统运行过程中的一些潜在风险,还可以对已加载到内存的应用程序做完整性验证.周期性验证方法具体如下:首先,设置一个初始检测周期,当程序加载到内存时,会计算该程序的 hash 值并保存至配置信息列表和 vTPM 的 PCR 中.在每个检测周期结束时,会重新检测内存中的程序,同样将结果保存到配置信息列表 CL 和 PCR 中,然后发送至 TTP 做验证.通过这种周期性的检测,以减少攻击对运行环境造成的损害.

2.4 周期性验证

现有的完整性验证方法只能做静态验证,即在操作系统启动后对文件系统做一次性的检查,很难保证整个

© 中国科学院软件研究所 http://www.jos.org.cn

运行环境的可信性.为解决上述问题,TCEE 提出一种周期性验证方法,它不仅可以检测出系统运行过程中的一些潜在风险,还可以对已加载到内存的应用程序做完整性验证.

周期性验证方法是:首先,设置一个初始检测周期.当程序加载到内存时,会计算该程序的 hash 值并保存至配置信息列表和 vTPM 的 PCR 中.在每个检测周期结束时,会重新检测内存中的程序,同样将结果保存到配置信息列表和 PCR 中,然后发送至 TTP 做验证.通过这种周期性的检测,以减少攻击对运行环境造成的损害.

3 实现和评价

3.1 原型系统实现

实现了基于 TCEE 的原型系统,其体系结构如图 5 所示.其中,使用软件 libtpm[27]仿真硬件 TPM 硬件;使用 QEMU+KVM 作虚拟机管理器;在 QEMU 中实现 vTPM 模块,支持对 TPM 硬件的仿真,这样,开启的虚拟机即拥有一个虚拟 TPM 硬件,并以此作为虚拟机的可信根;由 libtpm 充当 vTPM 的后端.

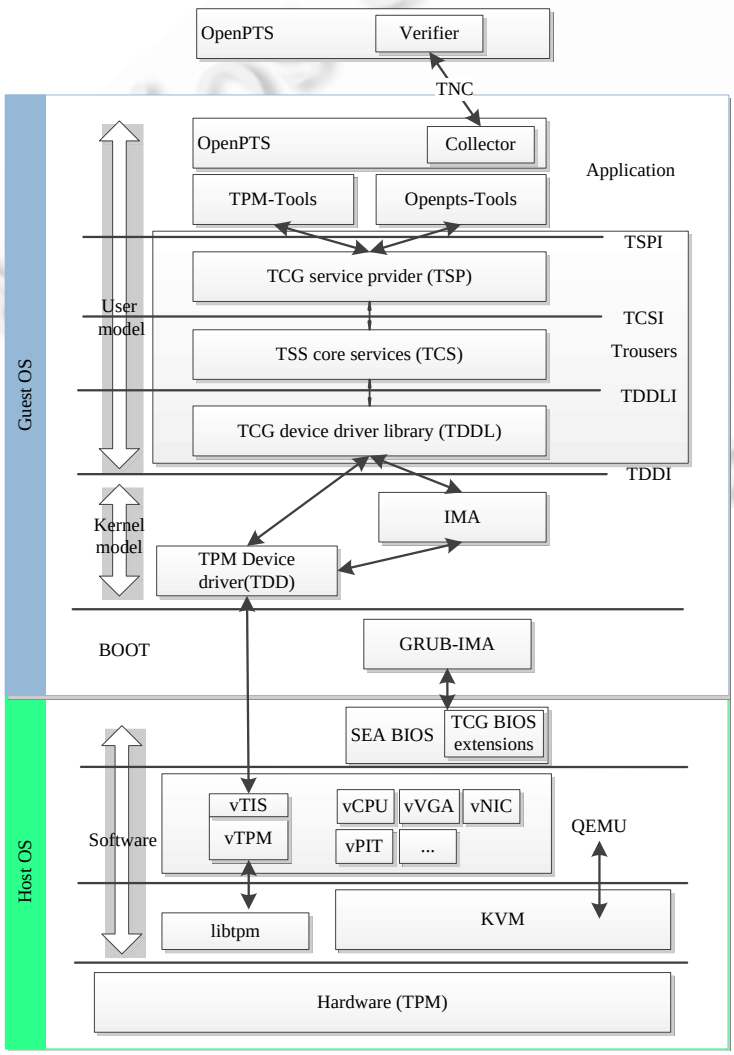


Fig.5 Prototype architecture based on TCEE mechanism

图 5 基于 TCEE 的原型系统体系结构

© 中国科学院软件研究所 http://www.jos.org.cn

使用 TCG 软件体系结构栈 TSS(TCG software stack)[10]构建原型系统的其他部分:TDD(TPM device driver)是 TPM 设备驱动程序,是唯一与 TPM 直接交互的模块;由 TDDL(TCG device driver library)调用,TDD 向下与 vTPM 的前端接口 vTIS(TPM interface specification)交互,调用 TPM 接口,使用 TPM 功能;SeaBIOS 是一个开源的 BIOS 实现,可以增加其他附加功能.在 TCEE 原型系统中,SeaBIOS 支持 TPM 初始化,支持 CRTM,支持 TCG BIOS 扩展等;GRUB-IMA 是一个添加了 TCG 标准规定的度量功能的 Linux 增强型 boot loader,可实现 TCG 标准的可信链启动过程.

使用 IBM 实现的完整性度量体系结构 IMA[26]进行运行时可信证据收集,在 openPTS[11]的基础上实现远程验证服务的客户端(以 openPTS 的 Collector 模块为基础)和服务器端(以 openPTS 的 Verifier 模块为基础).客户端部署在用户虚拟机环境中,服务器端部署在 TTP 中.远程验证主要分为 4 个子模块,即客户端初始化子模块,运行信息收集子模块,服务器端验证初始化子模块,可信性远程验证子模块.

3.2 实验和结果分析

实验和评价的目标包括两个方面:(1) 可信机制 TCEE 的有效性;(2) 可信机制 TCEE 给用户运行环境带来的额外代价.原型系统的部署配置信息见表 1.

Table 1 Configuration information for the TCEE prototype

表 1 TCEE 原型系统的配置信息

配置项	物理服务器	虚拟机
CPU	Core2 Duo T7700,2.40GHZ,共 24 核	Core2 Duo T7700,2.40GHZ
内存	32GB	4GB
二级缓存	4MB	4MB
硬盘容量	1TB	30GB

3.2.1 有效性实验及分析

选用 3 个典型的恶意程序或攻击来评价 TCEE 的有效性,其中:Wirenet.1[21]是用户态应用程序,以常用浏览器为目标,攻击 Mac OS X 系统以及 Linux 系统.一旦用户安装了这款恶意软件,它便能随时窃取用户在浏览器、邮箱或其他应用中输入的密码,它还能记录用户键盘按键信息;enyelkm[22]是内核态木马,其使用虚拟终端方式而不是普通的 shell 方式以躲避登录记录;第 3 个恶意软件是我们自己写的缓冲区溢出攻击程序(称为 buff_attack),其特殊性在于,它不是新创建一个进程或内核模块,而是改变现有进程的运行时行为.

如图 6 所示,当 Wirenet.1 加载和运行后,TTP 远程验证服务端通过对用户环境发起周期验证,得到可信性破坏的检测结果,其中,第 1 行显示该恶意程序的进程 ID、进程名称和所属用户,第 2 行是其十六进制摘要值;第 3 行是验证状态,显示该进程不可信.与此类似,对 Enyelkm 的检测结果如图 7 所示.

```
pid=29637, name="Wirnet.1", user=root
digest(hex)=0x1d80af672b98f9c64b0d91a74f64e92766fe3728
result=UNTRUSTED
```

Fig.6 Experimental results on Wirenet.1

图 6 Wirenet.1 验证结果

```
pid=27166, name="enyelkm", user=root
digest(hex)=0x5fa1b0118d39c514e922fae718d65e136f6ed901
result=UNTRUSTED
```

Fig.7 Experimental results on Enyelkm

图 7 Enyelkm 验证结果

我们进一步构造典型的运行时缓冲区攻击,以评价 TCEE 对于程序运行时攻击的有效性.在攻击开始前,被攻击进程的栈结构 main 函数只有一个 char 型数组 buffer,占 96 字节.利用 libc 的字符串函数和标准输入输出函数并不对数组做边界检查,因此会造成写入的数据超过 buffer 的长度,从而将函数的返回地址覆盖. buff_attack 向 buffer 数组中写入 128 字节数据,前 96 字节是 buff_attack 想要执行的 shell 脚本,后 32 字节是 buffer 的地址.这 32 字节将会覆盖原来的函数返回地址,当主函数返回时,会调用 buff_attack 注入的 shell 脚本.

如图 8 所示,通过 TCEE 运行时证据收集机制,可以在进行 buff_attack 攻击之前打印当前进程的返回地址,而在 buff_attack 攻击之后验证当前进程的返回值跟上一次记录的返回值是否一致,从而可以判断当前进程的

© 中国科学院软件研究所 http://www.jos.org.cn

栈返回值是否发生变化,如图 9 所示,进而检测 buff_attack 攻击.

```
root@yun:/home/lw/test#./stasmh &
[2] 3787
ret addr of main() is 00156e37
addr of buffer is bffff520
begin copy to overflow
copy end
root@yun:/home/lw/test#
```

Fig.8 Typical attack of stack overflow

图 8 栈溢出攻击实例

```
root@yun:/home/lw/dyIMA#./stack 3787
main bp=bffff588
ptrace read ret addr is bffff520
root@yun:/home/lw/dyIMA#
```

Fig.9 Results of the stack overflow attack

图 9 栈溢出攻击结果

3.2.2 性能实验及分析

当引入 TCEE 可信性验证和审计机制以后,在用户运行环境中配置信息收集模块会计算程序摘要值,并把摘要值存储到配置信息列表 CL 和 vTPM 的 PCR 寄存器中.所以,这会给程序的执行带来额外开销.因此,性能实验的目的是:通过对比引入 TCEE 的虚拟机相对于普通虚拟机的性能比较,分析 TCEE 带来的额外代价.为了更细致分析性能代价引入的原因,进一步把用户虚拟机分成 3 种设置,即:可信虚拟机(配置 TCEE 的完整架构,简称为 trust 类型)、半可信虚拟机(只在虚拟机中安装配置信息收集模块,而没有设置 vTPM 可信启动控制,简称为 semi-trust 类型)和非可信虚拟机(普通的虚拟机,没有进行 TCEE 机制的任何设置,简称为 none-trust 类型).

性能评价主要分为微观性能测试(micro-benchmark)和宏观性能测试(macro-benchmark).微观性能测试选择最常用的 10 个 Linux 命令进行测试,分别计算它们在相同机器配置的 trust 类型和 none-trust 类型虚拟机中运行时间比值,如图 11 所示,定义命令执行时间比值=普通虚拟机命令执行时间/可信虚拟机命令执行时间.宏观性能测试选用典型的 IO 密集型测试工具、计算密集型测试工具和混合类型工作负载(hybrid workload)测试工具对相同机器配置的 trust 类型、semi-trust 类型和 none-trust 类型虚拟机性能进行对比测试.测试工具的名称和说明见表 2.

Table 2 Description of the performance measurement tools used in this paper

表 2 性能测试使用工具介绍

名称	功能描述
IOZone[23]	可对多种文件操作进行测试,包括文件读、写、mmap、异步 IO,通过计算出吞吐率得到系统的 I/O 性能
BYTEmark[24]	采用 10 种计算密集型算法,根据每秒迭代次数测试系统的 CPU,FPU,memory 性能
PostMark[25]	测试文件系统在邮件系统或电子商务系统中的性能.这类负载的特点是频繁、大量地存取小文件. 其测试原理是:创建一个测试文件池,对文件池进行一系列的事务(transaction)操作,例如删除、读、写, 最后统计单位时间内完成事务操作的数量
TPCC-UVa[26]	遵循 TPCC 规范模拟联机交易处理系统(OLTP 系统),多个终端并发的向仓库发送交易请求, 依据仓库处理的请求数衡量系统的事物处理能力

如图 10 所示,测试的命令包括 who,vi,rm,more,ls,grep,find,cp,cd 和 cat.每个命令均进行了 1 万次的实验,计算平均执行时间.从图中可以看出,none-trust 类型执行时间都要比 trust 类型执行时间短,可信证据收集带来的代价最大增幅为 30%(find 命令),最小增幅为 11%(cd 命令).但在执行时间上二者的差距很小,最大差距也在 3ms 以下,对用户实际使用的影响可以忽略.

图 11 是使用表 2 的性能测试工具对不同类型的用户虚拟机运行环境进行性能测试和对比的情况.其中,对于 I/O 操作,TCEE 引入的额外性能代价在于可信证据收集和对 vTPM 的 extend 操作,这些代价主要作用于当文件被打开时对文件内容的度量,并将 hash 值 extend 到 vTPM 的 PCR 寄存器中;而对文件系统的读写性能影响很小.也即,TCEE 主要影响文件的打开延迟.但是,iozone 测试的文件 I/O 吞吐率并没有包括文件的打开时间,所以从图 11(a)中可以看出,3 种类型的用户虚拟机,其吞吐率是基本一致的.特别是随着文件大小的增加,磁盘 I/O 逐渐成为系统 I/O 链条的瓶颈后,可信机制引入的代价基本可以忽略,运行环境吞吐率表现为实际磁盘 I/O 速度,3 种类型的用户虚拟机吞吐率曲线几乎重合.

© 中国科学院软件研究所 http://www.jos.org.cn

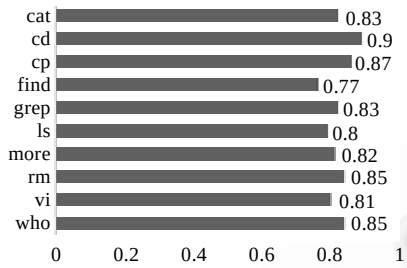
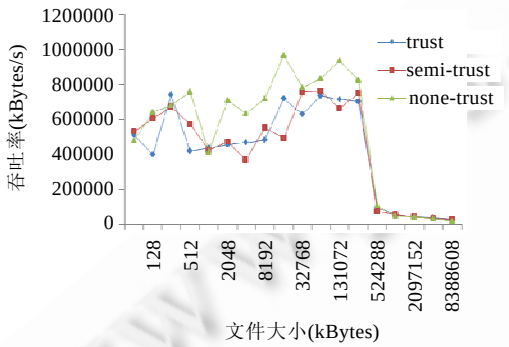
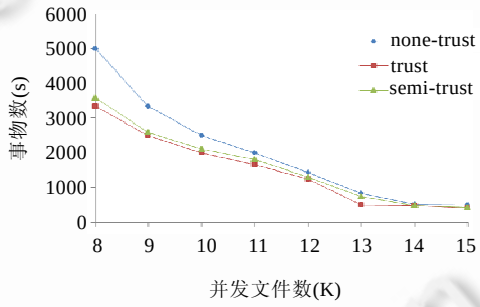


Fig.10 Performance overhead incurred by TCEE on typical system commands

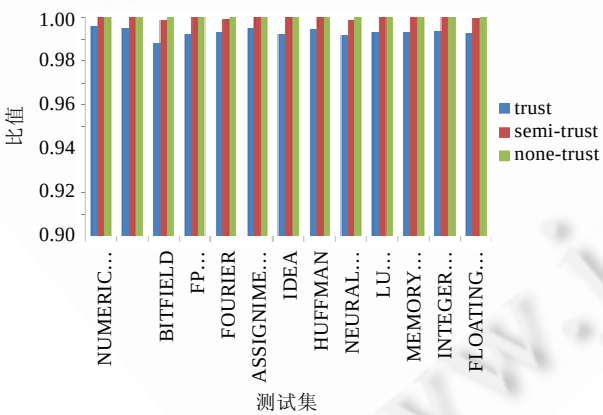
图 10 TCEE 对系统典型命令所引入的性能代价



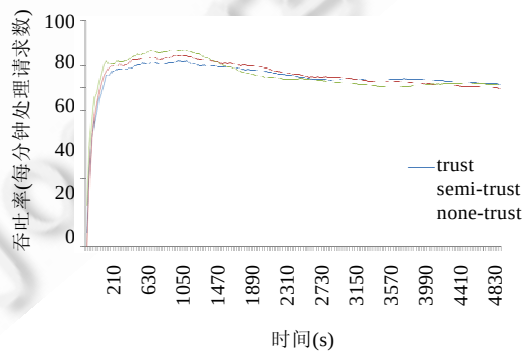
(a) 使用 IOzone,在特定数据块大小时(64KB), I/O 吞吐率随文件大小变化情况



(b) 使用 PostMark,文件事务处理能力随文件池数量变化情况



(c) 使用 ByteMark,MEMORY INDEX,INTERGER INDEX, FLOATING-POINT INDEX 这 3 个综合指标所代表的计算能力在不同 benchmark 下的比较



(d) 使用 TPCC-UVa,请求处理吞吐率随时间变化情况

Fig.11 Performance comparison of different user virtual machines taken by the measurement tools

图 11 使用性能测试工具对不同类型用户虚拟机运行环境进行性能对比

对于以 PostMark 为代表的文件事务型处理应用,因为在每一个事务操作时,可信证据收集模块都要记录事务操作进程运行代码的摘要值,写入到配置列表文件中,并同时 extend 到 vTPM 的 PCR 寄存器中,所以减慢了每

© 中国科学院软件研究所 http://www.jos.org.cn

个事务处理的时间,导致每秒完成的事务数量减少,如图 11(b)所示.进一步可以注意到,随着并发文件数的增加,二者之间的差距越来越小.因为若有相同的事务操作(如对同一个文件进行读操作),则可信证据收集模块不必将摘要写入到 CL 和 PCR 中,节省了时间;当并发文件增多时,相同的事务操作相对增多,则由可信系统带来的额外耗时相对减少,所以导致其事务处理能力差距变小.所以,越是大规模的文件处理,可信带来的额外代价相对越小.

如图 11(c)所示,对于计算密集型应用,一般而言,可信虚拟机计算能力低于半可信虚拟机,半可信虚拟机计算能力低于非可信虚拟机.这是因为在运行每种计算程序时,可信证据收集模块都要记录运行程序 hash 值并 extend 到 TPM 中(半可信虚拟机只需记录运行程序的 hash 值),这些操作都会消耗一定时间,从而导致得到计算能力下降.但这 3 种物理机计算能力差别很小,但是由于只在程序开始运行前才会计算运行程序代码的 hash 值,并 extend 到 vTPM 寄存器中,而当程序运行后,则与正常的虚拟机相同,所以从图 11(c)可以看到,3 种类型虚拟机的计算性能差别很小(小于 2%性能代价).因此,对于计算密集型任务,TCEE 可信机制不会引入过多额外代价.

TPCC-UVa 是遵循 TPCC 规范的模拟联机交易处理系统 OLTP(online transaction processing)测试工具,用以评价 TCEE 机制引入的数据分析性能损耗.实验中,TPCC-UVa 配置为 1 小时测试时间,9 个仓库,120s 启动时间,无数据库清理操作.如图 11(d)所示.前 120s 终端逐渐启动,系统的吞吐率随之快速增加;达到系统处理峰值后,由于没有进行数据库清理操作,数据库的残留信息会影响系统处理能力,吞吐量缓慢下降.跟 PostMark 类似,可信虚拟机的吞吐率在达到峰值前,比非可信虚拟机的吞吐率低;而达到峰值后,由于非可信虚拟机积累的残留数据较多,性能下降较快.

4 总 结

给用户提供一个可证明、可验证的可信运行环境,是云计算模式面临的重要问题.本文提出一种面向云计算模式的运行环境可信性动态验证机制 TCEE,通过引入可信第三方 TTP,该机制可周期性(可以根据现阶段的攻击状况动态调整下一阶段的检测周期)检测内存和文件系统中的攻击,针对用户虚拟机运行环境的可信性进行远程验证和审计,避免由用户维护可信验证的相关信息和机制,同时也能够避免云平台敏感信息的泄露.实现基于 TCEE 的原型系统,通过对典型的攻击和威胁进行测试和评价,验证了 TCEE 的有效性;通过使用典型的性能测试工具进行对比测试,TCEE 引入的额外性能代价可接受.

References:

[1] Chen Y, Paxson V, Katz R. What's new about cloud computing security? Technical Report, UCB/EECS-2010-5, Berkeley: University of California at Berkley, 2010.

[2] Ko RKL, Jagadpramana P, Mowbray M, Pearson S, Kirchberg M, Liang QH, Lee BS. TrustCloud: A framework for accountability and trust in cloud computing. In: Proc. of the 2nd IEEE World Congress on Services. 2011. 584–588. [doi: 10.1109/SERVICES.2011.91]

[3] Jansen W, Grance T. Guidelines on Security and Privacy in Public Cloud Computing. NIST, 2011.

[4] Marco S. BlackHat presentation demo vids: Amazon, part 4 of 5, AMIBomb. 2009. http://www.sensepost.com/blog/3797.html

[5] Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. Above the clouds: A Berkeley view of cloud computing. Technical Report, UCB/EECS-2009-28, Berkeley: University of California at Berkley, 2009.

[6] Berger S, Cáceres R, Pendarakis D, Sailer R, Valdez E, Perez R, Schildhauer W, Srinivasan D. TVDc: Managing security in the trusted virtual datacenter. ACM SIGOPS Operating Systems Review, 2008,42(1):40–47. [doi: 10.1145/1341312.1341321]

[7] Jinesh V. Migrating your existing applications to the AWS cloud. 2010. http://www.Amazon.com

[8] Butler B. Eucalyptus: We're the amazon of private cloud companies. IDG Technical Report, 2012.

[9] Berger S, Cáceres R, Goldman KA, Perez R, Sailer R, van Doorn L. vTPM: Virtualizing the trusted platform module. In: Proc. of the 15th Conf. on USENIX Security Symp. 2006. 305–320.

[10] Trusted Computing Group. TCG Software Stack (TSS) Specification—Version 1.2 Golden. 2007.

© 中国科学院软件研究所 http://www.jos.org.cn

[11] OpenPTS. http://sourceforge.jp/projects/openpts

[12] Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D. Terra: A virtual machine based platform for trusted computing. In: Proc. of the 9th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2003. 193−206. [doi: 10.1145/1165389.945464]

[13] Cheng G, Jin H, Zou DQ, Zhang XW. Building dynamic and transparent integrity measurement and protection for virtualized platform in cloud computing. Concurrency and Computation: Practice and Experience, 2010,22(9):1893−1910. [doi: 10.1002/cpe.1614]

[14] Khan I, Rehman H, Anwar Z. Design and deployment of a trusted eucalyptus cloud. In: Proc. of the IEEE Int'l Conf. on Cloud Computing. Washington: IEEE, 2011. 380−387. [doi: 10.1109/CLOUD.2011.105]

[15] Krautheim FJ. Private virtual infrastructure for cloud computing. In: Proc. of the 2009 Conf. on Hot Topics in Cloud Computing. USENIX Association, 2009.

[16] England P, Loeser J. Para-Virtualized TPM sharing. In: Proc. of the 1st Int'l Conf. on Trusted Computing and Trust in Information Technologies. 2008. 119−132.

[17] Kursawe K, Schellekens D. Flexible uTPMs through disembedding. In: Proc. of the ACM Symp. on Information, Computer and Communications Security. 2009. 116−124.

[18] Stumpf F, Eckert C. Enhancing trusted platform modules with hardware-based virtualization techniques. In: Proc. of the 2nd Int'l Conf. on Emerging Security Information, Systems and Technologies. 2008. 1−9.

[19] Liu ZW, Feng DG. TPM-Based dynamic integrity measurement architecture. Journal of Electronics & Information Technology, 2010,32(4):875−879. [doi: 10.3724/SP.J.1146.2009.00408]

[20] Bertholon B, Varrette S, Bouvry P. CERTICLOUD_a novel TPM-based approach to ensure cloud IaaS security. In: Proc. of the 4th Int'l Conf. on Cloud Computing. 2011. 1−8.

[21] http://news.drweb.com/show/?i=2679&lng=en&c=14

[22] http://www.enye-sec.org

[23] IOzone. http://www.iozone.org/

[24] http://www.tux.org/~mayer/linux/bmark.html

[25] https://communities.netapp.com

[26] Llanos DR. TPCC-UVa: An open-source TPC-C implementation for global performance measurement of computer systems. SIGMOD Record, 2006,35(4):6−15. [doi: 10.1145/1228268.1228270]

[27] Reiner S, Zhang XL, Trent J, Sailer R, Zhang XL, Jaeger T, van Doorn L. Design and implementation of a TCG-based integrity measurement architecture. In: Proc. of the USENIX Security Symp. 2004.

[28] Mario S, Stamer H. A software-based trusted platform module emulator. In: Proc. of the Trusted Computing-Challenges and Applications. Berlin, Heidelberg: Springer-Verlag, 2008. 33−47. [doi: 10.1007/978-3-540-68979-9_3]

[29] Trusted Computing Group. http://www.trustedcomputinggroup.org

刘川意(1982－),男,四川乐山人,博士,讲师,主要研究领域为计算机系统结构,云计算与云安全,数据管理与数据保护.
E-mail: cy-liu04@mails.tsinghua.edu.cn

唐博(1987－),男,硕士,主要研究领域为云计算与可信计算.
E-mail: tangbo1987@gmail.com

林杰(1987－),男,博士生,主要研究领域为计算机网络,信息安全.
E-mail: jie_lin@bupt.edu.cn

© 中国科学院软件研究所 http://www.jos.org.cn