可信的云计算运行环境构建和审计

刘川意[1),3)] 王国峰[2),3)] 林 杰[2),3)] 方滨兴[2),3)]

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

摘 要 可信的云计算运行环境很大程度决定了云服务的推广和有效使用.采用可信计算技术的基本假设是所有在运行环境中安装的内核模块和应用程序都需要事先指定并已知其执行代码的完整性信息.这个假设在云用户实际使用时往往是不可能的.所以该文提出了一种将虚拟可信根 vTPM 和可信审计技术结合起来的用户可信运行环境构建与审计机制.对于体系结构栈中可事先固定的、在启动和运行时不变的组件,通过 vTPM 作可信根保证可信启动;对于运行过程中变化的、无法事先确定的组件,文中提出了一种可信审计的方法,即对用户虚拟机的运行态环境进行证据收集和证据审计,监测实际使用中用户运行环境的可信性.若用户运行环境处于不可信状态,则及时通知用户采取相应措施.最后文中基于该可信机制的原型系统对其有效性和性能代价进行定量测试和评价,实验结果表明:该机制针对典型的不可信威胁是有效的,且引入的性能代价对用户实际使用的影响可以忽略.

关键词 云计算;可信审计;虚拟可信平台模块;可信计算

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2016.00339

Practical Construction and Audit for Trusted Cloud Execution Environment

LIU Chuan-Yi[1),3)] WANG Guo-Feng[2),3)] LIN Jie[2),3)] FANG Bin-Xing[2),3)]

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

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

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

Abstract Trustworthiness is a critical factor for the large-scale use of cloud services. However, traditional Trusted Computing technologies only work with the assumption that all the kernel modules and user-space processes installed in the tenant's virtual machine should be pre-fixed and the integrity digests should be pre-confirmed. This paper proposes a practical mechanism for trusted cloud environment construction and audit, which combines virtual Trusted Platform Module (vTPM) and trust evidence audit. vTPM guarantees the trusted boot up, while the audit method check the trust status of the run-time virtual machine based on auditing the trusted evidence collected from the memory of the user virtual machine. A prototype is also implemented according to the above mechanism. Experimental results show that the mechanism is effective and the performance overhead incurred is minor.

Keywords cloud computing; trust audit; virtual trusted platform module; trusted computing

收稿日期:2013-08-27;在线出版日期:2015-12-18.本课题得到国家自然科学基金(61202081)资助.**刘川意**,男,1982 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为云计算与云安全、数据安全与数据保护.E-mail:cy-liu04@mails.tsinghua.edu.cn.**王国峰**,男,1988 年生,博士研究生,主要研究方向为云安全.**林 杰**,男,1987 年生,博士研究生,主要研究方向为计算机网络、信息安全.**方滨兴**,男,1960 年生,教授,中国工程院院士,主要研究领域为信息与网络安全、内容安全.

1 引 言

云服务的推广使用需要云计算的可信性[1-2].但云计算提供给用户(Tenant)的运行环境是以虚拟机作为载体的[3],用户的运行环境和数据都存放在云端,从而失去了对物理环境的直接控制,云计算是否可信是云服务需要面对的重要问题.如何为用户提供可信的云计算服务,总结起来要面临的技术挑战如下:

(1)在云计算模式下,控制权在云提供商(Cloud Provider)和用户之间进行了分割.云提供商单方面申明可信很难让用户信服.所以用户运行环境的可信性,对于用户或独立第三方而言应该是可验证的.云提供商为了吸引潜在用户,也倾向于证明自己是可信的.然而,用户以虚拟机终端远程连接虚拟机进行管理和使用,或管理运行在云提供商的平台之上的程序或任务.其对运行虚拟机的物理硬件或云平台信息的了解是有限的,更很难知晓云平台是怎样组织和实现的.

(2)传统的可信计算(Trusted Computing)技术利用可信任基和可信任链,通过审计验证可保证服务器的启动过程是安全可信的.但云计算面临的问题在于:用户的运行环境以虚拟机为载体,但仅仅保证服务器的可信启动是不够的,还要保证虚拟机运行环境安全可信,没受到恶意篡改或窃听.为了解决此问题,一种方法是扩展可信任链(Trusted Chain),将其扩展到虚拟机内部,如 vTPM (virtual Trusted Platform Module)[4].然而,扩展可信任链的方法基于一个基本假设,即所有在用户环境中安装的内核模块和应用程序都需要事先指定并已知其执行代码的完整性信息(多是对内核或应用程序的执行代码计算其 Hash 摘要值).这个假设在云用户实际使用中往往不成立.

针对以上挑战,本文提出了一种将虚拟可信根 vTPM 和可信审计技术结合起来的用户可信运行环境构建与审计机制.通过 vTPM 可信根,保证虚拟机在启动过程中 BIOS(Basic Input Output System)、启动程序、操作系统内核等体系结构栈底层的完整性;通过可信审计的方法对用户虚拟机的运行态环境进行证据收集和证据审计(什么时候收集证据和进行可信审计可以根据实际情况制定相应策略,如周期性或事件触发的收集和审记),及时检测在实际使用中无法事先固定的用户运行环境的可信性.若用户运行环境处于不可信状态,则及时通知用户,并采取相应措施.

本文第 2 节介绍和分析国内外关于云计算可信运行环境的相关研究工作;第 3 节详细描述本文设计的可信的用户运行环境的构建和审计机制;第 4 节给出基于上述机制的原型系统实现过程以及对上述机制的有效性和引入的性能代价进行实验评价和分析;第 5 节对本文工作进行总结.

2 相关工作

对于如何构建可信的云计算运行环境,目前主要有两类工作.第 1 类工作是从虚拟机管理器出发,设计 TVMM(Trusted Virtual Machine Monitor)[3],把虚拟机管理器作为可信任基 TCB(Trusted Code Base),然后扩展可信任链(Trusted Chain),将其扩展到虚拟机内部.

Terra 系统[5]在可信的虚拟机管理器上运行不同的虚拟机系统,使得不同应用程序的运行环境也不相同.该系统由 Garfinkel 等人[5]在非开源商用虚拟机管理器 VMware GSX 上设计实现,不开源,不利于推广.

PVI(Private Virtual Infrastructure)是一种新的策略管理安全模型,由 Krautheim[6]主要针对云计算提出,这种管理与安全模型通过划分云计算中服务提供商和客户的安全责任,降低了各自的安全风险.

Khan 等人[7]利用 Eucalyptus 云平台,通过远程验证虚拟机尤其是存储控制器(SC)的完整性来保证与 VM(Virtual Machine)绑定的虚拟存储环境也是可信的.这种可信计算机制不考虑对用户数据的保护,并且由于数据的动态变化,从而该机制无法真正保护数据的完整性、隐私性等.

Cheng 等人[8]基于 Xen,在假定云提供商可信、云的服务提供商不可信的前提下,设计和实现了一种可信的虚拟机运行平台.用户可以按照完整性要求保存敏感数据,只有用户信任的程序才能访问敏感数据,并提供内存保护机制.但是这种方案需要修改 VMM(Virtual Machine Monitor),故带来很高的复杂性,且不具有通用性.

第 2 类工作是扩展可信任链(Trusted Chain),在可信的服务器基础之上,将可信任链扩展到虚拟机内部,如 vTPM(virtual Trusted Platform Module).

Berger 等人[4]通过 Xen 虚拟机管理器,利用硬

件虚拟化技术在虚拟化平台上为每个虚拟机创建一个虚拟的 TPM 实例，在 dom0 端通过虚拟 TPM 管理器管理虚拟机中虚拟的 TPM 实例，并响应其发出的请求.

Stumpf 等人[9]通过硬件 TPM(Trusted Platform Module)复用的方式，在硬件 TPM 上为每个虚拟机构建各自不会相互干扰的 TPM 使用环境. 这种方式使机制关系非常复杂，可信任链加长.

Paul 等人[10]采用虚拟机共享 TPM 的方式，通过修改虚拟机管理器实现，导致可信任链较长，其验证比较繁琐. Kursawe 等人[11]指出设计和实现 TPM 应尽量简单，认证方式不能过于复杂. 他们重新定义可信边界，设计和实现了 uTPM，降低了使用的复杂度，并且支持不同的运行环境. 但这种方式中部分数据容易暴露 uTPM 的硬件信息，如远程认证签名数据.

除以上工作外，国内外学者还做了其他相关的研究工作，Liu 等人[12]基于完整性度量架构技术 IMA(Integrity Measurement Architecture)[13]设计了一种动态度量架构，可以度量虚拟机运行时的完整性. Bertholon 等人[14]基于 TPM 模块实施方案应对 IaaS 云平台中的可信和完整性问题，提出了两个协议，前者协议为基于硬件 TPM 的远程资源验证协议 TCRR(TPM-based Certification of a Remote Resource)，保证虚拟机管理器的完整性；后者协议为基于虚拟 TPM 的 VerifyMyVM 协议，验证虚拟机的完整性. 但上述方法需要改动虚拟机，在虚拟机中安装内核模块和用户程序，且与用户虚拟机操作系统相关，不利于推广.

3　用户可信运行环境构建与审计机制

在云计算模式下，虚拟机是运行用户程序或任务的主要载体，所以保证虚拟机运行环境的可信性才能保证云计算的可信性. 影响用户运行环境可信性的典型因素如：虚拟机镜像遭受篡改和破坏；运行的虚拟机实例被替换或者迁移；用户虚拟机中安装有恶意的内核级或用户级程序；虚拟机实例受到外来攻击等.

而云提供商为了争取用户而趋向于为用户环境的可信性提供支持和证据，所以本文跟同类工作采用同样的假设[4,13,15]，即：云提供商的物理服务器是可信的，不能随意安装恶意程序，同时云提供商的物理硬件也是安全可信的，没有被恶意篡改，并且有严格的控制策略保护其完整性，防止如冷启动攻击(Cold boot attack)等硬件攻击.

3.1　体系结构

可信的云用户运行环境主要涉及两个机制，即可信的体系结构栈构建和运行时可信性审计. 前者通过扩展可信任链，在可信虚拟机管理器 VMM(Virtual Machine Monitor)基础之上，通过 vTPM 将可信机制传递到虚拟机内部，即用户使用的虚拟机运行环境；后者则在物理服务器 Host 中部署用户信任的虚拟机内省 VMI(Virtual Machine Introspection)[16]和审计策略定制模块，由 VMI 将被审计虚拟机的内存内容旁路和备份，并由审计策略定制模块进行审计分析. 图 1 以 Xen[17]虚拟机管理器为例，展示了用户可信运行环境构建与审计的体系结构. 其中，由物理服务器中部署的 TPM 芯片作为可信任根，通过软件 vTPM 将可信任链由物理服务器扩展到 VMM，从而进一步延伸到虚拟机体系结构栈中. 但是因为用户运行环境可能是动态改变的(如软件升级、安装新软件、虚拟机配置改变等)，所以另一方面，用户虚拟机运行时的环境是否可信是由审计策略定制模块 Auditor 发起对用户虚拟机的审计请求，由虚拟机内省 VMI 模块通过对相应虚拟机内存数据进行获取和拷贝，交由审计策略定制模块进行解析和分析的.

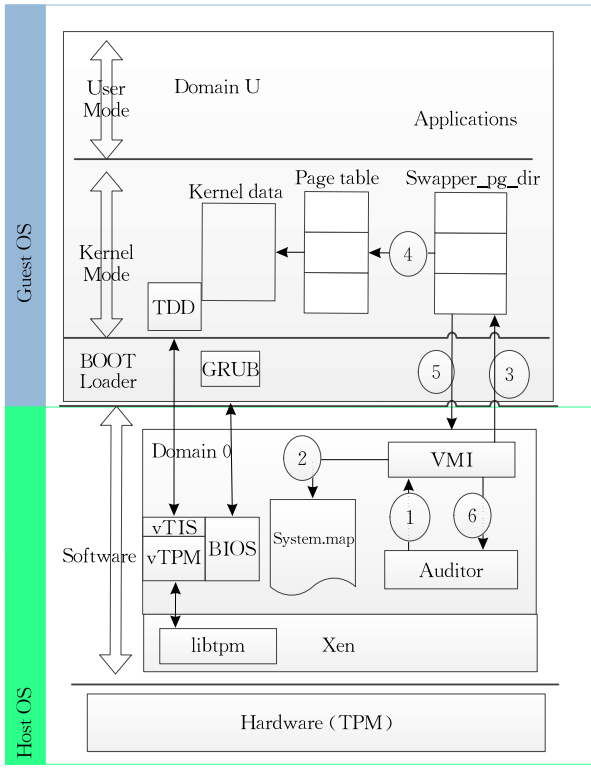


图 1　用户可信运行环境体系结构

3.2 基于可信任链的用户环境启动过程

使用 TCG 规范的可信计算技术[18]保证了云平台物理服务器的可信启动.

如图 2 所示为用户虚拟机的可信启动过程,首先是可信的基于硬件 TPM 的服务器,扩展可信任链到虚拟机内部. 可信任链的传递依次经过物理服务器、操作系统内核、虚拟机管理器. 用户以虚拟机终端远程连接虚拟机进行管理和使用,或管理运行在云提供商的平台之上的程序或任务. 为了保证用户的虚拟机运行环境是可信的,使用虚拟 TPM 作为虚拟机的可信根,通过哈希度量所有涉及的可执行代码,将度量哈希值扩展到虚拟 TPM 相应的 PCR(Platform Configuration Register)寄存器中. 当虚拟机启动后,核心可信度量根 CRTM(Core Root of Trust Measurement)首先计算自身代码的哈希值并存储到 PCR0 寄存器中,然后验证操作系统启动程序(Boot loader)的完整性,计算其哈希值并存储到 PCR4 寄存器中. 系统控制权转交给 Boot loader 后,再由其计算整个操作系统内核镜像的完整性,并将哈希值保存到 PCR5 寄存器中,最后加载操作系统内核,进入运行状态. 通过以上过程也就保证了用户虚拟机的可信启动.

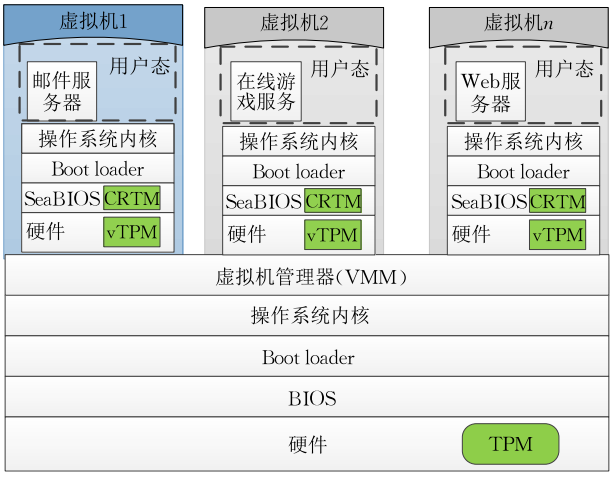


图 2 用户虚拟环境启动过程的可信任链依赖与控制转移过程

3.3 用户环境运行时可信证据的收集与审计

当需要进行可信证据收集时,策略定制模块 Auditor 发送命令给虚拟机内省模块,获取可信证据. 如图 1 所示,该流程主要包括以下步骤.

(1)策略定制模块 Auditor 发送命令给虚拟机内省模块.

(2)虚拟机内省模块从本地的内核符号表中读取内核的虚拟地址信息,如果本地内核符号表与虚拟机的内核不匹配,则通过网络查询对应的内核符号表.

(3)根据虚拟机 CR3 寄存器的内容进行寻址转换操作,映射到页表目录(Page Directory),如图 3 所示.

(4)由页表目录映射到页表(Page Table)最终接入虚拟机管理器,从而得到所需的内存数据.

(5)得到的内存数据以二进制的形式逐级返回给虚拟机内省模块.

(6)数据传送给模块 Auditor 并保存下来,作为审计时的证据.

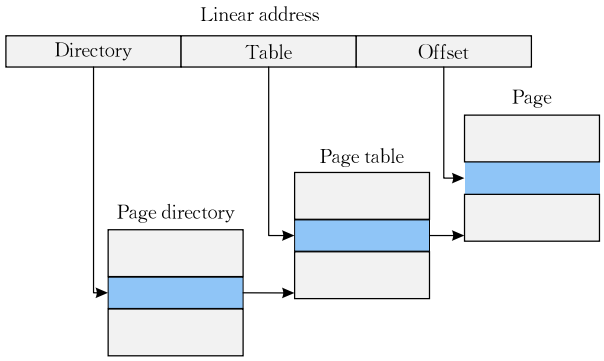


图 3 虚拟机内存地址转换涉及的主要数据结构

用户运行环境运行时的可信性可以通过 5 个方面进行审计,即进程、模块、打开的文件、网络以及内存结构数据. 审计方法跟用户虚拟机的操作系统相关,不失一般性,本文以 Windows 操作系统为例进行说明.

3.3.1 进程审计

进程审计主要检测和查看用户态恶意程序. 操作系统内核会事先分配内存页到特定的内存池中. 当创建一个进程时,操作系统从内存池中将存储空间以页为单位分配给该进程. 每个进程由一个 _EPROCESS 结构维护其信息,如图 4 所示. Windows 提供的进程列举功能和常规进程扫描工具往往通过遍历所有活动进程连接起来的 ActiveProcessLinks 双向链表而得到系统中正在运行的活跃进程. 但是经过特别设计的恶意程序会删除某些进程在该链表中的表项,从而避开扫描,起到隐藏进程的作用,如 FU Rootkit①. 本文的审计方法从进程的内存页分配机制出发,“盯住”创建进程时分配内存页的数据结构,

① Virus and threats descriptions. Rootkit. Win32. Fu. Accessed from [EB/OL]. http://www.f-secure.com/v-descs/fu.shtml

即由 POOL_HEADER 结构(池头结构)保存与进程分配有关的信息,通过分析 POOL_HEADER 结构不仅可以获得系统中运行的进程,还能发现隐藏进程的信息.

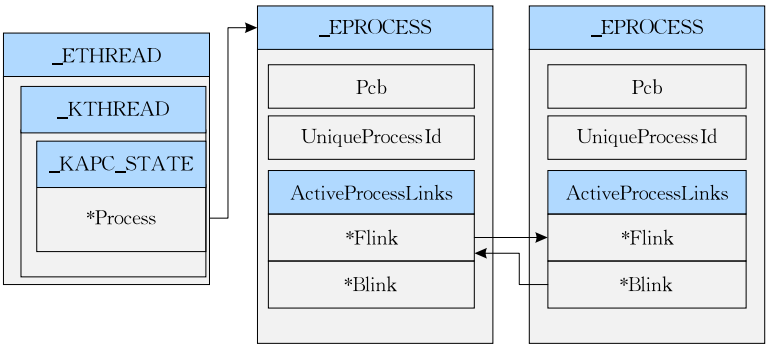


图 4　Windows 操作系统进程结构

3.3.2　模块扫描

模块扫描查看虚拟机运行环境加载的内核态驱动程序和动态链接库. Windows 操作系统的模块结构如图 5 所示,全局结构 TEB(Thread Environment Block)含有指向进程环境块 PEB(Process Environment Block)的指针,其中 PEB 是管理所有模块信息的结构体. 每个模块的信息由 LDR_DATA_TABLE_ENTRY(结构)管理,所有模块通过双向链表 InLoadOrderLink 连接起来. PEB 中的子结构 PEB_LDR_DATA 含有指向所有模块的双向链表,所有模块通过该双向链表连接起来. 大多数模块扫描工具通过遍历该链表,便可知道系统所加载的模块. 恶意程序也往往通过修改该双向链表的结构来实现隐藏模块的目的,如 FU Rootkit.

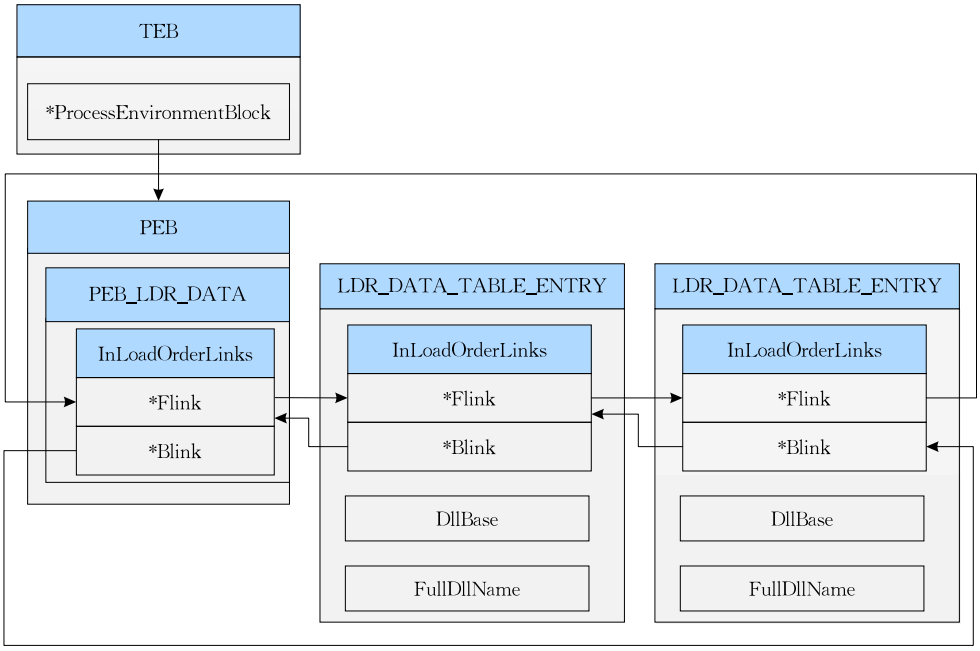


图 5　Windows 操作系统模块结构

本文的可信机制除了进行以上链表扫描外,还通过分析 POOL_HEADER 以发现针对链表指针修改的模块隐藏. 这是因为 Windows 操作系统采用池标记来区分不同的模块,当模块被分配内存时,系统需要指定其池标记. POOL_HEADER(池头结构)含有每个分配单元的信息和池标记,所以通过扫描物理内存的池标记来找出 LDR_DATA_TABLE_ENTRY 结构,进而能够发现隐藏的模块.

3.3.3　文件审计

文件审计是为了查看用户环境在运行时打开了哪些文件. 在 Windows 系统中,每一个打开的文件对应一个 FILE_OBJECT 内存结构,该结构由操作系统的分页池统一进行分配. 文件的创建过程如图 6 所示. 当要打开一个文件时,需要两个动态链接

库:C 语言运行时动态链接库和 Windows 动态链接库.应用程序的打开文件操作调用 C 语言运行时的库函数 fopen,fopen 接着调用 Windows 动态链接库函数 CreateFile.之后,Windows 动态链接库会调用 Ntdll.dll 中的函数 NtCreateFile.Ntdll.dll 含有进入内核模式的系统服务调度器的指令,系统服务调度器通知 I/O 管理器进行文件创建.I/O 管理器负责管理文件系统、缓存管理器、设备驱动程序、网络驱动程序,当其收到文件创建命令后,通知对象管理器创建文件对象的操作,成功创建后,对象管理器将对象句柄返回给 I/O 管理器.最后 I/O 管理器将文件句柄返回给 Windows 动态链接库.因此,通过文件扫描可以发现进程所打开的文件.

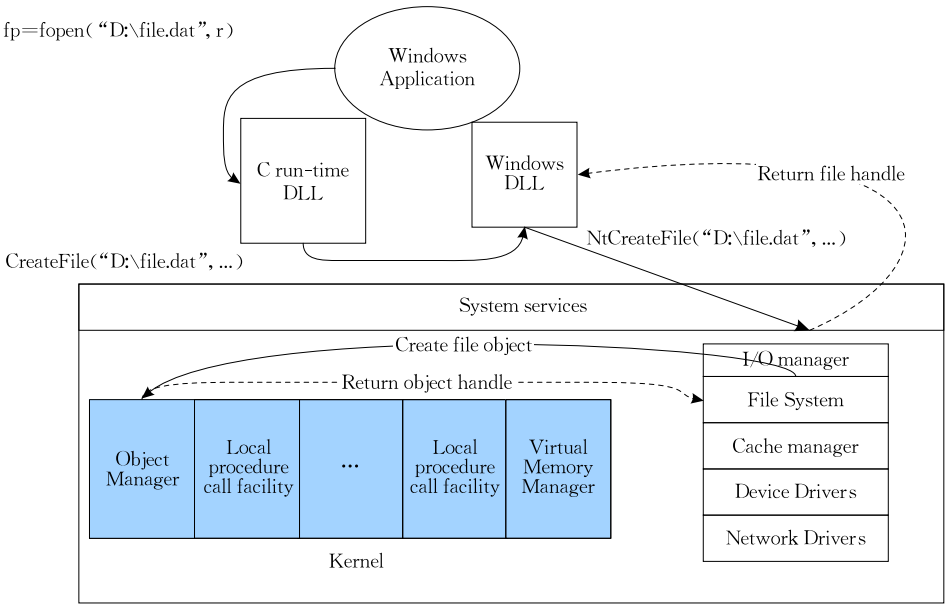


图 6 Windows 操作系统文件创建流程和涉及的内存数据结构

在审计过程中,由于文件对象的创建会与 POOL_HEADER 结构相关联,POOL_HEADER 中会存放文件对象的池标志等信息,所以通过确定这种池标志找到相应的 FILE_OBJECT 结构,就可以扫描出系统中进程打开的所有文件.

3.3.4 网络连接扫描

网络连接扫描是为了查看运行环境中网络连接和活动状态.Windows 系统的 TCP/IP 体系结构如图 7 所示.以数据包发送为例,应用程序将要发送的数据通过 Socket 接口形成 IP 数据包,交由 TCP/IP 驱动程序 Tcpip.sys 处理,处理过程主要有 3 个部分:首先,Tcpip.sys 将数据包发送给防火墙驱动程序 Ipnat.sys,防火墙检查该数据包是否是 Internet 控制消息协议(ICMP),如果 ICMP 设定为阻止,那么防火墙就丢弃该数据包.然后,防火墙检查数据包是否是点到点隧道协议(PPTP),如果是,就分析数据包以确定通用路由封装(GRE)的调用 ID,使得 GRE 数据包能够传入.完成之后防火墙将数据包发回给 Tcpip.sys,由 IP 转发组件确定下一跳 IP 地址和端口.其次,Tcpip.sys 将数据包发送给 IP 过滤驱

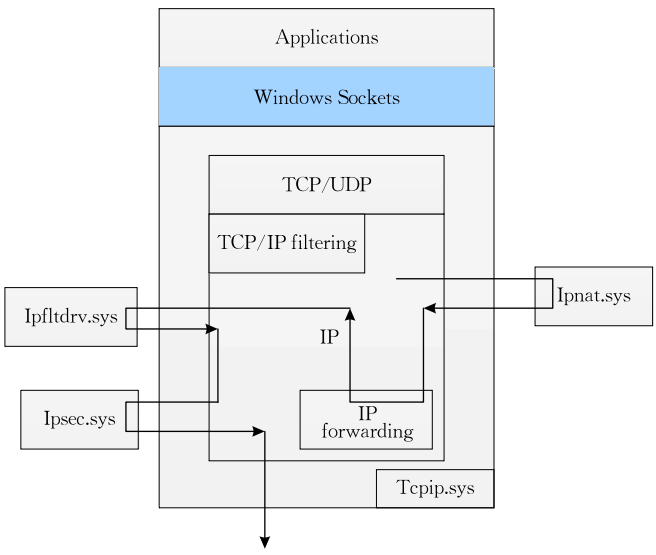


图 7 Windows 操作系统中 TCP/IP 体系结构

动程序 Ipfltdrv.sys,通过比较下一跳 IP 地址和系统配置的出口 IP 包过滤器,如果地址是允许出口的,就将数据包返回给 Tcpip.sys,如果地址是不允许的,就丢弃.最后,Tcpip.sys 将数据包发送给 Ipsec驱动程序 Ipsec.sys,Ipsec 确定是否让数据包

通过、阻止或保护. 如果让数据包通过，则将数据包不做处理返回给 Tcpip.sys；如果阻止数据包，则将数据包丢弃；如果保护数据包，则对数据包进行 Ipsec 加密后返回给 Tcpip.sys.

以上过程中的相关信息将会保存在 Tcpip.sys 驱动程序某一特定符号所指向的单向链表结构中. 因此，通过扫描该单向链表可审计其网络连接信息，包括进程号、发送端 IP 地址、端口 IP 地址、接收端 IP 地址和端口等.

3.3.5　内存数据结构审计

内存中特定的数据可以采用字符串搜索的方式进行审计. 字符串的搜索位置分为内核空间和进程空间，通过搜索内核空间和每个进程的空间，将符合字符串数值的二进制数据转换为字符串，同时得到一些额外的审计信息，包括该字符串在内存中的物理地址、该字串处于内核空间还是进程空间、该字符串的虚拟地址等.

4　实现和评价

4.1　实验环境

基于上述机制的原型系统采用软件 libtpm[19] 模拟硬件 TPM，充当 vTPM 的后端，虚拟机管理器使用开源 Xen 平台，在 Xen 平台中实现 vTPM 模块仿真硬件 TPM，使每个虚拟机拥有一个 vTPM，作为虚拟机的可信根，其体系结构如图 1 所示. TDD (TPM Device Driver)模块与 TPM 硬件交互，作为硬件 TPM 的设备驱动程序，该驱动在目前主流操作系统版本，如 Windows，Linux，Mac OS 中默认安装. 使用 VMI 开源工具 VMITools[20] 旁路和获取用户虚拟机内存数据.

原型系统的部署配置信息如表 1 所示.

表 1　原型系统配置

配置项	物理服务器	虚拟机
CPU	Intel Xeon X5650，2.67GHz，共 24 核	Intel Xeon X5650，2.67GHz
内存	32GB	4GB
二级缓存 (L2 Cache)	12288KB	4MB
硬盘容量	1TB	20GB

4.2　实验结果及分析

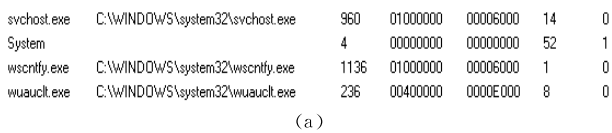
实验和评价以构建可信的云计算运行环境机制的有效性和可信机制带来的额外代价作为目标进行分析.

4.2.1　有效性实验及分析

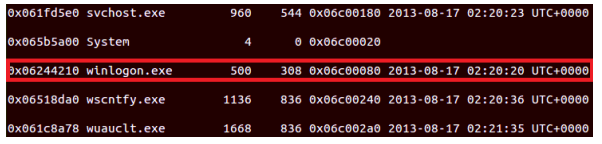
与传统可信计算的假设场景不同，云用户的行为是无法事先严格规定的，同时，用户虚拟机的配置、安装软件、自定制程序等都可能是动态变化的，所以无法采用可信任链的方式对用户虚拟机的所有程序都事先固定其 Hash 摘要值. 因此，本文的机制将可信任链的方式与内存审计的方式结合起来，通过vTPM可信根来保证用户运行环境的虚拟硬件、BIOS 等的可信性，通过内存审计来检测和仲裁运行时的不可信行为.

以 Windows 环境中的典型内核态木马 Win32.Fu Rootkit 为例. FU Rootkit 是一种可以隐藏进程的恶意程序，能够将自身隐藏起来而不被宿主机发现.

FU Rootkit 会在系统中安装内核态驱动程序，通过驱动程序来删除活动链表 PsActiveProcessList 中的进程信息，而主流安全工具，如 Poison Ivy①，主要通过扫描 PsActiveProcessList 链表来获取当前环境的进程信息，因而无法检测到隐藏的进程. 如图 8(a)所示，采用主流安全工具查看系统进程，并不能发现 FU Rootkit 创建的进程.



(a)



(b)

图 8　使用主流安全工具和本文可信机制对恶意隐藏进程进行检测的对比

而通过本文提出的可信机制对内存信息进行审计，可通过判断进程的内存页使用情况从而检测隐藏进程. 进程内存页的分配会存储在 POOL_HEADER 中，POOL_HEADER 是一个操作系统分配内存页的池子，进程在创建时，POOL_HEADER 就存有该进程的相关信息，即使 Rootkit 将进程隐藏，该进程在 POOL_HEADER 中的内存页分配的标志也不会立即消失. 所以对 POOL_HEADER 链表进行扫描，可以发现隐藏的进程以及目前已经结束的进程，如图 8(b)所示.

与此类似，FU Rootkit 通过修改 LDR_DATA_TABLE_ENTRY 指针，删除模块在双向链表 InLoadOrderLink 中的对应项，可隐藏系统模块，从

① Poison Ivy [EB/OL]. http://www.poisonivy-rat.com

而避开采用主流安全工具的扫描，如 Icesword[①] 无法检测被隐藏的模块，其结果如图 9(a)所示. 而本文提出的可信机制是通过扫描和审计 POOL_HEADER 结构，所以即使模块从 InLoadOrderLink 链表中删除，其 POOL_HEADER 结构也含有该模块的信息，如图 9(b)所示.

```
win32k.sys      0xBF800000   0x001C3000   0x29104000        75  \SystemRoot\System32\win32k.sys
dxg.sys         0xBF9C3000   0x00012000   0x29104000        78  \SystemRoot\System32\drivers\dxg.sys
IsDrv122.sys    0xF62FC000   0x00034000   0x09104000        85  \SystemRoot\System32\Drivers\IsDrv122.sys
HTTP.sys        0xF6420000   0x00041000   0x09104000        84  \SystemRoot\System32\Drivers\HTTP.sys
```

(a)

```
0x062df748 win32k.sys          0xbf800000   0x1c3000 \SystemRoot\System32\win32k.sys
0x06495538 dxg.sys             0xbf9c3000    0x12000 \SystemRoot\System32\drivers\dxg.sys
0x064b2b28 cirrus.dll          0xbff60000    0x17000 \SystemRoot\System32\cirrus.dll
0x0644e2e8 IsDrv122.sys        0xf62fc000    0x34000 \SystemRoot\System32\Drivers\IsDrv122.sys
0x06244af0 HTTP.sys            0xf6420000    0x41000 \SystemRoot\System32\Drivers\HTTP.sys
```

(b)

图 9 使用主流安全工具和本文可信机制对隐藏模块进行检测的对比

4.2.2 性能实验及分析

性能实验包括证据收集和审计的性能评价以及可信机制对虚拟机正常用户环境的性能影响.

证据收集主要有同步和异步 2 种方式. 同步方式是把用户虚拟机的内存转储(dump)为硬盘文件，再对文件进行可信性审计；异步方式则不需要每次审计都对虚拟机的完整内存映像进行转储，而可以有针对性地、更细粒度地访问和获取特定的内存内容，与此同时并进行审计分析.

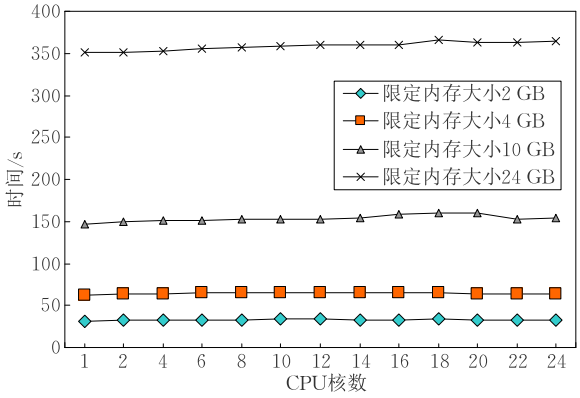


图 10 用户虚拟机 CPU 核数对可信证据收集完成时间的影响

一般来说，影响证据收集性能的因素主要包括用户虚拟机内存的大小，以及用户虚拟机的计算性能. 首先，分析用户虚拟机的计算配置对证据收集性能的影响. 图 10 展示了在给定虚拟机内存大小，并保持虚拟机其他配置不变的情况下，CPU 核数(Cores)对同步证据收集完成时间的影响. 从图 10 可以发现，在典型的内存配置下(2 GB，4 GB，10 GB，24 GB)，内存镜像的转储时间几乎是一条直线，即表明：虚拟机的计算能力对可信证据的收集几乎没有影响. 不失一般性，在接下来的实验中，用户虚拟机配置为 6 核 CPU. 另外可以看出，内存镜像的转储时间是随着内存大小增加而线性增加的，这是因为内存镜像转储的主要时间开销在 I/O 操作上.

图 11 给出了在同步方式下，进行一次可信证据收集和审计操作(以列举当前系统中运行的所有进程为例)所完成的时间. 由此可以发现，完成时间跟虚拟机内存大小基本成正比关系，这是因为绝大部分的时间花费在将内存数据转储到文件中. 以 10 GB 内存配置为例，证据收集花费的时间是 155.05 s，而进程审计的时间花费是 1.87 s，只占前者的 1.2%. 而对于审计操作来说，由于其主要处理字符串匹配和数据结构搜索的情况，跟内存转储文件的大小没有关系，所以审计操作的完成时间并不随内存大小的增加而线性增加，如图 12 所示.

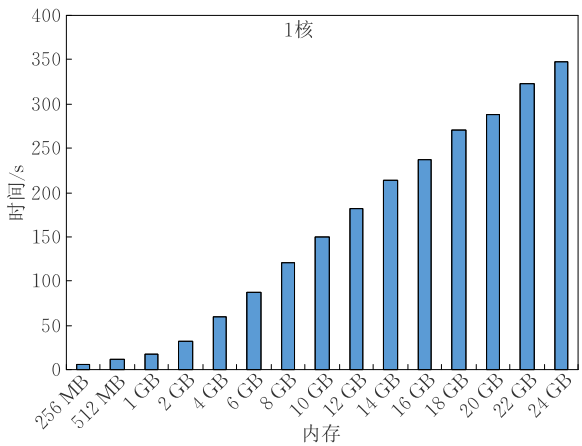
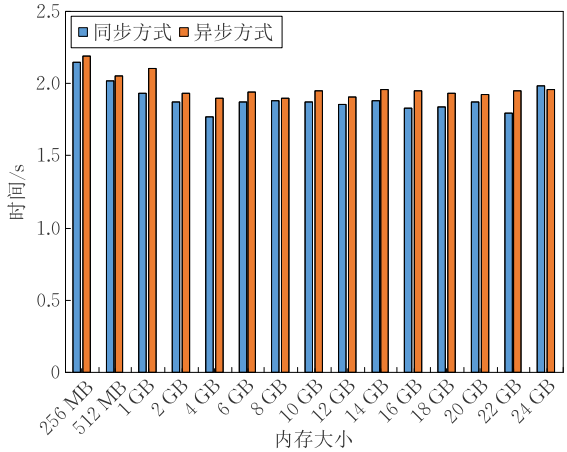


图 11 在同步方式下，一次可信证据收集和审计操作在不同虚拟机内存配置下的完成时间

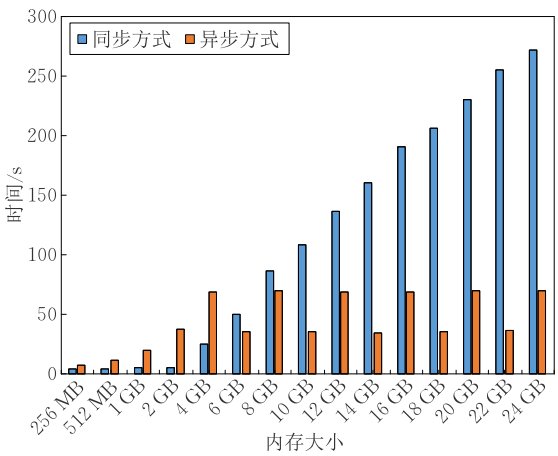
图 12 展示了在异步方式下，可信证据收集和审计的性能，以及在典型的审计操作下，异步方式的完成时间与直接对转储文件进行审计的时间对比. 从图 12 可以看出，因为异步方式并不需要先把整个内存映像转储到文件中，而可以直接对细粒度获取的特定内存内容进行审计，所以异步方式下证据收集加上审计操作的总共完成时间跟同步方式下的审计时间是相当的. 而从具体数值来看，异步方式大大节省了一次证据收集和审计的时间. 以进程审计为例，在 10 GB 内存配置下，同步方式的完成时间是 156.92 s，而异步方式的完成时间是 1.95 s.

测试引入可信机制对用户虚拟机运行环境带来

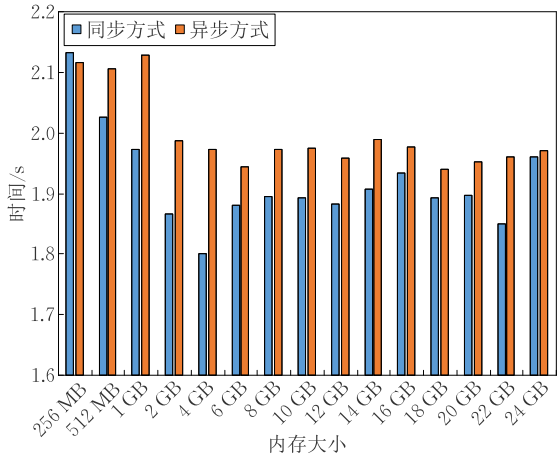
① Icesword [EB/OL]. http://icesword.softonic.cn



(a) 进程审计



(b) 文件审计



(c) 网络连接扫描

图 12　在异步方式下，可信证据收集和审计的完成时间以及与同步方式的审计时间的对比

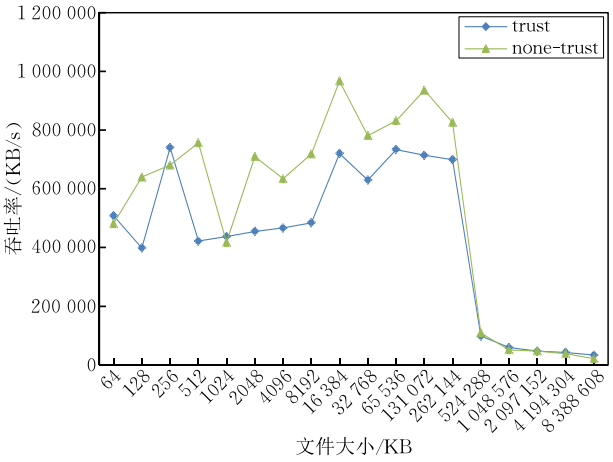
的性能损耗，可以参照未引入可信机制之外的硬件设施及软件设施相同的普通虚拟机的性能. 因为可信证据的审计是对用户虚拟机内存数据的旁路操作，而且跟用户对虚拟机的操作可以异步进行，所以本文的性能评价主要考虑可信任链所引入的代价.

可信用户运行环境会计算涉及的可执行代码的哈希值并扩展到 vTPM 的 PCR 寄存器中，这样势必会给程序带来额外的执行代价. 对相同机器配置的虚拟机，选用不同类型的测试工具对虚拟机性能进行对比测试. 测试工具的名称和说明如表 2 所示.

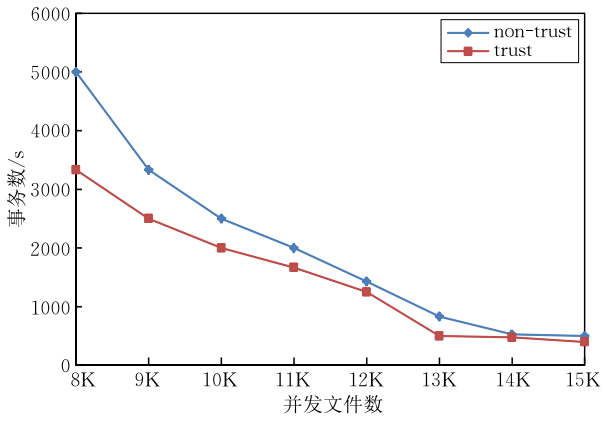
表 2　性能测试使用工具介绍

名称	功能描述
IOZone	IOZone 是一个 Benchmark 工具，可对不同操作系统的多种文件系统的读写性能进行测试，计算得出文件系统的读写性能.
BYTEmark	BYTEmark 基准测试套件利用不同的计算密集型算法，测试系统的处理器、高速缓存、协处理器、Memory 性能.
PostMark	PostMark 是用来模拟邮件服务器或电子商务系统的行为，并测试其性能的 Benchmark 工具. PostMark 包括三个阶段. 在第一阶段中文件池被创建. 在下一阶段执行 4 种类型的操作：文件创建，删除，读取和追加. 在最后一个阶段，池中的所有文件都将被删除.
TPCC-UVa[21]	TPC-C 是用于测量高端系统性能的 Benchmark 工具，而 TPCC-UVa[21] 是一个遵循 TPC-C 的开源项目. 通过在一段时间内向系统并发发送交易请求，模拟一系列分布式在线交易(OLTP 系统)的执行，可测量不同的计算机的速度或分析个别组件(包括硬件和软件)的行为，以及它对系统整体性能的影响.

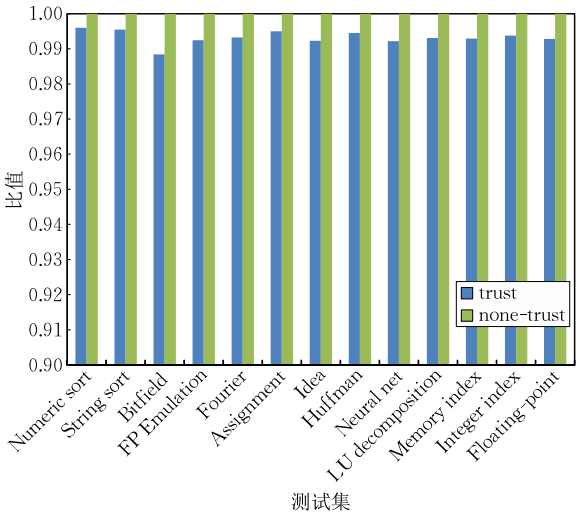
使用表 2 的性能测试工具对引入可信机制的虚拟机运行环境与未引入可信机制的虚拟机运行环境作性能测试和对比，结果如图 13 所示. 从图 13(a)可以看出，对于 I/O 操作，iozone 测试结果表明引入可信机制的虚拟机运行环境和未引入可信机制的虚拟机运行环境 I/O 吞吐率基本一致，并且随着文件大小的增加，引入可信机制的虚拟机运行环境引入的性能损耗基本可以忽略. 这是因为引入可信机制的虚拟机运行环境的额外性能代价主要在于文件打开时对可信证据度量哈希和对 vTPM 的扩展操作，



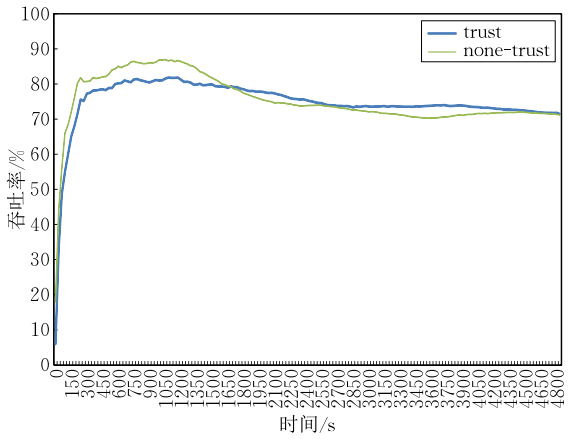
(a) 使用IOzone①,在特定数据块大小时(64 KB),I/O吞吐率随文件大小变化情况



(b) 使用PostMark②文件事务处理能力随文件池数量变化而变化的情况



(c) 使用ByteMark③依据不同指标对比计算能力



(d) 使用TPCC-UVa,请求处理吞吐率随时间变化情况

图 13 使用性能测试工对不同类型用户虚拟机运行环境进行性能对比

而对文件的读写性能影响很小,但 iozone 测试并不考虑文件的打开时间.

图 13(b)所示为对引入可信机制的虚拟机运行环境和未引入可信机制的虚拟机运行环境关于每秒完成文件事务型数量的对比,可以看出引入可信机制的虚拟机运行环境比未引入可信机制的虚拟机运行环境每秒完成的事务数量有所减少.这是因为对于文件事务型处理测试,如 PostMark,由于处理每个事务时,可信机制需要对事务操作进程运行代码进行哈希,获取摘要值,并扩展到 vTPM 的 PCR 寄存器中,故每个事务完成时间会有所增加.但是可信机制对于相同的事务操作不必再次获取摘要,也不用将摘要保存到寄存器,所以,越是大规模的文件处理,文件并发数越大,可信机制越可以节省操作时间,可信机制带来的额外代价相对越小,因而引入可信机制与未引入可信机制之间的差距也相对越小.

对于计算密集型操作,如图 13(c)所示,一般而言引入可信机制的虚拟机运行环境低于半可信虚拟机运行环境,半可信虚拟机运行环境低于未引入可信机制的虚拟机运行环境.从图 13(c)可以看出可信机制、半可信机制及未引入可信机制之间的虚拟机运行环境计算性能差别不大,性能损耗不足 2%.这是因为引入可信机制的虚拟机运行环境只在计算程序运行前执行可信机制、执行可信证据收集和对 vTPM 的扩展操作,而当程序运行后引入可信机制的虚拟机运行环境与未引入可信机制的虚拟机运行环境相同,性能损耗相对更小.

本文还利用 TPCC-UVa 测试工具评价可信机

① IOzone [EB/OL]. http://www.iozone.org
② Postmark [EB/OL]. https://communities.netapp.com
③ Bytemark [EB/OL]. http://www.tux.org/~mayer/linux/bmark.htm

制引入的数据分析性能损耗. TPCC-UVa[21]是一个遵循 TPC-C 的开源项目. 通过在一段时间内向系统并发发送交易请求，来模拟一系列分布式在线交易(OLTP 系统)的执行. 在此实验中 TPCC-UVa 测试时间配置为 1 h，仓库为 9 个，启动时间为 120 s. 如图 13(d)所示，前 120 s 启动时间里，引入可信机制的虚拟机与未引入可信机制的虚拟机吞吐率在到达峰值以前，引入可信机制的虚拟机运行环境的吞吐率相对稍低，而在各自吞吐率达到峰值之后，可信机制带来的额外代价相对较小.

5　总　结

如何为用户提供可信的云计算服务是云计算模式面临的重要问题. 但采用可信计算技术的基本假设是所有在用户虚拟机中安装的内核模块和应用程序都需要事先指定并已知其执行代码的完整性信息. 这个假设在云用户实际使用时往往不成立. 针对该问题，本文提出了一种将虚拟可信根 vTPM 和可信审计技术结合起来的用户可信运行环境构建与审计机制. 对于体系结构栈中可事先固定的、在启动和运行时不变的组件，如 BIOS、启动程序、操作系统内核，通过 vTPM 可信根保证可信的启动；对于运行过程中变化的、无法事先确定的组件，如内核模块、静态或动态链接库、用户态程序，本文提出了一种可信审计的方法，对用户虚拟机的运行状态环境进行证据收集和证据审计，及时检测在实际使用中无法事先固定的用户运行环境的可信性. 若用户运行环境处于不可信状态，则及时通知用户，并采取相应措施. 本文实现了基于该可信机制的原型系统，并用实验的方法进行了有效性评价和性能代价评价，结果表明，该机制针对典型的不可信攻击是有效的，且引入的性能代价对用户实际使用的影响可忽略.

参　考　文　献

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

[2] Ko R K L, Jagadpramana P, Mowbray M, et al. TrustCloud: A framework for accountability and trust in cloud computing//Proceedings of the 2nd IEEE World Congress on Services. Washington, USA, 2011: 584-588

[3] Michael A, Armando F, Rean G, et al. Above the clouds: A Berkeley view of cloud computing. University of California at Berkley, USA: Technical Report UCB/EECS-2009-28, 2009

[4] Berger S, Cáceres R, Goldman K A, et al. vTPM: Virtualizing the trusted platform module//Proceedings of the 15th USENIX Security Symposium. Vancouver, Canada, 2006: 305-320

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

[6] Krautheim F. John Private virtual infrastructure for cloud computing//Proceedings of the 2009 Workshop on Hot Topics in Cloud Computing, USENIX Association. San Diego, USA, 2009: 1-5

[7] Khan I, Rehman H, Anwar Z. Design and deployment of a trusted Eucalyptus cloud//Proceedings of the IEEE International Conference on Cloud Computing. Washington, USA, 2011: 380-387

[8] Cheng G, Jin H, Zou D, et al. Building dynamic and transparent integrity measurement and protection for virtualized platform in cloud computing. Concurrency and Computation: Practice and Experience, 2010, 22(13): 1893-1910

[9] Frederic S, Claudia E. Enhancing trusted plat for modules with hardware-based virtualization technique//Proceedings of the 2nd International Conference on Emerging Security Information, Systems and Technologies. Cap Esterel, France, 2008: 1-9

[10] Paul E, Jork L. Para-virtualized TPM sharing//Proceedings of the 1st International Conference on Trusted Computing and Trust in Information Technologies. Villach, Austria, 2008: 119-132

[11] Kursawe K, Flexible D S. uTPMs through disembedding//Proceedings of the ACM Symposium on Information, Computer and Communications Security. Sydney, Australia, 2009: 116-124

[12] Liu Zi-Wen, Feng Deng-Guo. TPM-based dynamic integrity measurement architecture. Journal of Electronics & Information Technology, 2010, 32(4): 875-879

[13] Sailer R, Zhang Xiao-Lan, Jaeger T, van Doorn L. Design and implementation of a TCG-based integrity measurement architecture//Proceedings of the Security Symposium. San Diego, USA, 2004, 13: 16-16

[14] Bertholon B, Varrette S, Bouvry P. Certicloud: A novel TPM-based approach toensure cloud IaaS security//Proceedings of the 4th International Conference on Cloud Computing. Washington, USA, 2011: 1-8

[15] Zhang Feng-Zhe, Chen Jin, Chen Hai-Bo, Zang Bin-Yu. Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization//Proceedings of the 23rd ACM Symposium on Operating Systems Principles. Cascais, Portugal, 2011: 203-216

[16] Garfinkel T, Mendel R. A virtual machine introspection based architecture for intrusion detection//Proceedings of the Network and Distributed Systems Security Symposium. San Diego, USA, 2003, 3: 191-206

[17] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization//Proceedings of the ACM SIGOPS Operating Systems Review. Bolton Landing, USA, 2003: 164-177

[18] Trusted Computing Group. TCG Software Stack (TSS) Specification—Version 1.10 Golden, 2003

[19] Mario S, Stamer H. A software-based trusted platform module emulator//Lipp P, Sadeghi A R, Koch K M eds. Trusted Computing—Challenges and Applications. Berlin: Springer, 2008: 33-47

[20] Brendan D-G, Payne B, Lee W. Leveraging forensic tools for virtual machine introspection. Georgia Institute of Technology, 2011

[21] Lanos D R. TPCC-UVa: An open-source TPC-C implementation for global performance measurement of computer systems. ACM SIGMOD Record, 2006, 35(4): 6-15

LIU Chuan-Yi, born in 1982, Ph.D., associate professor. His research interests include computer systems, including cloud computing and cloud security, data security and data protection.

WANG Guo-Feng, born in 1988, Ph.D. candidate. His current research interest is cloud security.

LIN Jie, born in 1987, Ph.D. candidate. His research interests include computer network, information security.

FANG Bin-Xing, born in 1960, professor, Member of Chinese Academy of Engineering. His research interests include information and network security, content security.

Background

This paper belongs to Cloud Computing and Cloud Security area. Trustworthiness is a critical factor for the large-scale use of cloud services, and this paper focuses on the trustworthiness of cloud execution environment. Generally speaking, related works can be divided into two categories. The first approach constructs a trusted Virtual Machine Monitor (VMM) to hold the user virtual machines. This approach has the advantage of not modifying the low-level architecture stack. But it cannot be applied to commodity VMMs. The later approach uses Trusted Computing technologies to extend the current trusted chain into virtual machines, e. g. using virtual Trusted Platform Module (vTPM) as the virtual machine trusted root. However, it only works with the assumption that all the kernel modules and user-space processes installed in the tenant's virtual machine should be pre-fixed and the integrity digests should be pre-confirmed.

This paper proposes a practical mechanism for trusted cloud environment construction and audit, which combines virtual vTPM and trust evidence audit. vTPM guarantees the trusted boot up, while the audit method check the trust status of the run-time virtual machine based on auditing the trusted evidence collected from the memory of the user virtual machine. A prototype is also implemented according to the above mechanism. Experimental results show that the mechanism is effective and the performance overhead incurred is minor.

This paper is supported by the National Natural Science Foundation of China under Grant No. 61202081: "Study on the Trustworthiness of Cloud Providers", which researches the construction and audit for the tenant trusted execution environment.