ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.2, February 2010, pp.373−387
doi: 10.3724/SP.J.1001.2010.03789

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

基于TPM的运行时软件可信证据收集机制*

古 亮[1,2], 郭 耀[1,2+], 王 华[1,2], 邹艳珍[1,2], 谢 冰[1,2], 邵维忠[1,2]

[1](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[2](北京大学 信息科学与技术学院 软件研究所,北京 100871)

Runtime Software Trustworthiness Evidence Collection Mechanism Based on TPM

GU Liang[1,2], GUO Yao[1,2+], WANG Hua[1,2], ZOU Yan-Zhen[1,2], XIE Bing[1,2], SHAO Wei-Zhong[1,2]

[1](Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, China)

[2](Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: yaoguo@sei.pku.edu.cn

Gu L, Guo Y, Wang H, Zou YZ, Xie B, Shao WZ. Runtime software trustworthiness evidence collection mechanism based on TPM. *Journal of Software*, 2010,21(2):373−387. http://www.jos.org.cn/1000-9825/3789.htm

Abstract: This paper extends the software trustworthiness evidence framework to include the runtime software trustworthiness evidence. To collect software trustworthiness evidence in an objective, genuine and comprehensive way, it proposes a runtime software trustworthiness evidence collection mechanism based on trusted computing technology. Based on the features provided by TPM (trusted platform module), as well as the late launch technology, a trusted evidence collection agent is introduced in an operating system kernel. The agent can securely monitor executing programs and collect their trustworthiness evidence accordingly. The agent also provides some trusted services for programs to collect application specific evidences and guarantees the trustworthiness of these evidences. This mechanism has good scalability to support various applications and software trustworthiness evaluation models. This paper also implements a prototype for the agent based on Linux security model in Linux. Based on the prototype, it studies the trustworthiness evaluation for executing a client program in a distributed computing environment. In this application, the performance of prototype is studied, and the feasibility of this approach is demonstrated.

Key words: high confidence software; software trustworthiness evaluation; software trustworthiness evidence; evidence collection; trusted computing; TPM (trusted platform module)

摘 要: 扩展了已有的软件可信性证据模型,引入了运行时软件可信证据,从而提供了更为全面的软件可信证据模型.为了提供客观、真实、全面的可信证据,提出了一种基于可信计算技术的软件运行时可信证据收集机制.利用可信平台模块(trusted platform module,简称 TPM)提供的安全功能,结合“最新加载技术(late launch)”,在操作系统层

* Supported by the National Basic Research Program of China under Grant No.2009CB320703 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant Nos.2007AA01Z462, 2007AA010304, 2008AA01Z133 (国家高技术研究发展计划(863)); the Fund for Creative Research Groups of China under Grant No.60821003 (国家创新研究群体科学基金)

Received 2009-06-15; Revised 2009-09-11; Accepted 2009-12-07

引入了一个可信证据收集代理.此代理利用 TPM,可以客观地收集目标应用程序的运行时可作为软件可信证据的信息,并保障可信证据本身的可信性.该可信证据收集机制具有良好的可扩展性,能够支持面向不同应用的信任评估模型.基于 Linux Security Module,在 Linux 中实现了一个可信证据收集代理的原型.基于该原型,分析了一个分布式计算客户端实例的相关可信属性,并且分析了可信证据收集代理在该应用实例中的性能开销.该应用实例验证了该方案的可行性.

关键词: 高可信软件;软件可信性评估;软件可信证据;软件可信证据收集;可信计算;TPM(trusted platform module)

中图法分类号: TP311 **文献标识码:** A

现有的软件技术不能满足不断增长的软件质量需求,经常发生的各种软件失效和威胁给人们造成了巨大的损失[1,2].随着Internet的快速发展和普及,软件系统由基于集中封闭的计算平台向网构软件转化,软件的开发、运行和维护也从传统的封闭发展为开放、动态和多变[3,4].在这种开放的环境下,如何获得和保证软件在开发和运行时的可信性质,已经成为软件理论和技术的重要研究领域.

如果一个软件的行为总是与预期一致,则称该软件可信[5].软件可信性是软件质量的一种特殊的表现形式.它所关注的是使用层面的综合化的质量属性及其保障形式,涉及多个质量属性的集合以及这些属性的综合与平衡.图 1 所示为软件生命周期涉及到的主要活动.高可信软件技术关注的问题可以分为两个方面:其一是如何构造高可信的软件产品,涉及了生产高可信软件制品的相关技术;其二是如何评估和保障软件产品运行时的可信性,涉及了软件运行时可信性评估和保障技术.由高可信软件开发技术获得高可信的软件制品,这是软件运行时可信性的前提.高可信软件系统技术是软件制品中所承载的功能和非功能需求得到正确实现的重要保障.

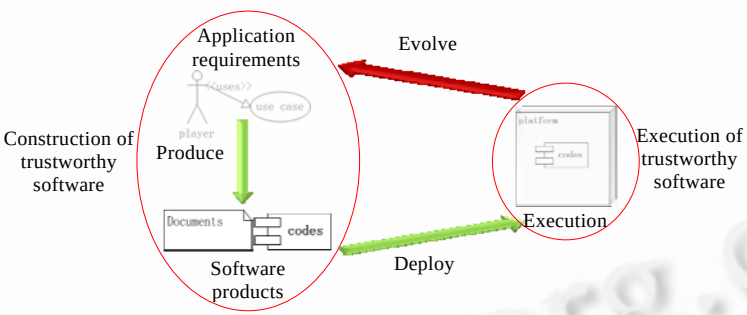


Fig.1 Software lifecycle and high confidence software technology

图 1 软件生命周期与高可信软件技术

如何度量软件的可信性是当前高可信软件研究的重要内容[5,6].软件可信性的度量需要依据所获得的软件可信证据.软件可信证据是指所有与软件相关的,能够反映其某种可信属性的度量值、文档或其他信息[7].软件的可信特征贯穿于软件的整个生命周期,对软件可信性的评估也应综合软件生命周期的各个阶段的软件可信证据.在开放的网络环境下,可信证据本身会面临各种威胁,例如伪造、篡改和破坏.如何全面而准确地获得软件可信证据、如何保障可信证据的有效性,对软件可信性评估具有重要的意义.

已有的软件可信评估存在两个问题:软件可信性评估应是全面的,对软件可信性的评估还应考虑运行时的软件可信证据;如何可信地获得软件运行时的信息,这是进行有效的软件可信性评估的基础.

软件可信证据是软件可信性评估的前提和基础.从软件生命周期出发,TRUSTIE(trustworthy software tools and integration environment)[8]把软件可信证据划分为 3 类:开发阶段证据、提交阶段证据和应用阶段证据.该模型中的应用阶段证据主要是关注于软件的应用效果,而忽略了一个非常重要的软件可信证据来源:运行时的软件可信性证据.在运行阶段软件的运行情况客观地反应了软件在特定环境下的可信特征,如软件运行时的性能、可靠性、安全性等,因而运行时的软件可信证据是评估软件可信性的有利证据.

软件部署运行的平台通常都分布于开放网络环境下各自独立的信任域内,在这种环境下,运行时的软件可

信证据收集和软件可信性评估很可能是由互不信任的实体分别进行的.进行运行时软件可信证据收集的实体并不能保证客观地完成证据收集的工作,因为它对运行时平台具有完全的控制能力,能够以任意的方式修改软件运行平台和运行时的软件可信证据.传统的安全机制并不能解决这些问题.此外,现有软件系统中存在的安全问题为客观、准确地记录运行时的软件可信证据制造了很大的困难.例如,攻击者通过对操作系统的攻击而获得对系统的控制权,可以很容易地破坏或者篡改运行时应用程序的软件可信证据.如何进行跨信任域的运行时软件可信证据收集、如何保障运行时可信证据的安全性,是软件可信性评估需要解决的问题.

针对上文提出的两个问题,本文完成了两部分的工作:其一,扩展了已有的软件可信性评估模型和软件可信证据模型,把运行时软件可信证据作为一个重要的部分添加到了软件可信证据模型中;其二,提出了一种基于可信计算(trusted computing)技术[9]的软件可信证据收集机制,从而能够可信地收集运行时软件可信证据.

在扩展的软件可信性评估模型中,本文强调了可信证据收集的重要性.在扩展的可信证据模型中,本文引入了运行时的软件可信证据.本文利用可信平台模块TPM(trusted platform module)提供的安全功能,结合最新加载(late launch)[10−12]和可信虚拟机监视器(trusted virtual machine monitor,简称TVMM)[13]来保护在操作系统层引入的可信证据收集代理(trustworthiness evidence collection agent,简称TECA).可信证据收集代理可以客观地收集目标应用程序的运行时软件可信证据.利用TPM提供的完整性报告功能,可信证据收集代理根据应用程序的特征来记录程序执行过程中的相关信息,这些信息包括目标程序在特定时刻的完整性状态、程序执行时间相关的数据、程序所依赖的配置文件状态、程序操作序列、程序所依赖的系统部分、程序日志等.可信证据代理利用TPM提供的安全存储功能来保护记录的可信证据,并利用TPM对证据进行签名,从而可以保证证据的完整性和来源的可验证.

为了支持应用特定的软件可信证据收集,可信证据代理基于 TPM 的安全功能提供了完整性度量、签名、可信时间服务和安全保护的可信服务.应用程序可以利用这些可信服务来保证应用特定的软件可信证据,例如用户的反馈等.可信证据收集代理位于操作系统层,所以能够根据指定的列表来监控关注的程序和其他相关对象,因而本文介绍的可信证据收集机制具有可扩展性,能够根据不同应用的软件可信性评估模型进行可信证据收集.基于 Linux Security Module,我们在 Ubuntu 7.10 中实现了一个可信证据收集代理的原型.基于该原型,我们分析了分布式计算中的一个客户端实例的相关可信属性,并分析了可信证据收集代理在该应用实例中的性能,证明了我们的方案的可行性.本文提供的可信证据收集机制可以为软件可信性评估提供有力的支撑.

1 可信计算基础

根据可信计算组织的标准[9],一个可信计算平台应该提供数据完整性、数据的安全存储和计算平台身份的证明等功能.而这些功能是由如下可信计算平台的基本特征来保证的:安全输入输出(secure I/O)、存储屏蔽(memory curtaining)、密封存储(sealed storage)、平台远程证明(platform remote attestation).TPM是可信计算平台的核心模块和信任根.TPM本身是一种SoC芯片,提供了所有与安全相关的基础功能,包括随机数产生、密钥相关操作、签名、安全存储、完整性度量和报告等功能.TPM中包含了一组用于度量平台配置完整性的平台配置寄存器(platform configuration register,简称PCR).这组PCR寄存器配合SHA-1 的Hash算法完成对数据块的状态记录,即PCR的扩展(extend):

$$PCR\_Extend(m)\text{: } PCR_i^{new} \leftarrow SHA\text{-}1(PCR_i^{old} \| m),$$

其中,m是当前操作要度量的数据块.基于完整性度量和签名机制,TPM可以向特定的请求方提供平台的完整性报告(integrity report)机制.TPM本身的可信性是由其特别的设计、实现以及硬件保护机制来保证的.我国也制定了具有自主知识产权的可信密钥模块(trust cryptographic module,简称TCM)[14]相关标准.

最新加载技术(late launch)结合硬件虚拟技术,允许在不重启平台的情况下为程序的执行提供全新加载的运行环境,从而保证程序执行的完整性.最新加载技术可以为远程证明提供动态信任根(dynamic root of trust).Intel推出了它的可信执行技术(trusted execution technology,简称TXT)[10,11],与之对应地,AMD也提出了安全虚拟机(secure virtual machine,简称SVM)[12]的CPU架构扩展.此外,基于软件的虚拟技术来提供隔离的执行环境的

研究和应用也受到了广泛的关注[15].

最新加载技术可以在系统运行的任意时刻安全地加载一个虚拟机监视器(virtual machine monitor,简称VMM)或者是一个安全内核(security kernel).为了引导一个VMM,处于CPU保护模式的代码调用一个特权指令(Intel TXT技术提供的指令是GETSEC[SENTER],AMD的SVM提供的指令是SKINIT).该特权指令提供对装载VMM的硬件保护机制.例如,SKINIT指令会禁止DMA对包含了要加载的VMM代码的内存块的操作,并在执行转移到VMM之前禁止中断和调试访问.在TXT中,GETSEC[SENTER]也在提供一个已度量加载环境(measured launched environment,简称MLE)的同时提供对特定内存区域的DMA的访问限制,其中,MLE的完整性是由另外一个认证代码(authenticated module)模块来验证的.

2 扩展的软件可信性评估模型和软件可信证据模型

2.1 软件可信性

如果一个软件的行为总是与预期一致,则称该软件可信[5].软件可信属性是用以描述和评价软件系统可信的一组属性.软件可信属性可被细化成多级子属性.文献[8]中定义的软件可信属性包括可用性(availability)、可靠性(reliability)、安全性(security)、实时性(real time)、可维护性(maintainability)和可生存性(survivability).上述每个特性包含了若干子特性,这些属性构成了软件可信属性模型.软件的很多可信属性都是可以通过分析软件运行时可信证据获得的.例如,软件安全性可以通过分析程序的执行序列和相关输入输出来获得,软件性能可以通过分析软件运行时间来获得,可靠性和可用性也可以通过分析程序执行情况来获得.

2.2 软件可信性评估模型

王怀民等人[7]提出的软件可信性评估模型包含3个重要部分:软件可信等级定义、软件可信证据模型和软件可信等级评定.本文明确了软件可信证据收集的重要性,对软件可信性评估模型扩展为如图2所示的模型,把软件可信证据收集作为一个重要的元素添加到软件可信性评估模型中.对于特定类型的软件,基于用户对软件所期望的可信属性的满意程度给出软件可信等级的定义.依据特定的可信分级定义,可以确定软件可信证据模型,而该证据模型决定了实现软件可信证据的收集机制的实现.由软件可信证据收集机制获得的证据将会在进行软件可信性评估时被评估者用于进行软件可信性评定.

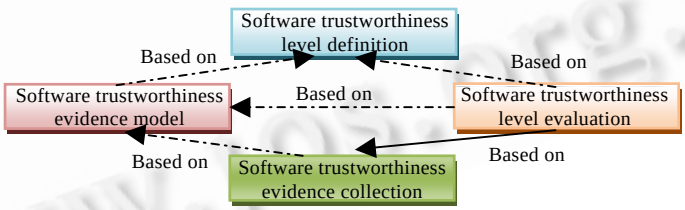


Fig.2 Software trustworthiness evaluation model

图2 软件可信性评估模型

2.3 软件可信证据模型

软件可信证据是软件可信性评估的前提,因而本文首先介绍一个扩展的软件可信证据模型,如图3所示.基于文献[8]的软件可信证据,本文对软件可信证据模型进行了扩展.在原有的开发阶段证据、提交阶段证据、应用效果证据基础上添加了运行时证据.

软件开发阶段的证据用于反映软件设计与生产过程中影响软件可信属性的各种因素,包括开发人员、开发环境、软件制品以及软件过程保障技术等.开发阶段证据的获得方式包括:记录软件开发过程中所有的开发人员的活动、各个阶段的软件制品(包括代码和文档)、规范、评审和测试数据等.提交阶段的证据关注于反映通过如下手段获得的软件的可信性信息:自动化的分析、测试和验证工具以及人工分析和评估.获取提交阶段证

据的方式包括:记录对软件制品进行分析所采用的工具和方法以及进行该分析的评测人员.应用阶段的证据能够反映软件实际应用过程中满足应用需求的程度,关注于软件应用的广泛程度、用户满意度和第三方评价等.软件运行时的可信证据主要包括与软件运行状态相关的信息,例如,程序代码状态的变迁、程序操作/行为、程序的运行时间、程序的配置、程序的输入输出等.特定的应用程序的可信性评估可能需要应用特定的可信证据,如即时通讯工具和分布式科学计算所需要的可信证据是与其应用相关的.

可信性证据必须具备以下 3 个特征[8]:(1) 客观性;(2) 关联性,或称相关性;(3) 可获得性.如何保证软件可信证据的以上 3 个特征,这是可信证据收集机制需要解决的问题.

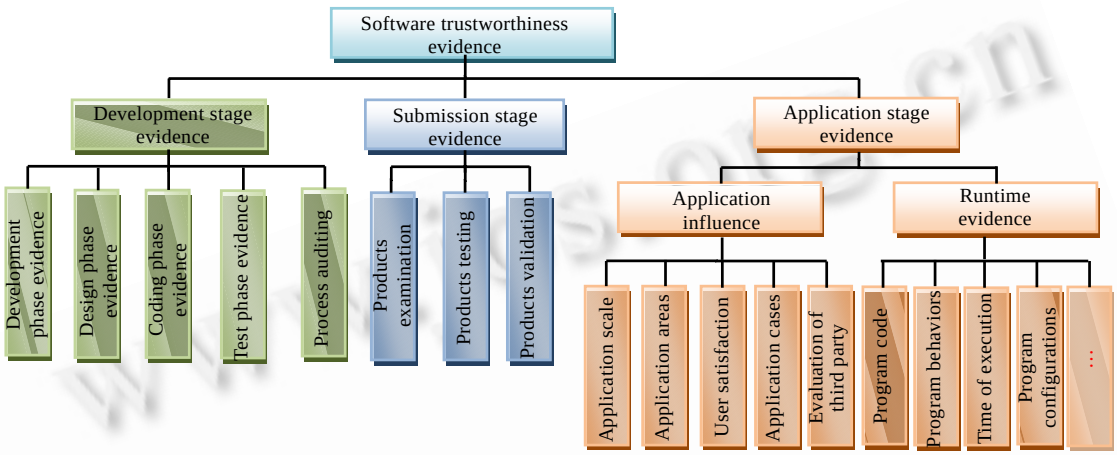


Fig.3 Extended software trustworthiness evidence model

图 3 扩展的软件可信证据模型

3 基于可信计算的软件运行时可信证据收集机制

结合 TPM 的基本安全功能和可信虚拟机的隔离机制,本文提出了一个基于可信证据收集代理的运行时软件可信证据收集机制,如图 4 所示.对应于第 2.3 节介绍的扩展的软件可信证据模型,软件可信证据收集包括 4 个部分:开发阶段证据收集、提交阶段证据收集、应用效果证据收集、运行时证据收集.

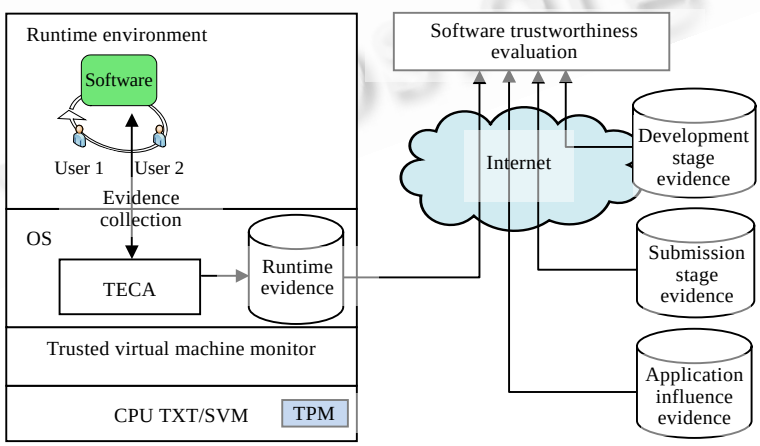


Fig.4 Software trustworthiness evidence collection mechanism based on trustworthiness evidence collection agent (TECA)

图 4 基于可信证据收集代理的软件可信证据收集机制

开发阶段的证据是由软件开发者来提供的,因而在开发阶段需要客观地记录相关的信息,并以可靠的方式交付给评估者.提交阶段证据收集关注于在提交时刻对软件所进行的各种检验和测试的相关证据的收集,需要记录对软件进行的各种分析和测试及获得的相应结论.应用效果证据收集主要是采用问卷和反馈的方式进行,即对使用软件的用户进行调查以获取软件应用的满意度、广泛程度、分布领域等.运行时证据收集是在软件实际运行的过程中客观、全面地记录所有与软件可信性相关的信息.运行时证据收集主要采用监控目标软件运行的方式进行,对于特定的应用,也可以由软件主动调用可信证据收集代理来完成应用特定的可信证据收集.

本文在操作系统层中引入一个可信证据收集代理,由它对运行时的程序进行监控,并记录下相关的可信证据.由于是在操作系统层中实现的可信证据收集代理,它可以监控软件运行时的状态并记录相关的信息,从而可以保证运行时软件可信证据的相关性和可获得性.利用可信虚拟机监视器来为可信证据收集代理提供运行环境,其本身的正确执行可以得到保证,从而能够保证运行时证据的客观性.利用 TPM 提供的安全功能,可以确保可信证据的安全.

3.1 可信证据收集代理的正确执行保障

我们利用可信计算技术、最新加载技术和可信虚拟机监视器 TVMM 来保证可信证据收集代理的正确运行.由 TPM,TXT 或者 SVM 支持的 CPU 以及 TVMM 成了可信证据收集代理的可信计算基(trusted computing base,简称 TCB).利用 TCB 来支持可信证据收集代理的原理是:由 TPM 和最新加载技术来保证虚拟机监视器的正确加载,利用虚拟机监视器可以为可信证据收集代理提供强制隔离的运行空间,从而可以保证可信证据收集代理的正确运行.

首先利用最新加载技术在系统运行的任意时刻安全地加载一个虚拟机监视器(virtual machine monitor,简称VMM)或者一个安全内核(security kernel).该VMM的正确执行是由最新加载技术来保障的.启动TVMM之后,TVMM可以为包括可信证据收集代理在内的操作系统内核提供一个隔离的运行环境,这样的一个环境可以称为一个可信域(trusted domain).可信证据收集代理监控应用程序的运行时状态,并记录下相关的软件可信性证据.可信域可以由虚拟机提供,例如Xen[15],也可以结合最新加载技术的CPU来提供.为了保证可信服务的兼容性和可扩展性,我们采用虚拟机来提供隔离的执行环境.

如果运行时可信证据的收集是从系统启动一直持续到系统运行结束,那么也可以采用可信计算的静态信任根,在系统启动时刻启动一个 TVMM,让 TVMM 提供的隔离机制来保证可信证据收集代理持续运行的可信性.

3.2 可信证据收集代理中的可信服务

为了支持可信证据收集代理客观地完成软件运行时可信证据的收集,可以利用 TPM 的基本功能,在可信证据代理中实现了以下功能:度量服务、可信时间服务、签名服务、安全存储服务.

度量服务:在运行时,软件可信证据以各种形态存在,例如代码、配置文件、数据等.需要一种方式来记录各种不同形态的证据.利用 TPM 的完整性报告机制,可以对各种软件可信证据进行客观的记录.

可信时间服务:时间信息在很多情况下就是非常重要的软件可信证据.例如,运行过程中软件运行的时间、软件操作的时间间隔以及相对顺序等,可信证据的时间属性也是一个重要的参考因素.与时间相关的软件可信性属性值的获得也需要可信的时间服务的支持,例如 Web Services 中的响应时间等信息.可信时间服务是通过一个时间服务进程访问一个可信的时间服务器来实现的,通过隔离保护机制,这个可信时间服务能够提供正确的时间戳服务,从而能够为所有需要添加时间记录的软件可信证据提供可信的时间标记.

签名服务:对于获得的软件可信证据,可以利用 TPM 对软件可信证据进行签名,从而可以验证其来源的可信性.

安全存储服务:在软件可信证据收集的过程中,必然涉及到证据的存储和转移,利用 TPM 的安全存储机制,可以有效地保护所获得的证据.

以上的 4 个服务,是可信证据收集代理正确记录可信证据的必需功能.这些功能是可信证据收集代理的主

要组成部分.由于我们的可信证据收集代理是位于系统内核层的,它无法获取一些应用特定的可信证据.为了提供对应用特定的可信证据的收集,可信证据服务代理也对外提供以上 4 个基本功能的服务接口,从而应用程序可以根据需要调用这 4 个服务,以记录和保护那些应用特定的证据.例如,应用程序中所获得的用户的反馈信息等,可以利用这些服务对反馈证据进行保护和记录.

3.3 软件运行时可信证据

在特定软件运行的过程中,除了程序本身的状态之外,那些可能影响程序运行状态的相关对象的状态也都应该被记录下来,以作为评估其相关可信属性的依据.另外,还需要记录的证据就是相关对象的操作.

软件平台P中的一个对象记为o,静态的系统组成可以表示为$P=\{o_1,o_2,\ldots,o_i\}$.依据对象的所属关系来划分,会影响程序运行状态的对象包括:

- 程序自身的代码及其配置文件和动态链接库;
- 系统所提供的内核模块,系统的配置文件等;
- 程序所依赖的其他独立程序及其配置文件以及其他会直接或者间接影响目标程序可信属性的对象.

一个特定的软件,需要记录的运行时可信证据是应用相关的.如何决定哪些对象的状态信息和对象操作应该被记录,是由需要评估的软件可信属性来决定的.也就是说,会影响到指定软件可信属性的信息都应被记录下来.例如,在对程序正确执行的远程证明中,所收集的证据是目标程序所依赖的所有程序和数据对象的集合[16].所有与目标软件o_s可信性评估相关的对象构成一个集合$D(o_s)=\{o_1,o_2,\ldots,o_j\}$,$D(o_s)\subseteq P$.

在运行系统中的各种对象的组成部分在不同层次上体现为不同粒度的元素.例如,从文件粒度来看,一个软件对象可以看成是一个文件的集合.可信证据收集代理只能以它所能分辨的最小粒度的对象来完成对象的度量.文件是最为广泛接受的软件组织粒度,因而本文以文件粒度来完成除输入、输出以外的可信证据的记录;对于程序的输入、输出,直接记录数据的状态.如果需要完成更细粒度的度量,例如程序代码文件中的指定代码块,则需要额外的支持机制.本文中,每个对象o可以表示为一个文件的集合$o=\{f_1,f_2,\ldots,f_j\}$,f_j表示的是组成对象o的一个文件.

对于对象操作的记录,其本质上就是记录一个特定的动作的发出者、接受者以及操作的类型和时间.在时间t,对象o_i作用在对象o_j之上的一个a类动作可以记为$act(o_i,o_j,a,t)$.在系统中,可能的操作类型包括读、写、创建、删除、调用、执行、锁定等.所有记录下来的操作形成一个集合$EventList$.

3.4 软件可信证据收集

软件可信证据收集是对相关对象的状态进行度量的过程,涉及两个主要的问题:一是度量方法,二是度量时机.

3.4.1 度量方法

在运行平台中,各种对象主要是以文件或者内存中的数据块的方式存在的,可以直接采用 TPM 的完整性度量机制来记录这些对象的状态.当然,对于具有明确语义结构的对象,如程序的配置文件,可以采用基于完整性度量的语义度量方法,即利用完整性度量来记录其每个元素的状态,并利用各个元素之间的语义关系来组织和保存相应的记录.本文实现的可信证据收集代理利用 TPM 的完整性报告机制来记录对象的状态.

3.4.2 度量时机

从记录可信证据的角度出发,我们可以把这些对象划分为以下的类别,每个类别的对象在特定的时机进行度量:

- 可执行文件:可以加载到内存直接执行的可执行代码集合,可以利用 TPM 的完整性度量机制在可执行程序执行之前记录可执行文件的状态.
- 配置文件:包含了程序运行过程中所需要的配置参数的数据文件,在程序使用该配置文件的时刻记录配置文件的状态.
- 动态链接库:目标程序运行时需要动态加载并链接到程序进程空间中的代码集合,包括系统动态链接

库、目标程序自带动态链接库和第三方动态链接库.动态链接库在进行动态链接之前进行度量.

- 系统内核模块:程序运行时依赖的内核模块,包括内核中的关键数据结构,如中断描述符表(interrupt descriptor table,简称 IDT).在程序使用到特定的系统内核模块时度量其状态.
- 程序输入/输出:程序的输入和输出反映了程序的起始条件和运行结果.程序的输入来源有很多种,可能来自键盘输入、进程间通信或者数据文件等,在程序输入时刻记录相应的参数和程序的状态.输出结果也与程序开始运行的状态一同记录下来.

3.4.3 运行时可信证据收集算法

TPM的身份证明密钥(attestation identification key,简称AIK)记为$\{AIK_{pub},AIK_{priv}\}$,其中,AIK_{priv}是私钥,被保护在TPM中.利用密钥k对信息m进行加密的操作记为$Encrypt\{k,m\}$,使用密钥k对信息m进行签名的操作记为$SIG\{k,m\}$,对特定数据块d的Hash度量是$SHA\text{-}1(d)$,TPM提供的Extend操作是TPM_extend.对每一个文件对象,维护了一个状态日志$LOG(f)$.通过可信时间服务获得的当前时间记为t.

根据特定的软件可信评估模型,首先指定一个相关对象集合$D(o_s)$.在运行时,可信证据收集代理监控$D(o_s)$中的对象,相关的对象状态发生改变或者进行了特定的操作时,可信证据收集代理会记录对象状态的改变和相应的动作.证据收集算法分为两个部分:一个是记录对象的状态(算法 1),另一个是记录对象的操作(算法 2).

对象状态可以由组成其的文件的状态来表示,一个文件在特定时间 t 的状态记录元表示为

$$record(f_i,t)=\langle f_i,t,SHA\text{-}1(f_i\|t),SIG(AIK_{priv},SHA\text{-}1(f_i\|t))\rangle,$$

其中,"‖"表示两个字符串的连接操作,而每个文件的所有状态记录构成一个列表$recordList(f_i)$.相应地,o_s在t时刻的状态由所有组成文件对象的状态联结起来表示,即

$$record(o_s,t)=record(f_1,t)\|record(f_2,t)\|\ldots\|record(f_i,t),f_i\in o_s.$$

o_s所有的状态记录保存在$recordList(o_s)$中.系统中对象的操作可以记录为

$$act(o_i,o_j,a,t)=\langle o_i,o_j,a,t,SIG(AIK_{priv},record(o_i,t)\|record(o_j,t))\rangle.$$

在运行平台中,一般情况下可执行文件的状态很少会发生改变,因而可以利用缓存机制来提高度量的效率.类似于文献[17],我们可以引入一个对象状态改变标志 Hash 表 *Changed*,记录对象的状态从上次度量以来是否发生改变,通过对文件读写的监测可以跟踪确定其是否发生更改;另外一个 Hash 表 *RecentMesurement* 记录了文件最近一次度量的结果.记录对象的状态的基本步骤是,首先通过 *Changed* 表确认对象状态是否改变,如果已经发生改变,则利用 TPM 的完整性度量功能得到最新的文件状态,并更新表 *Changed* 和表 *RecentMesurement*;如果对象状态没有发生改变,则直接在表 *RecentMesurement* 中取出对象最近的度量结果.因而,对于状态未发生变化的对象,可信证据收集代理在其未改变状态的时间内只需对其度量 1 次.

对于对象的操作,可信证据收集代理通过监视目标程序的运行来获知特定事件的发生,根据事件类型来决定要记录的信息.通常记录的证据包括对象的状态、操作的类型和时间.

算法 1. 记录对象状态算法:$Measure(o_s,t)$.

输入:Hash表$Changed$,Hash表$RecentMeasurement$,$o_s=\{f_1,f_2,\ldots,f_j\}$,当前时间$t$.

输出:$\{recordList(f_i)|f_i\in o_s\}$,$o_s$度量结果$record(o_s,t)$,$recordList(o_s)$.

1. $record(o_s,t)$="";
2. FOR i=1 to j DO
3. IF ($Changed(f_i)$==true) THEN
4. $m_i=SHA\text{-}1(f_i)$;
5. $RecentMeasurement(f_i)=m_i$;
6. $Changed(f_i)$=false;
7. ELSE
8. $m_i=RecentMeasurement(f_i)$;
9. ENDIF

10. $record(f_i,t)=\langle f_i,t,m_i,SIG(AIK_{pub},m_i||t)\rangle$;
11. $recordList(f_i)=recordList(f_i)\cup\{record(f_i,t)\}$;
12. $record(o_s,t)=record(o_s,t)||m_i$;
13. $LOG(f_i)=SHA\text{-}1(m_i||LOG(f_i))$;
14. ENDFOR
15. $recordList(o_s)=recordList(o_s)\cup\{record(o_s,t)\}$;
16. $LOG(o_s)=SHA\text{-}1((record(o_s,t)||LOG(o_s))$;

算法 2. 记录对象动作算法:$RecordAct(o_i,o_j,a,t)$.

1. $Measure(o_i,t)$; $Measure(o_j,t)$;
2. $act(o_i,o_j,a,t)=\langle o_i,o_j,a,t,SIG(AIK_{priv},o_i||o_j||a||t)\rangle$;
3. $EventList=EventList\cup\{act(o_i,o_j,a,t)\}$;
4. 对于所有会被改变状态的$f_i\in o_j$,$Changed(f_i)$=true;
5. $LOG(act)=TPM\_extend(o_i||o_j|a||LOG'(act))$;

3.4.4 软件可信证据的安全性分析

由于运行环境的开放性,不可避免地存在着对软件可信证据的安全威胁.首先是传统的安全问题,即攻击者的破坏,包括修改已有记录、添加无效记录等.利用可信服务中的安全存储机制,可以保护软件可信证据免受其他攻击者的窃取和篡改.第 2 类会影响证据可信性的因素来自软件运行平台的所有者.平台所有者能够完全控制他们的系统,可以添加、修改和删除任何内容,包括软件可信证据.出于商业利益或者其他的因素,可以伪造或者修改相关的软件可信证据.例如,试图修改可信证据的记录时间.利用可信证据收集代理的可信时间服务,可以准确地记录可信的时间戳,从而防止伪造无效的软件可信性证据.对于不利于平台所有者的记录,平台所有者可能会试图删除这些证据.为了检测是否出现证据缺失,我们利用散列函数的方法来产生度量结果日志.在每一条软件可信证据中,利用f_i对应的$LOG(f_i)$可以检测是否出现证据缺失的情况.$LOG(f_i)$记录了针对同一信息实体f_i之前所有的证据信息的叠加扩展(extend),即

$$LOG(f_i)=SHA\text{-}1(SHA\text{-}1(SHA\text{-}1(\ldots)||LOG''(f_i))||LOG'(f_i)).$$

通过对f_i的所有证据按照时间顺序,用SHA-1 散列函数进行迭代的扩展,将最后获得的值与$LOG(f_i)$比较,如果两者的值是一致的,则没有出现证据缺失.

4 可信证据收集代理的实现

我们利用Linux Security Module(LSM)[18]在Ubuntu 7.10 中实现了一个可信证据收集代理的原型(如图 5 所示).LSM为了支持可扩展的Linux内核访问控制机制而提供了一系列系统安全相关的钩子函数,通过这些植入Linux内核的钩子函数,可以动态地拦截应用程序发起的系统调用,从而监控应用程序的运行情况.在证据收集代理中,利用TPM提供的完整性度量功能对指定的文件进行完整性度量,并利用TPM的安全功能对度量结果进行签名和加密保护.

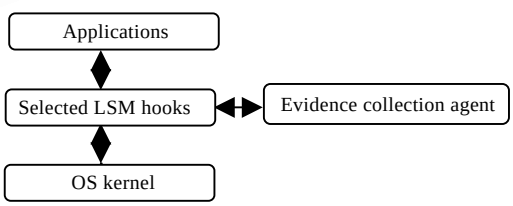


Fig.5 Evidence collection agent based on LSM

图 5 基于 LSM 的证据收集代理

为了收集程序执行过程的证据,我们利用了以下相关 LSM 钩子函数:程序执行、文件操作、进程间通信、

Socket 操作、内核操作.通过重新实现文件加载到内存时的钩子函数,对加载的文件类型进行判定,从而度量可执行程序和动态链接库.在文件被访问时,分析文件操作和目标文件类型,对指定的配置文件或数据文件进行度量.如果需要监控进程间通信,则可以拦截进程间通信信息,记录参与的进程的状态.如果需要记录网络来源数据,则可以监测对网络 socket 的操作,并记录相关的数据.对于涉及到对内核模块进行监控的情况,我们在 *insmod*()和 *modeprobe*()插入证据收集代理调用,从而可以记录内核模块加载的状况.

由于在应用程序的执行中添加了一层可信证据收集代理,必然会增加系统调用的延迟,其中主要的延迟来自于对相关文件的度量.由于系统中的可执行文件对象改变的可能性很小,因而引入更改监控和度量结果 Hash 表可以迅速地降低度量的延迟.在对特定文件对象进行度量之前,首先确定其是否被修改,如果没有,则在 Hash 表中取出最近一次对其度量的结果.因为对于一段时间内未发生改变的对象,可信证据收集代理在这段时间内只需要对其度量 1 次,因此大幅度降低了度量的开销.

我们将在下一节应用中给出可信证据收集代理在分布式计算的应用实例中的性能分析.

5 可信证据收集机制的应用案例研究

本节以分布式计算(distributed computing)为例来分析可信证据代理所获取的运行时软件可信证据,并讨论其所能反映出的软件可信属性.分布式计算通常是由一个中心管理节点和众多的分布式客户端构成的,例如伯克利大学BOINC(Berkeley open infrastructure for network computing)项目[19].管理节点负责协调所有参与客户端的工作,分发任务到各个客户端,并对客户端返回的结果进行处理.这些客户端所在的平台处于各自独立的信任域内,平台所有者可以任意的方式来修改运行在平台中的程序及其状态,传统的安全机制不能确保从客户端获得完全真实、客观的软件可信性证据.本文提出的基于TPM的运行时软件可信证据收集机制可以应用在类似BOINC的分布式计算环境中.

5.1 BOINC客户端的运行时软件可信属性

为了安全、有效地完成众多分布式客户端的协调,中心管理节点需要一种可靠的方式对 BOINC 客户端进行可信性评估,从而能够支持更加灵活的任务分配策略,例如，把重要的计算任务交给最为可信的计算节点.

分布式计算客户端的部分运行时软件可信属性见表 1.这些属性主要关注于客户端在各个计算平台上的性能、安全性、可靠性等相关的运行时信息.在特定的应用场合下,例如不同网络拓扑或者应用目的,可以选择特定的运行时软件可信属性集合,再结合开发阶段、提交阶段、应用效果等其他可信属性来确定一个面向特定应用的软件可信性评估模型.这些不同的评估模型体现为不同的软件可信属性集合以及对这些可信属性进行综合的相关系数.

Table 1 Property of runtime software trustworthiness for BOINC client

表 1 BOINC 客户端的运行时软件可信属性

Runtime software trustworthiness properties	Description
Performance	The time for Boinc client to finish specified tasks
Code integrity	The integrity state of Boinc client code before execution
Response time	The time interval between servers' sending request and receiving response from the Boinc client
Availability	The probability for Boinc client to provide service at specified time interval
Reliability	In the specified tasks set, the ratio of valid computing results
…	…

5.2 BOINC客户端运行时可信证据收集

根据特定的软件可信性模型,利用可信证据收集代理,可以客观地记录 BOINC 客户端的运行时软件可信证据,例如,可以记录系统内和 BOINC 客户端运行相关的操作信息.图 6 所示为可信证据收集代理获取的 BOINC 客户端启动时刻相关程序和动态链接库的状态,以及相应的执行起始时间和程序执行的发起者.图中每条证据记录的每个字段以":"分隔开,其中,第 1 个字段是用两个部分来表示的时间:秒和纳秒,第 2 个字段是对应的可执

行文件在被加载执行时刻的度量值,第 3 个字段是被执行的可执行文件或者动态链接库的名称,第 4 个字段是被执行程序的调用者或者启动者,最后一个字段是我们在可信证据收集代理中记录的对象操作类型.图 6 中,每条记录结尾的 1 表示的是可执行程序和动态链接库的加载.

可信证据收集代理位于操作系统内核,能够动态地监测应用程序的执行情况,从而能够完成对指定信息的记录.

```
1243780826. 596852744:3ad9e3b575b1ba61da8663c4b933778d7362db61 /home/lgu/Desktop/BOINC/boincmgr :boincmgr :1
1243780826. 640798909:8a55df5e6f65c95cbca6f235622602615a99af49 /usr/lib/gconv/UTF-32.so :boincmgr :1
1243780827. 181092331:d39051222d5fc52b9f2dcab690bf505f2be617b6 /home/lgu/Desktop/BOINC/boinc :boinc :1
1243780827. 209014011:6a417d472669a10708272fae197a85282e393499 /home/lgu/Desktop/BOINC/libcudart.so :boinc :1
```

Fig.6 Evidences for executable files in BOINC client initialization

图 6 BOINC 客户端启动时可执行文件的证据

5.3 BOINC客户端的可信性评估

利用可信证据收集代理可以获取到软件可信性属性相关的证据,通过对这些原始信息的分析,可以对运行时软件可信属性进行评估,进而依据软件可信性评估模型来评估软件的可信性.软件运行时的证据记录了对象的状态、操作、时间等信息,利用这些信息可以分析与对象状态、操作相关的软件可信属性,例如,性能、安全性、可靠性、可用性、程序执行正确性等.在安全性方面,可以通过分析目标软件的操作序列来确定软件的运行过程是否被攻击过或者被篡改过.利用可信证据收集代理,通过记录 BOINC 客户端执行特定计算任务的开始和结束时间,可以获得特定平台上 BOINC 客户端的性能评估.类似地,通过检查 BOINC 客户端的代码状态,可以确定客户端代码的完整性.通过检查特定时间段内进行计算服务的时间比例,可以获得 BOINC 客户端的可用性.通过统计选定计算任务集合中 BOINC 客户端提交的有效计算结果比例,可以获得其可靠性.

5.4 可信证据收集代理在BOINC应用中的性能分析

图 7 所示为 BOINC 客户端在使用了可信证据收集代理和没有使用可信证据收集代理情况下的启动时间对比(深色为没有使用可信证据收集代理,浅色为使用的情况),其中,boinc 文件的大小为 1.7MB,这里的启动时间是指 BOINC 客户端的 boincmgr 开始执行到 boinc 加载完毕开始执行的时间间隔.在一次系统启动之后,我们连续 4 次启动 BOINC 客户端,这 4 次执行的时间如图 7 所示,其中,横坐标为启动次数,纵坐标为执行时间(单位:秒).实验是在如下的环境中进行的:Ubuntu 7.10,Linux-2.6.27,CPU:Intel(R) Core™2 Duo CPU E8400 @3.00GHz,2.0GB RAM.从实验结果可以看出,在使用了可信证据收集代理的情况下,boinc 第 1 次启动的时间显著地增加了,这部分增加的时间是用于完成对 boinc 文件的度量.在第 2 次启动开始,由于 boinc 文件没有发生改变,而且度量值是直接在高速缓存中获得的,所以之后的启动时间与没有使用可信证据收集代理时的启动时间是基本相同的.

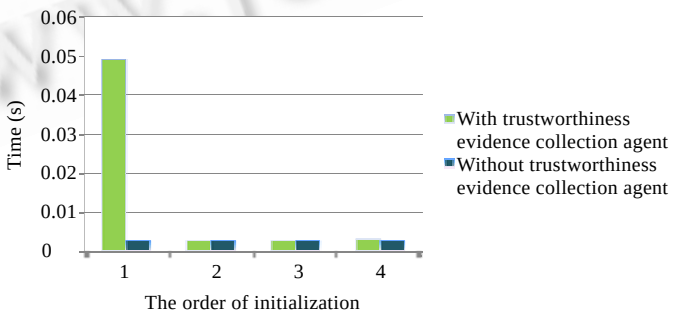


Fig.7 Initiation time for BOINC client

图 7 BOINC 客户端启动时间

为了进一步测试可信证据收集代理的性能开销,我们又在相同的硬件平台上,在内核为 Linux-2.6.29 的

Ubuntu 8.04 中实现并测试了可信证据收集代理.我们在可信证据收集代理关闭和开启的两种状态下各依次启动了 18 次 BOINC 客户端(18 次 BOINC 客户端启动都在同一次系统启动以后).实验结果表明,在可信证据收集代理开启的情况下,boinc 的平均启动时间比在可信证据收集代理关闭的情况下延迟了 7.4%.从总体上来看,由于采用了度量缓存机制,故而对于几乎不会发生更改的可执行文件,随着访问次数的增加,可信证据收集代理的性能开销比例会进一步降低.

6 讨 论

可信证据收集机制是依据可信证据模型来实现的.软件可信评估模型决定了软件可信评估所需要的可信证据,即运行时需要记录的相关信息.可以针对特定的信任评估模型来定制运行时收集的可信证据.例如,假设特定的评估模型不关注时间因素,则可以不记录时间相关的信息;系统中不影响目标软件制品可信性的其他对象也可以被忽略.对于特定应用中的软件可信性评估,需要着重研究其软件可信性评估模型.为了支持不同的软件可信性评估模型,可以引入可配置的软件证据收集机制.

7 相关工作

长期以来,学术界和工业界对软件可信性进行了广泛的研究.最早由Anderson提出了可信系统(trusted system)的概念[20].其后的一段时间内,对可信系统研究的内容主要集中在操作系统自身安全机制和支撑它的硬件环境上.在 1985 年,Laprie[21]提出了可信计算(dependable computing)的概念,将其定义为“系统提供可信赖的计算服务的能力,这种可信赖性是可以验证的”;后来,由Avizienis等人[22,23]指出了系统可信性(dependability)的 6 个主要方面,即可靠性、可用性、安全性、私密性、完整性和可维护性.可信计算组织(Trusted Computing Group,简称TCG)把信任(trust)定义为一个设备将以特定的方式行事的期望[9],如果一个实体能够满足或者达到这个期望,则称这个实体是可信的.从商业的角度,微软公司围绕安全性、私密性、可靠性和业务完整性提出把可信赖的计算(trustworthy computing)作为一种普适的服务(计算体验)[24].美国的许多政府部门,包括DARPA(Defense Advanced Research Projects Agency),NSF(National Science Foundation),NASA(National Aeronautics and Space Administration),NSA(National Security Agency),NIST(National Institute of Standards and Technology),FAA (Federal Aviation Administration),FDA(Food and Drug Administration),都参与了可信软件系统的研究和开发[25,26].他们认为,高可信的软件系统(high confidence system)应该以容易理解和可以预测的方式行事,并且它必须能够抵抗恶意攻击和自然发生的灾害,同时也不会引发或促使事故的发生以及导致不可接受的损失.ISO/IEC 15408 标准[27]认为,可信是指参与计算的组件、操作或过程在任意的条件下是可预测的,并能抵御病毒和物理干扰.陈火旺等人[28]认为,软件系统的可信性是指系统需要满足的关键性质,违背这些关键性质则会造成不可容忍的损失.王怀民等人[29]从资源共享和资源能力角度出发,认为软件的可信性可以分为身份可信、能力可信和行为可信.TRUSTIE(高可信软件生产工具与集成环境)项目根据软件可信评估的需求制定了一系列的软件可信分级规范,然而相关规范还未全面地考虑软件运行时的可信证据收集.ISO 9126 标准也给出了软件质量的模型[30],并界定了相关的软件质量属性.

信任管理是软件安全特性和其他可信属性逐渐融合的研究领域[29,31,32],关注于软件可信评估和基于软件信任决策.周明辉等人从信任角度出发,介绍了一个基于中间件的可定制信任管理框架来支持软件服务的动态可信性[33].已有的工作主要关注于封闭或者相同管理域内的软件可信性问题,而面向开放环境、跨组织和管理域的可信软件机理研究还处于发展阶段[29].现有的信任管理机制还需要可信的决策依据信息来源.本文提出的可信证据收集机制可以作为开放网络环境下信任管理的决策依据信息来源.

在可信计算技术提出之后,其主要功能远程证明受到了广泛的关注.TCG组织提出的远程证明是一种证明远程平台完整性的解决方案[9].TCG远程证明需要收集与平台完整性相关的证据.但是,可信计算中的远程证明针对的是平台配置的完整性,与系统运行时的可信性还有一定的差异.李晓勇等人提出了针对系统行为的远程证明[34].基于程序在系统中的依赖关系,文献[16]提出了对远程程序执行过程的正确性证明.由Sailer等人[17]提

出的Linux完整性度量架构利用TCG技术来记录系统加载过程中程序的完整性状态,以提供平台启动完整性的证明.而本文面向软件可信性评估,收集了系统文件对象的状态、系统中对象之间的操作,能够支持更为全面的软件可信性评估.

8 结束语

本文介绍了一种基于可信计算技术的软件可信证据收集机制.该可信证据收集机制能够为软件可信性评估提供客观、全面的软件运行时可信证据,为软件可信评估提供可靠的前提和基础.利用 TPM 的安全功能、最新加载技术以及可信虚拟机,软件可信证据收集代理的执行过程的可信性可以得到保障.本文介绍的可信证据收集机制可以依据特定的软件可信性评估模型来监控特定的软件运行时信息,因而具有很好的应用扩展性.为了支持应用特定的运行时可信证据的获取和保护,可信证据收集代理能够提供由 TPM 支持的可信服务,应用程序通过调用这些可信服务来记录应用特定的可信证据.本文基于 Linux Security Module 实现了软件可信证据收集代理的原型,并讨论了它在分布式计算的客户端可信性评估中的应用.

为了支持不同的软件可信性评估模型,下一步的工作是引入软件运行时监控信息的管理机制,以实现针对不同的可信评估模型进行证据收集机制的配置管理.软件的可信证据收集应贯穿于软件的整个生命周期,我们下一步将会继续研究在软件开发阶段、提交阶段应用本文提出的运行时可信证据收集机制,以客观地收集和保护相关的软件可信证据.

致谢 在此,我们向在本文写作过程中给予指导和建议的北京大学信息科学技术学院的梅宏教授、金芝教授、陈向群教授、赵霞博士、邵凌霜博士表示感谢,对提出宝贵意见的审稿人表示感谢.

References:

[1] European space agency—The inquiry board. Ariane 5 Flight 105 Inquiry Board Report. Paris: European Space Agency Press, 1996.

[2] Stone B. 11 charged in the theft of 41 million card numbers. The New York Times, 2008. http://www.nytimes.com/2008/08/06/business/06theft.html

[3] Yang FQ, Mei H, Lü J, Jin Z. Some discussion on the development of software technology. Acta Electronica Sinica, 2002, 30(12A): 1901−1906 (in Chinese with English abstract).

[4] Lü J, Ma XX, Tao XP, Xu F, Hu H. Research and progress on Internetware. Science in China (Series E), 2006,36(10):1037−1080 (in Chinese).

[5] TRUSTIE (Trustworthy software tools and integration environment). 2009. http://www.trustie.org (in Chinese)

[6] Shen CX, Zhang HG, Feng DG, Cao ZF, Huang JW. Survey of information security. Science in China (Series E), 2007,37(2): 129−150 (in Chinese).

[7] TRUSTIE. Software trustworthiness evidence framework specification working draft 1.0. Version 2, 2009 (in Chinese). http://www.trustie.net/UserFiles/File/%5BTRUSTIE-STE%5D%E8%BD%AF%E4%BB%B6%E5%8F%AF%E4%BF%A1%E8%AF%81%E6%8D%AE%E6%A1%86%E6%9E%B6%E8%A7%84%E8%8C%83%5BV2.0%5D-200906.pdf

[8] TRUSTIE. Software trustworthiness classification specification. Version 2, 2009 (in Chinese). http://www.trustie.net/UserFiles/File/%5BTRUSTIE-STC%5D%E8%BD%AF%E4%BB%B6%E5%8F%AF%E4%BF%A1%E5%88%86%E7%BA%A7%E8%A7%84%E8%8C%83%5BV2.0%5D-200906.pdf

[9] Trusted Computing Group. TCG specification architecture overview (revision 1.4). Revision 1.4, 2007. http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf

[10] Intel. Intel® trusted execution technology. 2006. http://www.intel.com/technology/security/downloads/TrustedExec_Overview.pdf

[11] Intel. Intel® trusted execution technology software development guide. 2008. http://download.intel.com/technology/security/downloads/315168.pdf

[12] AMD. AMD64 Virtualization Codenamed “Pacifica” Technology-Secure Virtual Machine Architecture Reference Manual. 2005. http://www.mimuw.edu.pl/~vincent/lecture6/sources/amd-pacifica-specification.pdf

[13] Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D. Terra: A virtual machine-based platform for trusted computing. In: Scott ML, Peterson LL, eds. Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP 2003). New York: ACM Press, 2003. 193−206.

[14] State Password Administration Committee in China. Functionality and Interface Specification of Cryptographic Support Platform for Trusted Computing. Beijing: State Password Administration Committee in China, 2007 (in Chinese).

[15] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. In: Scott ML, Peterson LL, eds. Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP 2003). New York: ACM Press, 2003. 164−177.

[16] Gu L, Ding X, Deng RH, Xie B, Mei H. Remote attestation on program execution. In: Xu S, Nita-Rotaru C, Seifert JP, eds. Proc. of the STC 2008. 2008. 11−20.

[17] Sailer R, Zhang X, Jaeger T, van Doorn L. Design and implementation of a TCG-based integrity measurement architecture. In: Proc. of the 13th USENIX Security Symp. Berkeley: USENIX, 2004. 223−238.

[18] Wright C, Cowan C, Smalley S, Morris J, Kroah-Hartman G. Linux security modules: General security support for the Linux kernel. In: Proc. of the 11th USENIX Security Symp. Berkeley: USENIX, 2002. 17−31.

[19] Berkeley University. Berkeley open infrastructure for network computing (BOINC). 2009. http://boinc.berkeley.edu/

[20] Anderson JP. Computer security technology planning study. Technical Report, ESD-TR-73-51, Bedford: Electronic Systems Division, L.G. Hanscom Field, 1972.

[21] Laprie JC. Dependable computing and fault tolerance: Concepts and terminology. In: Proc. of the 15th IEEE Int’l Symp. on Fault-Tolerant Computing. Los Alamitos: IEEE Computer Society Press, 1985. 2−11.

[22] Avizienis A, Laprie JC, Randell B, Landwehr CE. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Dependable Secure Computing, 2004,1(1):11−33.

[23] Avizienis A, Laprie JC, Randell B. Fundamental concepts of dependability. Technical Report, cs-TR-739, Newcastle University, 2001.

[24] Microsoft. Trustworthy computing. 2002. http://www.microsoft.com/mscorp/twc/default.mspx

[25] High Confidence Software and Systems Coordinating Group. High confidence software and systems research needs. 2001. http://www.ccic.gov/pubs/hcss2research.pdf

[26] Sub committee on networking and information technology research and development committee on technology national science and technology council. The Networking and Information Technology Research and Development Program. 2009. http://www.nitrd.gov/Pubs/2009supplement/NITRD-09Supp_FINAL-Web.pdf

[27] ISO/IEC. ISO/IEC 15408-1: Information technology—Security techniques—Evaluation criteria for IT security—Part 1: Introduction and general model. 1999. http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=27632

[28] Chen HW, Wang J, Dong W. High confidence software engineering technologies. Acta Electronica Sinica, 2003,31(12A): 1933−1938 (in Chinese with English abstract).

[29] Wang HM, Tang YB, Yin G, Li L. Trustworthiness of Internet-based software. Science in China (Series E), 2006, 36(10): 1156−1169 (in Chinese).

[30] ISO. ISO/IEC 9126 software engineering—Product quality—Part 1: Quality model. 2001. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749&commid=45020

[31] Xu F, Lü J. Research and development of trust management in Web security. Journal of Software, 2002,13(11):2057−2064 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/2057.pdf

[32] Jøsang A, Ismail R, Boyd C. A survey of trust and reputation systems for online service provision. Decision Support Systems, 2006, 43(2):618−644.

[33] Zhou MH, Mei H, Jiao WP. A customizable trust management framework based on middleware. Acta Electronica Sinica, 2005, 33(5):820−826 (in Chinese with English abstract).

[34] Li XY, Shen CX, Zuo XD. An efficient attestation for trustworthiness of computing platform. In: Proc. of the 2006 Int'l Conf. on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2006). Washington: IEEE Computer Society Press, 2006. 625−630.

附中文参考文献:

[3] 杨芙清,梅宏,吕建,金芝.浅论软件技术发展.电子学报,2002,30(12A):1901−1906.

[4] 吕建,马晓星,陶先平,徐锋,胡昊.网构软件的研究与进展.中国科学(E 辑),2006,36(10):1037−1080.

[5] TRUSTIE.国家可信软件资产库及协同生产环境.2009. http://www.trustie.org

[6] 沈昌祥,张焕国,冯登国,曹珍富,黄继武.信息安全综述.中国科学(E 辑),2007,37(2):129−150.

[7] TRUSTIE.软件可信性证据框架规范.2009. http://www.trustie.net/UserFiles/File/%5BTRUSTIE-STE%5D%E8%BD%AF%E4%BB%B6%E5%8F%AF%E4%BF%A1%E8%AF%81%E6%8D%AE%E6%A1%86%E6%9E%B6%E8%A7%84%E8%8C%83%5BV2.0%5D-200906.pdf

[8] TRUSTIE. 软件可信分级规范.2009. http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf

[14] 中国国家密码管理局.可信计算密码支撑平台功能与接口规范.北京:中国国家密码管理局,2007.

[28] 陈火旺,王戟,董威.高可信软件工程技术.电子学报,2003,31(12A):1933−1938.

[29] 王怀民,唐扬斌,尹刚,李磊.互联网软件的可信机理.中国科学(E 辑),2006,36(10):1156−1169.

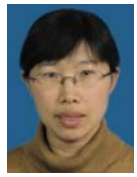
[31] 徐锋,吕建.Web 安全中的信任管理研究与进展.软件学报,2002,13(11):2057−2064. http://www.jos.org.cn/1000-9825/13/2057.pdf

[33] 周明辉,梅宏,焦文品.基于中间件的可定制信任管理框架.电子学报,2005,33(5):820−826.

古亮(1982−),男,四川隆昌人,博士生,CCF 学生会员,主要研究领域为软件工程,可信计算,系统安全.

郭耀(1976−),男,博士,副教授,CCF 高级会员,主要研究领域为系统软件,低功耗系统设计,编译技术,软件工程.

王华(1978−),男,博士,CCF 学生会员,主要研究领域为操作系统,可信计算,系统安全.



邹艳珍(1976−),女,博士,讲师,CCF 会员,主要研究领域为软件工程,软件复用,软件构件技术.

谢冰(1970−),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件复用,软件构件技术.

邵维忠(1946−),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件工程环境,面向对象技术,软件复用与构件技术.