可信终端动态运行环境的可信证据收集机制

谭 良[1,2],陈 菊[1],周明天[3]
(1. 四川师范大学计算机学院,四川成都 610068;2. 中国科学院计算技术研究所,北京 100190;
3.电子科技大学计算机科学与工程学院,四川成都 610054)

摘 要: 可信计算的链式度量机制不容易扩展到终端所有应用程序,因而可信终端要始终保证其动态运行环境的可信仍然困难.为了提供可信终端动态运行环境客观、真实、全面的可信证据,提出了可信终端动态运行环境的可信证据收集机制.首先,在可信终端的应用层引入一个可信证据收集代理,并将该代理作为可信平台模块(trusted platform module,简称 TPM)链式度量机制的重要一环,利用 TPM 提供的度量功能保证该代理可信;然后通过该代理收集可信终端的内存、CPU、网络端口、磁盘文件、策略配置数据和进程等的运行时状态信息,并利用 TPM 提供的可信存储功能,保存这些状态信息作为终端运行环境的可信证据,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,为支持面向不同应用的信任评估模型提供基础.在 Windows 平台中实现了一个可信证据收集代理的原型,并以一个开放的局域网为实验环境来分析可信证据收集代理所获取的终端动态运行环境可信证据以及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: 可信计算; 可信平台模块; 动态运行环境; 可信证据; 可信终端
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112 (2013)01-0077-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.01.015

Trustworthiness Evidence Collection Mechanism of Running Dynamic Environment of Trusted Terminal

TAN Liang[1,2], CHEN Ju[1], ZHOU Ming-tian[3]
(1. *College of Computer, Sichuan Normal University, Chengdu, Sichuan* 610068, *China*;
2. *Institute of Computing Technology of Chinese Academy of Sciences, Beijing* 100190, *China*;
3. *School of Computer Science & Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan* 610054, *China*)

Abstract: Chain measurement mechanism of trusted computing don't easily extend to all applications in the terminal, so it is difficult for the terminal to always maintenance trust of the dynamic running environment of the terminal. To collect trustworthiness evidence in an objective, genuine and comprehensive way, this paper proposes a trustworthiness evidence collection mechanism of trusted terminal running dynamic environment. Firstly, a trusted evidence collection agent, whose creditability is assured by the measurement function of trusted platform module (TPM), is introduced by an application system in the terminal, the main function of which is collecting the information of the terminal dynamic running environment including memory, process, CPU, port of net, disk files, configure data and so on, and saving those evidences in Database or files by TPM. This mechanism has good scalability for various trustworthiness evaluation models. This paper also implements a prototype of the agent in Windows platform, and analyses the performance of agent in a local network distributed computing environment. This application demonstrates the feasibility of this mechanism.

Key words: trusted computing; TPM (trusted platform module); running environment; trustworthiness evidence; trusted terminal

1 引言

目前,可信计算是信息安全技术研究的热点[1~10],因其结合了底层的计算技术和密码技术,在防御终端外部攻击的同时也能隔绝终端内部的威胁.根据可信计算组织的标准[1],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:

收稿日期:2011-06-21;修回日期:2012-03-18
基金项目:国家自然科学基金(No.60970113);四川省青年科技基金(No.2011JQ0038)

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).可信计算的基本思想是在计算机系统中首先建立一个信任根,再建立一条信任链,一级测量认证一级,一级信任一级,把信任关系扩大到整个计算机系统,从而确保计算机系统的可信.因此,信任根、信任链传递和度量是可信计算的基本问题[2].

TCG 组织规定,TPM 是可信计算平台的核心模块和信任根.其本身是一种 SoC 芯片.TPM 不仅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,而且还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).信任链传递过程[1]为 CRTM→BIOS→OS Loader→OS→application.度量的方式如下:

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

度量过程首先从 CRTM(Core Root of Trust for Measurement)对 BIOS 进行完整性度量,将度量结果存储于可信平台 PCR 模块中,并且保存相应度量日志信息,再将保存在 PCR 中的度量结果与基准值比较,判别 BIOS 的完整性[4,5],若完整性没被破坏,则信任 BIOS 并将控制权传递给 BIOS;接着是用 BIOS 对 OS Loader 进行度量,将度量结果保存于可信平台模块 PCR 中,并相应保存度量日志信息,同样与基准值比较判断 OS Loader 的完整性,若完整性得以保持,则信任 OS Loader 并相应传递控制权;最后从 OS Loader 度量 OS 的完整性,直至将度量进行到应用程序.这样就形成了信任链,并实现了信任传递的目的.

但是 TCG 的该链式度量机制很难将信任传递到终端的应用层[2],这是因为,首先 TPM 是一个存储资源有限的芯片,终端所有应用程序的度量模式不可能完全采用 $PCR\_Extend(m)$,度量结果也不可能都存储于 PCR 中;其次,终端应用程序执行次序并不存在固定的串行关系,所以增量度量的值不容易确定;第三,对应用程序进行完整性度量[4,10,11]存在一定难度.这是由应用程序的特点决定的.(1)一个应用程序的执行可能涉及许多相关的其他模块(如内核模块、静态库、动态库、脚本、插件等),而这些模块被加载的顺序每次也不一定相同.例如一个动态链接的程序运行时需要加载动态库,而一个脚本执行却需要调用解释器;(2)一个应用程序是否可信、功能是否正常合理,还与其配置文件、安全策略文件等是否保持完整性有很大的关系;(3)在用户的每一次操作活动期间,所执行的应用程序不一定每次都相同,执行顺序也没有必然的规律,且不一定会用到所有被允许的应用.上述特点导致操作系统在装载或执行应用程序时,不可能对每个应用程序进行有效的完整性度量,保证其可信性.这样导致了恶意应用程序可能破坏终端动态运行环境[12],对于运行的整个终端也就意味着存在风险、不可信.因此,如何保证终端动态运行环境的可信是一个非常紧迫的研究课题[2].

2 相关工作

保证终端动态运行环境的可信的一个主要途径是"度量代码的完整性",这一方法的难点在于应用程序的完整性度量.目前,在这一方面已经取得了一些成果.BIND[13]是一个应用于分布式系统的运行时代码完整性验证服务,它度量的目标不是整个程序代码而是程序运行中的输入输出数据,它通过对数据的完整性测量来证明程序代码的完整性.其概念模型如图 1 所示.在这个模型中,用户的命令和 IP 地址作为输入数据,获得的反馈信息作为输出数据,BIND 系统能够保证这些数据在 P_A、P_B、P_C 不被篡改.但 BIND 不够灵活,比较适合于嵌入式系统.

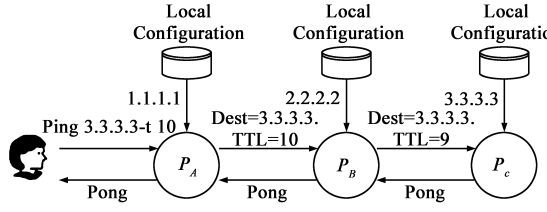

图1 BIND系统的概念模型

Pioneer[14]是第一个在不可信的环境中提供外部代码完整性验证的服务系统,其基本框架和工作流程如图 2 所示.与 BIND 不同,整个验证过程均基于软件方法,验证方式采用"校验和".应用 Pioneer 进行代码验证有许多限制条件,如可信第三方必须知道被验证代码运行环境中的 CPU 型号、时钟频率、内存时延等硬件信息,Pioneer 不支持对称多线程,不能产生系统管理中断调用等等,除此以外,还必须保证验证方法采用的是最优的"校验和"获取算法.文献[15]详细指出了 Pioneer 系统的不足.显然,Pioneer 离实际的应用还有较大的距离.

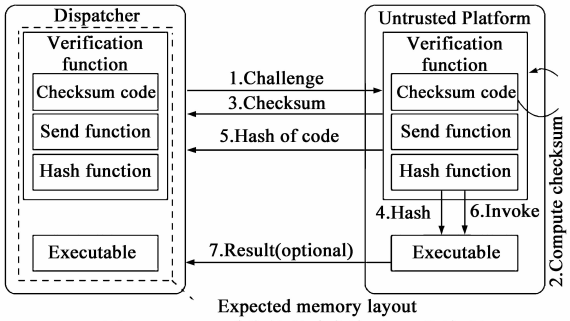

图2 Pioneer系统的基本框架和工作流程

CoPilot[16]是一个保障 Linux 内核运行时完整性的系统,防止 Linux 在内存中被 rootkit 篡改.其框架如图 3 所示.它周期性地扫描内存中的内核,计算内核关键部分的 Hash 值并与期望值进行比较,从而判定内核的状态并报告给外部系统.CoPilot 运行在 PCI 卡上,采用 DMA 方式直接访问内存,通过特定的端口与评估者进行通信.在现有的代码完整性证明系统中,CoPilot 是比较接近实用的一个系统,但仍然有很多限制,首先,CoPilot 没有扫描 CPU 的寄存器,所以不会知道 CPU 当前运行的代码区域,具有盲目性;其次,CoPilot 并没有度量内核的数据段,这给 rootkit 篡改数据留下了隐患.

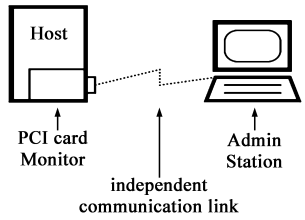

图3　CoPilot系统的基本框架

显然,文献[13,14,16]主要研究的是在不可信的终端环境中如何度量应用程序代码,度量的方法与 TCG 的度量机制不同,也不涉及信任传递.文献[17]提出一种动态可信应用传递模型(DATTM),在保持应用装载的灵活性基础上,着重考虑了应用之间的权限隔离问题,最大程度地实现最小特权和按需即知等安全基本原则,进一步改善了系统度量和策略执行的效率和安全问题.但该方法并没完全遵循 TCG 链式度量机制,而且需要建立软件限制表.文献[18]中,采用了与 TCG 完全不同的度量机制,其对应用程序的度量和信任传递采用 DRM 的方式.应用软件的开发者开发的软件需要得到硬件厂商的签名,而可信终端必须对该签名进行测量,测量通过方可运行.应用软件的升级、维护均采用这种方式.整个过程的完成需要 PKI 基础设施的支持,信任链的传递表现为一条可信证书链.采用这种方式的好处是有成熟的基础设施支持.但这种方式涉及到内容提供商和软件制造商之间的利益折中,也涉及到终端用户的隐私保护;文献[19]提出了一个基于 linux 的安全完整性度量系统的设计和实现,所有加载到 linux 的可执行内容在运行前都必须被 TPM 度量,该系统扩展了 TCG 的可信度量到动态可执行内容,也就是该系统可以利用 TPM 从 BIOS 一直度量到应用程序.尽管如此,该方法仍然没有完全遵循 TCG 链式度量机制;文献[20]提出了基于 TPM 的运行时软件可信证据收集机制,扩展了已有的软件可信证据模型,引入了运行时软件可信证据,利用 TPM 提供的安全功能,结合"最新加载技术",在操作系统层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.但是仅仅是针对应用软件,对整个终端的初始环境和运行环境缺乏保障措施;文献[21]利用 TPM 的可信存储技术,通过扩展系统中传统的基于用户身份的访问控制模型到基于代码的访问控制模型.对资源的访问不仅需要检查用户身份,而且需要检查程序的代码 ID,而程序的代码 ID 是通过对程序的完整性度量来体现.该方法尽管可以加强对资源的访问安全,但与 TCG 提出的链式度量机制信任传递模型完全不同,不能保证终端初始环境的可信.文献[22]是利用 TPM 的度量机制建立按需服务的网络平台,包括应用程序的完整性度量.文献[23]基于可信计算的动态完整性度量架构,提出一种基于可信计算的操作系统动态度量架构(DIMA),该架构能按需对系统中活动的进程或模块进行动态实时的完整性度量与监控;文献[24]提出了基于软件行为的可信动态度量,将度量粒度细化到一次行为的引用上,在一定程度上解决了系统工作后的动态可信性问题;这方面的工作还包括文献[4,10]等,这里就不再一一列举.

从上面的分析可以看出,在如何通过对"代码的完整性度量"来保证终端动态运行环境的可信方面已经取得了一些可喜成果.但是以上文献的一个共同特点是,在将可信传递到应用程序时对应用程序的度量并没有完全遵照 TCG 的链式度量机制,大多数文献都是通过建立新的体系结构和框架对应用程序进行完整性度量.我们认为,一方面在终端通过某种方式对所有应用程序进行完整性度量尽管可能,但不容易实现,而且不可能是链式度量,信任没法从一个应用程序传递到一个应用程序,也没法传递到整个终端动态运行环境.另一方面,即使是经过度量的应用程序,也不可能保证终端整个动态运行环境的可信,因为对应用程序进行完整性度量只能保证该应用程序没有被恶意程序修改,但不能保证该应用程序本身是否破坏终端的动态运行环境,如,泄露内存、劫持网络、破坏文件等.因此,通过收集终端运行环境的动态信息,并采用某种信任模型来评估终端运行时环境的可信是另一条可行的办法,而如何在终端动态运行环境里收集客观、真实、全面的信息是这一办法面临的首要问题.因此,本文提出了可信终端动态运行环境的可信证据收集机制,为某种信任模型来评估终端运行时环境提供依据.

3　终端可信证据收集理论模型

用 T_{ED} 表示终端可信证据收集理论模型,T_{Agent} 表示可信证据收集代理,T_{Info} 表示可信证据信息.则有:

$$T_{ED} = T_{Agent} + T_{Info} \quad (1)$$

式(1)必须保证代理的可信和证据信息的可信.

3.1　可信证据收集代理的理论模型

为了保证 T_{Agent} 可信,我们扩展可信终端的信任传递过程为:

CRTM→BIOS→OS Loader→OS→T_{Agent}→Application

如图 4,将 T_{Agent}作为可信平台链式度量的重要一环.这种方式是可行的,T_{Agent}的度量可由 OS 主导完成,并由 OS 将 T_{Agent}程序作为第一个应用程序首先启动.

定义 1 可信证据收集代理 $T_{Agent}=(A_{code},A_{dll},A_{cofile})$,其中 code 代表收集代理的运行代码,dll 代表其相关的动态链接库,cofile 代表该代理的策略配置文件.

T_{Agent}的完整性度量依然遵守 TCG 的链式度量机制.在 TPM 的 24 个 PCR 存储单元中,选用保留的 $PCR_{16},PCR_{17},PCR_{18}$来存储 $A_{code},A_{dll},A_{cofile}$的完整性度量值.得

$$PCR\_Extend(A_{code}):PCR_{16}^{new}\leftarrow SHA-1(PCR_{15}^{old}\parallel A_{code}),$$

$$PCR\_Extend(A_{dll}):PCR_{17}^{new}\leftarrow SHA-1(PCR_{16}^{old}\parallel A_{dll}),$$

$$PCR\_Extend(A_{cofile}):PCR_{18}^{new}\leftarrow SHA-1(PCR_{17}^{old}\parallel A_{cofile}),$$

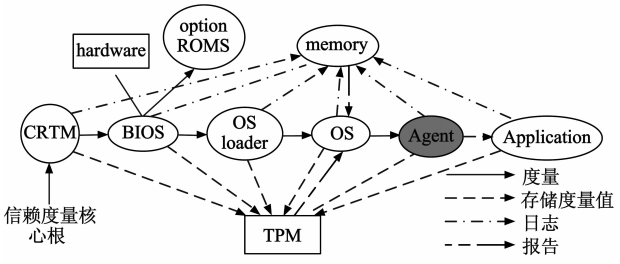

图4 T_{Agent}可信度量示意

3.2 可信证据收集代理的理论模型

定义 2 可信终端运行环境 $E=(P,M,C,N,D,R)$,其中 P 表示进程,M 表示内存,C 表示 CPU,N 表示网络端口,D 表示磁盘文件,R 表示策略数据.

定义 3 终端运行环境证据 $Evid=\{O_\alpha\mid\alpha\in E\}$.

由定义 2 可得 $Evid=\{O_P,O_M,O_C,O_N,O_D,O_R\}$,其中:

O_P 表示终端进程的所有基本信息,$O_P=\sum_{i=1}^{n}O_{P_i}$,$n$ 为整数,表示终端总的进程数,O_{P_i}表示进程 P_i 的基本信息,包括进程名、用户名、用户权限、文件类型、文件位置、文件大小、占用空间、创建时间、修改时间、访问时间、进程属性等.

$O_M=\{M_{sum},\sum_{i=1}^{m}M_{p_i}\}$,其中 M_{sum}表示终端内存的基本信息,包括总内存大小、已使用空间、空闲量、缓存大小和交换区大小、泄露内存的总量等,而 M_{p_i}表示进程 p_i 的内存泄露基本信息,包括泄露内存的进程名、泄露的内存大小等,m 为整数,表示存在泄漏内存的总进程数;

O_C 表示终端 CPU 的使用率;

O_N 表示终端网络端口数据包基本信息,用 d_j 表示经过 j 端口收发的数据包基本信息,包括进程名、收发时间、数据包大小、源地址、源端口、目的地址和目的端口等,则 $O_N=\sum_{j=1}^{h}d_j$,h 表示端口数;

$O_D=\{D_B,D_{file}\}$,其中 D_B 表示磁盘的基本使用信息,包括磁盘类型、磁盘文件系统类型、磁盘容量、可用空间、已用空间、剩余空间等.D_{file}表示磁盘文件的操作信息集合,显然 $D_{file}=\sum_{j=1}^{r}O_{file_r}$,$O_{file_r}$包括 $file_r$ 的文件名、类型、打开方式、路径、大小、占用空间、创建时间、操作的类型(读、写、删除)、操作时间等;O_R 表示操作策略数据的基本信息,包括用户名、进程名、操作方式(读、写、删除)、操作时间等;

定义 4 T_{Agent}的密钥空间 $K=\{SRK,PTK,k,AIK\}$.

出于对私密性和安全性的考虑,T_{Agent}的密钥空间采用与终端 TPM 密钥空间相似的双链式结构,如图 5.SRK 是 TPM 的存储根密钥,SRK 是在激活或重置 TPM 时产生,总存在于 TPM 内,并且是一个不可迁移密钥.SRK 是 TPM 菊花式密钥结构的根.PTK 是平台可迁移存储密钥,是一个非对称密钥,由 SRK 封装,该密钥的功能是用来封装用户可迁移加密对称密钥 k.终端启动后第一载入 TPM 的密钥通常是 PTK,这个密钥是系统管理员所有,其授权信息是公开的.用户可迁移加密对称密钥 k 用来加密用户的信息.AIK 是用户的身份密钥,是一个非对称密钥,由 SRK 封装,不可迁移,用来表明用户的身份和签名.T_{Agent}密钥空间的操作函数就是 TSS 提供的密钥操作函数.因此,后文中凡是需要相关密钥的地方直接使用密钥,略掉密钥操作函数.

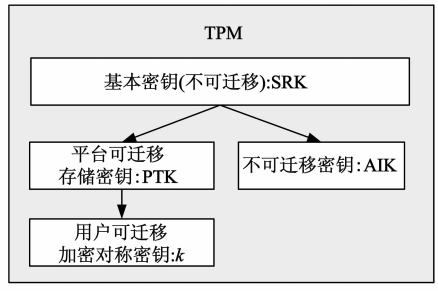

图5 T_{Agent}的密钥空间

定义 5 可信证据的操作函数集 $F=\{get,encrypt,sign,time,store,del\}$.

$get:E\times t\rightarrow O$ 表示在 t 时刻下取得终端运行环境具体对象的可信证据.例如 t 时刻的内存可信证据明文 $get(E(M),t)$;

$encrypt:O\times K\rightarrow O$ 表示用 TPM 的可迁移对称密钥 k 对可信证据 O 加密.如 $encryt(k,O)$

$sign:O\times K\rightarrow O$ 表示用 AIK_{priv}(身份认证密钥的私钥)对可信证据明文的摘要值进行签名.如 $sign(AIK_{priv},SHA-1(O))$.对于获得的可信证据,可以利用 TPM 对可信证据进行签名,从而可以验证其来源的可信性.

time:表示获得终端当前时间.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,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.与时间相关的可信性属性值的获得也需要可信的时间服务的支持,例如网络包的获取时间、内存泄漏的获取时间等信息.*time* 函数是通过获得本机物理时间来实现的,为所有需要添加时间记录的可信证据提供可信时间标记.

store:表示将可信证据经过处理,存储到可信的设施中,如数据库、可信物理存储设备等.在可信证据收集的过程中,必然涉及到证据的存储和转移,*store* 可以完成这一功能.

del:表示在一定授权许可条件下对可信证据的删除.

定义 6　可信证据收集机制 $TEM = \{T_{Agent}, E, O, K, F, t\}$.

可信终端动态运行环境的可信收集机制可以展现为为如图 6.在时刻 t,可信证据收集代理根据使用者发出的收集对象收集命令,按照要求收集终端运行环境中指定对象的明文信息,这些明文信息在收集后会马上交给 TPM 进行再次加工.最后 *TPM* 将加工后的信息在交返给可信证据收集代理,由代理负责处理.

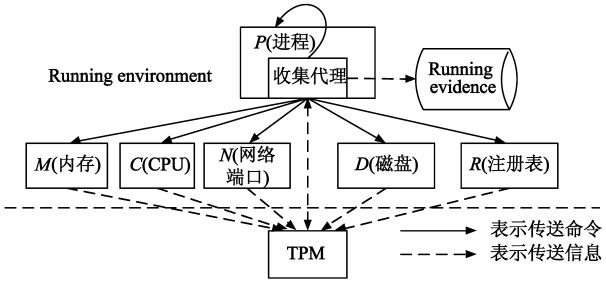

图6　可信终端动态运行环境可信证据收集机制

3.3　终端运行环境可信证据的收集算法

根据 E 指定的相关对象集合,在运行时,可信证据收集代理监控 E 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关的动作.

可信证据收集通常分为两个部分:一个是记录对象的状态,另一个是记录对象的操作.对象 X 在特定时间 t 的状态记录元表示为

$$recordState(X,t) = \langle encrypt(O_X,k), t, SHA-1(O_X \| t), sign(AIK_{pri}, SHA-1(O_X \| t))\rangle$$

其中,"‖"表示两个字符串的连接操作.相应地,X 在 t 时的操作可以记录为:

$$recordAct(p_i,x,a,t) = \langle get(p_i,t), get(X,t), a, sign(AIK_{pri}, recordState(p_i,t) \| recordState(O_X,t))\rangle$$

在 E 集合指定的对象中,有的对象只需记录其状态信息,有的对象既需要记录对象的状态信息,又需要记录对象的动作信息.分析如下:

- 对象 P,只记录所有进程的状态信息;
- 对象 M,只记录内存的状态信息,包括泄漏内存的状态信息;
- 对象 C,只记录 CPU 的状态信息;
- 对象 N,只记录端口数据包基本信息;
- 对象 D,不仅要记录磁盘的基本信息,而且要记录磁盘文件的基本信息和操作信息;
- 对象 R,不仅要记录策略数据的基本信息,而且要记录策略数据的操作信息;

对于对象 P、M、C、N 以及对象 D 中的 D_B,其状态信息改变较为频繁,以 Δt 为时间间隔进行收集,并利用 TPM 的可信存储机制进行存储.对于对象 D 的 O_{file_r} 以及 R,其状态信息改变较少,因而可以利用缓存机制来提高度量的效率.类似于文献[25],我们可以引入一个文件或策略数据状态改变标志 *Hash* 表 *Changed*,记录文件或策略数据的状态从上次度量以来是否发生改变,通过对文件或策略数据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* 记录了文件或策略最近一次度量的结果.记录文件或策略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.

算法 1　记录 O_{file_r} 及 R 对象状态算法:$Measure(O_{file_r}, t)$, $Measure(R,t)$

输入:*Hash* 表 *Changed*, *Hash* 表 *RcentMeasurement*, $O_{file_r} = \{O_{file_1}, O_{file_2}, \cdots, O_{file_r}\}$,当前时间 t.

输出:$\{recordList(O_{file_i}) \mid O_{file_i} \in O_{file_r}\}$, O_{file_r} 度量结果 $recordState(O_{file_r}, t)$, $recordList(O_{file_r})$.

```
1.  Record_state(O_file_r, t) = ""
2.  FOR i = 1 to r DO
3.    IF(Change(O_file_i) = = true)THEN
4.      m_i = SHA-1(O_file_i);
5.      RcentMeasurement(O_file_i) = m_i;
6.      Change(O_file_i) = = false;
7.    ELSE
8.      m_i = RcentMeasurement(O_file_i);
9.  ENDIF
10.  record(O_file_i, t) = ⟨O_file_i, t, m_i, SIG(AIK_pub, m_i || t⟩;
11.  recordList(O_file_i) = recordList(O_file_i) ∪ {record(O_file_i, t)};
12.  record(O_file_r, t) = record(O_file_r, t) || m_i;
```

13. $LOG(O_{file_r}) = SHA\text{-}1((m_i \| LOG(O_{file_r}))$;
14. ENDFOR
15. $recordList(O_{file_r}) = recordList(O_{file_r}) \cup \{record(O_{file_r}, t)\}$;
16. $LOG(O_{file_r}) = SHA\text{-}1((record(O_{file_r}, t) \| LOG(O_{file_r}))$;

$Measure(R, t)$算法与 $Measure(O_{file_r}, t)$,一样,限于篇幅,这里就不再累述.

算法 2 记录 O_{file_r} 及 R 对象动作算法: $recordAct(p_i, O_{file_r}, a, t)$, $recordAct(p_i, R, a, t)$

1. $recordState(p_i, t)$, $Measure(O_{file_r}, t)$;
2. FOR $i = 1$ to r DO
3. IF($Change(O_{file_i}) = =$ true)THEN
4. $recordActList(p, O_{file_r}, a) = recordActList(p, O_{file_r}, a) + recordAct(p_i, O_{file_j}, a, t)$
5. ENDIF
6. $EventList = EventList \cup \{recordAct(p_i, O_{file_j}, a, t)\}$;
7. $LOG(act) = TPM\_extent(p_i \| O_{file_j} \| a \| LOG'(act))$;

$recordAct(p_i, R, a, t)$算法与 $recordAct(p_i, O_{file_r}, a, t)$一样,这里就不再累述.

算法 3 终端进程可信证据收集算法: $CollectEvidence(E, P, K, F, t)$, $processlist$ 表示进程证据集合, $envmember \in E$.

1. $Processlist = ""$;
2. IF($envmember = =$ $process$)
3. FOR $i = 1$ to n DO
4. $processlist = processlist \cup recordState(O_{P_i}, t)$;
5. ENDFOR
6. $m_p = SHA\text{-}1(processlist)$;
7. $Store(processlist)$;
8. $Store(m_p)$;
9. ENDIF

算法 4 终端内存可信证据收集算法: $CollectEvidence(E, M, K, F, t)$, $Memorylist$ 表示内存证据集合, $envmember \in E$.

1. $Memorylist = ""$;
2. IF($envmember = =$ $memory$)
3. $Memorylist = Memorylist \cup recordState(M_{sum}, t)$;
4. FOR $i = 1$ to m DO
5. $Memorylist = Memorylist \cup recordState(M_{p_i}, t)$;
6. ENDFOR
7. $m_m = SHA\text{-}1(Memorylist)$;
8. $Store(Memorylist)$;
9. $Store(M_m)$;
10. ENDIF

算法 5 终端 CPU 可信证据收集算法: $CollectEvidence(E, C, K, F, t)$, $CPUlist$ 表示 CPU 证据集合, $envmember \in E$.

1. $CPUlist = ""$;
2. IF($envmember = =$ CPU)
3. $CPUlist = CPUlist \cup recordState(O_C, t)$;
4. $m_c = SHA\text{-}1(CPUlist)$;
5. $Store(CPUlist)$;
6. $Store(m_c)$;
7. ENDIF

算法 6 终端网络端口数据包可信证据收集算法: $CollectEvidence(E, N, K, F, t)$, $Networkcardlist$ 表示网络端口相关证据集合, $envmember \in E$.

1. $Networkcardlist = ""$;
2. IF($envmember = =$ $Networkport$)
3. FOR $i = 1$ to h DO
4. $Networkcardlist = Networkcardlist \cup recordState(d_i, t)$;
5. ENDFOR
6. $m_d = SHA\text{-}1(Networkcardlist)$;
7. $Store(Networkcardlist)$;
8. $Store(M_d)$;
9. ENDIF

算法 7 终端磁盘可信证据收集算法: $CollectEvidence(E, D, K, F, t)$, $Disclist$ 表示磁盘证据集合, $envmember \in E$.

1. $Disclist = ""$;
2. IF($envmember = =$ $disc$)
3. $Disclist = Disclist \cup recordState(D_B, t)$;
4. $Disclist = Disclist \cup Measure(O_{file_r}, t)$;
5. $Disclist = Disclist \cup recordAct(p_i, O_{file_r}, a, t)$;
6. $m_D = SHA\text{-}1(Disclist)$;
7. $Store(Disclist)$;
8. $Store(M_D)$;
9. ENDIF

算法 8 终端策略文件可信证据收集算法: $CollectEvidence(E, R, K, F, t)$, $policyfilelist$ 表示策略文件证据集合, $envmember \in E$.

1. $policyfilelist = ""$;
2. IF($envmember = =$ $policyfile$)
3. $policyfilelist = policyfilelist \cup Measure(R, t)$;
4. $policyfilelist = policyfilelist \cup recordAct(p_i, R, a, t)$;
5. $M_R = SHA\text{-}1(policyfilelist)$;
6. $Store(policyfilelist)$;
7. $Store(M_R)$;
8. ENDIF

4 可信证据收集机制具体实施

我们在 windows XP 平台了开发了可信证据收集代理的服务端程序和客户程序.服务端是一个 deamon 服务程序,没有用户界面.服务程序的体系结构如图 7 所示.当终端启动时,可信证据收集代理服务端程序自动启动.可信证据收集代理服务端通过读取配置文件和相关本地文件进行一系列的初始化工作,并启动可信证据收集代理服务器线程.可信证据收集代理服务器

在建立连接时,我们并没有选择典型的一个 Socket 主线程开多个子线程的服务器模型,而采用 Socket 非阻塞模式的 Select 模型来将可信证据收集代理服务器设计成只有一个线程,而该线程中包含了多条连接的服务器模型.当可信证据收集代理服务器监听到客户端的连接请求时,建立连接,并创建新的套接字对象.建立连接完成,一方面,可信证据收集代理服务器接收来客户端转发的指令,调用指令实施模块进行指令解析、存储、转发或执行等功能,在辅助文件和 TPM 的帮助下进行证据的可信收集与存储,另一方面,可信证据收集代理服务器将收集的信息存储于本地,并反馈信息给客户端.

客户端程序的运行界面如图 8 所示.为了直观,图 8 信息框中用明文展示收集的可信证据.

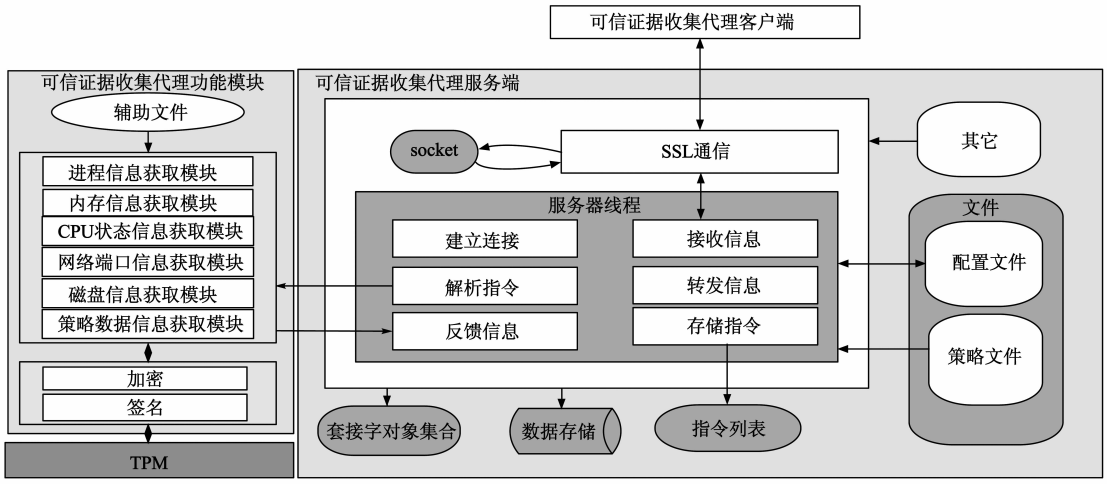

图7 可信证据收集代理服务端的模块结构图

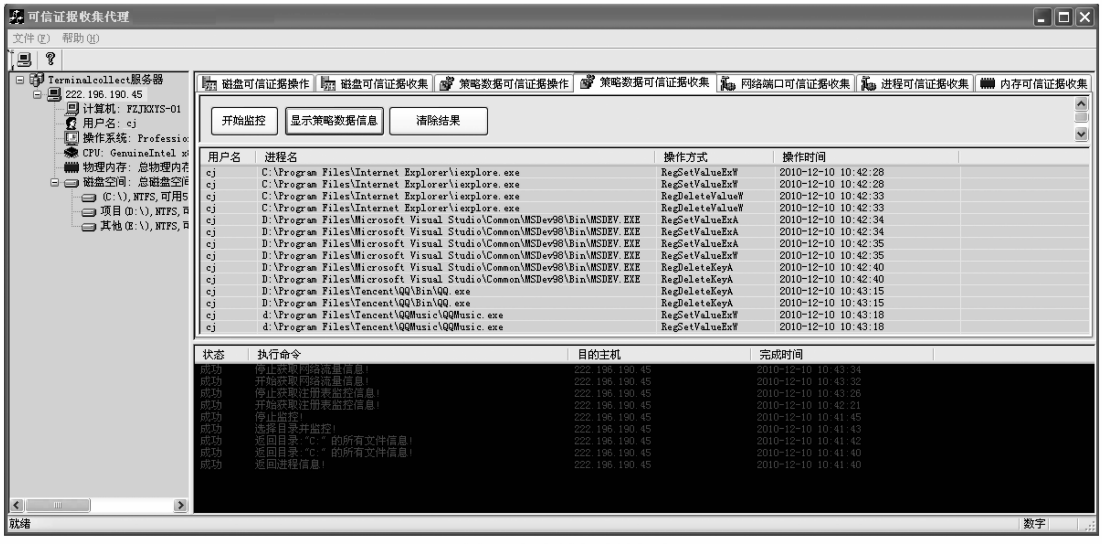

图8 客户端运行界面

我们用 Jtpm 模拟 TPM 的功能,Jtpm 是一个具有 TPM 功能的中间件平台.TPM 及可信证据收集代理服务端程序之间的关系如图 9 所示.由图 9 可以看出,可信证据收集代理服务端程序与 TPM 之间由 TSS 链接.TSS 由三个逻辑组件构成:TCG 设备驱动程序库(TDDL)、TCG 核心服务(TCS)、TCG 服务提供者(TSP).TDDL 为 TPM 定义了标准接口(TDDLI)以及提供了用户模式和内核模式之间的转换;TCS 提供了标准接口(TCSI)以及管理 TPM 的资源;TSP 为应用程序提供了丰富的面向对象的接口(TSPI);可信证据收集代理服务端程序通过 TSP 对 TPM 进行访问.如图 7 所示,TSP 通过接口 TSPI 接收来自收集代理的命令参数,TSP 做了相应的操作、处理后将命令进一步封装成包通过 TCSI 交给 TCS,TCS 做了相应的操作、处理后将包转化成 TPM 能够识别的字节流经由 TDDL 和 TDD 交送给 TPM.TDD 主要负责接收 TDDL 发送过来的字节流并且将其转发给 TPM,待 TPM 处理完后再将信息传回.

5 讨论

本文中提出的终端可信证据收集机制具有较强的

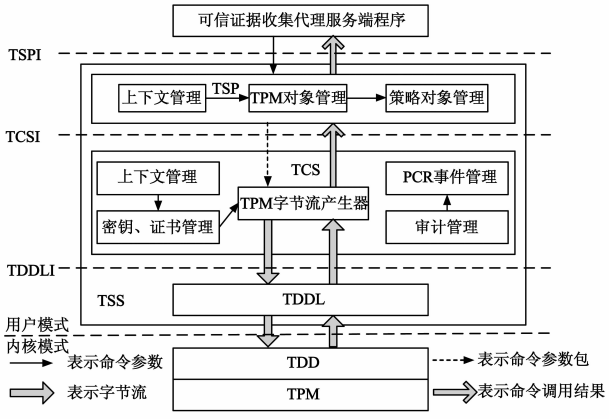

图9 TPM及可信证据收集代理服务端程序之间的关系

可扩展性.可信证据收集机制是依据可信证据模型来实现的.终端可信评估模型决定了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,如PTM model[26]、George′s model[27]以 Sun′s model[28]等.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;系统中不影响终端运行环境可信性的其他对象也可以被忽略.为了支持不同的可信性评估模型,可以引入可配置的可信证据收集机制.

对于信任链如何传递到应用程序,已有的一种方法是开发针对应用程序的完整性度量框架.如文献[4,10,16]等,这也是一种可行的办法.但此种方法的理论框架、合理的实现方式以及性能开销,很少有文献完整地讨论.为此,我们针对应用程序完整性度量的特殊性,跳出 TCG 信任链的传递模式,建立新的终端运行环境证明框架,防止恶意代码在平台中运行.而可信证据的收集机制是这种框架的第一步.

本文提到的终端可信证据收集机制与当前安全领域的研究热点——入侵检测的第一步工作有相似之处,许多著名的入侵检测系统,如,Snort[29]等,其第一步工作也是收集主机运行时信息.甚至与许多主机监控系统、杀毒软件等也有相似之处.但终端可信证据收集机制利用 TCG 信任传递机制保证收集代理是可信的,并利用 TPM 保证证据的来源以及完整性是可信的,因而增加了证据信息的准确性.不过,目前入侵检测技术的许多评估模型,如基于贝叶斯推理的评估模型、基于数据挖掘的评估模型、基于遗传算法的评估模型、基于支持向量机的评估模型、基于神经网络的评估模型以及基于人工免疫的评估模型等,可以应用到对终端运行环境是否可信的评估中.

6 结束语

本文介绍了一种基于可信计算技术的终端动态环境可信证据收集机制.该可信证据收集机制能够为之终端动态环境的可信性评估提供客观、全面的运行时可信证据,为终端动态运行环境的可信评估提供可靠的前提和基础.利用 TPM 的安全功能、以及可信虚拟机,终端动态环境可信证据收集代理的静态可信和执行过程的可信性可以得到保障.本文介绍的终端动态运行环境的可信证据收集机制可以依据特定的可信性评估模型来监控特定的终端运行时信息,因而具有很好的应用扩展性.

为了判断终端环境是否可信,一方面,必须建立以可信证据为基础的信任模型;另一方面,为了支持不同的可信评估模型,引入终端运行时可信证据收集的管理机制,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.

参考文献

[1] Trusted Computing Group. TPM main specification, Version 1.2 rev. 85 [EB/OL]. http://www. trustedcomputinggroup. com, 2011-4-25.

[2] 沈昌祥,张焕国,冯登国等.信息安全综述[J].中国科学 E 辑,2007,37(2):129 – 150.
Shen Changxiang, Zhang Hu- anguo, Feng Dengguo, etal. Survey of information security [J]. Science in China, Series: E, 2007, 37(2): 129 – 150. (in Chinese)

[3] B Kauer. OSLO: Improving the security of trusted computing [A]. Proceedings of the 16th USENIX Security Symposium [C]. Boston, MA, USA, 2007. 6 – 10.

[4] T Jaeger, R Sailer, U Shankar. PRIMA: policy-reduced integrity measurement architecture[A]. SACMAT '06[C]. Lake Talon, California, USA, 2006. 19 – 28.

[5] L Gu, X Ding, R H Deng, B Xie, H Mei. Remote attestation on program execution[A]. Conference on Computer and Communications Security Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing[C]. Alexandria, Virginia, USA, 2008. 11 – 20.

[6] H Wang, Y Guo, X Chen. Saconf: Semantic attestation of software configurations[A]. ATC '09: Proceedings of the 6th International Conference on Autonomic and Trusted Computing, Brisbane[C]. Australia, 2009. 120 – 133.

[7] Aarthi Nagarajan, Vijay Varadharajan, Michael Hitchens, ALOPA: Authorization logic for property attestation in trusted platforms[A]. Proceedings of the 6th International Conference on Autonomic and Trusted Computing[C]. Brisbane, Australia, 2009. 134 – 148.

[8] C Gebhardt, C Dalton. Lala: a late launch application[A]. Proceedings of the 2009 ACM Workshop on Scalable Trusted Computing[C]. New York, NY, USA, 2009. 1 – 8.

[9] 谭良,孟伟明,周明天.直接匿名证言协议的性能估算新方法[J].计算机学报,2012,35(7):1553 – 1562.

[10] R Sailer, X Zhang, T Jaeger, L v Doorn. Design and implementation of a TCG-based integrity measurement architecture [A]. Proceedings of the 13th USENIX Security Symposium [C]. San Diego, CA, USA, 2004. 223 – 238.

[11] 李晓勇,左晓栋,沈昌祥.基于系统行为的计算平台可信证明[J].电子学报,2007,35(7):1234 – 1239.
Li Xiao-yong, Zuo Xiao-dong, Shen Chang-xiang. System behavior based trustworthiness attestation for computing platform [J]. Acta Electronica Sinica, 2007, 35(7): 1234-1239. (in Chinese)

[12] CNCERT/CC. CNCERT/CC2005-2010 年网络安全工作报告 [EB/OL]. http://www. cncert. org. cn/upload/2005-2010CNCE RTCCAnnualReport_ Chinese. pdf, 2011-5-17.

[13] Elaine Shi, Adrian Perrig, Leendert Van Doorn. BIND: A time-of-use attestation service for secure distributed systems [A]. Proceedings of IEEE Symposium on Security and Privacy[C]. Oakland , CA , USA, 2005. 154 – 168.

[14] Arvind Seshadri, Mark Luk, Elaine Shi, et al. Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms[A]. Proceedings of ACM Symposium on Operating Systems Principles (SOSP)[C]. Brighton, UK, 2005. 1 – 16.

[15] Arvind Seshadri. Pioneer web page[EB/OL]. http://www.cs.cmu.edu/~arvinds/pioneer.html, 2011-04-3.

[16] Nick L Petroni Jr, Timothy Fraser, Jesus Molina, William A Arbaugh. Copilot-a coprocessor-based kernel runtime integrity monitor[A]. USENIX Security Symposium[C]. San Diego, CA, 2004. 179 – 194.

[17] 李晓勇,沈昌祥.一个动态可信应用传递模型的研究[J].华中科技大学学报:自然科学版,2005,33(12):310 – 312.
Li Xiaoyong, Shen Changxiang. Research to a dynamic application transitive trust mode[J]. J Huazhong Univ of Sci & Tech (Nature Science Edition), 2005, 33(12): 310 – 312. (in Chinese)

[18] Garfinkel T, Pfaff B, Chow J, et al. Terra: A virtual machine-based platform for Trusted Computing[A]. SOSP'03[C]. Bolton Landing, New York, USA, 2003. 193 – 206.

[19] R Sailer, X Zhang, T Jaeger, L van Doorn. Design and implementation of a TCG-based integrity measurement architecture [EB/OL]. IBM Research Report, http://www. ece. cmu. edu/~adrian/731-sp04/readings/rc23064. pdf, 2011-2-15.

[20] 古亮,郭耀,王华,等.基于 TPM 的运行时软件可信证据收集机制[J].软件学报,2010,21(2):373 – 387.
Gu Liang, Guo Yao, Wang Hua, et al. Runtime software trustworthiness evidence collection mechanism based on TPM [J]. Journal of Software, 2010, 21(2): 373 – 387. (in Chinese)

[21] Paul England, Butler Lampson, John Manferdelli, Marcus Peinado, Bryan Willman. A trusted open platform[J]. Computer archive, 2003, 36(7): 55 – 62.

[22] Hiroshi Maruyama, Frank Seliger, Nataraj Nagaratnam, et al. Trusted Platform on Demand (TPod)[R]. US Patent Publication (Source: USPTO) No. US 7591014 B2 published on 15-Sep-2009.

[23] 刘孜文,冯登国.基于可信计算的动态完整性度量架构[J].电子与信息学报,2010,32(4):875 – 879.
Liu Zi-wen, Feng Deng-guo. TPM-based dynamic integrity measurement architecture[J]. Journal of Electronics & Information Technology, 2010, 32(4): 875 – 879. (in Chinese)

[24] 庄 ▮,蔡勉,李晨.基于软件行为的可信动态度量[J].武汉大学学报(理学版),2010,56(2):133 – 137.
Zhuang Lu, Cai Mian, Li Chen, Software behavior-based trusted dynamic measurement[J]. Journal of Wuhan University (Natural Science Edition), 2010, 56(2): 133 – 137. (in Chinese)

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

[26] S Yoshizawa, A Baba, K Matsunami, et al. Unsupervised speaker adaptation based on sufficient HMM statistics of selected speakers[A]. Eurospeech 2001[C]. 2001. 1219 – 1222.

[27] Carl I Cohen, Carol Magai, Robert Yaffee, Lorna Walcott-Brown. Racial differences in syndromal and subsyndromal depression in an older urban population[J]. Psychiatric Services, 56: 1556 – 1563.

[28] See S. Sun's grid model[J]. Grid Economics and Business Models, 2004, 23: 173 – 184.

[29] Snort-lightweight intrusion detection for networks[EB/OL]. http://www.snort.org, 2011-4-13.

作者简介

谭　良　男,1972 年生于四川泸县.电子科技大学博士,中国科学院计算技术研究所博士后.研究方向为信息安全、可信计算.
jkxy_tl@sicnu.edu.cn

陈　菊　女,1985 年出生于四川内江,硕士研究生,研究方向可信计算.

周明天　男,1937 生于广西容县,教授,研究方向网络计算、信息安全.