软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(8):2084−2103 [doi: 10.3724/SP.J.1001.2012.04115]

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

可信终端动态运行环境的可信证据收集代理*

谭 良[1,2,3+], 陈 菊[1,2]

[1](四川师范大学 计算机科学学院,四川 成都 610066)

[2](四川省可视化计算与虚拟现实重点实验室(四川师范大学),四川 成都 610066)

[3](中国科学院 计算技术研究所,北京 100190)

Trusted Agent for Collecting Trustworthiness Evidence in Terminal Dynamical Running Environment

TAN Liang[1,2,3+], CHEN Ju[1,2]

[1](College of Computer Science, Sichuan Normal University, Chengdu 610066, China)

[2](Key Laboratory of Visualization in Scientific Computing and Virtual Reality of Sichuan (Sichuan Normal University), Chengdu 610066, China)

[3](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: tanliang2008cn@126.com

Tan L, Chen J. Trusted agent for collecting trustworthiness evidence in terminal dynamical running environment. *Journal of Software*, 2012,23(8):2084−2103 (in Chinese). http://www.jos.org.cn/1000-9825/4115.htm

Abstract: The chain measurement mechanism of trusted computing doesn't easily extend to all applications in the terminal, so it is difficult for the terminal to always maintain the trust of the dynamic running environment of the terminal. To collect trustworthiness evidence in an objective, genuine, and comprehensive way, a trusted evidence collection agent based on TPM is designed and developed. Its main function is collecting the critical objects in the dynamic environment of the terminal, such as memory, process, disk files, network ports, policy data, and so on. First, the static and dynamic creditability of the agent is assured by the measurement function of trusted platform module (TPM) and isolation mechanism of trusted virtual machine monitor (TVMM), and then the creditability of original and transmit of the collecting evidences is assured by the encryption and signature function. This paper also implements a prototype of the agent in Windows platform. Based on the prototype, the paper examines the trustworthiness evaluation for executing the agent program in a local area network distributed computing environment. In this application, the performance of prototype is studied, and the feasibility of this approach is demonstrated.

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

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

* 基金项目: 国家自然科学基金(60970113); 四川省青年基金(60903073)
收稿时间: 2011-01-24; 定稿时间: 2011-08-24

可信仍然较为困难.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,设计并实现了一个基于可信平台模块(trusted platform model,简称 TPM)的终端动态运行环境可信证据收集代理.该代理的主要功能是收集可信终端内存、进程、磁盘文件、网络端口、策略数据等关键对象的状态信息和操作信息.首先,通过扩展 TPM 信任传递过程及其度量功能保证该代理的静态可信,利用可信虚拟机监视器(trusted virtual machine monitor,简称 TVMM)提供的隔离技术保证该代理动态可信;然后,利用 TPM 的加密和签名功能保证收集的证据的来源和传输可信;最后,在 Windows 平台中实现了一个可信证据收集代理原型,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: TPM;运行环境;可信证据;可信终端;代理

中图法分类号: TP309 **文献标识码**: A

可信计算的一个基本目标是为了保证终端的可信,从源头上解决信息安全的威胁问题.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(trusted platform module,简称 TPM)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).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.

但是,可信计算集团TCG(trusted computing group)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:

第一,TPM 是一个存储资源有限的芯片,终端所有应用程序的度量结果不可能都存储于平台配置寄存器 PCR(platform configuration register)中.

第二,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[3−5]存在一定难度,这是由应用程序的特点决定的:

a) 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.

b) 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.

c) 操作系统平台之上的应用一般不是单一的,而且这些应用之间并不存在必然的顺序关系.

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都进行有效的完整性度量来保证其可信性.这就导致了恶意应用程序可能破坏终端动态运行环境[6],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].

文献[7−9]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[10−16]均基于 TPM 的度量功能,通过建立“应用程序代码完整性度量的体系结构和框架”来保证终端动态运行环境的可信,但是以上文献的一个共同特点是,将可信从信任根 TPM 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.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,信任无法从一个应用程序传递到一个应用程序,也无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.因此我们认为,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.本文设计与开发可信终端动态运行环境的可信证据收集代理的目标就是在终端动态运行环境里收集客观、真实、

全面的信息,为某种信任模型来评估终端运行时环境提供依据.

1 相关工作

目前,可信终端动态运行环境的可信证据收集代理的研究成果还比较少.文献[17]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来作为软件可信证据的信息,并保障可信证据本身的可信性.但是在该文献中,并没有详细介绍该代理的设计与实现过程,而且该代理的功能有限,仅仅是针对终端运行环境中应用软件的状态信息和操作信息,对整个终端的初始环境和运行环境均缺乏保障措施.

本文讨论的可信终端动态运行环境的可信证据收集代理与已有的终端监控代理在部分功能上有些类似,而监控代理的研究成果很多,如文献[18–20]等.文献[18]介绍了一个远程监控系统,该监控系统每隔一段时间就自动从远程服务中心获取被监控终端系统的诊断信息,这些信息最后会整合到可检索数据库当中.该远程监控系统可以监控多个终端.其中的一个被监控终端看成是 master(管理者),剩余的终端都从属于 master 为其命名为 slave,master 将 slave 的诊断信息存储内存单元中.该监控系统的基本框架如图 1 所示.该文虽然实现了终端的监控,但是没有充分地考虑到监控系统的安全机制,更不用说存在的终端内部威胁.文献[19]设计了一种测试方法,可以让远程终端系统证明它自身的真实性和可信赖性.在通过了该方法的测试后,终端就可以被授权分配资源、为分布计算提供服务.这种测试方法适用于拥有传统网络接口的普通终端,可以避免潜在的软件、硬件攻击,但是以测试软件来保护测试软件,这种机制本身就存在风险性.文献[20]提出的基于超椭圆曲线密码的混合代理签名方案体现了低通信消耗与低系统开销的设计原则,充分利用了椭圆曲线密码密钥长度短、效率高、安全强度大的优势,但是仍然采用软件自身来保证软件的安全性.无论什么终端监控系统,其第 1 步工作都是收集终端系统中的用户行为信息.

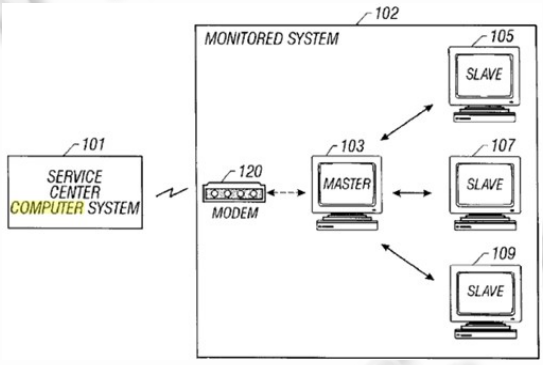


Fig.1 Basic framework of auto remote monitor agent of terminal

图 1 自动远程终端监控系统基本框架

另外,本文讨论的可信终端动态运行环境的可信证据收集代理与已有的入侵检测系统(intrusion detection system)在部分功能上有些类似.入侵检测是从计算机网络或计算机系统中的若干关键点收集信息并对其进行分析,从中发现网络或系统中是否有违反安全策略的行为和遭到袭击的迹象的一种机制.到现在为止,入侵检测系统的理论研究基本成熟,商业化的基于主机的入侵检测产品有:ISS RealSecure OS Sensor、Emerald expert-BSM、金诺网安 KIDS 等.商品化的基于网络的入侵检测产品包括:国外的 ISS RealSecure Network Sensor、Cisco Secure IDS、CA e-Trust IDS、Axent 的 NetProwler 以及国内北方计算中心 NISDetector、启明星辰天阗黑客入侵检测与预警系统和中科网威“天眼”网络入侵侦测系统等.无论是主机入侵检测系统还是网络入侵监测系统,其第 1 步工作仍是收集网络或计算机系统中的用户行为信息.

总结起来,无论是终端监控代理还是入侵检测系统,其共同的缺点都是:

(1) 没有相关机制保证它们自身的可信性;

(2) 要么没有安全措施保证从终端运行环境中得到的信息其来源和传输可信,要么采用比较复杂的安全机制,如 PKI、密钥管理中心等.

因此,本文设计与实现的可信终端动态运行环境的可信证据收集代理,不仅具有通常终端监控代理和入侵检测系统的信息收集功能,而且还要保证其自身的静态可信和动态可信以及所收集到的信息的来源和传输可信.这些功能的实现,均需要可信终端中 TPM 的参与.TPM 有两个重要的作用:

(1) TPM 要保证监控代理是可信的,利用 TPM 的信任传递过程扩展保证监控代理的启动可信,利用 TVMM 提供的隔离技术保证该代理运行可信;

(2) TPM 要给收集到的明文信息进行加密和签名,加密是确保信息在终端和网络传递中的安全,即使被黑客窃听了也不能获得信息的具体明文,签名是确保收到的信息是从要监控的服务端发送过来的,防止有的黑客窃取了信息,冒充被监控端发送虚假信息.

2 可信终端动态运行环境可信证据收集代理的需求规定

可信终端动态运行环境的可信证据收集代理运行在可信终端,通过接受来自监控端的命令对终端动态运行环境里各种对象的信息进行客观、真实、全面的收集,为某种信任模型来评估终端运行时环境提供依据.因此,可信终端动态运行环境的可信证据收集代理应该具有如下的功能需求和非功能需求:

功能需求包括:

- 协议管理.代理客户端发送命令,服务端解析协议并完成相应的功能.
- 信息收集.主要是收集可信终端动态运行环境的内存、进程、磁盘文件、网络端口、策略数据等关键对象的状态信息和操作信息.
- 信息签名与加密.代理需要对信息签名,保证来源可信;也需要对信息加密,保证传输可信.

非功能需求包括:

- 可信保障机制,保障代理服务端静态可信和动态可信.

3 可信终端动态运行环境可信证据收集代理的可信保证机制

对于可信终端动态运行环境可信证据收集代理的非功能需求中的可信保障机制,我们通过 TPM 提供.本节首先介绍 TMP 及其安全机制,然后再重点阐述可信终端动态运行环境可信证据收集代理的链式度量机制.

3.1 TPM及其安全机制

可信计算平台[21]的核心是 TPM.TPM 基于密码体系和 TCG 的硬件平台规范产生、存储并管理密钥,进而由密钥的使用实施各种验证(如认证软、硬件合法性、向远端认证平台身份、加解密数据等),依赖认证和加密的结果,确保计算机系统中各种服务和应用的正常执行.TPM 软件栈 TSS(TPM software stack)是用来支持使用 TPM 的可信计算软件,是向 TPM 提供服务的主要功能模块,它向应用程序提供进入 TPM 的功能接口(这些访问接口在 TCG 文档中均以函数形式给出).其中,TSS 内核服务很重要的一项任务就是实现密钥管理,由 3 个逻辑组件构成:TPM 设备驱动程序库 TDDL(TPM device driver library)、TSS 核心服务 TCS(TSS core service)、可信服务提供者 TSP(trusted service provider).TDDL 为 TPM 定义了标准接口 TDDLI(TDDL interface)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(TCS interface)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(TSP interface).如图 2 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,再将信息以此传回.具体处理机制如图 2 所示.

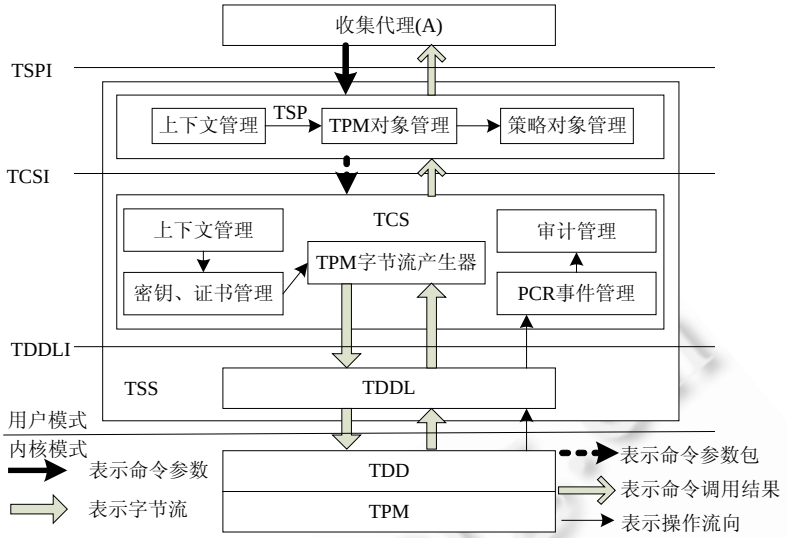


Fig.2 Information processing mechanism of TPM

图 2 TPM 的信息处理机制

TPM 是可信计算平台的核心模块和信任根.信任链传递过程为

CRTM→BIOS→OS Loader→OS→Application.

度量结果存储于平台配置寄存器 PCR 中.度量的方式如下:

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

TPM 一般只提供 24 个 PCR 单元,编号为 0~23,所以 i 的取值范围应为 $0 \leqslant i \leqslant 22$.

3.2 可信终端动态运行环境可信证据收集代理的的链式度量

首先:将含有终端监控代理的可信终端信任传递过程扩展为 CRTM→BIOS→OS Loader→OS→TVMM→Agent→Application,如图 3 所示.

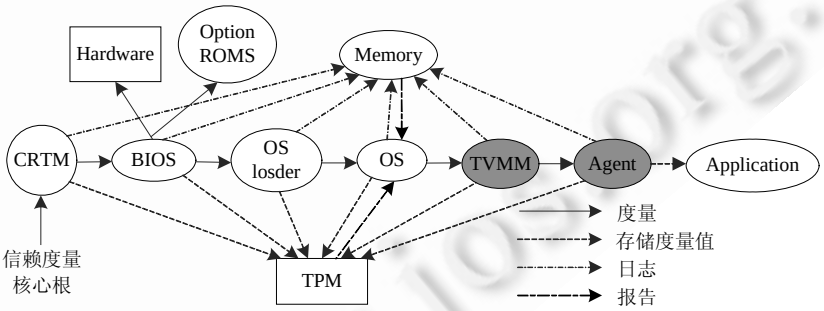


Fig.3 Measurement of TVMM and a trusted evidence collection agent by TPM

图 3 TVMM 和可信证据收集代理的可信度量

定义 1. 可信虚拟机监视器 $TVMM=(TV_{code},TV_{dll},TV_{cofile})$,其中,$TV_{code}$ 代表可信虚拟机监视器的运行代码,TV_{dll} 代表动态链接库,TV_{cofile} 代表该虚拟监视器的策略配置文件.

TVMM 的完整性度量遵守 TCG 的链式度量机制.选用保留的 $PCR_{16},PCR_{17},PCR_{18}$ 来存储 $TV_{code},TV_{dll},TV_{cofile}$ 的完整性度量值,得到:

$$PCR\_Extend(TV_{code}): PCR_{16}^{new} \leftarrow \text{SHA-1}(PCR_{15}^{old} \| TV_{code},$$
$$PCR\_Extend(TV_{dll}): PCR_{17}^{new} \leftarrow \text{SHA-1}(PCR_{16}^{old} \| TV_{dll}),$$
$$PCR\_Extend(TV_{cofile}): PCR_{18}^{new} \leftarrow \text{SHA-1}(PCR_{17}^{old} \| TV_{cofile}).$$

定义 2. 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_{19},PCR_{20},PCR_{21} 来存储 A_{code},A_{dll},A_{cofile} 的完整性度量值,得到:

$$PCR\_Extend(A_{code}): PCR_{19}^{new} \leftarrow \text{SHA-1}(PCR_{18}^{old} \| A_{code}),$$
$$PCR\_Extend(A_{dll}): PCR_{20}^{new} \leftarrow \text{SHA-1}(PCR_{19}^{old} \| A_{dll}),$$
$$PCR\_Extend(A_{cofile}): PCR_{21}^{new} \leftarrow \text{SHA-1}(PCR_{20}^{old} \| A_{cofile}).$$

4 可信终端动态运行环境可信证据收集代理的总体设计

本可信证据收集代理主要分为两大部分:一个是代理服务端,主要任务是实时接收理客户端传送过来的信息、协议解析、命令执行,并将需要的数据收集起来经过 TPM 加工后送给代理客户端;另一个是代理客户端,主要功能是对代理服务端进行控制、传送用户命令,并接收代理服务端的反馈信息.

4.1 可信终端动态运行环境可信证据收集代理服务端的体系结构

可信证据收集代理服务端的体系结构图如图 4 所示.从可信证据收集代理服务端体系结构可以看出,该服务端主要分成 3 个部分:可信证据收集代理服务器内核、可信证据收集功能模块和 TPM 提供的安全功能.

- 可信证据收集代理服务器内核包含建立连接模块、接收信息模块、发送信息模块、协议解析模块、策略配置文件解析模块以及存储信息模块.建立连接模块的功能是通过 Socket 与客户端建立连接,Socket 采用非阻塞模式的 Select 模型来将控制服务器设计成只有一个线程、线程中包含了多条连接的服务器模型.当服务端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(存放在套接字对象集合中).即每个套接字对象集合中的一个套接字对象对应了一条连接,同时也对应了客户端的一个子线程;接收信息模块的功能是接收客户端发送过来的信息以及可信证据收集功能模块采集的信息;发送信息模块的功能是将接收信息模块收集到的可信证据发送给客户端;协议解析模块的功能是解析客户端发送过来的应用层协议;存储信息模块的功能是存储可信证据收集功能模块收集的可信证据;策略配置文件解析模块的功能是在证据收集代理服务端主程序启动时,读取配置文件和相关本地文件进行一系列的初始化工作.
- 可信证据收集功能模块包括进程信息获取模块、内存信息获取模块、CPU 状态信息获取模块、网络端口信息获取模块、磁盘信息获取模块以及策略数据信息获取模块.进程信息获取模块的功能是采集进程的可信证据,如进程名、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;内存信息获取模块的功能是采集内存的可信证据,如总内存大小、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、泄露内存的进程名、泄露的内存大小等;CPU 状态信息获取模块的功能是采集 CPU 的可信证据,即终端 CPU 的使用率;网络端口信息获取模块的功能是采集网络端口数据包的可信证据,如进程名、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;磁盘信息获取模块的功能是采集磁盘的可信证据,如磁盘类型、磁盘文件系统类型、磁盘容量、可用空间、已用空间、剩余空间、磁盘文件名、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;策略数据信息获取模块的功能是采集策略数据的可信证据,如操作策略数据用户名、进程名、操作方式(读、写、删除)、操作时间等.
- TPM 提供的安全功能包括加密、度量.加密主要是对可信证据收集功能模块收集到的可信证据明文信息(用 *m* 表示)进行加密(加密使用的密钥是服务端与 TPM 事先协商好的对称密钥 *k*);度量又分为摘要

和签名,摘要使用的是 SHA-1 算法,即需计算 SHA-1(*m*),而签名使用的密钥为 *AIK* 的私钥(*AIK* 为用户的身份密钥,是一个不可迁移非对称密钥,用来表明用户的身份和签名),签名的内容为摘要 *m* 后的数据.

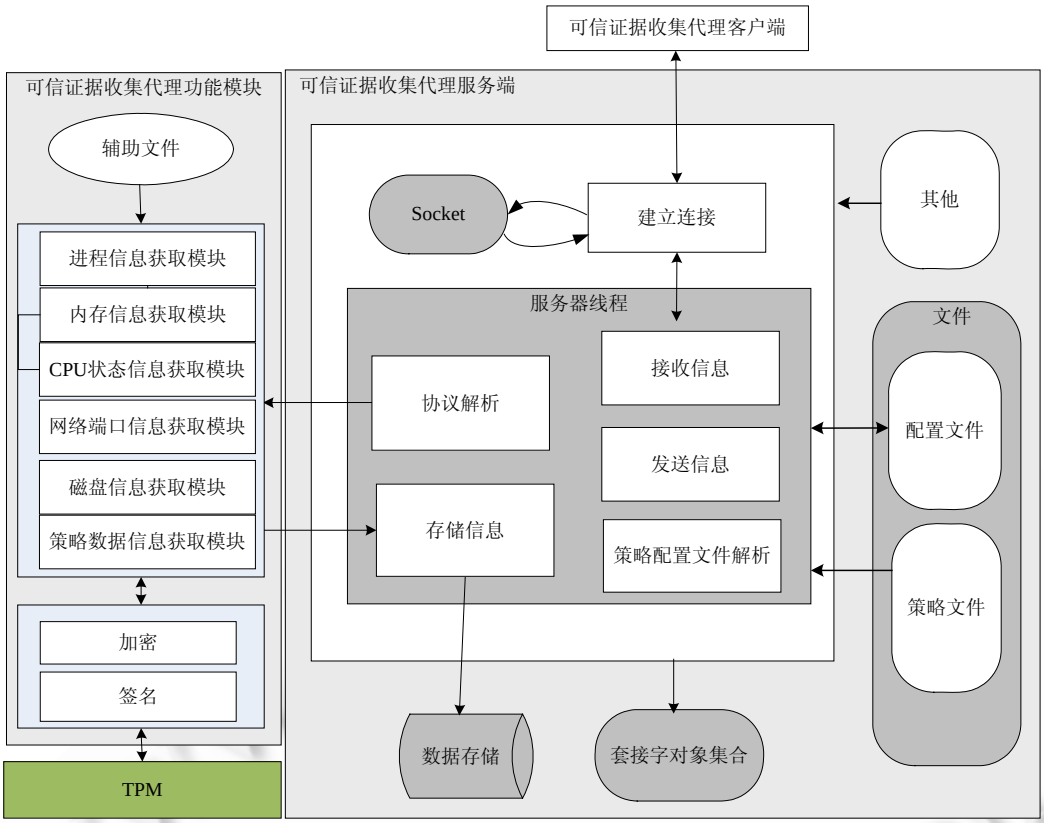


Fig.4 Architecture of server of a trusted evidence collection agent

图 4 可信证据收集代理服务端体系结构

4.2 可信终端动态运行环境可信证据收集代理客户端的体系结构

可信证据收集代理客户端的模块结构如图 5 所示.从可信证据收集代理客户端体系结构可以看出,该客户端主要包含建立连接模块和可信证据收集代理主线程模块.建立连接模块的作用与服务端建立连接模块的作用相同,客户端主线程可以生成多个子线程,一个子线程和一个服务端连接.客户端子线程包含发送信息模块、接收信息模块、信息加密模块、信息解密模块、生成协议模块以及策略配置文件解析模块.发送信息模块的功能是发送客户端的信息给服务端;接收信息模块的功能是接收服务端发送来的信息;信息加密模块的功能是将客户端与服务端通信过程中的敏感信息加密;信息解密模块的主要功能是解密信息;生成协议模块的功能是根据用户操作生成相应的协议由发送信息模块发送到服务端;策略配置文件解析模块的功能是证据收集代理客户端主程序启动时,读取配置文件和相关本地文件进行一系列的初始化工作.

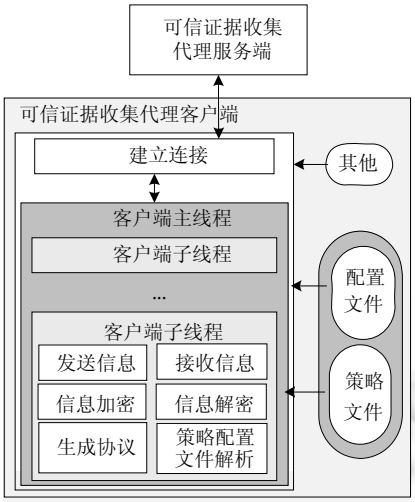


Fig.5 Architecture of the client of a trusted evidence collection agent

图 5 可信证据收集代理客户端体系结构

4.3 可信终端动态运行环境可信证据收集代理通信协议的设计

在应用层,客户端和服务端共同约定了一种通信协议,使得它们在交换信息时遵循统一的规则.该协议的具体包格式如图 6 所示.该协议包只包含两段:一是 Type,二是数据段.

- Type:表示的是客户所选择的指令.由于可信证据收集代理通常需要监控终端的进程、内存、CPU、网络端口数据包、磁盘和策略文件,因此,Type 段的设计如图 6 所示.即 Type 共 32 位、4 个字节,高 28 位保留,用于可信证据收集代理日后的功能扩展.用低 4 位比特的二进制来表示不同的指令,见表 1.
- Data:用来填充与协议相关的数据.例如,当进行文件监控时,监控中心必须将被监控文件的路径发送给服务端,当服务端要发送一个指令的相应反馈结果时就将其填充到 data 字段.

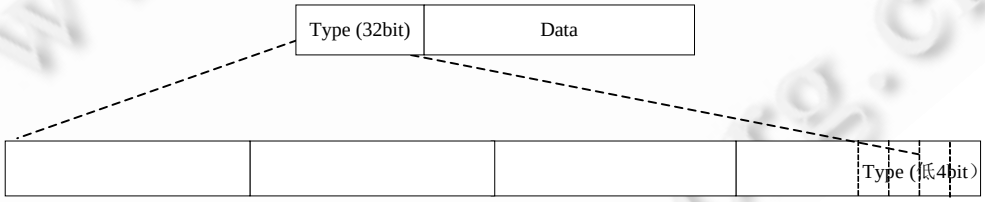


Fig.6 Communication protocol between client and server

图 6 客户端与服务端的通信协议

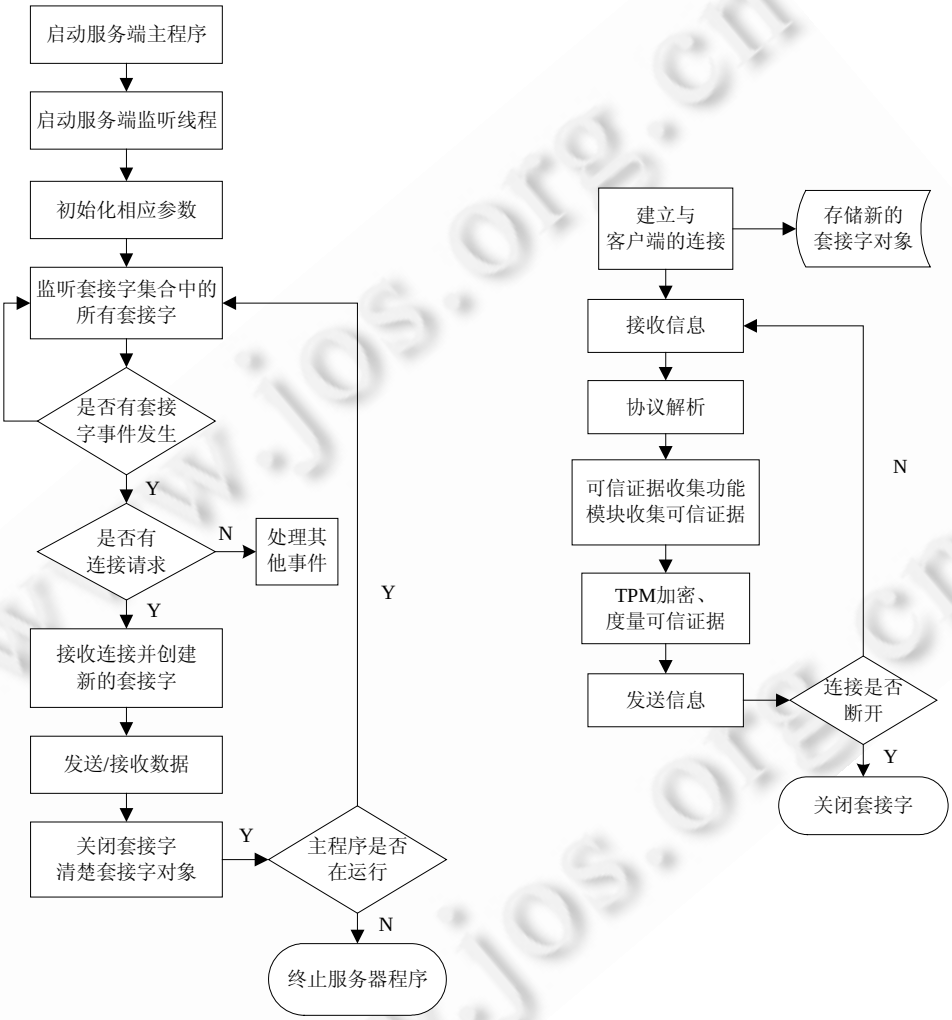
Table 1 Semantic type of communication protocol

表 1 通信协议的语义类型

消息标识	操作描述
0000	收集服务端进程可信证据
0001	收集服务端内存可信证据
0010	收集服务端 CPU 可信证据
0011	收集服务端网络端口数据包可信证据
0100	收集服务端磁盘可信证据
0101	收集服务端策略文件可信证据

4.4 代理服务端和客户端的处理流程

可信证据收集代理的服务端主程序启动时,通过读取配置文件和相关策略文件进行一系列的初始化工作,并启动代理的服务器线程.当代理服务器在建立连接时,我们并没有选择一个 Socket 主线程开多个子线程的服务器典型模型,而是采用 Socket 非阻塞模式的 Select 模型来将代理服务器设计成只有 1 个线程、线程中包含了多条连接的服务器模型.当服务端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.即每个套接字对象集合中的一个套接字对象对应了一条连接,同时也对应了客户端的一个子线程.建立连接完成,代理服务器接收来自客户端发出的指令,调用指令实施模块进行指令解析,转到相应的可信证据收集代理功能模块,对具体的动态运行环境(进程、内存、CPU、网络端口、磁盘和策略数据)搜集可信证据(如图 7 所示).



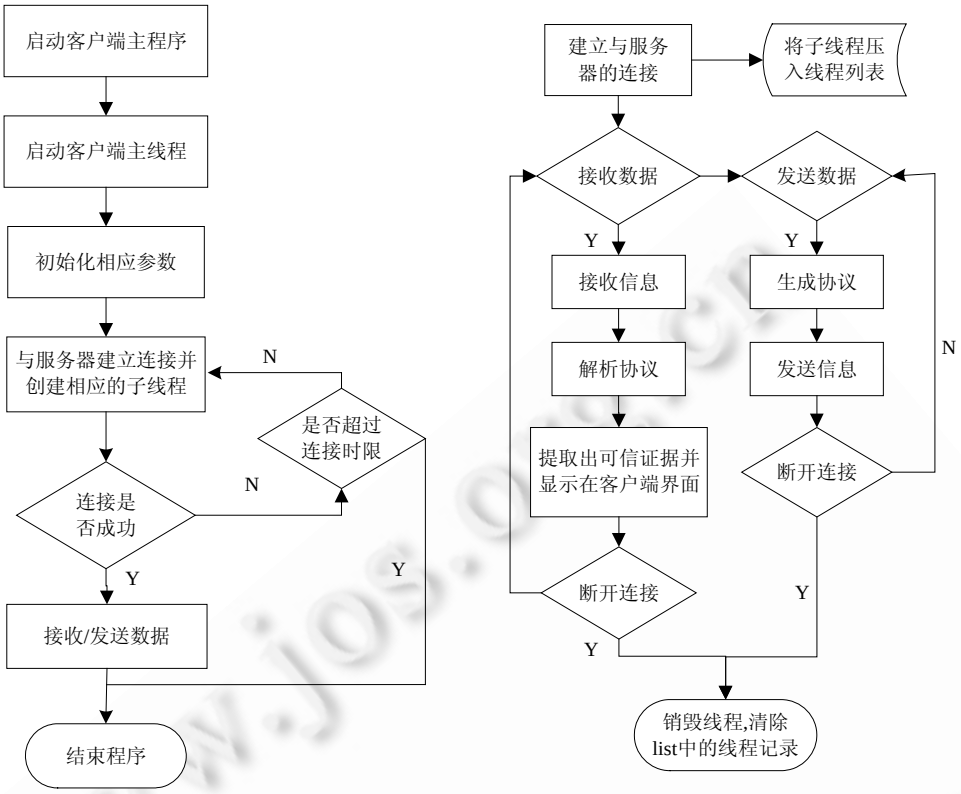
(a) 服务端运行控制流程 (b) 服务端信息处理流程

Fig.7 Procedure in sever

图 7 服务端处理流程

当可信证据收集代理的客户端程序启动时,也通过读取配置文件和相关策略文件进行一系列的初始化工作.然后客户端启动多个线程,向服务端发出连接请求.建立连接后,由相信的线程收集由服务端返回的反馈信

息,直到用户停止该线程为止(如图 8 所示).



(a) 客户端运行控制流程 (b) 客户端信息处理流程

Fig.8 Procedure in client

图 8 客户端处理流程

4.5 可信证据收集代理的动态执行保障

我们利用可信虚拟机监视器 TVMM 来保证可信证据收集代理的正确动态运行,例如 Xen[22].首先,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;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,这样的环境可以称为一个可信域(trusted domain).可信证据收集代理监控终端动态运行环境状态,并记录相关的可信性证据.

5 可信终端动态运行环境可信证据收集代理在 Windows 平台上的实现

我们在 Windows XP 平台上实现了可信终端动态运行环境可信证据收集代理,开发环境是 VC 6.0+OpenSSL,OpenSSL 的作用是模拟 TPM 的摘要和加密功能.可信终端动态运行环境可信证据收集代理的服务端是一个 deamon 程序,没有用户界面,我们采用 Socket 非阻塞模式的 Select 模型将服务器设计成只有 1 个线程、线程中包含了多条连接的服务器模型.可信终端动态运行环境可信证据收集代理的客户端是一个一般的 Windows 应用程序,我们采用多线程模型,客户端主线程实时的根据客户端使用者的操作(进程、内存、CPU、网络端口、磁盘和策略数据可信证据收集)来建立相应的子线程,子线程与服务端建立相应的链接并与服务端通信.

可信证据收集代理客户端的核心任务是传送指令和信息处理,功能和实现算法简单.可信证据收集代理服务端的核心任务是收集终端动态环境的信息,服务端的信息收集模块有:内存信息收集模块、策略数据信息收集模块、进程信息收集模块、磁盘文件信息收集模块以及网络信息收集模块.这些模块收集到的相关信息需要经过 TPM 进行加工处理,以保证信息来源和网络传输的可信性.因此,本节重点介绍这些模块在 Windows 平台上的实现算法.

5.1 内存信息收集模块

某些入侵活动在入侵系统时,可能需要在内存页非法写入信息或者非法占用内存等,我们可以通过内存页的监控来发现这些入侵活动.内存信息获取模块的功能是采集内存的可信证据,如总内存大小、已使用空间、空闲量(GlobalMemoryStatus())、缓存大小和交换区大小(getMemoryPageFileInfo())、泄露内存的总量、泄露内存的进程名、泄露的内存大小等,如图 9 所示.

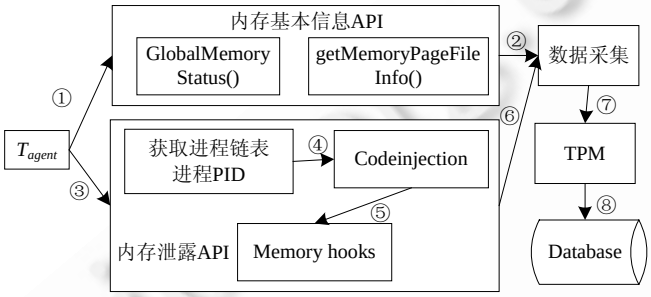


Fig.9 Architecture of sub-system memory monitor

图 9 内存监控子系统架构

算法 1. 内存信息收集.

① 收集代理服务端接到收集内存信息的命令马上启动内存信息收集模块,调用内存基本信息 API 收集内存的基本信息(如总内存大小、已使用空间等).

使用到的函数主要有 GlobalMemoryStatus 和 getMemoryPageFileInfo.

② 内存基本信息收集模块将收集到的数据全部交付给数据采集模块.

③ 检测内存是否存在内存泄露.首先从进程链表里面获取每个进程的 PID.

使用到的函数为 CreateToolhelp32Snapshot,Process32First 和 Process32Next.

④ 根据进程 PID 找到进程在内存中的地址,利用代码注入技术将收集内存泄露信息代码注入到内存.

⑤ 利用 memory hooks 监控内存的各项操作.

⑥ 进程内存泄露的相关信息交付给数据采集模块.

⑦ 数据模块将收集起来的全部数据交付给 TPM 加密、度量.

⑧ TPM 将已经加密、度量的数据存放到数据库里.

5.2 策略数据信息收集模块

Windows 平台上的策略数据通常是置于注册表中,因此,策略数据信息收集模块实际上是收集注册表的信息,其主要功能是收集操作注册表数据用户名、进程名、操作方式(读、写、删除)、操作时间等.在 Windows 平台中,注册表是整个系统的“枢纽”,一切活动在注册表里都有记录.入侵者一旦入侵系统,往往通过修改注册表来隐藏自己的入侵活动,清除自己的“入侵”痕迹.通过对注册表状态及操作信息的收集可以发现被修改的注册项,从而可以发现入侵的痕迹.

策略数据信息收集模块主要利用 Detours 来实现 API HOOK 的功能.Detours 是 Microsoft 开发的一个库,它可以拦截 x86 机器上的任意的 win32 API 函数.Detours 库可以拦截任意的 API 调用,它替换目标 API 最前面

的几条指令,使其无条件地跳转到用户提供的拦截函数,而被替换的 API 函数的前几条指令被保存到 trampoline 函数中,trampoline 保存了被替换的目标 API 的前几条指令和一个无条件转移,转移到目标 API 余下的指令.当执行到目标 API 时,直接跳到用户提供的拦截函数中执行,这时,拦截函数就可以执行自己的代码了.当然,拦截函数最后调用 trampoline 函数,trampoline 函数将调用被拦截的目标 API,目标 API 调用结束后又会返回到拦截函数.如图 10 所示.

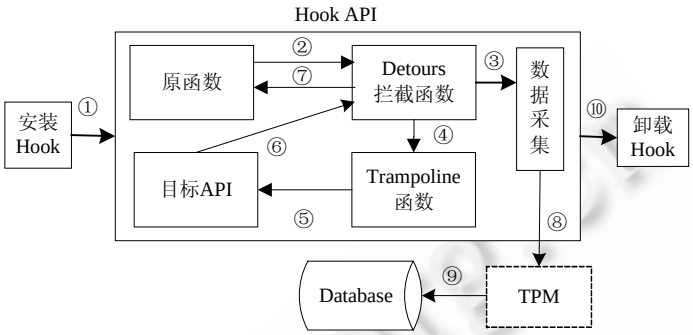


Fig.10 Architecture of the sub-systems of register monitor

图 10 注册表监控子系统架构

算法 2. 注册表状态及操作信息收集.

① 收集代理服务端接到收集策略数据信息的命令马上启动策略数据信息收集模块,通过调用 Windows 系统目录下 user32.DLL 中的 SetwindowsHookEx 函数来安装钩子.

② 将 Detours 挂接到源函数对注册表的操作处,将该 API 函数前几条指令替换成一个无条件转移指令,并且将被替换的指令和一条无条件转移指令存入 trampoline,当该 API 函数执行到那条无条件转移指令时就直接进入用户自定义的拦截函数.

③ Detours 拦截函数将该操作的相关数据(如注册表的操作类型、键名、键值类型、操作所属进程、进程的路径等)记录下来交给数据采集模块.

④ 调用 trampoline 函数将程序转移到目标 API.

⑤ 目标 API 继续完成未完成的 API 调用.

⑥ 控制权交给截获函数,而拦截函数将执行恰当的收尾工作.

⑦ 控制权返回到源函数调用处.

⑧ 数据采集模块将手机到的数据交给 TPM 加密、度量.

⑨ TPM 将已经加密、度量的数据存放到数据库里.

⑩ 如果用户不再想继续监控注册表的操作情况,就可以用 UnhookwindowsHookEx 函数将钩子卸载.

5.3 进程、CPU信息收集模块

有些木马程序在入侵系统时刻意隐藏自己的进程,使自己在系统进程管理器中不显示,从而使管理员很难发现.这时,我们可以从系统底层来读取进程相关对象的操作信息,使入侵活动的进程无法隐蔽.进程信息获取模块的功能是采集进程的可信证据,如进程名(CreateToolhelp32Snapshot()、Process32First()和 Process32Next())、用户名(GetUserName())、用户权限、文件类型(SHGetFileInfo())、文件位置(用 GetModuleFileName()得到应用程序的文件名再用_splitpath()分析文件名得到路径)、文件大小(FindFirstFile())、占用空间、创建时间、修改时间、访问时间(GetFileTime())、进程属性等;一次完整收集 CPU 使用率、进程基本信息、进程相关文件基本信息和操作信息的算法如图 11 所示.

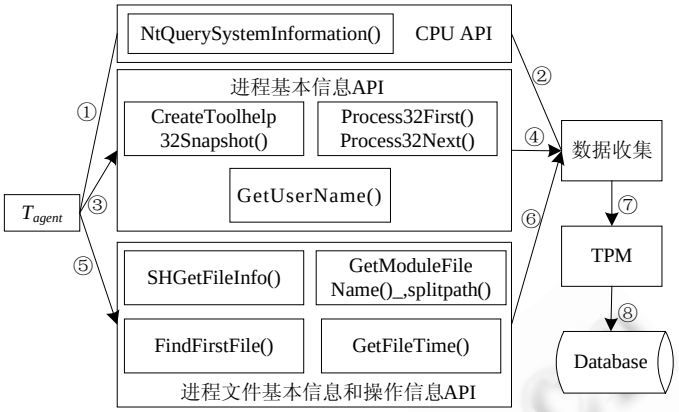


Fig.11 Architecture of sub-system process and CPU monitor

图 11 进程和 CPU 监控子系统架构

算法 3. CPU 使用率、进程基本信息收集.

① 收集代理服务端接到收集 CPU 信息的命令马上启动 CPU 监控模块,即利用 CPU API 获取 CPU 的使用率,主要使用了 NtQuerySystemInformation 函数.

② 得到 CPU 的使用率后数据统一由数据收集模块收集起来.

③ 收集代理服务端接到收集进程信息的命令后马上启动进程监控模块.利用进程基本 API 获取进程的基本信息,主要使用了 CreateToolhelp32Snapshot,Process32First,Process32Next 和 GetUserName 函数.

④ 当得到进程的基本信息以后,数据统一由数据收集模块收集起来.

⑤ 利用进程文件基本信息和操作信息 API 收集与进程相关的文件的基本信息和操作信息,主要使用的函数有 SHGetFileInfo,GetModuleFileName,_splitpath,FindFirstFile 和 GetFileTime.

⑥ 当得到进程的文件相关信息以后,数据统一由数据收集模块收集起来.

⑦ 数据模块将收集起来的数据全部交付给 TPM 加密、度量.

⑧ TPM 将已经加密、度量的数据存放到数据库里.

5.4 磁盘信息收集模块

入侵行为必定要修改文件,删除文件或增加文件,因此,我们可以通过获取文件的状态信息来判定是否有文件被修改,从而发现不可信证据.

磁盘信息获取模块的功能是采集磁盘的可信证据,如磁盘类型(GetDriveType(),判断是可移动磁盘、软盘、本地硬盘、网络磁盘、CD-ROM、RAM 磁盘)、磁盘文件系统类型(GetVolumeInformation())、磁盘容量、可用空间、已用空间、剩余空间(GetDiskFreeSpaceEx())、磁盘文件名、类型(SHGetFileInfo())、打开方式、路径、大小(FindFirstFile())、占用空间、创建时间、操作的类型(读、写、删除)、操作时间(ReadDirectoryChangesW(), GetFileTime())等.磁盘信息收集算法如图 12 所示.

算法 4. 磁盘信息收集.

① 收集代理服务端接收到收集磁盘基本信息收集命令后马上启动磁盘信息收集模块.利用基本信息收集 API 直接收集磁盘的基本信息,主要使用了 GetVolumeInformation,GetDiskFreeSpaceEx 和 GetDriveType 函数.

② 当得到磁盘基本信息以后,数据统一由数据收集模块收集起来.

③ 从客户端接收到具体收集可信证据的文件路径,通过磁文件盘基本信息和操作信息 API 获取该路径文件的基本信息和操作信息.主要使用的函数有 SHGetFileInfo,FindFirstFile,ReadDirectoryChangesW 和 GetFileTime.

④ 所获得的所有文件信息全部由数据采集模块收集.

⑤ 数据采集模块将收集到的所有数据交付给 TPM 加密、度量,并将已经加密和度量了的数据存入数据库.

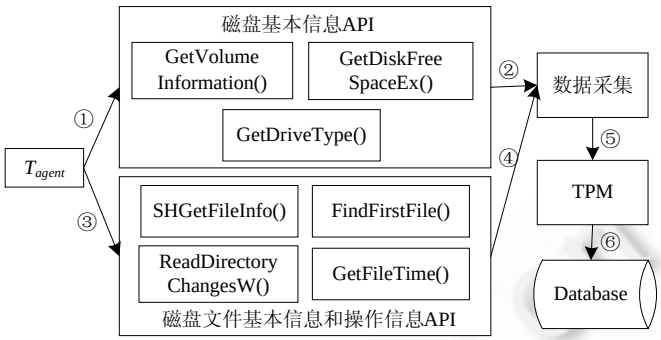


Fig.12 Architecture of sub-system of disk files monitor

图 12 磁盘文件监控子系统架构

5.5 网络端口信息收集模块

网络端口信息收集模块的功能是实时地收集流经网卡的数据包的信息,提取数据包的时间、大小、数据头、源地址、源端口、目的地址和目的端口等信息.之所以选择从网卡截获数据包,因为所有的网络数据必须流经网卡,入侵程序亦不例外.因此,我们可以使用现成的工具 winpcap 来从网卡上截获数据包,将获取到的数据包信息传送回给监控中心.winPcap 是由伯克利分组捕获库派生而来的分组捕获库,它是在 Windows 操作平台上来实现对底层包的截取过滤.winPcap 为用户级的数据包提供了 Windows 下的一个平台.winPcap 是 BPF 模型和 Libpcap 函数库在 Windows 平台下网络数据包捕获和网络状态分析的一种体系结构,这个体系结构是由一个核心的包过滤驱动程序、一个底层的动态连接库 Packet.dll 和一个高层的独立于系统的函数库 Libpcap 组成.

网络端口信息收集算法如图 13 所示.WinPcap 接口一共包含两个接口(wpcap.dll 和 packet.dll).它们为应用程序提供了一个底层的 API.使用 packet.dll 就可以调用 WinPcap 的相关函数.

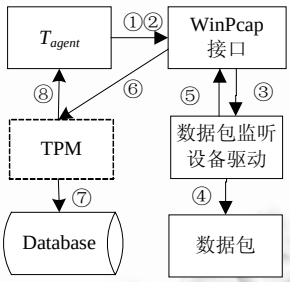


Fig.13 Architecture of sub-system of net port monitor

图 13 网络端口监控子系统架构

算法 5. 网络端口信息收集.

① 收集代理服务端接收到收集网络端口信息收集命令后马上启动网络端口信息收集.

② 获取到可用的网卡,对应到具体代码是 WinPcap 的 pcap_findalldevs 函数,如果没有找到网卡,则整个数据包监控结束.

③ 如果找到网卡就打开网卡接口,对应到具体代码就是 pcap_open_live 函数.

④ packet.dll 这个接口用来将应用程序和数据包监听设备驱动隔离开来,并且启动数据包监听设备驱动.具体代码是调用 pcap_setfilter 函数把一个过滤器与核心驱动抓包会话关联起来,相关的过滤器将被应

用到所有的来自网络的数据包上.

⑤ 设定监听端口,启动数据包监听驱动后就开始抓取流经网卡相关端口的数据包.

⑥ 通过 WinPcap 接口调用 WinPcap 的 pacp_loop 函数接收数据包,将信息传送给回调函数 packet_handler.

⑦ packet_handler 回调函数将数据包信息传给数据采集模块.

⑧ 数据采集模块将收集到的所有数据交付给 TPM 加密、度量.

⑨ TPM 将加密、度量好的数据存入数据库.

6 可信证据收集机制的应用案例研究

本节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,并由此评估终端的可信性.在该实验环境中,我们选用两台计算机:一台为普通 PC,基本配置为 Windows vista home basic @2007 service pack 1,CPU:Intel(R)Core(TM)2 Duo CPU T6670 @2.20GHz,1.0GB RAM,用于部署可信证据收集代理服务端程序;另一台为服务器,基本配置为 Windows XP Professional 2002 Service pack 3,CPU: Intel(R)Core(TM)2 Duo CPU E7500@2.93GHz,2.0GB RAM,用于部署可信证据收集代理客户端程序.普通 PC 终端在访问服务器上的某种服务之前,服务器需要通过可信证据收集代理服务端程序判断该终端的动态运行环境是否可信.值得注意的是,由于终端和服务器处于各自独立的信任域内,终端所有者可以任意修改运行在平台中的程序及其状态,因此,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.本文提出的可信终端动态运行环境的可信证据收集机制可以应用于此类开放的分布式计算环境中.

6.1 终端运行环境可信证据收集

通过运行在服务器上的可信证据收集代理客户端程序向运行在终端的可信证据收集代理服务端程序发出收集相关可信证据指令,即可获得终端的相关可信证据.下面分别分类列出该终端收集的动态运行环境可信证据(注:通过该代理获得的原始可信证据是密文).

(1) 进程信息获取模块收集到的可信证据如图 14 所示.第 1 列为进程名,第 2 列为用户名,第 3 列为用户的权限(增、删、改和查),第 4 列为文件类型,第 5 列为文件位置,第 6 列为文件大小,第 7 列为占用空间,第 8 列为创建时间,第 9 列为修改时间,第 10 列为访问时间,第 11 列为进程的属性.

P_Name	Use_Name	Permission	File_type	File_path	File_size	Space	Found_time	Alter_time	Visit_time	attribute
d6c058525cde0f4	523320de68bf7a912	44b42340cd3ebe332bf3	338dec6c4d37a4849b4272	f3375c1d8223b26a46e12b2155	ed1e6d2e7dd2e	a563a44eaad9d	e698ed48705139c5	30884f4021a379e5	928cafc666d8cb8	3b819df1dfd9958788b8f913
f14db7d699382ba2	85db1caf2361fb16b	44b42340cd3ebe3353c2	44b42340cd3ebe332e4021	d0bc9ff658a81f75f599277f089	6f979930ee5fa	d9431d8c95db5	7a12412c8d4d4e80	af941d293711d3a9	d9efa8e8dfe947c	695f662288b5daa9cdf5f814
f3557e512bd4b074	90194f84682a3cd5f	D3353c21affd5753efbe	44b42340cd3ebe332e4021	81e7e976cf8ff33c5cbc996ed4	312b6e82d51d7	50582e3986e05	7db385e1bb2ead0e	2cb6f1ac68c697a2	fca4c36ddbb4490	247da1923750587361 5d311c

Fig.14 Trust evidence of process

图 14 进程可信证据

(2) 内存信息获取模块收集到的可信证据图 15 所示.第 1 列为内存的总大小,第 2 列为已使用空间,第 3 列为当前空闲量,第 4 列为缓存大小,第 5 列为交换区大小,第 6 列为内存泄露的总量,第 7 列为内存泄露的进程名,第 8 列为进程内存泄露的大小.

Memory_size	Used_space	Free_size	Cache_size	swaparea_size	reveal_size	P_name	P_reveal_size
7943d115e484ab	5da17735f5089d7e59	44b42340cd3ebe3353c	c5ff7e463f80d2612	84d9bec5b9674c6685	6f184194d535f746	3f55c4fecfbe41bb	e2c782a5ba33997
f94285a9a6eed3c5	67a1f4b41265ac0cd3	ebb42340cd3ebe332e4	c435b02534257c2d4	2de653d35073eb896c	b7c376e9a373f2	4da5b0fc1812645c42	e2c782a5ba33997
c6d639fd841fb5b	1a62797f87ddebf538	be33b42340cd3ebe332	ff62bff2e11498bfc	2bfae9f2ceb0a61c4c	cf7e102d4c90e58	ecea8692be7e67d17e	bb844b1191faf146

Fig.15 Trust evidence of memory

图 15 内存可信证据

(3) CPU 状态信息获取模块收集到的可信证据如图 16 所示,这一列表示 CPU 现在的使用率.

Use_rate
493299df249b2c3e
60281f4b79d19c3
b4dfa432a2cb53b6

Fig.16 Trust evidence of CPU

图 16 CPU 可信证据

(4) 网络端口信息获取模块收集到的可信证据如图 17 所示.第 1 列为接收数据包的进程名,第 2 列为收发时间,第 3 列为数据包大小,第 4 列为数据包源地址,第 5 列为数据包源端口,第 6 列为数据包目的地址,第 7 列为目的端口.

P_name	time	Packet_size	og_address	og_port	pu_address	pu_port
f9b0116eb110a718e74ab	d0b7ee7281c789b0	b54c36ccfb6ea473	7820af387446d7aa19d8d3c5997e76	316971eb3f15ba34	7820af387446d7cf99e665de9bb29	316971eb3f15ba34
4da5b0fc1812645c42c31	d0b7ee7281c789b0	4db9b22b2cb2fe3	7820af387446d7aa19d8d3c5997e76	ad659b299c316ea	2861ca949562c1579044727e1a2c1	ad659b299c316ea
eb1116e5f623824319f0c	5fb3ead9445f6235	9b6be19ea98635df	7820af387446d7aa19d8d3c5997e76	16378c63d4f8b1c	58362152c920471e98ec7a6e8d2fa	16378c63d4f8b1c

Fig.17 Trust evidence of net port

图 17 网络端口可信证据

(5) 磁盘信息获取模块收集到的可信证据如图 18 所示.第 1 列为磁盘类型,第 2 列为磁盘文件系统类型,第 3 列为磁盘容量,第 4 列为可用空间,第 5 列为已用空间,第 6 列为剩余空间,第 7 列为文件名,第 8 列为文件类型,第 9 列为打开方式,第 10 列为文件路径,第 11 列为文件大小,第 12 列为文件占用空间,第 13 列为创建时间,第 14 列为操作的类型,第 15 列为操作时间.

Used_space	surplus	File_name	File_type	Openway	File_path	File_size	Space	Create_time	Operate_type	Operate_time
7ba72ef5c91de31e	32e737a0e55454	8fc59c7e3614be74a	a213231306611651	5862301c3bf8bc	44b42340cd3ebe332bf32f863fa2	fd3be5bdbc57633	f5cbv91de31e13838	0e93ee8632aff2739b	5da17735f5089d7	56e9d09861211024 5c4117
1bde578ff4f128ab	be791599baafa8(	070ca8fc59c7e3614be	5e3dfbad54aaa21	165973e77c1f90	905d1fdf443d9152b62f80f43f9d	4e4d9fa03bd4936	064dac7614cd0ec44	9b775a5d0739845694	1a62797f87ddebf5	9cce929bec667559cbbf79
cc5e9b924ccd4064	4ac9c6211beeeb1	32de070ca8fc59c7e36	f3ea1b5e3dfbad5	c3912c749692d7	a2caa1c243a2342822564bcfd7e0	da9b16fec656a5	28abf9913503c7456	529a5c17d1bc43cc79	67a1f4b41265ac67	9cce92f5ecbd459cbbf797

Fig.18 Trust evidence of disk files

图 18 磁盘文件可信证据

(6) 策略数据信息获取模块收集到的可信证据如图 19 所示.第 1 列为用户名,第 2 列为进程名,第 3 列为操作方式(读、写、删除),第 4 列为操作时间.

User_name	p_name	operate_way	operate_time
44b42340cd3ebe332e40215275	8c3063810831fc33c095dac5eb1bea917e	3fc3bbfa42d755bdc4bbbb9ecaab80f3cba	35e583b4772d311a263
44b42340cd3ebe332e40215275	8c3063810831fc33c095dac5eb1bea917e	387d3fdc80f914f6277a3515d46c254eed8	a5cc941783afa41cef3
44b42340cd3ebe332e40215275	8c3063810831fc33c095dac5eb1bea917e	3fc3bbfa42d755bde3e9399fa4c344a9e6b	ac549fb2180383ebb0b

Fig.19 Trust evidence of register

图 19 注册表可信证据

6.2 终端的可信性评估

终端动态运行环境的可信证据记录终端所有重要软硬件的可信证据的状态、操作、时间等信息.利用这些信息可以分析与对象状态、操作相关的可信属性,例如性能、安全性、可靠性、可用性等.在安全性方面,可以通过分析 E 中各个对象的状态和操作序列来确定终端运行环境是否被攻击过或者被篡改过.因此,利用可信证据收集代理可以获取到整个终端动态运行环境的可信证据,通过对这些原始信息的分析,可以对终端的可信性进行评估.对于如何评估、采用什么样的评估模型等,我们将另文加以讨论.

6.3 可信证据收集代理在终端中的性能分析

可信证据收集代理服务端程序在一定程度上会影响终端的性能,我们将从启动时间、CPU 负载、内存消耗等方面对此进行详细讨论.而可信证据收集代理客户端程序功能简单,对运行环境的影响很小,这里不再讨论.

图 20 所示为终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,代理文件的大小为 1.45MB,这里的启动时间是指终端的可信证据收集代理服务端程序开始执行到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 次启动代理,这 4 次执行的时间如图 20 所示,其中,横坐标为启动次数,纵坐标为执行时间(单位:s).实验是在如下的环境中进行的:Windows XP Professional 2002 Service pack 3,CPU:Intel(R)Core?2 Duo CPU E7500@2.93GHz,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,代理程序第 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代理文件的度量.从第 2 次启动开始,由于代理文件没有发生改变,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.这个结果与文献[17]相似.

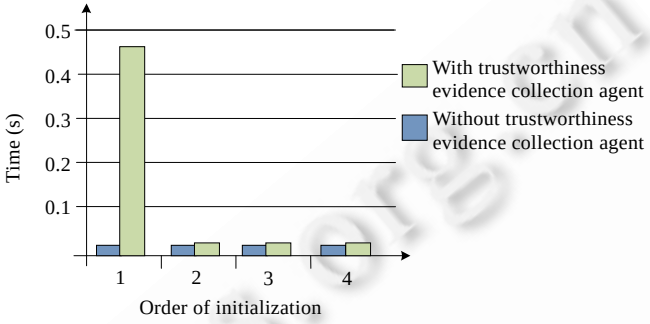


Fig.20 Start time of the trusted terminal

图 20 可信终端启动时间

为了测试终端 CPU 负载,我们首先关掉终端所有无关的应用进程,待其稳定后测试 CPU 的利用率,如图 21 所示,图中显示的是在 4 个不同的时间点 CPU 的利用率(间隔 5 分钟).从该图可以看出,当关闭所有应用程序以后,CPU 的利用率几乎为 0.

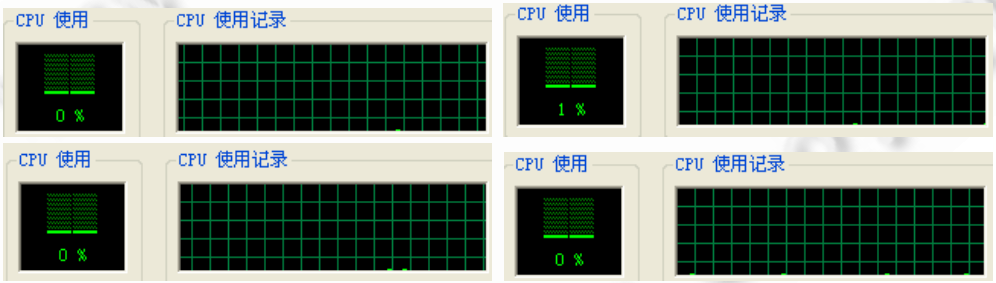


Fig.21 CPU performance of terminal when server of the trusted evidence collection agent shuts down

图 21 未开启可信证据收集代理服务端时终端 CPU 的性能

然后启动监控代理,待其稳定后再测试 CUP 的利用率.如图 22 所示,图中显示的是在 4 个不同的时间点 CPU 的利用率(间隔 5 分钟).从该图可以看出,启动监控代理后 CPU 的利用率变动范围在 2%~15%,常在 5%~12%之间波动.也就是说,收集代理给 CPU 带来低于 15%的开销.

其间,我们在可信证据收集代理稳定运行期间,在 4 个不同时间点启动了一常用应用软件——浏览器 IE(间隔 5 分钟),如图 23 所示.该图将可信证据收集代理引起的 CPU 负载与 IE 引起的 CPU 负载进行比较.从图中可以看出,启动一个网页 CPU 的负载峰值分别为 42%,47%,50%,49%;多次测试发现,其变化范围为 40%~55%.因此,收集代理带来的 CPU 负载远远低于其他常用的应用软件.

为了测试内存消耗,我们首先关掉终端所有无关的应用进程,测试内存占用率,如图 24(a)所示;然后启动收集代理,再次测试内存占用率,如图 24(b)所示.从前后两个图可以直观地看出,收集代理带来的内存开销为 3M,

非常小,对终端内存的使用几乎没有影响.

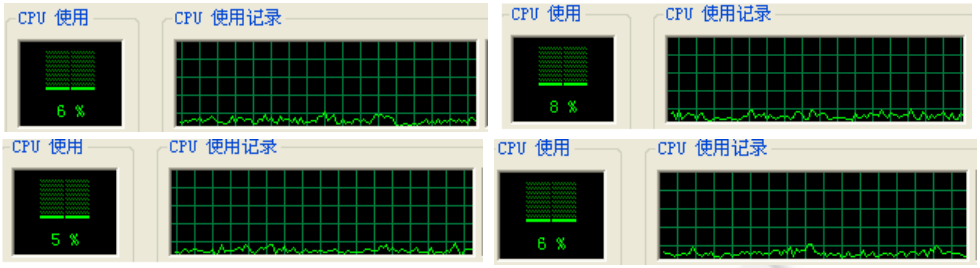


Fig.22 CPU performance of terminal when server of the trusted evidence collection agent runs

图 22 开启可信证据收集代理服务端时终端 CPU 的性能

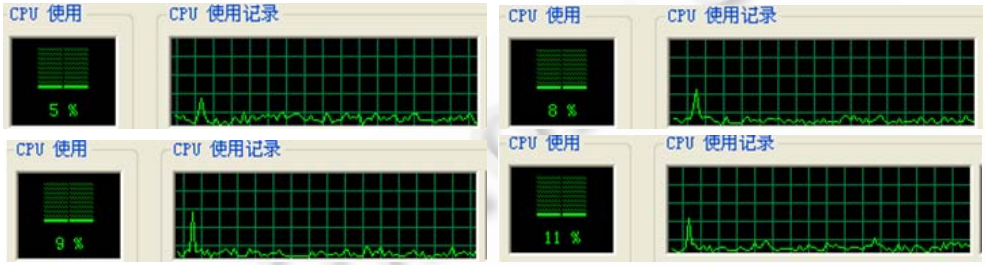


Fig.23 CPU performance of terminal when IE runs

图 23 开启 IE 浏览器时终端 CPU 的性能

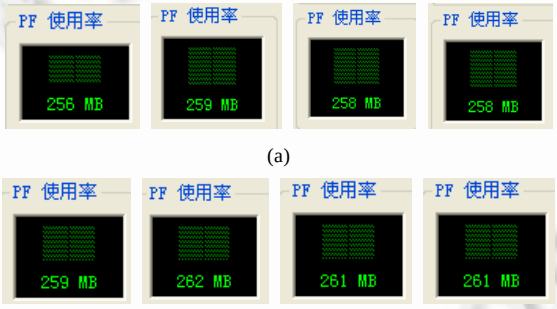


Fig.24 Compare of memory expense of terminal when server of the trusted evidence collection agent runs and shuts down

图 24 开启和关闭可信证据收集代理服务端时终端内存开销对比

7 讨 论

本文中提出的终端可信证据收集代理具有较强的可扩展性.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[23],George's model[24]以及 Sun's model[25]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代理.

本文提到的终端可信证据收集代理与当前安全领域的研究热点——入侵检测的第 1 步工作有些相似之

处,也与许多主机监控系统,甚至杀毒软件等也有些相似之处.但终端可信证据收集代理利用 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.但是,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.

8 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代理.该可信证据收集代理能够为其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代理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.

为了支持不同的可信性评估模型,下一步的工作是引入终端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.终端运行时的可信证据收集应贯穿于终端的整个运行过程中,我们下一步的工作也包括通过这些可信证据如何评估终端动态环境的可信性.

References:

[1] Trusted Computing Group. TPM main specification. Main Specification Version 1.2 rev. 85. Trusted Computing Group, 2010.

[2] 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 with English abstract).

[3] Jaeger T, Sailer R, Shankar U. PRIMA: Policy-reduced integrity measurement architecture. In: Proc. of the SACMAT 2006. 2006. 19−28. [doi: 10.1145/1133058.1133063]

[4] 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. San Diego, 2004.

[5] Li XY, Zuo XD, Shen CX. System behavior based trustworthiness attestation for computing platform. Acta Electronica Sinica, 2007,35(7):1234−1239 (in Chinese with English abstract).

[6] CNCERT/CC. CNCERT/CC2005-2010 Report. 2010 (in Chinese). http://www.cncert.org.cn/upload/2005-2010CNCERTCCAnnual Report_Chinese.pdf

[7] Kim GH, Spafford EH. The design and implementation of tripwire: A file system integrity checker. In: Proc. of the 2nd ACM Conf. on Computer and Communication Security. Fairfax, 1994. 18−29. [doi: 10.1145/191177.191183]

[8] Jaeger T, Sailer R, Shankar U. PRIMA: Policy-reduced integrity measurement architecture. In: Proc. of the 11th ACM Symp. on Access Control Models and Technologies. Lake Tahoe: ACM Press, 2006. 19−28. [doi: 10.1145/1133058.1133063]

[9] Litty L, Lagar-Cavilla HA, Lie D. Hypervisor support for identifying covertly executing binaries. In: Proc. of the 17th USENIX Security Symp. Berkeley, 2008. 243−258.

[10] Tygar JD, Yee B. Dyad: A system for using physically secure coprocessors. Technical Report, CMU-CS-91-140R, Carnegie Mellon University, 1991.

[11] Clark PC, Hoffman LJ. BITS: A smartcard protected operating system. Communications of the ACM, 1994,37(11):66−70. [doi: 10.1145/188280.188371]

[12] Arbaugh WA, Farber DJ, Smith JM. A secure and reliable bootstrap architecture. In: Proc. of the 1997 IEEE Symp. on Security and Privacy (S&P'97). Oakland, 1997. 65−71.

[13] Dyer JG, Lindemann M, Perez R, Sailer R, van Doorn L, Smith SW. Building the IBM 4758 secure coprocessor. IEEE Computer, 2001,34(10):57−66. [doi: 10.1109/2.955100]

[14] Maruyama H, Seliger F, Nagaratnam N, Ebringer T, Munetoh S, Yoshihama S, Nakamura T. Trusted platform on demand. Technical Report, RT0564, IBM, 2004.

[15] 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. San Diego, 2004. 223−238.

[16] Li X, Shi WC, Liang ZH, Liang B, Shan ZY. Design of an architecture for process runtime integrity measurement. Microelectronics and Computer, 2009,26(9):183−186 (in Chinese with English abstract).

[17] 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 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3789.htm [doi: 10.3724/SP.J.1001.2010.03789]

[18] Desai AK, Bendar SA. Automatic remote computer monitoring system. In: Proc. of the US Patent 5. 2000. 703−781.

[19] Kennell R, Jamieson LH. Establishing the genuinity of remote computer systems. In: Proc. of the 11th USENIX Security Symp. USENIX, 2003. 295−308.

[20] Wang XC, Liu YN. Hierarchical network security monitor system based on agents. Computer Engineering, 2007,33(24):172−174 (in Chinese with English abstract).

[21] Sean WS. Trusted Computing Platforms: Designand Applications. New York: Springer-Verlag, 2005.

[22] 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. [doi: 10.1145/1165389.945462]

[23] Yoshizawa S, Baba A, Matsunami K, Mera Y, Yamada M, Shikano K. Unsupervised speaker adaptation based on sufficient HMM statistics of selected speakers. In: Proc. of the Eurospeech 2001, Vol.2. 2001. 1219−1222.

[24] Cohen CI, Magai C, Yaffee R, Walcott-Brown L. Racial differences in syndromal and subsyndromal depression in an older urban population. Psychiatric Services, 2005,56(12):1556−1563. [doi: 10.1176/appi.ps.56.12.1556]

[25] See S. Sun's grid model. In: Proc. of the 1st IEEE Int'l Workshop on Grid Economics and Business Models (GECON 2004), Vol.23. 2004. 173−184.

附中文参考文献:

[2] 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.

[5] 李晓勇,左晓栋,沈昌祥.基于系统行为的计算平台可信证明.电子学报,2007,35(7):1234−1239.

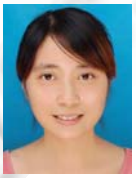
[6] CNCERT/CC. CNCERT/CC2005-2010 年网络安全工作报告. http://www.cncert.org.cn/upload/2005-2010CNCE RTCCAnnual Report_Chinese.pdf

[16] 李霄,石文昌,梁朝晖,梁彬,单智勇.进程运行时完整性度量的体系结构设计.微电子学和计算机,2009,26(9):183−186.

[17] 古亮,郭耀,王华,邹艳珍,谢冰,邵维忠.基于 TPM 的运行时软件可信证据收集机制.软件学报,2010,21(2):373−387. http://www.jos.org.cn/1000-9825/3789.htm [doi: 10.3724/SP.J.1001.2010.03789]

[20] 王新昌,刘育楠.基于 Agents 的层次型网络安全监控系统.计算机工程,2007,33(24):172−174.

谭良(1972－),男,四川泸县人,博士,教授,主要研究领域为可信计算,网络计算.



陈菊(1987－),女,硕士生,主要研究领域为可信计算.