ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.7, July 2007, pp.1715−1729
DOI: 10.1360/jos181715

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

基于安全操作系统的电子证据获取与存储*

丁丽萍[1,2,3+], 周博文[1,3], 王永吉[1,4]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](北京警察学院,北京 102202)

[3](中国科学院 研究生院,北京 100049)

[4](中国科学院 软件研究所 计算机科学重点实验室,北京 100080)

Capture and Storage of Digital Evidence Based on Security Operating System

DING Li-Ping[1,2,3+], ZHOU Bo-Wen[1,3], WANG Yong-Ji[1,4]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Beijing Police College, Beijing 102202, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[4](State Key Laboratory for Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62565272, Fax: +86-10-82620803, E-mail: dingliping@itechs.iscas.ac.cn

Ding LP, Zhou BW, Wang YJ. Capture and storage of digital evidence based on security operating system. *Journal of Software*, 2007,18(7):1715−1729. http://www.jos.org.cn/1000-9825/18/1715.htm

Abstract: In this paper, a kind of security operating system with the mechanism of real-time forensics (called SeFOS) is presented, the general architecture of SeFOS is described, the model of its forensics behaviors is analyzed with some formal method descriptions, and the method of completely collecting and safely storing for the digital evidences is presented. The forensics model of SeFOS is inside the kernel and the evidences are obtainted from system processes, system calls, resources assigning inside the kernel and network data. Finally, a simulated experiment is designed to validate the efficiency of SeFOS.

Key words: post-mordem forensics; real-time forensics; operating system; forensics behavior model; data collection; security and protection

摘 要: 基于实时取证的思想,提出了一种安全可取证操作系统(security forensics operating system,简称 SeFOS)的概念和实现思路.提出了其总体结构,建立了该系统的取证行为模型,对其取证服务和取证机制进行了分析并作了有关形式化描述,阐述了证据数据的采集和安全保护方法,提出把取证机制置于内核,基于进程、系统调用、内核资源分配和网络数据等获取证据的方法,并通过模拟实验验证了 SeFOS 的可取证性.可取证操作系统的研究对于进一步

* Supported by the National Natural Science Foundation of China under Grant No.60373053 (国家自然科学基金); the State Education Ministry's Scientific Research Foundation for the Returned Overseas Chinese Scholars under Grant No.[2003]406 (国家教育部留学回国人员科研启动基金); the Research Collaboration Between the Chinese Academy of Sciences and the Royal Society of the United Kingdom under Grant Nos.20030389, 20032006 (中国科学院与英国皇家学会联合资助项目); the Plan of Hundreds Scientists in the Chinese Academy of Sciences (中国科学院百人计划)

Received 2006-02-24; Accepted 2006-06-07

研究可取证数据库管理系统(forensic database management system,简称 FDBMS)和可取证网络系统(forensic network,简称 FNetWork)具有重要意义.

关键词: 事后取证;实时取证;操作系统;取证行为模型;数据采集;安全保护
中图法分类号: TP309 文献标识码: A

事后取证是目前普遍采用的取证方式,即在系统被攻击或者犯罪行为发生后,取证调查人员对可疑计算机开展诸如恢复数据、获取数据、分析鉴定等调查工作,收集所有可能获取的数据来进行事件重构,以确认犯罪行为实施的时间、地点和方式[1−5].然而,犯罪分子在作案后可能会彻底删除或者混淆证据来隐藏他们的行为;而且系统中的某些事件,如正在进行的文件的修改、进程的中断、内部进程通信和内存的使用情况等,或许不会在被攻击的系统中留下事后证据.所以,需要一个运行中的监控器来把这些事件实时记录下来[1]以获取全面而充分的证据,支持调查人员得出具有较强的确定性的结论.研究表明,对于操作系统的进程等实时数据的取证越来越受到重视.例如,2005 年在美国召开的国际电子证据取证研讨会(digital forensics research workshop,简称 DFRW 2005)上提出的挑战(challenge)就要求获取隐藏的进程(hidden processes)作为证据[2],还要求弄清楚进程的隐藏方法.虽然 Linux 系统提供了 *who*,*w*,*ps* 和 *top* 等了解进程的运行状态以及存活情况的命令以用来获取可疑进程,但是,这些命令运行在用户态,入侵系统的黑客可以轻松地找到这些进程监控程序的磁盘映像进行删除甚至替换.依赖于这些命令进行证据的获取是不可靠的.为此,本文提出了一种内核级取证系统,该系统采用计算机取证的多维过程模型[3],根据不同的取证需求定制取证策略,针对系统调用、进程、文件、内核资源分配和网络数据等数据源进行证据获取,保证了实时证据和事后证据的全面获取.同时,该系统基于安全操作系统设计,因而称为安全可取证操作系统(security forensics operating system,简称 SeFOS).

本文第 1 节介绍相关研究.第 2 节建立 SeFOS 的取证行为模型,并对其进行分析.第 3 节论述 SeFOS 的数据采集.第 4 节说明证据数据的安全保护.第 5 节是模拟实验验证.最后是结论.

1 相关研究

传统的电子证据数据源是网络日志和磁盘状态[4,5].例如,Snort 可以记录网络传输日志[6],TCT(the coroner's toolkit)可以恢复删除的文件并提供和文件产生、修改或者访问的时间有关的信息.其他事后证据获取工具有 Encase,FTK,Ilook, SMART 等.这些取证工具共同的缺陷是没有记录系统运行过程中的潜在证据——日志,仅考虑了事后取证.

Backtracker[7]构造了一个图形界面,通过在操作系统调用级记录运行的事件和系统对象向管理员展示了潜在的会引起系统被攻击的监测点的事件序列.事件通过对于诸如读、写、执行、fork 等系统调用的监测被标识.系统对象就是进程、文件和文件名.然而,它不能监测在内存映射对象中的操作.而且,该模块受管理员控制,如果进攻者取得了管理员身份,则很容易停掉该模块.网络日志仅仅进行加密是不够的,因为密钥是可以获取的.磁盘镜像仅仅提供了文件系统的最后状态而不能提供进攻过程的信息.SNARE(system intrusion analysis & reporting environment)[8]是一个入侵检测和分析环境,它的目的在于通过一个运行在 Linux 操作系统的可动态加载的内核守候进程解决这一问题.它能够提供给取证调查分析人员有关系统事件日志的信息,例如网络连接、文件和目录的读写、用户和组标识的修改以及应用程序进程的改变.SNARE 的功能包括通过系统调用级可加载模块的执行捕获事件、由定义了事件类型和系统可审计对象的守候进程进行审计和位于高层的取证分析 GUI(graphic user interface)表达层来浏览捕获的系统事件.它不提供任何对于事件序列重构的自动取证分析.把分析任务留给对高一层的系统事件进行观察的调查员.它把文件看作原子实体.共享内存访问不被审计.同时,它没有提供取证过程的监督和取证模块的保护机制.Forensix 提供了取证分析的准确性,同时减少了手工操作[9].Forenix 基于 SNARE 并在 3 个关键点上对其进行了扩展:在系统调用级监测目标系统的执行;提供了一个对所有行为的可独立应用的观察;当系统调用级的信息通过一个私有界面存储的时候,提供了一个安全的支持系统和一个数据库技术来支持对于获取的日志的高层查询.Forensix 使用了与 Backertracker 同样的系统调用和

对象日志技术.它通过调用参数和返回值扩大了潜在证据的收集.他们的主要不同在于分析的步骤上:Forensix依赖于数据库技术来处理随意的查询和从探测点到攻击源跟踪事件序列,并且它也把一个文件对象作为一个原子实体而不进一步细化,它用一种计时的方法来解决内存访问问题但仍然不能把内存装载和存储操作日志记录下来.虚拟机技术也被用于取证分析并提供了一个时间重构的框架.一个虚拟机监控器是一个模拟硬件的软件执行.运行在虚拟机中的操作系统被称为客户操作系统,以区分运行在裸机上的宿主操作系统.ReVirt 是执行在宿主操作系统中的一个入侵检测和分析系统[10],它可以记录每一个在虚拟机中的指令执行的日志,允许一条指令一条指令地回放虚拟机的操作过程.它能够在可能被非决策影响的时候重建事件.尽管 ReVirt 提供了一个系统运行的全部日志记录,重放虚拟机也不会对那些导致系统问题的时间序列的重构进行任何分析.日志和重放机制试图提供一个重建潜在进攻的功能,并把分析留给调查者.Livewire 最初是作为一个 IDS(intrusion detection system)来构思的[11],但提供了一个实时事件的日志能力来进行取证分析.在一个 VMM(virtual machine monitor)中,它在客户操作系统中监控特别定义的事件.一个更新的研究工作在 Backtracker 中加入了对有文件对象参加的事件的细粒度取证.在文件系统中的读写操作也在系统调用级被监控,使用了偏移量参数来确定一个进程是否读了一个以前曾经被另一个进程修改过的文件的一部分.结果是减少了搜索空间、搜索时间和错误依赖.文献[1]基于对上述问题的研究,提出了一种对共享内存映射文件(shared memory-mapped files)进行监控的系统,试图以内存映射文件为数据源,在系统调用级实时获取证据.但是,对于取证模块的按需部署、安全保护等问题,文献[1]中并没有提及.

2 SeFOS 的取证分析

2.1 SeFOS的行为表示模型

根据文献[12]提出的一般意义上的操作系统行为表示模型,本文给出了 SeFOS 的取证行为表示模型.如图 1 所示.

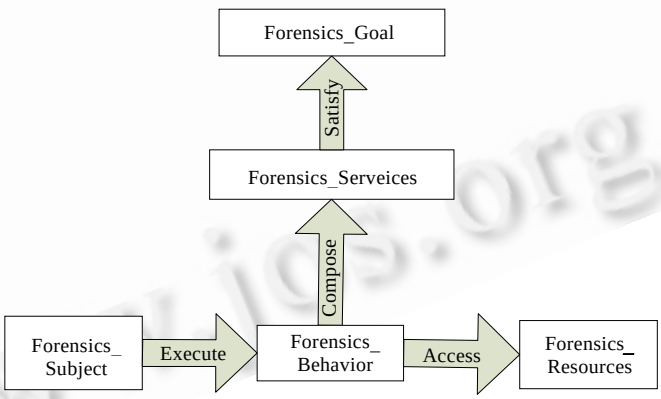

Fig.1 Behavior model of SeFOS

图 1 SeFOS 的取证行为模型示意图

定义 **1**(取证行为模型). 定义如下五元组表示取证行为模型:

(*Forensics_Goal*,*Forensics_Subject*,*Forensics_Behavior*,*Forensics_Resources*,*Forensics_Serveices*).

其中,*Forensics_Goal* 是取证的目的,表示可取证操作系统中的取证模块设计的目的,指用这样的模块作取证的行为的目的;*Forensics_Subject* 是取证的主体,表示取证过程中的行为主体,例如,取证人员作为系统的一个特殊用户(有别于系统管理员的用户)可以是取证过程的主体,取证进程是系统地址空间中的主体,取证主机是分布式系统中的主体等等;*Forensics_Behavior* 是取证的行为(也称为活动),表示主体在系统运行中的活动,如取证人员的登陆、证据源数据的获取、取证模块的配置等;*Forensics_Resources* 是取证的资源,表示系统中用于取证的

资源,例如系统调用、进程、内核资源分配等;*Forensics_Serveices* 是取证的服务,表示一组目的在于获取证据的活动.

五元素之间的交互关系如图 1 所示.纵向来看,自上而下,取证目的由若干取证服务实现,取证服务由一组取证活动组成.横向来看,自左至右,系统的取证主体通过一系列的活动耗费或者创造资源.由此得出如下规则:

规则 **1**. 设 $fb_j(j=1,2,3,\ldots,n)$表示某一特定的取证活动,fs 为某取证服务,FB 为全部取证活动的集合,则$\forall fs$,$\exists fb_1,fb_2,fb_3,\ldots,fb_n,fb_j\in FB$,满足$(fb_1\wedge fb_2\wedge fb_3\wedge\ldots\wedge fb_n)\rightarrow fs$.

即每一个取证服务都要由一个取证行为序列完成.

规则 **2**. 设 $fs_j(j=1,2,3,\ldots,n)$代表一系列特定的取证服务,fg 表示某一个取证的目的,则$\forall fg,\exists fs_1,fs_2,fs_3,\ldots,fs_n$,满足$(fs_1\vee fs_2\vee fs_3\vee\ldots\vee fs_n)\rightarrow fg$.

即每一个取证目的由一个或多个取证服务达到.

规则 **3**. 设 $f_sub_j\rightarrow$表示取证主体,fr_j 表示取证资源,$fr_j>0$ 表示产生的资源,$fr_j<0$ 表示消耗的资源$(j=1,2,3,\ldots,n)$,则$(f_sub_i=f_sub_j)\wedge(fb_i=fb_j)\rightarrow fr_i=fr_j$.

即同样的主体通过相同的取证活动将产生或消耗同样的系统取证资源.

根据以上规则,可以对具体模型的正确性进行形式化验证.

2.2 SeFOS的取证服务

取证的目的决定了 SeFOS 应具有的服务.从司法的角度看,获取的证据应该满足证据的可采用性标准,即证据的客观性标准、关联性标准和合法性标准[13].由于计算机取证的研究必须受本国法律法规的制约[14],SeFOS 的取证模块应该提供的服务包括:

- 取证策略的按需定制.采用多模块支持多种需求的方式,按照各种不同的需求设计出不同的取证模块,用户可以根据自己的需求选择所需要的模块.各个取证模块可以根据取证人员的配置,动态地改变自己的状态(enable 或者 disable).具体地,把用户的取证需求转化成新案件的格式,对新的案件以案例库为样本空间进行归类,根据以往发生过的同类案件所采用的取证策略,组合生成新案件的取证策略,并用该策略配置系统的取证功能.这一服务保证了证据的关联性.
- 证据文件和取证进程的安全保护.对证据文件和取证进程的安全保护是为了防止原始证据被非法访问、伪造或变造.SeFOS 以我国法律法规为依据,取证模块提供的取证服务通过访问控制、取证进程保护和证据文件保护提供主体合法、形式合法的证据.
- 全面的证据数据源.SeFOS 的取证模块以进程、系统调用、文件、内核资源分配和网络数据等作为取证的数据源,能够实时地获取全面的证据,为所获取的证据的关联性提供了保证.
- 遵循多维取证模型[3].多维取证过程模型是按照法律、法规的规定程序设计的合理的取证过程模型.按照该模型获取的证据保证了电子证据的形式、程序和提取的方法合法化,同时也具备了客观性.

2.3 SeFOS的取证机制

SeFOS 的取证模块位于 Linux 内核,是一个动态加载模块.这样的设计主要是为了保证取证进程的安全性.

- SeFOS 的结构

SeFOS 的总体结构如图 2 所示.在该结构中,根据具体的取证需求,通过案例库和 *K*NN(*K*-nearest neighbor)分类器确定该需求所归属的案件类型.然后,根据同类案件的取证策略组合确定新的取证策略,取证模块选择器则根据取证策略确定所需要的取证模块,取证模块组合器对于选中的取证模块进行组合,内核中的取证模型则利用组合好的取证模块根据多维取证过程模型[3]进行证据的获取,已经获取的证据及其形成的证据监督链数据被存入数据库,以便于进一步的分析、鉴定和出示.

- 取证主体

在安全操作系统中,进程被认为是唯一的“主体”(即只有进程是活动的实体)[12],进程可以访问文件系统客体(file system object,简称 FSO)、进程间通信(inter-process communication,简称 IPC)客体和网络端口(socket)等.

这里,取证主体就是处于活动状态的取证进程.在 Linux 中,进程的属性如下:

pid_t pid:进程 ID

char comm.[16]:进程名

volatile long state:进程状态

pid_t pgrp:进程组创建进程 ID

unsigned long start_time:开始时间

struct files_struct *files:打开文件列表,用于获取进程打开了哪些文件、管道和网络端口

struct linux_binfmt *binfmt:用于获取进程执行时的参数信息和环境变量

uid_t uid, gid_t gid:运行进程的用户的真实用户 ID 和组 ID(RUID,RGID)

uid_t euid, gid_t egid:用于权限检查(文件系统除外)的有效用户 ID 和组 ID(EUID,EGID)

uid_t suid, gid_t sgid:保存的用户 ID 和组 ID(SUID,SGID),用于权限切换

uid_t fsuid, gid_t fsgid:用于文件系统访问检查的用户 ID 和组 ID(FSUID,FSGID)

gid_t groups[NGROUPS]:补充组,用户有成员资格的组列表

struct fs_struct *fs:进程所在文件系统信息.如根、创建掩码等

kernel_cap_t cap_effective,cap_inheritable,cap_permitted;int keep_capabilities:1:能力信息.

进程的这些属性记录的信息包括:进程的组属性、进程的身份、进程的运行环境以及对资源的访问控制信息等,这些都是安全操作系统中的重要信息.

- 取证活动

系统中的取证活动包括取证人员的登陆、取证策略的确定、取证模块的组合、系统配置、证据的获取、证据的保存、证据的传输和证据的表示等等.这些活动都是由取证进程完成的.

- 取证资源

取证资源主要是指对什么样的数据源进行证据的获取.在 SeFOS 中,本文采用的数据源包括进程、文件、系统调用、内核中的关键数据和网络数据.这些本文将在第 3 节进行详细叙述.

- 取证策略

取证策略就是计算机取证的一个规划,要确定取证的内容,以及如何配置才能最大限度地获取所需要的证据数据而最小化对于系统和目标计算机介质的影响等等.取证策略要依赖于建立取证策略模型来加以描述.为了满足不同的取证需求,SeFOS 的取证策略为多级取证策略.本文在此参照文献[15]的方法,给出取证策略的形式化表示.

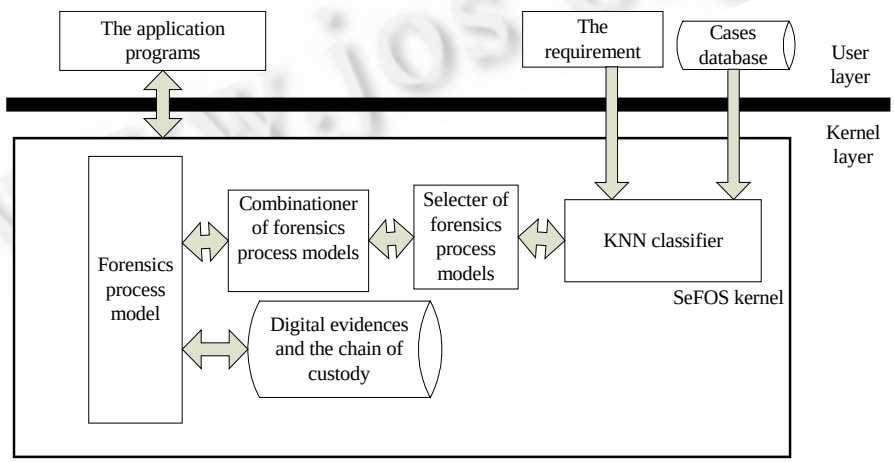

Fig.2 General architecture of SeFOS

图 2 SeFOS 的总体结构

定义 **2**(多级取证策略模型框架). 多级取证策略模型定义为

$$MFP=(SU,OB,FB,FPB,G).$$

其中,*SU* 表示主体的类型,*SU*=(*U*,*P*,*FU*),*U* 是用户的集合,*P* 是进程的集合,*FU* 是具有取证人员特权的用户.*OB* 是客体的集合,*FB* 是取证行为的集合,*FPB* 是取证策略库,*G* 是要达到的取证目标.

定义 **3**(取证策略项). 取证策略项是一个十元组:

(*CaseID*,*Subject*,*Object*,*Action*,*Evidence_Type*,*Frequent*,*T*1:*T*2,*Result*,*Harsh_Value*,*Investigator*).

其中,*CaseID* 是通过 KNN 分类器确定的案件的类别编号.*Subject*,*Object*,*Action* 分别表示主体、客体和行为.*Evidence_Type* 表示证据种类,*Frequent* 表示取证的频度,*T*1 表示取证的开始时间,*T*2 表示取证的结束时间,*Result* 表示 *Action* 的结果,*Harsh_Value* 表示该条证据的 *Harsh* 值,最后一项 *Investigator* 表示具体的取证人员的数字签名.

例如,(23,*Process*2,*F*1,*Create*,*System_Call*,1,7:22,*Both*,*H*1,*Tom*)表示对于 23 号案件,进程 *Process*2 对于文件 *F*1 的创建操作无论成功与否(both),获取相应的系统调用证据,获取 1 次,取证的时间是每天 7:00~22:00,获取的证据的 Harsh 值是 *H*1,该策略的配置人为用户 Tom.

定义 **4**(取证策略规则库 **FPB**). 取证策略规则库由多个取证策略项(称为初始取证策略项)和若干取证策略规则组成,其中,取证策略项和取证规则可以根据具体的取证需求来定义.

基于上述讨论,本文用 B 方法[16]建立由若干取证模块组成的取证系统的抽象机.

定义 **5**(取证系统抽象机).

非形式规范.整个系统包括若干个取证模块(每个模块对应一个取证策略),需要按需求对这些模块进行选择、组合和配置以完成取证功能.而每一个取证模块由实体、客体和操作等部分组成,管理的实体包括:用户、证据数据等.操作包括证据的获取和证据的存储.用户包括取证人员和其他用户.证据数据包括进程、系统调用、文件、系统网络数据和内核中的关键数据资源.系统要满足的非形式规则有:取证模块是根据取证策略确定的;取证策略则是按照取证需求、案例库和取证策略规则库确定的;取证系统需要安全保护;取证人员有取证系统管理和配置权限;证据数据要记录 hash 值、时间戳和取证人员.当证据存储溢出等意外情况发生时,系统操作失败.

下面是 User,Evidence,Forensics,Forensics_System 四个抽象机的简单定义(由于篇幅限制,略去了有关操作的详细描述):

- 抽象机 User

MACHINE

User

SETS

USER

CATEGORY={*investigator*,*administrator*,*normal*}

CONSTRANTS

privilege

PROPERTIES

$privilege \in CATEGORY \rightarrow (enable,disable)$

$privilege=\{investigator \mapsto enable$

$administrator \mapsto disable, normal \mapsto disable\}$

VARIABLES

user,*category*,*allowance*

INVARIANT

$user \in USER \wedge category \in user \rightarrow CATEGORY \wedge$

allowance∈*user*→*NAT*

INITIALIZATION

user,*category*,*allowance*:=∅,∅,∅

OPERATIONS

i←*create_user*(*a*)≙

PRE

a∈*NAT*∧*user*≠*USER*

THEN…END;

i←*read_user*≙

PRE user≠∅ THEN *i*∈*user* END;

modify_user(*i*,*k*)≙…;

modify_allowance(*i*,*a*)≙…;

END

- 抽象机 Evidence

MACHINE

Evidence

SETS

EVIDENCE;

STATUS={*available*,*stored*}

HARSH

VARIABLES

evidence,*type*,*status*,*caseId*,*harsh*

INVARIANT

evidence∈*EVIDENCE*∧

type∈*evidence*∧

status∈*evidence*→*STATUS*∧

caseId∈*evidence*→*NAT*

harsh∈*evidence*→*HARSH*

INITIALIZATION

evidence,*type*,*status*,*substitute*:=∅,∅,∅,∅

OPERATIONS

e←*create_evidence*(*i*)*A*

PRE

i∈*NAT*∧*evidence*≠*EVIDENCE*

THEN

…

END;

modify_type(*e*,*i*)≙…;

modify_unavailable(*i*)≙…;

PRE

e∈*evidence*

THEN

Status(*e*):=*stored*

END;

create_harsh(*e*)≜…;

END

- 抽象机 Forensics

MACHINE

Forensics

USES

User,*Evidence*

SETS

FORENSICS

PLOLYCY

POLIYCY_CONTENT={*CaseID*,*Subject*,*Object*,*Action*,*Evidence_Type*,*Frequent*,*T*1:*T*2,*Result*,*Harsh_Value*,*Investigator*}

CASE

VARIABLES

policy

INVARIANT

policy∈*POLIYCY*→*POLIYCY_CONTENT*∧

forensics∈*FORENSICS*

INITIALIZATION

policy,*forensics*:=∅,∅

OPERATIONS

Collect(*e*)≜…;

store(*e*)≜…;

END

- 抽象机 Forensics_System

该抽象机是以上 3 个抽象机的组合,后边加了几个操作,以便定义错误报告.

MACHINE

Forensics_System

EXTENDS

User,*Evidence*,*Forensics*

OPERATIONS

b←*some_user_exists*≜…;

b←*some_evidence_exists*≜…;

b←*evidence_available*≜…;

b←*quantity_of_evidence_not_exceed*≜…;

END

3 SeFOS 的数据采集

数据采集是计算机取证的基础,完整而有效的数据采集是准确、及时地获取证据的基础.数据采集模块应该能够根据取证策略的要求高效地获取有效而全面的证据数据.一般地,在可取证操作系统中,证据数据源包括

进程、文件、系统调用、内核中的关键数据和网络数据.证据数据的采集流程如图 3 所示.

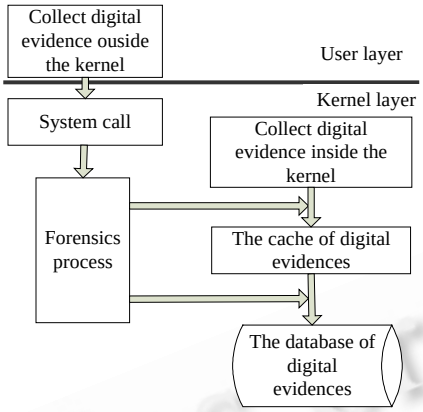

Fig.3 Process of the evidence collection

图 3 证据数据的采集流程

3.1 证据数据源

不同的取证需求对应不同的取证策略,不同的策略决定了不同的证据数据源的集合.下面讨论 SeFOS 的证据数据源.

3.1.1 来自进程的数据

这里的进程是指系统中正在运行的进程所组成的集合.进程类证据的特征属性可以用一个向量表示:

(*PID*,*MemoSpace*,*CpuTime*,*MemoAddress*,*Privilege*),

其中,*PID* 是进程的 *ID*,*MemoSpace* 是进程占用的内存空间,*CpuTime* 是该进程运行占用 CPU 的时间,*MemoAddress* 为运行时所在的地址空间,*Privilege* 表示运行权限.

基于进程这一数据源,可以根据获取的特征值确定进程运行的指定代码空间段的指令序列,确定进程以什么样的权限执行什么样的系统操作.

3.1.2 来自文件系统的数据

文件系统是操作系统的重要组成部分[17].这里,作为证据数据源的文件是指操作系统中的由取证策略确定的关键文件的集合.文件类证据的特征值也可以用向量表示为

(*File_Type*,*File_PID*,*File_Content*,*Accessed_Type*,*Accessed_Privilege*),

其中,*File_Typeb* 表示文件的类型,*File_PID* 表示操作该文件的进程 ID,*File_Content* 表示文件的内容,*Accessed_Type* 表示文件被访问的形式,*Accessed_Privilege* 表示访问权限.

基于文件这一数据源,可以获取什么样的文件被什么样的进程以什么样的权限访问过,并且可以确定访问的形式是读、写还是修改.

3.1.3 来自系统调用的数据

系统调用是操作系统核心向用户提供的操作硬件设备、请求内核服务的接口.系统调用接口位于用户态与核心态之间.一般的系统调用过程是:用户程序通过系统调用向操作系统内核请求服务,操作系统内核完成服务,将结果返回给用户进程.系统调用提供了访问内核的机制,提高了系统的安全性,保证了应用程序的可移植性.Linux 是利用软中断机制实现系统调用的,用户程序通过 int 0×80 中断指令陷入核心,操作系统核心执行中断服务例程,并将结果返回给用户程序[18].

获取的系统调用相关的数据可以表示为

(*SystemCall_Name*,*SystemCall_Function*,*ObjectInformation*,*Start_Parameter*),

其中,*SystemCall_Name* 表示系统调用的名称,*SystemCall_Function* 表示系统调用的功能,*ObjectInformation* 表示

客体及其信息,*Start_Paramete* 表示入口参数.

3.1.4 来自内核中的其他关键数据

内核是 Linux 操作系统的核心.它管理着所有的系统线程、进程、资源和资源分配.与其他操作系统不同,Linux 操作系统允许用户对内核进行重新设置.在操作系统中,与进程相关的所有信息都存在于该进程的进程控制块中,以便于进程的控制和管理.通过对系统内核资源的取证,可以获取与有关事件相关的系统资源分配状况.

(*Cpu_load*,*MemoSpace*,*DiskSpace*),

其中,*Cpu_load* 表示 CPU 负载,*MemoSpace* 表示内存空间,*DiskSpace* 表示磁盘空间.

3.1.5 来自系统的网络数据

网络数据主要包括主机中与网络相关的内核数据.用向量表示为

(*IP_Address*,*Protocol*,*Host_Bandwidth*,*Network_Package*),

其中,*IP_Address* 表示连接的 IP 地址,*Protocol* 表示网络协议,*Host_Bandwidth* 表示主机带宽,*Network_Package* 表示网络数据包.

系统网络数据的获取可以达到网络证据实时获取的目的.

以上数据源包含的内容存在交叉,可以通过设定优先级和标记的方法避免重复采集.

3.2 证据数据的采集

数据源确定以后,需要确定如何采集数据.把系统调用作为数据采集点不仅可以获取以上各类数据源的数据,而且可以帮助获取通过隐蔽通道访问系统资源的数据.而资源的分配不是通过系统调用实现的,它是由系统统一分配给用户使用的.在安全操作系统中,由于访问控制的实施,有些用户进程不能直接访问内核的数据或者直接调用内核的函数.所以,把数据采集点确定为系统调用和内核资源.这里,把事件级取证和系统调用级取证相结合,按照事件确定取证的进程和系统调用等,获取的系统调用又要支持事件的重构.SeFOS 由取证策略确定取证事件,由取证事件确定需要获取的主体、客体和操作.

定义 **6**(取证事件). 取证事件可以定义为一个三元组:(*Su*,*Ob*,*Op*).其中,*Su* 表示主体,就是事件的实施者,一般指进程;*Ob* 表示客体,是事件的被实施者;*Op* 是形成这一事件的操作.比如,一个进程 *p*1 对于文件 *f*1 进行了读操作,这一事件可以用 *E* 表示,*E*=(*p*1,*f*1,*read*).这里的 *p*1 是主体,*f*1 是客体,操作是 *read*.

基于上述定义,内核中的取证确定为按主体取证、按客体取证和按操作取证 3 类.按主体取证就是根据取证需求确定的取证策略,如果要对若干可疑主体进行取证,就可以把这些主体通过配置确定为取证主体,取证模块会把与该主体有关的事件全部捕获并进行存储.按客体取证就是当取证策略要求对于若干客体进行取证时,取证机制根据配置就会捕获与可疑客体相关的事件.按操作取证就是取证进程按照要求对与特定的操作相关的事件进行取证.一般来说,SeFOS 可以通过系统配置确定:关键主体涉及的进程等数据的捕获、关键客体涉及的进程等数据的捕获、关键操作涉及的进程等数据的捕获.当然,也可以把上述 3 种方法结合起来使用.

文献[19]叙述的系统调用劫持技术,通过修改对应系统调用表和中断描述符表的相应表项的值,可以实现系统调用劫持,获取系统调用队列.参照文献[20],对于每一个系统调用设置两个钩子函数:在系统调用的入口处设置一个钩子函数,获取有关的参数;在系统调用的结束处插入一个钩子函数获取事件是否成功的信息.这两个钩子函数表示如下:

1) *foren_validate*(*int foren_num, va_list args*)

函数功能:此函数在系统调用的入口处插入,用于判定是否对该系统调用进行取证和新建取证记录,获取证据信息.

参数说明:*int foren_num* 是证据记录编号.*args* 是证据信息,此参数为可变参数,不同的事件取不同的值,是一个参数列表.

2) *foren_to_buf*(*long value*)

函数功能:在系统调用返回之前调用该函数,其作用是将取证记录进行存储.

参数说明:参数代表系统调用的返回值,通过这个返回值可以确定系统调用的出口.

加入取证钩子函数之后的系统调用代码结构如图 4 所示.

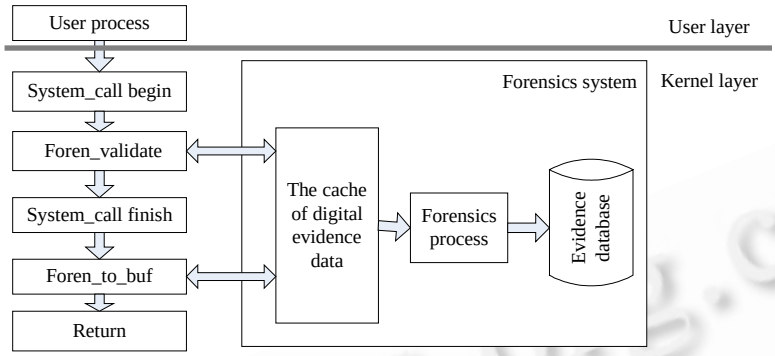

Fig.4 Capture of system call

图 4 对系统调用的取证

4 电子证据的安全保护

电子证据的安全保护包括取证进程的安全保护和对于获取的证据数据的安全存储.

4.1 取证进程的安全保护

文献[21]分析了 3 种日志文件保护机制,提出了采用改进的访问控制的方法来保护文件和进程.阐述了一种安全隔离环境的方法,即在内核层用强制访问控制机制构造隔离的安全环境,让取证进程运行于这个安全隔离环境中,其他进程无法对此环境进行访问,从而也就无法对取证进程和文件进行访问.由于系统设置成了动态可配置的,可以利用 Linux 可扩展内核技术让用户应用程序向内核植入(inject)代码以扩展其功能.这样,被植入的代码就要在核心态运行,需要有一个实时检查机制来验证代码的安全性,保证植入的代码不会导致系统的崩溃.文献[22]使用软件故障孤立技术来实现扩展模块的故障孤立,保证系统不受到扩展模块的破坏.文献[23]也使用软件故障孤立技术来限制被植入代码可访问的内存地址范围.代码安全性检查无疑增加了系统的负担,问题产生的根本原因是那些被植入的扩展性代码对核心来说是不可信任的.为了减轻系统的负担,就需要假设被植入的代码是安全的、可信任的.因此,我们假定取证系统是可信的.Linux 系统提供了 *who*,*w*,*ps* 和 *top* 等察看进程信息的系统调用.但是,这些命令执行程序工作在用户态,本身就不安全,入侵系统的黑客可以轻松地找到这些进程监控程序的磁盘映像,进行删除甚至替换.而且,使用命令监控进程要占用内存空间,会覆盖一些有用的证据数据.因此,必须在核内对进程实施保护.SeFOS 利用 Linux 的内核进程保护技术实现了取证进程的安全保护.

本文借鉴文献[24]的思路构建了内核中的进程保护机制.其运行过程为:在无干扰系统环境下,全面地运行系统中的安全进程,分析和搜集 Linux 环境下这些进程的相关信息:

(*Process_Id*,*Process_Name*,*Process_exe_mapping*,*Start_Time*,*Parent_Process*),

其中,*Process_Id* 是进程的 ID,*Process_Name* 表示进程的名称,*Process_exe_mapping* 是进程的可执行映像,*Start_Time* 进程的开始时间,*Parent_Process* 代表进程度的父进程信息.这样就形成了一张“系统安全进程列表”,作为进程监控的依据.监控代码在进程调度过程中实时地搜集系统中运行进程的信息.如果发现进程不在“系统安全进程列表”当中,则马上通过终端输出该进程的 PID 号、名称、进程的可执行映像等信息,或者通过声音向用户报警,等待用户处理.在这个等待过程中,终止调度该进程,直到用户作出响应(放行该进程或者杀死该进程).判断报警的进程是否是取证进程,如果是,运行并隐藏该进程.如果特别的用户(不同于系统管理员的取证用户)放行了该进程,则可以将该进程加入“系统安全进程列表”,以完善该列表;如果是一般用户在使用过程中放行了某个进程,那么,需要将该用户的用户名和身份记录下来,并且将放行的进程记录下来,存为日志,作为取证用户

在审核用户行为或者修改“系统安全进程列表”时的一个有力的依据.在系统运行过程中,如果发现“系统安全进程列表”中的某些重要进程(包括 *kswapd*,*bdflush* 等)不处于运行状态,则马上将该进程“遗失”的信息存入文件,以备在系统的恢复过程中对它们进行针对性的恢复.根据不同情况,有的需要马上停机,恢复进程,有的则可以现场恢复,如图 5 所示.

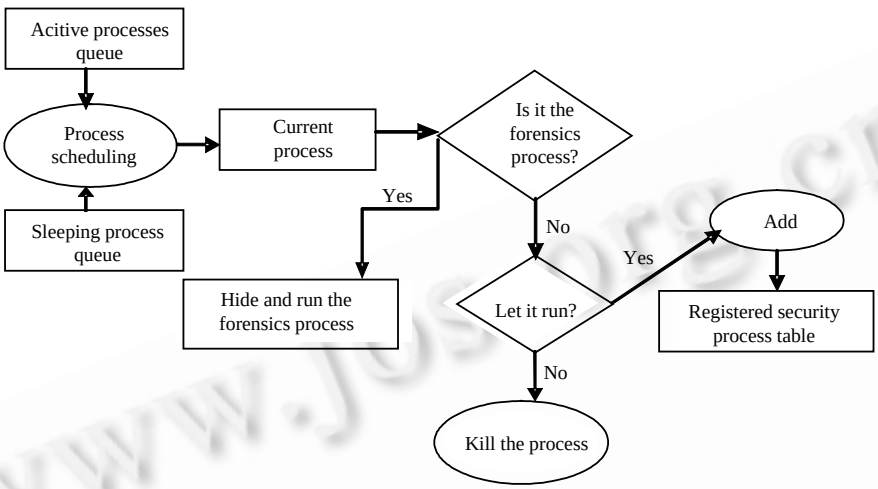

Fig.5 Protection mechanism in the kernel of SeFOS

图 5 SeFOS 内核中的取证进程保护机制

4.2 证据数据的安全存储

证据数据的安全存储包括证据文件的安全和证据数据的溢出两个方面的问题.证据文件的安全可以采用加密文件系统实现[25,26].文献[27]提出了证据包(digital evidence bags)的概念来保证证据文件的安全性.数据的存储溢出是由于获取的证据数据量大而导致的存储空间不足的问题.这一问题可以采用两种方式加以解决:缓冲机制和专用证据数据服务器.

参考文献[28]的审计数据存储方法,SeFOS 采用的证据数据的存储方法是:在采集点获取的证据记录先存放在在核心区开辟的审计记录缓冲区中,等待位于内核的取证进程将其保存到磁盘数据库文件中.为了防止证据记录的溢出并保证系统的可用性,可以对取证系统的数据文件所使用的最大磁盘空间作一个限定,记为 Max_fdbsize.当证据文件将要发生溢出时,内核中的取证进程将停止向磁盘写入证据记录,但要写入标志性记录(FOREN_DATA_SWITCH).同时,根据设定的报警比例,当写入的记录达到 Max_fdbsize 的一定比例时,取证系统要向取证人员发送电子邮件并弹出报警窗口.证据数据库记录取证人员的信息(保证主体合法),对于每条证据记录要赋予 hash 值并加入时间戳.同时,取证进程要进行安全保护,取证人员作为系统的一个独立于系统管理员和其他用户的角色要采取访问控制措施.即,系统中只有取证人员这一角色可以进行取证机制的配置、证据数据的处理等操作,其他任何角色均不赋予该权限.

证据文件的转储分为以下两个阶段:

第 1 个阶段是证据文件验证码的传输.取证模块在生成一个证据文件之后,首先为这个证据文件计算验证码并传送给证据库模块,然后再释放文件锁,将文件写入磁盘文件系统.为了防止入侵者伪造验证码,原型系统在取证模块和证据库中各自设置了一个计数器.当取证模块和证据库建立连接后,两个模块的计数器同时清零.在发送证据验证码时,取证模块将计算得到的验证码和当前的计数器数值一并加密后发送给证据库模块,并等待证据库模块的回复消息.收到回复消息后,检查回复消息中的序号,如果与当前计数器数值相等,则递增计数器.如果等待回复消息超时,则取证模块自动以当前计数器数值为序号重发验证码.另一方面,证据库在接受到验证码后,首先将验证码的序号与自己的当前计数器值进行比较,若相等,则将验证码保存在证据验证码列表

里,把当前计数器数值加密后回复给取证模块,然后递增计数器;否则,拒绝验证码入库,不发送回复消息.

第 2 个阶段是证据文件本身的入库.证据库模块从取证模块接受到一个完整的证据文件以后,计算出它的验证码并与验证码列表中的当前码值进行比较,两者一致则同意证据入库,并将验证码列表的当前指针向下移动一位;否则,说明证据文件已经遭到入侵者的非法修改,证据库模块会拒绝接受证据文件,并向取证管理员发出入侵警报.

5 实验验证

假定目标主机为银行的一台用于处理客户业务的计算机,有一位用户名为 Smith 的用户被犯罪分子以网络钓鱼(phishing)的形式窃取了网络银行账号的用户名和密码.SeFOS 的实时取证功能可以获取该犯罪分子的作案证据.取证策略可以确定为:对于所有的登陆用户的行为进行取证.利用设置的取证策略获取该用户登录的证据信息,确定犯罪嫌疑人的登陆时间、IP、登陆后所发生的事件等,作为案件侦破的依据和证据,以便在日后的诉讼活动中予以出示.

由取证策略确定的需要获取的证据包括获取用户登陆和操作的进程以及有关的系统网络数据、获取与网银有关的进程和有关的系统调用等数据、获取与网络银行有关的操作等.

表 1 是用 SeFOS 获取的实时证据数据列表.进程(6062,login,0.06s,1116K,N/A),(6092,bash,0.07s,1754K,Smith/customer)和(6105,passwd,0.06s,956K,Smith/custommer)表示有人盗用了 Smith 的帐户并修改了帐户的密码;系统调用(open,file,(“/etc/passwd,O_RDWR,600)),(write,file,(3,0x6355A400,10))和(close,file,(3))表明该用户打开了/etc/passwd 文件并重写了文件的内容;网络数据(69.211.195.11,telnet,70Kbps,86)表示 IP 地址为 69.211.195.11 的用户使用 telnet 协议登陆主机,内核资源分配(0.3%,108848K,400M)表明当时的 CPU 负载是 0.3%,内存空间的占用是 108 848K,磁盘空间占用是 400M.据此我们可以认定,犯罪嫌疑人以被害人用户 Smith 的身份在 2006-02-01/19:03~2006-02-01/19:10 这段时间里曾经从 IP 地址是 69.211.195.11 的终端登陆到目标主机,进行了登陆帐户和修改密码等操作.根据 IP 地址可以确定嫌疑人的地理位置.登陆的时间也是重要的证据或线索.表 1 的数据为进一步地分析和鉴定提供了基础数据.

Table 1 Evidences collected by SeFOS

表 1 SeFOS 获取的证据数据

Evidence_type	Evidence_list	*T*1 (the start time)	*T*2 (the end time)
Processes	(6062,login,0.06s,1116K,N/A)	2006-02-01/19:03	2006-02-01/19:10
	(6092,bash,0.07s,1754K,Smith/customer)	2006-02-01/19:04	2006-02-01/19:09
	(6105,passwd,0.06s,956K,Smith/customer)	2006-02-01/19:06	2006-02-01/19:09
System calls	(open,file,(“/etc/passwd,O_RDWR,600))	2006-02-01/19:06:00.04	2006-02-01/19:06:00.06
	(write,file,(3,0x6355A400,10))	2006-02-01/19:06:00.07	2006-02-01/19:06:00.10
	(close,file,(3))	2006-02-01/19:06:00.10	2006-02-01/19:06:00.11
Network data	(69.211.195.11,telnet,70Kbps,86)	2006-02-01/19:03	2006-02-01/19:10
Resources assignments	(0.3%,108848K,400M)	2006-02-01/19:03	2006-02-01/19:10

6 结 论

事后取证难以获取全面的证据数据,因此影响了取证调查人员得出结论的正确性.本文在对现有的基于操作系统的计算机取证技术进行深入分析和研究的基础上,提出了一种基于安全操作系统的可取证操作系统 SeFOS.该系统基于 Linux 开发,取证模块动态加载,运行于内核,实现了实时取证的目的.本文重点描述了 SeFOS 的总体结构,建立了 SeFOS 的取证行为模型,对该模型进行了有关形式化描述,分析了 SeFOS 的取证服务和取证机制,并对 SeFOS 的数据采集和证据数据的保护进行了论述.经实验验证,SeFOS 可以实时获取电子证据数据.今后的研究工作应该是:对各种取证策略的进一步研究,对取证系统抽象机的进一步精化,基于 KNN 的案件分类器的构造,取证模块的组合方法,取证进程和证据数据文件的安全保护机制的进一步研究,取证机制对操作系统运行效率的影响的研究等等.

References:

[1] Sarmoria CG, Chapin SJ. Monitoring access to shared memory-mapped files. In: Proc. of the 2005 Digital Forensics Research Workshop (DFRWS). New Orleans, 2005. http://www.dfrws.org/2005

[2] Betz C. DFRWS 2005 challenge report. 2005. http://www.dfrws.org/2005/challenge/ChrisBetz-DFRWSChallengeOverview.html

[3] Ding LP, Wang YJ. Study on multi-dimension computer forensics model. NetInfo Security, 2005,58(10):73−74 (in Chinese with English abstract).

[4] Eckstein K, Jahnke M. Data hiding in journaling file systems. In: Proc. of the 2005 Digital Forensic Research Workshop (DFRWS). New Orleans, 2005. http://www.dfrws.org/2005

[5] Wang W, Daniels TE. Network forensics analysis with evidence graphs (demo proposal). In: Proc. of the 2005 Digital Forensic Research Workshop (DFRWS). New Orleans, 2005. http://www.dfrws.org/2005

[6] Roesch M. Snort-Lightweight intrusion detection for networks. In: Proc. of the 13th USENIX Conf. System Administration (LISA'99). Seattle: USENIX Association, 1999. 229−238. http://www.usenix.org/events/lisa99/full_papers/roesch/roesch.pdf

[7] Baliga A, Gupta N, Kaufman L, Mekaraj P, Tjang A, Xu WY. Network monitoring and forensics. http://www.research.rutgers.edu/~aratib/presentations/forensics_paper.pdf

[8] Guide to SNARE for windows. http://www.intersectalliance.com/resources/Guide_to_SNARE_for_Windows.pdf

[9] Goel A, Feng WC, Maier D, Feng WC, Walpole J. Forensix: A robust, high-performance reconstruction system. 2005. http://www.eecg. toronto.edu/~ashvin/publications/sdcs-2005.pdf

[10] Dunlap GW, King ST, Cinar S, Basrai MA, Chen PM. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In: Proc. of the 2002 Symp. on Operating Systems Design and Implementation (OSDI). 2002. http://www.eecs.umich.edu/~kingst/ revirt.pdf

[11] Garfinkel T, Rosenblum M. A virtual machine introspection based architecture for intrusion detection. In: Proc. of the 2003 Network and Distributed System Security Symp. (NDSS). 2003. http://suif.stanford.edu/papers/vmi-ndss03.pdf

[12] Liang HL. Research on enforecement of secure operating system supporting multiple security policy [Ph.D. Thesis]. Beijing: Institute of Software, the Chinese Academy of Sciences, 2002 (in Chinese with English abstract).

[13] The adoption standard of evidences (in Chinese with English abstract). http://www.flgw.cn/lunwen/law/200512/24983.html

[14] Ding LP, Wang YJ. Study on relevant Issues about law enforcement and computer forensics. Journal of Software, 2005,16(2): 260−275 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/260.htm

[15] He YZ, Li L, Feng DG. A generic audit policy model on multilevel secure DBMS. Journal of Software, 2005,16(10):1774−1783 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1774.htm

[16] Abrial JR, Wrote; Qiu ZY, Trans. The B-Book: Assigning Programs to Meanings. Publishing House of Electronics Industry, 2004 (in Chinese).

[17] Alexandrov AD, Ibel M, Schauser KE, Scheiman CJ. Ufo: A personal global file system based on user-level extensions to the operating system. ACM Trans. on Computer Systems, 1998,16(3):207−233.

[18] Mao DC. Linux Kernel Code Analysis. Hangzhou: Zhejiang University Press, 2001 (in Chinese).

[19] Shi JQ, Fang BX, Hu MZ, Li B. Linux system call hijacking: Technical principles, application and detection. Computer Engineering and Application, 2003,32:167−170 (in Chinese with English abstract).

[20] Chen H. The design and performance of security audit system [MS. Thesis]. Beijing: Institute of Software, the Chinese Academy of Sciences, 2004 (in Chinese with English abstract).

[21] Sun B, Sun YF, Zhang XF, Liang B. Research and protection of the digital evidence collecting system. Journal of Computer Research and Development, 2005,42(8):1422−1426 (in Chinese with English abstract).

[22] Cheng BQ, Yin BL. Dynamic expansion embed OS. Mini-Micro Systems, 2003,24(2):216−217 (in Chinese with English abstract).

[23] Bershad BN. Extensibility, safety and performance in the SPIN operating system. In: Proc. of the 15th ACM Symp. on Operating Systems Principles. 1995. http://portal.acm.org/citation.cfm?id=224077&coll=Portal&dl=GUIDE&CFID=25040769&CFTOKEN=96584822#

[24] Li Y. How to monitor and protect the security processes under Linux. 2005 (in Chinese with English abstract). http://www.gbunix.com/print.php?articleid=1299

[25] Xing CL, Qing SH, Li LP. The design and implementation of an encrypted file system for Linux. Computer Engineering and Application, 2005,17:101−104 (in Chinese with English abstract).

[26] Schneier B, Kelsey J. Secure audit logs to support computer forensics. ACM Trans. on Information and System Security, 1999,2(2): 159−176.

[27] Turner P. Unification of digital evidence from disparate sources (digital evidence bags). In: Proc. of the 2005 Digital Forensics Research Workshop (DFRWS). New Orleans, 2005. http://www.dfrws.org

[28] Zhang XF. Security audit and audit-based intrusion detection [Ph.D. Thesis]. Beijing: Institute of Software, the Chinese Academy of Sciences, 2004 (in Chinese with English abstract).

附中文参考文献:

[3] 丁丽萍,王永吉.多维计算机取证模型研究.信息网络安全,2005,58(10):73−74.

[12] 梁洪亮.支持多安全政策的安全操作系统的研究和实现[博士学位论文].北京:中国科学院软件研究所,2002.

[13] 证据的采用标准.http://www.flgw.cn/lunwen/law/200512/24983.html

[14] 丁丽萍,王永吉.计算机取证的相关法律技术问题研究.软件学报,2005,16(2):260−275. http://www.jos.org.cn/1000-9825/16/260.htm

[15] 何永忠,李斓,冯登国.多级安全 DBMS 的通用审计策略模型.软件学报,2005,16(10):1774−1783. http://www.jos.org.cn/1000-9825/16/1774.htm

[16] Abrial JR,著;裘宗燕,译,B 方法.北京:电子工业出版社,2004.

[18] 毛德操.Linux 内核情景分析.杭州:浙江大学出版社,2001.

[19] 石金桥,方滨兴,胡名曾,李斌.Linux 系统调用劫持:技术原理、应用及检测.计算机工程与应用,2003,32:167−170.

[20] 陈慧.安全审计系统的设计与实现[硕士学位论文].北京:中国科学院软件研究所,2004.

[21] 孙波,孙玉芳.电子数据证据收集系统的研究与保护.计算机研究与发展,2005,42(8):1422−1426.

[22] 程步奇,尹宝林.可动态扩展的嵌入式操作系统.小型微型计算机系统,2003,24(2):216−217.

[24] 李洋.如何监控和保护 Linux 下进程的安全.2005. http://www.gbunix.com/print.php?articleid=1299

[25] 邢常亮,卿斯汉,李丽萍.一个基于 Linux 的加密文件系统的设计与实现.计算机工程与应用,2005,17:101−104.

[28] 张相峰.安全审计与基于审计的入侵检测[博士学位论文].北京:中国科学院软件研究所,2004.

丁丽萍(1965－),女,山东青州人,博士,副教授,主要研究领域为信息安全,操作系统,计算机取证,软件工程.

周博文(1985－),男,硕士生,主要研究领域为信息安全,操作系统,计算机取证.

王永吉(1962－),男,博士,研究员,博士生导师,主要研究领域为实时系统,网络优化,智能软件工程,非线性优化理论,实时混合控制理论,实时嵌入式操作系统.