一个安全操作系统的通用审计框架*)

夏 耐 茅 兵 谢 立

(南京大学软件新技术国家重点实验室 南京210093)

摘 要 本文首先介绍了审计的基本概念和本文所描述工作所基于的项目背景。然后切入本文的重点——一个按照POSIX1003.1e标准,通用的安全操作系统审计框架的设计工作,以及此框架按照GB 17859-1999第三级别要求在一个Linux安全增强的操作系统上的具体实现工作。同时本文比较了国内相似的工作,分析了整个审计框架的优点与不足,并展望了将来的工作。

关键词 安全操作系统,审计,日志,主体,客体,事件相关信息

A Generic Audit Framework for Secure Operating

XIA Nai MAO Bing XIE Li

(The State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

Abstract This paper first introduces the basic conceptions of secure auditing and the project background of the work. As the principal part of this work, the design of a platform and level independent audit framework conforming to POSIX standard and its implementation to a security enhanced linux platform conforming to the GB 17859-1999 third security level standard is presented. Then, this paper pays respect to the similar domestic works analyzing the framework's advantages and disadvantages. Finally, the view of the future work is depicted.

Keywords Secure operating system, Audit, Log, Subject, Object, Case related information

1 前言

所谓审计就是在安全系统中记录、检查以及分析研究程序或者用户的安全相关行为的一系列操作,它已经成为安全操作系统中一个不可或缺的安全机制。

本文首先介绍该工作的项目的背景,其次描述当前一般通用Linux系统中简单审计记录和它们的不足,然后叙述我们怎样根据TCSEC[2]对审计的要求以及POSIX1003.1e[1]中审计实现的标准来进行一个平台和安全级别无关的通用审计框架的设计工作,以及该框架在Linux平台上按照GB 17859-1999第三级别要求的实现工作做具体的描述,最后我们比较国内类似的相关工作给出结论,并预示将来的工作。

2 知识和项目背景

审计的定义首先是由Anderson提出[3],他试图使用审计记录来追踪系统的威胁行为。但是他没有提出怎样改变系统的结构来得到审计记录,而基于已经有审计记录的前提假设以上的。Bonyun[4]认为,一个仔细设计的标准统一地用于审计的进程应该成为安全系统必不可少的一个部分。一般说来,审计的定义可以从审计数据的获取、审计数据的约简、审计记录的分析这三个阶段来理解。

普通Linux内核提供了一套简单的审计机制,大致分成三种:连接时间日志、进程统计(process accounting)、系统日志-syslog。

这套审计机制在一定程度上满足了通用操作系统安全考虑的要求,但是对于一个安全操作系统是不够的,这主要是因为,首先审计的粒度不够,这些审计能够粗略地描述系统中用户和程序的行为,但是对于每个行为的细节(如某个行为所涉及的系统调用)描述不够,不能够精确定位用户和程序的可能对于系统安全产生危害的行为。其次,审计和系统安全机制结合不够紧密,由于非基于一个安全内核,并且读写审计记录文件的为普通进程,审计机制容易被恶意程序绕过和攻击。所以,需要一种安全的全功能的审计机制来对整个系统的安全解决方案提供支持。

本文所描述的工作最终目的是为安全操作系统SOFTOS提供一个符合标准的审计模块。所以这里简单介绍一下SOFTOS的特性以及审计在其中的地位。

安全操作系统SOFTOS是一个符合计算机信息系统安全保护等级划分准则(GB 17859-1999)第三级的多用户、多任务、UNIX类操作系统,SOFTOS在内核中实现了国家标准第三级所需的安全机制,是一个具有自主版权的、具有高度安全保证的开放式的安全操作系统。它的主要特性包括:综合强制访问控制(MAC)和自主访问控制(DAC)及基于进程的访问控制(PBAC)的访问控制机制、最小特权管理、可信路径、隐蔽通道分析、禁止客体重用、安全审计。安全审计机制对于SOFTOS中其它模块的运行提供详细的记录,作为其它模块的监控机制,用以保证系统安全机制的正常运行。所以它是SOFTOS中举足轻重的一个安全模块,它的功用不可或缺。

3 设计与实现

本设计所面对的是一个Linux安全增强系统SOFTOS[6],由于当前基于Linux的安全增强系统的系统在国内外都有很多,审计系统的设计各不相同,彼此接口规范和审计数据结果互不兼容,因此我们的设计并不是特别针对SOFTOS,而是一个建议性质的、通用的审计框架。这个通用框架能对于不同的级别的类似系统、不同审计的要求搭建不同粒度、

*)本文研究得到国家高技术研究发展计划863课题(2001AA144010)资助。夏 耐 硕士研究生,网络安全及系统安全。茅 兵 教授,人机交互、分布计算和并行处理系统安全。谢 立 博导,教授,分布式计算和系统安全。

不同审计事件的审计机制。同时我们期望这个通用框架在已出现的或者未出现的类似系统中能得到应用，增强彼此审计之间的兼容性，并且为 SOFTOS 的将来升级和扩展工作做了铺垫。

所以，我们在以下的设计介绍中首先介绍这样一个符合 IEEE Std 1003.1e 的审计框架（当然所面对的系统都是 POSIX 兼容的），然后再描述按照 GB 17859-1999 第三级别的要求在 Linux 安全增强系统中使用该审计框架所得到的审计机制。

3.1 通用审计框架

由于该框架的通用性，我们需要屏蔽一些和具体级别相关的细节，比如具体的事件类型、事件相关信息等。首先我们来看一下这个框架的组织，如图1所示。

从图中我们可以看出该框架的三个主要组成部分：

1）审计函数库，包括一组用于产生处理审计数据以及审计机制向系统或者文件提交审计数据的审计 I/O 的函数所组成。

2）审计管理进程，用一个后台进程或者线程专门管理系统审计，如管理系统审计缓冲区，查找打开系统审计日志文件等。

3）审计数据的约简与分析，将产生的审计数据进行约简分析以便作为系统安全的参考。

这里我们的主要工作在于审计数据的收集以及审计数据的存放，约简与分析工作所设计的内容太多，超出本文的论述范围，这里不作详细说明。

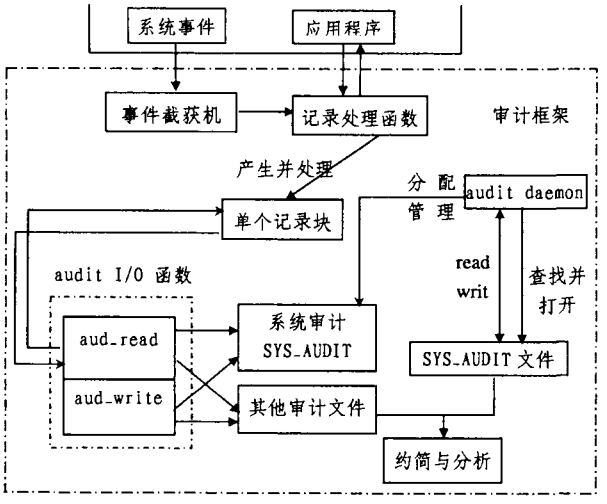

图1 框架结构图

为此，我们分四个步骤来设计这样一个框架。

1）通用的审计数据结构和数据存储方式，以及辅助的库函数。

审计数据结构描述的是一个审计记录所包含的各个部分，根据 IEEE Std 1003.1e 标准，我们依次定义审计记录头（*aud_hdr_t*），审计主体属性描述符（*aud_subj_t*），客体属性描述符（*aud_obj_t*），事件相关信息组描述符（*aud_evinfo_t*）和审计信息结构（*aud_info_t*）。

审计记录头：

```
typedef struct aud_hdr_s{
    magic_t des_magic;                  /* 记录头魔数 */
    size_t aud_hdr_size;                /* 本记录头长度 */
    aud_event_t event_type;             /* 事件类型 */
    /* 由于篇幅关系，省略部分非重要内容  …… */
    aud_subj_t* subj_link;              /* 主体描述符链 */
    aud_obj_t* obj_link;                /* 客体描述符 */
    aud_evinfo_t* event_link;           /* 事件信息组链 */
    unsigned char reserved[AUD_HDR_RES] /* 为具体开发保留 */
} aud_hdr_t;
```

其中的大部分的具体定义含义参照 IEEE Std 1003.1e，这里需要说明的是，这里的定义除了个别字段以外，都采用的是自定义类型。目的是为了不同级别系统上具体实现对各个字段可能有的不同需求。记录头魔数的定义是标准上没有的，这里添加的目的是对记录结构合法性的一种简单验证机制，不让非审计记录数据充当我们处理的对象。最后，为了扩展性考虑，保留 *AUD_HDR_RES* 个字节为不同实现中其他辅助的字段准备。

主体属性描述符和客体描述符的内容完全符合 IEEE Std 1003.1e 关于主体、客体字段的要求，由于篇幅关系，这里就不加列出。同样里面大部分的类型仍使用自定义类型。

事件信息组描述符：

```
typedef struct aud_evinfo{
    magic_t des_magic;              /* 描述符魔数 */
    aud_info_t* info_buffer;        /* 审计信息结构链 */
    struct aud_evinfo* next;        /* 下一个事件相关信息组 */
} aud_evinfo_t;
```

审计信息结构：

```
typedef struct aud_info{
    magic_t des_magic;                  /* 结构魔数 */
    aud_info_type_t aud_info_type;      /* 信息类型 */
    aud_item_id_t aud_item_id;          /* 信息内容标识 */
    size_t aud_info_length;             /* 信息长度 */
    void* aud_info_p;                   /* 信息体指针 */
    struct aud_info* next;              /* 下一个信息结构 */
} aud_info_t;
```

审计信息结构可以作为描述一条主客体或者事件相关信息的信息条目，一般来说它作为一个审计记录里面用以存储事件相关信息的结构体，也可以调用审计记录处理函数通过该结构体返回主客体/事件相关信息。

审计记录的存储可以分为三种：内存中动态结构、文件中静态二进制结构和文件中静态字符型结构。

审计记录在内存中的组织必须能方便审计处理函数的生成以及处理，它必须能动态地扩展和删减。所以它的存储必须采用如链表和指针等的动态结构，本框架中一个典型的审计记录的动态组织方式如图2所示。

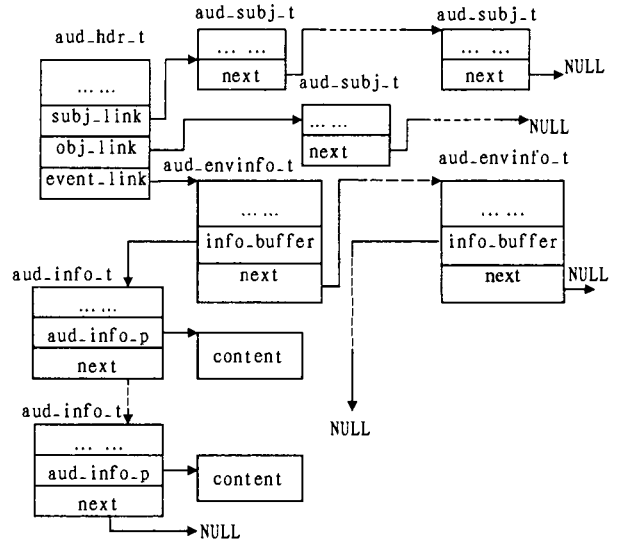

图2 审计记录内存中组织

如果将一个审计记录写入文件，就必须同时规定一个审计记录的连续的静态的存储方式，这就需要每个结构以及整个记录都能有办法确定本身的大小，比如有长度字段，所有链

表都能有办法确定链表条目的个数。一个典型的文件中静态审计记录的组织如图3所示。

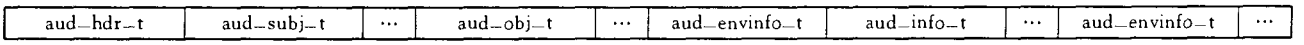

图3 审计记录文件中二进制组织

当然对于整个审计文件，需要有固定的文件格式头来维护审计记录的信息。

为了方便系统和用户使用，对于二进制的审计记录的操作，如二进制记录的产生和删除，记录内部信息的填充与获取，记录的不同存储格式之间的转换，以及审计记录文件的读写都需要有特定的库函数来完成。本框架内次组函数的实现（调用参数、函数语义及返回值）完全按照 IEEE Std 1003.1e 要求，这里不一一列出。

二进制的审计日志格式的设计是出于对模块简单化以及模块速度的考虑，但是出于一些其它方面的考虑，比如本审计模块与其它通用审计工具的交互，审计记录的可读性以及在异构平台之间的兼容性等，一种以字符格式存储的审计日志格式是必要的。所以，我们同时设计了一种可读字符格式存储的日志格式，格式的规范遵从 Matt Bishop[10]的标准审计日志格式，如表1所示。

表1 字符格式记录格式定义表

控制字串	含义	控制字串	含义
#S#	一个记录开始	#Fc#	改变域分隔符为 c
#E#	一个记录结束	#Cc#	改变非打印字符分隔符为 c
#N#	下一个记录（#E#S）	#I#	忽略下一个域，作注释用
#	域分隔符号	\	非打印字符分隔符为 c
\十六进制值\	表示一个16进制的值	属性=值	将“属性”为“值”

举例来说打开文件的事件 *AUD_AET_OPEN*，用可读字符格式表示就为：

 # S # event_type = AUD_AET_OPEN # subject_pid = 123 # subject_uid = 555 # object_uid = 0
 # object_type = AUD_OBJ_BLOCK_DEV # info_AUD_PATHNAME = /dev/cdrom
 # info_AUD_OFLAG = ro # info_AUD_MODE = \0\ # info_AUD_RETURN_ID = -1 # E #

这个记录表明了一个 *pid* 为123，*uid* 为555的进程企图以只读方式打开 cdrom 设备文件，返回结果是打开失败。其中以 *subject_* 开头，*object_* 开头，*info_* 开头的分别为主体属性，客体属性和事件信息属性。*AUD_AET_OPEN* 时间的时间相关信息的定义在 IEEE Std 1003.1e 中有严格定义，本文稍后也有所描述。

2）审计后台维护进程及对缓冲区的管理。

专门用一个后台进程来维护管理系统审计是非常必要的，它的主要职责是查找和生成系统审计文件，维护管理系统审计记录缓冲，注册前来递交和查询系统审计信息的某些特权进程。作为一个和系统安全密切相关的进程，该进程应该受到 TCB 的严格保护。该进程的状态转换关系如图4所示。

从状态转换图中，我们可以看到，对缓冲区刷新的操作的来源可以是多种，我们这里借鉴了 Linux 2.4.x 内核虚存管理进程 kswapd[7]的方法，对审计缓冲区的刷新操作的触发有三种来源：

·当前缓冲区容量不能够满足本次审计记录写入的要求。

·前一次刷新操作间隔时间过长，这样做的目的是使系统审计物理文件的内容不过于陈旧，同时也是减少突然系统 crash 带来的损失。

·审计系统的关闭以及整个系统的重新启动。

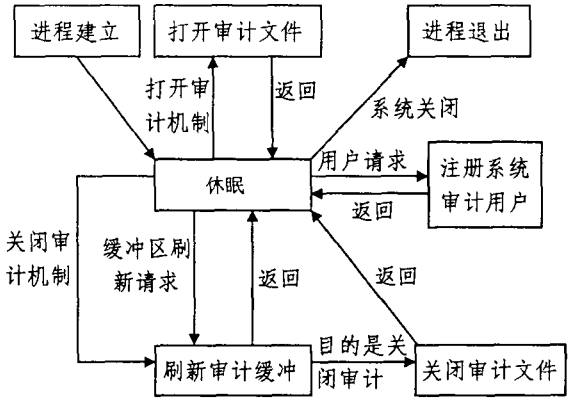

图4 审计维护进程状态转换图

由于缓冲将满所造成的刷新操作本质上涉及到一个非常简单而又经典的读/写者问题，或者说生产者与消费者问题。具体的实现系统可以根据所基于的实现平台上的操作系统内核同步机制来达到这个目的。举例来说，对于类 Linux 内核，对等待队列（wait_queue）以及进程的等待状态（TASK_INTERRUPTIBLE 和 TASK_UNINTERRUPTIBLE）的支持可以很容易地达到这样的效果。

3）审计机制的控制接口。

系统中有关审计的控制包括系统审计机制的打开/关闭，审计机制的各类参数（如缓冲大小，需要审计的事件类型，系统审计路径等）的设定。审计控制接口的目的是为了让系统审计员或者系统管理能按照要求控制审计机制。由于这里设计的是一个通用的框架，因此该阶段给出的是一个与具体平台和系统无关的审计控制接口，具体用户工具如何和该接口交互信息，该阶段暂不考虑。

在框架中，审计的控制采用命令字和命令参数来达到单控制入口的，控制接口的函数声明为“*void aud_ctrl (aud_ctrl_cmd_t cmd, void * args, void * ret);*”其中 *cmd* 为控制命令字，*args* 为命令参数，*ret* 为控制返回信息结构。一个典型的控制接口命令（打开，关闭审计功能，设定事件屏蔽位，刷新审计缓冲）的对应关系一般来说如表2所示。

表2 控制接口定义

cmd	args	ret
AUD_CTRL_OPEN	系统审计日志文件名	≥0成功/ <0 失败
AUD_CTRL_CLOSE	NULL	≥0成功/ <0 失败
AUD_CTRL_SWITCH	系统审计事件屏蔽位	≥0原屏蔽位/ <0 失败
AUD_CTRL_FLUSH	NULL	≥0缓冲内容（字节）/ <0 失败

4)审计约简与分析。

由审计机制产生的初始的审计信息是详细的、大量的且冗余度很高的信息。为了从大量的信息中抽取对系统安全有用的信息,我们必须对审计记录文件做约简和分析工作。而通常这样的约简和分析的工作都和入侵检测(Intrusion Detection)紧密联系,由于这部分的内容比较复杂且自成体系,超出本文知识范围,这里不作详细叙述。本框架在这部分的主要工作主要是:

1) 过滤太多的安全相关性不高的冗余信息,主要是 TCB 本身行为(加载共享运行库)产生的太多的审计事件。

2) 将相关的审计记录事件粗略约简成程序的语义行为。

3) 某些特定入侵检测方法(如系统调用短序列)所需信息从审计记录中抽取。

这部分的工作还不完善,有待更深入的研究和开发工作。它将成为将来工作的重点之一。

3.2 框架在 Linux GB 17859-1999 第三级别系统上的具体实现

由于上面描述的是一个和具体级别无关的 POSIX 兼容的通用的框架,因而在实现某个具体系统的时候必须对这个框架进行具体的实现。这里,我们针对的是两个具体的大方面,一个是 GB 17859-1999 第三级别,另一个是 Linux 增强系统。

对于第一个大方面,主要的工作包括具体化审计可以记录的事件类型即对框架中事件抽象类型进行具体的定义(如定义成某个具体的 *enum* 类型),每个事件相关信息的扩充即按照 GB 17859-1999 第三级别系统的要求对框架中基本事件相关信息定义的扩充,以及操作系统对审计框架的支持。操作系统对审计框架的支持是指操作系统的各项安全策略以及访问机制能够确实保护审计机制的安全,使得它成为 TCB 的一部分不受非法主体干扰。

鉴于篇幅,GB 17859-1999 第三级别要求不再累述。SOFTOS 中对通用审计框架具体化过程中实现了从文件操作到安全控制操作的200个审计事件的截获。同时对于每个可截获的审计事件定义了相关的事件信息,一个典型的事件以及它的相关信息如表3所示。

AUD_AET_OPEN:打开一个文件

表3 open 事件相关信息

信息类型 (aud_info_type_t)	信息含义	信息内容标识 aud_item_t
AUD_TYPE_STRING	打开文件路径	AUD_PATHNAME
AUD_TYPE_INT	打开操作的 flag	AUD_OFLAG
AUD_TYPE_MODE	创建文件的 mode 字	AUD_MODE
AUD_TYPE_INT	返回值(文件句柄)	AUD_RETURN_ID

这样的一组信息能够被通用审计框架中的一个 *aud_evinfo_t* 描述符中连接的若干个 *aud_info_t* 结构所描述。

SOFTOS 对审计机制的保护主要基于访问控制中的强制访问控制(MAC)对审计接口、系统审计文件、审计共享库函数的访问控制,以及基于进程访问控制(PBAC)对审计管理进程的保护(如使得审计管理进程不可打开未经过特别认证的审计日志文件)。此外,SOFTOS 的分权机制中产生的系统审计员(用户名 sysaud,id 号499),对审计控制工具和控制接口拥有特有的运行与控制权,任何其他用户对审计机制的控制行为都会被系统安全策略视为非法。

对于第二个大方面,涉及到审计机制在某个具体操作系统内核上的运行方法,包括具体审计信息的获取途径,审计接口的具体实现方式,审计库的具体实现方式,审计管理进程的运行方式以及审计的工具集。我们对以上的各方面在 Linux 安全系统中的具体实现作如下说明:

·审计信息的获取途径:虽然是针对一个具体的操作系统,但是 Linux 操作系统是一个多平台且频繁升级更新的系统,因此,审计信息的获取力求做到和某一个具体平台无关以及考虑到以后整体内核升级的方便。另一方面,由于获取的途径审计信息的来源遍布操作系统的很多动作中,信息来源比较分散,这对于平台无关和升级方便是很不利的。但是相对来说,操作系统调用入/出口是一个用户/内核动作信息的汇集点,所以,我们把系统调用作为统一的审计信息采集点,即对原来的安全内核的系统调用进行包装,获取所需要的审计信息,而系统调用的其它功能和返回值不变。它的示意图如图5所示。

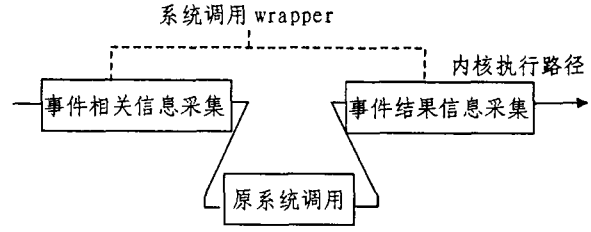

图5 审计信息的截获

我们使用系统调用 wrapper 的主要目的一是为了审计子系统的独立性,尽可能减少与其他子系统过分的耦合,尤其是松散的耦合点,减少子系统之间错误的累计;二是为了达到审计子系统的替换的方便,审计子系统可以作为一个独立的内核模块插入,并且,审计子系统不受内核次要版本升级带来的非重大数据结构或者算法变动带来的不良影响。采用此方法,SOFTOS 在很短时间内实现了从 Linux2.2.x 增强内核到 Linux2.4.x 增强内核升级过程中审计功能的平滑移植。

·审计接口的实现:Linux 下内核和用户态程序交互的方式一般有系统调用、proc 文件系统、字符设备。我们为审计接口,包括审计控制接口选用系统调用的方式,这样也是便于上面所说的审计信息统一获取的方便。

·审计库实现:审计库函数目的是为了提供用户操作审计记录和审计文件的手段,为了不同方面的需要,我们将审计库按照 ELF(Executable and Linking Format)格式生成、so 格式的共享库和.a 格式的静态库。

·审计管理进程:审计管理进程必须能够很好地与系统集成,同时由于它是和安全密切相关的,它必须受到 TCB 的严格保护。为此,我们把它设计成内核态进程(kernel thread),让它在整个系统启动时在许多应用程序之前加载,确保不让其受到非 TCB 进程的干扰。

·审计工具集:审计工具集主要面向的用户是系统审计员,它们按照功能可以分作三类,审计控制工具,审计调试工具,审计数据查看工具和审计数据约简分析工具。这些工具大多是审计框架中审计接口的用户级别的包装和利用,因此,这里不详细叙述。

由于该实现针对的系统是 Linux 的安全增强系统,而 Linux 是一个类 UNIX 系统,因此,该实现对于其它 UNIX 上的实现都是有参考价值的。

4 已有工作与比较

我们在设计这样一个通用的审计框架以及实现的过程中，已经注意到国内已经有了相关方面的工作。《安全操作系统审计的设计与实现》[8]，《Linux Shell 安全审计机制的扩展》[9]中所描述的工作（特别是前者）与我们的工作有很多类似的地方，但是我们的工作不是简单地重复前人的工作，从前面的叙述不难看出我们工作相对国内前人工作的特色之处：

·开放的通用框架的设计。通用框架的设计基于流行的国际标准--POSIX，与具体的实现平台和具体针对的安全级别无关。框架中从数据结构到函数接口的国际标准化使得该框架能够面向更广泛的应用。审计实现要求遵从 TCSEC，进而从实现上也达到与国际接轨。

·审计不仅仅面向系统，审计函数库和系统审计的用户接口使得应用程序能够自主地利用审计机制。这对于审计是个必要的且有力的扩充。

·实现方面的审计子模块与系统其他模块之间的相对高独立性，以及审计截获对系统切入口的统一化，使审计模块和系统内核的独立升级成为可能。

结论与展望 本文所叙述的是一个符合 POSIX1003.1e 标准的通用的审计框架的设计工作以及该框架在 GB 17859-1999 第三级别的 Linux 安全增强系统上的实现工作。审计框架设计部分的工作，希望能对类似的 POSIX 的兼容系统的审计实现都能有所启发和帮助，实现部分的工作也成为了 SOFTOS 的一个子模块。然而这部分的工作主要集中在审计数据的获得和对所获得的审计数据的简单约简之上，面向安全应用（如 IDS）的审计记录分析的工作尚且不足。这方面的工作有望成为以后工作的一个重点，从而进一步完善这个审计框架。

参 考 文 献

1 Draft Standard for Information Technology-Portable Operating System Interface (POSIX)-Part 1: System Application Program-Interface (API)- Amendment #: Protection, Audit and Control Interfaces [C Language] IEEE Standards Department 1997

2 Orange Book Parts I and II: THE CRITERIA and RATIONALE AND GUIDELINES NCSC/DOD/NIST Dec. 1985

3 Anderson J P. Computer Security Threat Monitoring and Surveillance. James P. Anderson Co., Fort Washington, PA, 1980

4 Bonyun D. The Role of a Well-Defined Auditing Process in the Enforcement of Privacy Policy and Data Security. In: Proc. of the 1981 IEEE Symposium on Security and Privacy, 1981. 19～26

5 A Guide to Understanding Audit in Trusted Systems, The Rainbow Books. National Computer Security Center 1987

6 茅兵. 基于 Linux 的安全操作系统的开发. 2000 年中国自由软件发展战略研讨会暨第一届中国自由软件应用论坛会刊. 中国国家高技术智能计算机系统专家组与中国共创软件联盟主办，北京，2000

7 Linux kernel 2.4.2. http://www.kernel.org

8 刘海峰，卿斯汉，刘文清. 安全操作系统审计的设计与实现. 计算机研究与发展，2001，38(10)

9 汪立东，方滨兴. Linux Shell 安全审计机制的扩展. 软件学报，2002，13(1)

10 Bishop M. A standard audit trail format. In: Proc. of the 18th National Information Systems Security Conf. Baltimore, Maryland, USA, 1995

（上接第106页）

参 考 文 献

1 Comer D E. Computer Networks and Internets. Prentice Hall International, Inc., 1997

2 Steinmetz R, Nahrstedt K. Multimedia Computing, Communications and Applications. Prentice Hall International, Inc., 1995

3 Tekalp A M. Digital Video Processing. Prentice Hall International, Inc., 1996

4 胡飞，朱耀庭，朱光喜. Internet 视频点播差错控制. 计算机研究与发展，2002，39(1)：28～34

5 Hu Fei, Zhu Guangxi, Zhu Yaoting. Enhanced ARQ-based Packet Loss Recovery for Real-time Communication. In: Proc. of Intl. Conf. on Info-tech & Info-net (ICII'2001), Beijing, Oct. 2001

6 Girod B, Faerber N. Feedback-based error control for mobile video transmission. Proc. IEEE, Special Issue on Video for Mobile Multimedia, 1999, 87(10), 1707～1723

7 Zhu Q, Wang Y. Error concealment in visual communications. Compressed Video over Networks. Narcel Dekker, Inc., 2000

8 Hong M C, Kondi L, Scwab H, Katsaggelos A. Video error concealment techniques. Signal Processing: Image Communications, special issue on Error resilient Video, 1999, 14(6-8)：437～492

9 胡飞，朱光喜，朱耀庭. Internet 视频通信差错隐藏技术研究. 计算机科学，2002，19(4)：124～127

10 Video coding for low bitrate communication. ITU-T Recommendation H.263, 1998

11 Video codec for audiovisual services at p×64kbits. ITU-T Recommendation H.261, 1993

12 Information technology - Generic coding of audio-vidual objects - Part 2: Visual, Final Proposed Draft Amendment 1. ISO/IEC JTC 1/SC 29/WG 11 N2802 (MPEG-4), July 1999

13 Wen J. Villasenor J. A class of reversible variable length codes for robust image and video coding. In: Proc. IEEE Intl. Conf. on Image Proc., Santa Barbara, CA, Oct. 1997

14 Wen J, Villasenor J. Reversible variable length codes for robust image and video transmission. In: 1997 Asilomar Conf. Pacific Grove, CA, Nov. 1997

15 Naka N, Adachi S, Saigusa M, Ohya T. Improved error resilience in mobile audio-visual communications. In: IEEE Intl. Conf. on Universal Personal Communications, Tokyo, Japan, Nov. 1995, 1., 72～706

16 Coete G, Shirani S, Kossentini F. Optimal mode selection and synchronization for robust video communications over error prone networks. IEEE Journal on Selected Areas in Communications, May 1999

17 Zhang R, Regunathan S L, Rose K. Video coding with optimal Inter/Intra-mode switching for packet loss resilience. IEEE Journal on Selected Areas in Communications, June 2000

18 Wenger S, knorr G, Ott J, Kossentini F. Error resilience support in H.263+. IEEE Trans. on Circuit ans System for Video Technology, 1998, 8(6)：867～877

19 Kondi L, Ishtiaq F, Katsaggelos A K. Joint source-channel coding for scalable video. In: Proc. SPIE conf. On Visual Communications and Image Processing, San Jose, CA, Jan. 2000

20 Vaishampayan V A. Design of multiple description scalar quantizers. IEEE Trans. Info. Theo. 1993, 39：821～834

21 Chung D-M, Wang Y. Multiple description image coding using signal decomposition and reconstruction based on lapped orthogonal transforms. IEEE Trans. on Circuits and Systems for Video Technology, 1999, 9(6)：895～908

22 Ingle A, Vaishampayan V A. DPCM system design for diversity systems with applications to packetized speech. IEEE Trans. Speech and Audio Processing. 1995, 3：48～57

23 Goyal V K, Kovacevic J, Arean R, Vetterli M. Multiple description transform coding of images. IEEE Int. Conf. Image Proc. (ICIP98), Chicago, 1998, 1：674～678

24 Wang Y, Orchard M, Reibman A. Optimal pairwise correlating transforms for multiple description coding. IEEE Int. Conf. Image Proc. (ICIP98), Chicago, 1998, 1：679～683