多级安全 DBMS 的通用审计策略模型*

何永忠[1,2+], 李 斓[3], 冯登国[1]

[1](信息安全国家重点实验室(中国科学院 软件研究所),北京 100080)

[2](中国科学院 研究生院,北京 100049)

[3](上海交通大学 信息安全工程学院,上海 200030)

A Generic Audit Policy Model on Multilevel Secure DBMS

HE Yong-Zhong[1,2+], LI Lan[3], FENG Deng-Guo[1]

[1](State Key Laboratory of Information Security (Institute of Software, The Chinese Academy of Sciences), Beijing 100080, China)

[2](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

[3](School of Information Security, Shanghai Jiaotong University, Shanghai 200030, China)

+ Corresponding author: Phn: +86-10-62528254 ext 803, E-mail: yzhe@is.iscas.ac.cn

Received 2004-07-27; Accepted 2004-10-10

He YZ, Li L, Feng DG. A generic audit policy model on multilevel secure DBMS. *Journal of Software*, 2005,16(10):1774−1783. DOI: 10.1360/jos161774

Abstract: This paper proposes a generic audit policy model on multilevel secure DBMS. The model is powerful expressively which not only expresses audit policy based on periodical time constraints, but also implements audit policy deduction based on rules. Furthermore, fine-grained audit policies are possible in this model with the introduction of object attribute predicate. The decidability of the model is proven and a decidability algorithm is presented.

Key words: audit; policy model; multilevel secure DBMS

摘 要: 提出了一种基于多级安全数据库管理系统的通用审计策略模型.该模型具有丰富的表达能力,既可以表达基于时间的审计策略,也可以实现基于规则的审计策略推衍.通过引入对象的属性谓词,还可以表达细粒度的审计策略.证明了该模型的可判定性,并给出了判定任意一个事件是否需要审计的算法.

关键词: 审计;策略模型;多级安全数据库

中图法分类号: TP309 文献标识码: A

随着计算机以及网络的普及和广泛应用,信息系统的安全问题也变得越来越严重.虽然有很多安全技术,如

* Supported by the National Natural Science Foundation of China under Grant Nos.60025205, 60373048, 90304007 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2004AA147070 (国家高技术研究发展计划(863))

作者简介: 何永忠(1969－),男,重庆人,博士生,主要研究领域为密码学,系统安全;李斓(1977－),男,博士,主要研究领域为系统安全;冯登国(1965－),男,博士,研究员,博士生导师,主要研究领域为网络与信息安全.

加密、访问控制、入侵检测等可以用来应对各种安全威胁,但是这些技术并不能完全保障系统的安全.事实上,在系统实际运行中,安全威胁中很大一部分都源于内部人员攻击,而入侵检查和访问控制等机制对这类攻击的防范能力非常有限;另一方面,对于很多外部入侵事件,入侵检测工具不能作出正确的响应.在这些情况下,作为安全事件追踪分析和责任追究的审计机制有着不可替代的作用.审计机制在多级安全数据库中更为重要.因为在支持多级安全的数据库系统中,隐通道可以绕过强制安全策略的限制,可能构成对系统的严重安全威胁.有些隐通道很难在系统实现中完全消除,对这些隐通道一般可以采用审计的方法[1].通过对构成隐通道场景中操作序列进行审计,可以威慑内部人员利用隐通道进行的非授权通信,也可以在事后检查是否存在恶意代码的攻击.因此,对多级数据库设计灵活的审计机制更加必要.

对于审计子系统来说,一个简单的审计策略是,对数据库系统中发生的所有事件都审计,可以完全满足安全分析和责任追踪的需求.然而,这样将大大降低系统的时间效率和空间效率,并且记录大量无用的事件信息.因此,对系统中所有操作事件都审计是不现实的,也是不必要的.审计系统应该具有配置审计事件的能力,灵活、有效的审计配置功能可以使得审计日志在尽量少占用时间和空间的前提下,为安全事件分析和责任追究提供足够多的信息.然而,已有的商业化数据库系统对灵活的审计设置的支持非常有限.因此,研究并提出具有灵活结构、表达能力丰富、形式化的审计策略模型对多级安全系统的设计和开发都有着重要的意义.

针对目前实际系统对审计支持的局限性,本文提出的审计策略模型主要解决以下几方面的问题:

首先,对于大型的数据库管理系统,由于包括了大量的数据对象和用户,需要支持多个审计管理员进行分布式管理,如何实施全局的审计策略设置以及如何协调各个审计管理员之间的关系是值得研究的.

其次,需要研究如何为审计管理提供灵活的基于时间的审计.这种功能是很有必要的.比如,审计策略要求在下班时间或者某些重要的时段发生的事件需要更全面和详细的审计,这时就可以使用支持时间约束的审计规则.否则,需要审计管理员在不同的时段重新配置审计策略,给管理带来很大的不便.

最后是细粒度审计设置的研究.在一些数据库中,针对元组的细粒度审计一般是通过触发器机制实现的.这种方式不利于审计管理员进行全局的一致性的审计策略管理.在本文提出的模型中,我们通过引入审计对象属性谓词来实现灵活的细粒度的审计策略配置.

需要指出的是,在多级安全的数据库中,一方面要求审计子系统应该支持对隐通道的审计,另一方面,审计子系统本身在设计时也应该尽量避免引入新的隐通道.本模型提出的细粒度审计策略设置可以用来辅助对隐通道的审计.本模型还规定审计策略的安全级别满足一定的性质以防止引入新的隐通道.

本文第 1 节详细描述基于多级安全数据库管理系统的通用审计策略模型的框架结构.第 2 节讨论审计策略模型的核心结构审计策略规则库的组成以及策略表达语言.第 3 节给出保证模型安全性必须满足的不变量.第 4 节讨论审计策略规则库的可判定性问题并给出了判定算法.第 5 节介绍与本文相关的研究工作.第 6 节总结全文.

1 多级审计策略模型基本框架

多级安全数据库是实施多级强制安全策略(比如 Bell LaPadula 模型[2])的数据库管理系统.从抽象的模型层次来看,数据库与操作系统的不同之处主要是数据库中引入了事务以及数据模型,因此,在审计策略模型中包括了事务类型和会话类型.我们的审计策略模型中对客体对象抽象为树状结构,与具体数据模型无关,因此既可以用于关系数据模型,也可以用于对象数据模型以及半结构化数据模型,如 XML 文档数据库.多级安全的数据库中所有的主客体都有相应的安全级别标签,并且树状结构的客体对象的安全级别满足从树根到叶子的升序关系.这样,根据 BLP 模型的简单安全特性(任何用户只允许读该用户的安全级别支配的客体),任何级别的用户看到的对象视图也构成一棵树.用户的安全标签可以是 TRUSTED,表明该用户是可信的,可以违反强制安全策略.引入可信主体的目的是为了方便管理员的操作.用户也属于一种特殊的客体对象,因为用户属性修改、删除用户等操作的对象就是用户名.审计策略模型的核心是审计策略规则库,实施审计的子系统将根据审计策略规则库决定是否对一个事件审计.下面给出多级审计策略模型的定义.

定义 **1**(多级审计策略模型框架). 多级审计策略模型 $MAP=(SU,O,A,L,APB,Constrants)$,其中:$SU=(U,S,T,M)$

为主体类型,包括 $U=\{u_1,u_2,u_3,...\}$ 是用户的集合; $M \subseteq U$ 是拥有审计特权的用户集合;$S=\{s_1,s_2,s_3,\ldots\}$是会话的集合; $T=\{t_1,t_2,t_3,...\}$ 是事务的集合. $O=\{o_1,o_2,o_3,...\}$ 是对象的集合; $A=\{op_1,op_2,op_3,...\}$ 是操作的集合; L 是安全标签的集合;APB 为审计策略规则库; $F \subset A$ 是对审计策略规则库的操作.以上所有的集合都是有限集合.Constrants 包括以下关系:

$LR \subseteq L \times L$ 是定义在 L 上的一个格关系;

$U \subseteq O$,该关系规定用户也是一个对象;

$ULabel: U \to L \bigcup \{\text{TRUSTED}\}$,其中 TRUSTED 表示可信主体.该关系表示每个用户都对应一个安全标签;

$OLabel: O-U \to L$,每个对象客体都对应一个安全标签;

$SU: S \to U$,每个会话对应一个用户;

$TS: T \to S$,每个事务对应一个会话,注意同一会话可以对应多个事务.

$H(O)$,对象层次树形结构,有唯一一个的树根 o_1 .如果 o_i 是 o_j 的父节点(用谓词 $DP(o_i,o_j)$ 表示),则 $OLabel(o_j) \geq OLabel(o_j)$.

2 审计策略规则库

审计策略规则库是审计策略模型的核心部分,根据审计策略规则库可以确定对任何一个事件是否应该审计以及审计的频度.实施审计的子系统将根据上述判定结果实施审计或者不审计.本节我们将依次给出审计策略规则库各个组成部分的定义.

由于本模型支持时间约束的审计策略,我们先给出时间约束的定义.文献[3]引入了周期时间的概念,由于该定义直观上易于理解,并且能够表达常见的时间约束,因此本模型也采用这种时间描述形式.

定义 **2**(区间周期时间)[3]. 我们考虑线性序离散时间 $T=\{t_0,t_1,t_2,\ldots\}$,每个元素称为时刻.区间周期时间 $([t_s,t_e],PT)$ 表示在时间区间内 $[t_s,t_e]$ 且属于周期时间 PT 的所有时刻的集合.PT 是基于日历 C 来定义的.日历定义为可数个连续时间区间的集合, $C_1 \subseteq C_2$ 当且仅当 C_1 的每一个时间区间都被 C_2 中有限个时间区间集合包含(同样可定义区间周期时间的包含关系). $PT=\sum_{i=1}^{n} O_i \cdot C_i \triangleright r \cdot C_d$.其中

$$O_i \in 2^N \bigcup \{all\},\ C_i \subseteq C_{i-1}, i=1,2,...,n,\ C_d \subseteq C_n .r,n \in \mathrm{N}.$$

N 是自然数集合.PT 对应的时间区间集合 $\Pi(PT)$为区间长度为 $r \cdot |C_d|$ ($|C_d|$ 表示日历 C_d 一个时间区间的长度),开始时刻为下面定义集合 S 中的一个时刻的所有时间区间.(1) 如果 n=1,S 包含日历 C_1 中所有的开始时刻.(2) 如果 n>1, $O_n=\{n_1,...,n_k\}$,则 S 包含的时刻是日历 C_n 的第 $n_1,...,n_k$ 个时间区间的开始时刻构成的集合与 $\Pi(\sum_{i=1}^{n-1} O_i \cdot C_i \triangleright 1 \cdot C_{n-1})$ 的交集.如果 $O_n=\{all\}$,则 $O_n=\{1,2,3,...\}$.

定义 **3**(区间周期时间的关系)[3]. 区间周期时间对应的时刻集合定义为

$$\Xi([t_s,t_e],PT)=\{\tau \mid t_s \leq \tau \leq t_e, it \in \Pi(P), \tau \in it\}.$$

两个区间周期时间 $p_1 \subseteq p_2$ 当且仅当 $\Xi(p_1) \subseteq \Xi(p_2)$.

在审计策略中,我们可以根据一个操作的执行结果决定是否进行审计.常用的操作执行结果有操作成功和操作失败.为了进一步控制对某些类型操作结果的审计,我们还定义了自主访问控制不允许导致的操作失败、强制访问控制不允许导致的失败,以及在系统禁止多实例时用户企图创建多实例对象导致的失败.通过对操作失败原因的细分,可以方便地按照违反安全策略的类型对操作进行审计策略定义.

定义 **4**(操作执行结果 ***RES***). 操作执行结果是操作执行是否成功或者失败以及失败类型.操作执行结果集合 $RES=\{r_1,r_2,r_3,...\}$,并且基本操作执行结果集合 $\{\text{SUCCESSFUL},\text{UNSUCCESSFUL},\text{BOTH}\} \subset RES$,以及操作失败类型集合 $FRES=\{\text{EDAC},\text{EMAC},\text{EPOL},...\} \subset RES$.其中 SUCCESSFUL 表示操作成功执行,UNSUCCESSFUL 表示操作失败,BOTH 表示操作成功或者失败,EDAC 表示自主访问失败,EMAC 表示强制访问失败,EPOL 表示多实例失败(当系统禁止多实例时创建导致多实例的对象).除了基本操作执行结果以外,RES 还可以包括其他类型的操作执行结果.

在一次会话中,同一个用户对同一个对象进行同一个操作,而且操作结果也相同,那么审计策略可以规定审

计次数(审计频度),即每次访问都进行审计或者整个会话只审计一次.由于在数据库系统中一个会话可以相对于多个事务,因此审计策略还有一个选择是:规定每个事务审计一次.审计频度的引入可以帮助审计管理员在配置审计策略时,在审计记录信息的充分性和审计效率之间进行平衡.

定义 5(审计频度 ***FREQ***). 审计频度是相同类型的事件(用户、操作、操作对象、操作结果相同)在一个系统运行阶段中审计的次数.集合 *FREQ* 定义为 $FREQ=\{\text{SESSION},\text{TRANSACTION},\text{ACCESS}\}$,其中 SESSION 表示审计记录等于会话频度,即在一个会话中同样的访问事件只记录一次.TRANSACTION 表示事务频度,ACCESS 表示访问频度.

在数据库系统中,对象客体有很多属性,比如对象的名称、类型、宿主等.对于元组对象,还包括元组中元素属性的值.通过引入对象属性函数,就可以基于对象属性进行审计策略配置,提供灵活的细粒度的审计策略规则.比如,通过元组属性,可以控制对元组级访问的审计.

定义 6(对象属性函数 ***ATTR***). 对象的属性函数集合 $ATTR=\{at_1,at_2,at_3,...\}$,并且定义 $\{Name,Label,Type,Owner,Elem\}\subseteq ATTR$ 为基本的对象属性函数.注意这些函数都是偏函数,可在某些对象上没有定义.其中 $Name:O\to STRING$,把对象映射为对象的名称,*STRING* 为字符串类型; $Label:O\to L$,返回对象的标签, $Type:O\to TYPE$,返回对象的类型; $Owner:O\to U$,返回对象的宿主; $Elem:O\times ELEMENT\to VALUE$,返回元组对象的元素值,其中 *ELEMENT* 是元组属性名集合,*VALUE* 定义为值集合,是最基本的抽象数据类型,模型中其他类型都是它的子类型,如字符串类型、对象类别类型、用户类型等.

在前面定义的基础上,我们可以定义基本的审计策略项.

定义 7(审计策略项). 审计策略项是九元组 (a,d,q,u,p,f,r,s,m),其中 $a\in A\cup\{\text{ALLACT}\}$ 是需要审计的操作,ALLACT 表示所有的操作; $d\in O$ 表示审计对象所在的范畴,所谓审计范畴是指被审计的对象都在以对象 d 为根的子树中; q 是关于对象的谓词,一般形式是 $at_{i1}(o,...)=v_{i1}\wedge...\wedge at_{ij}(o,...)=v_{ik}$, $v_{i1},\dots,v_{ij}\in VALUE$; $u\in U\cup\{\text{ALLUSER}\}$,ALLUSER 表示所有的用户; p 是区间周期时间约束, $f\in FREQ$, $r\in RES$, $s=\{+,-\}$,表示审计或者禁止审计. $m\in U\cup\{\text{SYS}\}$ 表示设置该审计策略规则的用户,当审计设置用户为 SYS 时,表示由系统设置的强制审计项.注意,审计策略项不能包含变量.

例 1(审计策略项):

(SELECTTABLE,DB1,*Name*(*o*)=$T_1\wedge$*Type*(o)=TABLE,ALLUSER,
⟨[2004,2006],all.Months+all.Days+{18}.Hours ▷ 15.Hours⟩,
TRANSACTION,BOTH,+,ALICE).

该审计策略表示,对于操作 SELECTTABLE,操作对象是在数据库 DB_1 中的类型为表的对象 T_1,所有的用户,时间从 2004 年~2006 年每天的 18 点~第 2 天的 9 点,无论操作结果如何,在一个事务中审计一次.该审计策略是 ALICE 设置的.

下面定义审计策略闭项.与审计策略项相比,审计对象范畴和审计对象谓词两个元素被确定的审计对象取代,周期时间约束为具体时刻.因为审计策略闭项与系统事件(将在后面定义)形式上更为接近,所以,根据审计策略闭项可以方便地判定一个事件是否需要审计.

定义 8(审计策略闭项). 审计策略闭项定义为八元组 (a,o,u,τ,f,r,s,m),其中 τ 为时刻,其他都是常量,与审计策略项定义相同.

不同的审计策略项可能会表示相同的审计策略.下面定义的审计策略公理就是对审计策略项之间的蕴涵关系的形式化规定,用户定义的任何审计策略库都包含这些规则.

定义 9(基本审计策略推衍规则). 推衍规则的一般形式是 $aud_1,aud_2,...,aud_k\xrightarrow{c}aud_{k+1}$,推衍规则符号 $\xrightarrow{c}$ 前面的审计策略项是前提,后面是推衍结果.推衍符号上的 c 是推衍的约束条件,当 $c=\tau$ 时表示没有约束.注意,推衍规则符号 $\xrightarrow{c}$ 和逻辑蕴涵符号 $\to$ 的区别,前者一定会出现推衍约束条件.

(1) 否定推衍规则

$$(a,d,q,u,p,f,r,+,m)\xrightarrow{\tau}\neg(a,d,q,u,p,f,r,-,m),$$
$$(a,d,q,u,p,f,r,-,m)\xrightarrow{\tau}\neg(a,d,q,u,p,f,r,+,m).$$

否定推衍规则的含义是,审计策略项的非就是审计符号 s 取反.下面规则的含义都非常清楚,这里不再一一说明.

(2) 关于 ALLACT,ALLUSER 的推衍规则

$$(\text{ALLACT},d,q,u,p,f,r,s,m)\xrightarrow{a\in A}(a,d,q,u,p,f,r,s,m),$$

$$(a,d,\text{ALLUSER},q,p,f,r,s,m)\xrightarrow{u\in U}(a,d,q,u,p,f,r,s,m).$$

(3) 关于客体层次的推衍规则

$$(a,d,q,u,p,f,r,s,m)\xrightarrow{d'\in O\wedge P(d,d')}(a,d',q,u,p,f,r,s,m)\ .$$

其中谓词 $P(o_i,o_j)$ 表示 o_i 是 o_j 的祖先父节点. $P(o_i,o_j)$ 可由对象的父子关系谓词 DP 来确定.

(4) 关于时间约束的推衍规则

$$(a,d,q,u,p,f,r,s,m)\xrightarrow{p'\subseteq p}(a,d,q,u,p',f,r,s,m)\ .$$

(5) 关于对象属性谓词的推衍规则

$$(a,d,q,u,p,f,r,s,m)\xrightarrow{(q'\to q)}(a,d,q',u,p,f,r,s,m)\ .$$

注意,该规则与其他规则的不同之处在于,前提的要求是 $q'\to q$,而不是 $q\to q'$ (注意, $q\to q'$ 是通常的逻辑蕴涵,与审计策略推衍符号不同),实质上,这是因为我们要求该规则推出的审计策略项 (a,d,q',u,p,f,r,s,m) 所审计的对象(在范畴 d 中,且满足对象属性谓词 q' 的所有对象)是前提中的审计策略项的审计对象集合的子集.为了推导简便,规定,

$$at_{i1}(o,...)=v_{i1}\wedge...\wedge at_{ik}(o,...)=v_{ik}\to at_{j1}(o,...)=v_{j1}\wedge...\wedge at_{jr}(o,...)=v_{jr}\ .$$

当且仅当 $\forall l:1,...,r\cdot\exists l'\cdot at_{jl}=at_{il'}\wedge v_{jl}=v_{il'}$.

(6) 关于执行结果 RES 的推衍规则

$$(a,d,q,u,p,f,\text{BOTH},s,m)\xrightarrow{\tau}(a,d,q,u,p,f,\text{SUCCESSFUL},s,m),$$

$$(a,d,q,p,f,\text{BOTH},s,m)\xrightarrow{\tau}(a,d,q,u,p,f,\text{UNSUCCESSFUL},s,m).$$

以及

$$(a,d,q,u,p,f,\text{UNSUCCESSFUL},s,m)\xrightarrow{r\in FRES}(a,d,q,u,p,f,r,s,m)\ .$$

(7) 审计策略闭项推衍规则

$$(a,d,q,u,p,f,r,s,m)\xrightarrow{cond}(a,o,u,\tau,f,r,s)\ .$$

其中 $cond$ 为

$$(d=o\vee P(d,o))\wedge q(o)\wedge\tau\in\Xi(p)$$

$$\wedge ULabel(m)<>\text{TRUSTED}\to Label(o)=ULabel(m)\ .$$

该条件中的 $(d=o\vee P(d,o))\wedge q(o)\wedge\tau\in\Xi(p)$ 含义是审计对象 o 必须是 d 或者 d 的孩子,并且满足谓词 q,而时刻 τ 是区间周期时间 p 所对应的时刻集合的元素.条件的后一部分的含义是非可信的审计管理员只能对本级别的对象 o 审计,可信的审计管理员可以对任何级别的对象进行审计.

除了不同审计策略库都包含的基本审计策略推衍规则以外,用户还可以根据需要定义其他的审计策略推衍规则.引入这些规则可以使得审计管理员通过少量的定义生成大量的审计策略,从而方便审计管理员定义和维护审计策略,减轻审计管理员的负担.

定义 **10**(扩展审计策略推衍规则). 假设 $aud_1,aud_2,...,aud_k$ 以及 aud 都是审计策略项或者审计策略项的非 $\neg aud_i$,那么扩展审计策略推衍规则的一般形式是

$$aud_1,aud_2,...,aud_k\xrightarrow{\tau}aud\ .$$

如同审计策略公理一样,在审计策略衍生公式中,审计策略项的分量可以是变量,也可以是常量.在 aud 中分量要么是前提中出现的变量,要么是常量(比如具体的用户,对象,操作,或者关于属性的谓词表达式).简便起见,规定在蕴涵符号后边的审计策略项不包括“非”,这不影响衍生公式的表达能力.

定义 2~定义 10 是审计策略规则库的所有组成部分的定义,下面给出审计策略库的完整组成定义.

定义 **11**(审计策略规则库 ***APB***). 审计策略规则库由多个审计策略项(称为初始审计策略项),基本审计策略

推衍规则和扩展审计策略推衍规则组成.其中审计策略项和扩展审计策略推衍规则可以根据需要来定义,而 *APB* 中必须包含全部基本审计策略推衍规则.

3 审计策略模型不变量

本节我们给出审计策略模型应该满足的安全性质(或者称为模型不变量).

因为我们的审计策略模型是基于多级数据库管理系统,所有的审计策略项或者审计策略衍生规则也是有安全级别的,为了避免隐通道,我们规定了审计策略的安全级别应该满足的性质.通过信息流分析方法可以证明满足下列性质的审计策略模型没有隐通道.

性质 **1**. 审计策略(包括审计策略项和扩展审计策略规则)的安全级别等于设置审计的用户策略的安全级别.对于可信审计管理员设置的审计策略,审计策略的级别设置为最高的安全级别.假设 *ap* 是用户 *u* 设置的审计策略,则:

$$ULabel(u)=\text{TRUSTED}\rightarrow OLabel(ap)=\text{SystemHigh},$$
$$ULabel(u)\langle\rangle\text{TRUSTED}\rightarrow OLabel(ap)=ULabel(u).$$

其中 SystemHigh 是系统的最高安全级,支配所有其他安全级别.

性质 **2**. 审计策略中审计管理员参数 *m* 与审计策略的设置者相同.假设 *ap* 是用户 *u* 设置的审计策略,则:

$$Auditor(ap)=u\ .$$

其中 *Auditor*(*ap*)是审计策略 *ap* 对应的审计管理员参量 *m*,因此在用户设置审计策略时,审计管理员的值就确定了,不会存在变量.

根据强制安全策略,只有用户的级别支配对象级别时,才可以读访问该对象.下面的性质要求非可信的审计管理员的级别应该支配审计策略中的范畴 *d* 的级别.

性质 **3**. 如果是非可信的审计管理员,其设置的审计策略项 *ap* 中的范畴 *d* 的级别应该被审计管理员的级别支配.

$$ULabel(Auditor(ap))\langle\rangle\text{TRUSTED}\rightarrow ULabel(Auditor(ap))\geq Label(Domain(ap)).$$

其中 *Domain*(*ap*)是审计策略项 *ap* 的范畴.

4 *APB* 的可判定性

在讨论 *APB* 的可判定性之前,我们先定义什么是审计策略项的推衍.

定义 **12**(审计策略项的推衍).

我们把审计策略规则库“推衍”出一个审计策略项(或者闭项)记为: $APB\mapsto(a,d,q,u,\tau,f,r,s,m)$ 或者 $APB\mapsto(a,o,u,\tau,f,r,s)$.审计策略项推衍关系是下面的递归关系形成的最小关系:

(1) 如果该审计策略项是 *APB* 中的初始审计策略项,则 $APB\mapsto(a,d,q,u,\tau,f,r,s,m)$.

(2) 如果一个推衍规则 $aud_1,aud_2,...,aud_k\xrightarrow{c}aud_{k+1}$ 前提中的审计策略项 $aud_1,aud_2,...,aud_k$ 都可以由 *APB* 推衍,且该推衍规则的约束条件 *c* 满足,同时,当模型的 3 个不变量也满足时,该推衍规则结论中的审计策略项也是可以推衍的,即 $APB\mapsto aud_{k+1}$.

从 *APB* 推衍某个审计策略项的过程可以形成一棵推衍树.推衍树的树根是目标推导审计策略项,孩子节点对应推衍规则前提的一个审计策略项,父节点对应推衍规则的结论审计策略项.对于审计策略规则库 *APB*,一个重要的问题是,是否存在算法判定任何一个审计策略闭项能否可由审计策略库推导出来.我们知道,在一般的基于一阶谓词逻辑的系统中不存在这样的判定算法.对于本文定义的 *APB* 来说,我们将证明 *APB* 是可判定的.下面首先定义 *APB* 的可判定性,然后构造一个判定算法,并证明该算法的正确性,从而证明 *APB* 是可以判定的.

定义 **13**(***APB*** 的可判定性). 对于任意的审计策略闭项 (a,o,u,τ,f,r,s) ,如果存在有限步骤的算法确定 $APB\mapsto(a,o,u,\tau,f,r,s)$ 是否成立,则称 *APB* 是可判定的.

定义 **14**(审计策略项/闭项的匹配). 两个可能包含变量的审计策略项 (a,d,q,u,p,f,r,s,m) 和 $(a',d',q',u',p',f',r',s',m')$ (或者两者前面都有否定符 $\neg$)是匹配的,当且仅当对于它们中每一对参量,

$a,a';d,d';...;m,m'$,如果两者都是常量则必须是相同的;如果其中一个为常量,另一个为变量,则称该常量为变量的匹配常量.注意,在前面描述的推衍树中,已经推衍出的审计策略项如果与推衍规则前提中的审计策略项相匹配就可以推衍出新的审计策略项.因此,为了后面描述算法的需要,我们把匹配也作为一种特殊的推衍规则(类似地,有审计策略闭项的匹配,这里不再单独定义).

算法 **1**. 审计策略闭项判定算法 CAPD.

本算法对于一个给定的审计策略规则库 APB 和一个审计策略闭项 (a,o,u,τ,f,r,s,m),判定该审计策略闭项是否可以由 APB 推导出来,即 $APB\mapsto(a,o,u,\tau,f,r,s)$ 是否成立.

本算法的基本思想是,首先定义扩展审计策略项集合,其初始值包含所有的基本审计策略项;然后生成新的项并入扩展审计策略项集合,方法是判断每个扩展推衍规则的所有前提策略项是否可以由扩展审计策略项集合以及基本推衍规则推衍,如果可以,就把该扩展推衍规则生成的项加入扩展审计策略项中,重复该过程直到不能找到新的审计策略项;最后,判定扩展审计策略项和基本审计推衍规则是否可以推衍出目标的审计策略闭项.

第 1 步.生成扩展审计策略项集合.

1) 初始化扩展审计策略项集合 $Eap=\varnothing$.对所有的基本审计策略项 aud, $Eap=Eap\cup\{aud\}$.

2) 对一个扩展推衍规则 rule,它的每一个前提审计策略项 $aud_i(i=1,2,\ldots,k)$,调用下面定义的子例程 CompuSatItems(aud_i),计算可以从当前 Eap 和基本推衍规则推衍的且与 aud_i 匹配的全部审计策略项,记为 Items(aud_i).当每个 aud_i 对应的 Items(aud_i)都计算出来后,每次从式

$$\text{Items}(aud_1)\times\text{Items}(aud_2)\times\ldots\times\text{Items}(aud_k)$$

中取一个策略项 k 元组,与规则 rule 匹配,生成一个策略项并入 Eap 中,直到全部的 k 元组都处理完.

3) 对每一个扩展推衍规则 rule 都执行第 2)步.

4) 如果当前的 Eap 与第 2)、第 3)步执行前比较它的元素没有增加(我们把它称为完备 Eap),算法进入下面的第 2 步;否则转到 2)步继续循环执行.

第 2 步.判定审计策略闭项 (a,o,u,τ,f,r,s,m) 是否可由完备 Eap 和基本推衍规则推衍.本步结束,输出判定结果,整个算法结束.

注意,该步计算过程与子例程 CompuSatItems(aud_i)非常类似,不同的是只要找到一个匹配的项即可,所以算法可在 CompuSatItems 基础上进一步优化.具体的步骤不再详述.

子例程 CompuSatItems(aud).

本子例程计算可以从当前扩展审计策略项集合 Eap 和基本推衍规则所能推衍的并且与输入参数 aud 匹配的全部审计策略项,返回值就是这些审计策略项的集合(注意,aud 可能包含变量).算法的思想是宽度优先的次序扩展每一个节点,构造一棵逆向推衍树,初始化根节点为 aud.

1) 对已经构造的推衍树中每个未扩展节点 $node$(对应的审计项记为 aud_k),

1.1) 首先把 aud_k 依次匹配各个基本推衍规则的结论部分,如果与某推衍规则匹配,就用 aud_k 中的常量替换该推衍规则对应变量的所有出现(包括约束条件),然后把该推衍规则前提策略项添加为以 $node$ 为父节点的孩子节点.

1.2) 然后把 aud_k 与 Eap 中所有审计策略项 aud_j 匹配,如果匹配,则把 aud_j 生成为 $node$ 的孩子节点,并标记为真叶子节点.

2) 上述过程结束时,把 $node$ 标记为已扩展的节点.重复 1),采用同样的方法扩展所有未扩展的非叶子节点,生成他们的孩子节点.没有未扩展的非叶子节点时进入本子例程的第 3)步.

注意,本算法对扩展一个节点使用的推衍规则有一个要求,即从该节点的孩子节点到根的路径上,任意一条推衍规则只能使用一次.后面的算法正确性证明中我们将说明这个要求的合理性.

3) 当构造的推衍树中没有未扩展非叶子节点时,从根开始递归计算与各个节点匹配的合法审计策略项集合(所谓合法,就是可以从当前 Eap 和基本推衍规则推衍的不含有变量的审计策略项).每个节点匹配的合法审计策略项集合 Mi 的计算方法是,对它的每一个孩子节点匹配的合法审计策略项,如果满足对应的推衍规则约束条件,那么按照该规则生成的审计策略项就属于 Mi;如果孩子是叶子节点,则该叶子节点的策略项属于 Mi.

4) 当与根节点匹配的合法审计策略项集合计算完毕后就把该集合作为结果返回.

如果以后需要再次使用上面的算法判定一个审计策略闭项是否可推衍,那么第一次调用算法后就可以生成完备 *Eap*,之后判定时算法的第 1 步可以不用再执行了,这样就极大地提高了效率.

引理 **1**. 对任何 *APB* 和任意的审计策略闭项 (a,o,u,τ,f,r,s),算法 CAPD 执行有限的步后结束.

证明:因为算法的第 2 步与 CompuSatItems 类似,我们只需要分别证明子例程 CompuSatItems 是可以停机的,算法的第 1 步是可以停机的就可以了.

首先证明 CompuSatItems 是可以停机的.由算法可知,构造的推衍树的度不大于基本推衍规则数目和 *Eap* 元素个数之和,并且推衍树的深度不大于基本推衍规则个数之和加 1(因为一条路径上每条规则最多使用一次,加上匹配 *Eap* 审计策略项的匹配规则一次).这就表明推衍树是有限的.另外我们还需证明,对于任意约束谓词,当其中的变量都用常量替换后,它是可以判定的.推衍规则中的约束条件包括: $a\in A$, $u\in U$, $d'\in O\wedge P(d,d')$, $p'\subseteq p$, $q'\to q$, $r\in FREQ$, $(d=o\vee P(d,o))\wedge q(o)\wedge\tau\in\Xi(p)$ 以及 $ULabel(m)\langle\rangle\text{TRUSTED}\to Label(o)=ULabel(m)$. 显然,由于操作集合、对象集合等都是有限集合,所以,除了 $p'\subseteq p$, $\tau\in\Xi(p)$, $q'\to q$,其他约束是否满足都很容易根据系统状态来加以判定.而根据周期时间的定义, $p'\subseteq p$, $\tau\in\Xi(p)$ 也是可以判定的.对于 $q'\to q$,由前面关于对象属性谓词的推衍规则的说明中可以知道,它也是可以判定的.所以 CompuSatItems 是可以停机的.

其次证明算法 CAPD 的第 1 步是可以停机的.由于 CompuSatItems 每次返回的项集合都是有限的,且推衍规则的数目是有限的,所以易知 CAPD 的第 1 步中的第 2)步以及第 3)步可以在有限步内结束.第 4)步循环,当 *Eap* 不再增加新的元素时才会退出,因此,我们只需要证明 *Eap* 的元素个数存在一个最大的上界即可.由算法分析可知,对 *Eap* 中的任意项,其中的常量必然来自于初始策略项或者扩展推衍规则中已有的常量,所以 *Eap* 中不同项的数目最多等于审计策略项每一个参量对应的常量数目之乘积.所以 *Eap* 是有限的.

所以,算法 CAPD 执行有限的步后结束. □

由上面的引理可以很容易地估计算法的时间复杂度.

引理 **2**. 对于任何审计策略闭项,算法 CAPD 输出 true,当且仅当 $APB\mapsto(a,o,u,\tau,f,r,s)$.

证明:必要性是显然成立的.当算法输出为 true 时,可以根据算法,构造一棵从叶子节点为初始审计策略项到根节点为审计策略闭项的推衍树,所以 $APB\mapsto(a,o,u,\tau,f,r,s)$.

充分性.当 $APB\mapsto(a,o,u,\tau,f,r,s)$ 成立时,必定存在一棵推衍树 *tree*,树根为 (a,o,u,τ,f,r,s),叶子为初始审计策略项(显然,该推衍树所有节点的审计策略项都不包含变量).首先证明,对于任意一个 *Eap*,如果使用基本推衍规则可推衍的审计策略项 *aud*,子例程 CompuSatItems(*aud*)也可以找到相同的项.由于基本推衍规则只有一个前提、一个结论,所以推衍一个审计策略项的推衍树退化为一个推衍线性序列.我们只需证明,对于任意的推衍序列,可以等价变化为每条推衍规则只出现一次,并且变换后的序列最终推衍出同样的策略项 *aud*.由于 CompuSatItems(*aud*)构造的推衍树必然有一条路径与变换后的推衍序列匹配,所以子例程 CompuSatItems(*aud*)也可以找到相同的项.我们注意到,因为审计策略闭项规则的结论是八元组的策略闭项,所以,如果推衍序列中有审计策略闭项推衍规则,则它必然只有一条,而且是最后一条规则.又因为其他基本推衍规则都是对某一个参量的变换,所以可以互相交换位置,不会影响最终的推衍结果.比如,推导序列

$$(a,d,...)\mapsto_{\text{rule1}}(a',d,...)\mapsto_{\text{rule2}}(a',d',...).$$

将 rule1 和 rule2 互换,得到

$$(a,d,...)\mapsto_{\text{rule2}}(a,d',...)\mapsto_{\text{rule1}}(a',d',...).$$

推衍的结果是一样的.这样,我们可以把同一推衍规则的不同出现交换集中在一起.考察每一条基本推衍规则,只有关于对象层次、时间约束和对象属性谓词的推衍规则可能出现两次以上,对每条重复的规则只保留头尾两个审计策略项,它们正好用一条审计推衍规则可以进行推衍.这样得到的推衍序列每条基本推衍规则最多出现一次.

然后,可以证明,*tree* 中每一个由扩展推衍规则生成的策略项 *aud* 都属于算法 CAPD 第 1 步执行完后的最大 *Eap*.可用数学归纳法来证明.对于 *tree* 中每一个由扩展推衍规则生成的策略项 *aud*,按照从它到叶子的所有路径中最多使用了 *i* 个扩展推衍规则来进行分类.对 *i* 进行归纳.若 *i*=1,则 *aud* 的前提项到叶子只经过 0 个扩展推衍

规则,这样,其前提项都可以由初始审计策略项和基本推衍规则来推衍,根据前面的结论,其前提项都可由子例程 CompuSatItems 找到,所以 *aud* 属于 *Eap*.假设 $i \le k$ 的 *aud* 都属于 *Eap*,则推衍树上 $k+1$ 的 *aud* 的前提项到叶子方向的所有推衍路径上,不经过任何扩展推衍规则都可以找到属于 *Eap* 的审计策略项.所以可以由 *Eap* 和基本推衍规则推衍.这就证明了 *tree* 中每一个由扩展推衍规则生成的策略项 *aud* 都属于算法 CAPD 第 1 步执行完后的完全 *Eap*.所以,对于目标审计策略闭项 (a,o,u,τ,f,r,s),它到叶子方向的任何推衍路径上,不经过任何扩展推衍规则都可以找到属于 *Eap* 的审计策略项,所以它可以由子例程 CompuSatItems 找到.这样,算法 CAPD 的输出为 true. □

定理 **1**. *APB* 是可判定的.

根据引理 1,2,显然 *APB* 是可判定的.

5 相关工作

审计是保障信息系统安全的重要技术.关于审计的已有研究主要集中在审计系统的体系结构、审计记录的逻辑和物理储存方式[4−7]等方面,或者研究如何通过审计记录来发现入侵行为以及内部人员权限的滥用问题[6,7].就目前公开的研究资料来看,还很缺乏关于如何配置审计方面的研究.一方面,表达能力强大的审计策略语言有现实的需求;另一方面,对于多级数据库来说,其中的客体对象和操作类型复杂,还涉及到多级安全和隐通道问题,因此,建立形式化的审计策略模型对开发多级安全数据库管理系统是非常重要的.与本文相关的研究主要包括多级安全数据库的研究、安全策略模型的研究,特别是访问控制策略的研究.

多级强制安全策略多采用 Bell LaPudula 模型,已有的多级安全 DBMS[8,9]也大多基于该模型.本文的模型也是建立在采用 BLP 模型的 DBMS 之上的.本文引用的多级 DBMS 系统数据模型采用的是抽象的对象层次结构模型,没有对 DBMS 中的具体对象,如表、视图、元组等进行进一步的规定和描述.原因有两个:一是考虑到模型的通用性,较高的抽象层次描述模型可以使其能适应各种具体的数据模型,如对象的或者关系的数据模型;另一个原因是,就我们要建立的审计策略模型来说,这种抽象的对象层次模型已经足够刻画该审计策略模型的主要特性.

审计策略模型一般不作为安全策略模型的一部分[10].因此,在已有的系统设计中,都没有为审计策略建立模型.然而,从我们的实际设计开发来看,建立审计策略模型是非常必要的.审计策略模型在内容和形式上与安全策略模型特别是访问控制策略模型有很多相似之处.因此,本文的研究借鉴了很多关于自主访问控制模型的研究成果.本文采用的基于规则的方法借鉴了 Jajodia 等人[11]提出的基于逻辑语言的授权的思想,它使得访问控制策略与访问控制策略模型独立开来,这样就可以在同一个策略模型中表达不同类型的访问控制策略.本文的周期时间约束来源于 Bertino 等人[3,12]在访问控制模型中支持周期时间的授权及其时态推理的思想.

本文的审计策略模型与 Bertino 的支持时间约束的访问控制模型有以下几方面的不同:首先,在我们的模型中,对象采用了范畴和对象的属性谓词来描述,具有更大的灵活性.需要注意的是,如果还把审计策略项作为谓词,则相应的推导系统就是高阶谓词逻辑.本文通过引入基于推衍规则来推衍审计策略项的方法避免了这个问题.其次,审计策略项中包含了审计频度、执行结果等与审计策略相关的属性,这样导致审计策略规则的一致性和完备性与访问控制策略有很大的不同.

6 结 论

本文提出了一种基于多级安全数据库的灵活、通用的审计策略模型.模型中提出了审计对象的层次范畴结构以及审计禁止规则的方法来应对和解决分布式审计管理问题.该模型具有较强的表达能力,可以表达基于时间的审计策略,也可以实现基于规则的审计策略推衍.通过引入对象的属性谓词,还可以表达细粒度的审计策略.通过时间约束和细粒度的审计策略的支持,可以方便地设置对潜在隐通道的审计.本文定义的审计模型不变量可以保证审计策略模型本身不会引入隐通道.本文还证明了审计策略规则库的可判定性质,并给出了判定一个事件是否需要审计的算法.虽然本文提出的审计策略模型是基于多级安全数据库的,但是适当的修改也可以

用于多级安全的操作系统或者其他应用系统,因此具有较大的适应性.

我们下一步的工作是进一步放宽对扩展审计策略推衍规则的形式限制,以便表达更为复杂、通用的策略推衍机制,还需要研究审计策略模型的语义以及审计策略规则库的动态更新算法等.

References:

[1] National Computer Security Center. A guide to understanding covert channel analysis of trusted systems. Technical Report, NCSC-TG-030, National Computer Security Center, 1993.

[2] DE BL, LaPadula LJ. Secure computer systems: Unified exposition and multics interpretation. Technical Report, MTR-2997, Bedford: MITRE Corporation, 1976.

[3] Bertino E, Bettini C, Ferrari E, Samarati P. A temporal access control mechanism for database systems. IEEE Trans. on Knowledge and Data Engineering, 1996,8(1):67−80.

[4] Wee C.LAFS: A logging and auditing file system. In: Proc. of the 11th Annual Computer Security Applications Conf. Los Alamitos: IEEE Computer Society Press, 1995. 231−240.

[5] Bishop M. A standard audit trail format. In: Proc. of the 18th National Information Systems Security Conf. Washington DC: National Computer Security Center, 1995. 136−145.

[6] Helman P, Liepins G. Statistical foundations of audit trail analysis for the detection of computer misuse. IEEE Trans. on Software Engineering, 1993,19(9):886−901.

[7] Biskup J, Flegel U. Transaction-Based pseudonyms in audit data for privacy respecting intrusion detection. LNCS 1907, Berlin: Springer-Verlag, 2000. 28−48.

[8] Sandhu R, Chen F. The multilevel relational (MLR) data model. ACM Trans. on Information and System Security, 1998,1(1): 93−132.

[9] Lunt TF, Denning DE, Schell RR, Heckman M, Shockley WR. The SeaView security model. IEEE Trans. on Software Engineering, 1990,16(6):593−607.

[10] National Computer Security Center. A guide to understanding security modeling in trusted systems. Technical Report, NCSC-TG-010, National Computer Security Center, 1992.

[11] Jajodia S, Samarati P, Subrahmanian VS. A logical language for expressing authorizations. In: Proc. of the 1997 IEEE Symp. on Security and Privacy. Los Alamitos: IEEE Computer Society Press, 1997. 31−42.

[12] Bertino E, Bettini C, Ferrari E, Samarati P. An access control model supporting periodicity constraints and temporal reasoning. ACM Trans. on Database Systems (TODS), 1998,23(3):231−285.