多安全策略集成性问题的分析与解决*)

吴新勇 熊光泽 桑 楠
(电子科技大学计算机科学与工程学院 成都610054)

摘 要 解决多安全策略的集成性问题是安全操作系统支持多策略和动态策略的基础。本论文采用形式化的方法为安全系统建立了全局安全状态的迁移模型,以TE和RBAC策略为例分析了不同策略作用下安全关联行为对安全状态的影响,并根据T&R集成模型提出了解决多策略集成性和一致性问题的思想,为安全操作系统的实现奠定了基础。

关键词 安全策略,安全操作系统,状态迁移模型,集成性

Analysis and Resolution of Integration of Multi Security Policies

WU Xin-Yong XIONG Guan-Ze SANG Nan
(Computer Science and Engineering College, UEST of China, Chengdu 610054)

Abstract Resolution of integration of multi security policies is the base problem in secure OS which supports multi security policies and dynamic policies. This paper uses a state transition approach to formally analyze a security system and presents the different effects resulted by security-depended actions of different policies on security states. We will analyze TE, RBAC model and T&R model (their integration model), present a thinking to tackle the hybrid integration and consistency of multi-policies, which lay the foundation for implementing secure OS.

Keywords Security policy, Secure OS, State transition model, Integration

1 引言

安全操作系统是保障信息系统良好运作的基石。经过30多年的发展,安全操作系统目前的研究重点集中在支持多策略和动态策略[1]。多策略要求系统支持当今流行的商用或军用安全策略模型;动态策略要求在多策略的基础上,系统可以在线升级策略,并可保持系统的一致性和连续性。由于策略之间的差异性,安全操作系统的研究必须解决多策略的可集成性问题。

虽然各种安全策略之间上有关联性,但安全内核在策略验证时必须分别满足多策略逻辑。多策略的集成涉及到系统设计的完备性和合理性,既兼顾所有策略的需求,又要使得系统开销最小。多策略的集成要求所有策略对特定系统是可实现的,即要求满足Schneider[2]提出了安全策略可实施的判定谓词:

$$P(\Pi):(\forall\ \sigma\in\Pi:\hat{P}(\sigma))$$

其中Π表示安全相关的行为集,σ表示行为序列,比如用户对对象实体的创建、删除、读、写和修改等,而$\hat{P}$是访问控制规则集,即策略的实施部分,可实施的策略要求所有的σ能够满足访问控制规则。行为集Π触发了系统状态的变迁,主要是访问权限的变化。采用形式化的方法分析安全关联操作引起的系统安全状态迁移模型有助于理解各种策略之间的逻辑关系,从而获得多策略集成的方案,它也是设计和实现安全核的必要前提,便于对安全关键系统的安全性进行分析和论证,特别是对安全策略和安全核的一致性的验证。本文利用形式化方法,定义了TE和RBAC策略作用下系统的安全状态集及状态迁移模型,分析了两种策略的可集成性,最后得出可实现的T&R集成模型,并且引申到其他的安全策略。对安全操作系统的实现及多策略的集成具有实践性指导意义。

本文第2节介绍了安全策略,定义了安全状态模型;第3、4节分别分析了TE和RBAC策略下的安全状态迁移模型;第5节就TE和RBAC集成模型T&R模型的可实现性展开讨论;第6节分析了多策略集成的支撑技术;最后给出了结论。

2 安全策略及安全状态模型

2.1 安全策略

安全策略是一套规则或约束集,用来管理一个组织如何管理、保护和发布敏感信息。在计算机系统里,我们关心的是

*)本文获十五国防预研项目资助。吴新勇 博士研究生,主要研究方向:系统安全及可靠性。熊光泽 教授,博士生导师,主要研究方向:实时计算机系统及软件开发支持。桑 楠 副教授,主要研究方向:高可靠实时软件。

7 Burns A, Wellings A. HRT-HOOD: A Structured Design Method for Hard Real-Time Ada Systems[M]. Amsterdam, Netherlands: Elsevier Science, 1995
8 Booch G. Object-Oriented Design With Applications [M]. Redwood City, CA, USA: Benjamin Cummings, 1991
9 Jacobson I, Christerson M, Jonsson P, Overgaard G. Object-oriented Software Engineering: A Use Case Driven Approach [M]. Boston, MA: Addison-Wesley, 1992
10 Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide[M]. Boston, MA: Addison-Wesley, 2001
11 Douglass B P. Real-Time UML: Developing Efficient Objects for Embedded Systems[M]. Boston, MA: Addison-Wesley, 2000
12 Frappier M, Habrias H. Software Specification Methods [M]. London: Springer, 2001
13 Derrick J, Boiten E. Refinement in Z and Object-Z[M]. London: Springer, 2001
14 Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software [M]. Boston, MA: Addison-Wesley, 1995

访问控制，所有的安全策略都可以看作一个访问控制矩阵(access control matrix)或其中一部分，矩阵的每一行是一系列的〈客体，权限集〉二元组，称为执行环境；每一列是一系列的〈责任者，权限集〉二元组，称为 ACL(Access Control List)。我们的安全策略模型主要以 TE 和 RBAC 为例，两者都从访问控制矩阵发展而来，是目前流行的策略模型。

(1)TE(Type Enforcement)模型[3]：TE 将主体和客体分别归并成类(Types)和域(Domains)，它们之间是否有访问权由 DDT(Domain Definition Table)和 DIT(Domain Interaction Table)决定，而 DDT 和 DIT 由安全管理员负责管理和维护。

(2)RBAC(Role-Based Access Control)[4]：RBAC 在用户和访问权限之间引入了角色(role)的概念，用户按需要与特定的一个或多个角色相联系，角色又与一个或多个访问权限相联系，角色可根据实际需要生成或取消，用户通过会话(session)激活角色，绑定权限，角色其实是 ACL 的载体。RBAC 的优点是策略无关性，角色的概念贴近于现实世界的信息管理系统。

2.2 安全状态模型的基本定义

一套访问控制机制由系统状态集组成，包括主体、客体、权限及三者的关系。用户行为集将改变这种关系使系统由一种安全状态迁移到另一种，在某一已知状态下，判定算法将对行为的合法性作出决策，拒绝可能违反安全的操作，保证系统处于安全状态。根据以上思想我们给出了一组定义：

·全局安全状态 Δ：是主体 S、客体 O、权限 R 和权限管理域的映射关系集，比如 TE 的权限管理域为 TE 初始化权限表和迁移权限表，RBAC 的管理域为 UA(用户角色分派)、RA(角色权限分派)等，但最终的映射是主体对客体的访问权限。

$\Delta: O\times R\to P(S)$ ($P(S)$为 S 的幂集)

·可能的行为集 Π：每个行为定义了全局状态的迁移映射关系：

$\gamma\in\Pi:\Delta\xrightarrow{r}\Delta'$

在操作系统中主体对客体的访问操作很多，我们关心的是引发安全状态迁移的操作，主要有对象创建和删除、权限添加和移除、权限委托(权限迁移)或委托撤消等。

·访问判定：定义了一个对象是否可以用权限 r 访问另一个对象，它以逻辑推理的方式表示，等价于判定算法的实现。在给定安全状态下，主体 s 可以用权限 r 访问客体 o 表述为：$\exists\ \delta\in\Delta\cdot s\to(o,r)\mid s\in S, r\in R$，也可用四元组(s,o,r,$\delta$)表示。

本文后面的模型都将基于上面的定义，针对具体的安全策略添加新的属性。

3 基于 TE 的状态迁移模型

传统的 TE 策略模型源于访问控制矩阵的思想，它为每一个用户任务绑定一个域，为每一个对象绑定一种类型。系统常常维护两类访问表 DDT 和 DIT，分别控制主体对客体的访问和主体对主体的访问，主体同时具有客体的角色。TE 的不足在于随着表的增大，关系很快复杂化，为了避免模糊，便于简单清晰地表述访问关系，我们把域和类型的概念扩展，把 Domain 和 Type 视为同一种属性，绑定到主体和客体，则 TE 只有主体集、客体集、权限集和 TE 映射表。访问控制的另一个重要的研究主题是权限的迁移，即委托权限，为了跟踪和撤消迁移权限，全局安全状态将包含迁移权限表，系统总共维护两类访问控制表。基于以上分析，基于 TE 的状态迁移模型定义如下：

1)*全局安全状态* 包括主体集 S、客体集 O、权限集 R 和两类映射 T(TE 初始权限表)和 D(迁移权限表)

$T: O\times R\to P(S\times N)$

$D: S\times R\times O\to P(S\times N)$

上式中 N 表示非负整数集，T 表示主体 s($\in S$)被授予初始权限 r($\in R$)访问客体 o($\in O$)，并且该权限可以委托给其他主体使用，再委托深度为 i($\in N$)次，即(o,r)$\vdash\!\!\to$(s,i)；D 表示主体 s_1($\in S$)将其对客体 o 的访问权限 r($\in R$)委托给另一主体 s_2($\in S$)使用，并且 s_2可以再传递该权限给其他主体，后续传递次数为 i($\in$N)，即(s_1,r,o)$\vdash\!\!\to$(s_2,i)。D 专用于撤消迁移权限时使用，当撤消主体 s 的原始权限时，可以根据 D(s,r,o)查知所有经 s 委托了的权限。

2)*安全关联行为集 Π* 系统的全局安全状态随安全关联操作 γ($\in\Pi$)变迁，γ 的参数为操作前的全局状态。对每个操作我们给出其定义和状态的变迁。令初始全局安全状态 $\Delta:=(O,R,S,T,D)$，操作后的全局安全状态 $\Delta'=(O',R',S,T',D')$，上标表示该状态改变了。操作包括对象的创建(Create)、基本权限授予(Grant)、基本权限的撤消(Remove)、权限的委托(Entrust)、迁移权限的撤消(Revoke)、对象的删除(Delete)。

·Create：对象 s 创建一个新的对象 o。Create(s,o,r)：=(O',R,S,T',D')，其中 $O':=O\cup\{o\}$，$D':=D[(s,r,o)\vdash\!\!\to\phi\mid r\in R]$，

$$T':=\begin{cases}(s,1) & r=r_m\\ \phi & r\neq r_m\end{cases}$$

r_m 为可改变对 o 的基本权限的权力。

如果 s 以权限 r_m 创建了对象 o，则 s 拥有对 o 的访问权限修改的能力，并能够将此权限赋予其他对象，否则 T(s,o,r)为空集，因为 o 刚创建，没有其他对象拥有它对的访问权限。

·Grant(s,o,r,i)：=(O,R,S,T',D)：赋予主体 s 对客体 o 的权限 γ，允许的委托深度为 i，$T'=T[(o,r)\vdash\!\!\to T((o,r))\cup\{(s,i)\}]$。Grant 操作中 T 是唯一迁移的状态。

·Remove(s,o,r,i)：=(O,R,S,T',D)：撤消主体 s 访问客体 o 的权限。T'：$T=[(o,r)\vdash\!\!\to T((o,r)-\{(s,i)\}]$。该操作只有 T 的状态发生变化，这里撤消的权限是第一次赋予的初始权限，不是委托权限。

·Entrust(s1,s2,o,r,i)：=(O,R,S,T,D')：主体 s_1委托它对客体 o 的访问权限 r 给主体 s_2，委托深度为 i，即允许 s_2再向下委托 i 次。$D':=D[(s_1,o,r)\vdash\!\!\to D((s_1,o,r))\cup\{(s_2,i)\}]$。$D$ 的状态迁移反映了权限的显式委托。

·Revoke(s1,s2,o,r,i)：=(O,R,S,T,D')：主体 s_1撤消先前委托给主体 s_2的权限(o,r)，委托深度为 i。$D':=D[(s_1,o,r)\vdash\!\!\to D((s_1,o,r))-\{(s_2,i)\}]$。

·Delete(s)：=(O,R,S',T',D')：删除主体 s。其中 $S':=S-\{(s)\}$；$T':O\times R\to P(S'\times N)$；$D':S'\times R\times O\to P(S'\times N)$。Delete 操作不仅删除了主体 s 的初始权限，而且删除了它委托出去的所有权限，更新了 T 表和 D 表。

3)*访问控制逻辑* $\forall\ s\in S\cdot\exists\ s_x\in S, o\in O, r\in R, i\in N\cdot\{s,i\}\subset T((o,r))\wedge\{s,i\}\subset D((s_x,r,o))\to$Allow(s,o,r)：=true

对于用户 s 以权限 r 访问 o 的操作，只有当 s 存在于 TE 初始权限表或委托权限表内时才被允许。

4 基于 RBAC 的状态迁移模型

RBAC 的三个主要元素为用户、角色、权限，权限被绑定

到不同的角色，而用户以不同的角色来实施对对象的访问。角色的定义是分层的，形成了树形结构。本文讨论的是 RBAC 基本模型（即 $RBAC_0$），由于 RBAC 的权限继承的撤销能力很弱，RBAC 的状态迁移模型定义如下。

1）全局安全状态　包括用户集 U（主体）、角色集 R、权限集 P、S（会话）和 UA、RA、$User$ 和 $Roles$，其中：

$UA\subseteq R\times U$；$RA\subseteq P\times R$；$US:U\rightarrow S$；$RS:S\rightarrow 2^R$。

上式中 UA 表示代表用户的任务 u（$\in U$）申请一个权限 p（$\in P$），系统创建一个会话 s（$\in S$）将绑定了特定权限集的角色集 r（$\in R$）分配给任务，该任务通过角色行使其权力；RA 表示安全核创建一个角色 r（$\in R$），并绑定权限集；US 是每一用户对应单一会话 si 的映射；RS 表示会话 si 到角色集合 $RS(si)\subseteq\{r|US^{-1}(si),r)\in UA\}$ 的映射，并且 si 具有权限 $\bigcup_{r\in Roles(si)}\{p|(p,r)\in RA\}$。

2）安全关联行为集 Π　同 TE 模型一样，对每个操作我们给出其定义和状态的变迁。令初始全局安全状态 $\Delta:=(U,R,P,S,UA,RA,US,RS)$，操作后的全局安全状态为 Δ'。在 RBAC 模型中主体即用户，客体与权限集绑定到角色，主体对客体的访问判定最终反映为用户是否具有特定的角色，所以安全关联操作表现为对用户、角色及权限三者的操作：包括 Create_User（用户（任务）的创建）、Creat_Role（角色创建）、Assign_Roles（用户角色分配）与 Revoke_Roles（取消）、Assign_Perms（角色权限分配）与 Revoke_Perm（取消）操作、Create_Session（激活角色）、Delete_User（用户（任务）的删除）。

·Create_User(u,r)：$=(U',R,P,S,UA',RA,US,RS)$。$U':=U\cup\{u\}$；$UA':=UA\cup\{(r,u)\}$。用初始角色 r 创建用户 u，绑定到 UA 集中。

·Create_Role(r,p)：$=(U,R',P,S,UA,RA,US,RS)$。$R':=R\cup\{r\}$；$PA':=PA\cup\{(p,r)\}$。创建一个角色 r，并赋予它原始的权限 p。

·Assign_Roles $(u,R_x):=(U,R,P,S,UA',RA,US,RS)$。$UA':=UA\cup(\bigcup_{ri\in Rx}\{(ri,u)\}$。分配新的角色集 R_x 给用户 u。

·Revoke_Roles $(u,R_x):=(U,R,P,S,UA',RA,US,RS)$。$UA':=UA-(\bigcup_{ri\in Rx}\{(ri,u)\}$。解除角色集 R_x 与用户 u 的绑定。

·Assign_Perms $(r,P_x):=(U,R,P,S,UA,RA',US,RS)$。$RA':=RA\cup(\bigcup_{pi\in Px}\{(pi,r)\}$。绑定新的权限集 P_x 到角色 r。

·Revoke_Perms $(r,P_x):=(U,R,P,S,UA,RA',US,RS)$。$RA':=RA-(\bigcup_{pi\in px}\{(pi,r)\})$。从角色 r 中删除以前绑定的权限集 Px。

·Create_Session $(u,si,R_x):=(U,R,P,S',UA,RA,US',RS')$。$S':=S\cup\{si\}$；$US':=US[u\mapsto US(u)\cup\{si\}]$；$RS':=RS[\bigcup_{ri\in Rx}(si\mapsto ri)]$ 其中 $U_{ri\in Rx}\{(ri,u)\}\subseteq UA$。当用户 u 要激活角色集 R_x（或相应的权限）时，系统创建一个会话 s，绑定会话到用户，再绑定相应的角色集到会话，用户就可通过角色实施对对象的访问。用户对会话是一对一关系，用户对角色和会话对角色都是一对多的关系。

·Delete_User(u)：$=(U',R,P,S',UA',RA,US',RS')$。$U':=U-\{u\}$；$S':=S-US(u)$；$UA':=UA-(\bigcup_{ri\in R}\{(ri,u)\})$；$US':=U'\rightarrow S'$；$RS':=S'\rightarrow P(R)$；（$P(R)$ 为 R 的幂集）。删除一个用户 u，同时也要从全局安全状态中删除用户绑定的会话，用户与会话的映射，会话与角色的映射。

3）访问控制逻辑　$\forall u\in U\cdot\exists p\in P\cdot US(u)\neq\phi\wedge\bigcup_{ri\in RS(US(u))}\{(ri,u)\}\subset UA\wedge\bigcup rriERS(US(U))\{(p,ri)\}\subset RA\rightarrow Allow(u,p):=true$

用户（或任务）可施行权限为 p 的操作的判定条件为：只有用户绑定了具有权限 p 的角色 r，并且通过会话 s 激活了角色 r 时安全关联操作才被允许。

5　多策略的集成模型

在本文的前期研究工作中，我们定义了一种支持多策略安全体系结构[5]，但在实现上安全核不可能为每个策略实现一个逻辑判定器，这样做系统开销过大，而且还要解决策略一致性的问题（比如多策略判定的矛盾性），所以如何集成多策略是一个重要的问题。从 TE 和 RBAC 的状态迁移模型的分析可知，不同的安全策略虽然在全局安全状态上表现形式不同，但在安全关联操作后所处的状态都是相似的，比如主体的 Create、Delete、权限的 Grant（或 Assign）和 Revoke 等，基本原因在于许多安全策略都由访问矩阵发展而来，类似的策略模型还包括 BLP[6]和 Biba[7]等。因此，我们可以基于〈主体，客体，权限〉三元组的基本思想，集成多策略。本节以 TE 和 RBAC 的集成模型 T&R 为例，讨论多安全策略的集成性。

1）全局安全状态　在 T&R 模型中，我们把 Domain 和 Type 视为同一种属性，所以主体权限被绑定到 Type，客体也由 Type 定义，主体用 RBAC 的用户集表示，主体和权限间的映射仍然用角色来完成。RBAC 模型通过角色的跟踪很好地解决了权限迁移的问题，所以 T&R 模型没有定义迁移权限管理的设施。T&B 模型的全局安全状态表示为 $(U,R,T,P,S,UR,RT,TP,US,RS)$，其中

$UR\subseteq R\times U$；$RT\subseteq T\times R$；$TP\subseteq P\times T$；

$US:U\rightarrow S$；$RS:S\rightarrow 2^R$。

上式中 UR 表示用户任务 u（$\in U$）与角色 r（$\in R$）的绑定集，是由系统创建一个会话将绑定了特定 Type 集的角色 r（$\in R$）分配给任务 u，该任务通过角色行使其权力；RT 表示安全核创建一个角色 r（$\in R$），并绑定 Type 集；TP 是系统创建一个 Type，并为其绑定权限集。S、US 和 RS 的定义与 RBAC 模型相同。所以在 T&R 模型中主体的安全属性表示为 $[user:session:role:type]$，客体的安全属性表述为 $[type:permission]$。

2）安全关联行为集 Π　T&R 模型是以 TE 为基础定义，高层的抽象为 RBAC 模型，所以其安全关联操作以 RBAC 模型为主。令初始全局安全状态 $\Delta:=(U,R,T,P,S,UR,RT,TP,US,RS)$，操作后的全局安全状态为 Δ'。在 T&R 模型中主体即用户，客体与权限集绑定为 Type，角色再用 Type 表示，主体对客体的访问判定最终反映为用户是否具有特定的角色，所以安全关联操作表现为对用户、角色、Type 与权限之间的操作：包括 Create_User（用户（任务）的创建）、Create_Role（角色创建）、Create_Type（Type 创建）、Assign_Roles（用户的角色分配）、Assign_Types（为角色分配 Type）、Assign_Perms（为角色分配权限）、Revoke_Roles（取消角色）、Revoke_Types（取消 Type）、Revoke_Perms（解绑定权限）、Create_Session（用会话激活角色）、Delete_User（用户（任务）的删除）。

·Create_User(u,r_0)：$=(U',R,T,P,S,UR',RT,TP,US,RS)$。$U':=U\cup\{u\}$；$UR':=UR\cup\{(r_0,u)\}$。用初始角色 r_0 创建用户 u，并添加到 UR 集中。

·Create_Role(r,t_0)：$=(U,R',T,P,S,UR,RT',TP,US,RS)$。$R':=R\cup\{r\}$；$RT':=RT\cup\{(t_0,r)\}$。创建一个角色 r，并赋予它基本 Type：$t_0$。

·Create_Type(t,p_0):=($U,R,T',P,S,UR,RT,TP',US,RS$)。$T':=T\cup\{t\}$;$TP':=TP\cup\{(t,p_0)\}$。创建一个Type,并赋予它相应的基本权限 p_0。

·Assign_Roles(u,R_x):=($U,R,T,P,S,UR',RT,TP,US,RS$)。$UR':=UR\cup(\cup_{ri\in Rx}\{(ri,u)\}$。分配新的角色集 R_x 给用户 u。

·Assign_Types(r,T_x):=($U,R,T,P,S,UR,RT',TP,US,RS$)。$RT':=RT\cup(\cup_{ti\in Tx}\{(t_i,r)\})$。绑定 Type 集 T_x 到角色 r。

·Assign_Perms(t,P_x):=($U,R,T,P,S,UR,RT,TP',US,RS$)。$TP':=TP\cup(\cup_{pi\in Px}\{(pi,t)\})$。绑定新的权限集 P_x 到 t。

·Revoke_Roles(u,R_x):=($U,R,T,P,S,UR',RT,TP,US,RS$)。$UR':UR-(\cup_{ri\in Rx}\{(ri,u)\})$。解除角色集 R_x 与用户 u 的绑定。

·Revoke_Types(r,T_x):=($U,R,T,P,S,UR,RT',TP,US,RS$)。$RT':=RT-(\cup_{ti\in Tx}\{(ti,r)\})$。解除 Type 集 T_x 与角色 r 的绑定。

·Revoke_Perms(t,P_x):=($U,R,T,P,S,UR,RT,TP',US,RS$)。$TP':=TP-(\cup_{pi\in Px}\{(pi,t)\})$。从名为 t 的 Type 中删除以前绑定的权限集 Px。

·Create_Session(u,si,R_x):=($U,R,T,P,S',UR,RT,TP,US',RS'$)。$S':S\cup\{si\}$;$US':=US[u\mapsto US(u)\cup\{si\}]$;$RS':=RS[\cup_{ri\in Rx}(si\mapsto ri)]$其中$\cup_{ri\in Rx}\{(ri,u)\}\subseteq$UR。该会话创建操作与 RBAC 定义相同。

·Delete_User(u):=($U',R,T,P,S,UR',RT,TP,US',RS'$)。$U':=U-\{u\}$;$S':=S-US(u)$;$UR':=UR-(\cup_{ri\in R}\{(ri,u)\})$;$US':=U'\to S'$;$RS':=S'\to P(R)$(P($R$)为 R 的幂集)。该操作也与 RBAC 的定义相同。

3)*访问控制逻辑* 在 T&R 模型中,主体对客体的访问控制表述为用户对具有特定 Type 的对象的访问。访问请求表述为(u,p,t)($u\in U,t\in T,p\in P$),其访问控制逻辑如下:

$\forall u\in U,\exists t\in T,p\in P\cdot\{(p,t)\}\subset TP\wedge\cup_{ri\in RS(US(u))}\{(ri,u)\}\subset UR\wedge\cup_{ri\in RS(US(u))}\{(t,ri)\}\subset RT\wedge US(u)\neq\phi\to$Allow(u,p,t):=true

用户(或任务)可对对象施行权限为 p 的操作的判定条件为:映射通路(p→t→某角色)存在,并且用户 u 通过特定会话 s 激活了角色 r 时访问请求才被允许。

从以上形式化分析可知,T&E 模型能够实现访问控制的操作。事实证明,由于 RBAC 的策略中性特点,使得它具有良好的组织和管理结构,能够称为集成其他安全策略的框架模型。而 TE 结构简单,在属性简化后(把 Domain 和 Type 归为同一种属性),能够作为元策略对象,表达其他的策略属性。以 T&R 为框架的多策略集成模型能够统一全局的安全属性,包容其它策略,从而保持访问控制的一致性和连续性。

6 多策略集成的支撑技术

在安全操作系统中多安全策略集成最终以策略数据库的方式实现,一般的安全操作系统都提供了策略配置设施。安全操作系统的用户需要将自然语言表述的策略转换为特定格式的策略数据库,该库能够被安全核所识别和加载,因此需要策略描述语言和策略解释器。

我们为支持多策略的集成定义了一种多策略语言,用来配置安全策略和实现策略数据库。而多种策略采用分层方式定义,即先定义简单策略的类型,再用简单策略的类型定义复杂策略的类型。比如 RBAC 是一种中性的策略,既支持机密性策略,又支持完整性策略;既支持 MAC 策略,又支持 DAC 策略。而 TE 策略是基础的访问矩阵,将其作为元语言。我们设计的多策略语言以 RBAC 为框架,TE 为基础,根据用户的需要可以包容其他安全策略,比如 BLP、Biba 等等。多策略语言的定义如下:

```
Permissions:        Perms   p1{(obj,ops)};
TE Declare:         TE_Type   t1 {p1,p2,p3…};
RBAC Declare:       RBAC_Role   r1 TE_Type {t1,t2,…};
                    RBAC_User   u1{r1,r2,…}
Security_Object:    type:{∪pi}
Security_Subject:   user:{∪ri}
Access_Allow:       type1:type2:{∪Pi}
```

先定义基本的权限类型所允许的操作,比如权限 p1允许对对象 obj 作 ops 操作;定义 TE 的一个类型 t1,享有权限{p1,p2,p3,…};定义 RBAC 的一个角色 r1,具有 TE 类型{t1,t2,…},再进一步定义用户 u1,具体实现时省略会话,直接绑定角色{r1,r2,…};客体定义为带访问权限的 type,而主体定义为绑定了多个角色的用户;最后的访问判定规则其实是基本主客体 type 类型和权限集的匹配。多策略语言配置的策略通过策略解释器转换成策略数据库,最后在系统初始化时被加载。多策略语言的优点在于统一了全局的安全状态,简化了实现,保证了策略的一致性和连续性。

结束语 多策略的可集成性问题影响了系统支持多策略和动态策略。本文采用形式化的方法建立了系统的安全状态迁移模型,分析了在不同策略模型下,策略逻辑引起安全状态变迁的共同特性。描述了系统在 TE 和 RBAC 两种安全策略模型中的全局状态、安全关联行为对全局状态的影响及其访问控制判定逻辑,并分析了它们的集成模型 T&R 模型作用下系统安全状态的变化,证明了二者的可集成性,并扩展到其他的安全策略。系统安全状态模型的建立,不但有助于评估不同策略对安全核的影响,获取多策略集成的解决方法,而且为安全操作系统的实现提供了参考。

参 考 文 献

1 石文昌.安全操作系统研究的发展.计算机科学,2002,29(6)

2 Schneider F B. Enforceable security policies. ACM Transactions on Information and System Security (TISSEC),2000,3(1)

3 Badger L, et al. Practical Domain and Type Enforcementfor UNIX. In: Proc. of the 1995 IEEE Symposiumon Security and Privacy,Oakland,California,May 1995. 66~77

4 Ravi Sandhu. Role-based access control models. IEEE Computer, 1996,29(2):38~47

5 吴新勇,熊光泽.支持动态策略的安全核(Security Kernel)机制的研究.计算机科学,2002,29(11)

6 Bell D E,LaPadula L J. Secure Computer Systems:Mathematical Foundations:[ESD-TR-73-278]. Vol. I, AD 770 768, Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base,Bedford,MA,USA,Nov. 1973

7 Biba. Integrity Considerations for Secure Computer Systems: [ESD-TR-76-372]. Electronic Systems Division, Air Force Systems Command,Hanscom Air Force Base,Bedford,MA,USA, Apr. 1977