基于角色的受限委托模型*

徐 震+, 李 斓, 冯登国

(中国科学院 软件研究所 信息安全国家重点实验室,北京 100080)

A Constrained Role-Based Delegation Model

XU Zhen+, LI Lan, FENG Deng-Guo

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

+ Corresponding author: Phn: +86-10-82612797, E-mail: xuzhen@is.iscas.ac.cn, http://is.iscas.ac.cn

Received 2003-11-20; Accepted 2004-08-10

Xu Z, Li L, Feng DG. A constrained role-based delegation model. *Journal of Software*, 2005,16(5):970–978.
DOI: 10.1360/jos160970

Abstract: Delegation is an important security policy that should be supported by RBAC model. The basic idea of delegation is that some active entity in a system delegates authority to another active entity to carry out some functions on behalf of the former. The grantor of the delegated roles should be responsible for the usage of them, so constraints on the usage of the delegated roles are critical components of the whole delegation model. Currently, there're some models that extend RBAC model to support role delegation. However, their supports for constraints on the usage of delegated roles are very limited. This paper presents the requirements of role-based delegation, including temporary constraints, regular role dependency constraints, partial delegation constraints and propagation constraints. The former two kinds of constraints are modeled with a formal model – CRDM, which provides the foundation for applications in need of the constrained delegation.

Key words: information security; access control; delegation; constraint; delegation ticket

摘 要: 角色委托是 RBAC 模型需要支持的一种重要安全策略.它的主要思想是系统中的主动实体将角色委托给其他主动实体,以便以前者名义执行特定的工作.角色委托者要对委托角色的使用负责,所以对委托角色进行使用限制是整个模型的关键组成部分.目前已有一些模型扩展了 RBAC 模型以支持角色委托,但是这些模型对委托限制的支持非常有限.提出了角色委托限制的需求,包括临时性限制、常规角色关联性限制、部分性限制和传播限制.并且,给出了一个支持临时性限制和常规角色关联性限制的基于角色的委托模型.给出模型的形式化描述,为模型在实际环境中的应用奠定了基础.

关键词: 信息安全;访问控制;委托;限制;委托票据

中图法分类号: TP309 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60025205, 60273027 (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)); the National Grand Fundamental Research 973 Program of China under Grant No.G1999035802 (国家重点基础研究发展规划(973))

作者简介: 徐震(1976－),男,山东昌邑人,博士生,主要研究领域为大型网络和信息系统安全;李斓(1977－),男,博士生,主要研究领域为大型网络与信息系统安全;冯登国(1966－),男,博士,研究员,博士生导师,主要研究领域为密码学,信息安全.

基于角色的访问控制(RBAC)模型是一种适用于大型组织、有效的访问控制模型[1−4].近年来研究者和厂商作了大量工作,对 RBAC 模型进行了多方面的增强.RBAC 模型在主体和权限之间增加了一个中间桥梁——角色,权限被授予角色而管理员通过指定用户为特定的角色来为用户授权.目前,RBAC96[4]是得到信息安全领域广泛接受的 RBAC 参考模型.当前研究的一个主要工作是在 RBAC 参考模型之上扩展的表达能力,其中基于角色的委托受到了广泛的关注[5−8].

委托是一种重要的安全策略,它的主要思想是系统中的主动实体将权限委托给其他主动实体,以便以前者的名义执行一些工作[7].在如下几种情况下会发生委托[8]:

(1) 角色备份:某人出差或度假,他负责的工作需要继续执行.这样就需要将其工作权力委托给其他人,使工作可以继续进行.

(2) 协作工作:在组织中和组织间进行的工作需要相互协作.在这种情况下,需要赋予协作对方一定的访问权限以便进行信息共享.

(3) 权力下放:组织初始构建或重组织时,需要按照组织结构从高级到低级分配权限.

在委托中有一个隐含的原则就是委托人要对被委托人的行为负责.这就意味着委托人需要对其委托的角色进行严格限制,所以委托的限制是委托策略的重要组成部分.从情形(1)、情形 2 中可以看到,委托具有临时性.临时性体现在委托角色的使用期限、使用次数以及更严格的使用时间限制上.也就是说,角色的委托只是在某个时间段内有效;在这个时间段内使用角色的次数可能是有限的;并且可能仅在该时间段的一些周期性时间片内可用.在情形 2 探讨的协作工作中,不一定需要委托角色的所有权限,这就是我们所要讨论的部分性.而在情形 1~情形 3 中都可能出现委托角色权限传播的问题,这些权限的传播必须受到限制,就是我们讨论的受限传播性.在各种情形中,委托角色都可能具有常规角色关联性.例如在角色备份的情况下,委托角色只有在委托角色的用户没有激活角色时方可激活角色.而在协作工作的情况下,委托角色可能仅在特定用户激活特定角色时方可激活.

本文将在 RBAC 背景下讨论用户到用户的角色委托限制,对角色委托的临时性、常规角色关联性约束特性进行建模支持.第 1 节介绍研究的相关的工作.第 2 节给出基于角色的受限委托模型.第 3 节对文章进行了总结,并指出进一步研究的方向.

1 相关工作

1992 年,Ferraiolo 和 Kuhn 首次提出 RBAC,并且给出了 RBAC 中大部分元素,如角色激活、层次角色和限制等[4].由于 RBAC 模型可以有效地减少管理开销并且适用于组织级的访问控制,有关的研究和应用得到了迅速发展.Ferraiolo,Sandhu 等人在 1994 年后提出了有关 RBAC 模型的早期形式化定义[1,2],其中 Sandhu 等人给出了 RBAC 模型的一个比较完整的框架[2],也就是著名的 RBAC96 模型.经过 10 多年的发展,RBAC 本身的研究已日趋成熟,2004 年初 RBAC 已成为 ANSI 标准[9].

限制从一开始就是 RBAC 研究的主要内容之一,研究者通过扩展 RBAC 的限制来增强它的表达能力,以适应不同情况下的权限管理.早期对限制的研究主要集中在权责分离[10−13]上,稍后其研究内容扩展到其他方面,包括角色的用户数目限制[2]、时间和依赖性限制[14−16]、角色的使用限制以及限制的形式化描述语言[17,18].

其中,文献[14]给出了一个角色激活受时间约束的模型 TRBAC,该模型通过角色触发器(role trigger)支持角色激活的依赖关系.文献[15]的工作建立在文献[14]的基础之上,扩展了对层次角色环境下时间限制的支持.文献[16]主要探讨时间限制与权责分离等其他限制的关系.文献[15,16]在时间和依赖限制方面的表达能力与 TRBAC 基本一致.本文中的 CRDM 采用的时间表示与 TRBAC 一致,但采用了更为简单的依赖关系支持.此外,CRDM 在角色委托的背景下探索限制问题,它能够支持角色的使用限制.UCRBAC 模型[19]主要是增加了角色的使用限制,本文在这一方面可以看做是角色使用限制在角色委托上下文中的扩展.

在基于角色的委托模型方面,文献[6]分析了基于角色的委托模型的一些特性,包括委托的持久性、单调性(委托者是否会失去委托的权限)、委托的角色完整性(是否以角色为委托的最小粒度)、管理性、委托传播性等,并提出了一些可能的角色委托模型研究方向.这些特性的分析有助于我们对约束限制的研究.Barka 和 Sandhu

提出了一个基于角色的委托模型[7],该模型支持用户到用户的角色委托,并且非形式化地提出了一些扩展,包括委托取消、部分委托、多步委托和支持层次角色的委托.文章默认存在角色委托的有效期限,但是没有给出描述和相关探讨.实际上,部分委托和多步委托都可以通过委托的约束来描述.文献[5]中 Zhang 提出了 RDM2000,该模型可以支持层次角色环境下的多步委托.文献[8]提出了 PBDM 角色委托模型,它的最大特点是支持部分委托和角色到角色的委托.其中,部分委托是通过用户创建委托角色,并将其拥有的角色中的权限赋予新的角色实现的.实现方法上有一定的创新,但是这种方式导致模型的使用和管理变得更加复杂.可以看出,基于角色的委托模型限制方面已经有了初步的研究工作,而角色委托的限制方面的研究还有待展开.

此外,研究者对委托进行了大量的研究工作[20−24],但是委托限制方面的研究成果相对较少.其中,Bandmann 在文献[24]中提出了一个受限的委托模型,该模型通过正则表达来描述委托约束以便对委托树的结构进行限制,它可以很好地将权限的委托和使用分离,并且可以有效地限制被委托权限的传播.

总的来说,RBAC 模型的研究工作已经比较成熟,而其中限制的研究工作得到了充分的重视,是当前的研究热点之一.角色的委托模型是当前 RBAC 研究的关键内容之一,将限制引入角色委托模型是一项极有价值的工作.就目前的需求而言,角色委托的临时性、常规角色关联性限制是最为紧迫的.

2 基于角色的受限委托模型

2.1 基础元素

目前被广泛接受的 RBAC 模型是 Sandhu 等人提出的 RBAC96[3].RBAC96 定义了用户(*U*)、角色(*R*)、权限(*P*)及会话(*S*)等几个集合.基于这几个集合定义了一系列关系,包括:角色授权关系(*PA*)、用户的角色授权关系(*UA*)和角色的层次关系(*RH*)等;一些函数,包括:会话到用户的映射($user: S \to U$)、会话到角色集合的映射($roles: s \to 2^R$)等;以及一些相关约束.

在此基础上,Barka 和 Sandhu 提出了基于角色的委托模型——RBDM0[7].RBDM0 主要在 RBAC96 的基础上将 *UA* 划分为 *UAO* 和 *UAD*,分别表示用户与被授予的角色的关系和用户与其被委托的角色的关系.此外,还增加了二元关系 $can\_delegate \subseteq R \times R$ 来规定角色的委托权.本文将在 RBDM0 的基础上展开研究工作,相关详细定义参见文献[3,7].

本文采用了文献[25]提出的周期时间来表示有效期限和使用时间限制,其中主要包括日历、周期时间、函数 *Π*()、函数 *Sol*(),有关定义的详细内容参见文献[14,25].在以上定义的基础上,为了方便表述,本文引入一个函数 *Pti*(),定义如下.

定义 **1**. 令 *t* 为一个时间点,*pt*=([*begin*,*end*],*P*),其中[*begin*,*end*]是一个时间段,*P* 是一个时间表达式.函数 *Pti*(*t*,*pt*)的值为时间段[*begin*,*end*].若[*begin*,*end*]∈*Π*(*P*)且 *t*∈[*begin*,*end*],否则为[−1,−1](无效时间段).

2.2 模型定义

基于角色的角色委托模型(CRDM)主要支持临时性和常规角色依赖性委托限制,这两种限制结合到一起称为委托票据.其中,常规角色依赖用来表示委托角色的使用对常规角色激活情况的依赖.在某些情况下,委托角色在使用时要求特定的用户正在或不在使用某个角色.例如,被委托进行项目管理的员工只能在项目经理不再行使职权时激活相应角色,而实习大夫只能在常规大夫在场监督时才能行使大夫职责.

定义 **2**(激活依赖、函数 ***active***()、函数 ***inactive***()).

- 激活依赖是一个集合,其中的任意元素表示为 *ua* 或~*ua*,*ua*∈*UAO*,分别表示依赖的处于激活状态的用户、角色对和非激活状态的用户、角色对,激活依赖表示为 *DEP*.
- *active*(*dep*)={*ua*|*ua*∈*dep*},得到依赖的激活状态的用户、角色对集合.
- *inactive*(*dep*)={*ua*|~*ua*∈*dep*},得到依赖的非激活状态的用户、角色对集合.
- $\forall dep: DEP \bullet active(dep) \cap inactive(dep) = \varnothing \wedge dep = active(dep) \cup \{\sim x \mid x \in inactive(dep)\}$.

定义中 $ua=(u_1,r_1)$形式的元表示,使用该委托角色,需要保证在发生请求时角色 r_1 处于被用户 u_1 激活的状态,而~*ua* 形式的元素正好相反.定义中,第 4 条性质表示任意激活依赖 *dep* 仅由定义中两种形式的用户、角色

对组成,并且它们不相交.即不存在既依赖某个用户激活特定角色,同时又依赖于某个用户非激活特定角色,这种自相矛盾的情况.下面给出委托票据定义:

定义 **3**(委托票据、委托票据集、函数 ***map***()).

- 委托票据表示为一个 5 元组(uad,pt,n,ae,dep),其中 $uad\in UAD,pt\in PT,n\in N\cup\{0\},ae\in\{all,each\},dep\in DEP$,表示为 DT,我们用 $dt_{uad},dt_{pt},dt_n,dt_{ae},dt_{dep}$ 分别表示 $dt\in DT$ 的 5 个分量.
- 两个委托票据 dt_1,dt_2 相等当且仅当它们的 5 个分量分别相等,表示为 $dt_1=dt_2$.
- 委托票据集就是一个委托票据组成的集合,表示为 DTS.
- $\forall dts:DTS,\forall dt_1,dt_2:DT\bullet,dt_1,dt_2\in dts\leftrightarrow(dt_1\neq dt_2\leftrightarrow dt_{1uad}\neq dt_{2uad})$,表示委托票据集中针对同一个用户、角色对至多有一个委托票据.
- 令 $uad\in UAD,dts\in DTS$, $map(dts,uad)=\{dt\mid dt_{uad}=uad\}$,函数返回某委托票据集中做作用于特定被委托用户和委托角色对的委托票据集合.

委托票据中的 pt 用来表示角色委托的有效期限和使用时间约束;n 表示使用次数限制;ae 表示次数限制是周期时间中任意时间段($each$)的使用次数限制,还是整个周期时间(all)的;dep 表示该角色委托需要满足的常规角色激活依赖.函数 map 返回一个委托票据的集合,集合中的元素是 dts 中作用于 uad 的委托票据.根据前面的性质我们知道,这个集合或者为空或者只有一个元素.

在表 1 中我们给出了角色和用户被授予角色的情况.第 1 行表示系统中有角色 R_1,用户 U_1,U_6 被授予该角色而用户 D_1 被委托给该角色.表 2 中给出了针对上述用户、角色关系的委托票据的例子.DT_1 应用于(D_1,R_1),它规定在用户 D_1 在 2002 年和 2003 年中每个月的前 4 天可以至多使用一次角色 R_1,并且要求激活角色时 U_3 正在使用 R_3 而 U_4 不在使用 R_4.注意,不存在针对(D_4,R_2)的委托票据,在这种情况下,默认 D_4 对 R_2 的操作不受委托限制.

Table 1 Example of relationship between users and roles

表 **1** 用户角色关系的例子

Role	Regular users	Delegated users
R_1	U_1, U_6	D_1
R_2	U_2, U_5	D_2, D_4
R_3	U_3	D_3
R_4	U_4	

Table 2 Example of delegation tickets

表 **2** 委托票据的例子

DT_1	$((D_1, R_1),\{[2002\text{-}1\text{-}1, 2003\text{-}12\text{-}31], all.Months+\{1\}.Days \triangleright 4.Days,1,each\},\{(U_3,R_3),\sim(U_4,R_4)\})$
DT_2	$((D_2, R_2),\{[2002\text{-}1\text{-}1, 2003\text{-}1\text{-}1], all.Months+\{1,10\}.Days \triangleright 4.Days,1,all\},\{\sim(U_2,R_2),\sim(U_5,R_2)\})$
DT_3	$((D_3, R_2),\{[2002\text{-}1\text{-}1, 2003\text{-}1\text{-}1], all.Months+\{4\}.Days \triangleright 1.Days,2,all\},\{(U_2,R_2),\sim(U_5,R_2)\})$

角色请求表示用户对角色的操作请求,其中用户可以执行的操作包括激活角色和去活角色.激活角色表示用户开始使用角色,从而该角色处于被相应用户激活的状态.而去活角色则表示结束使用角色,即该角色不再处于被此用户激活的状态.在模型中角色请求来自两个方面:用户和系统,系统运行过程中,为了保证满足约束会自动生成一些角色请求.用户角色请求序列用来表示完全来自于用户的角色请求.

定义 **4**(角色请求、角色请求序列、用户角色请求序列、冲突角色请求).

- 角色请求表示为一个二元组(ua,act),其中 $ua\in UA$ 是用户、角色对,$act\in\{activate,deactivate\}$表示请求进行的操作,角色请求表示为 RRE.
- 角色请求序列是一个角色请求集合组成的无限序列,表示为 $RRES.RRES(t)$表示在时间点 t 发生的角色请求的集合(可能为空).
- 用户角色请求序列是一个角色请求序列,表示为 $URRES$.其中任意元素 $URRES(t)$中的角色请求都来自用户.
- 任意两个角色请求$(ua_1,act_1),(ua_2,act_2)\in RRES(t)$,如果 $ua_1=ua_2$ 且 $act_1=act_2$,我们称它们为冲突角色请求.

通过角色请求序列我们可以对系统用户的角色操作请求进行建模.例如,表 3 是一个角色请求序列,表示在

时间点 2002-1-1 之前没有任何角色请求.在 2002-1-1 开始有用户角色请求,某时间点角色请求集合中无内容表示为空.

Table 3 Example of user role request sequence

表 3 用户角色请求序列例子

Time	Set of role requests
2002-1-1	$((U_3,R_3),activate)((D_1,R_1),activate)$
2002-1-2	$((D_2,R_2),activate)$
2002-1-3	$((U_2,R_2),deactivate)((D_3,R_2),activate)$
2002-1-4	$((D_2,R_2),activate)$
2002-1-5	$\varnothing$

角色激活状态序列记录角色的使用状态历史,任意一个时间点上的角色激活状态表示系统中处于激活状态的用户、角色对.它是我们判定常规角色依赖是否满足以及进行依赖相关处理的依据.

定义 5(角色激活状态、角色激活状态序列).

- 角色激活状态是二元组(u,r)的集合,其中$(u,r)\in UA$,表示为 RAS.
- 角色激活状态序列是一个无限序列,表示为 $RASS$.它的第 t 个元素 $RASS(t)\in RAS$ 表示在时间 t 的角色激活状态.

定义 **6**(委托角色激活历史). 委托角色激活历史是一个无限序列,表示为 RAH.它的第 t 个元素表示为 $RAH(t)$,是二元组(u,r)的集合,其中$(u,r)\in UAD$,表示在时间点 t 成功激活委托角色的操作.

注意,委托角色激活历史中只包含成功激活委托角色的请求操作,实际上就是委托角色使用的历史.后面介绍的使用次数统计函数会根据这个序列计算出指定周期时间或时间段内用户使用委托角色的次数,这些函数用来描述委托角色使用次数限制.

定义 **7**(函数 ***inrah***()、函数 ***ura***()、函数 ***uraa***()).

- $inrah(ua,rah,t)=\begin{cases}1, ua\in rah(t)\\ 0, ua\notin rah(t)\end{cases}$,$ua\in UAD, rah\in RAH, t\geq 0$,函数根据 ua 是否在时间点 t 的委托角色激活历史中返回 1 或 0.
- $ura([begin,end],ua,rah)=\sum_{t=begin}^{end} inrah(ua,rah,t)$,$[begin,end]$是一个时间段,$ua\in UAD, rah\in RAH$,函数计算给定时间段内$[begin,end]$内 ua 对应的用户使用相应角色的次数.
- $uraa(pt,ua,rah)=\sum_{[begin,end]\in\prod(pt)} ura([begin,end],ua,rah)$,$pt\in PT, ua\in UAD, rah\in RAH$,函数计算给定周期时间 pt 中 ua 对应的用户使用相应角色的总次数.

2.3 模型语义

用户角色请求或时间变化可能使得约束无法满足,从而导致模型中相应的委托角色被去活.角色去活是通过系统自动产生去活角色请求来实现的,所以在系统状态定义过程中可能出现某一时间点上的角色请求冲突的情况,即出现同一用户对同一委托角色请求不同的操作.在模型中,我们假定只处理冲突角色请求中的去活请求.此外,常规角色请求与委托角色请求交织处理可能会导致系统执行状态不唯一,因为常规角色的激活状态会影响到委托角色是否可以成功激活.为了简化模型描述,我们给出下面的假设.

假设 1.同一时间点只处理冲突角色请求中的去活角色请求,且常规角色请求优先于委托角色请求.

在模型语义定义中,整个动态过程是通过一个快照的序列来表现的,每个快照是某个时间点的状态.

定义 **8**(系统历史). 对于一个委托票据集 $dts\in DTS$ 和一个用户请求序列 $urres\in URRES$,它们的系统历史是一个四元组$(srres,orass,rass,rah)$,其中 $srres\in RRES$ 是系统角色请求序列,$orass,rass\in RASS$ 分别是常规角色激活状态序列和委托角色激活状态序列,$rah\in RAH$ 是委托角色激活历史,其中在任意时间点 $t\geq 0$, $orass(t)\subseteq UAO$ 并且 $rass(t)\subseteq UAD$.

为了方便表示,我们分别定义常规角色的激活状态序列和委托角色的激活状态序列,根据定义可知,在任意时间点 $t\geq 0$ 的常规角色激活状态与委托角色激活状态的交集为空.下面我们给出执行模型的定义.

定义 **9**(执行模型). 我们称系统历史(*srres*,*orass*,*rass*,*rah*)是委托票据集 $dts\in DTS$ 和用户请求序列 $urres\in URRES$ 的一个执行模型,当且仅当初始状态下 *srres*(0),*orass*(0),*rass*(0),*rah*(0)的任意元素均为空并且对任意 $t>0$ 满足下列规则:

规则 **1**. 在时间点 $t-1$ 满足委托票据时间限制而在 t 不满足限制的委托角色激活将导致产生在时间点 t 的去活的系统角色请求,形式化地表示为

$$\forall ur:UAD\bullet\left(\begin{array}{l}\exists dt:DT\bullet dt\in dts\wedge ur=dt_{uad}\wedge ur\in rass(t-1)\wedge\\ t-1\in Sol(pt)\wedge t\notin Sol(pt)\end{array}\right)\rightarrow(ur,deactivate)\in srres(t).$$

规则 **2**. 常规角色激活请求不仅会导致产生相应的系统角色激活请求,还可能会生成去活依赖于该用户、角色对的委托角色激活的系统角色请求,形式化地表示为

$$\left(\forall(ur,activate):RRE\bullet\left(\begin{array}{l}(ur,activate)\in urres(t)\wedge\\ ur\in UAO\wedge ur\notin orass(t-1)\end{array}\right)\rightarrow(ur,activate)\in srres(t)\right)\wedge$$

$$\left(\forall ur:UAD\bullet\left(\begin{array}{l}\exists(ur_1,activate):RRE,\exists dt:DT\bullet dt\in dts\wedge\\ ur=dt_{uad}\wedge(ur_1,activate)\in urres(t)\wedge ur_1\in UAO\\ \wedge ur_1\notin orass(t-1)\wedge ur_1\in inactive(dt_{dep})\end{array}\right)\rightarrow(ur,deactivate)\in srres(t)\right).$$

规则 **3**. 常规角色去激活请求不仅会导致产生相应的系统角色激活请求,还可能会生成去活依赖于该用户、角色对的委托角色激活的系统角色请求,形式化地表示为

$$\left(\forall(ur,deactivate):RRE\bullet\left(\begin{array}{l}(ur,deactivate)\in urres(t)\wedge\\ ur\in UAO\wedge ur\in orass(t-1)\end{array}\right)\rightarrow(ur,deactivate)\in srres(t)\right)\wedge$$

$$\left(\forall ur:UAD\bullet\left(\begin{array}{l}\exists(ur_1,deactivate):RRE,\exists dt:DT\bullet dt\in dts\wedge\\ ur=dt_{uad}\wedge(ur_1,activate)\in urres(t)\wedge\\ ur_1\in UAO\wedge ur_1\in orass(t-1)\wedge ur_1\in active(dt_{dep})\end{array}\right)\rightarrow(ur,deactivate)\in srres(t)\right).$$

规则 **4**. 用户对委托角色的激活请求会生成相应的系统角色请求,形式化地表示为

$$\forall(ur,activate):RRE\bullet\left(\begin{array}{l}(ur,activate)\in urres(t)\wedge ur\in UAD\wedge\\ ur\in rass(t-1)\wedge t\in Sol(pt)\end{array}\right)\rightarrow(ur,activate)\in srres(t).$$

规则 **5**. 用户对委托角色的去活请求会生成相应的系统角色请求,形式化地表示为

$$\forall(ur,deactivate):RRE\bullet\left(\begin{array}{l}(ur,deactivate)\in urres(t)\wedge ur\in UAD\wedge\\ ur\notin rass(t-1)\wedge t\in Sol(pt)\end{array}\right)\rightarrow(ur,deactivate)\in srres(t).$$

规则 **6**. 系统角色请求中的常规角色激活请求会将非激活常规用户、角色对激活,形式化地表示为

$$\forall(ur,activate):RRE\bullet\left(\begin{array}{l}(ur,activate)\in srres(t)\wedge\\ ur\in UAO\wedge ur\notin orass(t)\end{array}\right)\rightarrow ur\in orass(t).$$

规则 **7**. 系统角色请求中的常规角色去活请求会将处于非激活状态的常规用户、角色对去活,形式化地表示为

$$\forall(ur,deactivate):RRE\bullet\left(\begin{array}{l}(ur,deactivate)\in srres(t)\wedge\\ ur\in UAO\wedge ur\in orass(t)\end{array}\right)\rightarrow ur\notin orass(t).$$

规则 **8**. 如果没有显式的激活或去活的系统角色请求,*orass* 不会自动发生变化,表示如下:

$$\forall ur:UAO\bullet\left(\begin{array}{l}(ur\in orass(t-1)\wedge(ur,deactivate)\notin srres(t))\rightarrow ur\in orass(t)\wedge\\ (ur\notin orass(t-1)\wedge(ur,activate)\notin srres(t))\rightarrow ur\notin orass(t)\end{array}\right).$$

规则 **9**. 用户成功激活委托角色必须满足委托票据的限制,表示为

$$\forall(ur,active):RRE\bullet$$
$$\left(\begin{array}{l}(ur,active)\in srres(t)\wedge ur\in UAD\wedge(ur,deactivation)\notin srres(t)\wedge\\ \left(\begin{array}{l}\forall dt''\,DT\bullet dt\in dts\wedge\\ \left(\begin{array}{l}map(dts,ur)=\varnothing\vee\\ \left(\begin{array}{l}dt\in map(dts,ur)\wedge t\in Sol(t)\wedge active(dt_{dep})\subseteq orass(t)\\ \wedge(\forall ur':UAO\bullet ur'\in inactive(dt_{dep})\rightarrow ur'\notin orass(t))\wedge\\ ((dt_{ae}=all\wedge uraa(dt_{pt},ur,rah)<dt_n)\vee\\ (dt_{ae}=each\wedge ura(Pti(t,dt_{pt}),ur,rah)<dt_n))\end{array}\right)\end{array}\right)\end{array}\right)\rightarrow ur\in rass(t),ur\in rah(t).$$

规则 **10**. 去活委托角色的请求会导致委托角色状态的相应变化,表示如下:

$$\forall(ur,deactivate)\left\{\begin{array}{l}(ur,deactivation)\in srres(t)\wedge\\ ur\in UAD\end{array}\right\}\rightarrow ur\notin rass(t).$$

规则 **11**. 如果没有显式的激活或去活的委托角色请求,*rass* 不会自动发生变化,表示如下:

$$\forall ur:UAD\bullet ur\in rass(t-1)\wedge(ur,deactivate)\notin srres(t)\rightarrow ur\in rass(t).$$

规则 1~规则 5 是 *srres*(*t*)的生成原则,根据委托票据 *dts*,*urres*(*t*)和 *t*–1 的执行模型产生;规则 6~规则 8 是 *orass*(*t*)的生成原则,根据 *srres*(*t*)和 *t*–1 的系统历史产生;*rass*(*t*)根据 *srres*(*t*),*orass*(*t*)和 *t*–1 的执行模型产生,满足规则 9~规则 11;*rah*(*t*)根据 *srres*(*t*),*orass*(*t*)和 *t*–1 的执行模型产生,满足规则 9.

定义 10 的规则中实际要求首先计算 *srres*(*t*),然后生成 *orass*(*t*),进而得到 *rass*(*t*)和 *rah*(*t*),它与假设 1 共同作用可以避免系统状态变化的二义性.委托票据的限制主要体现在规则 9 中,它规定只有满足限制的被委托用户才可以使用委托角色.同时,规则 9 中还体现了假设 1 中的去活角色请求优先的原则.而规则 1~规则 3 表现了限制依赖条件由满足到不满足变化时的处理.根据定义 10,我们有下面的定理.

定理 **1**. 任意执行模型始终满足其相应的委托票据集.

证明:令执行模型(*srres*,*orass*,*rass*,*rah*)是委托票据集 *dts*∈*DTS* 和用户请求序列 *urres*∈*URRES* 的一个执行模型.

在 *t*=0 时,执行模型中各个元素均为空,所以一定满足 *dts*.

我们假定在 *t*=*k* 时,执行模型满足 *dts*.

我们需要证明 *t*=*k*+1 时,执行模型仍然满足 *dts*.可能导致执行模型不满足 *dts* 的情况有 3 种:用户角色请求不满足临时性约束或常规角色依赖、委托票据适用的时间段过期和常规用户角色请求导致委托票据中的常规角色依赖无法满足,下面我们分别给出讨论:

(1) 在第 1 种情况中,根据规则 4、规则 5,*k*+1 时间点用户角色请求中的委托角色请求会进入 *srres*(*k*+1),在处理过程中,它们不满足规则 9,所以请求会被拒绝.

(2) 第 2 种情况指的是 *k* 时间点(*u*,*r*)∈*rass*(*k*),而时间点 *k*+1 不在它对应的委托票据适用的时间段内.这时,根据规则 1,系统将自动生成一个去活(*u*,*r*)的请求,根据规则 10,(*u*,*r*)不会出现在 *rass*(*k*+1)中.并且根据规则 9,如果同时有一个激活(*u*,*r*)请求,则该激活请求将被忽略.

(3) 在第 3 种情况下,根据规则 2、规则 3,在时间点 *k*+1 的常规用户角色请求会导致生成去活所有在时间点 *k* 依赖于该常规用户、角色对状态其处于激活状态的用户、委托角色对的系统角色请求(加入 *srres*(*k*+1)).而根据规则 6、规则 7、规则 10,上述用户、委托角色对不会出现在 *rass*(*k*+1)中.

(4) 根据规则 8、规则 11,只有 *sress*(*k*+1)会导致 *orass*(*k*+1)和 *rass*(*k*+1)发生变化.

综上所述,执行模型在 *k*+1 时间点满足 *dts*,得证. □

根据定义 10 和假设 1 可知,执行模型在运行过程中不会产生二义性,所以有如下定理:

定理 **2**. 给定委托票据集和用户角色请求序列,则它们有唯一的执行模型.

下面我们给出一个执行模型的例子,如图 1 所示.其中,用户、角色关系见表 1,委托票据集合和用户角色请求见表 2、表 3.

Time	System role request sequence
2002-1-1	((U_3,R_3),*activate*),((D_1,R_1),*activate*)
2002-1-2	((D_2,R_2),*activate*)
2002-1-3	((U_2,R_2),*activate*),((D_2,R_2),*deactivate*)
2002-1-4	((D_2,R_2),*activate*)
2002-1-5	((D_1,R_1),*deactivate*)

Time	Delegated role activaion status sequence	Regular role activaion status sequence
2002-1-1	(D_1,R_1)	(U_3,R_3)
2002-1-2	(D_1,R_1),(D_2,R_2)	(U_3,R_3)
2002-1-3	(D_1,R_1)	(U_3,R_3), (U_2,R_2)
2002-1-4	(D_1,R_1)	(U_3,R_3),(U_2,R_2)
2002-1-5	∅	(U_3,R_3),(U_2,R_2)

Time	Role activation history
2002-1-1	(D_1,R_1)
2002-1-2	(D_2,R_2)
2002-1-3	∅
2002-1-4	∅
2002-1-5	∅

Fig.1 Example of execution model
图 1 执行模型例子

在例子中我们看到,时间 2002-1-1 的两个用户角色请求生成两个系统角色请求,它们成功地执行并在委托角色激活状态中增加了(U_3,R_3),在常规角色激活状态中增加了(D_1,R_1).2002-1-2 的用户角色请求也是成功的.而 2002-1-3 的常规用户激活角色请求((U_2,R_2),*activate*)会导致(D_2,R_2)的依赖条件无法满足,产生一个额外系统角色请求((D_2,R_2),*deactivate*),结果是(U_2,R_2)被激活而(D_2,R_2)被去活.此外,2002-1-3 的((D_3,R_2),*activate*)不满足委托票据的时间约束未被加入系统角色请求序列,因为 D_3 只能在每个月的 4 号激活角色 R_2.而 2002-1-4 的请求((D_2,R_2),*activate*)会被拒绝,因为根据 DT_2 用户 D_2 在 2002-1-1~2002-1-4 只能使用一次角色 R_2.2002-1-5 的系统角色请求是根据委托票据生成的去活请求,因为 2002-1-5 不在 DT_1 允许的时间范围内(D_1,R_1)将被去活.

3 结束语

本文给出了基于角色的受限委托模型(CRDM),该模型扩展 RBDM0 以支持角色委托限制.我们主要探讨基于角色的委托模型中的限制问题,提出了角色委托限制的需求,其中包括临时性限制、部分性限制、常规角色关联性限制以及受限传播性限制.本文形式化地定义了基于角色的受限委托模型,该模型可以灵活地支持临时性限制和常规角色关联性限制.其中的临时性主要表现在委托有效期限制、激活时间限制、使用次数限制上.而常规角色关联性表现为常规角色依赖限制,它能够确保委托角色的激活满足特定的常规角色激活状态.

目前,CRDM 是基于扁平角色的,扩展支持角色继承是我们的工作之一.多步委托是进行角色委托传播限制研究的基础.首先,我们需要对它进行支持.在这个基础上进行委托角色传播限制是一个有趣的话题,可以考虑对传播树的结构进行限制——包括深度、宽度、每个节点选择范围以及传播中的部分委托.角色传播中的部分委托也是值得进一步研究的问题.此外,本文对各种角色委托的限制的描述相对独立,缺乏统一描述各种委托限制的框架,这也是我们正在考虑解决的问题.

References:

[1] Ferriaolo D, Cugini J,Kuhn R. Role-Based access control (RBAC): Features and motivations. In: Proc. of the 11th Annual Computer Security Application Conf. New Orleans: IEEE Computer Society Press, 1995. 241−248. http://csrc.nist.gov/rbac/ferraiolo-cugini-kuhn-95.pdf

[2] Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-Based access control models. IEEE Computer, 1996,29(2):38−47.

[3] Sandhu RS. Rationale for the RBAC96 family of access control models. In: Youman C, Sandhu R, Coyne E, eds. Proc. of the 1st ACM Workshop on Role-Based Access Control. New York: ACM Press, 1996.

[4] Ferraiolo D, Kuhn R. Role-Based access control. In: Proc. of the 15th National Computer Security Conf. 1992. 554–563. http://csrc.nist.gov/rbac/ferraiolo-kuhn-92.pdf

[5] Zhang LH, Ahn G-J, Chu B-T. A rule-based framework for role-based delegation. In: Sandhu RS, Jaeger T, eds. Proc. of the 6th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2001. 153−162.

[6] Barka E, Sandhu R. Framework for role-based delegation models. In: Proc. of the 16th Annual Computer Security Application Conf. IEEE Computer Society Press, 2000. 168−176. http://csdl.computer.org/comp/proceedings/acsac/2000/0859/00/08590168.pdf

[7] Barka E, Sandhu R. A role-based delegation model and some extensions. In: Proc. of the 23rd National Information Systems Security Conf. (NISSC 2000). 2000. http://www.list.gmu.edu/confrnc/nissc/rbdm00.pdf

[8] Zhang XW, Oh S, Sandhu RS. PBDM: A flexible delegation model in RBAC. In: Ferrari E, Ferraiolo D, eds. Proc. of the 8th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2003. 149−157.

[9] ANSI INCITS 359-2004. Role Based Access Control. American National Standard for Information Technology, 2004.

[10] Chen F, Sandhu R. Constraints for role-based access control. In: Youman C, Sandhu R, Coyne E, eds. Proc. of the 1st ACM Workshop on Role-Based Access Control. New York: ACM Press, 1996.

[11] Simon RT, Zurko ME. Separation of duty in role-based environments. In: Proc. of the 10th Computer Security Foundations Workshop. Washington, DC: IEEE Computer Society Press, 1997. 183−194. http://csdl.computer.org/comp/proceedings/csfw/1997/7990/00/79900183.pdf

[12] Gligor VD, Gavrila SI, Ferraiolo D. On the formal definition of separation-of-duty policies and their composition. In: Proc. of the 1998 IEEE Computer Society Symp. on Research in Security and Privacy. Washington, DC: IEEE Computer Society Press, 1998. 172−183.

[13] Jaeger T. On the increasing importance of constraints. In: Proc. of the 4th ACM Workshop on Role-Based Access Control. New York: ACM Press, 1999. 33−42. http://portal.acm.org/ft_gateway.cfm?id=319175&type=pdf

[14] Bertino E, Bonatti PA, Ferrari E. TRBAC: A temporal role-based access control model. ACM Trans. on Information and System Security, 2001,4(3):191−233.

[15] Joshi JBD, Bertino E, Ghafoor A. Temporal hierarchy and inheritance semantics for GTRBAC. In: Proc. of the 7th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2002. 74−83. http://shay.ecn.purdue.edu/~dmultlab/Security/sacmat2002.pdf

[16] Joshi JBD, Shafiq B, Ghafoor A, Bertino E. Dependencies and separation of duty constraints in GTRBAC. In: Proc. of the 8th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2003. 51−64. http://shay.ecn.purdue.edu/~dmultlab/Security/p313-joshi.pdf

[17] Ahn GJ, Sandhu R. Role-Based authorization constraints specification. ACM Trans. on Information and System Security, 2000, 3(4):207−226.

[18] Dong GY, Qing SH, Liu KL. Role-Based authorization constraint with time character. Journal of Software, 2002,13(8):1521−1527 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1521.pdf

[19] Xu Z, Feng DG, Li L, Chen H. UC-RBAC: A usage constrained role-base access control model. In: Qing SH, Gollmann D, Zhou JY, eds. Proc. of the 5th Int'l Conf. on Information and Communications Security. LNCS 2836, Heidelberg: Springer-Verlag, 2003. 337−347.

[20] Gasser M, McDermott E. An architecture for practical delegation in a distributed system. In: Cooper D, Lunt T, eds. Proc. of the 1990 IEEE Computer Society Symp. on Research in Security and Privacy. Oakland: IEEE Computer Society Press, 1990. 20−30.

[21] Gladny HM. Access control for large collections. ACM Trans. on Information Systems, 1997,15(2):154−194.

[22] Moffett JD, Sloman MS. The source of authority for commercial access control. IEEE Computer, 1988,21(2):59−69.

[23] Nagaratnam N, Lea D. Practical delegation for secure distributed object environments. Distributed Systems Engineering, 1998,5(4): 168−178.

[24] Bandmann O, Dam M, Firozabadi BS. Constrained delegation. In: Proc. of the 23rd Annual IEEE Symp. on Security and Privacy. Oakland: IEEE Computer Society Press, 2002. 131−143. http://csdl.computer.org/comp/proceedings/sp/2002/1543/00/15430131abs.htm

[25] Niezette M, Stevenne J. An efficient symbolic representation of periodic time. In: Finin TW, Nicholas CK, Yesha Y, eds. Proc. of the 1st Int'l Conf. on Information and Knowledge Management. LNCS 752, Springer-Verlag, 1992.

附中文参考文献:

[18] 董光宇,卿斯汉,刘克龙.带时间特性的角色授权约束.软件学报,2002,13(8):1521−1527. http://www.jos.org.cn/1000-9825/13/1521.pdf