扩展的基于角色的访问控制模型

薛 伟 怀进鹏

(北京航空航天大学 计算机学院, 北京 100083)

摘 要: 提出了一种扩展的基于角色的访问控制 RBAC(Role Based Access Control)模型——RTBAC (Role & Task Based Access Control)模型. 该模型在 RBAC96 模型之上引入了任务和任务实例的概念,形式化地定义了任务和任务实例的层次结构,界定了传统会话同任务实例之间的关系以及任务实例同权限之间的关系,并且提供了几种辅助函数. 该模型可以更为自然地描述业务流程和访问控制策略,更适合分布式协作应用,特别是工作流和组合服务. 基于该模型定义了一种新的动态职责分离约束——基于任务的动态职责分离约束,并且同传统动态职责分离约束进行了比较. 该约束可以更准确地刻画访问控制相关的系统运行时上下文的范围,从而提高运行时访问控制的效率.

关 键 词: 访问控制模型; 基于角色的访问控制; 运行时上下文; 动态职责分离

中图分类号: TP 393.08

文献标识码: A **文 章 编 号**: 1001-5965(2005)03-0298-05

Extended role-based access control model

Xue Wei Huai Jinpeng

(School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

Abstract: An extended RBAC(role based access control) model, RTBAC (role & task based access control) model was presented. The model introduced the notions of task and task instance into RBAC96 model, formally defined the hierarchies of tasks and task instances, specified the relationships between traditional sessions and task instances as well as the relationships between task instances and permissions. Several assistant functions were defined. The model could be used to depict daily business procedures and related access control policies more naturally, so was more suitable for distributed collaborative applications, especially for workflows and service compositions. Based on this model, a new dynamic separation of duty constraint, namely task-based dynamic separation of duty constraint, was formally defined and compared with traditional dynamic separation of duty constraints using a typical example. The new constraint can specify access control related system runtime context more accurately. It can increase the efficiency of access control at runtime.

Key words: access control model; role-based access control; runtime context; dynamic separation of duty

随着网络技术的发展,分布式协作应用逐渐成为主流的应用模式,地域分散的多个用户同时或先后与系统交互,完成各自的任务,并通过协作实现共同目标. 这种应用模式要求访问控制模型能够刻画影响访问决策的业务流程特征,一个著名的例子就是职责分离的安全原则[1~4]. 然而,现

收稿日期: 2003-09-28

基金项目: 国家 863 计划基金资助项目(2002AA113030,2003AA144150); 国家自然科学基金资助项目(90412011)

作者简介: 薛 伟(1977-),男,陕西定边人,博士生, xue@act.buaa.edu.cn.

有基于角色的访问控制 RBAC(Role Based Access Control)模型(如 RBAC96 模型[5]和 NIST RBAC 模型[6])主要还是静态模型,即主要用于维护用户的权限指派,其刻画系统运行时上下文的能力有限,无法满足分布式协作应用的需求. 在 RBAC96 模型族中,用于刻画系统运行时状态的组件只有会话(session)及相关映射关系,而会话仅刻画了单个主体同系统的交互,无法刻画主体之间的协作行为. 文献[2]提出了一个扩展的 RBAC 模型,引入了刻画系统运行时上下文的组件,并给出了一些职责分离的形式化定义,包括文献[1]中提出的基于历史的动态职责分离. 但是该文将应用简单理解为客体集合、操作集合和操作-客体对集合,无法刻画应用业务流程的逻辑结构; 其所刻画的系统执行历史只是简单的系统状态转移序列,无法刻画并发业务流程的内部结构和相互关系.

针对上述问题,本文在 RBAC96 模型族的基础上引入任务(task)和任务实例(task instance)的概念,提出了一种新的扩展 RBAC 模型——RTBAC(Role & Task Based Access Control)模型,该模型更为全面地刻画了分布式协作应用中与访问控制相关的系统运行时上下文. 本文首先给出 RTBAC 模型的形式化描述;然后基于该模型定义了一种新的动态职责分离 DSoD(Dynamic Separation of Duty)约束——基于任务的动态职责分离约束 TDSoD(Task-based Dynamic Separation of Duty);最后通过一个例子展示了 RTBAC 模型刻画系统运行时上下文的能力,并同相关工作进行了比较.

1　RTBAC 模型

为便于描述,除非特殊声明,本文用大写字母(包括带下标的大写字母)表示集合,用小写字母(包括带下标的小写字母)表示集合中的元素.

任务(或活动)是协作应用中经常用到的概念,它定义了一段业务流程. 协作应用的业务流程往往由多个任务构成,一个任务可以包含多个子任务. 任务实例(或活动实例)是任务的一次执行过程,一个协作应用在运行时往往同时维护着多个任务实例,在任务实例的执行过程中可能会创建子任务实例. 利用任务和任务实例的概念可以较为全面地刻画协作应用的业务流程在定义态和运行时的结构信息(运行时上下文),因此引入这 2 个概念来扩展 RBAC 模型. 由于并不打算取代原有的业务流程控制系统,因此在引入这 2 个概念的时候仅抽取了访问控制所需的一些特征,并不包含流程控制、任务实例状态等信息.

在 RTBAC 模型中,任务和权限一样,是不解释的符号,任务集合记作 T. 为刻画业务流程的逻辑结构,引入如下定义:

定义 1　TH 是 T 上的一个偏序关系,称作任务构成关系,记作 $\geq_T$,对于 $t_1, t_2 \in T$, $t_1 \geq_T t_2$ 意味着 t_2 是 t_1 的子任务.

没有子任务的任务被称作原子任务(atomic task),包含子任务的任务被称作组合任务(composite task). 原子任务集合记作 AT,组合任务集合记作 CT,且满足 $AT \cap CT = \varnothing$, $T = AT \cup CT$.

定义 2　任务实例是一个三元组<t, sub _ tis, executors>,其中 t 是该任务实例对应的任务,sub _ tis 是该任务实例执行过程中到目前为止创建的子任务实例的集合,executors 是参与该任务实例执行的用户的集合. 任务实例集合记作 TI.

为刻画业务过程执行的结构,引入如下定义:

定义 3　TIH 是 TI 上的一个受限的偏序关系,称作任务实例构成关系,记作 $\geq_{TI}$,对于 $ti_1, ti_2 \in TI$, $ti_1 \geq_{TI} ti_2$ 意味着 ti_2 是 ti_1 的子任务实例.

直接任务实例构成关系记作 $>>_{TI}$,如果 $ti_1 \geq_{TI} ti_2$ 且没有其它的任务实例 ti_3 使得 $ti_1 \geq_{TI} ti_3$ 且 $ti_3 \geq_{TI} ti_2$,那么 $ti_1 >>_{TI} ti_2$. $\forall ti, ti_1, ti_2, ti_1 >>_{TI} ti \wedge ti_2 >>_{TI} ti \Rightarrow ti_1 = ti_2$,即每个子任务实例只能有一个直接父任务实例. 没有子任务实例的任务实例称作原子任务实例,有子任务实例的任务实例称作组合任务实例,其集合分别记作 ATI 和 CTI. $ATI \cap CTI = \varnothing$, $TI = ATI \cup CTI$.

在 RBAC96 模型的基础上,通过增加任务及其实例、操作和约束组件,定义 RTBAC 模型(如图 1 所示)如下:

定义 4　RTBAC 模型是 RBAC96 模型的扩展,扩展部分如下:

1) T(任务集合)、TI(任务实例集合)及其上的关系 TH(任务构成关系)、TIH(任务实例构成关系).

2) T, TI, AT, ATI 上的一组函数

① execute: S→ATI,将一个会话映射到该会话目前所执行的原子任务实例(注:用户通过会话执行原子任务实例,进而参与包含该原子任务实例的组合任务实例的执行);

② is _ instance _ of: TI→T,将一个任务实例映射到一个任务,该任务实例是这个任务的一次执行;

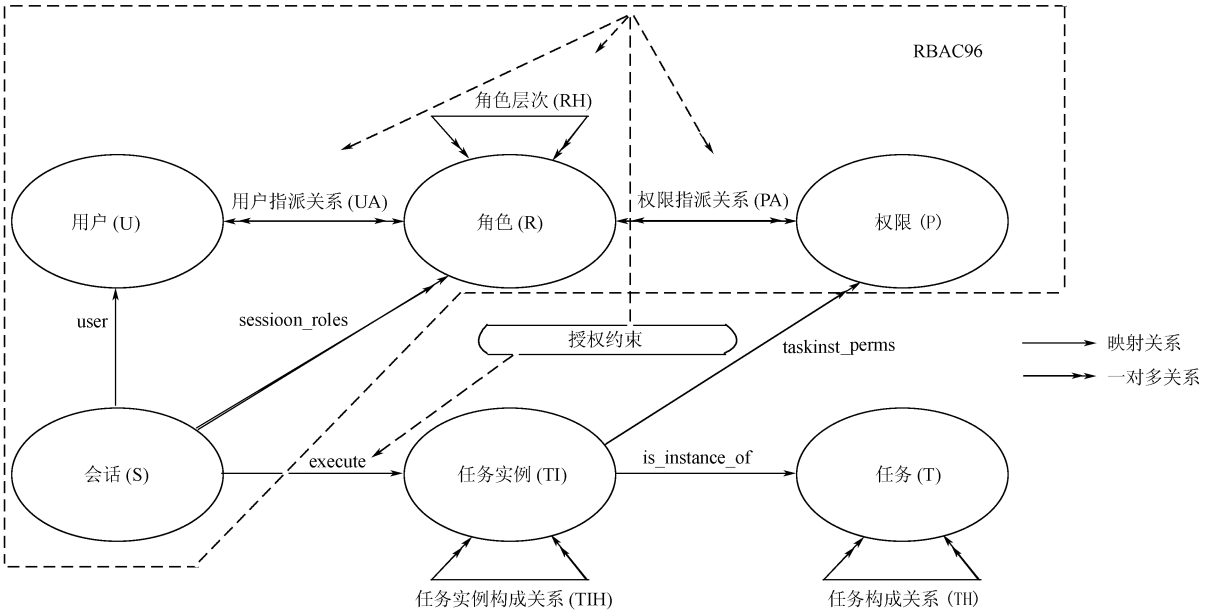

图1 RTBAC

③ perms _ needed _ by: AT→2^{2^P},将一个原子任务映射到完成该原子任务所需的权限的不同组合(注:完成一项原子任务可以有多种方式,需要不同的权限组合);

④ taskinst _ perms: ATI→2^P,将一个原子任务实例映射到完成该原子任务实例所用到的权限集合(注:在创建原子任务实例时完成资源分配,确定了该原子任务实例所用到的权限集合).

3) 一组约束:规定了 RTBAC 模型中其它组件,如 execute 的可接受的取值.

在定义4的基础上,为了便于应用,定义 T, TI, AT, ATI 上的一组辅助函数.

定义5 辅助函数

1) sub _ tasks: T→2^T,将一个任务映射到其所有子任务(该任务本身除外)组成的集合,sub _ tasks(t) = $\{t' \in T \mid t \geq_T t'\} \setminus \{t\}$;

2) participate _ in: S→2^{TI},将会话映射到其目前参与执行的任务实例集合,participate _ in(s) = $\{ti \in TI \mid \exists ti' \in ATI \cdot ti \geq_{TI} ti' \wedge execute(s) = ti'\}$.

2 基于任务的动态职责分离约束

众所周知,职责分离是一条重要的安全原则,其目的是确保只有多人合谋才能够完成欺诈,使欺诈所冒风险更大,从而抑制欺诈的发生. 职责分离可以分为静态和动态2类,其中静态职责分离限制权限的指派,动态职责分离限制系统运行时权限的激活和使用. 由于静态职责分离过于苛刻和脱离实际[1],因此动态职责分离是目前研究的重点.

在 RBAC 模型中,职责分离原则体现为职责分离约束,其定义和强制依赖于访问控制模型刻画系统运行时上下文的能力. NIST RBAC 模型采用会话来描述系统运行时上下文,它所定义的动态职责分离约束(即文献[4]中基于会话的动态职责分离约束)禁止用户在同一个会话中同时激活互斥的多个角色. 由于会话不足以描述系统运行时上下文,因此这种定义仅部分地限制了权限的激活,无法有效限制权限的使用. 例如,用户可以同时打开2个会话,在2个会话中分别激活互斥的角色,一样可以实现对互斥权限的同时使用.

文献[2]中给出了其它一些动态职责分离约束的形式化定义,分别是 DSoD(动态职责分离)、ObjDSoD(基于客体的动态职责分离)、OpDSoD(操作性动态职责分离)和 HDSoD(基于历史的动态职责分离). 然而这些定义具有如下局限性:

1) DSoD(即文献[4]中基于用户的动态职责分离约束):考虑了一名用户同时打开多个会话的情况,但是仅考虑了单个系统状态,无法处理用户先后激活多个会话或者在一个会话中先后激活多个互斥角色的情况;

2) ObjDSoD:使用了记录系统运行历史的函数 access _ history,能够在一定程度上处理先后执行互斥操作的问题. 但是,正如文献[1]所指出的那样,在某些情形下,一个用户在同一个客体上执行多个操作是合理的,但 ObjDSoD 不允许出现这

种情形. 此外,正如本文开头所指出的那样,由于其模型无法刻画应用业务流程的逻辑结构和并发业务流程间的关系,因此无法区分在同一客体上同一操作的多次执行,也就无法表达它们之间的互斥;

3) OpDSoD:同 DSoD 类似,因为没有考虑历史信息,所以无法处理用户先后激活多个会话,或者在一个会话中先后激活互斥角色的情况. 此外,正如文献[1]所指出的那样,OpDSoD 不允许一个用户拥有执行应用所有操作的权限,即使这些授权针对不同客体,这种限制过强;

4) HDSoD:结合了 ObjDSoD 和 OpDSoD 的优点. 但是,同 ObjDSoD 一样,HDSoD 无法区分在同一客体上同一操作的多次执行. 此外,其对历史的刻画过于简单,难于界定相关运行时上下文的准确范围.

文献[3]提出了一个职责分离约束的规范框架,该框架能以简单的语法刻画多种职责分离约束,并且区分了静态、动态和历史性 3 种约束上下文. 但这种上下文区分仍然很粗糙,没有结合业务流程进一步地明确历史性上下文. 此外,该文定义的针对任务执行的基于历史的职责分离约束(U,$\{p_1,\cdots,p_n\}$,h)假定任务的执行等价于一系列权限的调用,无法区分同一客体上同一操作的多次执行.

针对上述定义的不足,在 RTBAC 模型的基础之上定义一种新的动态职责分离约束如下:

定义 6　基于任务的动态职责分离约束是一个三元组<t,sts,threshold>,其中 $t \in T$ 是与该约束相关的任务;$sts \subseteq sub_tasks(t)$ 是该任务中存在互斥关系的子任务组成的集合

$$\forall t_1, t_2 \in sts \cdot t_1 \not\geqslant_T t_2 \wedge t_2 \not\geqslant_T t_1$$

即具有任务构成关系的子任务不能同时出现在 sts 中;threshold $\in \mathbf{N}$(自然数集合)是一个阈值.

对于系统中运行的任何任务实例,如果其对应的任务是 t,那么任何一个用户参与执行的以 sts 中元素为对应任务的子任务实例的数目都不得大于等于 threshold. 基于任务的动态职责分离约束的集合记作 TDSoD. 形式化地:$\forall tdsod \in TDSoD, \forall ti, ti', ti'' \in TI, \forall tiset \subseteq ti.sub_tis \cdot ti.t = tdsod.t \wedge \{ti'.t \mid ti' \in tiset\} \subseteq tdsod.sts \wedge |tiset| \geqslant tdsod.threshold \Rightarrow \bigcap_{ti'' \in tiset} ti''.executors = \varnothing$.

3　示例分析及相关工作比较

本节以文献中常用的支票处理任务(如图 2 所示)为例,进一步说明 RTBAC 模型刻画系统运行时上下文的能力,并与相关工作进行比较.

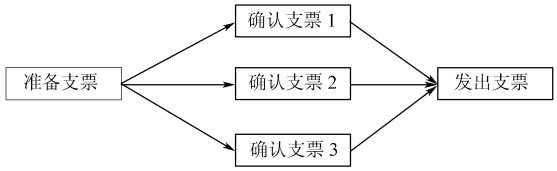

图 2　支票处理任务

在支票处理任务中,一名雇员负责准备支票并且指定帐户,然后由 3 个彼此独立的审查者对该支票和帐户进行确认,最后由另外一名雇员将该支票发出. 这是一个典型的组合任务,由准备支票、发出支票、确认支票 3 个子任务组成,其中前 2 个是原子任务,确认支票子任务又由 3 个并行的原子任务(确认支票 1、确认支票 2、确认支票 3)组成.

在这个场景中,存在如下的动态职责分离约束:

1) 准备支票子任务和发出支票子任务必须由不同的用户来完成,即<支票处理,{准备支票,发出支票},2>;

2) 确认支票的 3 个子任务必须由 3 个不同的用户来完成,即<确认支票,{确认支票 1,确认支票 2,确认支票 3},2>;

3) 在最严格的情况下,要求所有 5 个原子任务都由不同的用户来完成,即<支票处理,{准备支票,确认支票 1,确认支票 2,确认支票 3,发出支票},2>.

在 NIST RBAC 建议标准中,正常情况下完成上述任务需要由五名用户创建 5 个会话,并在各自的会话中激活适当的角色. 尽管 NIST RBAC 模型可以约束一名雇员无法在一次会话中同时激活准备支票子任务和发出支票子任务所需的角色,但是一名雇员可以同时打开 2 个会话,或者在任务执行的不同阶段先后打开 2 个会话,然后在两个会话中分别激活完成这 2 项子任务所需的角色,从而绕过上述第一个约束. DSoD 和 OpDSoD 也有类似的情况,而 ObjDSoD,HDSoD,文献[3]中的(U,$\{p_1,\cdots,p_n\}$,h)以及基于任务的动态职责分离约束可以避免这个问题.

再考虑上述第二个动态职责分离约束,确认支票的 3 个子任务是对同一客体的 3 次相同的操作,由于其模型刻画系统运行时上下文的能力不足,ObjDSoD,HDSoD 以及文献[3]的(U,$\{p_1,\cdots,p_n\}$,h)都无法区分同一操作的多次执行,因而无

法表达这个动态职责分离约束,而 RTBAC 可以通过区分子任务实例来区分同一操作的多次执行,因此可以定义这一动态职责分离约束.

此外,利用 RTBAC 模型中任务的概念,基于任务的动态职责分离约束准确地界定了为强制动态职责分离约束而要记录的系统运行时上下文的范围,因此可以有效地降低系统的运行时开销. 这是基于任务的动态职责分离约束相对于其他动态职责分离约束,特别是基于历史的动态职责分离约束的另一优点.

基于任务的访问控制 TBAC(Task-Based Access Control)模型[7,8]是 Thomas 和 Sandhu 提出的一种主动安全模型[9],它借鉴了主动数据库的思想,用面向任务或者面向事务的视角来取代传统的主体-客体的视角,随应用业务流程动态地改变主体的授权,在一定程度上实现了授权管理的自动化. 本文工作同 TBAC 的最大差别在于:本文仍沿用 RBAC96 模型所采用的主体-客体视角来看待应用系统中的访问控制,将权限指派同权限的激活和使用区分开来,通过各种动态授权约束的强制来限制权限的激活和使用,而不是频繁地改变主体的权限指派. 此外,TBAC 模型是独立于 RBAC 模型的,而 RTBAC 模型则是在 RBAC96 模型的基础之上扩展得到的,因此可以更容易地同 NIST RBAC 标准[6]相结合.

4 结束语

本文提出了一种扩展的 RBAC 模型——RTBAC 模型,它能够更全面地刻画与访问控制相关的系统运行时上下文,因此更适合分布式协作应用;本文还在 RTBAC 模型的基础之上定义了基于任务的动态职责分离约束,该约束可以准确界定相关的系统运行时上下文的范围,可以有效降低系统的运行开销.

多自治域协作场景(如 B2B 电子商务应用、网格应用、各种跨组织的联合(coalition)应用)中的访问控制是目前系统安全领域研究的一个热点. 目前正在研究 RTBAC 模型在这些场景下的应用.

参考文献 (**References**)

[1] Simon R T, Zurko M E. Separation of duty in role-based environments[A]. In: Proceedings of Computer Security Foundations Workshop X [C]. Washington: IEEE Computer Society, 1997. 183 ~ 194

[2] Gligor V D, Gavrila S I, Ferraiolo D F. On the formal definition of separation-of-duty policies and their composition [A]. In: Proceedings of 1998 Symposium on Research in Security and Privacy [C]. Washington: IEEE Computer Society, 1998. 172 ~ 185

[3] Crampton J. Specifying and enforcing constraints in role-based access control[A]. In: Proceedings of ACM Symposium on Access Control Models and Technologies [C]. New York: ACM Press, 2003. 43 ~ 50

[4] Ahn G J, Sandhu R. Role-based authorization constraints specification[J]. ACM Transactions on Information and System Security, 2000,3(4):207 ~ 226

[5] Sandhu R, Conyne E J, Lfeinstein H, *et al*. Role based access control models[J]. IEEE Computer, 1996,29(2):38 ~ 47

[6] Ferraiolo D F, Sandhu R, Gavrila S, *et al*. Proposed NIST standard for role-based access control[J]. ACM Transactions on Information and System Security, 2001,4(3):224 ~ 274

[7] Thomas R K, Sandhu R. Task-based authorization controls (TBAC): models for active and enterprise-oriented authorization management[A]. In: Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI: Status and Prospects[C]. London: Chapman&Hall, 1998. 262 ~ 275

[8] Thomas R K, Sandhu R. Task-based authorization: a research project in next-generation active security models for workflows [EB/OL]. http://lsdis.cs.uga.edu/ activities/NSF-workflow/roshan.html, 1996-4-16/2003-6-20

[9] Thomas R K. Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments[A]. In: Proceedings of the Second ACM workshop on Role-based Access Control[C]. New York: ACM Press, 1997. 13 ~ 19