基于 RBAC 的多等级移动 Agent 系统访问控制机制①

Role-Based Access Control for Multi-Grade Mobile Agent System

李爱宁 赵泽茂 (杭州电子科技大学 通信工程学院 浙江 杭州 310018)

摘 要： 针对 Agent 系统网络结构的多等级特点，在 RBAC96 模型的基础上，提出了一种基于角色的多等级 Agent 系统访问控制机制。通过制定多等级访问规则，划分 Agent 服务器中资源信息的属性和访问权限的级别，保证了 Agent 系统的多等级性；通过认证机制对来访 Agent 的身份、等级进行认证，保证了 Agent 系统的安全；通过角色作为中介配置用户和权限，提高了权限配置的效率和 Agent 系统灵活性。该机制算法已由脚本语言 python 编程实现。

关键词： 移动代理 多等级 访问控制 角色

1 引言

移动 Agent 是一种在分布式系统或协作系统中能持续自主地发挥作用的计算实体，具有智能性、代理性、移动性、主动性、协作性等特点[1]。移动 Agent 技术在资料搜集、指令下达、上下级状态监控等方面发挥着不可忽视的作用，在银行系统、电子商务、军事等领域具有积极的应用前景。

随着移动 Agent 技术在电子商务、分布式计算、信息搜索等领域的广泛深入，Agent 系统的安全问题变得日益重要。在移动 Agent 系统中，访问控制机制主要是用来保护 Agent 服务器中资源的安全，访问控制机制保证合法授权的移动 Agent 如何访问、获取他们所需要的 Agent 服务器中资源，同时保证 Agent 服务器中的资源免于被未授权的移动 Agent 访问[2]。

目前国内外出现的移动 Agent 系统访问控制策略有：基于存取控制列表 ACL 的访问控制方法；基于 PKI_PMI 属性证书的访问控制方法；基于移动 Agent 的协同访问控制方法；基于角色权限的访问控制方法等等[3-6]。但是这些访问控制策略，或者设计复杂或者不具有针对性，对于一些特殊的 Agent 系统并不适用。例如银行、军事等系统，它们与普通的分布式网络系统不同，有如下特点：

网络系统各节点存在着等级划分，具有不同的操作权限。比如，在银行系统中高级节点可以通过 Agent 对它的次级以及更低级的节点中的资源进行搜集、调用、修改等操作；而低级节点对高级节点或同级节点之间的操作会受到限制，低级节点只可以向高级节点汇报自己的状态，而无权获取高级节点或同级节点中的资源数据。由于系统结构的特殊性，必然导致这种多等级 Agent 系统的访问控制机制与普通的 Agent 系统不同。针对这种多等级移动 Agent 系统，在 RBAC96 模型的基础上，本文提出了一种基于角色的多等级 Agent 访问控制机制。

2 RBAC96模型

RBAC(Role-Based Access Control)，基于角色的访问控制，由 David Fenaido 和 Richard Kuhn 提出，是近年来 在信息安全领域访问控制方面的研究热点和重点。所谓角色，就是一个或一群用户在组织内可执行操作的集合，用户在一定的部门中具有一定的角色(比如，会计、出纳等)，其所执行的操作与其所扮演的角色的职能相匹配。在 RBAC 策略中，系统定义了各种角色，每种角色可以完成一定的职能，不同的用户根据其职能和责任被赋予相应的角色，一旦一个

① 基金项目:国防预研项目(513060401)
收稿时间:2008-10-29

用户成为某角色的成员，则此用户可以完成该角色所具有的职能。

RBAC96 模型[7,8]如图 1 所示，是由 Ravi Sandhu 和他领导的位于George Mason大学的信息安全技术实验室(LIST)于 1996 年提出的，该模型将传统的 RBAC 模型根据不同需要拆分成四种嵌套的模型并给出形式化定义，极大提高了系统灵活性和可用性，对 RBAC 进一步的深入研究奠定了基础。

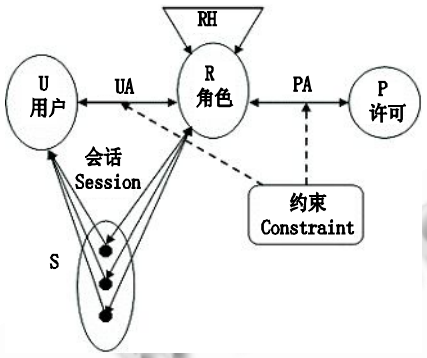

图 1　RBAC96 模型

下面对 RBAC96 模型做一个简单描述：图中模型基于三个实体集，用户集(U)，角色集(R)，权限集(P)。直观讲，一个用户是一个人或一个代理 Agent；一个角色是组织中一项工作或者一个职位；一个权限是对系统中一个或多个敏感对象进行某种方式访问的一个许可。

UA(User Assignment)是指用户分配，是从用户集合到角色集合的多对多映射，反映用户被分配的角色集。PA(Permission Assignment)，是从权限集合到角色集合的多对多映射，反映许可权对应的角色集。SA(Session Assignment)，是从会话集合到用户集合的函数映射，反映会话与用户的从属关系。RH(Role Hierarchy)，它表示角色层次关系，它是一种偏序关系，也计作：$\in$，表示继承，例如 $x \in y$ 表示角色 x 继承了分配给角色 y 的所有权限。S 是所有会话(session)的集合。

图中的双向箭头表示一个会话可以同时激活多个角色，用户的有效权限取决与会话中激活的角色所有权限的并集。图中单箭头表示，每个会话与单个用户关联。这个联系在会话的整个周期中是恒定的。一个用户可能同时有多个会话。

图中还定义了约束集(Constraint)，约束可以应用在先前定义的许多集合上，比如当一个用户拥有一个角色，那么它就不能在拥有另外一个角色了，这种角色互斥就可以看作一种约束。

3　访问控制机制设计实现

基于角色的多等级 Agent 访问控制机制，是在 RBAC 96 模型的基础上，针对系统的多等级性特点，通过制定多等级 Agent 系统的访问规则，建立认证机制以及制定相关的访问控制策略的基础上设计建立的。

3.1 多等级 Agent 系统的访问规则

在多等级 Agent 系统中各节点存在等级的划分，各级节点间有不同的访问权限。访问规则如下：等级高的节点可以通过 Agent 对它的次级以及更低级的节点中的资源进行搜集、调用、修改等操作；低级节点只可向高级节点汇报自己的状态，而无权获取高级节点或同级节点中的资源数据。即符合如下规则：

不上读，主体不可以读安全级别高于自己的客体。

下读下写，主体可以读安全级别低于自己的客体，并且可以将信息写入安全级别低于自己的客体。

同级不读不写，主体对处于与自己相同级别的客体不具有读、写权限。

3.2 访问控制系统总体结构

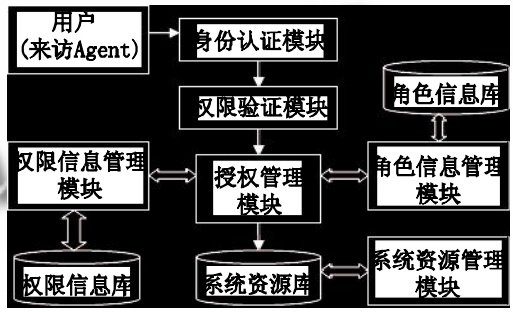

图 2　访问控制系统总体结构

访问控制系统的总体结构如图 2 所示，图中划分了几个功能模块，下面分别对每个模块的功能进行简单的描述。

(1)身份认证模块

身份认证模块主要是用来完成以下三方面的任务：识别来访 Agent 的身份，判断移动 Agent 是否为合法的。判断 Agent 来自哪个节点，识别出该节点的等级。判断该 Agent 的用途。

对于来访 Agent 的身份认证我们通过认证机制实

现。通过本机制，对 Agent 的身份、等级以及用途进行识别。

(2)角色/权限验证模块

角色/权限验证模块，主要是对访问用户(来访 Agent)的身份、用途进行验证划分，以确定该用户的角色信息以及与角色相匹配的权限信息。

(3)角色信息管理模块

角色信息管理模块的功能主要是用来定义和与创建系统中的角色，并对角色信息库中的角色信息进行管理配置，依据 RBAC96 模型，系统管理员可以抽象出系统中存在的角色以及各个角色之间的相互关系，并可以根据需要实时动态的创建和删除角色，角色集用 Role={role1，role2，role3...}来表示。

(4)系统资源管理模块

系统资源管理模块主要用来对各级节点中的被访问资源的属性进行划分。根据客体资源的类型(如文件，进程等)或者应用领域(如公告，保密资料等)进行分类。这样角色的授权就可以建立在抽象的客体分类的基础上，而不是具体的某一个客体。这样对每一个客体的访问授权会自动按照客体的分类类别来决定，不需要对每一客体都具体授权，使得授权管理更加方便、容易控制。

(5)权限信息管理模块

权限信息管理模块主要是对权限信息库中的权限信息进行管理，定义、管理系统中的权限。权限集用 Permission={p1，p2，p3...}来表示。系统中将权限定义为某种系统资源与某种操作的绑定，管理员可以通过对资源和操作的绑定，定义系统中的某种权限，并实现权限信息的修改与删除，达到权限管理的目的。

(6)授权管理模块

授权管理模块主要是用来实现角色与权限的配置和管理，即给不同的角色配置不同的权限，建立起二者的映射关系。

3.3 基于角色的多等级 Agent 访问控制系统实现

3.3.1 定义访问主体 Agent

在 Agent 系统中，Agent 是一段可执行的程序，它由系统中的 Agent 服务器平台创建、管理、删除，可根据需要进行各种复杂工作，如进行资料搜集、指令下达、状态监控等操作。

系统定义的 Agent 包含以下属性：用于标识来源节点身份的数字签名 DG(digital signature)，用途标识 DC(Duty Code)，等级标识 GC(Grade code)，可执行脚本程序 EP (Execute program) 及携带的信息 M(message)等。由 python 语言描述如下：def agent()：

Agent=str(Node Name)+str(Grade code)+str(Grade code)+
str("用于执行任务的程序代码 Execute program")/str(Message)

3.3.2 访问主体身份识别策略

在来访 Agent 对资源客体访问之前，被访问节点要首先对 Agent 的身份进行识别，辨别他来自哪个节点，并确定他的等级。身份识别过程如下：

(1) 身份认证模块收到来访 Agent 后首先判断 Agent 是否是来自本系统内部的合法节点。合法的来访的 Agent 会携带有节点的数字签名，根据此数字签名对 Agent 的身份进行确认，并进一步判断出节点的等级。

(2)确认出 Agent 的身份后，根据来访 Agent 中的用途标识符 DC，对照身份认证服务器中的用途标识符列表 DC={ L1，L2，L3，L4，L5，...}，对 Agent 的用途进行判断，以便于对其分配相应的角色。

表 1 Agent 用途列表

Agent 用途
L1：Collecting information(搜集信息)
L2：work report(汇报工作)
L3：Problem solving(排除故障)
L4：Transmit information(传达信息)
L5：State examination(核查节点状态)
…

3.3.3 角色/权限定义策略

判断即来访 Agent 的身份等级和用途之后，根据访问主体职责，进行用户角色分配 UA 操作，建立用户与角色集 Role 的映射函数，给用户分配角色。对于同一个用户可以分配不同的角色，其中每种角色可以完成一定的职能。角色信息管理服务器负责管理角色的创建、增添以及删除。角色具有名称和描述两个基本属性，比如在银行系统中可以定义如下角色集：

表 2　Role 集列表

Role 集
role1：department manager role(部门经理)
role2：　auditor role(审计员)
role3：　cashier role(出纳员)
role3：　accountant role(会计)
…

根据访问主体 Agent 的用途不同，对其分配不同的角色。如果，在实际的应用中，当系统的功能发生变化或演进时，如果某些角色失去了所需的操作功能，则需要删除角色的旧功能、增加新功能，或者定义新的角色。

权限由权限信息管理服务器进行定义划分，在本系统中将权限划分如下：Read，Execute，Write 三种类型。如表 3 所示：

表 3　Permission 权限列表

Permission 权限	
P1：Read	读取信息、数据
P2：Execute	管理、执行、删除、创建
P3：　Write	存储信息

角色作为连接用户和权限的中间媒介，一方面既是用户的集合，另一方面又是权限的集合。如下图 3 所示：

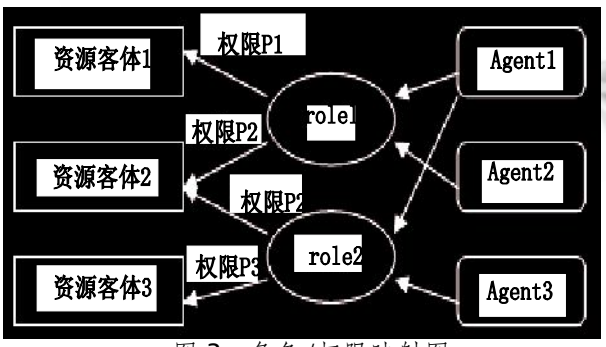

图 3　角色/权限映射图

比如，Agent 的用途为某央行对下级银行省行的信贷金额进行账目审计，则通过会话，建立 UA 映射关系,对此 Agent 需要分配两个角色：role1(信贷部经理)和 role2(审计员)。通过 role1 的管理权限调取信贷账目数据，然后通过 role2 的审计权限完成对账目的审计操作。

3.3.4 角色/权限约束规则

角色/权限约束规则主要是用来限制用户和权限之间指派的一种手段。在本访问控制机制中主要定义了如下约束规则：

user-user 冲突：几个用户不能同时指派给相同的角色。按照用户的申请时间排序，即遵循“First come，First get”的原则。同时，为防止一个用户长时间占用同一个角色，设定了“最长时限规则”。

privilege-privilege 冲突：几个权限不能同时指派给相同的角色，每个角色在一次操作中，只能有一类操作权限。

user-role 冲突：一个用户可以同时拥有多个角色，但是不能在一次会话中同时激活它们，仅当一次会话完成后，才能再进行下一次的用户角色会话。

角色继承：角色继承是指一个角色完全继承另一个角色的所有权限，因而对角色的授权可以通过角色继承的方式来实现。一般而言，角色所包含的权限是指所有直接赋予角色的权限以及所有他的父角色（包括父角色的父角色）的所有权限的并集。当一个角色继承了另一个角色后，它不仅继承了这个角色的权限，还继承了一些与其相关的约束。

角色依赖：是指若将一角色授予某一主体则必须满足这个主体已经具有另外一个角色的授权。角色依赖是可传递的。

最小权限原则：是指用户所拥有的权利不能超过他执行工作时所需的权限。当一个 Agent 访问某节点资源时，如果该操作不在 Agent 当前活跃角色的授权操作之内，该访问将被拒绝。

3.3.5 资源属性划分策略

作为 Agent 系统中的被访问客体，为便于对其进行管理需要对资源的属性进行定义分类。在本文的访问控制机制中，我们定义的资源的属性包括以下几个方面：资源的生命周期 life cycle，资源类型 resource type 以及操作 operation。

(1) 资源的生命周期 life cycle，是指 Agent 服务器资源的创建日期，保存时间等。

(2) 资源的类型 resource type，是指资源是动态的或者是静态的，可重用的或者是不可重用的，共享的或者互斥的等。

(3) 对资源的操作 operation，主要是指对资源的操作，比如可以对资源进行读、写、删除、修改等操作。

3.3.6 系统实现框图

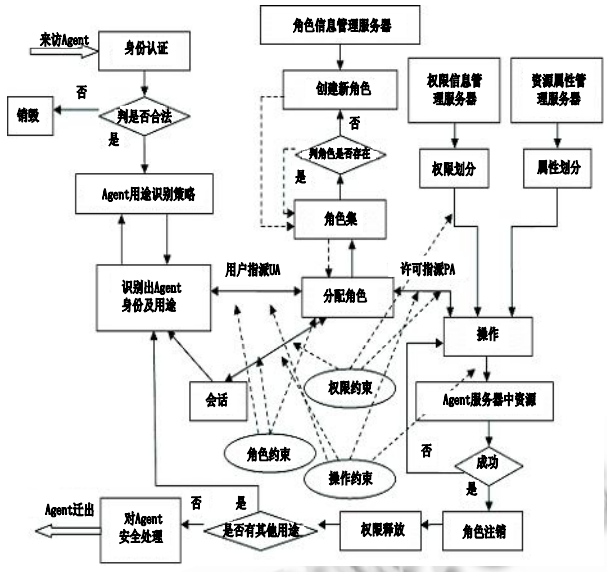

图 4　访问控制系统实现框图

访问控制系统实现框图如图 4 所示。系统的实现基于以上所定义的各种访问策略基础之上。实现过程的简单描述如下:

(1) 接收来访 Agent

(2) 对 Agent 的身份、来自节点的等级进行确认，启动: Personal Identify 和 Grade 函数。

(3)对 Agent 的用途确认，启动: Function Identify 函数。

(4)建立 Role 和 Agent 的会话: Session< role, Agent>，为 Agent 分配角色，并为角色指派权限

(5)分配的角色，根据指派的权限对资源进行访问。

(6)对资源访问完成之后，进行角色注销和权限释放。

(7)对完成访问任务的 Agent 进行安全处理。

(8)来访 Agent 迁出。

4　系统性能分析

本机制通过认证机制对来访 Agent 的身份、等级进行识别，保证了 Agent 服务器中的资源免于被未授权的移动 Agent 访问。通过所制定的 Agent 系统的多等级访问规则，实现了系统的多等级特点。通过添加角色定义策略、资源属性策略以及定义新的约束规则，可以有效控制用户、角色、权限三者映射关系，防止伪角色、非法权限的出现。

该机制支持授权管理，通过定义不同的角色作为沟通用户和资源的桥梁，当组织的功能变化或演进时，只需删除角色的旧功能、增加新功能，或定义新的角色，而不必更新每一个用户的权限设置，简化了对权限的管理。

此外，该机制与基于 PKI_PMI 属性证书的访问控制相比没有基于可信第三方，设计上简单，降低了实现的复杂度。

5　结束语

在 RBAC96 模型的基础上，针对系统的多等级特点，本文提出的基于角色的多等级 Agent 系统访问控制机制，实现了来自不同等级节点中的 Agent 对服务器资源的安全访问。该机制已有脚本语言 python 编程实现，实例表明，访问控制机制能有效保证移动 Agent 对服务器资源进行安全的访问。在实际应用中基于角色的多等级 Agent 系统访问控制机制，与我们课题组提出的 Agent 安全迁移机制相结合，可使多等级 Agent 系统的安全得到有效保证。

参考文献

1 周龙骧,刘添添.移动 agent 综述.计算机应用与软件, 2003,(11):19－23.

2 谭湘,顾毓清.移动 Agent 系统安全性研究综述.计算机研究与发展, 2003,40(7):984－992.

3 刘月琴,朱艳琴.基于角色 PMI 的权限认证模型.计算机应用与软件, 2008,25(2):103－105.

4 沈显君,杨进才,魏开平,等.基于 PKI/PMI 的移动代理安全访问.华中师范大学学报, 2004,38(3):284－288.

5 李栋栋,虎嵩林.一种统一授权和访问控制模型的设计和实现.计算机工程与应用, 2004,(6):80－84.

6 孙亚楠,石文昌,梁洪亮,等.安全操作系统基于 ACL 的自主访问控制机制的设计与实现.计算机科学, 2004,30(7):152－156.

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

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