基于属性的 RBAC 系统①

蒋凌志 (苏州高博软件职业技术学院 江苏 苏州 215163)

摘 要: 在 RBAC 系统中，各个用户所拥有角色的列表会存放在系统中，当用户进入系统前会进行验证检查，如果验证通过则让用户得到其所拥有的功能。但是这个验证工作对现今的有些应用略嫌不足，为此在原 RBAC 系统中加入了使用控制模块添加了多个属性，它允许验证、义务和条件三种决定策略去检查权限，让用户能在系统行进过程中去重新判断用户所能使用的功能。在一个 RBAC 系统中，在用户和被存取的对象中加入属性的概念可以让一些以前的RBAC系统经过改造获得新的应用,而保留RBAC本来的优点。

关键词: Attribute RBAC 使用控制

Attribute-Based RBAC System

JIANG Ling-Zhi

(Global Institute of Software Technology, Suzhou 215163, China)

Abstract: In the RBAC system, the list of each user's roles will be stored in the system. When users enter the system to verify checks before, they will be entitled if passed. However, the validation falls short of some applications today. For this purpose, a number of attributes are added to the control module in the original RBAC system, which allows authentication, obligations and conditions of the three decision strategies to check the permissions. In this way, users can re-determine their rights in the working process of the system. In a RBAC system, adding the concept of property to users can let some previous RBAC system adapted to acquire new applications, while retaining the original advantages of RBAC.

Keywords: attribute; RBAC; usage control

1 导论

在一个以角色为基础的访问控制(RBAC, Role-Based Access Control)系统中，按不同的权限来划分角色。并且一个用户可以被分配一种或多种的角色，系统根据其所分配的角色,让用户拥有相对应的权限。由于传统的访问控制只有针对授权(Authorization)作检查，对于现今有些系统的权限控制略嫌不足。美国学者 Jaehong Park 和 Ravi Sandhu 等人提出了使用控制(Usage Control, UCON)模型，UCON 用属性来判断用户目前所能使用的角色，达到更精确的控制。UCON 系统包含了各种可能的权限控制。要创建这一系统，工作量是非常大的。我们以前遗留的 RBAC 系统能否做一些改造，也能适应现在的需求? 答案当然是肯定的。我们在 RBAC 的系统中加入用户－属性、属性－角色这二种关系，可以让用户－角色关系更有弹性，有广泛的延伸。而且也可以作动态权责分离(Dynamic Separation of Duty)的检查。也就是说，用户可以同时拥有多种互斥的角色，在这样的系统中我们可以由用户属性 ，来决定用户目前拥有那一个互斥角色。

2 背景

2.1 Role-Based Access Control

以角色为基础的访问控制(Role-Based Access Control, 简称 RBAC)，由众多学者加以讨论改进后，

① 收稿时间:2009-04-14

最后由 NIST 组织加以整理后订出标准,主要是利用用户－角色－权限的三层关系来控制人员的权限，如图 1[1,2]。使得访问权限并不是根据用户的身份，而是用户通过角色来间接取得权限。因此我们在分派用户权限时，我们只要把相对应的角色分配给用户即可。

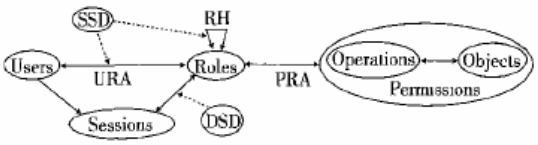

图 1　RBAC 模型

RBAC 架构共有六种基本组件：用户(User)、角色(Role)、权限(Permission)、执行期(Session)、用户－角色指派(URA)和权限－角色指派(PRA)。

Users：即欲使用对象的用户。

Roles：一般指的是组织中人员的职务，且会对应到某些权限，让拥有此角色的用户有相对应的权限。

Permission：可对某个对象操作或是存取的能力，如删除、修改。

Session：执行期，也就是用户在执行某一个任务的期间。

User－Role Assignment (URA)：用户和角色之间存在多对多的指派关系，同一用户可以拥有多个不同的角色，同一角色也可以指派给多个不同的用户。

Permission－Role Assignment (PRA)：角色和权利之间存在着多对多的指派关系，也就是说角色可以包含多个不同的权利，权利也可以被多个角色包含着。

Session－Roles：用户得到的角色集合对应到的任务期间。

User－Session：用户进入系统中，用户会对应到系统的人员集合的任务期间。在 RBAC 模块中是允许角色继承，角色间的继承也会继承角色所拥有的权限，因此不需要替每一个角色重新分配全部所需的权利，只需要分配没有继承的权利，可以降低管理上的负担。

2.2 Usage Control

由于现在的信息日新月异，传统的访问控制已经无法满足现今的系统 ，主要是因为传统访问控制只有针对授权(Authorization)决定策略来决定用户的权限，正因如此，Sandhu 和 Park 等人提出了使用控制模块来改善传统访问控制模块[3]。如图 2 所示：

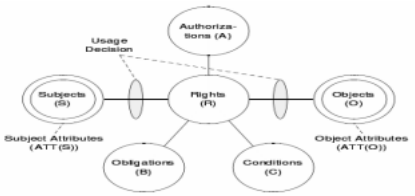

图 2　UCON 模型

使用控制模块架构共有八种基本元素：

(1) 主体(Subject)：通常是指用户。

(2) 客体(Object)：系统中被存取的资源。

(3) 权利(Right)：用户要存取对象必须拥有权利。

(4) 主体属性(Subject Attribute)：用户属性，使用属性值来做权限判断。

(5) 客体属性(Object Attribute)：用来跟用户属性比较，判断是否用户可以取得限。

(6) 授权(Authorization)：取决于用户属性和对象属性来做判断权限。

(7) 合约(Obligation)：用户必须要先触发某一事件。

(8) 条件(Condition)：通常是环境限制。

使用控制模块有二种特性[2]，一个是可变特性(Mutable Property)，描述 Subject 和 Object 的属性在存取的时候可以被改变。由图 2，我们可以得知，属性可能在三种状态下被 改变，pre-update 是指在存取前修改属性，ongoing-update 是指在运行中可以改变属性，post-update 是指在结束存取时改变属性。但是系统的属性只能被系统管理者修改。另一个是连续特性(Continuity Property)，是描述在存取时，可以随着时间或是触发了某个事件而会再次重新判断 Decisions 是否仍然有权利存取。决定策略可以在预先(before)和运行(Ongoing)中进行判断,而并不会在之后(after)判断，因为在这时决定策略并不会对使用有任合影响，而属性可能处于不变、先前更新、进行更新、之后更新，在先前决策时并不会有进行中的更新，因为这时还没有进到系统中。而条件的决定策略只有判断环境和系统状态，并不会更新属性。

3　RBAC and Usage Control

我们可以设想这样一个系统：图书管理系统，根

据登录的角色的不同，可以有后台管理，前台借书还书。这个系统用 RBAC 就可以胜任。后来对这个系统进行升级，前台可以在线阅读电子文档，在线阅读某个电子文档需花费每个小时多少积分等。那这个系统必须用 UCON 来实现。但是我们又不希望重新设计这个系统，能否在原来的系统上进行局部修改达到升级的功能?

3.1 在 RBAC 中引入 UCON 机制

在 RBAC 系统中是把系统内部分属于各分应用系统的权限管理机制统一纳入权限管理架构下管理。系统管理员能利用统一的管理界面管理用户帐户，其管理方式是采用用户－角色－权利－功能的对应的关系来管理,如图 1。对于具有相同角色的用户因其能登入的系统不一样 ，而使其能行使不同权限。

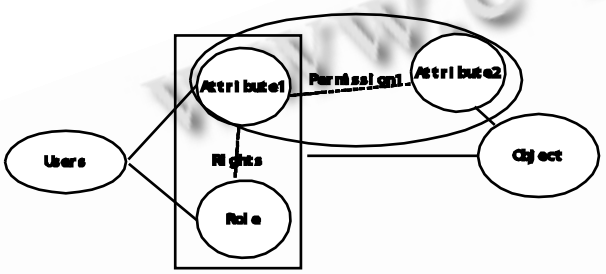

图 3 AB－RBAC 模型

当在 RBAC 中引入 UCON 机制时，会在 ROLE 与 OBJECT 中加入属性这个元素，成为基于属性的 RBAC(Attribute-based RBAC)，如图 3 所示。图中各元素与 RABC 中的同名元素含义相同。新增元素的含义如下:

Attribute1：用户属性，一个用户可以拥有多种不同的用户属性，用户会根据时间的不同或是触发的事件不同，而更新用户属性值,而更新后的属性值可以决定你是否还能行使这个角色的权限 。用户属性是一个决定角色的开关。

Attribute2：是系统对象的固有属性。

Permission：由 Attribute1 与 Attribute2 共同决定是否可对对象进行操作。

Rights：由 Attribute1 来做判断，决定是否启用角色给用户使用 。也就是说系统会由属性值的不同来决定用户所能使用的权限。 因此，我们在动态的系统中，可以事先指派角色给用户，而用户在系统运行过程中，可能会做某些事来更新用户属性 ,这时我们就可以利用用户的属性值来决定权限是否还能被用户使用。

Attribute-based RBAC 与原来的 RBAC 指派角色的方式相同 ，是用户－角色－权利－功能的方式，但是多了属性这一个元素。

3.2 属性定义

在本系统中，属性可以分为二种，分用户属性(Attribute1)和系统属性(Attribute2)。系统属性值通常是默认，而且不会被改变，如对象被存取所需花费的积分等。而用户属性值有二种，一种静态用户属性：用户可以要求系统管理者给予某一个值 ， 但是必须付出某种代价，例如用户登录一次系统时，系统就给用户增加一定的积分值。另一种为动态用户属性：当用户在执行时，才会取得用户属性值，当离开系统即会释放掉，例如“用户目前正在执行的角色”这个属性。 一个权利是否取得，是由用户属性，和系统属性来做逻辑判断 。 下列叙述使用属性判断权利的定义语法：

定义 1：进入系统前判断用户是否有权限

Ex：如果用户想要取得阅读文件的权利，用户的积分属性必须大(等)于文件阅读权限,用户才能取得权利。

GetStartPermission(r) => 用户积分≧文件的阅读权限。

定义 2：属性更新

Ex：用户想要在在线阅读一本书，必需花费 100 积分去阅读。

SetUpdate(s,n) =>更改用户的属性(用户积分 ，书本阅读积分(100))即当用户取得阅读权利后，会马上对其属性积分更新。

定义 3：系统中判断用户是否还有权限

Ex：用户想要在线阅读一本书，必需每小时花费 100 积分去阅读。

GetProcessPermission(s,n) =>(用户积分≧书本阅读积分(100 per hour))。

3.3 系统属性

系统属性可以是系统对象的属性，例如：在线阅读一本书需要 100 积分，100 积分就是读书操作的系统属性的值。在系统中，系统属性都是内建于数据库中，只能由系统管理员去修改。

3.4 用户属性

用户属性是附属于用户 ，用户可以拥有一至多个

用户属性 ，用户属性会随着用户触发了某一事件而作修改，如用户花了 10 积分去使用某功能，用户属性会立即更新，把相对应的用户属性扣除 10 积分。 因此，用户和用户属性会同时存在，当系统把角色指派给用户时，系统也会指派该角色的用户属性给用户，而用户属性的值，是由用户在系统执行中，视用户所执行的功能而做改变。

3.5 数据表

在此系统中我们规划了几张数据表来储存属性值。表 1 是属性总表，此表主要是存放系统中所有的属性。ATTRIBUTEID 是属性代码，每一个属性都会有一个唯一的代码。ATTRIBUTENAME 是属性名称，TYPE 是判断为用户属性或者是系统属性。RELATION 是代表使用者属性和系统属性的关系，每一个系统属性至少会对应到一个用户属性。USERID 为用户代码。ATTRIBUTEVALUE 为属性值。

表 1　属性总表

ATTRIBUTEID	ATTRIBUTENAME	TYPE	RELATION

表 2 是系统属性储存表，在这张表格中，会存放所有的系统属性 ，并且每个系统属性都有拥有一个属性值，一般而言，系统属性的值并不会被修改，若必须要修改，则必须由系统管理者来修改。

表 2　系统属性储存表

ATTRIBUTEID	ATTRIBUTENAME	ATTRIBUTEVALUE

表 3 是用户属性表，储放每位用户所拥有的属性 ，用户在系统中会不断的改变用户属性值，会立即更新此数据表。

表 3　用户属性值表

USERID	ATTRIBUTEID	ATTRIBUTENAME	ATTRIBUTEVALUE

3.6 权限检查

在系统中,我们实现了 预先决策 Pre-Authorization 和运行中决策 Ongoing-Authorization 两个模块,所以用户属性有 pre-update、ongoing-update 和 post-update 三种情况会去更新用户属性。由图 4 可以看出在系统中,会在系统进入前和运行中做权限的检查。用户在进入系统前，系统会检查用户属性，查看他是否有权限。比较用户属性表和系统属性表相对应的字段值，做预先决策，通过则允许进入系统。用户在系统中每隔一段时间或是用户触发事件时，用户属性值会随用户在系统中的变化而变化。这时也会立刻更新数据库的表格，系统会再做一次权限的检查，如果用户属性不符合系统属性则会被踢出系统，或取消某个权限。反之则会继续执行,直到用户结束系统。

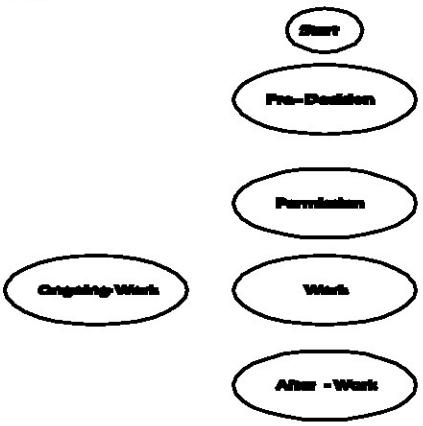

图 4　权限检查流程

4　结语

本文描述了，如何对一个现有的 RBAC 系统进行改造，引入 UCON 机制，分别在 ROLE 与 OBJECT 中添加了属性，达到了使用控制的一些功能。以达到对原有的 RBAC 系统进行升级可以满足现在系统的一些新需求。

参考文献

1 Sandhu RS, Coyne EJ, Feinstein HL, et al. Role-Based access control models. IEEE Computer, 1996,29(2): 38－47.

2 Sanhu R, Bhamidipati V, Munawer Q. The ARBAC97 model for role-based administration of roles. ACM-Transactions on Information and System Security, 1999,2(1):105－135.

3 Park J, Sandhu R. Towards usage control models: Beyond traditional access control. Proc. of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT02). Monterey: ACM, 2002.57－64.