文章编号:1000 - 582X(2003)11 - 0091 - 04

信息系统权限管理新方法及实现*

祖 峰,熊忠阳,冯 永
(重庆大学 计算机科学与技术学院,重庆 400044)

摘 要:提出了一种新的以基于角色的权限管理方法为主的信息系统权限管理方法(DERBAC),解决了现有权限管理方法存在适用情况限制的问题,使之比基于角色管理的模型具有更高的灵活性,弥补了基于角色管理的模型在系统规模较小时显得比较繁琐的不足,也克服了自主访问控制方法给系统规模扩大造成的不便。当信息系统规模发生变化时,能很好地提供有效的应用级权限管理。同时,还详细讨论了如何利用关系数据库来实现的方法。

关键词:信息系统;基于角色访问控制;自主访问控制;权限管理
中图分类号:TP309.2 **文献标识码**:A

随着计算机的普及和信息浪潮的冲击,将计算机引入到相关领域来提高效率已成为很多有识之士的共识。于是,很多信息系统被开发出来用于处理机关、学校、工厂每天都有的、越来越复杂的信息。但是,一个信息系统仅仅能处理这些信息是远远不够的,它还必须有可靠的控制管理机制,也就是通过对用户的权限进行管理,来防止信息的外泄和被破坏,使信息系统信息更安全。一个信息系统的用户权限管理有两个层次:一个是系统级,是由系统软件进行保障;另一个是应用级,就是开发信息系统时应该考虑解决的。

1 用户权限管理的常用方法

用户权限管理通常有 3 种方法:

1) 强制访问控制(Mandatory Access Control)。它是系统强制主体服从事先制定的访问控制政策。它预先定义了用户的级别,以及信息的级别,通过比较这两者的级别来进行合法性的验证。其缺点是这种事先的权限划分与现实可能存在一定的差距。同级的数据之间缺乏控制机制。而且,整个体系是事先制定的。要修改对一类信息的级别时比较繁琐[1]。

2) 自主访问控制(Discretionary Access Control)。它是在确认主体身份及所属组的基础上,对访问进行限定的一种控制策略,访问控制策略保存在一个矩阵中,矩阵的每个元素表示一个主体对一个客体的访问控制。它的优点是比较直观、易于理解,在规模较小时,它能显示系统的权限分配整体情况。但是,随着系统的扩大,它的规模成几何级数增长,使系统变得异常复杂。当系统复杂时,对用户权限的修改变得复杂。这样维护系统将耗费大量的人力,而且正确性难于得到保证[1]。

3) 基于角色的访问控制(Role - Based Access Control)。它主要由 3 个实体构成:用户、角色和权限。角色跟用户之间是一个多对多的关系,权限跟角色之间也是一个多对多的关系。用户跟权限之间没有直接的关系。角色是根据一个组织内工作性质的不同来设定的,用户根据所负的责任被赋予某种角色,因而也就拥有了相应的权限,用户可以通过被赋予不同的角色,而拥有不同的权限。如果有新的内容加入,角色能被赋予新的权限。权限既能赋予也能撤销。如图 1[2] 所示。

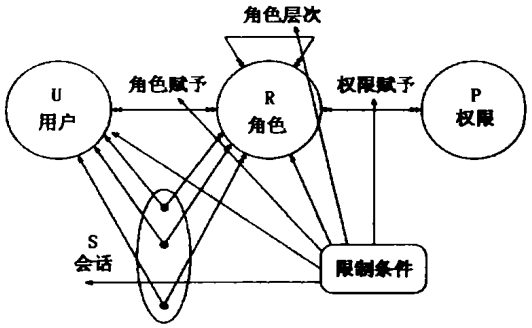

图 1 RBAC 模型

图 1 中用户跟角色之间的角色赋予用双箭头表示多对多的关系。角色与角色的层次关系以及角色跟权限之间的赋予关系,这两种多对多的关系也用双箭头

* 收稿日期:2003 - 06 - 10
作者简介:祖峰(1976 -),男,重庆人,硕士,主要研究方向:信息网络与信息系统。

来表示。会话集中每个黑点表示一个会话，与它对应的是一个用户以及多个角色。基于角色的访问控制有易于理解、易于管理、支持权限的继承、比 DAC 能支持更大的规模等优点。与用户和权限相比，角色是一个相对稳定的概念。与用户和角色之间的关系变化速度相比，权限和角色的关系变化是比较慢的。也就是说，用户所"扮演"的角色可能经常发生变化，但是角色所拥有的权限是相对稳定的[3]。

2 新方法的基本模型

在过去的十几年里，RBAC 越来越受到人们的重视。由于 RBAC 很好地模拟了大多数机构的现实情况，使得整个的权限机制规划起来很容易。但是，如果系统用户规模较小，每种角色只有一个两个，这时候角色就成了一种负担。RBAC 中的角色是相对固定，有哪些角色、每个角色有哪些权限是在一定时间的使用以后固定下来的。但是，现实的情况总是多变的，不可能是一成不变的。一些情况下，还是要暂时对一个用户赋予某种权限，而这个没有现成的角色适合。这种情况下，如果先建角色，加入用户，撤销权限的时候还要删除这个角色的定义，这样显得比较繁琐。同时可能引出 2 个安全问题[4]：1）角色权限赋予的时候出错，权限过大；2）其他用户由于某种原因意外加入到该角色中。如果 2 个问题同时出现，危害就更大。而如果在 RBAC 中引入直接用户权限，用户可以直接跟权限发生联系，就可以减小这些问题可能造成的影响。而且，由于引入用户直接权限，使得 DAC 可以被看作新模型的一个简化，当系统规模较小的时候可以只用到简化部分，弥补了 RBAC 在小系统中显得比 DAC 繁琐的问题。同时，它提供了升级的空间，使得系统规模扩大的时候，权限管理部分不需要重新规划以及做任何修改，使整个模型在各种系统中都能很好应用。同时，修改了原模型中角色、对象和操作三者之间不合理的关系。将原有的（Role x Object（ Role x Operation）关系改为（Role x Object x Operation），这种模型称作自主增强型基于角色存取控制模型（Discretionary Enhanced RBAC）。在保留 RBAC 基本模型的基础上新增加了用户权限的赋予，使用户直接跟权限产生联系。用户跟权限也是多对多的关系，如图 2 所示。与图 1 的区别在于细化了权限的定义，定义了多对多的对象与操作的关系。还有就是增加了用户跟对象之间的双箭头来表示它们之间的多对多关系。

其定义如下：

u 用户

Ri，Rj 角色

o，oa，ob 对象

op 操作

UrPre[u，o，op] 对对象 o 有执行 op 操作的权限的用户

RoPre[i，o，op] 对对象 o 有执行 op 操作的权限的角色

UrDPre[u，o，op] 对对象 o 有执行 op 操作的直接权限的用户

Role[o，op] 与对象 o、操作 op 相关的角色的集合

AcRole[o，op] 与对象 o、操作 op 相关的已激活的角色集合（就是当前会话中使用的角色的集合）。

M[i] 与角色 i 相关的授权用户。

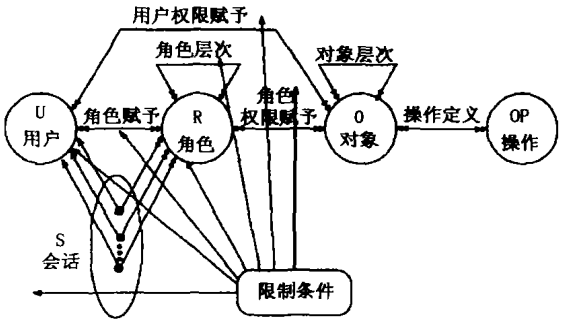

图 2 DERBAC 模型

2.1 用户权限的确定

用户不再只能通过角色才能获得一定权限，而是可以直接获得一定的权限。用户是否有某种权限要看他自身是不是有权限以及他所"扮演"的角色是不是有权限来决定。这种情况获得的权限被称作特权。因而，一个用户有某种权限简单表达式如下：

$$UrPre[u,o,op] = \exists(u,o,op)(UrDPre[u,o,op] \vee (RoPre[Ri,o,op] \wedge u \in M[Ri]))$$

因为，在 RBAC 中并没有定义人们所谈论的"负规则"。"负规则"就是每条规则得到的都是对某个主体的禁止访问而不是允许。在 RBAC 中是通过加限制的办法来实现相似功能的。这样一来就有一个权限冲突问题：用户的特权跟用户通过角色获得的权限，以及用户所"扮演"的不同角色所定义的权限可能是相互矛盾的。对于用户角色之间的冲突，可以通过会话的方式来解决，由于同一时刻一个用户只能激活一个会话，根据动态任务划分原则，在同一个会话中能激活的角色之间是不相关的。因而，角色之间的冲突是不会发生的。对于用户特权跟用户角色权限之间的冲突，定义了一个规则：用户特权优先。在用户特权有定义的情况下，权限就是用户特权，用户特权没定义时，用户角色权限才发挥作用。表达式如下：

$$UrPre[u,o,op] = \exists(u,o,op)(UrDPre[u,o,op] \wedge (\neg(UrDPre[u,o,op](RoPre[Ri,o,op] \wedge u \in (M[Ri]))$$

2.2 角色的赋予

角色与用户组的区别在哪儿呢? 用户组作为一组共有权限用户的集合在很多系统中应用。角色跟用户组的主要区别在于用户组是作为一组用户的集合,而角色一边是用户的集合,另一边是权限的集合。角色是作为把用户跟权限联系起来的中介。如果花两倍于确定组成员的时间来确定一个组的权限,并且组的权限和组的关系只能由系统管理员来修改的时候,那么组就跟角色很类似了。角色有 2 个特征:1)确定扮演角色的人跟确定角色的权限难易程度相当;2)对角色的赋予和角色权限的赋予应该集中于几个人。角色跟用户是多对多(U x R)的关系。角色赋予的效果可以描述为:

$UrPre[u,o,op] = \exists (Ri,u,o,op)(RoPre[i,o,op] \wedge u \in (M[Ri])$

但是由于限制条件的引入,角色的赋予就要受到一定的限制。这样,能够防止通过非法的授权而设置后门。限制,主要在新增用户、角色关系时体现。

2.3 角色的层次

角色分层次能够使权限定义更加精简,因为上级的角色不用重复的列出它从下级角色那儿继承来的权限[5]。角色跟角色的关系是一种偏序关系,也就是说这种关系是自反的、反对称和传递的。是自反的,因为角色能继承他自己的所有权限。是传递的,这个是自然的要求。是反对称的,因为避免了那种相互继承的情况。所有角色构成的是有向图而不是简单树。因为,有向图支持多个上级的角色继承一个低级角色的权限。如:一个雇员可能有管理员和项目经理两个上级角色,这两个角色都可以继承赋予雇员的权限。同样,它也支持一个上级角色继承多个下级角色。但是这个有向图中没有有向环,如果存在有向环,偏序关系就不会存在了[6]。角色层次的描述为:

$\forall (Ri,Rj,o,op)(Rj \in AcRole[o,op] \wedge Ri \preceq Rj \Rightarrow Ri \in AcRole[o,op]) \wedge (Rj \in Role[o,op] \wedge Ri \preceq Rj \Rightarrow Ri \in Role[o,op])$

2.4 权限的描述

虽然,在 RBAC 模型中将权限描述成一个实体,但是,这个描述太笼统了,不是很直观。权限一般被描述为"对一个对象有执行什么操作的权力"。这样就自然将原有的一个实体分化成两个实体。一个是要保护的对象,另一个是目的操作。他们之间也是多对多的关系。这样就可以更方便地来描述对象权限了。角色的权限可以描述为:

$RoPre[Ri,o,op] = \exists Ri((Ri \in Role[o,op]) \wedge (Ri \in AcRole[o,op]))$

2.5 受保护对象的层次

对受保护的对象进行分层次可以使权限定义更加精简,不需要重复列出子对象从父对象那儿继承来的权限。对于新加入的对象只要确定了它的父对象以后就自然有了对它的权限,省去了对其做权限规划这种繁琐工作。而且,不需要附加的机制来保证权限结构的完整。对父对象权限收回的同时,对子对象的权限也自然收回了。受保护对象之间的关系跟角色类似也是偏序关系,所有受保护对象构成的也是有向图,但是没有有向环。受保护对象的层次的描述为:

$\forall (Ri,oa,ob,op)(Ri \in AcRole[oa,op] \wedge oa \preceq ob \Rightarrow (Ri \in AcRole[ob,op])) \wedge (Ri \in Role[oa,op] \wedge oa \preceq ob \Rightarrow Ri \in Role[ob,op])$

2.6 操作的定义

操作的定义相对简单,对于每个对象的不同操作数目有限,因此不再分层次来进行管理。在选取操作类型时,尽量选择相互独立的操作。单独的操作没有实际意义,同对象联系起来才有完整的意义。

3 新方法在信息系统中的实现

现在的信息系统,大多后台都有关系数据库支撑。如果将上述模型用关系数据库来实现具有普遍的意义,并能发挥巨大的作用。关系数据库存放的数据是以条目为基础的。因此,在实现的时候,选用基于规则的实现方法就很自然。数据库中的每条数据就是对一个权限规则的描述。而规则规定了如何为一个用户赋予角色,以及如何对角色或者用户赋予权限。这样这些规则就能用来进行权限控制。

建表的时候很自然地将 4 个实体建成 4 个表(用户信息表、角色信息表、对象信息表、操作表)。描述用户跟角色的多对多关系就有了用户角色关系表。按照 RBAC 的要求对受保护的对象进行统一编码,并将角色以及用户同受保护的对象一起编码。这样就用一个统一的权限规则表来存储角色权限规则和用户权限规则。受保护的对象之间的关系也用一个表(对象关系表)来存储。因为角色之间的关系同受保护对象的关系相同,角色与受保护对象的表示方法相似,所以角色之间的关系也存在对象关系表中。限制分别用两个表来存储,用用户角色限制表来限制用户跟角色之间的关系,用角色权限表来限制用户以及角色所能获得的权限。

数据库的表结构如图 3 所示。

这里只列出了每个表的一些基本项目,应用的时候可以根据实际情况进行增删。通过验证合法的用户,通过用户角色关系表就能获得所"扮演"的角色。而对用户跟角色关系的限制是在把一个用户加入到一个角色去的时候,查找他有没有违反用户角色限制表中定义的规则来实现的。在用户角色限制表中加入新

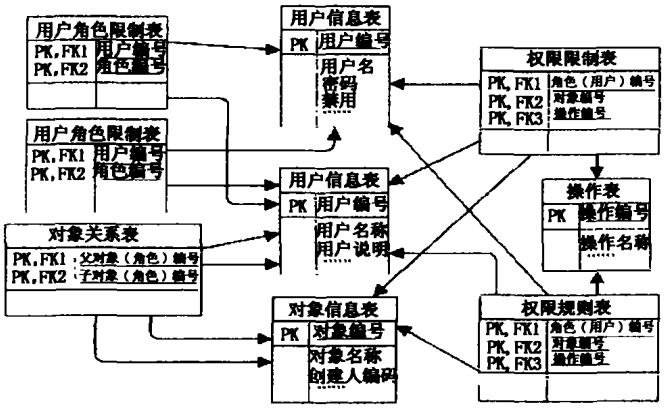

图 3 数据库表结构图

条目的时候,要同时删除用户角色关系表中已经存在的违反新加入的规则的数据。当生成一个新的会话时,就确定了用户"扮演"的角色,然后通过对象关系表找出角色的子角色编号,将他们合并起来作为鉴权对象集。然后,在权限规则表中,先选出直接有权的对象的编号和操作编号,然后通过对象关系表选出所有有权对象的编号以及操作编号。在要鉴权的时候,只要看比对对象的编号和操作的编号就可以了。权限限制表的应用方式与用户角色限制表相同。这样就形成了一套完整的信息系统的权限管理方法。

4 结束语

在实际应用中,受保护的对象可以是一个窗口、一个页面、一张表或者一个控件等,对象的选取比较灵活。具体的权限规则的定义在系统的应用过程中都还可以修改以致重建。在系统较小时可以不用角色,全用用户直接权限;当系统较大时,更多的使用角色权限。权限管理体系结构不会随着系统规模的增长而发生变化,可以平滑过渡,也使得自主增强型基于角色权限管理的机制适应能力较强。而且,通过在权限规则表中引入规则的有效时间区间,对一个用户能扮演的角色的总数、角色之间关系的传递次数以及受保护对象之间关系传递的次数加以限制,就能使整个权限管理机制更完善。所用的表结构充分考虑了这方面扩展的可能性,这套权限管理机制已成功应用到某省级热线的发布系统中。

参考文献:

[1] 叶锡君,许勇,吴国新. 基于角色访问控制在 Web 中的实现技术[J]. 计算机工程,2002,28(1):167 - 169.

[2] 张大江,钱华林. 一个利用数字证书实现的 RBAC 模型[J]. 小型微型计算机系统,2001,22(8):936 - 939.

[3] SANDHU R S, COYNE E J, FEINSTEIND H L, et al. Role - Based Access Control[J]. Models IEEE Computer,1996, 29(2):38 - 47.

[4] 张晓辉,王培康. 大型信息系统用户权限管理[J]. 计算机应用,2000,20(11):35 - 36.

[5] BARKLEY J F, CINCOTTA A V, FERRAIOLO D F,et al. Role Based Access Control for the World Wide Web[A]. NIST/NCSC, Proc 20th NIST-NCSC National Information Systems Security Conference[C]. USA: NIST/NCSC,1997.

[6] 左孝凌,李为鑑,刘永才. 离散数学[M]. 上海:上海科学技术文献出版社,1982.

New Privilege Management Method in Information System and Its Implementation

ZU Feng, XIONG Zhong-yang, FENG Yong

(College of Computer Science and Technology, Chongqing University, Chongqing 400044, China)

Abstract: A method of privilege management in information system-Discretionary Enhanced Role-Base Access Control (DERBAC) is presented and the applicable restriction problem of existing methods in dealing with Privilege management is solved. DERBAC is based on role-based access control. Compared to traditional RBAC, DERBAC is more flexible and efficient. It atones for the drawback that the traditional one is not fit for small system. It also conquers the inconvenience of upgrading the system with DAC. This method is efficient in any system whether it's large or small. On the condition of system size changing, provides efficient application - level privilege management. How to implement DERBAC with RDBMS is also introduced.

Keywords: information system; RBAC; DAC; privilege management

(编辑 张 苹)