IEEE计算机，第29卷，第2号，1996年2月，页 38-47.

基于角色的访问控制模型[Z]

Ravi S. Sandhu[*], Edward J. Coyne[k], Hal L. Feinstein[k]和Charles E. Youman[k]

于10月26日修订， 1995

摘要本文介绍了基于角色的访问控制（RBAC）的参考模型族，其中权限与角色相关联，并且用户已成为相应角色的成员。这大大简化了权限管理。角色与访问控制中用户组的概念密切相关。但是，角色将一侧的一组用户和另一侧的一组权限聚集在一起，而用户组通常仅定义为一组用户。

RBAC的基本概念起源于早期的多用户通信。
推杆系统。对RBAC的兴趣重新兴起的原因是对RBAC的通用可定制设施的需求以及对RBAC本身的管理进行管理的需求。结果，RBAC设施从简单到复杂。本文介绍了一种参考模型的新颖框架，以系统地解决RBAC的各个组成部分及其相互作用。

关键字：安全性，访问控制，角色，模型

该论文已被IEEE Computer接受发表。

所有信函均应发送至弗吉尼亚州费尔法克斯，乔治梅森大学，ISSE系MS 4A4 MS. Ravi Sandhu教授，弗吉尼亚州22030。电话：703-993-1659，传真：703-993-1638，电子邮件：sandhu@isse.gmu.edu.

这项工作的部分经费来自合同的50-DKNA-4-00122和50-DKNB-5-00188。
美国国家标准技术研究所。国家科学基金会（National Science Foundation）提供的CCR-9503560赠款也支持Ravi Sandhu的工作。

[*]SETA公司和乔治梅森大学

[k]SETA公司

1 介绍

基于角色的访问控制（RBAC）的概念始于1970年代首创的多用户和多应用程序在线系统。RBAC的中心概念是权限与角色相关联，并且用户被分配了适当的角色。这大大简化了权限管理。为组织中的各种工作职能创建角色，并根据用户的职责和资格为用户分配角色。可以轻松地将用户从一个角色重新分配给另一角色。合并新的应用程序和系统后，可以授予角色新的权限，并且可以根据需要从角色中撤消权限。

角色被正确地视为语义构造，围绕该语义构造了访问控制策略。角色将用户和权限组合在一起的特定集合是暂时的。由于组织的活动或职能通常不那么频繁地更改，因此角色更加稳定。

下面讨论了构建角色的几种不同动机。角色可以代表执行特定任务的能力，例如医师或药剂师。角色可以体现权限和责任，例如项目主管。权威和责任与能力不同。简·多伊（Jane Doe）可能有能力领导多个部门，但已被指派领导其中一个部门。角色可以完成通过多个用户（例如值班医师或值班经理）轮流工作的特定职责分配。RBAC模型和实现应能够方便地容纳角色概念的所有这些体现。

NIST [1]的最新研究表明，RBAC解决了商业和政府部门的许多需求。在对28个组织的这项研究中，发现访问控制要求受到多种因素的驱动，包括客户，股东和保险人的信任，个人信息的隐私，防止未经授权分配金融资产，防止未经授权使用长途电话电路以及遵守达到专业标准。研究发现，许多组织基于“个人用户在组织中所扮演的角色”来进行访问控制决策。许多组织倾向于集中控制和维护访问权限，而不是系统管理员个人决定，而是更多地根据组织的保护准则进行。该研究还发现，组织通常将其访问控制需求视为独特的，并认为可用产品缺乏足够的灵活性。

对RBAC的浓厚兴趣的其他证据来自标准领域。基于角色在Oracle 7中的实现，正在考虑将角色作为新兴的数据库管理系统SQL3标准的一部分。角色也已纳入通用标准[2]的商业安全配置文件中。RBAC还与当前的技术和业务趋势非常匹配。许多产品直接支持某种形式的RBAC，而其他产品则支持可以用于实现角色的紧密相关的概念（例如用户组）。

尽管公认的RBAC概念有用，但对于RBAC的含义却几乎没有共识。结果，RBAC是一个无定形的概念，被各种研究人员和系统开发人员以不同的方式解释，从简单到精巧和复杂。

本文介绍了由作者开发的包含四个参考模型的新颖框架，以提供一种系统的方法来理解RBAC，并对其在不同系统中的实现进行分类。我们的框架还将RBAC的管理与控制数据和其他资源的访问分开。

2 背景和动机

RBAC的主要目的是促进安全管理和审查。许多商业上成功的大型机访问控制系统都扮演着安全管理的角色。例如，操作员角色可以访问所有资源，但不能更改访问权限，安全管理员角色可以更改权限，但不能访问资源，审核员角色可以访问审核记录。在现代网络操作系统（例如Novell的NetWare和Microsoft Windows NT）中也可以找到这种角色的管理使用。

近期对RBAC的兴趣重新兴起已集中在应用程序级别上对RBAC的一般支持。在过去和今天，特定的应用程序都是使用在应用程序本身内编码的RBAC构建的。现有的操作系统和环境几乎不支持RBAC在应用程序级别的使用。这种支持开始出现在产品中。面临的挑战是要确定足够独立，但易于实施和使用，与应用程序无关的设施，以最少的定制来支持各种应用程序。

RBAC的复杂变体包括建立角色之间以及权限和角色之间以及用户和角色之间关系的能力。例如，可以将两个角色建立为互斥的，因此不允许同一用户同时担任这两个角色。角色还可以承担继承关系，其中一个角色继承分配给不同角色的权限。这些角色角色关系可用于实施安全策略，包括职责分离和授权。迄今为止，这些关系必须被编码到应用软件中。使用RBAC，可以一次为安全域指定它们。

使用RBAC，可以预定义角色-权限关系，这使将用户分配给预定义的角色变得简单。NIST研究[1]指出，与角色用户成员身份的更改相比，分配给角色的权限倾向于相对缓慢地进行更改。该研究还发现，允许管理员授予和撤销现有角色用户的成员资格，而又不赋予这些管理员创建新角色或更改角色权限分配的权限。

将用户分配给角色通常比分配给角色的权限需要更少的技术技能。在没有RBAC的情况下，也很难确定已向哪些用户授权了哪些权限。

访问控制策略体现在RBAC的各个组件中，例如角色权限，用户角色和角色角色关系。这些组件共同确定是否允许特定用户访问系统中的特定数据。RBAC组件可以由系统所有者直接配置，也可以由系统所有者委派的适当角色间接配置。在特定系统中实施的策略是系统所有者指示的各种RBAC组件的精确配置的最终结果。此外，访问控制策略可以在系统生命周期内逐步发展，在大型系统中几乎可以肯定。修改策略以满足组织不断变化的需求的能力是RBAC的重要优势。

尽管RBAC不影响策略，但它直接支持三个众所周知的安全性原则：最小特权，职责分离和数据抽象。支持最低特权，因为可以配置RBAC，因此仅将角色成员执行的任务所需的那些权限分配给角色。通过确保必须调用互斥角色来完成敏感任务来实现职责分离，例如要求会计人员和客户经理参与签发支票。数据抽象是通过抽象权限（例如帐户对象的贷方和借方）而不是操作系统通常提供的读取，写入，执行权限来支持的。但是，RBAC无法强制实施这些原则。安全人员可能会混淆RBAC，因此违反了这些原则。同样，数据抽象支持的程度将由实现细节确定。

RBAC并不是解决所有访问控制问题的灵丹妙药。需要更复杂的访问控制形式来处理需要控制操作顺序的情况。例如，采购申请需要采取各种步骤才能导致发出采购订单。RBAC不会尝试直接控制此类事件序列的权限。为此，可以在RBAC上分层放置其他形式的访问控制。Mohammed和Dilts [3]以及Thomas和Sandhu [4]讨论了其中一些问题。尽管RBAC可以作为建立此类控制的基础，但我们认为操作序列的控制不在RBAC的范围之内。

3 角色和相关概念

一个常见的问题是，“角色和组之间有什么区别？”在许多访问控制系统中通常提供用户组作为访问控制的单位。组的大多数实现与

角色的概念是，组通常被视为用户的集合，而不是权限的集合。角色既是一侧的用户集合，又是另一侧的权限集合。该角色是将这两个集合合在一起的中介。

考虑Unix操作系统。Unix中的组成员资格由两个文件/ etc / passwd和/ etc / group定义。因此，很容易确定特定用户所属的组或特定组的所有成员。基于与单个文件和目录关联的权限位，将权限授予组。要确定特定组具有什么权限，通常需要遍历整个flesystem树。因此，确定组的成员资格比确定组的权限要容易得多。而且，对组的权限分配是高度分散的。本质上，Unix flesystem的任何子树的所有者都可以将该子树的权限分配给组。（完成此操作的确切程度取决于所讨论的Unix的特定变体。）但是，即使组与我们的角色概念不同，Unix组也可以用于在某些情况下实现角色。

为了说明组与角色区分的质性，请考虑一个假设系统，在该系统中确定组成员资格所需的时间是确定组权限的两倍。假定组权限和成员资格只能由系统安全管理员更改。在这种情况下，组机制将非常类似于我们的角色概念。

前面的讨论提出了角色的两个特征，确定角色成员资格和角色权限应该大致同样容易，并且角色成员资格和角色权限的控制应相对集中在几个用户中。许多声称基于角色的机制都无法满足这些要求之一或全部。

通常会问一个有关角色与隔离专区之间关系的问题。机舱是用于国防和政府部门的安全标签结构的一部分[5]。隔间基于“需要知道”的概念，该需求具有类似于隔室标签下可用信息的语义含义，类似于角色的语义含义。但是，隔室的使用是针对标签格中单向信息流的特定策略。角色不假定这种特定策略。

自由裁量访问控制和强制访问控制之间存在长期的区别，分别称为DAC和MAC。这种区别来自国防部门的安全研究。MAC根据附加到用户（更准确地说，附加到主题）和对象上的安全标签来实施访问控制[5]。DAC根据权限或拒绝或单个用户（通常是对象所有者）配置的权限或对象对对象实施访问控制。可以将RBAC视为访问控制的独立组件，并在适当时与MAC和DAC共存。在这种情况下，只有当RBAC，MAC，

和DAC。我们还期望在许多情况下RBAC会独立存在。

与此相关的是，RBAC本身是一种酌处还是强制机制？答案取决于RBAC系统中的全权委托和强制性的精确定义以及权限，角色和用户的精确性质和配置。我们理解强制性是指各个用户对于将哪个权限或用户分配给角色没有任何选择，而由各个用户自行决定是否做出这些决定。如前所述，RBAC本身是政策中立的。RBAC的特定配置可能具有强烈的强制性主张，而其他配置可能具有强烈的自由裁量权。

4 参考模型族

为了理解RBAC的各个方面，我们定义了四个概念模型。这四个模型之间的关系如图1（a）所示，其基本特征如图1（b）所示。基本模型$RBAC_0$位于底部，这表明它是任何自称支持RBAC的系统的最低要求。$RBAC_1$和$RBAC_2$都包含$RBAC_0$，但添加了独立的
功能。它们被称为高级模型。$RBAC_1$添加了角色的概念
层次结构（角色可以从其他角色继承权限的情况）。$RBAC_2$添加了约束条件（这限制了dif-
RBAC的有效组件）。$RBAC_1$和$RBAC_2$彼此无法比拟。合并模型$RBAC_3$包括$RBAC_1$和$RBAC_2$，以及传递性，$RBAC_0$。

这些模型旨在用作与其他研究人员和开发人员使用的系统和模型进行比较的参考点。它们还可以作为产品开发和潜在客户评估的指南。目前，我们假设只有一个安全人员，他是唯一有权配置这些模型的各种集合和关系的人员。稍后，我们将介绍一个更复杂的管理模型。

4.1 基本模型

基本模型$RBAC_0$由图1（b）的那部分组成，未与这三个高级模型之一标识。有三组实体，分别称为用户（U），角色（R）和权限（P）。该图还显示了会话（S）的集合。

此模型中的用户是人。可以将用户的概念概括为包括智能自主代理，例如机器人，固定式计算机，甚至计算机网络。为简单起见，我们将用户视为人类。角色是组织内的职务职能或职务，具有与授予角色成员的权限和责任有关的一些关联语义。

许可是对访问系统中一个或多个对象的特定模式的批准。文献中还使用术语授权，访问权限和特权来表示许可。权限始终是肯定的，并赋予权限持有者在系统中执行某些操作的能力。对象是数据对象以及由计算机系统内的数据表示的资源对象。我们的概念模型允许对权限进行多种解释，从非常粗糙的粒度（例如，允许访问整个子网）到非常精细的粒度（其中访问单位是特定记录的特定来源）。一些访问控制文献讨论了“否定权限”，它拒绝（而不是授予）访问权限。在我们的框架中，拒绝访问被建模为约束而不是否定许可。

许可的性质在很大程度上取决于系统的实现细节及其类型。因此，用于访问控制的通用模型必须在某种程度上将权限视为未解释的符号。每个系统都保护其实现的抽象对象。因此，操作系统通过诸如读取，写入和执行之类的操作来保护文件，目录，设备和端口等内容。关系数据库管理系统通过诸如SELECT，UPDATE，DELETE和INSERT之类的操作来保护关系，元组，属性和视图。会计应用程序通过借记，贷记，转帐，创建帐户和删除帐户等操作来保护帐户和分类帐。应该有可能将贷记操作分配给某个角色，而不必强迫也将借记操作分配给该角色。

权限可以应用于单个对象，也可以应用于多个对象。例如，权限可以特定于对特定文件的读取访问权限，也可以一般性地与对属于特定部门的所有文件的读取访问权限一样。将各个权限合并为通用权限，以便可以将它们分配为单个单元的方式在很大程度上取决于实现。

图1（b）显示了用户分配（UA）和权限分配（PA）的关系。两者都是多对多关系。一个用户可以是许多角色的成员，而一个角色可以有许多用户。同样，一个角色可以具有许多权限，并且可以将相同的权限分配给许多角色。RBAC的关键在于这两个关系。最终，这是一个行使权限的用户。与直接将用户与权限相关联的角色相比，将角色放置为中介以使用户能够行使权限可以更好地控制访问配置和检查。

每个会话都是一个用户到可能有许多角色的映射，即，用户建立了一个会话，在该会话中，用户激活了他或她所属的角色的某些子集。从会话到图1（b）中的R的双向箭头指示同时激活了多个角色。用户可用的权限是该会话中激活的所有角色的权限的并集。每个会话都与一个用户相关联，如该会话中的单头箭头所示

到图1（b）中的U_0。在整个会话期间，这种关联保持不变。

用户可能同时打开多个会话，例如，每个会话都在工作站屏幕上的不同窗口中。每个会话可能有不同的
活动角色的组合。$RBAC_0$的此功能支持最小原则
特权。属于多个角色的用户可以调用这些角色的任何子集
适合该会话中要完成的任务。因此，作为强大角色的成员的用户通常可以保持禁用该角色，并在需要时显式激活它。我们将对各种约束（包括角色激活的约束）的考虑推迟到$RBAC_2$。因此，在$RBAC_0$中，完全取决于用户的
在给定会话中激活哪些角色的自由裁量权。$RBAC_0$也允许
在会话有效期内动态激活和停用的角色。的
会话的概念等同于访问控制文献中主题的传统概念。主题（或会话）是访问控制的单位，并且用户可以同时具有不同权限的多个主题（或会话）处于活动状态。

以下定义将上述讨论形式化。

定义1 $RBAC_0$模型具有以下组件：

- U，R，P和S（分别为用户，角色，权限和会话），
- PA P x R，角色分配关系的多对多权限，
- UA-U x R，多对多用户到角色分配关系，
- 用户：S！U，将每个会话s_i映射到单个用户user（s_i）（在会话生命周期中恒定）的函数，以及
- 角色：S！在图2^R中，该函数将每个会话s映射到一组角色角色.frj（用户$_i$）；r）2 U Ag（可随时间变化）和会话s_i拥有权限$U_{r2roles(si)}$fp j（p；r）2 PAg。 2

我们希望为每个角色分配至少一个权限，并且为每个用户分配至少一个角色。但是，该模型不需要此。

如前所述，$RBAC_0$将权限视为未解释的符号，因为权限的确切性质取决于实现和系统。我们确实要求权限适用于数据和资源对象，而不适用于RBAC本身的组件。修改集合U，R和P以及关系PA和UA的权限称为管理权限。稍后将在RBAC的管理模型中讨论这些内容。现在，我们假设只有一个安全专家可以更改这些组件。

会话受单个用户的控制。就模型而言，用户可以创建一个会话并选择激活该用户角色的某些子集。会话中活动的角色可以根据用户的判断进行更改。会话终止于

用户的主动性。（如果某些系统长时间处于非活动状态，则它将终止该会话。严格来说，这是一个约束，并且正确地属于$RBAC_2$。）

一些作者[6]除了权限外，还将职责作为角色的属性。义务是用户履行一项或多项任务的义务，这通常对于组织的平稳运行至关重要。我们认为职责是一个高级概念，不属于$RBAC_0$。我们还选择不将职责纳入我们的高级模型中。我们认为将诸如职责之类的概念纳入访问控制模型中需要进一步的研究。一种方法是将其视为类似于权限。其他方法可以基于新的访问控制范例，例如基于任务的授权[4]。

4.2 角色层次

如图1所示，模型$RBAC_1$引入了角色层次结构（RH）。几乎在文献中讨论角色时，几乎不可避免地包括了角色层次结构[7，8，9，10]。它们也通常在提供角色的系统中实现。

角色层次结构是构造角色以体现组织的权限和责任线的自然方法。角色层次结构的示例如图2所示。按照惯例，在这些图的顶部显示功能更强大（或高级）的角色，在底部显示功能更弱（或初级的）角色。在图2（a）中，最主要的角色是卫生保健提供者。医师角色是卫生保健提供者的高级角色，因此继承了卫生保健提供者的所有权限。除了从卫生保健提供者角色继承的权限之外，医师角色还可以具有其他权限。权限的继承是可传递的，因此，例如，在图2（a）中，初级保健医师角色从医师角色和保健提供者角色继承权限。初级保健医师和专科医生均从医师角色继承权限，但是每个角色都会直接分配不同的权限。图2（b）说明了权限的多重继承，其中项目主管角色同时从测试工程师和程序员角色继承。

从数学上讲，这些层次结构是部分订单。偏序是反射性，传递性和反对称关系。继承是反射性的，因为一个角色继承其自己的权限，在这种情况下，可传递性是自然的要求，并且反对称排除了彼此继承的角色，因此是多余的。

$RBAC_1$的正式定义如下。

定义2 $RBAC_1$模型具有以下组件：

- U，R，P，S，PA，UA和用户与$RBAC_0$相同，

- RH R x R是R上的一个偏序，称为角色层次或角色优势关系，也写为2，并且
- 角色：S！从RBAC修改2R以要求角色 0 fr j (9r) 2 i r）[（user（si）；r）2 UA] g（可以随时间变化），会话si具有权限Ur2roles(s)fp j（9r：r）[（p；r）2 PA] g。 2

请注意，允许用户使用其成员角色之前的角色的任何组合来建立会话。同样，会话中的权限是直接分配给会话角色的权限以及分配给这些角色中次要角色的权限。

有时在层次结构中限制继承范围很有用。考虑图2（b）的层次结构，其中项目主管角色比测试工程师和程序员角色都重要。现在，假设测试工程师希望对他们的角色保留一些权限，并阻止他们在层次结构中继承给项目主管。这种情况可能出于正当的原因而存在，例如，访问未完成的工作可能不适合担任高级职位，而RBAC可能有助于使这种访问测试工程师的能力。可以通过定义新的角色测试工程师并将其与测试工程师相关联来解决这种情况，如图2（c）所示。可以将测试工程师的私人权限分配给角色测试工程师。将测试工程师分配给角色测试工程师，并从测试工程师角色继承权限，这些权限也由项目主管角色在层次结构中向上继承。但是，测试工程师的权限不是由项目主管角色继承的。我们称测试工程师等角色为私人角色。图2（c）还显示了私有角色程序员。在某些系统中，私有角色的作用是通过阻止某些权限的向上继承来实现的。在这种情况下，层次结构无法准确描述权限分配。最好引入私有角色，并保持角色之间的层次关系完整。

图3更一般地显示了如何构建角色的私有子层次结构。图3（a）的层次结构具有四个任务角色T 1，T 2，T 3和T 4，所有这些角色都继承了公共项目范围内角色P的权限。层次结构顶部的角色S供项目主管使用。任务T 3和T 4是一个子项目，其中P 3是子项目范围的角色，而S3是子项目的监督角色。图3（c）中的角色T 1是任务T 1成员的私有角色。假设图3（a）的包含角色S3，T 3，T 4和P 3的子项目需要私有子层次结构在其中，项目的私有权限可以被S共享而无需继承。整个子层次结构以图3（c）所示的方式被复制。可以将S继承的权限分配给S3，T 3，T 4和P 3，而将私有权限分配给S3，T 3，T 4和P 3，仅允许它们在子项目中继承。与之前一样，子项目团队的成员被直接分配给S3，T 3，T 4或P 3。图3（c）清楚地表明了

私有角色存在于系统中，并协助访问检查以确定私有权限的性质。

4.3 约束

$RBAC_2$模型引入了约束的概念，如图1（b）所示。铝

尽管我们将模型称为$RBAC_1$和$RBAC_2$，但实际上并没有暗示进展。约束或角色层次结构可以首先引入。这是由图1（a）中$RBAC_1$和$RBAC_2$之间不可比的关系表示。约束是RBAC的重要方面，有时被认为是RBAC的主要动机。一个常见的例子是相互分离的角色，例如采购经理和应付账款经理。在大多数组织中（最小的组织除外），同一个人不得成为成员扮演两个角色，因为这可能导致欺诈。这很好众所周知的历史悠久的原则称为职责分离。

约束是用于制定更高级别组织策略的强大机制。一旦声明某些角色是互斥的，就无需过多考虑将各个用户分配给角色。然后可以将后者的活动下放和下放，而不必担心会损害组织的总体政策目标。只要RBAC的管理完全集中在单个安全性工具中，约束就是有用的便利。但是，通过安全性方面的谨慎护理，在很大程度上可以达到相同的效果。但是，如果RBAC的管理是分散的（将在后面讨论），则约束将成为一种机制，高级安全人员可以通过该机制来限制可以行使管理特权的用户的能力。这使首席安全官可以列出可接受的范围，并将其作为对参与RBAC管理的其他安全官和用户的强制性要求。

关于$RBAC_0$，约束可以应用于UA和PA关系以及各种会话的用户和角色功能。约束是谓词，应用于这些关系和函数的谓词返回“可接受”或“不可接受”的值。约束也可以看作是某种适当形式语言的句子。直观上，可以根据约束的种类和性质更好地查看约束。我们非正式地讨论约束，而不是用正式的符号表示约束。因此，以下定义。

定义3 $RBAC_2$与$RBAC_0$相同，只是要求有一组约束来确定$RBAC_0$的各个组件的值是否可接受。仅允许接受值。 2

实施方面的考虑通常要求可以有效检查和实施的简单约束。幸运的是，在RBAC中，简单的约束可以

很长的路要走。现在，我们讨论一些我们认为可以合理实施的约束。应用于用户分配关系的大多数（如果不是全部）约束具有适用于权限分配关系的对应项。因此，我们并行讨论了对这两个组件的约束。

在RBAC上下文中最常提到的约束是互斥角色。可以将同一用户最多分配给一个互斥集中的一个角色。这支持职责分离。提供此约束几乎不需要动力。权限分配的双重约束在文献中几乎没有提及。实际上，权限分配上的互斥约束可以为职责分离提供额外的保证。这种双重约束要求可以将相同的权限分配给互斥集中的最多一个角色。考虑两个互斥的角色，客户经理和采购经理。就UA而言，相互排斥是指一个人不能同时担任这两个角色。就PA规范而言，相互排斥是指不能将相同的权限分配给两个角色。例如，不应将签发支票的权限分配给两个角色。通常，将这样的权限分配给帐户经理角色。对PA的互斥约束将防止许可被无意或恶意分配给采购经理角色。更直接地，对PA的排除约束是限制强大权限分配的有用方法。例如，角色A或角色B是否获得特定帐户的签名权限并不重要，但是我们可能要求两个角色中只有一个获得此许可。

更一般而言，用户在各种角色组合中的成员资格可以认为是可接受的或不可接受的。因此，用户可以在不同项目中担任程序员角色和测试人员角色的成员是可以接受的，但是在同一项目中同时担任这两个角色是不可接受的。权限分配也类似。

用户分配约束的另一个示例是角色可以具有最大数量的成员。例如，一个部门的主席只有一个人。类似地，也可以限制单个用户可以属于的角色数量。我们称这些基数约束。相应地，可以向其分配权限的角色数量可能具有基数约束，以控制强大权限的分配。应该注意的是，最小基数约束可能难以实现。例如，如果某个角色的占用人数最少，那么如果其中一名失踪者该怎么办？系统如何知道这种情况已发生？

必备角色的概念基于能力和适当性，因此，只有在用户已经是角色B的成员时，才可以将用户分配给角色A。例如，仅可以分配已经是项目角色的那些用户的用户。到该项目中的测试任务角色。在此示例中，先决条件角色比假定的新角色小。在实践中，无与伦比的角色之间的前提条件不太可能发生。权限分配的双重约束

在PA关系的角色末端应用更多。出于一致性考虑，要求仅当角色已经拥有权限q时才可以将权限p分配给该角色可能是有用的。例如，在许多系统中，读取文件的权限需要读取文件所在的目录的权限。分配前者权限而没有后者将是不完整的。

仅当在向用户分配用户标识时保持适当的外部纪律时，用户分配约束才有效。如果为同一个人分配了两个或多个用户标识，则分隔和基数控件将崩溃。用户标识符和人类之间必须存在一一对应的关系。类似的论点适用于权限约束。如果相同的操作受到两个不同权限的批准，则RBAC系统将无法有效地执行基数和分隔约束。

约束也可以应用于会话，以及与会话关联的用户和角色功能。用户可以是两个角色的成员是可以接受的，但该用户不能同时在两个角色中都处于活动状态。对会话的其他限制可能会限制用户可以同时处于活动状态的会话数。相应地，可以限制分配了权限的会话数。

角色层次结构可以视为约束。约束是分配给初级角色的权限也必须分配给所有高级角色。或等效地，约束是分配给高级角色的用户也必须分配给所有初级角色。因此从某种意义上说，$RBAC_1$是冗余的，被包含在内由$RBAC_2$。 但是，我们认为认识角色的存在是适当的层次结构本身。仅通过引入它们就可以减少约束权限分配或用户分配的冗余。最好直接支持层次结构，而不是通过冗余分配间接支持层次结构。

4.4 合并模型

$rbac_3$ 结合$RBAC_1$ 和$RBAC_2$ 同时提供角色层次结构和。将这两个概念结合在一起会产生几个问题。

约束可以应用于角色层次结构本身，如图1（b）中RH的虚线箭头所示。角色层次结构必须是部分顺序。该约束是模型固有的。其他限制可能会限制给定角色可能具有的高级（或初级）角色的数量。也可以将两个或多个角色限制为没有共同的高级（或初级）角色。这些更改约束在更改角色层次结构的权限已分散的情况下很有用，但是首席安全官希望限制进行更改的总体方式。

约束和层次结构之间会产生微妙的相互作用。假设在图2（b）中声明测试工程师和程序员角色是互斥的。项目主管的角色违反了这种相互排斥。在某些情况下

高级角色对这种互斥约束的违反可能是可以接受的，而在其他情况下则是不可接受的。我们认为该模型不应排除一种或另一种可能性。基数约束也会出现类似情况。假设一个用户最多可以分配一个角色。分配给图2（b）中的测试工程师角色是否违反了此约束？换句话说，基数约束是否仅适用于直接成员资格，或者它们还延续到继承的成员资格上？

图2（c）的层次结构说明了在存在私人角色时约束是如何有用的。在这种情况下，可以声明测试工程师^{0}，程序员^{0}和项目主管角色是互斥的。由于这些职位没有共同的高级职位，因此没有冲突。通常，私人角色不会与其他任何角色有共同的上级，因为他们是层次结构中的最大元素。因此，始终可以指定互斥的私人角色而不会引起任何冲突。可以声明私有角色的共享对等方的最大基数约束为零成员。这样，必须将测试工程师分配给测试工程师^{0}角色。测试工程师角色是与项目主管角色共享权限的一种方式。

5 管理模式

到目前为止，我们已经假定RBAC的所有组件都在单个安全策略的直接控制之下。在大型系统中，角色的数量可以成百上千。管理这些角色及其相互关系是一项艰巨的任务，通常是高度集中的任务，并委派给一小组安全管理员。因为RBAC的主要优点是方便了权限管理，所以很自然地提出如何使用RBAC来管理RBAC本身。我们认为使用RBAC来管理RBAC将是RBAC成功的重要因素。在这里，我们只能谈一些主要问题。

我们提到了文献中已经讨论过的一些访问控制管理方法。ISO已制定了许多与安全管理相关的标准和文件。可通过顶层系统管理概述文档[11]来解决这些问题。ISO模型是面向对象的，并且包括基于包含的层次结构（目录包含文件，文件包含记录）。可以将角色整合到ISO方法中。

存取权传播的模型由来已久，其中传播权由特殊控制权控制。其中最先进的是Sandhu的类型化访问矩阵模型[12]。尽管通常很难分析甚至很简单的权利传播规则的后果，但这些模型表明，可以构成简单的原语，以产生非常灵活和表现力强的系统。

Mofet和Sloman [13]是管理RBAC的一个例子。

根据角色域，所有者，经理和安全管理员定义一个相当精细的模型。在他们的工作中，权威不是从一个中心来控制或委派的，而是在彼此之间只有有限信任的独立经理之间进行协商。

我们的RBAC管理模型如图4所示。该图的上半部分基本上与图1（b）相同。图4中的约束适用于所有组件。图4的下半部分是管理角色和管理权限的上半部分的镜像。预期管理角色AR和管理许可AP分别与常规角色R和许可P不相交。该模型显示，权限只能分配给角色，管理权限只能分配给管理角色。这是一个内置的约束。

图4的上半部分在$RBAC_0$，$RBAC_1$，$RBAC_2$和$RBAC_3$。下半部分的复杂程度可能在整个$ARBAC_0$，$ARBAC_1$，$ARBAC_2$和$ARBAC_3$，其中A表示管理。一般而言，我们希望管理模型比RBAC更简单模型本身。因此，可以使用$ARBAC_0$来管理$RBAC_3$，但是使用$ARBAC_3$来管理$RBAC_0$似乎毫无意义。

同样重要的是要认识到约束可以跨越图4的上半部分和下半部分。我们已经声明了一个内置约束，即只能将权限分配给角色，而将管理权限只能分配给管理角色。如果管理角色相对于常规角色是互斥的，那么我们将遇到这样的情况，安全管理员可以管理RBAC，但自己不能使用任何特权。

行政层级的管理怎么样？原则上，可以构建第二级管理层次结构来管理第一级，依此类推。我们认为甚至不需要第二级的管理层次。因此，管理层次结构的管理由单个首席安全官负责。这对于单个组织或组织内的单个管理单位是合理的。这些单元如何相互作用的问题并未在我们的模型中直接解决。

RBAC中的管理权限可以看作是修改用户分配，权限分配和角色层次关系的功能。在管理模型中，必须明确定义授权这些管理操作的权限。这些权限的确切性质是具体实现，但是它们的一般性质几乎相同。

管理模型中的主要问题之一是如何确定赋予行政角色的行政权限的范围。为了说明这一点，请考虑图3（a）中所示的层次结构。图3（b）的管理层次结构显示了一个单一的首席安全角色（CSO），该角色比SO1，SO2和SO3这三个安全角色更高。范围界定问题涉及图3（a）的哪些角色可以

由图3（b）中的哪个角色来管理。让我们说CSO角色可以管理图3（a）的所有角色。假设SO1管理任务T1。通常，我们不希望SO1也自动继承管理初级角色P的功能。因此，SO1的范围可以完全限于T1。同样，SO2的范围可以限制为T2。假设SO3可以管理由S3，T3，T4和P3组成的整个子项目。然后，SO3的范围由顶部的S3和底部的P3限制。

通常，每个管理角色都将映射到它负责管理的角色层次结构的某个子集。管理的其他方面也需要确定范围。例如，SO1只能将用户添加到T1角色，但是将其删除则需要CSO采取行动。更一般而言，我们不仅需要确定管理角色管理的角色的范围，还需要确定角色管理的权限和用户的范围。控制角色层次结构本身中的更改也很重要。例如，由于SO3管理S3和P3之间的子层次结构，因此可以授权SO3向该子项目添加其他任务。

6 结论

我们提出了一系列RBAC模型，这些模型系统地涵盖了从简单到复杂的整个范围。这些模型为该领域的其他研究和开发提供了共同的参考框架。我们还提出了一种管理模型，通过该模型可以使用RBAC进行自我控制。这支持了我们的观点，即RBAC与策略无关，而不是特定安全策略的模型。

要实现RBAC的承诺，还有许多工作要做。该领域的突出研究问题之一是开发一种系统的方法来设计和分析RBAC配置。最近已经有一些关于角色层次结构设计和分析的研究报告[8，9，14]。如前所述，在文献中很少有关于RBAC上下文中约束的讨论。约束的分类和分类法将很有用。应该制定一个正式的说明和执行约束的符号，以及一些难于执行的措施。在更高级别的政策目标方面，能够推理约束并分析RBAC配置的净效果的能力是一个重要的开放研究领域。RBAC的管理方面需要进一步的工作。开发一个系统的方法论以一个无框架的框架处理角色层次结构，约束和RBAC管理的设计和分析是一项具有挑战性的研究目标。这些悬而未决的问题中的许多问题是交织在一起的，将需要采用综合的方法来解决。

致谢作者感谢NIST的David Ferraiolo和Janet Cugini在本文进行中提供了有用的意见。作者还感谢匿名评论者的评论和建议对论文的改进。

参考文献

[1] David F. Ferraiolo, Dennis M. Gilbert和Nickilyn Lynch。检查联邦和商业访问控制政策的需求。在NIST-NCSC国家计算机安全会议上，第107-116页，马里兰州巴尔的摩，9月 20-23 1993.

[2] 通用标准编辑委员会。信息技术安全评估通用标准，1994年12月。0.9版，初稿。

[3] Imtiaz Mohammed和David M. Dilts。基于动态用户角色的安全性设计。计算机与安全，13（8）：661-671， 1994.

[4] Roshan Thomas和Ravi S. Sandhu。基于任务的授权模型的概念基础。在IEEE计算机安全基础研讨会7，第66-79页，新罕布什尔州弗兰科尼亚 1994.

[5] 拉维·桑德（Ravi S.基于格的访问控制模型。IEEE计算机，11月26（11）：9-19 1993.

[6] 德克·琼斯（Dirk Jonscher）。通过积极的机制来扩展访问控制的职责。在B. Thuraisingham和CE Landwehr中，数据库安全性编辑 VI:现状与前景，第91-111页。北荷兰 1993.

[7] David Ferraiolo和Richard Kuhn。基于角色的访问控制。在第15届NISTNCSC全国计算机安全会议上，第554-563页，马里兰州巴尔的摩，10月13日至16日1992.

[8] M.-Y.胡，萨德穆尔建和丁挺。ADAM面向对象设计和分析环境中基于用户角色的安全性。在J. Biskup，M。Morgernstern和C.Landwehr中，《数据库安全性VIII：现状和前景》的编辑。北荷兰， 1995.

[9] Matunda Nyanchama和Sylvia Osborn。基于角色的安全系统中的访问权限管理。在J. Biskup，M。Morgernstern和C.Landwehr中，《数据库安全性VIII：现状和前景》的编辑。北荷兰 1995.

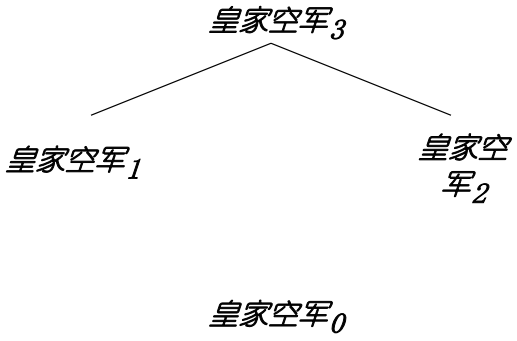
[10] SH von Solms和Isak van der Merwe。使用面向角色的方法来管理计算机安全。计算机与安全，1994，13（8）：673-680。

[11] ISO / IEC10040。信息技术-开放系统互连-系统管理概述。

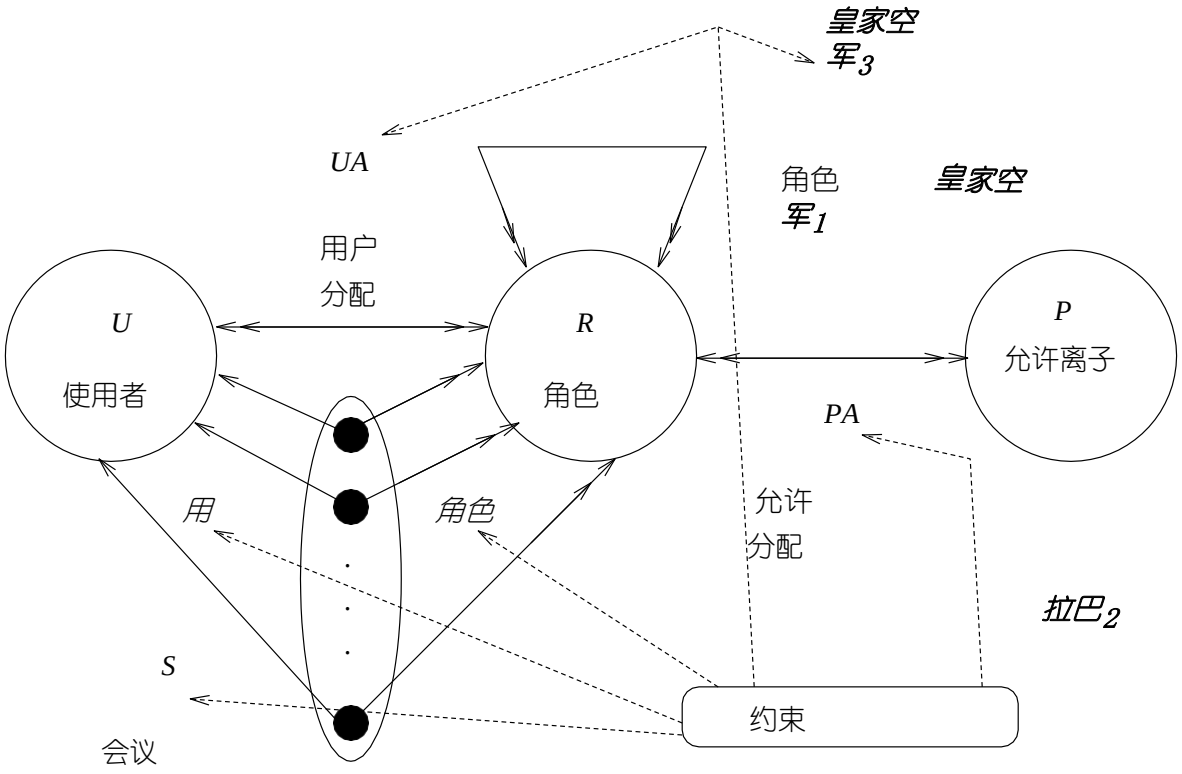
[12] 拉维·桑德（Ravi S.类型化访问矩阵模型。在IEEE计算机学会安全与隐私研究研讨会论文集，第122-136页，加利福尼亚州奥克兰，5月 1992.

[13] 乔纳森·D·莫菲特和莫里斯●斯洛曼。权力下放。在I. Krishnan和W. Zimmer中，《集成网络管理》 II的编辑，第1页， 595-606. Elsevier科学出版社BV（北荷兰）， 1991.

[14] Eduardo B. Fernandez，吴捷和Minjie H. Fernandez。面向对象的数据库授权中的用户组结构。在J. Biskup，M。Morgernstern和C.Landwehr中，《数据库安全性VIII：现状和前景》的编辑。北荷兰， 1995.

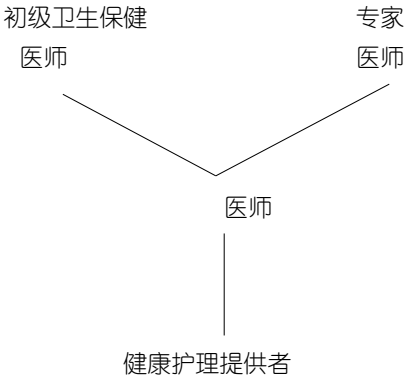


(a) RBAC模型之间的关系

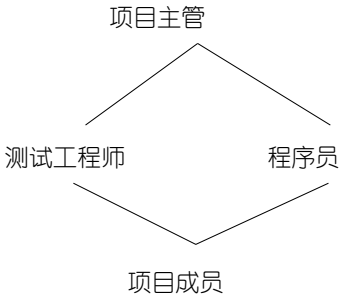


(b) RBAC模型

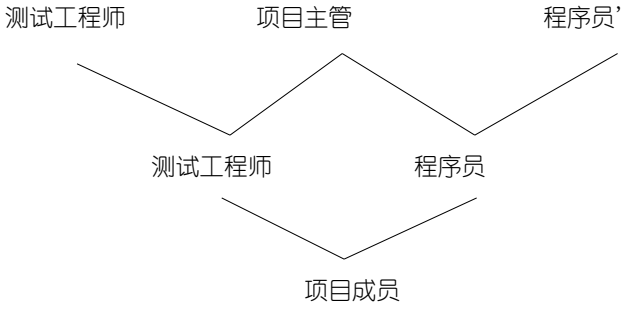
图1：RBAC模型家族



(a)

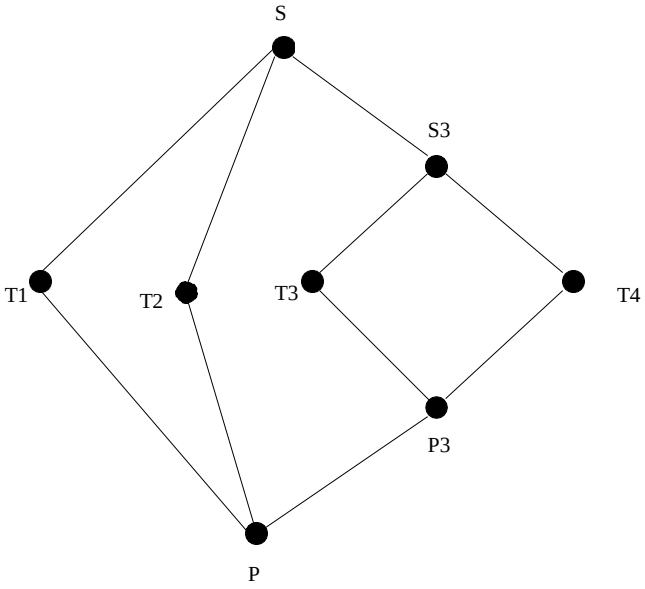


(b)

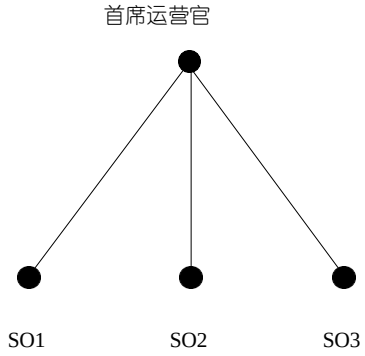


(c)

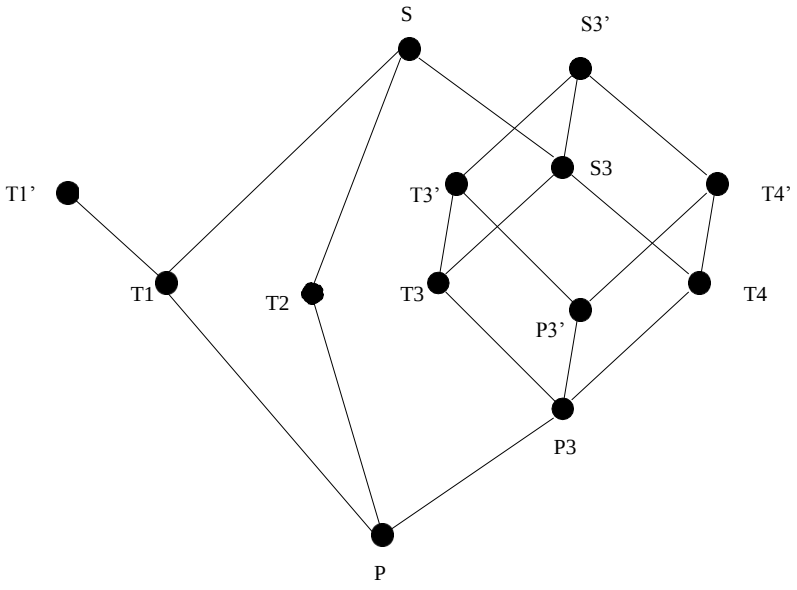
图2：角色层次结构示例



(a) 角色层次结构



(b) 行政角色等级



(c) 私人和有作用的角色

图3：项目的角色层次结构

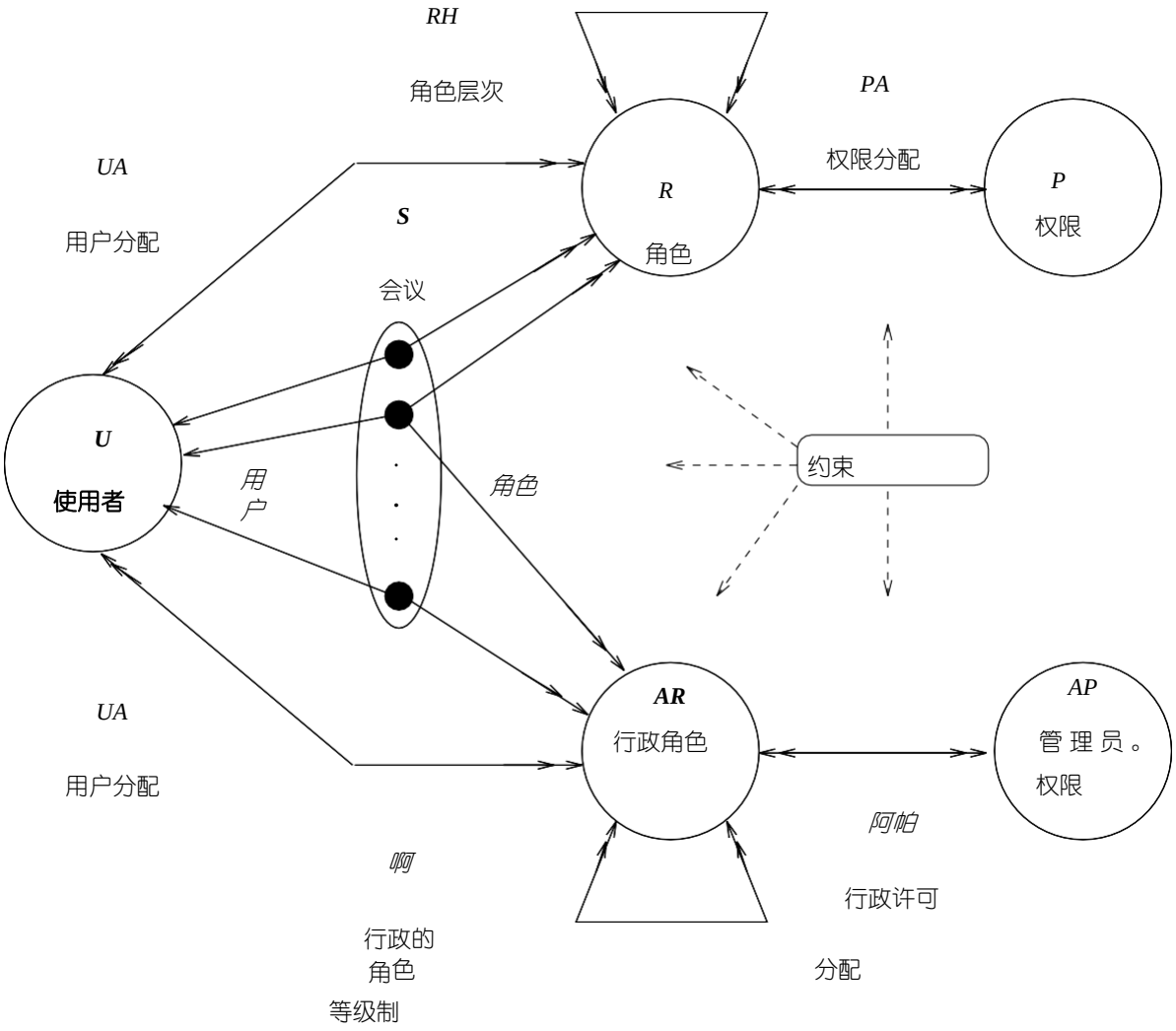

RH
角色层次
PA
UA
用户分配
S
会议
权限分配
R
角色
P
权限
U
使用者
用户
角色
约束
UA
用户分配
AR
行政角色
AP
管理员。
权限
阿帕
啊
行政的
角色
等级制
行政许可
分配

图4：RBAC管理模型