文章编号:1001-2486(2004)03-0057-06

基于任务和角色的分布式工作流安全模型*

付松龄,谭庆平
(国防科技大学计算机学院,湖南 长沙 410073)

摘 要:针对现有基于角色访问控制的缺陷和分布式工作流管理系统的特性,在传统的基于角色的访问控制模型中引入任务集(Tasks)、任务实例集(Task Instances)和任务上下文(Task Context)的概念,将传统的user-role-permission权限赋予结构修改为user-role-task-permission权限赋予结构,建立了基于任务和角色的访问控制模型,给出了其形式化定义。该模型解决了传统的基于角色访问控制中的动态适应性差和最小权限约束假象的问题,用于分布式工作流管理系统,提高了安全性、实用性。

关键词:RBAC;基于任务和角色的访问控制;分布式工作流管理系统;任务;任务实例;任务上下文
中图分类号:TP393 文献标识码:A

Security Task & Role-based Distributed Workflow Model

FU Song-ling, TAN Qing-ping
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: This paper introduces the concept of tasks, task instances and task context into traditional role-based access control model according to the weaknesses of the current role-based access control and the characteristics of distributed workflow system. We propose a task & role-based access control model, whose architecture is not user-role-permission but user-role-task-permission, and its formal definition. This model overcomes the weaknesses of the bad dynamic adaption and the fake constraint of the least privilege. It can enhance the security and practicability of the distributed workflow system.

Key words: RBAC; task & role-based access control; distributed workflow management system; task; task instance; task context

工作流安全白皮书[1]中指出,工作流的基本安全问题包括认证(Authentication)、授权(Authorization)、访问控制(Access Control)、审计(Audit)、数据保密性(Data Privacy)、数据完整性(Data Integrity)、防否认(Non Repudiation)、安全管理(Security Management & Administration)等。分布式工作流管理系统中最重要的安全机制[2]是加密、数字签名、访问控制和多级安全。由于分布式工作流系统的最大特点是将总的工作任务分成更小的任务项,由多人分工在分布于不同地方的工作流执行机(装有工作流引擎的机器)上协作完成,这就需要很好的访问控制模型来对这些协作人员的访问进行管理控制,既要保证不让用户执行未授权的任务,又要保证授权用户顺利地执行已经授权了的任务。如果系统对这些协作人员的安全防护不够,不可避免地就会出现一些人员利用工作之便进行非法操作的可能。所以,在上述安全问题中访问控制最重要。

访问控制的目的是为了限制主体对客体的访问权限,从而使系统在合法范围内使用。传统的访问控制包括自主访问控制(DAC)、强制访问控制(MAC)、基于任务的访问控制(TBAC)和基于角色的访问控制(RBAC)。其中,DAC太弱,MAC太强,TBAC应用面不广,而以RBAC最为灵活、有效。

本文在传统的基于角色的访问控制的基础上引入任务集(Tasks)、任务实例集(Task Instances)和任务上下文(Task Context)的概念,并将RBAC的user-role-permission权限赋予结构修改为user-role-task-permission权限赋予结构,提高了工作流系统中的安全性。本文首先介绍传统的基于角色的工作流安全模型,然后在分析传统模型缺陷的基础上提出了改进的基于任务和角色的工作流安全模型。

* 收稿日期:2003-12-27
基金项目:国家863计划资助项目(2003AA001023)
作者简介:付松龄(1978—),男,硕士生。

最后给出结论。

1 基于角色的访问控制模型

1.1 RBAC96模型

在访问控制领域中，Ravi Sandhu在1996年提出的基于角色的访问控制模型[3]（RBAC96模型）是被频繁引用的一个模型，美国国家标准与技术研究所(NIST)已经将其作为制作规范的基础，它是基于角色访问控制模型的权威参考文档。RBAC96模型主要定义了以下几个部件：用户集(Users)、角色集(Roles)、会话集(Sessions)、角色层次(Role Hierarchy, RH)、用户赋予关系(UA)、权限赋予关系(PA)和约束(Constraints)。它定义了$RBAC_0$、$RBAC_1$、$RBAC_2$、$RBAC_3$四种模型，其中，$RBAC_0$是基本模型，$RBAC_1$在此基础上引入了角色层次的概念，$RBAC_2$在$RBAC_0$的基础上引入了约束的概念，而$RBAC_3$则是前几种模型结合在一起的结果。详细模型可参看文献[3]。

1.2 基于角色的工作流安全模型

Savith Kandala 和 Ravi Sandhu 在2002年提出的工作流安全模型[4]沿用RBAC96中的符号，并引入工作流中的任务、任务实例的概念，定义了一个任务与任务实例之间的函数映射关系。该模型利用工作流系统的特点(任务与任务实例之间的关系)对工作流的访问控制重新建模。其简化模型(详细模型可参看文献[4])可表述如下：

- TT——任务集；TI——任务实例集；$OP = \{execute, commit, abort\}$——操作集；
- 状态集 $S = \{Initial, Executing, Committed, Aborted\}$；
- 可能的操作 $PossibleOperations = \{(Initial, execute), (Executing, commit), (Executing, abort)\}$；
- 状态转换集 $T = \{(Initial, execute, Executing), (Executing, commit, Committed), (Executing, abort, Aborted)\}$；
- 当前状态集 $CurrentState(ti)$表示任务实例 ti 的当前状态；
- $\mathcal{T}: TT \rightarrow 2^{TT}$使得对任意 $a, b \in TT$，如果 $a \neq b$，则 $\mathcal{T}(a) \cap \mathcal{T}(b) = \phi$；
- $EP = OP \times TT$ 表示明权限关系；$IP = OP \times TI$ 表示暗权限关系；$P = EP \cup IP$ 表示权限集；
- 角色的明权限集 $EPA \subseteq R \times EP$；
- 角色的暗权限集 $IPA = \{(ri, op, ti) | [\exists (ri, op, t) \in EPA] \wedge [ti \in \mathcal{T}(t)] \wedge [(CurrentState(ti), op) \in PossibleOperations] \wedge [(StartCondition(ti) = TRUE)]\}$；
- 角色的权限集 $PA = EPA \cup IPA$。

上述模型中有一个建模错误，EP 与 $OP \times TT$ 之间以及 IP 与 $OP \times TI$ 之间都应该是“$\subseteq$”关系，而不应该是“=”关系。为了尊重原文，没有在上述模型中做出更正。

2 基于任务和角色的访问控制模型

2.1 传统RBAC模型的缺陷

RBAC模型的特点就是权限直接与角色相关，其简化模型如图1所示。

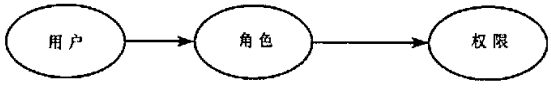

图1 简化的RBAC模型

Fig.1 Simple RBAC model

这种访问控制结构使得将RBAC模型应用到工作流系统中时产生了一些缺陷。

缺陷1：数据冗余且动态适应性差。在上述的基于角色的工作流安全模型中，虽然作者在传统RBAC模型的基础上根据工作流系统中以任务为中心的特点引入了任务的概念，但是作者却把操作与

任务捆绑在一起作为权限集来处理。这种方式要求对每个任务的每个操作都要生成一个类似于二元组(TT,OP)形式的权限,这就有了数据冗余的问题。这种模型的缺陷类似于基于行的自主访问控制中的"前缀表"机制的缺陷,除了难于管理这些二元组之外,当要添加、撤销、改变某个任务的访问权时,还涉及到许多权限(二元组(TT,OP))的更新,这种方式效率低且动态适应性较差。而在分布式工作流管理系统中,动态权限的合法生成和及时撤销却恰恰是最重要的。

缺陷2:最小权限约束假象。由于有最小权限的约束,表面上看传统的RBAC模型好像不存在问题,但实际上这个约束在强调任务特性的工作流系统中却只是个假象,有两个原因可以造成这个假象。第一,在RBAC96模型中明确规定每个用户可以同时拥有多个session,而且只要满足UA关系,每个session可同时激活多个不互斥的角色,这就使得用户具有自己全部session中激活了的所有角色的权限;第二,传统的RBAC模型中,权限是直接与角色相关的,这就决定了权限的粒度最多只能细化到角色一级。最小权限约束只是角色的最小权限约束,而现实世界中一个角色往往可以执行多项任务,也即,不管用户是否执行任务,只要用户激活某一角色就拥有该角色的全部权限。实际上,用户在正常工作时的一个时间点上通常都只是用一个角色执行一项任务,但是以上两点都使得用户在实际操作时拥有多于单独执行一项任务的权限,所以定义的最小权限约束只是个假象,并没有达到真正的最小权限约束。

这种假象的例子很多,比如,用户做任务 A 的时候,本来只能访问数据库 X,但是他实际上却拥有其session中的其它任一激活了的角色的任一权限,比如,访问数据库 Y,而实际的要求却很可能不允许这两个数据库直接交互。对工作流系统来说,虽然在用户执行任务之前多了任务实例化这一限制(任务在实例化之前只是个模板,不能被执行),但是只要采用传统的RBAC模型,这种假象就无法避免,在很多情况下都会产生这种假象。比如,如果有某一任务不需要实例化,或者有某两个实例化过程完全一样的任务,再或者前一任务的结果就是后一任务的实例化,那么这样的任务所对应的权限即使在该任务不被执行时也同样会被用户所具有,因此,即使系统没有分配这种任务的某任务实例给用户,用户也可以做。

2.2 T&RBAC模型

究其根源,产生上述RBAC模型缺陷的原因就是RBAC模型的三级访问控制结构。这种结构决定了RBAC的这些先天性的缺陷,所以,在现有的以RBAC模型为基础的系统中,都存在着上述几类缺陷,在工作流系统中,由于任务的概念非常突出,所以,这种缺陷也就更为突出。为了解决上述问题,必须改变这种结构,在此将任务集和任务实例集的概念明确引入RBAC模型中,形成基于任务和角色的访问控制模型(T&RBAC)。也有其它一些文献将任务这个概念引入RBAC模型中,如文献[4],但是它们都只是简单地将任务附加在权限之上作为一种复合权限使用,没有改变传统RBAC模型的访问控制结构,不能解决传统RBAC模型的那些缺陷,本文将任务独立出来作为一个单独的概念,将任务集置于角色集和权限集之间,将传统RBAC模型的三级结构修改为四级结构,如图2所示。

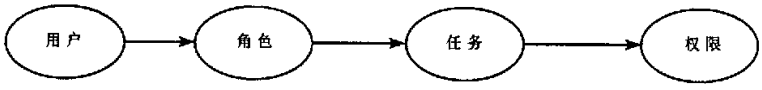

图2 简化的T&RBAC模型

Fig.2 Simple T&RBAC model

整个工作流系统中的用户都赋予特定的角色,再规定每个角色可以执行哪些任务以及每项任务的最小访问执行权限。在这样的访问控制结构中,权限不再直接与角色相关,而必须通过任务才能与角色关联起来,而这时的权限也不再表示操作与任务,而只是单纯地表示某操作。权限可以作为任务的属性来实现,不必建立类似于传统RBAC模型中的二元组权限,大大降低了数据冗余度。另外,不管哪个用户用赋予他的什么角色登录,都必须在执行授权任务实例期间才拥有该任务实例的权限,这样就很自然地实现了权限的动态分配和撤销,大大提高了工作流安全系统的动态适应性,这就解决了传统RBAC模型中的第一个缺陷。

在T&RBAC模型中,由于权限直接与任务相关,所以,这时的最小权限约束就很自然地细化到了任务一级,这种粒度恰恰是实际工作流系统所需要的。T&RBAC模型的这两个"自然"的特性就使得它不存在最小权限约束的假象。到此为止,传统RBAC模型中的上述几类缺陷都得到了解决。

在现实生活中,即使是同一个操作,不同的操作人员也有不同的对象访问范围[5]。比如,在一个软件公司中,对同一个"修改软件需求文档"这一操作而言,项目组中的成员只能修改本项目组所拥有的需求文档,而公司项目总负责人却可以修改所有项目组的需求文档;再如,在银行中,正常的业务操作不可能在夜间(下班时间)进行,但是,这些操作在白天是允许的也是必须的。

传统的RBAC模型没有解决这类问题。由于本文所提出的模型中多了一个任务集,其安全模型的关键也在于对任务的执行权限的控制上,所以在此引入了task context这一概念,它包含了某一特定任务执行时的所有安全信息,这些安全信息构成了任务执行时的系统环境。没有这个环境,该任务不能被执行。有了task context这个概念,就可以保证某一权限只能在某特定的时间段、特定的地点、在输入条件满足的情况下由具有特定角色的用户执行。

2.3 概念定义及系统实现

几个基本概念定义如下:

执行工作流机(WfM) 安装有工作流引擎的服务器,用户直接与之相连以完成工作任务。

任务上下文(Task Context) 对任务是否能被执行所加的约束条件,任务上下文包括了执行任务所需要的用户ID以及登录的角色ID,哪个执行机,什么时间,输入条件(数据)等数据。

角色(roles) 角色表示任务集的一个子集。

任务权限赋予关系(Task Permissions Assignment,TPA) 为各任务模板定义其所需要的最小权限集。

实例权限赋予关系(Instances Permissions Assignment,IPA) 根据任务权限赋予关系以及任务与任务实例之间的映射关系来确定具体的任务实例的权限集。

任务角色赋予关系(Task Roles Assignment,TRA) 为各角色定义可以执行哪些任务。

实例角色赋予关系(Instances Roles Assignment,IRA) 根据任务角色赋予关系和任务与任务实例之间的映射关系得到角色可以执行的任务实例集。

实际操作中,角色总是通过IRA、IPA才能真正获得权限。由以上讨论对该T&RBAC模型进行建模,结果可以表述如下:

U(用户集)、S(会话集)、UA(用户赋予关系)、RH(角色层次关系)、TT(任务模板集合)、TI(任务实例集合)、$\mathscr{T}: TT \to 2^{TI}$(任务模板与任务实例之间的映射关系)的定义与1.2节中定义一致。

另外再增加(或重定义)如下组件:

- $WfM = \{WfM_1, WfM_2, \cdots, WfM_n\}$——执行工作流机的集合;
- R——角色集(任务集的子集);
- $TPA \subseteq TT \times P$——任务权限赋予关系,将每一项任务对应一个最小权限集;
- $IPA \subseteq TI \times P$——实例权限赋予关系,由 $\mathscr{T}$ 和 TPA 决定;
- $TRA \subseteq R \times TT$——任务角色赋予关系;
- $IRA \subseteq R \times TI$——实例角色赋予关系,由 $\mathscr{T}$ 和 TRA 决定;
- $taskContext = (uID, crID, UA, RH, TPA, TRA, WFMi, time, input)$——任务上下文。

定义几个方法:

- 可以分配给角色 r 的任务:$tasks(r) = \{tt \mid (\exists r' \leqslant r)[(r', tt) \in TTA]\}$;
- 在任务上下文环境中,用户可以执行的任务实例:

$$taskInstances(taskContext) = \{ti \mid (\exists r' \leqslant r)(\exists tt \in tasks(r'))$$
$$[ti \in \mathscr{T}(tt)] \wedge [canExecute(taskContext, tt) = true]\}$$

- 在任务上下文环境中,用户具有的权限:

$$permissions(taskContext, ti) = \{p \mid \exists tt[(ti \in \mathscr{T}(tt)) \wedge (ti \in taskInstances(taskContext)) \wedge ((tt, p) \in EPA)]\}$$

另外,为了保证一个任务实例只被一个用户在某 *taskContext* 环境下只执行一次,还约定:任务状态之间的转换具有不可逆转性,执行失败的任务实例将被删除,必须重新判断用户是否可执行,并实例化之后才能重新执行。

到此为止,T&RBAC 模型的建模工作就基本完成了,现在给出基于任务和角色的访问控制(T&RBAC)模型的示意图,如图 3 所示。

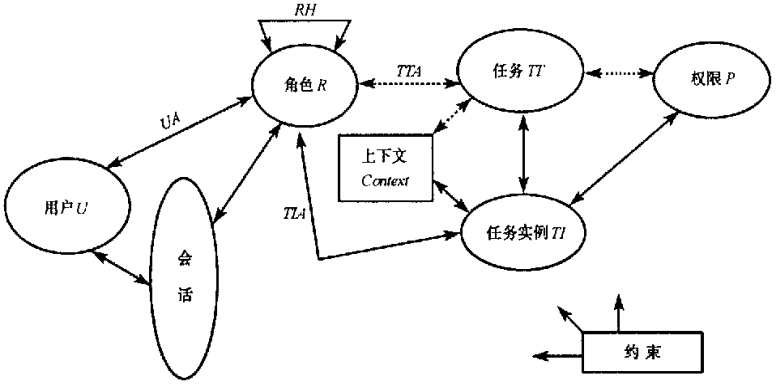

图 3 T&RBAC 模型

Fig.3 T&RBAC model

T&RBAC 模型图在传统的 RBAC 模型图[3] 的基础上增加了任务、任务实例和任务上下文这几个组件。用户和任务都被分别赋予角色,而权限则被赋予任务。用户在会话过程中,以某一角色执行一项任务,从而获得该任务所具有的权限。传统 RBAC 模型中所定义的一些约束(基数约束、静态职责分离等)同样作用于整个模型。任务上下文约束作用于任务,在运行过程中则作用于任务实例。

T&RBAC 模型图中的虚线表示在实际的工作流系统运行过程中,角色只能执行任务实例,也只有任务实例才真正拥有任务所对应的权限,这就保证了角色在任务执行期间的权限赋予和任务非执行期间的权限撤销。

下面给出在我们开发的工作流管理系统中任务执行授权的顺序图(如图 4 所示)。

3 结 论

在工作流系统中,存在两种权限,执行任务实例的权限和在执行任务实例过程中所必须的操作权限。本文在已有 RBAC 和安全工作流模型的基础上提出一种增强的分布式工作流安全模型,该模型在已有 RBAC 模型的基础上明确提出将任务集(task)加入 RBAC 模型形成基于任务和角色的访问控制模型(T&RBAC),且引入任务上下文(task context)的概念,解决了将传统 RBAC 模型应用到工作流管理系统中时所产生的一些缺陷。该模型将任务集引入传统 RBAC 模型之后,将工作流系统中的两种权限都与用户很自然地隔离开来,增强了系统的安全性,是企业中实际安全控制流程的真实再现。

基于任务和角色的访问控制模型是在工作流系统已有组件(任务、组织模型、任务与任务执行者之间的映射关系)基础上构造的访问控制模型。安全强度介于自主访问控制和强制访问控制之间。与没有采用访问控制模型的工作流系统相比,采用新模型的工作流系统增加了访问控制检查。对于有访问控制需求的工作流系统而言,这些检查是必不可少的。检查本身所带来的额外开销并不大,因此新模型对工作流系统的效率不会造成过大影响。

T&RBAC 模型不仅具有传统 RBAC 模型的所有优点,而且克服了它所固有的诸多缺陷,更能与工作流管理系统无缝结合,在将该模型应用于我们自主研发的基于 J2EE 的分布式工作流管理平台项目的过程中,我们发现该模型与基于传统 RBAC 模型的工作流系统相比,安全性得到了极大的增强。

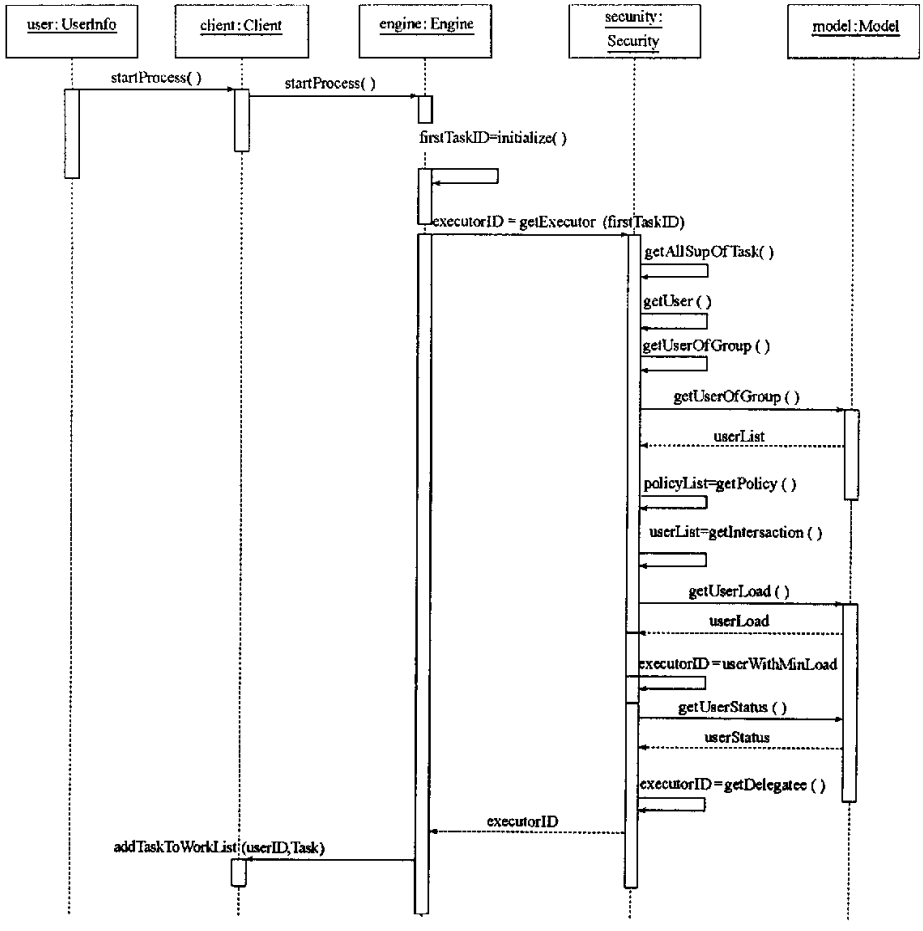

图4 任务指派顺序图

Fig.4 Task assignment sequence diagram

参考文献:

[1] WfMC. TC00-1019:Workflow Management Coalition Workflow Security Considerations White Paper[S]. http://www.wfmc.org, 1998.

[2] Miller J A, et al. Security in Web-based Workflow Management Systems[C].ACM Trans. on Information and System Security, 2002.

[3] Sandhu R S, Coyne E J, Feinstein H L, Youman C E. Role Based Access Control Models[J]. In IEEE Computer, 1996,29(2):38-47.

[4] Kandala S,Sandhu R.Secure Role-based Workflow Models[J].In Proceedings of the 15th IFIP WG 11.3 Working Conference on Database Security, Kluwer 2002:45-58.

[5] Kumar A, Kamik N, Chafle G. Context Sensitivity in Role-based Access Control[J]. ACM SIGOPS Operating Systems Review, July 2002.

[6] 邓集波，洪帆. 基于任务的访问控制模型[J]. 软件学报,2003,14(1):76-82.

[7] 石文昌，孙玉芳. 多级安全政策的历史敏感性[J]. 软件学报,2003,14(1):91-96.