主动授权规则实施支持空间特性的 RBAC*

汤 铸, 鞠时光, 陈伟鹤
(江苏大学 计算机科学与通信工程学院, 江苏 镇江 212013)

摘 要: 为满足安全策略或者角色定义的变化,系统或模型应该提供一种灵活的机制实施支持空间特性的 RBAC。引入了 OITE(on-if-then-else)主动授权规则实施支持空间特性的 RBAC,定义了支持空间特性的 RBAC 中各基本元素与 OITE 之间的映射关系。使用 OITE 作为实施机制,可以在不同粒度上实施带有空间特性角色约束,并且可以将支持空间特性的 RBAC 应用在多个领域中。最后简要讨论了授权规则如何从支持空间特性的 RBAC 安全策略中自动产生。

关键词: 安全策略; 角色; 主动授权规则; 带有空间特性角色约束
中图分类号: TP309 **文献标志码**: A **文章编号**: 1001-3695(2010)04-1496-04
doi:10.3969/j.issn.1001-3695.2010.04.081

Active authorization rules for enforcing RBAC with spatial characteristics

TANG Zhu, JU Shi-guang, CHEN Wei-he
(*School of Computer Science & Telecommunication Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

Abstract: To meet security policy or role structure changes, systems or models should provide a flexible mechanism for enforcing role-based access control (RBAC) with spatial characteristics in a seamless way. This paper used on-when-then-else authorization rules for enforcing RBAC with spatial characteristics. And showed the mapping between the basic elements in RBAC with spatial characteristics and the OWTE rule specification. Established OWTE rules as an enforcement mechanism, which could realize role-based constraints with spatial characteristics at different granularities and make RBAC with spatial characteristics usable in diverse domains. Finally discussed briefly how these authorization rules could be automatically generated from security policies using RBAC with spatial characteristics.

Key words: security policy; role; active authorization rules; role-based constraints with spatial characteristics

0 引言

基于角色访问控制(RBAC)[1]基本思想是用户通过角色来获得所需的操作权限。每一个角色直观上来说可以看成是一个职务,代表着与该职务相关的一系列责任、义务及由此确定的相应权限。后来,RBAC 被扩展以支持各种约束,如支持上下文感知的约束[2,3]、支持时间约束[4]等。随着无线定位技术的发展,支持空间特性的 RBAC 已成为当前的研究热点。

Frode 等人[5]为无线通信网开发了 SRBAC 模型。该模型比较简单,位置的空间粒度是固定的,且只是几何值而没有任何语义。当用户在给定的位置上时,角色被自动激活。另外,他们简单提出了带有按特性的职责分离约束。

Bertino 等人在文献[6]中提出了 GEO-RBAC 模型,扩展了传统的 RBAC 模型以用于处理空间的数据信息的访问控制。用空间实体来表示客体、用户的地理位置,以及受地理位置约束的角色;用户角色是否被激活取决于用户所在的位置。同时,为了使模型具有灵活性和可重用性,引入了角色模式(role schema)的概念。将空间环境下的职责分离约束进行扩充以解决不同粒度(空间角色模板/空间角色实例层)、不同维度(空间/非空间)和不同的约束验证时间(静态/动态)。

当前模型和系统都不提供实施支持空间特性 RBAC 的通用方法。例如,有的仅支持空间职责分离约束,而不支持空间角色激活基数约束。同样,这些模型也不能满足访问控制策略和角色定义的变化。例如,一家企业需要改变职工的工作区域,那么角色的作用域就必须修改。在大型企业中,为了支持众多的角色需要数目庞大的约束,手动修改并且维护这些约束不但易出错而且开销大,对管理员是一个沉重的负担。因此,系统不但应能够自动重写并维护各种约束,而且应提供处理约束的统一的、透明的方法。

OITE 是一种增强型 event-condition-action(ECA)[7]主动规则,它给系统提供了基于事件的主动能力。使用 OITE 实施访问控制策略不但能基于系统状态的变化实时采取相应的动作,而且能够满足访问控制策略和角色定义的变化。例如,当在某空间位置上用户激活角色时,所有相关的约束都被满足并保持 TRUE,直到该角色被去除激活。在角色变为无效前,任何一个约束变为 FALSE 时,该角色立即被去除激活。为使得系统更安全,需要自动检测并预防恶意动作,而不是人为干预。例如,在某空间位置上,当用户激活某角色超过一定次数,那么角色激活基数约束被触发并且一些关键的授权被禁用,同时通知系统管理员。本文引入了一些操作。表 1 给出了各种操作的含义。

收稿日期: 2009-06-30; **修回日期**: 2009-08-21 **基金项目**: 江苏大学高级人才启动基金资助项目(07JDG031)

作者简介:汤铸(1982-),男,江苏扬州人,博士研究生,主要研究方向为空间数据库安全技术(atng0651@126.com);鞠时光(1955-),男,教授,博导,博士,主要研究方向为数据库理论与应用、信息安全、计算机图形学;陈伟鹤(1974-),男,副教授,博士,主要研究方向为信息安全、数据库理论.

表1　不同状态的操作

操作类型	含义
assign_u(u,r,LOC)/deassign_u(u,r,LOC)	分配/取消用户*u*在空间位置*LOC*的角色*r*
enable_r(r,LOC)/disable_r(r,LOC)	使角色在空间位置*LOC*有效/无效
activate_r(u,r,LOC)/deactivate_r(u,r,LOC)	用户*u*在空间位置*LOC*激活/不激活角色*r*
θ(LS,S(activate_r(r,LS))	返回角色*r*在区域*LS*中被激活的次数
session_role(u,s,r,LOC))	返回用户在位置*LOC*上会话*s*中的会话角色
SessionRoles(s)	返回会话*s*中的会话角色
EnableSessionRoles(s)	返回会话*s*中的有效角色
check_SDSoD_Set(user,r,LOC)	检查在位置*LOC*上角色*r*是否与用户已有的角色共同存在空间互斥角色集中
add_session_r(u,r,loc_s)	将*r*添加到该用户会话的激活角色集中

1　基于事件的主动授权规则

OITE 是一种基于事件的主动授权规则,是 ECA[7] 规则增加了备选动作和增强的操作符语义,以支持空间数据的授权管理。OITE 由五部分组成:a)约束名 R_{name};b)触发约束的事件 E_i;c)检查条件$\langle C_1, C_2, \cdots, C_n \rangle$;d)当条件满足时触发的动作集$\langle A_1, A_2, \cdots, A_n \rangle$;e)当条件不满足时触发的备选动作集$\langle AA_1, AA_2, \cdots, AA_n \rangle$。发生一个事件可能触发多个规则也可能触发嵌套规则等,OITE 规则表示如下

rule [R_{name}
　on　event$\langle E_i \rangle$
　if　$\langle C_1, C_2, \cdots, C_n \rangle$
　then　$\langle A_1, A_2, \cdots, A_n \rangle$
　else　$\langle AA_1, AA_2, \cdots, AA_n \rangle$]

1)简单事件　在系统中所有预定义的操作都被认为是简单事件。例如,在关系数据库管理系统中的插入、删除等数据操作,操作系统中的文件操作。这些事件可以被用来实施支持空间特性 RBAC 中的各种操作,如当一个用户从某一个空间位置移动到另一个位置上时,空间事件触发某一些规则从而激活/撤销空间角色。

2)条件　条件与事件间是多对多的映射关系,当发生事件时,条件就被赋值。例如,当一个用户要在某公司外部打开一个受保护的公司文件时,系统检查是否安全并作出相应的授权。

3)动作和备选动作　一旦事件被检测到,相关所有条件被赋值 true,预定义的系统动作$\langle A_1, A_2, \cdots, A_n \rangle$就被执行。另一方面,当前的事件处理模型都不能处理条件赋值为 false 的情况,而在 OITE 中,当条件被赋值 false 时,备选动作$\langle AA_1, AA_2, \cdots, AA_n \rangle$被触发。

4)复杂事件　在许多情况下除了简单事件外经常需要复杂事件。该事件是由多个简单事件或多个复杂事件使用事件操作符组合而成[7],如 AND、OR、NOT、SEQUENCE、PLUS 等事件操作符。已经增强了这些事件操作符以支持访问控制。

5)SEQUENCE(E_1, E_2)　使用该操作符,E_1 必须发生在 E_2 前。此时,SEQUENCE 事件就被检测到。例如,在某位置 *LOC* 上,调用一个空间角色时,尽管该角色已被激活,但是该角色在处于有效状态前是不起作用的,只有当用户位于角色作用的空间区域范围内时角色才有效。当该角色满足有效状态这个先决条件,用户才能激活该角色;而当角色处于无效状态时,则不能被激活。那么就可以使用该操作符。

6)OR(E_1, E_2)　当 E_1 和 E_2 中任意一个发生时,OR 事件就被检测到,并且相应的约束被触发。

7)PLUS(E_1,Δ)　该 PLUS 事件既可以是时间事件,也可以是空间事件。当事件 E_1 发生 Δ 时间后,该事件就被检测到。例如,限制在一个特定空间区域内,一个角色被激活 Δ 时间后被自动撤销,否则会引发安全隐患。同样,也可以用来支持空间角色的基数约束。例如,在某空间位置上控制激活角色事件的发生次数。

8)APERIODIC(E_1, E_2, E_3)　当事件 E_2 发生在 E_1 和 E_3 之间时,该事件就被检测到。例如,在空间角色的空间区域约束中,用最小外包矩形来表示角色作用的空间区域,用最小外包矩形的左上角点 C_i 和右下角 C_j 来表示最小外包矩形。当用户的位置处于 C_i 与 C_j 之间激活角色时,该角色就能处于激活状态。

9)PERIODIC(E_1, T, E_2)　在两个事件之间间隔规定的时间 T 后,该事件就被检测到。该操作符可以被用来周期性的检查系统并发出报告。

所有上述的操作符都可以用来支持带有空间特性 RBAC 模型不同的操作,提供空间数据的授权管理。

2　带有空间特性角色约束的实施

本章首先将支持空间特性的 RBAC 的基本元素映射到 OITE 主动规则,然后使用 OITE 规则实现支持空间特性 RBAC 访问控制策略。

2.1　实体关系建模

支持空间特性的 RBAC 包含四个基本元素集,即用户集 *U*、空间角色集 *R*、权限集 *P*、位置集 *LOC*,所有的这些元素都被认为是实体。在企业中,所有的用户,如个人、代理人;所有的空间角色,如经理、出纳;所有的空间位置,如办公室、会议室被建模为实体 *U*、*R*、*LOC* 的实例。

在某空间位置上实体 *U* 和 *R* 之间存在多对多的关系 *M*:*N*,同样,根据角色—权限的分配、角色层次、位置层次等,其他实体之间相互关联。因此,实体之间的关联描述了构成实体关系模型[8]的关系。在本文中仅考虑在空间位置上实体 *U* 与 *R* 之间的关系。施加该关系上的空间职责分离约束、空间区域约束、空间角色激活基数约束使得仅当实体的实例满足这些约束时,才被允许分配该关系。例如,当实体 *U* 的实例要在位置 *LOC* 上激活实体 *R* 的实例时,施加在用户—角色激活关系上的约束就被检查是否满足。

当事件发生时,相关的主动授权规则就被触发。在传统的 RBAC 中仅实体 *U* 的实例能触发简单事件,如用户需要访问、分配、激活等。但是在支持空间特性的 RBAC 中,空间事件,如传感器的位置,也能触发简单事件。同样,复杂事件总是被组成它的多个简单或复杂事件触发。

支持空间特性的 RBAC 中的所有的空间约束都可以通过设置适当的规则条件以及使用复杂事件操作符来实施。例如,当一个用户试图在某空间位置上通过触发一个事件激活一个空间角色,那么就要检查条件:是否用户在该空间位置上有权限激活该角色,该角色被激活前是否已经处于有效状态等条件。因此,实体 *U* 的实例以及空间事件等作为事件;使用复杂事件和条件设置空间约束,例如分配、激活、允许/拒绝访问请

求等操作作为动作和备选动作。

2.2　空间约束的实施

假设用户列表 user$L$、角色列表 role$L$、会话列表 session$L$ 中包含用户、角色以及会话的信息,并假设在规则中使用的各种函数都可以获得。将带有空间特性的角色约束分为三类,即空间对象区域约束、空间对象职责分离约束和空间对象角色激活基数约束。由于篇幅限制,本文使用 OITE 规则实施部分具有代表性的约束。

2.2.1　空间对象区域约束的实施

对两个空间位置 $C_i(X_i,Y_i)\in LOC$, $C_j(X_j,Y_j)\in LOC$, 如果 $X_i<X_j$ 或 $X_i=X_j\wedge Y_i<Y_j$, 则 C_i 在 C_j 的左边,记为 $C_i<C_j$。对于任意空间区域 CS 找出最小外包矩形, 然后用该最小外包矩形的左上角点 C_i 和右下角 C_j 来表示,即 $CS=\{\langle C_i,C_j\rangle \mid C_i, C_j\in LOC, i<j\}$ 表示由 $\langle C_i,C_j\rangle$ 确定的空间区域。

定义 1　空间对象区域约束。记为 $\zeta(CS,S)$,表示在指定区域范围 CS 内,能否进行一次会话 S,即能否给用户分配角色,或给角色授予权限,或改变角色状态。

在支持空间特性的 RBAC 模型中,引入了角色状态的概念。角色处于以下四种状态之一:有效(enabled)、无效(disabled)、激活(activated)、去除激活(deactivated)。角色的状态之间可以相互转换。无效状态表示该角色不可以被任何用户在会话中使用。处于无效状态的角色可以在特定的情况下转为有效。有效状态表明该角色可被那些扮演该角色的用户,并且该用户的位置在本角色作用域内的用户激活,从而用户行使相应权限。若有用户激活了某个角色,那么该角色就处于激活状态。下面,用 OITE 规则实施控制用户激活角色的空间区域约束。例如,图 1 表示的空间对象区域约束用来控制用户激活角色。在 $\langle C_2(X_2,Y_2),C_4(X_4,Y_4)\rangle$ 区域范围内角色 r_1 是有效状态,在 $\langle C_1(X_1,Y_1),C_3(X_3,Y_3)\rangle$ 区域范围内用户建立一个会话,在该会话内用户 u 激活角色 r_1,但由于角色 r_1 仅在 $\langle C_2,C_4\rangle$ 区域范围内是有效状态,因此用户仅能在 $\langle C_2,C_3\rangle$ 区域范围内激活角色 r_1。

改变角色状态的约束集,由以下两种约束构成:

a)角色状态有效约束(RoleState-Area1):$S\times LOC\rightarrow 2^R$,记为

$\zeta(CS,S(\text{enable\_}r(r,LOC)))=\{r\mid r\in R\wedge r\in \text{sessionRoles}(s)\wedge \text{contains}(CS,LOC)=\text{TRUE}\}$

b)角色状态无效约束(RoleState-Area2):$S\times LOC\rightarrow 2^R$,记为

$\zeta(CS,S(\text{disable\_}r(r,LOC)))=\{r\mid r\in R\wedge r\in \text{enableSessionRoles}(s)\wedge \text{contains}(CS,LOC)=\text{FALSE}\}$

用户被分配了空间角色 r_1,为了执行 r_1 允许的某些操作他首先必须激活 r_1,因此在将 r_1 加入到用户会话的激活角色集前要触发空间约束。

当用户试图在空间位置 LOC 上激活角色 r_1 时,调用函数 active_r,这就引发了事件 E_1,该事件触发 OITE 规则检查用户在位置 LOC 上能否激活角色 r_1。

EVENT E_1 = activate_r(u, r_1, LOC)
RULE[RoleState-Area1
ON E_1 IF (user in user$L$)&&(sessionId *in* session$L$) &&(sessionId in user_sessions(user)) &&(r_l NOT IN session_role(u,s,r_l,LOC)) && enable_r(r_1,LOC)&&Contains($<L_2(X_2,Y_2,L_4(X_4,Y_4)>$,$LOC$)&&enable_$r$($r_l$,$LOC$)&& ((assign_$u$($u$,$r_1$,$LOC$) is TRUE) ‖ (assign_$u$($u$,$r$,$LOC$) &&($r$, r_l) $\in RH$) is TRUE))
then add_session_r (u, r_l, sessionId, LOC)
else raise error "Access Denied Cannot Activate"]

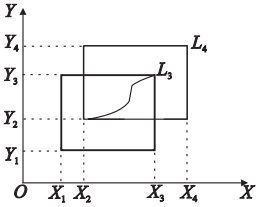

图1　控制角色状态的空间对象区域约束

当用户要在空间位置 LOC 上激活角色 r_l 时使用 RoleState-Area1 规则。首先检查用户是否存在 user$L$ 中,会话标志符 sessionId 是否存在于列表 session$L$ 中,且用户 u 是否拥有该会话。一旦检查通过,则使用函数 assign_u 和 enable_r 检查在位置 LOC 上用户是否已经被分配角色 r_l 或者 r_l 的高级角色,并且 r_l 是否为有效状态,因为在空间角色在有效前被激活是不起作用的;然后使用 session_role 检查 r_l 是否已经在将被激活的会话中被激活,最后检查用户当前的位置 LOC 是否包含在空间区域 $\langle L_2(X_2,Y_2,L_4(X_4,Y_4)\rangle$ 中。一旦上述所有的条件都被检查通过,则调用函数 add_session_r 将角色 r_l 添加到激活角色集中,r_l 在该用户会话中被激活。

2.2.2　空间对象职责分离约束的实施

空间职责分离约束是在某空间位置上施加在特定空间角色集上的限制,在支持空间特性的 RBAC 模型中引入空间职责分离约束是因为在实际的应用场合中需由不同的人行使不同的职责,达到相互牵制的目的。

定义 2　两个及以上角色不能同时无效(RoleState-SSoD)。对冲突角色集合 R,在同一空间位置 LOC 不可能有两个或两个以上角色处于无效状态。

例如,在医院某办公室,nurse 和 doctor 角色不可能同时处于无效状态。对于医院和办公室区域,找出其最小外包矩形,然后用这个最小外包矩形的左上角点 L_i 和右下角点 L_j 来表示该区域,那么医院和办公室区域分别表示为([start$D$],[end$D$])和([officeL_j],[officeL_i])。

EVENT start$D$ = event corresponding to date expression
EVENT end$D$ = event corresponding to date expression
EVENT E_1 = disable_r(nurse,LOC)
EVENT E_2 = disable_r(doctor,LOC)
EVENT E_3 = OR(E_1, E_2)
EVENT E_4 = aperiodic([start$D$],E_5,[end$D$])
EVENT E_5 = aperiodic([officeL_j],E_3,[officeL_i])
RULE[RoleState-SSSoD
ON E_4
if(if disable_r (nurse,LOC) = = true
((if enable_r(doctor,LOC) is TRUE) return true
else return FALSE)
else if disable_r(Doctor,LOC) = = true
((if enable_r(nurse,LOC) is true) return true
else return FALSE))
then (if disable_r (nurse,LOC) = = true then disable_r (nurse,LOC)
else if disable_r(doctor,LOC) = = true then disable_r(doctor,LOC))
else (if disable_r(nurse,LOC) = = true raise error "denied as doctor Already Disabled"
else if disable_r(doctor,LOC) = = true raise error "denied as nurse already disabled"]

RoleState-SSSoD 限制在医院某办公室不允许 doctor 和 nurse 两个角色同时处于无效状态。使用([start$D$],[end$D$])和([officeL_j],[officeL_i])描述医院和办公室区域,他们可以表示任意类型的简单事件和复杂事件,例如,[start$D$]可以表示处于最小外包矩形左上角点 L_i 的右下方事件,[end$D$]可以表示处于最小外包矩形右下角点 L_j 的左上方事件。当在空间位置 LOC 上使角色 nurse 无效时,发生事件 E_1,这将会触发 OR 事件 E_3,并且传递触发 aperiodic 事件 E_4 和 E_5。当要在医院的办公室使角色 nurse 无效时,事件 E_4 就触发了规则 RoleState-SSSoD。RoleState-SSSoD 通过检查角色 doctor 在该办公室是否有效来确定 nurse 是否可以无效,如果 doctor 为有效状态那么 Nurse 可以为无效,反之发出错误信息;当用户要求 doctor 在该办公室无效时使用同样的方法。

2.2.3　空间对象角色激活基数约束的实施

使用空间信息来定义基数限制的方法可以让人们细粒度地表达访问控制策略。一个角色处于激活状态意味着至少有一个用户激活了这个角色。当某一角色处于激活状态,并发生了去除激活事件,这时,如果仅有一个会话使用到该角色,那么该角色转为无效状态,否则,该角色仍然保持激活状态。处于有效或者激活状态的角色,可在无效事件的影响下,转为无效状态。

定义 3　角色激活基数约束(RA-Area)。记为 $\theta(LS, \max r, S(\text{activate\_}r(r, LS)))$。其中:$LS=\langle L_i, L_j\rangle$ 表示空间区域;activate_$r(r, LS)$ 表示在该空间区域内角色 r 被激活;max$r$ 为角色最大数目约束。

限制在一个特定空间区域内,一个角色激活基数的最大上限,不管用户是否是在不同会话或是由不同用户激活角色。例如在学校中,正校长的职位只能由一个用户担任。下面将给出在某个空间区域内角色 r_1 最多被激活五次的例子。同样,用户角色最大数目约束(maxur)即限制在一个特定空间区域内,一个角色被一个特定用户激活基数的最大上限也可以使用类似的实施机制。

EVENT E_1 = add_session_r (u, r_1, LOC, sessionId)
EVENT E_2 = deactivate_r(u, r_1, LOC)
RULE[RA-Area1
　ON E_1
　if if($\theta(LS_j, S(\text{activate\_}r(r_1, LS_j))) \leqslant 5$) return TRUE
　　else return FALSE
　　THEN perform action 〈add role r_1 to session with sessionId〉
　　ELSE raise error "Maximum Number of Roles Reached"]
　rule[RA-Area2
　　ON E_2
　　if true
　　then $\theta(LS_j, S(\text{activate\_}r(r_1, LS_j))) = \theta(LS_j, S(\text{activate\_}r(r_1, LS_j))) - 1$]

上述角色激活基数约束限制在空间区域 LS_j 内,角色 r_1 激活基数的最大上限为 5。角色 r_1 被激活五次数后,此角色将不能再被激活,即使该角色仍处于有效状态。因此用户激活 r_1 的会话放入阻塞会话队列。当一个用户试图在位置 LOC 上激活角色 r_1 时,就触发 3.2.1 节激活角色 r_1 的 RoleState-Area1。当用户满足所有的条件时,函数 add_session_r 被调用,将 r_1 添加到该用户会话的激活角色集中,调用这个函数就引发事件 E_1,该事件触发规则 RA-Area1。通过使用函数 $\theta(LS_j, S(\text{activate\_}r(r_1, LS_j)))$ 返回角色 r_1 在区域 LS_j 中被激活的次数,检查在该区域是否角色激活的最大上限已经达到,如果未达到,那么该角色就允许被激活,否则 RA-Area1 发出"maximum number of roles reached"的出错报告。同样,当角色 r_1 被撤销时,事件 E_2 就发生,并且函数 $\theta(LS_j, S(\text{activate\_}r(r_1, LS_j)))$ 被调用,减少在该区域 r_1 被激活的基数使得该角色在新的会话中能够被激活。

用户在一个空间区域内建立会话,如果存在与该会话相关约束,系统自动触发该约束,控制会话运行;如果不存在与该会话相关约束,则会话一直运行到结束或有其他会话中止当前这个会话为止。会话在它的生命周期中将在运行、阻塞、错误和结束四个状态间转换,如图 2 所示。一个会话由阻塞到运行,或由运行到阻塞的状态转换,其依据是对约束的判定。用户创建一个会话,如果会话满足与该会话特性相关的约束,则放入运行会话队列,如果不满足则放入阻塞会话队列,如果会话违反系统安全策略,则结束该会话。

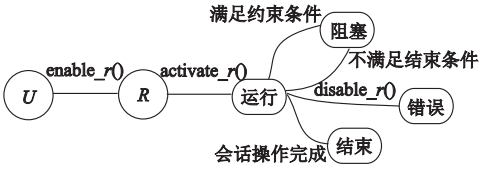

图2　会话状态间的转换

3　事件驱动框架

对访问控制策略的支持不但依赖于系统提供语言的表达能力,而且依赖于实施策略的框架。为了满足需要,传统的 RBAC 已经被广泛扩展,以支持更多的应用。因此,就需要系统也进行相应的扩展或修改,以支持 RBAC 的扩展。而且系统应该拥有一个通用的灵活的框架支持这些扩展。

同一个安全策略可以产生不同的空间约束,可以通过图形化的用户界面或者 XML 文件定义安全策略,初始的规则集从安全策略的定义中自动产生。修改安全策略相应的各种主动授权规则就被自动的修改。在实现中,用户使用图形工具(如 widget 工具包),采取 3.1 节描述的实体——关系模型的形式定义高级访问控制策略。图 3 显示了使用 OITE 规则实施空间约束的事件驱动框架,该框架处理访问控制策略的定义、策略的验证、规则的产生和维护该框架支持的所有功能都是基于事件驱动的。

角色表、用户表等包括所有用户、角色等信息。支持空间特性 RBAC 安全策略的定义是由管理员使用用户图形界面或者 XML 文件定义。一旦策略定义好,这些策略就被策略验证模块验证:在系统表中是否存在这样的角色、对象、用户等。OITE 生成器模块从已被验证的策略中产生初始授权规则集实施支持空间特性的 RBAC。

一旦授权规则被创建,这些规则就被用来控制对客体的访问。基于条件/约束,用户自动扮演角色。当安全策略发生变化时,各种授权规则就被自动重写。

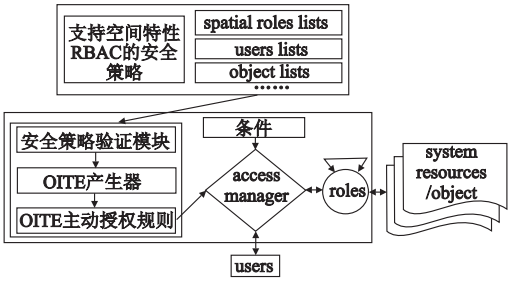

图3　基于OITE规则的事件驱动框架

(下转第 1502 页)

$$C_3 = \{\underset{\substack{j=1,2,\cdots,4\\ j\in X_{B_3}}}{x_j}\}\oplus(\{\underset{x'_{t_3}\in X'_A}{x'_{t_3}}\}\oplus s_1) = \{\underset{\substack{j=1,2,\cdots,4\\ j\in X_{B_3}}}{x_j}\}\oplus(\{\underset{x'_1\in X'_A}{x'_1}\}\oplus s_3) = (00000001) = 1$$

于是 $y_1 = f(C_1) = 47; y_2 = f(C_2) = 91; y_3 = f(C_3) = 14$。

(3)A 在公告牌上公开有序数组 $(y_1, y_2, y_3) = (47, 91, 14)$。

2)密钥恢复阶段

假设任意两个子密钥持有者(不妨设为 B_1、B_2)欲恢复系统密钥,首先 B_1、B_2 从公告牌上查到 $y_1 = 47, y_2 = 91$,然后向 A 询问 t_1 和 t_2,利用其秘密子密钥 $s_1 = 3, s_2 = 1$ 计算出 $C_1 = 4$, $C_2 = 8$;然后利用 Lagrange 内插多项式:

$$f(x) = \sum_{i=1}^{t} y_i \prod_{1\leqslant j\leqslant t, j\neq i} \frac{x - C_j}{C_i - C_j}$$

计算出 $f(x) = \sum_{i=1}^{t} y_i \prod_{1\leqslant j\leqslant t, j\neq i} \frac{x - C_j}{C_i - C_j} = 47\frac{x-8}{4-8} + 91\frac{x-4}{8-4} = 3 + 11x \bmod 13$。于是系统密钥 $s = f(0) = 3$。

5 结束语

基于灰色加密系统,本文给出了一种新的密钥分存方案。除了具有保密性、可行性外,本方案具有以下一些特点:

a)秘密子密钥可多次使用。恢复密钥时,每个成员提交的是其屏蔽子密钥 $C_i = \{\underset{\substack{j=1,2,\cdots,m\\ j\in X_{B_i}}}{x_j}\}\oplus(\{\underset{x'_{t_i}\in X'_A}{x'_{t_i}}\}\oplus s_i)$,其他成员无法通过 B_i 求出 s_i 的秘密子密钥 。即每个成员的秘密子密钥并没有因为系统密钥的恢复而被公开,从而可以继续使用。

b)系统更新。若出于某种原因而需要更换系统密钥时,系统只需重新选择一个 $(l-1)$ 次多项式 $f'(x)$,满足 $f'(0) = s'$ 为新的系统密钥,然后利用新的 $(l-1)$ 次多项式 $f'(x)$ 更新公告牌上的有序数组 $(y_1, y_2, \cdots, y_k)$ 即可。

c)增删成员

(a)当有新成员加入时,系统只需为新成员 B_{k+1} 随机地生成子密钥 s_{k+1},并在公告牌上的有序数组 $(y_1, y_2, \cdots, y_k)$ 中增加一个元素 y_{k+1}。其中 $y_{k+1} = f(C_{k+1})$ 即可。

(b)当要删除某个成员 B_i 时,系统只需重新选择一个 $(l-1)$ 次多项式 $f'(x)$,满足 $f'(0) = s'$ 为系统密钥,然后利用新的 $(l-1)$ 次多项式 $f'(x)$ 更新公告牌上的有序数组 $(y_1, y_2, \cdots, y_k)$,此时无须计算 y_i(可令 y_i 仍为原值或置 y_i 项为空),那么,B_i 原有的子密钥 s_i 即可无效。

d)更新个别的秘密子密钥。当某个成员 B_i 的子密钥泄密时,系统只需为该成员重新分配子密钥 s'_1,之后重新选择一个 $(l-1)$ 次多项式 $f'(x)$,满足 $f'(0) = s$,并利用新的 s'_1 和 $f'(x)$ 更新公告牌上的有序数组 $(y_1, y_2, \cdots, y_k)$,而不必更改其他成员的子密钥。

特别说明:本文是全部研究成果的一部分。基于多齿中央加密系统的密钥分存方案,将在后面的文章中给出。

参考文献:

[1] SHAMIR A. How to share a secret[J]. Proc of Communications of the ACM, 1979, 22(11): 612-613.

[2] BACKLY G R. Safeguarding cryptographic keys[C]// Proc of Proceedings of the National Computer Conference of AFIPS. Montvale, NJ: AFIPS Press, 1979: 313-317.

[3] DENG J L. Control problems of gray system[J]. Systems and Control Letters, 1982, 1(5): 285-294.

[4] DENG J L. Introduction to gray system theory[J]. The Journal of Grey System, 1989, 1(1): 1-24.

[5] SHI K Q. Gray information relation theory[M]. Taipei: Quan Hua science and Technology Press, 1994: 121-124.

[6] SHI K Q, CHEN T S. A grey general lock and central public cryptosystem(I)[J]. Journal of Grey System, 2000, 12(4): 331-340.

[7] CHEN T S, SHI K Q. A grey general lock and central public cryptosystem (II) [J]. Journal of Grey System, 2001, 13(1): 57-64.

[8] SHI K Q, CHEN T S. On the grey encryption problems of information security(I) [J]. Journal of Grey System, 2000, 12(3): 215-224.

[9] CHEN T S, SHI K Q. On the grey encryption problems of information security(II)[J]. Journal of Grey System, 2000, 12(3): 255-262.

(上接第 1499 页)

4 结束语

本文研究了 OITE 规则如何作为一种通用的机制在不同粒度上实施各种带有空间特性角色的约束。该约束实施机制非常灵活,不但能够基于系统状态的变化实时作出相应的动作,而且能够满足安全策略和角色结构的变化。同时简要讨论在 OITE 规则实施支持空间特性 RBAC 的事件驱动框架中如何由安全策略自动产生规则。

还有一些问题需要在后续的研究中解决。例如,各种空间约束应该被形式化描述,产生的各种规则应该被验证。

参考文献:

[1] FERRAIOLO D F, SANDHU R, GAVRILA E, *et al*. Proposed NIST standard for role-based access control[J]. ACM Trans on Information and System Security, 2001, 4(3): 224-274.

[2] BACON J, MOODY K, YAO W. A model of OASIS role-based access control and its support for active security [J]. TISSEC, 2002, 5(4): 492-540.

[3] 张宏, 贺也平, 石志国. 一个支持空间上下文的访问控制形式模型[J]. 中国科学 E 辑: 信息科学, 2007, 37(2): 254-271.

[4] JOSHI J B D *et al*. A generalized temporal role-based access control model[J]. IEEE Trans on Knowledge and Data Engineering, 2005, 17 (1): 4-23.

[5] HANSEN F, OLESHCHUK V. Spatial role-based access control model for wireless networks[C]// Proc of the 58th IEEE Vehicular Technology Conference (VTC'03). Orlando: IEEE Computer Society, 2003: 2093-2097.

[6] BERTINO E, CATANIA B, LUISA DAMIANI M, GEO-RBAC: a spatially aware RBAC[J]. ACM Trans on Information Systems and Security, 2006: 1-34.

[7] ADAIKKALAVAN R, CHAKRAVARTHY S. SnoopIB: interval-based event specification and detection for active databases[J]. Data and Knowledge Engineering (DKE), Elsevier, 2006, 59(1): 139-165.

[8] RAMAKRISHNAN R, GEHRKE J. Database management systems [M]. 3rd ed. New York: McGraw-Hill, 2000.