ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.8, August 2007, pp.2002−2015
DOI: 10.1360/jos182002

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

细粒度的基于信任度的可控委托授权模型*

翟征德+, 冯登国, 徐 震

(中国科学院 软件研究所 信息安全国家重点实验室,北京 100080)

Fine-Grained Controllable Delegation Authorization Model Based on Trustworthiness

ZHAI Zheng-De+, FENG Deng-Guo, XU Zhen

(State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62661713, Fax: +86-10-62661700, E-mail: zhaizhengde@is.iscas.ac.cn, http://home.is.ac.cn

Zhai ZD, Feng DG, Xu Z. Fine-Grained controllable delegation authorization model based on trustworthiness. *Journal of Software*, 2007,18(8):2002−2015. http://www.jos.org.cn/1000-9825/18/2002.htm

Abstract: A fine-grained controllable delegation authorization model (FCDAM) suitable for open environments is presented. It integrates the merits of both RBAC (role based access control) and role-based trust management and can effectively control the propagation of permissions of different sensitivity levels in roles. An approach for assigning trustworthiness thresholds to permissions in local access control policy is discussed. The RT_0 framework is extended to support trustworthiness and the algorithm of calculating the values of trustworthiness of entities in the extended framework is proposed. The usage of the FCDAM model is illustrated through a typical example.
Key words: delegation; trustworthiness; propagation of permission; trust management; authorization model

摘 要: 综合基于角色的访问控制和信任管理各自的优势,提出了一个适用于开放式环境的细粒度可控委托授权模型——FCDAM(fine-grained controllable delegation authorization model),基于信任度实现了对角色中具有不同敏感度的权限传播控制.提出了为本地策略中的权限分配信任度阈值的方法,为RT_0添加了信任度支持,给出了在这种扩展后的信任管理系统中计算实体信任度的算法,并结合具体实例对模型的使用进行了说明.
关键词: 委托;信任度;权限传播;信任管理;授权模型
中图法分类号: TP309 **文献标识码**: A

基于角色的访问控制(role based access control,简称RBAC)[1,2]通过引入角色的概念实现了用户和权限的逻辑分离,权限被授予角色,用户通过管理员为其分配的角色获得相应的权限,显著地降低了授权管理的代价[3].RBAC的另一个优势是可以较好地支持授权约束[4].在Internet这样的开放式环境中,存在大量彼此陌生的自治实体,实体之间经常需要动态地进行数据和服务的交换.这种计算环境难以沿用传统的集中控制式资源管理模式,资源的拥有者也难以依据陌生实体的身份进行授权,这使得传统的RBAC难以直接在这种场景下应用.基于角色的信任管理[5,6],提供了一种在开放式环境下的陌生实体之间进行授权的有效途径,为实现Internet环境下网

* Supported by the National Natural Science Foundation of China under Grant No.60603017 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z454 (国家高技术研究发展计划(863)); the National Key Technology R&D Program of China under Grant No.2006BAH02A02 (国家科技支撑计划)

Received 2007-02-26; Accepted 2007-05-31

络资源的虚拟化提供了有力的技术支持,其核心的授权机制是通过委托实现的.然而,这些信任管理模型大都没有很好地解决委托深度控制问题.例如,RT_0[5]中的授权凭证$A.r_1 \leftarrow B.r_2$实际上是将为角色$A.r_1$分配成员的权力委托给了实体B,B可以通过颁发$B.r_2 \leftarrow D$来使D获得$A.r_1$,也可以通过$B.r_2 \leftarrow C.r_3$将这种权力再次委托给C,而后,C又可以颁发$C.r_3 \leftarrow E$来使E获得$A.r_1$.从A的角度来看,随着委托深度的增加,实体离授权源越远,其可信程度越低.另外,位于委托路径中前部的委托授权的信任度会影响所有后继委托的受托者获得授权的信任度,例如,如果$B.r_2 \leftarrow C.r_3$的信任度较低,则E取得$A.r_1$的信任度也将较低.总之,影响实体取得授权的信任度的因素包括委托深度和委托路径中委托自身的信任度.在这种情况下,首先,A可能希望只有那些信任度大于某个阈值的实体才能激活$A.r_1$;其次,即使那些可以激活$A.r_1$的实体之间也存在着可信程度上的差别,而角色$A.r_1$中的某些权限可能是高度敏感的,并非$A.r_1$的所有成员都可以访问,只有信任度大于某个给定阈值的实体才能行使这些权限.由此可见,基于角色的委托模型需要在角色和权限两个层次上进行委托深度控制,前者限制可以获得该角色的成员;后者根据角色中权限的不同敏感度提供细粒度的权限传播控制.这两者是相互联系的:如果一个实体的信任度不足以使他行使角色中的任何一个访问权限,则允许实体获得该角色是没有意义的.传统的信任管理模型不能有效支持这种细粒度的委托控制,难以保证权限传播的可控性,不利于系统的安全性.

本文综合RBAC和信任管理各自的优势,提出了一个适合开放式环境的委托授权模型(fine-grained controllable delegation authorization model,简称FCDAM),较好地解决了上述权限传播控制问题,有助于精确实现Internet环境下网络资源的访问控制,为构建面向互联网资源共享的虚拟计算环境提供了有力的授权技术支持.该模型为RBAC中的角色/权限分配关联信任度阈值以标识权限的敏感性,基于信任管理的方法建立起陌生实体到本域角色之间的授权分配并计算实体的信任度,通过比较实体信任度和角色中权限的信任度阈值,确定其是否可以获得该权限.RBAC角色层次引起的权限继承会影响角色中权限的信任度阈值,本文给出了一种在层次RBAC下为角色中的权限分配信任度阈值的方法.在传统的基于角色的信任管理中,一个实体要么完全信任另一个实体,要么完全不信任,这与现实情况不符.此外,由于链接名称(linked name)和角色交集(intersection)等主体形式的存在,凭证链不一定是简单的线性结构,而可能是复杂的图结构,这使得实体信任度的计算复杂化.本文对RT_0语言进行了扩展,使其支持可以量化的委托信任度,提出了一种在扩展后的信任管理框架中计算实体信任度的方法,并通过扩展文献[5]中的算法实现了一个随着凭证发现过程计算实体信任度的算法.

1 相关工作

文献[7–9]对 RBAC 模型中的委托深度控制进行了探讨,提出了各自的约束机制和实施方案.这些工作使用的委托约束大都直接指定基于本次委托而进行的后继委托的受托者的标识或者属性信息(如所在的用户组),是一种适用于封闭环境和已知用户群体的约束,而在开放式的、基于未知用户群体的环境下,基于本次委托进行的后继委托的参与者是不可预料的;其次,这些工作采用的约束实施机制大都假设存在 1 个或者几个中心节点对系统中新的委托凭证进行合法性验证,而这与开放式环境的非集中式管理特性是相冲突的.开放式环境中的每个管理域都是独立的自治实体,控制一组资源,拥有自己的管理和访问控制策略,可以随意发布自己的委托授权凭证,系统很难在委托凭证发布时进行合法性检查,较为可行的方法是在凭证链发现过程中动态地进行检查.此外,文献[8,9]的委托方案都是完全委托(每次委托整个角色),不能支持对角色中权限的细粒度委托控制.

信任管理[10–14]已引起了研究者的广泛关注,然而,已有的工作大都集中在信任管理引擎的构建上,对于委托过程中的权限传播问题没有给予足够的重视.Keynote[11]没有采用任何委托控制.SPKI采用布尔值来控制委托过程,可以指定本次委托是否允许任何后继委托,其表达能力是非常有限的.Cassandra[12],DL[13]和RT_{+0}[14]都是用整数值进行委托深度控制,允许指定可以基于本次委托而进行的后继委托的深度.这种方法仅通过委托深度来表达实体的可信度,不能完整和精确地反映实体的可信程度.RT_0[5]和RT_1[6]都没有考虑委托过程中权限传播控制的问题.文献[15]首先提出了使用主观信任度来进行委托深度控制的思想,然而,该工作存在着如下两个不足:(1) 其授权源(本地访问控制策略)采用的是访问控制列表(ACL),而 ACL 是一种延展性较差的方案,难以在分布式环境中广泛应用;(2) 其提出的信任计算方法假设凭证链是线性结构,而这种假设对于任何一个实际可用的

信任管理系统都是难以满足的.不同于传统的 PKI 公钥证书链,基于角色的信任管理中委托凭证的主体可以表现为链接名称和角色交集等形式,凭证链表现为一种图结构,这复杂化了信任度的计算过程.

文献[16]提出了使用信任关系来实现陌生实体到本域角色的分配.该文没有考虑对委托深度进行限制,同样也不支持对角色中的权限施加细粒度的传播控制.

2 基于信任度的授权策略

传统的基于角色的信任管理忽略了角色和权限的区别,同样采用委托凭证的方式来表示角色/权限分配,从而有利于信任管理引擎进行一致性验证(proof of compliance).然而,这也带来两个缺陷:(1) 角色/权限分配通常代表了本管理域的访问控制策略,这种信息在本域内是无条件可信的,对于其他域而言也不是必需的.因此,这种表示方式并不是必然的,它不仅会增加凭证链发现和验证的代价,而且会造成本域访问控制策略的泄漏;(2) 采用委托凭证表示角色/权限分配不利于RBAC约束的实施.在RBAC中,角色/权限间的使用约束[4]如何在传统的基于角色的信任管理中实施,还是一个未知的问题.与这些传统的信任管理模型不同,本文明确区分角色与访问权限,将角色/权限分配看作是本域的访问控制策略(这与规范[17]是一致的),使用信任管理的方法来建立实体/角色分配,远端的实体通过委托凭证链取得本域中的角色,通过本域的访问控制策略获得访问权限.这种方法保持了系统的开放性,并且可以较为容易地支持RBAC授权约束,因此,它综合了信任管理和RBAC各自的优势,而避免了其缺陷.

在RBAC中,角色是一组权限的集合,角色中的权限可能存在敏感程度上的差异.例如,多个医疗机构结成联盟关系,彼此向对方开放自己的医疗信息,来自其他医疗机构的工作人员通过域间信任管理系统获得本医疗机构中的高级用户(prime users)角色,从而可以查询本机构的医疗统计信息和医疗记录.显然,与查询医疗统计信息相比,查询个人医疗记录是一项高度敏感的权限,只有那些对于本机构而言具有较高信任度的用户才能行使该权限.而按照传统的信任管理方式,所有能够获得该角色的用户彼此没有区别,都能够行使角色中的所有权限,这是不恰当的.

本文通过为每个角色/权限分配关联一个信任度阈值来反映权限的敏感程度,只有那些以大于该阈值的信任度取得角色的实体才能获得访问权限.角色的授权权限不仅包括其直接分配的权限,还包括通过角色层次继承来的权限.一个重要的问题是如何确定这些继承来的权限的信任度阈值.在角色中逐一地为所有这些权限指定阈值显然不是一个延展性好的方案,而且不利于保持RBAC原有的授权语义.本文中采用的方法是为角色继承关系指定信任度阈值衰减系数,使得同一权限在后代角色(descendent)[2]中关联的信任度阈值通过角色层次上的信任度阈值衰减后在祖先角色(ascendent)[2]中自动获得一个新的、相对较小的信任度阈值.

定义 1(RBAC 模型基础元素).

ROLES,*OPS*,*OBS* 分别代表角色集、操作集、对象集.

$PRMS \subseteq 2^{(OPS \times OBS)}$为权限集.

定义 2(信任度、信任度阈值、信任度阈值衰减系数).

$TLS=TTS=[0.0,1.0]$分别为信任度和信任度阈值的集合.

设 $t \in TLS$,$t=1.0$ 表示该实体是完全可信的;$t=0.0$ 表示实体是完全不可信的.

$TACS=[0.0,1.0]$是信任度阈值衰减系数的集合.

定义 3(带有信任度阈值的角色/权限分配). $PA \subseteq ROLES \times PRMS \times TTS$ 是带有信任度阈值的角色/权限分配.设 $(r,p,t) \in PA$,则一个以信任度 $t'(t' \geq t)$取得角色 r 的实体可以行使权限 p.信任度阈值是一种对角色中权限的使用(usage)约束.

定义 4(角色层次). $DRH \subseteq ROLES \times ROLES$,是角色间的直接支配关系.*RH* 是 *DRH* 的自反、传递闭包,代表了角色层次.

例 1:表 1 和图 1 给出了一个网上书店的访问控制策略,包括带信任度阈值的角色/权限分配和角色层次.任何人都可以通过取得角色 Guest 浏览该网站的商品(*p_view*);Ordinary 角色允许网上定购商品(*p_order*)和进行

在线电子支付(*p_credit*);Discount 角色允许购书时享受折扣(*p_discount*);Special 角色允许享受货到付款服务(*p_pod*)和延期付款(例如购买半年之内)服务(*p_delay*),后者被认为是一项敏感权限,所以其信任度阈值较高.

Table 1 Permission assignment of the RBAC policy example

表 1 示例 RBAC 策略的角色权限分配

Role	Permission	Threshold value
Guest	*p_view*	0.0
Ordinary	*p_order*	0.70
Ordinary	*p_credit*	0.70
Discount	*p_discount*	0.80
Special	*p_pod*	0.60
Special	*p_delay*	0.94

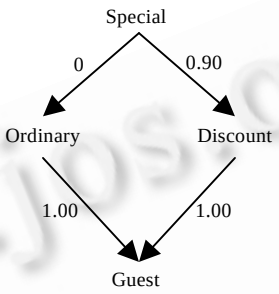

Fig.1 Role hierarchy of the RBAC policy example

图 1 示例 RBAC 策略的角色层次

RBAC 角色层次上的祖先角色继承后代角色的所有权限.若$(r,p,t)\in PA$,则一个以大于等于 t 的信任度取得 r 的实体可以行使权限 p.设$(r',r)\in DRH$,则 r'继承 r 的所有权限.现在的问题是,一个实体以怎样的信任度取得 r'时可以行使权限 p.在现实生活中,对于行使同一权限的两个不同用户,通常认为职务较高的人更为可信,这是因为与取得一个较低的职位相比,用户取得较高的职位需要更为可靠的背景和行为历史,这本身蕴含了更高的可信度.例如,同样是打开银行的保险柜这种行为,行长通常比出纳具有更高的置信度.同样,对于行使访问权限 p,一个以信任度 t 取得角色 r'的用户要比一个以同样信任度取得 r 的用户具有更高的置信度.因此,一个以信任度 t'取得 r'的用户,即使 $t'<t$,仍然可以行使 p,即角色 r'中权限 p 的信任度阈值低于或等于它在 r 中的阈值.这对于保证授权系统的合理性是十分必要的,因为若假设 r'中 p 的信任度阈值为 t'并且 $t'>t$,若用户 u 以信任度 $t''(t<t''<t')$取得了角色 r',则 u 不能直接从 r'中访问 p,但由于 u 至少可以信任度 t''激活 r,则他可以通过角色 r 访问到 p.本文通过在角色层次上引入信任度阈值衰减系数使得后代角色中的权限自动地在祖先角色中获得一个较小的信任度阈值.

定义 5(角色层次上的信任度阈值衰减、函数 fCoeff、函数 fAttenu).

$ATTENU\subseteq DRH\times TACS$,是角色层次上的每个直接继承关系到其对应的信任度阈值衰减系数的映射.若$((r',r),m)\in ATTENU,(r,p,t)\in PA$,则角色 r'中 p 的信任度阈值为 $m\times t$.图 1 给出了示例策略的角色层次及其信任度阈值衰减系数.

函数 $fCoeff(drh:DRH)\rightarrow TACS$ 给出两个存在直接继承关系的角色之间的信任度阈值衰减系数,$(drh, fCoeff(drh))\in ATTENU$.

函数$fAttenu(l:RH)\rightarrow 2^{TACS}$给出角色层次上两个存在继承关系(直接或间接)的角色之间的信任度阈值衰减的积累效果, $fAttenu((r_n,r_0))=\left\{m\mid m=\prod_{i=n-1,\ldots,0\wedge(r_{i+1},r_i)\in DRH} fCoeff(r_{i+1},r_i)\right\}$. $fAttenu((r_0,r_n))$的值是一个集合,这是因为在r_n和r_0之间可能有多条路径.若$(r_0,p,t)\in PA$且$(r_n,r_0)\in RH$,则r_n中权限p的信任度阈值为$t\times\min(fAttenu(r_n,r_0))$.例如在图 1 中,从guest到special存在两条路径,从guest经ordinary到special的衰减积累效果是 1.00×0.80,从guest经

discount到special的衰减积累效果是 1.00×0.90.

信任度阈值衰减系数是本域中访问控制策略的一部分,由本域的策略管理员负责设置.设$(R_a,R_b)\in DRH$,管理员如果认为R_b中的权限对于角色R_a的用户成员而言仍然是相对敏感的,则可以指定一个接近 1 的系数;否则,如果希望R_a的用户成员能够较容易地行使R_b中的权限,则可以指定一个较小的系数.信任度阈值衰减系数和信任度阈值的引入没有改变RBAC策略的基本性质,如果在没有引入信任度的RBAC策略中,角色r能够拥有权限p,则在对应的引入信任度后的RBAC中,角色r仍然拥有p,只不过此时p关联了一个信任度阈值,只有那些可信度高于此阈值的实体才能实际地行使该访问权限.

定义 6(函数 authorizedPrms). 函数 $authorizedPrms(r'{:}ROLES)$计算角色 r'拥有的权限及其信任度阈值,

$$authorizedPrms(r')=\{(p,t')|\exists r{:}ROLS\cdot\exists t{:}TTS\cdot\exists a{:}TACS\cdot r'\geq r\wedge(r,p,t)\in PA\wedge a\in fAttenu(r',r)\wedge t'=a\times t\}.$$

表 2 给出了示例 RBAC 策略中角色的授权权限及其信任度阈值.

Table 2 Authorized permissions and their trustworthiness thresholds

表 2 授权权限及其信任度阈值

Role	Permission	Threshold value	Role	Permission	Threshold value
Guest	*p_view*	0.0	Special	*p_view*	0.0
Ordinary	*p_view*	0.0	Special	*p_order*	0.56
Ordinary	*p_order*	0.70	Special	*p_credit*	0.56
Ordinary	*p_credit*	0.70	Special	*p_discount*	0.72
Discount	*p_view*	0.0	Special	*p_pod*	0.60
Discount	*p_discount*	0.80	Special	*p_delay*	0.94

定义 7(角色的激活信任度阈值).

角色的激活信任度阈值是实体激活该角色所需要的最小信任度,其值等于该角色所有直接分配的权限的信任度阈值的最小值.如果实体的信任度不足以使他访问该角色中的任何直接分配的权限,则允许其激活该角色是没有意义的.

函数 $tRoleActivation(r{:}ROLES)\rightarrow TTS$ 给出角色 r 的激活信任度阈值,

$$tRoleActivation(r)=\min(\{t|\exists p{:}PRMS\cdot(r,p,t)\in PA\}).$$

例如,$tRoleActivation(special)=0.60$.

本节对传统 RBAC 策略进行了扩展,通过在角色/权限分配上引入信任度阈值来表示角色中不同权限的敏感度,实体必须以超过此阈值的信任度取得角色才能行使该权限.本节只考虑了 RBAC 策略中的角色/权限分配及其信任度阈值,一个完整的授权模型还必须解决另一个问题:实体(尤其是其他域的陌生实体)如何取得本域的角色及其取得角色时的信任度.

3 信任度的计算

本节首先在RT_0中引入主观信任度,讨论了在这种扩展后的信任管理框架中计算信任度的方法,然后给出了一个同步进行凭证链发现和信任度计算的算法ProofGraphTrust.

3.1 信任度计算模型

在FCDAM中,远端的实体通过委托凭证链向本管理域证明,他可以取得本域中的某个角色.本授权模型使用了与RT_0类似的委托形式,不同之处在于,FCDAM中的每个委托都与一个信任度关联,它代表了委托人对本次委托的信任程度.在RT_0中,委托人要么完全信任受托者并为其签发委托凭证,要么完全不信任受托者并且不为其颁发凭证,不能表达一定程度上的信任.这里通过为每个RT_0凭证关联一个主观信任度来解决这个问题,并将这种扩展后的信任管理框架称为RT_0T(RT_0 with trustworthiness).将信任度引入RT_0并没有改变其原有的授权方式和过程,RT_0T仍然保持RT系统原有的性质(如单调性、凭证图的完备性等),只是增强了授权过程的可控性.下面给出 4 种委托凭证及其语义解释:

(1) $A.r\leftarrow B$ with t,其中,A 和 B 都是实体名,r 是角色名,t 是信任度.其含义是,A 定义 B 以信任度 t 取得 $A.r$ 的

成员身份.它通常用于定义本域中的实体/角色分配.

(2) $A.r \leftarrow B.r_1$ with t,其中,A和B是两个不同的实体名,r和r_1是角色名,t是信任度.其含义是,A定义所有$B.r_1$的成员可以信任度t取得$A.r$的成员身份.它通常允许一个管理域(A)以一定的信任度(t)向另一个域(B)委托定义本域中角色($A.r$)的成员的权力.

(3) $A.r \leftarrow A.r_1.r_2$ with t,其中,A是实体名,r和r_1是角色名,t是信任度,$A.r_1.r_2$称为链接角色.其含义是,如果实体B具有角色$A.r_1$,且B又定义C具有$B.r_2$角色,则C被A承认以信任度t取得角色$A.r$.这种形式允许以一定的信任度向其他实体(所有具有$A.r_1$角色的实体)委托定义本域中角色($A.r$)的成员的权力,而不需要指定这些实体的具体名称.

(4) $A.r \leftarrow f_1 \cap f_2 \cap \ldots \cap f_n$ with t,其中,f_i是一个实体名、角色或以A开头的链接角色,$f_1 \cap f_2 \cap \ldots \cap f_n$称为角色交集.其含义是,$A$定义同时具有所有$f_i$的成员身份的实体以信任度$t$取得角色$A.r$.

在开放式环境下,每个域自主地定义自己的委托凭证(如域A可以颁发所有以$A.r_1, A.r_2, \ldots, A.r_n$开头的委托凭证).每个凭证都负责将本域中的角色委托给本域的用户(类型(1)的凭证),或者委托给满足一定条件的外域用户(类型(2)~类型(4)),并且声明一个信任度t,该信任度反映了本域对这次委托过程的信任程度.如果本域认为受托者是高度可信的,则t的值应该接近于 1;否则,t的值应该较小.由于每个域都只能直接委托本域中的角色(域A只能委托$A.x$角色),每个域在签发委托凭证时必须依据受托者的可信程度来指定一个确切的信任度t;否则,可能造成事实上足够可信的外域实体不能访问本域中的资源(t过小)或者不够可信的实体访问到了本域中的资源(t过大),这对于本管理域都是不利的,因此,每个域在委托时都会尽力指定合适的信任度.外域的实体取得本域中的某个角色所需要的凭证链包含了多个中间域各自颁发的委托凭证,实体取得本域角色的最终可信度要综合考虑凭证链中所有凭证蕴含的信任度.

在PKI系统中,身份证书仅包含了实体的名称和公钥信息,实体在进行身份认证时构建的证书链是一个简单的线性结构.在单纯的线性结构中,整个链的最终信任度可以由链路中各个证书蕴含的信任度经过简单的计算得到,例如,链路的最终信任度可以是链路中证书信任度的乘积.然而,实用的信任管理系统需要表达较为复杂的实体间的委托关系,例如,RT_0中的链接角色和角色交集,这使得凭证链不再保持线性结构,而是形成一种图结构,从而复杂化了信任度的计算.

文献[5]给出了RT_0的凭证链发现算法ProofGraph.它是通过构建一个证明图(proof graph)来实现的.本节首先介绍了在RT_0T中计算信任度的方法,然后给出了通过改进ProofGraph以实现在RT_0T中同步进行凭证链发现和信任度计算的ProofGraphTrust算法.

下面给出在例 1 的网上图书销售场景中使用的委托凭证.该书店(store)隶属于一个全国性的连锁销售集团(org),该书店和一些大学结成协作联盟关系,为这些大学的教师提供特殊的销售服务(货到付款和延期支付等).为了推进这种联盟关系的发展,该书店通过委托凭证允许某些已经与其结成联盟关系的大学向其推荐新的大学加入到联盟中.这个场景涉及了 5 个管理域及其各自颁发的凭证.

Store:

Store.ordinay←Org.member with 1.0 //总部的注册会员可以使用书店的普通服务

Store.special←Org.member∩Store.ally.teacher with 1.0 //联盟中大学的教师如果同时又是总部的会员,则可以享有特殊优惠

Store.ally←UniA with 0.96; //UniA 是联盟大学

Store.ally←UniA.recommended with 0.9; //向 UniA 委托推荐新大学进入联盟的权力

UniA:

UniA.recommended←UniB with 0.8 //UniA 推荐 UniB 进入联盟

UniA.recommended←UniB.recommended with 0.85; //UniA 向 UniB 委托推荐新大学进入联盟的权力

UniA.teacher←Li with 1.0;

UniB:

UniB.recommended←UniC with 0.84; //UniB 推荐 UniC 进入联盟

UniB.teacher←Wang with 1.0;

UniC:

UniC.teacher←Liu with 1.0

Org:

Org.member←Li with 0.95; Org.member←Wang with 1.0; Org.member←Liu with 0.58

上述凭证对应的凭证图如图 2 所示.设 e 是实体,r'是角色或链接角色或角色交集,我们关心的是 e 能否取得 r'以及 e 取得 r'的信任度(如果能够取得).如果 e 能够取得 r',则其信任度取决于从 e 到 r'的路径.例如,图 2 中 Li 取得角色 Store.special 的信任度取决于路径 Store.special $\xleftarrow{*}$ Li 中各条边的信任度,Liu 取得角色 Store.ally.teacher 的信任度取决于路径 Store.ally.teacher $\xleftarrow{*}$ Liu 中各条边的信任度.这种路径中可能包含 3 种类型的边:

(1) $A.r\leftarrow e'$(e'是实体、角色、链接角色或角色交集).它是由凭证 $A.r\leftarrow e'$ with t 直接得到的.例如图 2 中的边:
org.member←Li,Store.special←Org.member∩Store.ally.teacher,Store.ally←UniA.recommended.

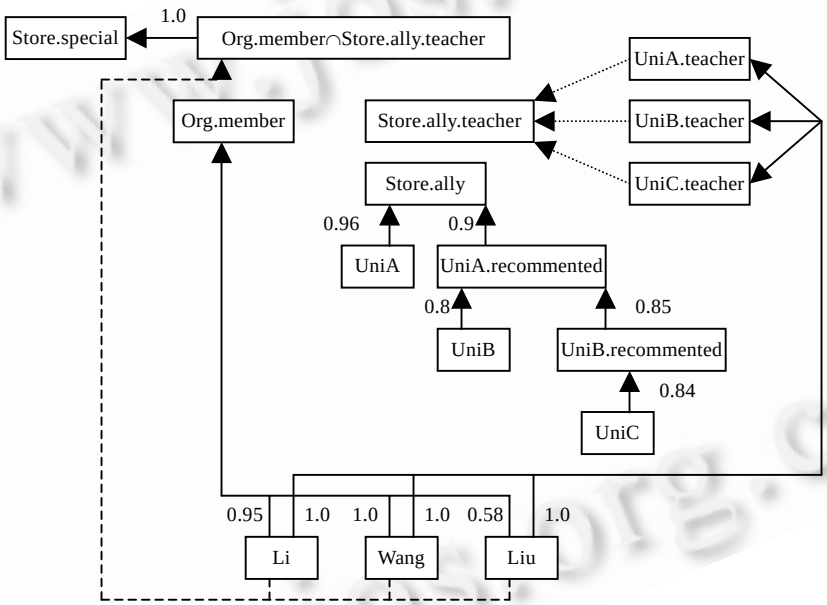

Fig.2 An example of credential graph

图 2 凭证图示例

(2) $f_1\cap f_2\cap\ldots\cap f_n\leftarrow e$($e$是实体).它不是直接由凭证得到的,而是由凭证图中的其他证明路径导出的,这些路径包括 $f_1\xleftarrow{*}e$, $f_2\xleftarrow{*}e$,..., $f_n\xleftarrow{*}e$, 例如,图 2 中的边 Org.member∩Store.ally.teacher←Li 是由路径 Org.member←Li 和路径 Store.ally.teacher $\xleftarrow{*}$ Li 导出的.

(3) $A.r_1.r_2\leftarrow B.r_2$.它是由如下证明路径导出的:$A.r_1\xleftarrow{*}B$.例如,边Store.ally.teacher←UniB.teacher是由路径 Store.ally $\xleftarrow{*}$ UniB导出的.其中类型(2)和类型(3)的边称为导出边,导出边所依赖路径的集合称为其支持集.

假设前面讨论的 3 种类型的边的信任度都是已知的,并设函数 toe 返回边的信任度,则可以计算整个路径的信任度.设 $e_{n+1}\xleftarrow{*}e_0$ (n>0)中包含的边依次为$e_{n+1}\leftarrow e_n$,$e_n\leftarrow e_{n-1}$,…,$e_1\leftarrow e_0$,设函数top计算路径的信任度,则top的定义如下:

$$top(e_{n+1}\xleftarrow{*}e_0)=\begin{cases}toe(e_1\leftarrow e_0), & n=0\\ f(toe(e_{n+1}\leftarrow e_n),top(e_n\xleftarrow{*}e_0)), & n>0\end{cases}.$$

其中,f 是某个预定义的函数,用于从同一路径上前后两个信任度计算一个合成的信任度.f 需要满足:(1) 对于任意 $a\in[0,1.0],b\in[0,1.0],0\le f(a,b)\le 1.0$,这保证在输入两个合法的信任度时,$f$ 总是能得到一个新的合法的信任度;(2) $f(a,b)\le\min(a,b)$,这保证随着委托深度的增加,信任度总是衰减的.满足上述条件的 f 包括乘法运算和 min 函数等.例如,设 f 为乘法,且设边 Org.member∩Store.ally.teacher←Li 的信任度为 m,则路径 Store.special $\xleftarrow{*}$ Li 的信任度为 $1.0\times m$.在本文中,除非特别说明,f 默认为乘法操作.

由上述讨论可知,如果 $e_{n+1}\xleftarrow{*}e_0$ 中所有边的信任度都是已知的,则整条路径的信任度是容易计算的.下面讨论如何计算上述 3 种边的信任度.

1) 对于(1)型边,它们直接对应于委托凭证,因此其信任度就是对应的凭证中包含的信任度 t.

2) 对于(2)型边,它是由其支持集中的路径导出的,因此其信任度取决于支持集中路径的信任度的最小值,即 $toe(f_1\cap f_2\cap\ldots\cap f_n\leftarrow e)=\min(top(f_1\xleftarrow{*}e),\ldots,top(f_n\xleftarrow{*}e))$.这里使用min操作来确定导出边的信任度,其含义是实体 e 能以可信度 t 取得 $f_1\cap f_2\cap\ldots\cap f_n$ 当且仅当它至少能以可信度 t 取得每一个 f_i,例如,

toe(Org.member∩Store.ally.teacher←Li)=min(top(Org.member ←Li),top(Store.ally.teacher $\xleftarrow{*}$ Li)).

3) 对于(3)型边 $A.r_1.r_2\leftarrow B.r_2$,它是由路径 $A.r_1\xleftarrow{*}B$ 导出的,所以,$toe(A.r_1.r_2\leftarrow B.r_2)=top(A.r_1\xleftarrow{*}B)$.例如,$toe$(Store.ally.teacher←UniB.teacher)=top(Store.ally←UniB)=toe(UniA.recommended←UniB)×toe(Store.ally←UniA.recommended).

第(2)、(3)类型边是由其支持集导出的,支持集中的路径中可能还含有其他(2)、(3)类型的边,因此,上述计算过程是递归的.

在上述计算模型中,实体最终取得角色的信任度取决于委托路径中的每步委托的信任度,当 f 函数为乘法运算时,所有委托步对最终信任度的影响都是相同的,委托路径中任何一次委托的可信度较低,都将使得实体取得角色的最终可信度较低,从而使得实体难以获得角色中高敏感度的权限.委托路径中有些边并不直接对应于凭证,而是从其支持路径中导出的,因此,当确定这些边的信任度时,需要计算其支持路径的信任度.这是一个递归的过程,最终会归结到直接对应于凭证的边的信任度.例如,要计算图 2 中 Wang 取得 Store.ally.teacher 的信任度,

top(Store.ally.teacher $\xleftarrow{*}$ Wang)=toe(UniB.teacher←Wang)×toe(Store.ally.teacher←UniB.teacher)

=1.0×top(Store.ally $\xleftarrow{*}$ UniB)

=1.0×(toe(UniA.recommended←UniB)×toe(Store.ally←UniA.recommended))

=1.0×(0.8×0.9).

ProofGraphTrust 算法.

ProofGraphTrust 算法基于文献[5]中的 ProofGraph 算法,但对其进行了扩展,实现了同步的凭证链发现和信任度计算.ProofGraphTrust 的核心思想是:利用新发现的凭证不断扩展证明图,根据凭证类型确定监视器(monitor)关联的信任度,将新确定的角色的成员及成员取得该角色的信任度向前传播,将新确定的实体的角色及实体取得该角色的信任度向后传播.附录给出了算法的细节.

下面通过实例说明该算法的执行.初始时,把 store.ally.teacher 加入到 *b-proc-queue* 中,调用该节点的 *b-activate*(),然后调用 ProofGraphTrust 节点的 *run*()函数,执行后向搜索算法,得到的证明图如图 3 所示.为了清楚地说明算法的执行过程,图中的每个节点上都标明了其创建次序、获得的后向解及这些解的可信度.在图中,有向边代表一个监视器(monitor),边上的数字表示该监视器关联的信任度.从图中可以看出,该算法正确地计算出了 3 个可以取得 Store.ally.teacher 角色的用户及其相应的信任度.ProofGraphTrust 算法是对 ProofGraph 的改进,实现同步的凭证发现和信任度计算,但没有改变原算法的安全特性.根据文献[5]中的定理 6 和定理 7,ProofGraphTrust 算法的有效性和完备性可以得到保证.

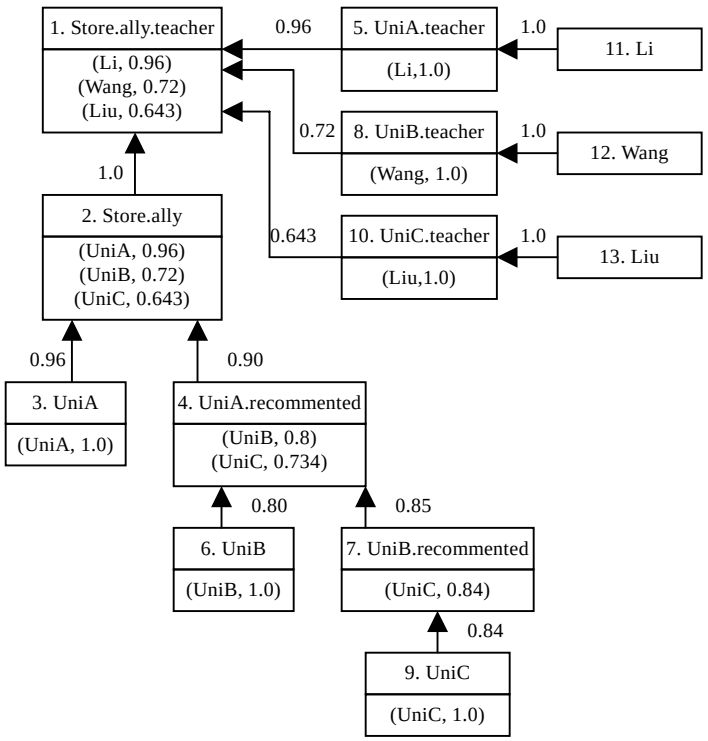

Fig.3 An example of the execution of *ProofGraphTrust.run*()

图 3 *ProofGraphTrust.run*()执行示例

定理 1. 对于RT_0T的凭证集合C,设N为C中凭证的数目,$M=size(C)$(定义见文献[5]),并假设找到所有定义角色$A.r$的凭证所耗费的时间线性于这种凭证的数目,则算法ProofGraphTrust的时间复杂度为$O(N^3+NM)$,空间复杂度为$O(NM)$.

证明:算法的执行时间为执行过程中所有解通过证明图上的边进行传播的次数所限定,空间复杂度为证明图的节点数、边数和所有节点的解的数目所限定.设C中所有带有信任度的凭证去掉信任度以后得到的凭证集合为C',则在C上执行算法ProofGraphTrust和在C'上执行ProofGraph的凭证链发现过程是相同的.除了solution和monitor的形式发生改变之外(关联了信任度参数),两种算法产生的证明图是相同的,为相同的图节点产生的solution的数目也是相同的,并且这些solution在图中传递的过程和次数也是相同的.因此,两种算法具有相同的时间复杂度和空间复杂度.参照文献[5]的定理 3,ProofGraphTrust的时间复杂度为$O(N^3+NM)$,空间复杂度为$O(NM)$. □

下面通过实例来说明完整模型的使用.设例 1 中的书店采用了如图 1 所示的访问控制策略,各个域采用前述委托凭证进行信任管理,则表 3 给出了 ProofGraphTrust 计算出的实体取得角色的信任度.

Table 3 The trustworthiness values of entities getting roles

表 3 实体取得角色的信任度值

Entity	Role	Trustworthiness value
Li	Store.special	0.95
Wang	Store.special	0.72
Liu	Store.special	0.58

用户 Li 可以行使角色 Store.special 中的所有权限,Wang 可以行使该角色中除了延期支付之外的所有权限,Liu 则不能激活 special 角色.

4 模型的安全性质

在一个完整的授权过程中,实体首先向远端的域提交资源访问请求(op,o),系统查找包含了该访问权限的本地角色,然后使用 ProofGraphTrust 算法计算实体是否能够取得该角色及其信任度,将这个信任度与角色中权限的信任度阈值作比较可以确定实体是否能够行使该权限.

定理 2. 实体 e 能够获得本域访问权限 p 当且仅当凭证图中存在一条凭证路径 l 证明他可以信任度 t 取得某个本域角色 r,满足$(p,m)\in authorizedPrms(r)$且 $t\geq m$.

证明:必要性.e 获得权限 p 的唯一方法是通过取得本域中包含了 p 的某个角色 r,如果凭证图中不存在这样一条路径,则 e 无法取得相关角色,进而无法获得 p.如果 e 通过任何一条凭证路径 $r \xleftarrow{*} e$ 取得 r 的信任度都小于 m,则根据模型的授权条件,e 无法获得 p.

充分性.如果存在这样一条路径 $r \xleftarrow{*} e$,对 r 节点执行 *activate*()函数,然后调用 ProofGraphTrust 算法,根据文献[5]的定理 7,e 最终会成为 r 的后向解,并且,该算法计算出 e 取得 r 的信任度 t'满足 $t'\geq t$(当不存在信任度更高的路径时,有 $t'=t$),由于 $t'\geq t\geq m$,所以,e 能够获得 p. □

与传统的信任管理模型相比,FCDAM 的优势在于细粒度的权限传播控制.它提供了两种手段来控制角色中敏感权限的传播:(1) 管理员可以提高本地访问控制策略中权限的信任度阈值,例如上述例子中,Store 域的管理员为延期支付这种敏感权限设置了值为 0.94 的信任度阈值,这保证了只有充分可信的实体才能行使该权限;(2) 在委托过程中,委托者可以在为受托者签发的凭证中设定恰当的信任度,以限制所有通过该委托和后继委托取得角色的实体的信任度,例如,Store 通过签发凭证 Store.ally←UniA.recommended 并设置信任度为 0.90,就可以使所有由 UniA 委托授权的实体(包括 UniA 直接委托的和所有后继委托的受托实体)都不可能行使延期支付权限.

5 结束语

RBAC降低了权限管理代价,并且提供了良好的授权约束支持,但其传统的用户/角色分配方式不能满足像Internet这样的开放式环境下陌生实体间授权的需求.基于角色的信任管理通过委托为开放式环境提供了一种灵活、有效的授权方式,但大多数模型都没有很好地解决委托深度控制问题.本文综合两者各自的优势,提出了一个基于信任度的委托授权模型,实现了在开放式环境下对角色中的权限的细粒度传播控制.本文通过为角色/权限分配关联信任度阈值的方法较好地反映了角色中权限的不同敏感程度.传统的基于角色的信任管理系统不能有效地描述实体间的信任程度.实用的信任管理系统中凭证链呈现较为复杂的图结构,而不再是简单的线性结构.本文为RT_0语言添加了信任度支持,给出了在这种扩展的信任管理系统中计算实体信任度的算法,从而可以实现基于实体的信任度控制其所能获得的角色中的访问权限.FCDAM实现了一种适用于开放式环境的支持细粒度的权限传播控制的委托授权方案,有助于精确实现Internet环境下网络资源的访问控制.

致谢 在此,我们向本文的评审专家表示感谢,感谢他们对本文所提出的宝贵意见和建议.

References:

[1] Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-Based access control models. IEEE Computer, 1996,29(2):38−47.

[2] ANSI INCITS 359-2004. Role based access control. American National Standard for Information Technology, 2004.

[3] Ferraiolo DF, Cugini J, Kuhn DR. Role-Based access control (RBAC): Features and motivations. In: Proc. of the 11th Annual Computer Security Application Conf. New Orleans: IEEE Computer Society Press, 1995. 241−248.

[4] Joshi JBD, Bertino E, Latif U, Ghafoor A. A generalized temporal role based access control model. IEEE Trans. on Knowledge and Data Engineering, 2005,17(1):4−23.

[5] Li NH, Winsborough WH, Mitchell JC. Distributed credential chain discovery in trust management (full version). In: Proc. of the 8th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2001. 156−165. http://crypto.stanford.edu/

~ninghui/papers/disc.pdf

[6] Li NH, Mitchell JC, Winsborough WH. Design of a role-based trust management framework. In: Heather H, ed. Proc. of the IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society Press, 2002. 114−130.

[7] Wainer J, Kumar A. A fine-grained, controllable user-to-user delegation method in RBAC. In: Proc. of the 10th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2005. 59−66.

[8] Bandmann O, Dam M, Firozabadi BS. Constrained delegation. In: Proc. of the 23rd Annual IEEE Symp. on Security and Privacy. Oakland: IEEE Computer Society Press, 2002. 131−143.

[9] Zhang LH, Ahn GJ, Chu BT. A rule-based framework for role-based delegation. In: Sandhu RS, Jaeger T, eds. Proc. of the 6th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2001. 153−162.

[10] Blaze M, Feigenbaum J, Lacy J. Decentralized trust management. In: Proc. of the '96 IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society Press, 1996. 164−173. http://citeseer.ist.psu.edu/blaze96decentralized.html

[11] Blaze M, Feigenbaum J, Ioannidis J, Keromytis A. The KeyNote trust-management system version 2. IETF RFC 2704, 1999.

[12] Becker MY, Sewell P. Cassandra: Distributed access control policies with tunable expressiveness. In: Proc. of the 5th IEEE Int'l Workshop on Policies for Distributed Systems and Networks (POLICY 2004). Los Alamitos: IEEE Computer Society Press, 2004. 159−168.

[13] Li NH, Grosof BN, Feigenbaum J. Delegation logic: A logic-based approach to distributed authorization. ACM Trans. on Information and System Security (TISSEC), 2003,6(1):128−171.

[14] Hong F, Zhu X, Wang SB. Delegation depth control in trust-management system. In: Proc. of the 19th Int'l Conf. on Advanced Information Networking and Applications (AINA 2005). Washington: IEEE Computer Society, 2005. 411−414.

[15] Liao JG, Hong F, Zhu GM, Yang QW. Trustworthiness-Based authorization delegation model. Chinese Journal of Computers, 2006, 29(8):1265−1270 (in Chinese with English abstract).

[16] Chakraborty S, Ray I. TrustBAC-Integrating trust relationships into the RBAC model for access control in open systems. In: Proc. of the 11th ACM Symp. on Access Control Models And Technologies. New York: ACM Press, 2006. 49−58.

[17] Organization for Advancement of Structured Information Standard (OASIS). Core and hierarchical role based access control (RBAC) profile of XACML v2.0. 2005.

附中文参考文献:

[15] 廖俊国,洪帆,朱更明,杨秋伟.基于信任度的授权委托模型.计算机学报,2006,29(8):1265−1270.

附 录

算法 1. ProofGraphTrust.

ProofGraphTrust 是一种面向对象风格的算法,包含了两个基本类:ProofGraphTrust 和 ProofNodeTrust,以及 3 个帮助类:BLinkingMonitorTrust,BIntersectionMonitorTrust,FlinkingMonitorTrust. 该算法对文献 [5] 中的 ProofGraph 进行了修改和扩展.

ProofGraphTrust 的实例变量:

nodes:证明图节点;edges:证明图中的边;*b-proc-queue*:后向处理队列;*f-proc-queue*:前向处理队列.

ProofNodeTrust 的实例变量:

b-proc-state:后向处理状态;

f-proc-state:前向处理状态;

b-solutions:前向解集合,每个解都是(*s*,*t*)的形式,*s* 为实体,*t* 为信任度;

f-solutions:后向解集合,每个解都是(*s*,*t*)的形式,*s* 为角色,*t* 为信任度;

b-sol-monitors:后向解监视器集合,每个监视器为(*n*,*l*)的形式,*n* 为图节点,*l* 为从当前节点向 *n* 传播 solution 时的信任度阈值衰减系数;

f-sol-monitors:后向解监视器集合,每个监视器为(*n*,*l*)的形式,*n* 为图节点,*l* 为从当前节点向 *n* 传播

solution 时的信任度阈值衰减系数.

```
ProofGraphTrust: run()
{    while (one of b-proc-queue and f-proc-queue is not empty)
     {if (b-proc-queue nonempty) {n=b-proc-queue.dequeue(); n.b-process();}
     if (f-proc-queue nonempty) {n=f-proc-queue.dequeue(); n.f-process();}}
}
ProofGraphTrust: addNode(e)
{     if (a node n exists for e) {return n;} else {creates one and return it;}
}
ProofGraphTrust: addEdge(e2←e1,t)
{     n1=addNode(e1); n2=addNode(e2);
     if (edges.contains(n2←n1))) {return;}
          edges.add(n2←n1);
     n1.add-b-sol-monitor(n2,t);   //添加monitor
     if (n2.b-proc-state!=unprocessed) {n1.b-activate();}
     n2.add-f-sol-monitor(n1,t);   //添加monitor
     if (n1.f-proc-state!=unprocessed) {n2.f-activate();}
}
ProofNodeTrust(A.r): b-process()
{     b-proc-state=processed; creds=find all credentials defining A.r;
     foreach (credential "A.r←e with t" in creds)
          {addNode(e);addEdge(A.r←e,t);}   //新添加边的信任度来源于凭证
}
ProofNodeTrust(A.r1.r2): b-process()
{     b-proc-state=processed; n=addNode(A.r1);
     n.add-b-sol-monitor(new BLinkingMonitorTrust(A.r1.r2),1);   //此时,monitor的信任度阈值衰减系数为 1
     n.b-activate();
}
BLinkingMonitorTrust(A.r1.r2): add-b-solution(B,t)
{     addNode(B.r2);
     addEdge(A.r1.r2←B.r2,t);   //边的信任度等于B到A.r1的路径的信任度
}
ProofNodeTrust(D): b-process()
{     b-proc-state=processed;
     add-b-solution(D,1);   //D 对自身是一个信任度为 1 的 solution
}
ProofNodeTrust(f1∩…∩fk): b-process()
{     b-proc-state=processed; m=new BIntersectionMonitorTrust(f1∩…∩fk);
     foreach (j in 1,...,k) {n=addNode(fj);
          n.add-b-sol-monitor(m,1);   //monitor 的信任度阈值衰减系数为 1
          n.b-activate();}
}
```

BIntersectionMonitorTrust($f_1\cap\ldots\cap f_k$): *add-b-solution*(B,t)
{ if (B is being added for the first time) {n=*addNode*(B); n.*f-activate*();}
else if (B has been added k times)
{let (B,t_1)$\in f_1$.*b_solutions*,…,(B,t_k)$\in f_k$.*b_solutions*;
addEdge($f_1\cap\ldots\cap f_k\leftarrow B$, min($t_1$,...,$t_k$)); //新边的信任度为支持集中各路径信任度的最小值
b-solutions.add(B,min(t_1,…,t_k));} //添加后向解
}
ProofNodeTrust(e): *f-process*()
{ *f-proc-state=processed*;
if (e is a role $B.r_2$) {*add-f-solution*($B.r_2$,1);
n=*addNode*(B); //$B.r_2$对自身信任度为 1
n.*add-f-sol-monitor*(new *FLinkingMonitorTrust*($B.r_2$),1); //衰减系数为 1
n.*f-activate*();}
creds=find all credentials having e as its body;
foreach (credential $A.r\leftarrow e$ with t) {*addNode*($A.r$);
addEdge($A.r\leftarrow e$,t);} //边信任度来源于凭证
if (e is an intersection) {return;}
creds=find all credentials like $A.r\leftarrow f_1\cap\ldots\cap f_k$ with t in which $f_j=e$ for some j;
foreach (credential $A.r\leftarrow f_1\cap\ldots\cap f_k$ with t in creds)
{*add-f-solution*($\langle f_1\cap\ldots\cap f_k,j\rangle$,1); //信任度为 1
n=*addNode*($f_1\cap\ldots\cap f_k$); n.*b-activate*();}
}
FLinkingMonitorTrust($B.r_2$): *add-f-solution*($A.r_1$,t)
{ *addNode*($A.r_1.r_2$);
addEdge($A.r_1.r_2\leftarrow B.r_2$,$t$); //边的信任度等于$B$到$A.r_1$的路径的信任度
}
ProofNodeTrust(e): *add-b-solution*(s,t)
{ if ((s,l) exists in *b-solutions*){if (l<t) replace((s,l),(s,t)); //保留信任度较大的解
return;}
b-solutions.add(s,t);
for each ((n,c) in *b-sol-monitors*){
n.*add-b-solution*(s,f(t,c));} //计算 solution 的新信任度并向前传播
}
ProofNodeTrust(e): *add-f-solution*(s,t)
{ if ((s,l) exists in *f-solutions*) {if (l<t) replace((s,l),(s,t)); //保留信任度较大的解
return;}
f-solutions.add(s,t);
foreach ((n,c) in *f-sol-monitors*) {n.*add-f-solution*(s,f(c,t));} //计算 solution 的新信任度并向后传播
if (e is an entity D && s is a partial solution ($f_1\cap\ldots\cap f_k$,j) && all k pieces have arrived)
{let the k pieces be (f_1,t_1),…,(f_1,t_k);
addEdge($f_1\cap\ldots\cap f_k\leftarrow D$, min($t_1$,…,$t_k$)); //新边的信任度为支持集中各路径信任度的最小值
f-solutions.add_solutions($f_1\cap\ldots\cap f_k$, min(t_1,…,t_k));} //添加前向解

```
}
ProofNodeTrust(e): add-b-sol-monitor(n,t)
{       b-sol-monitors.add(n,t);
      foreach ((s,l) in b-solutions) {n.add-b-solution(s,f(l,t));}   //根据新的 monitor 向前传播解
}
ProofNodeTrust(e): add-f-sol-monitor(n,t)
{       f-sol-monitors.add(n,t);
      foreach ((s,l) in f-solutions) {n.add-f-solution(s,f(t,l));}   //根据新的 monitor 向后传播解
}
```

翟征德(1979－),男,山东淄博人,博士生,主要研究领域为信息与系统安全.

徐震(1976－),男,博士,副研究员,主要研究领域为信息与系统安全.

冯登国(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络与系统安全.

****

第 9 届国际信息与通信安全会议(ICICS 2007)

征 稿 通 知

2007 年国际信息与通信安全会议是(ICICS 2007)第 9 届 ICICS 系列会议，将于 2007 年 12 月 12 日－15 日在中国河南郑州召开。与前 8 届 ICICS 系列会议相同，ICICS 2007 将为国内外信息安全学者与专家齐聚一堂，提供探讨国际信息安全前沿技术的难得机会。作为国际公认的第一流国际会议，ICICS 2007 将进一步促进国内外的学术交流，促进我国信息安全学科的发展。本次学术会议将由中国科学院软件研究所和北京大学软件与微电子学院主办，由中安科技集团承办，并得到国家自然基金委员会和河南省人民政府信息化办公室的大力支持。

会议欢迎来自全世界所有未发表过和未投递过的原始论文，内容包括，但不限于以下内容：访问控制；计算机病毒与蠕虫对抗；认证与授权；应用密码学；生物安全；数据与系统安全；数据库安全；分布式系统安全；电子商务安全；欺骗控制；网格安全；信息隐藏与水印；知识版权保护；入侵检测；密钥管理与密钥恢复；基于语言的安全性；操作系统安全；网络安全；风险评估与安全认证；无线安全；安全模型；安全协议；可信计算。

投稿须知：作者提交的论文，必须是未经发表或未并行地提交给其他学术会议或学报的原始论文。所有提交的论文都必须是匿名的(没有作者名字，单位名称，致谢或其他明显透露身份的内容)。论文必须用英文，并以 PDF 或 PS 格式提交。排版字号为 11pt，且论文不能超过 12 页(A4 纸)。所有提交论文必须在无附录的情形下是可理解的。如果提交论文未遵守上述要求，论文作者将自行承担论文未通过形式审查而拒绝接受论文的风险。审稿将由 3 位程序委员匿名评审，评审结果为：以论文形式接受；以短文形式接受；拒绝接受。

ICICS2007 会议论文集将由德国 Springer 出版社作为 LNCS 系列出版，可在会议期间获取。凡接受论文的作者中，至少有 1 位必须参加会议，并在会议上报告论文成果。

投稿截止时间：2007 年 8 月 1 日

通知接受时间：2007 年 9 月 17 日

发表稿提交截止时间：2007 年 10 月 1 日

大会网址：http://www.ICICS2007.org.cn/