带时间特性的角色授权约束*

董光宇, 卿斯汉, 刘克龙

(中国科学院 软件研究所 信息安全技术工程研究中心,北京 100080)
E-mail: dgy@ercist.iscas.ac.cn; l_kelong@sina.com
http://www.ercist.ac.cn/

摘要: 授权约束是基于角色的访问控制模型中一个非常重要的部分.已经有方法来对约束进行形式化的描述,但主要集中在有关约束的静态特性的描述上.提出一套形式化地描述时间特性的模型,使之能够描述带时间特性的授权约束.对已有的角色授权约束模型,即约束描述语言进行了深入研究,并进行了相应的扩展.这种扩展后的语言称为 RCLT(role-based constraints language with time-character).同时,对 RCLT 的时间效率的优化以及在 RCLT 规则被违反时如何恢复合法状态的问题进行了形式化的讨论.RLCT 语言被证明可以很好地表示出对授权约束的时间特性的需求,但在效率优化和状态恢复方面要找到实用、高效的通用算法还需要作进一步的工作,这也是将来研究工作的主要目标.

关 键 词: 信息安全;访问控制;角色;约束;时间特性
中图法分类号: TP309 文献标识码: A

访问控制是信息安全和网络安全中非常重要的一个方面.本文要讨论的问题主要基于一种较新的访问控制模型:基于角色的访问控制(role-based access control,简称 RBAC).在 RBAC 之中,访问客体的权力被赋予角色,而不是用户[1].当一个角色被赋予一个用户时,此用户就拥有了一个角色所包含的权限.这种机制降低了复杂性、成本和错误授予访问权力的可能性.

在 RBAC 模型中,授权约束(authorization constraint,也可简单称为 Constraint,即约束)规定了访问许可被赋予角色时,或角色被赋予用户时,以及当用户在某一时刻激活一个角色时所应遵循的强制性规则[2].G.-J.Ahn[3]提出了一个形式化表达约束的方法,使用了一种称为 RCL2000(role-based constraints language2000)的语言,其语义的合理性和完全性得到了很好的证明,并且具有很强的表现能力.但它仍缺少对时间特性的处理.

在本文中,我们专注于对授权约束的时间特性的研究.需要明确的是,我们追求的不是如何实现一两种特定的时间约束,而是要构建一般的、形式化的表达方法.我们将在无时间特性的授权约束的形式化表达的基础上,作时间特性的扩展,以利用已有的研究成果.

本文第 1 节介绍 RBAC 模型以及用来表述角色授权约束规则的规范语言 RCL2000.第 2 节对授权约束的时间特性需求进行分析.第 3 节提出一个带时间特性的约束描述语言,它是对 RCL2000 的扩展,称为 RCLT(role-based constraints language with time-character).第 4 节讨论了 RCLT 的表现能力.第 5 节总结了全文.

1 RCL2000 的结构

RCL2000 的结构包含两部分,RBAC 模型本身的结构和 RCL2000 扩充的结构.

* 收稿日期: 2001-03-28; 修改日期: 2001-08-09
基金项目: 国家自然科学基金资助项目(60083007);国家重点研究发展规划 973 资助项目(G1999035810)
作者简介: 董光宇(1977－),男,广西南宁人,硕士,主要研究领域为信息安全理论与技术;卿斯汉(1939－),男,湖南隆日人,研究员,博士生导师,主要研究领域为信息安全理论与技术;刘克龙(1971－),男,安徽桐城人,博士,助理研究员,主要研究领域为信息安全理论与技术.

在描述和构建 RBAC 时,RBAC96[4]是一个经常被使用的模型,而且 RCL2000 就是基于 RBAC96 的结构的设计,因此我们的框架基本上基于 RBAC96 模型.

1.1 RBAC结构

RBAC96 模型的构成元素如下[4]:

定义 1(**RBAC** 结构).

$U=\{u_1,u_2,...,u_m\}$,是所有用户(user)的集合; $R=\{r_1,r_2,...,r_n\}$,是所有角色(role)的集合; $P=\{p_1,p_2,...,p_o\}$,是所有访问许可权(permission)的集合; $S=\{s_1,s_2,...,s_p\}$,是所有会话(session)的集合.会话在时间特性的研究中占有最重要的地位,因为整个模型的时间特性集中地体现在会话的时间特性上.

$UA\subseteq U\times R$,从用户集合到角色集合的多对多映射,表示用户被赋予的角色. $PA\subseteq P\times R$,从许可集合到角色集合的多对多映射,表示角色被赋予的许可. $SA\subseteq S\times U$,从会话集合到用户集合的函数映射,表示会话与用户的从属关系. $RH\subseteq R\times R$,定义在 R 上的偏序关系,称为角色层次关系.如果 $(r_1,r_2)\in RH$,则定义 $r_1\preceq r_2$.

定义 2(**RBAC** 的全局函数).

$role\_set: U\cup P\cup S\to 2^R$:

$u_i\in U: role\_set(u_i)=\{r\mid (u_i,r)\in UA\}$

$p_i\in P: role\_set(p_i)=\{r\mid (p_i,r)\in PA\}$

$s_i\in S: role\_set(s_i)=\{r\mid (user\_set(s_i),r)\in UA\}$

$user\_set: R\to 2^U: user\_set(r_i)=\{u\mid (u,r_i)\in UA\}$

$user: S\to U: user(s_i)=u,(s_i,u)\in SA$

$permission\_set: R\to 2^P: permission\_set(r_i)=\{p\mid (p,r_i)\in PA\}$

$permission\_set: S\to 2^P: permission\_set(s_i)=\bigcup_{r\in role-set(s_i)} permission\_set(r)$

以上的定义没有考虑层次关系 RH 所具有的作用,在层次关系中,考虑两个角色 r_1 和 r_2,且 $r_1\preceq r_2$,则所有属于 r_2 的用户 u 也应该拥有属于 r_1 的访问许可.所以需要定义几个与层次关系有关的全局函数.

定义 3(反映层次关系的全局函数).

$role\_set^H: U\to 2^R: role\_set^H(u_i)=\{r\mid (\exists r'\succeq r)[r'\in role\_set(u_i)]\}$

$role\_set^H: P\to 2^R: role\_set^H(p_i)=\{r\mid (\exists r'\succeq r)[r'\in role\_set(p_i)]\}$

$role\_set^H: S\to 2^R: role\_set^H(s_i)=\{r\mid (\exists r'\succeq r)[r'\in role\_set(s_i)]\}$

$permission\_set^H: R\to 2^P: permission\_set(r_i)^H=\{p\mid (\exists r\preceq r_i)[p\in permission\_set(r)]\}$

$permission\_set^H: S\to 2^P: permission\_set(s_i)^H=\bigcup_{r\in role-set^H(s_i)} permission\_set(r)$

全局函数的定义是开放的,也就是说,我们可以根据需要添加新的全局函数.

1.2 授权约束(**authorization constraint**)

定义 4(**RBAC** 系统的状态).

我们称在某一时刻,八元组 $q=(U,R,P,UA,PA,SA,RH,S)$,标明了 RBAC 系统在那个时刻的状态[5].

定义 5(授权约束判定).

授权约束判定是一个一元函数,用来判定某个 RBAC 系统的可能的状态 q(是否符合当前的安全策略).

$$constra\,int\_valid: RSP\to\{\text{true},\text{false}\}: constra\,int\_valid(q)=\begin{cases}\text{true}, & \text{此状态被系统接受}\\ \text{false}, & \text{此状态不被系统接受}\end{cases}.$$

其中,RSP 是该 RBAC 系统的状态空间.

1.3 RCL2000的结构

G.-J.Ahn 提出一种形式化表述约束的方法[3],他称这种语言为 RCL2000.在原有 RBAC 结构的基础上,

RCL2000 增加了两个不确定函数.

定义 **6**(不确定函数 ***OE*** 与 ***AO***).

$OneElement: OE(X) = x_i, x_i \in X;\quad AllOther: AO(X) = X - \{OE(X)\}.$

RCL2000 另外还包含了 3 个集合:

$CR = \{cr_1, cr_2, ..., cr_s\}, cr_i \subseteq R$,冲突角色组的集合; $CP = \{cp_1, cp_2, ..., cp_t\}, cp_i \subseteq P$,冲突许可组的集合; $CU = \{cu_1, cu_2, ..., cu_u\}, cu_i \subseteq U$,冲突用户组的集合.

此语言的核心在于 OE 和 AO 两个不确定函数,文献[3]还提出并证明了两个定理:合理性定理(soundness theorem)和完全性定理(completeness theorem),以说明 RCL2000 和严格形式的一阶谓词逻辑(RFOPL)的等价性.下面用一个例子来说明 RCL2000 的语法和语义.例子是一个最简单形式的静态 SOD,为了简化,不考虑层次.

约束要求.没有用户能被赋予两个冲突的角色.即同一个冲突组中的角色不能拥有同一个用户.

表达式 1: $|\ role\_set(OE(U)) \cap OE(CR)\ | \leqslant 1$.

表达式 2(与表达式 1 等价,但形式更复杂): $user - set(OE(OE(CR))) \cap user - set(AO(OE(CR))) = \varnothing$.

表达式 1 的 RFOPL 形式: $\forall u \in U, \forall cr \in CR: |\ role - set(u) \cap cr\ | \leqslant 1$.

RCL2000 本身没有定义时间特性, RCLT 将在 RCL2000 的基础上作时间特性的扩充.

2 对角色授权约束的时间特性的分析

2.1 激活时间约束

规定用户、角色或者访问许可只能在某些时间范围内可以激活.这条约束是很自然的,例如在一个组织中,有工作时间的限制,在非工作时间的范围内,将不允许某些操作的发生,或者不允许某些用户登录进系统.这样做可能是为了安全的因素考虑,也有可能是为了简化管理过程.

2.2 激活时间长度约束

规定用户、角色或者访问许可每次只可以激活不超过一个固定长度的时间.当某个操作很重要、很关键时,如果它激活时间过长,就会有被人盗用而产生严重后果的可能.

2.3 定长时间内激活时间长度限制

规定用户、角色或者访问许可在一定的时间长度内的累计激活时间不超过一个规定的上限.尤其是对于用户的限定特别有用,可以控制用户的平均登录时间.

3 对 RCL 2000 的时间特性扩展(RCLT)

3.1 时间的定义

定义 **7**(时间点序列).

$T = \{..., t_i, t_{i+1}, ...\}$,当 $t_i \in T, i \in Z$ 时,定义集合里的元素是有序的,即 $t_i < t_j \Leftrightarrow i < j, t_i = t_j \Leftrightarrow i = j$,还定义 $t_i - t_j = i - j$.

$t_i \in T$ 代表一个真实世界的时间点,但具体代表哪个时间点,则根据实际的实现而定,为讨论方便,定义一个函数映射 t_i 到真实的时间:

$real\_time(t_i)$:规定 $real\_time(t_i) - real\_time(t_j) > 0$,如果 $i > j$.

这个函数与系统具体实现相关.

若约定 $real\_time(t_{i+1}) - real\_time(t_i) = \Delta t, \Delta t > 0$ 为常数,则更容易理解和处理. $real\_time(t_0)$ 表示 t_0 所代表的真实时间,当此值确定,若 Δt 也确定,则所有 $real\_time(t_i)$ 也都确定了.

定义 **8**(时间区间集合).

$$TR \subseteq T \times T(t_i, t_j) \in TR \Leftrightarrow t_i < t_j .$$

3.2 RCLT结构

RCLT 继承了 RCL2000 的所有元素,并作了相应的扩充.

定义 **9**(**RCLT** 扩展的元素).

$currenttime \in T$,是一个全局的原子函数,表示当前时间.

$OLDS = \{os_1, os_2, ...\}$, 且$OLDS \cap S = \varnothing$,$OLDS$ 为已经不再处在激活状态的会话的集合.我们作一个约定:会话从激活的时候开始,一直到结束,其所包含的角色集合以及其他属性不变.如果一个用户想改变自己某个会话的属性,则可以在结束这个会话的同时,开始另外一个新的会话,结束的会话将被放到 $OLDS$ 里去.

$$STR \subseteq (S \cup OLDS) \times TR, |(\{s\} \times TR) \cap STR| \leq 1, \text{且} (s, tr) \in STR \Leftrightarrow (s \in S) \vee (s \in OLDS).$$

当 $(s, (t_i, t_j)) \in STR$,则 t_i 表示会话 s 的开始(激活)时间,t_j 表示 s 的结束时间.对于 $s \in S$, s 是活动会话,无结束时间,t_j:=$currenttime$+1.

定义 **10**(**RCLT** 的全局函数).

$\text{start} : TR \to T : \text{start}((t_i, t_j)) = t_i$

$\text{end} : TR \to T : \text{end}((t_i, t_j)) = t_j$

$\text{start\_time} : S \cup OLDS \to T : \text{start\_time}(s) = t_i, \text{if } (s, (t_i, t_j)) \in STR$

$\text{end\_time} : S \cup OLDS \to T : \text{end\_time}(s) = t_j, \text{if } (s, (t_i, t_j)) \in STR$

$\text{active\_length} : S \cup OLDS \to T : \text{active\_length}(s) = (\text{end\_time}(s) - \text{start\_time}(s))$

$\text{in\_range} : T \times TR \to \{\text{true}, \text{false}\} : \text{in\_range}(t, (t_i, t_j)) = (t \geq t_i) \wedge (t \leq t_j)$

$\text{not\_in\_range} : T \times TR \to \{\text{true}, \text{false}\} : \text{not\_in\_range}(t, (t_i, t_j)) = (t < t_i) \vee (t > t_j)$

$\text{in\_range} : T \times 2^{TR} \to \{\text{true}, \text{false}\} : \text{in\_range}(t, tr_i) = \neg\text{not\_in\_range}(t, OE(tr_i))$

定义 **11**(**RCLT** 规则).

一条 RCLT 规则有两个部分:

(1) 局部元素定义.一般是几个此规则要使用的集合.

(2) 表达式.

例如,有一个时间规则,要求用户 u_1 只能在某些时间段内激活会话,则此规则可以写成:

规则 1:

局部定义: $ALLOWTR_{u1} = \{altr_1, altr_2, ...\} \subseteq TR$

表达式: $user(OE(S)) = u_1 \Rightarrow in\_range(currenttime, ALLOWTR_{u1})$

如果要求每个用户都有其允许激活的时间段,则此规则可改写为

规则 2:

局部定义: $USERALLOWTR \subseteq U \times TR$

表达式: $in\_range(currenttime, (\{user(OE(S))\} \times TR) \cap USERALLOWTR)$

3.3 RCLT规则的触发时机

RCLT 增加了一个重要的全局函数 $currenttime$,以代表当前时间.这个函数改变了 RBAC 系统的状态表示.新的 RBAC 系统的状态现在表示成一个十元组: $q = \{U, R, P, UA, PA, SA, RH, S \cup OLDS, STR, currenttime\}$.

当 $currenttime$ 每次发生变化时,系统都应该判断一下状态的合法性.这样,规则判断的频率会增大.可以用折衷的方法来缓解效率的问题,比如把规则分成 RCLT 和 RCL2000 两部分,当只是因 $currenttime$ 变化而引起状态变化时,则只进行 RCLT 规则的判定.但如果 RCLT 规则很多,或某些 RCLT 规则本身复杂度较高,则效率还是难以得到提高.

估算一下上面两个 RCLT 规则的计算量.为了节省篇幅,将不作详细的推导和证明.最后得出的结果为

$$O(\sum_{s \in S}(|USERALLOWTR| + |ALLOWTR(user(s))|))$$
$$= O(|S| \cdot |USERALLOWTR|) + O(|USERALLOWTR|)$$
$$= O((|S| + 1) \cdot |USERALLOWTR|).$$

如果当时的 $|S|=10, |USERALLOWTR|=1000$, $\Delta t=1s$, 则每分钟的计算量级将达到 $6.6\cdot 10^5$,这还只是一个较简单规则的计算量.由此看出,效率是加入时间特性后必须解决的问题.

为了解决判定 RCLT 规则的效率问题,我们引进"下一翻转时间函数"的概念.通过分析一般的时间约束策略可以看出,即使 Δt 较小而造成 *currenttime* 变化过快,RCLT 规则的判定结果相对来说仍比较稳定,不会频繁地发生变化.因此我们引入一个概念函数.

定义 **12**(下一翻转时间函数).

$next\_turn\_time(q, X)=nt\in T$, q 是 RBAC 系统的状态,X 是某一个 RCLT 规则.

函数的输入参数有两个,q 为状态十元组,可以写成 $q=(q', STR, currenttime)$,q'不会随时间而变化.X 是一个 RCLT 规则,可以认为是一个一元函数, $X(q)\in\{\text{true},\text{false}\}$.函数的输出结果有如下要求:

设$next_turn_time(q, X)=nt$,则一定有$nt>currenttime$,并有:

$X((q', nt))=\neg X((q', currenttime))$,且

当$currenttime\le t_1<nt$时, $X((q', t_1))=X((q', currenttime))$.

简单地说,在 q'没有变化的情况下, $next\_turn\_time(q, X)$ 表示大于 *currenttime*,且使 X 函数的输出结果发生翻转的最小时间.如果有多个 RCLT 规则,则它们会生成各自的下一翻转时间,系统则取这些时间点中最小的那个值作为系统的下一个判断 RCLT 规则的时间点.

下面提出一种算法,来计算某些包含 *in_range* 函数的规则的 *next_turn_time* 函数.在提出算法之前,先引进一个对时间区间集合作操作的函数:

$merge\_range: 2^{TR}\to 2^{TR}: merge\_range(TRSet_1)=TRSet_2$,并满足:

$tr\in TRSet_1\Leftrightarrow tr\in TRSet_2$,且当$tr_1, tr_2\in TRSet_2, tr_1\ne tr_2$时,则

$(end(tr_1)<start(tr_2))\vee(end(tr_2)<start(tr_1))$.

函数 *merge_range* 把一个时间区间集合转化成一个作用等价的时间区间集合,且新的区间集合里的元素即时间区间互不重叠,也就是把原来有重叠的时间区间合并成一个新的时间区间.

算法 **1**.

输入:系统当前的状态 $q=(q', STR, currenttime)$;一个 RCLT 规则 X,且 X 的表达式的形式为:

$(...\ in\_range(currenttime, TRset)\ ...), in\_range: T\times 2^{TR}\to\{\text{true},\text{false}\}$, 除了已经标明的 *currenttime* 函数,表达式的其他部分不再包含 *currenttime* 函数.

输出:$next\_turn\_time(q,X)$.

计算过程:

(1) 计算 $TRset_2=merge\_range(TRset)$

(2) 如果 $in\_range(currenttime, TRset)=\text{true}$,计算:

i. $nt=\text{end}(tr)$,其中$tr\in TRset_2$,且$in_range(currenttime, tr)=\text{true}$

转到(4).

(3) 否则,有 $in\_range(currenttime, TRset)=\text{false}$,计算:

$$nt=\min_{tr\in TRSet_2, start(tr)>currenttime} start(tr)$$

(4) nt 即为所求的 X 规则的下一翻转点.

可以看出,算法 1 计算了一种简单形式的 RCLT 规则的 *next_turn_time* 函数,具体的证明从略.由于 X 可以任意定义,要找到一个通用的算法,有相当的难度,在今后的研究中将着重解决这个问题.从以上的讨论可以看出,此函数对解决 RCLT 的处理效率问题是非常重要的,它使 RBAC 系统避免了在每一个时间点都计算一遍所有的 RCLT 规则.

3.4 安全状态恢复

若在某时间点,当计算某条 RCLT 规则时,结果为 false,也就是说,当前的状态不合法即不安全,则要进行处理使系统回到安全的状态.在引入时间特性之前,RCL2000 的规则在进行某个操作之前进行判断,当表达式不被满足时,操作就不被允许.在 RCLT 中,是 *currenttime* 的递增操作引起了 RCLT 的规则判断,即使表达式不被满足,也不能采用不允许此操作进行的方法.RCLT 的方法是中止一些活动会话,设法恢复安全状态.

这里涉及到两个新问题:(1) 怎样选择一个(或多个)活动会话进行终止;(2) 当一个会话终止之后,必然会影

响其他规则的判定结果,导致要选择更多的会话来终止.这两个问题总结起来,就是如何选出一个会话集合(terminating-session set,简称 TSS),进行终止操作,以最终回到安全状态.我们力求找到一个最优的 TSS(optimum TSS,简称 OTSS).

这里所说的最优,是一个并不精确的概念,而且对于不同的安全策略、不同的系统实现以及不同的用户看来,最优的标准也不尽相同,因此只能提出一些基本准则:(1) TSS 包含的会话数量比较小;(2) 会话有可能具有优先级属性,这样,优先级低的会话将会更优先地被放入 TSS 中;(3) 尽量不影响过多的用户.

一个最基本的生成 TSS 的算法,就是对所有可能的会话集合子集,在系统中尝试终止这些会话,然后计算所有规则,若能得到一个合法的状态,则此子集为一个可行的 TSS.在所有得到的可行 TSS 之中,选取一个符合上述准则中最好的,作为最后的 TSS.如果找不到可行的 TSS,即意味着即使把所有的会话都结束也无法得到合法状态,这时就说明约束规则的设定有问题,应重新设计.

这个算法最后总能找到一个理想的 TSS,但是也涉及到时间效率的问题.如果$|S|$很大,这个算法就要进行$2^{|S|}$次全部规则的计算.我们的想法是,如果能找到一个贪心方法,每次挑出一个会话来终止,一直到系统回到合法状态为止.此算法的复杂度将能控制在$|S|$以下,但是如何使算法能得到一个理想的 TSS,则不是一个容易解决的问题.我们将在以后的研究中,力求找到这样的一个理想算法.

4 RCLT 的表现能力

我们使用 RCLT 对第 2 节提出的几个时间约束进行表示,以说明 RCLT 的表现能力.当然,RCLT 的表现能力不止于对已知的时间约束的表示,而对于将来可能出现的新的时间约束的需求的表示,RCLT 同样有很强的潜力.

4.1 激活时间约束

对于用户 u_1 的约束:

局部定义: $ALLOWTR_{u1} \subseteq TR$

表达式: $user(OE(S)) = u_1 \Rightarrow in\_range(currenttime, ALLOWTR_{u1})$

对于角色 r_1 的约束:

局部定义: $ALLOWTR_{r1} \subseteq TR$

表达式: $r_1 \in role\_set(OE(S)) \Rightarrow in\_range(currenttime, ALLOWTR_{r1})$

对于访问许可 p_1 的约束:

局部定义: $ALLOWTR_{p1} \subseteq TR$

表达式: $p_1 \in permission\_set(OE(S)) \Rightarrow in\_range(currenttime, ALLOWTR_{p1})$

4.2 激活时间长度约束

对于用户 u_1 的约束:

局部定义: $ACTIVELEN_{u1} > 0$

表达式: $user(OE(S)) = u_1 \Rightarrow active\_length(OE(S)) \le ACTIVELEN_{u1}$

对于角色 r_1 的约束:

局部定义: $ACTIVELEN_{r1} > 0$

表达式: $r_1 \in role\_set(OE(S)) \Rightarrow active\_length(OE(S)) \le ACTIVELEN_{r1}$

对于访问许可 p_1 的约束:

局部定义: $ACTIVELEN_{p1} > 0$

表达式: $p_1 \in permission\_set(OE(S)) \Rightarrow active\_length(OE(S)) \le ACTIVELEN_{p1}$

4.3 定长时间内激活时间长度约束

对于用户 u_1 的约束:

局部定义: $RANGELEN_{u1} > 0, ACTIVELEN_{u1} > 0$

表达式:

$$\sum_{s\in SSet1} active\_length(s) \quad + \sum_{s\in SSet2} (active\_length(s)\_start(s)) \le ACTIVELEN,$$

其中:

$SSet1 = \{s \mid s \in S \cup OLDS, start(s) + RANGELEN \ge currenttime\}$,

$SSet2 = \{s \mid s \in S \cup OLDS, (start(s) + RANGELEN < currenttime) \wedge (end(s) + RANGELEN \ge currenttime)\}$.

5 结 论

RCLT 在 RCL2000 上作了时间特性的扩展.RCLT 引入的时间特性主要体现在会话的开始时间和结束时间特性上,并增加了一个系统时钟,名为 currenttime 的全局函数.RCLT 的表现能力通过几个例子进行了说明,但其实际的表现能力并不仅限于此.对一个实际的 RCLT 系统来说,时间效率和安全状态的恢复是必须解决的问题,本文提出了一些解决这两个问题的思路,但是,要更好地解决这些问题,还需要进一步的研究.

References:

[1] Sandhu, R. Issues in RBAC . In: Proceedings of the ACM RBAC Workshop. MD: ACM Press, 1996. 21~24.

[2] Jaeger, T. On the increasing importance of constraints. In: Proceedings of 4th ACM Workshop on Role-Based Access Control. Fairefax, VA: ACM Press, 1999. 33~42.

[3] Ahn, G.-J. The RCL2000 language for specifying role-based authorization constraints [Ph.D. Thesis]. Fairfax, VA: George Mason University, 1999.

[4] Sandhu, R., Coyne, E.J., Feinstein, H.L., *et al*. Role-Based access control models. IEEE Computer, 1995,29(2):38~47.

[5] Chen, Fang, Sandhu, R. Constraints for role-based access control. In: Proceedings of the ACM RBAC Workshop. MD: ACM Press, 1996. 39~46.

Role-Based Authorization Constraint with Time Character*

DONG Guang-yu, QING Si-han, LIU Ke-long

(*Engineering and Research Center for Information Security Technology, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: dgy@ercist.iscas.ac.cn; l_kelong@sina.com

http://www.ercist.ac.cn/

Abstract: Authorization constraint is an important part in role-based access control models. There are some methods to specify constraint formally, but they mainly focus on the static character. A formal model is proposed to describe time character of constraint in this paper. A founded constraint model, namely a constraint specification language, is studied in detail, and is extended. The language is called RCLT (role-based constraints language with time-character) after extending. At the same time, the problems of optimizing the efficiency of RCLT system and restoring to the safe state when rules are violated are discussed. RCLT is testified to be able to express the time character of constraint well. However, there are further works to be done for finding practical, efficient and general algorithms for optimizing efficiency and the restoration of safe state, which is the main goal of future research.

Key words: information security; access control, role; constraint; time character

* Received March 28, 2001; accepted August 9, 2001

Supported by the National Natural Science Foundation of China under Grant No.60083007; the National Grand Fundamental Research 973 Program of China under Grant No.G1999035810