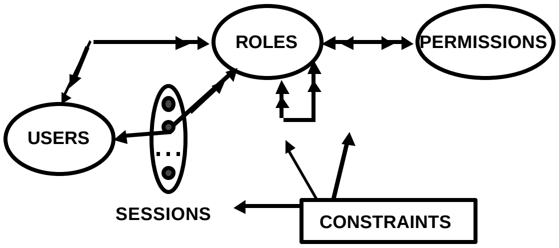
The *RCL2000* Language for Specifying Role-Based Authorization Constraints

Gail-Joon Ahn

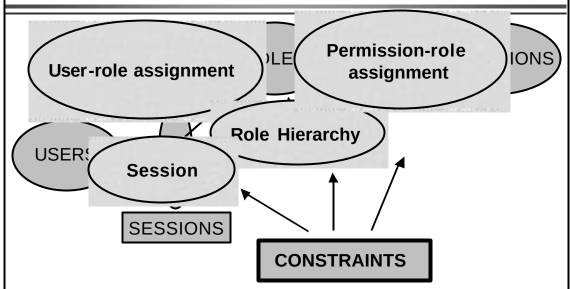
ABSTRACT

- **This presentation includes**
  - **The first formal (and intuitive) language for role-based authorization constraints**
  - **A formal semantics for this language**
  - **Demonstration of the expressive power of the language**
  - **Characterization of role-based constraints into prohibition and obligation constraints**

RBAC96



RBAC96



SEPARATION OF DUTY (1)

- **SOD is fundamental technique for preventing fraud and errors**
- **Related Work**
  - **Enumerate several forms of SOD**
  - **Little work on specifying SOD in a comprehensive way**

SEPARATION OF DUTY (2)

PURCHASING MANAGER **ACCOUNTING PAYABLE MANAGER**

PROHIBITION

- Separation of Duty constraints

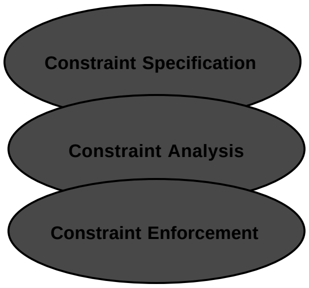
OBLIGATION

- Every faculty member must be assigned to at least one departmental committee

RESEARCH PLAN

- Need to specify these constraints
  - Language
- Show the meaning of expression
  - Formal semantics
- Expressive power of the language
  - Well-known constraints and simulations
- Analysis of the work
  - Characterization

BIG PICTURE



WHO IS THE USER

- Security Researcher
- Security Policy Designer
- Security Architect

RCL 2000

- RCL 2000 (Role-based Constraints Language 2000)
- Specification Language
  - to formally express constraints in role-based systems
- Most components are built upon RBAC96

BASIC ELEMENT (from RBAC96)

- U : a set of users
- R : a set of roles
  - RH Í R ´ R : role hierarchy
- OBJ : a set of objects
- OP : a set of operations
- P = OP ´ OBJ : a set of permissions
- S : a set of sessions

BASIC ELEMENT (from RBAC96)

- UA : a many-to-many user-to-role assignment relation
- PA : a many-to-many permissions-to-role assignment relation

SYSTEM FUNCTIONS (from RBAC96)

- user : R ® 2^U
- roles : U È P È S ® 2^R
- sessions : U ® 2^S
- permissions : R ® 2^P
- operations : R ´ OBJ ® 2^{OP}
- object : P ® 2^{OBJ}

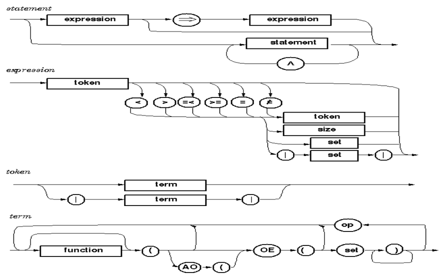
BASIC ELEMENT (beyond RBAC96)

- CR : all conflicting role sets
- CU : all conflicting user sets
- CP : all conflicting permission sets

NON-DETERMINISTIC FUNCTIONS (beyond RBAC96)

- introduced by Chen and Sandhu (1995)
- oneelement (OE)
  - ●oneelement(X) = x_i, where x_i Î X
- allother (AO)
  - ●allother(X) = X - {OE(X)}
    = X - {x_i}
  - should occur along with OE function

SYNTAX



EXAMPLES OF CONSTRAINT EXPRESSION

Conflicting roles cannot have common users

- |roles(OE(U)) Ç OE(CR)| £ 1

Conflicting users cannot have common roles

- roles(OE(OE(CU))) Ç roles(AO(OE(CU))) = f

Users cannot activate two conflicting roles

- |roles(sessions(OE(U))) Ç OE(CR)| £ 1

Users cannot activate two conflicting roles in a single session

- | roles(OE(sessions(OE(U)))) Ç OE(CR)| £ 1

FORMAL SEMANTICS

- **Reduction Algorithm**
  - **to convert a constraint expression to a restricted form of first order predicate logic (RFOPL)**
- **Construction Algorithm**
  - **to construct a constraint expression from RFOPL**

REDUCTION ALGORITHM

OE(OE(CR))Î roles(OE(U)) Þ AO(OE(CR)) Ç roles(OE(U)) = Æ

1. OE(OE(CR))Î roles(OE(U)) Þ (OE(CR) - {OE(OE(CR))}) Ç roles(OE(U)) = Æ

2. " crÎ CR : OE(cr)Î roles(OE(U)) Þ (cr - {OE(cr)}) Ç roles(OE(U)) = Æ

3. " crÎ CR, " rÎ cr : rÎ roles(OE(U)) Þ (cr - {r}) Ç roles(OE(U)) = Æ

4. " crÎ CR, " rÎ cr, " uÎ U : rÎ roles(u) Þ (cr - {r}) Ç roles(u) = Æ

RFOPL STRUCTURE

- **sequence part : predicate**
- **" rÎR, " uÎU : rÎroles(u)**
- **" x_2Î x_1, " x_3Î x_2, " x_4Î x_3 : predicate**

CONSTRUCTION ALGORITHM

" crÎ CR, " rÎ cr, " uÎ U : rÎ roles(u) Þ (cr - {r}) Ç roles(u) = Æ

1. " crÎ CR, " rÎ cr : rÎ roles(OE(U)) Þ (cr - {r}) Ç roles(OE(U)) = Æ

2. " crÎ CR : OE(cr)Î roles(OE(U)) Þ (cr - {OE(cr)}) Ç roles(OE(U)) = Æ

3. OE(OE(CR))Î roles(OE(U)) Þ (OE(CR) - {OE(OE(CR))}) Ç roles(OE(U)) = Æ

4. OE(OE(CR))Î roles(OE(U)) Þ AO(OE(CR)) Ç roles(OE(U)) = Æ

SOUNDNESS AND COMPLETENESS

- **Theorem 1** *Given RCL2000 expression* a , a *can be translated into RFOPL expression* b *. Also* a *can be reconstructed from* b *.*

  C(R(a)) = a

- **Theorem 2** *Given RFOPL expression* b *,* b *can be translated into RCL2000 expression* a *. Also* b´ *which is logically equivalent to* b *can be reconstructed from* a *.*

  R(C(b)) =b´

SEPARATION OF DUTY CONSTRAINTS

- **Specification of SOD constraints identified by Simon and Zurko (1997) and formulated by Virgil et al (1998)**
- **Identify new SOD properties**
  - **Role-centric**
  - **User-centric**
  - **Permission-centric**

ROLE-CENTRIC SOD CONSTRAINT EXPRESSION

- **Static SOD**

  : Conflicting roles cannot have common users

  U = $\{u_1,u_2,\dots u_n\}$, R = $\{r_1,r_2,\dots r_n\}$,

  CR = $\{cr_1,cr_2\}$: $cr_1 = \{r_1,r_2,r_3\}$, $cr_2 = \{r_a,r_b,r_c\}$

  - **|roles(OE(U)) Ç OE(CR)| £1**

PERMISSION-CENTRIC SOD CONSTRAINT EXPRESSION

- **SSOD-CP**
  - **|permissions(roles(OE(U))) Ç OE(CP)| £1**
- **Variations of SSOD-CP**
  - **SSOD-CP Ù**
    |permissions(OE(R)) Ç OE(CP)| £1

USER-CENTRIC SOD CONSTRAINT EXPRESSION

- **SSOD-CU (User-centric)**
  - **SSOD-CR Ù |user(OE(CR)) Ç OE(CU)| £1**

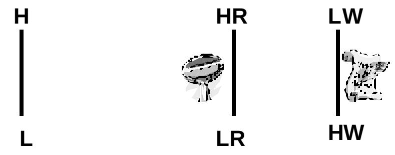
DYNAMIC SOD

- **User-based DSOD**
  - **|roles(sessions(OE(U))) Ç OE(CR)| £1**
- **User-based DSOD with CU**
  - **|roles(sessions(OE(OE(CU)))) Ç OE(CR)| £1**
- **Session-based DSOD**
  - **|roles(OE(sessions(OE(U)))) Ç OE(CR)| £1**
- **Session-based DSOD with CU**
  - **|roles(OE(sessions(OE(OE(CU))))) Ç OE(CR)| £1**

CASE STUDIES

- **Lattice-based access control**
  - **Ravi Sandhu (1993, 1996)**
- **Chinese Wall policy**
  - **Ravi Sandhu (1992)**
- **Discretionary access control**
  - **Sandhu and Munawer (1998)**

LATTICE-BASED ACCESS CONTROL



- Subject *s* can write object *o* only if 𝕝(*s*) £ 𝕝(*o*)
- Subject *s* can read object *o* only if 𝕝(*o*) £ 𝕝(*s*)

Constraints on UA: ***Each user is assigned to exactly two roles xR and LW***

LATTICE-BASED ACCESS CONTROL

- **AR = {ar1, ar2}**
  - **ar1={HR, HW}, ar2={LR, LW}**
- **ASR = {asr1, asr2}**
  - **asr1={HR, LW}, asr2={LR, LW}**

- **Constraint on UA:**
  - **roles(OE(U)) = OE(ASR)**
- **Constraint on sessions:**
  - **roles(OE(sessions(OE(U)))) = OE(AR)**

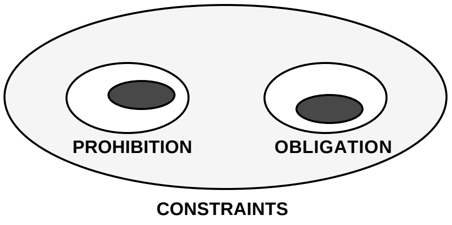
PROHIBITION CONSTRAINTS

- **Forbid the RBAC component from doing (or being) something which is not allowed to do (or be)**
  - **Separation of duty constraints**

OBLIGATION CONSTRAINTS

- **Force the RBAC component to do (or be) something**
  - **LBAC-RBAC, Chinese Wall-RBAC simulation**

CONSTRAINTS CHARACTERIZATION



SIMPLE PROHIBITION CONSTRAINTS

- **Type 1**
  - **½*expr*½£ 1**
- **Type 2**
  - ***expr* = f or ½*expr*½= 0**
- **Type 3**
  - **½*expr1*½<½*expr2*½**

SIMPLE OBLIGATION CONSTRAINTS

- **Type 1**
  - ***expr* ¹ 0 or ½*expr*½> 0**
- **Type 2**
  - **Set X = Set Y**
- **Type 3**
  - **obligation constraints Þ obligation constraints**
- **Type 4**
  - **½*expr* ½= 1**
    - **½*expr*½= 1 Þ ½*expr*½£ 1Ù ½*expr*½> 0**

CONTRIBUTIONS

- **Developed the first formal and intuitive language for role-based authorization constraints**
- **Provided a formal semantics for this language**
- **Demonstrated the expressive power of the language by**
  - **specifying well-known separation of duty constraints**
  - **identifying new role-based SOD constraints**
  - **showing how to specify constraints identified in the simulations of other policies in RBAC**
- **Characterized role-based constraints into prohibition and obligation constraints**

FUTURE WORK

- **Extension of RCL 2000**
  - **Applying it the formalization of some realistic security policies**
- **Implementation Issue**
  - **Tool for checking syntax and semantic as well as visualization of specification**
- **Enforcement of constraints**