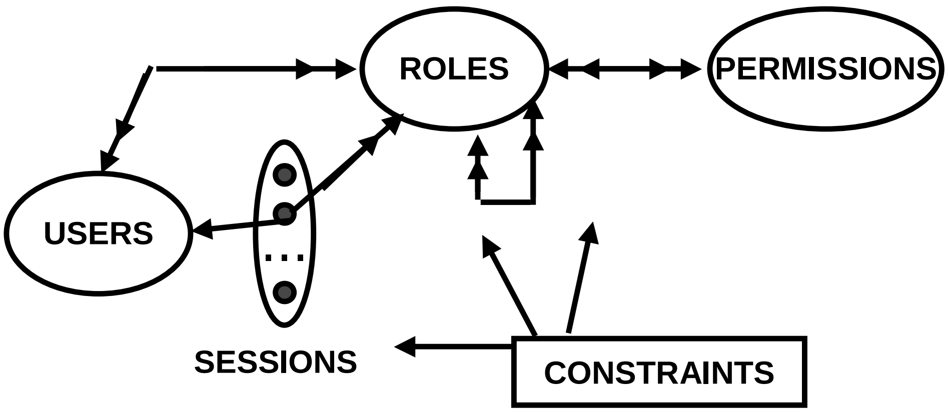
The *RCL2000* Language for Specifying Role-Based Authorization Constraints

Gail-Joon Ahn

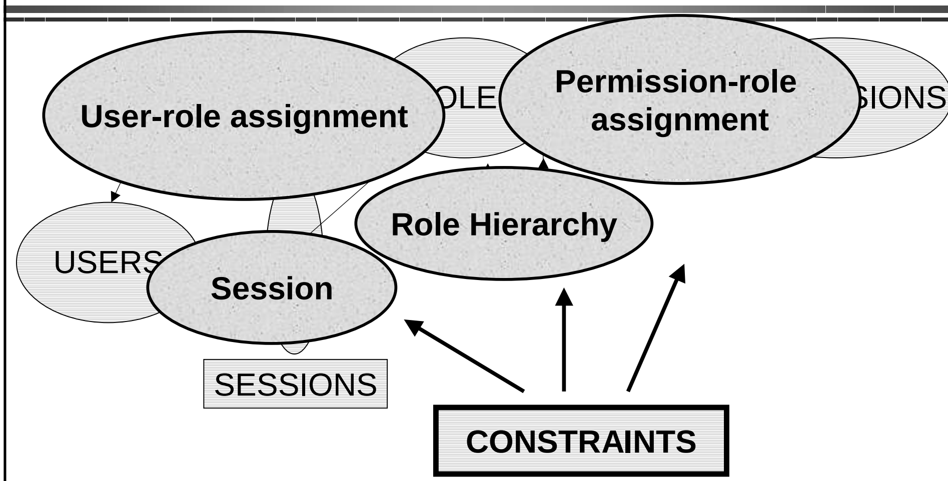
ABSTRACT

- **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

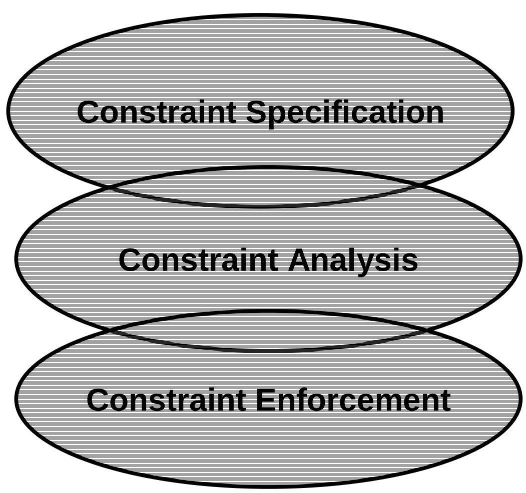
OBLIGATION

- 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
  - $RH \subseteq R \times R$: role hierarchy
- OBJ : a set of objects
- OP : a set of operations
- $P = OP \times OBJ$: 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** : $R \rightarrow 2^{U}$
- **roles** : $U \cup P \cup S \rightarrow 2^{R}$
- **sessions** : $U \rightarrow 2^{S}$
- **permissions** : $R \rightarrow 2^{P}$
- **operations** : $R \times OBJ \rightarrow 2^{OP}$
- **object** : $P \rightarrow 2^{OBJ}$

BASIC ELEMENT (beyond RBAC96)

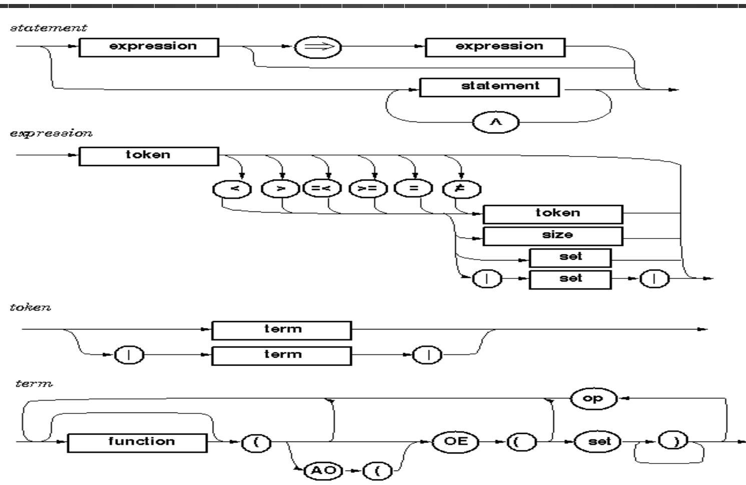
- **CR : all conflicting role sets**
- **CU : all conflicting user sets**
- **CP : all conflicting permission sets**

NON-DETERMINISTIC FUNCTIONS (beyond RBAC96)

- **introduced by Chen and Sandhu (1995)**
- **oneelement (OE)**
  - **oneelement(X) = x_i, where $x_i \in X$**
- **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

- **$|roles(OE(U)) \cap OE(CR)| \leq 1$**

Conflicting users cannot have common roles

- **$roles(OE(OE(CU))) \cap roles(AO(OE(CU))) = \phi$**

Users cannot activate two conflicting roles

- **$|roles(sessions(OE(U))) \cap OE(CR)| \leq 1$**

Users cannot activate two conflicting roles in a single session

- **$| roles(OE(sessions(OE(U)))) \cap OE(CR)| \leq 1$**

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

$OE(OE(CR)) \in roles(OE(U)) \Rightarrow AO(OE(CR)) \cap roles(OE(U)) = \emptyset$

1. $OE(OE(CR)) \in roles(OE(U)) \Rightarrow (OE(CR) - \{OE(OE(CR))\}) \cap roles(OE(U)) = \emptyset$
2. $\forall cr \in CR : OE(cr) \in roles(OE(U)) \Rightarrow (cr - \{OE(cr)\}) \cap roles(OE(U)) = \emptyset$
3. $\forall cr \in CR, \forall r \in cr : r \in roles(OE(U)) \Rightarrow (cr - \{r\}) \cap roles(OE(U)) = \emptyset$
4. $\forall cr \in CR, \forall r \in cr, \forall u \in U : r \in roles(u) \Rightarrow (cr - \{r\}) \cap roles(u) = \emptyset$

RFOPL STRUCTURE

- **sequence part : predicate**
- $\forall r \in R, \forall u \in U : r \in roles(u)$
- $\forall x_2 \in x_1, \forall x_3 \in x_2, \forall x_4 \in x_3$: **predicate**

CONSTRUCTION ALGORITHM

$\forall cr \in CR, \forall r \in cr, \forall u \in U : r \in roles(u) \Rightarrow (cr - \{r\}) \cap roles(u) = \varnothing$

1. $\forall cr \in CR, \forall r \in cr : r \in roles(OE(U)) \Rightarrow (cr - \{r\}) \cap roles(OE(U)) = \varnothing$

2. $\forall cr \in CR : OE(cr) \in roles(OE(U)) \Rightarrow (cr - \{OE(cr)\}) \cap roles(OE(U)) = \varnothing$

3. $OE(OE(CR)) \in roles(OE(U)) \Rightarrow (OE(CR) - \{OE(OE(CR))\}) \cap roles(OE(U)) = \varnothing$

4. $OE(OE(CR)) \in roles(OE(U)) \Rightarrow AO(OE(CR)) \cap roles(OE(U)) = \varnothing$

SOUNDNESS AND COMPLETENESS

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

  $C(R(\alpha)) = \alpha$

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

  $R(C(\beta)) = \beta'$

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

  $U = \{u_1,u_2,\ldots u_n\}$, $R = \{r_1,r_2,\ldots r_n\}$,
  $CR = \{cr_1,cr_2\}$: $cr_1 = \{r_1,r_2,r_3\}$, $cr_2 = \{r_a,r_b,r_c\}$

  - $|roles(OE(U)) \cap OE(CR)| \leq 1$

PERMISSION-CENTRIC SOD CONSTRAINT EXPRESSION

- **SSOD-CP**
  - **$|permissions(roles(OE(U))) \cap OE(CP)| \leq 1$**

- **Variations of SSOD-CP**
  - **SSOD-CP $\wedge$ $|permissions(OE(R)) \cap OE(CP)| \leq 1$**

USER-CENTRIC SOD CONSTRAINT EXPRESSION

- **SSOD-CU (User-centric)**
  - **SSOD-CR $\wedge$ $|user(OE(CR)) \cap OE(CU)| \leq 1$**

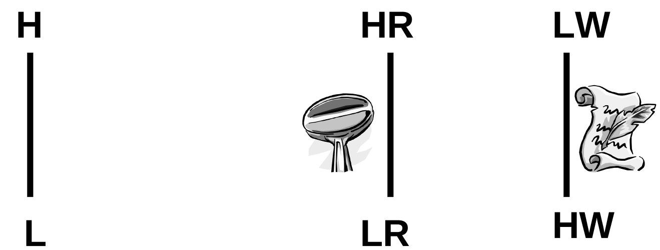
DYNAMIC SOD

- User-based DSOD
  - |roles(sessions(OE(U))) ∩ OE(CR)| ≤1
- User-based DSOD with CU
  - |roles(sessions(OE(OE(CU)))) ∩ OE(CR)| ≤1
- Session-based DSOD
  - |roles(OE(sessions(OE(U)))) ∩ OE(CR)| ≤1
- Session-based DSOD with CU
  - |roles(OE(sessions(OE(OE(CU))))) ∩ 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 $\lambda(s) \leq \lambda(o)$
- Subject *s* can read object *o* only if $\lambda(o) \leq \lambda(s)$

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)**

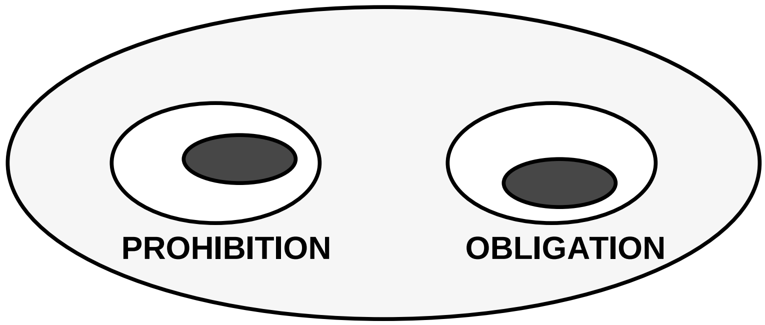
PROHIBITION CONSTRAINTS

- **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**
  - $|\textit{expr}| \leq 1$
- **Type 2**
  - $\textit{expr} = \phi$ **or** $|\textit{expr}| = 0$
- **Type 3**
  - $|\textit{expr1}| < |\textit{expr2}|$

SIMPLE OBLIGATION CONSTRAINTS

- **Type 1**
  - $expr \neq 0$ **or** $|expr| > 0$
- **Type 2**
  - **Set X** $=$ **Set Y**
- **Type 3**
  - **obligation constraints** $\Rightarrow$ **obligation constraints**
- **Type 4**
  - $|expr| = 1$
    - $|expr| = 1 \equiv |expr| \leq 1 \wedge |expr| > 0$

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**