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

User-role assignment
Permission-role assignment
Role Hierarchy
Session

CONSTRAINTS
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

- Constraint Specification
- Constraint Analysis
- Constraint Enforcement
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, roles* : $U \cup P \cup S \rightarrow 2^R$
- sessions : $U \rightarrow 2^S$
- permissions, 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
BASIC ELEMENT (beyond RBAC96)

- CR1: all conflicting role sets
- CR2: all conflicting role sets
- CR3: all conflicting role sets
- ..... 

NON-DETERMINISTIC FUNCTIONS (beyond RBAC96)

- introduced by Chen and Sandhu (1995)
- oneelement (OE)
  - oneelement(X) = x_i, where x_i ∈ 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**
  - $|\text{roles(OE(U))} \cap \text{OE(CR)}| \leq 1$

- **Conflicting users cannot have common roles**
  - $\text{roles(OE(OE(CU)))} \cap \text{roles(AO(OE(CU)))} = \emptyset$

- **Users cannot activate two conflicting roles**
  - $|\text{roles(sessions(OE(U)))} \cap \text{OE(CR)}| \leq 1$

- **Users cannot activate two conflicting roles in a single session**
  - $|\text{roles(sessions(OE(U)))} \cap \text{OE(CR)}| \leq 1$
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

\[ \text{OE(OE(CR))} \in \text{roles(OE(U))} \Rightarrow \text{AO(OE(CR))} \cap \text{roles(OE(U))} = \emptyset \]

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

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

CONSTRUCTION ALGORITHM

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

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

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

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

4. \( OE(OE(CR)) \in \text{roles}(OE(U)) \Rightarrow AO(OE(CR)) \cap \text{roles}(OE(U)) = \emptyset \)
SOUNDNESS AND COMPLETENESS

- **Theorem 1** Given RCL2000 expression $\alpha$, $\alpha$ can be translated into RFOPL expression $\beta$. Also $\alpha$ can be reconstructed from $\beta$.
  \[
  C(R(\alpha)) = \alpha
  \]

- **Theorem 2** Given RFOPL expression $\beta$, $\beta$ can be translated into RCL2000 expression $\alpha$. Also $\beta^\prime$ which is logically equivalent to $\beta$ can be reconstructed from $\alpha$.
  \[
  R(C(\beta)) = \beta^\prime
  \]

SEPARATION OF DUTY CONSTRAINTS

- Identify new SOD properties
  - Role-centric
  - User-centric
  - Permission-centric
ROLE-CENTRIC SOD
CONSTRAINT EXPRESSION

- **Static SOD**
  - Conflicting roles cannot have common users
    
    \[ U = \{u_1, u_2, \ldots, u_n\}, \quad R = \{r_1, r_2, \ldots, r_n\}, \]
    
    \[ CR = \{cr_1, cr_2\} : \quad cr_1 = \{r_1, r_2, r_3\}, \quad cr_2 = \{r_a, r_b, r_c\} \]
    
    \[ |\text{roles}(\text{OE}(U)) \cap \text{OE}(CR)| \leq 1 \]

PERMISSION-CENTRIC SOD
CONSTRAINT EXPRESSION

- **SSOD-CP**
  - \[ |\text{permissions}(\text{roles}(\text{OE}(U))) \cap \text{OE}(CP)| \leq 1 \]

- **Variations of SSOD-CP**
  - \[ \text{SSOD-CP} \land |\text{permissions}(\text{OE}(R)) \cap \text{OE}(CP)| \leq 1 \]
USER-CENTRIC SOD
CONSTRAINT EXPRESSION

- SSOD-CU (User-centric)
  - SSOD-CR \( \land |\text{user(OE(CR))} \cap \text{OE(CU)}| \leq 1 \)

DYNAMIC SOD

- User-based DSOD
  - \( |\text{roles(sessions(OE(U)))} \cap \text{OE(CR)}| \leq 1 \)

- User-based DSOD with CU
  - \( |\text{roles(sessions(OE(OE(CU))))} \cap \text{OE(CR)}| \leq 1 \)

- Session-based DSOD
  - \( |\text{roles(OE(sessions(OE(U))))} \cap \text{OE(CR)}| \leq 1 \)

- Session-based DSOD with CU
  - \( |\text{roles(OE(sessions(OE(OE(CU)))))} \cap \text{OE(CR)}| \leq 1 \)
CASE STUDIES

- **Lattice-based access control**
- **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

- $AR = \{ar_1, ar_2\}$
  - $ar_1=\{HR, HW\}$, $ar_2=\{LR, LW\}$
- $ASR = \{asr_1, asr_2\}$
  - $asr_1=\{HR, LW\}$, $asr_2=\{LR, LW\}$

- Constraint on UA:
  - $\text{roles}(OE(U)) = OE(ASR)$
- Constraint on sessions:
  - $\text{roles}(OE(\text{sessions}(OE(U)))) = OE(AR)$

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

PROHIBITION

OBLIGATION

CONSTRAINTS
SIMPLE PROHIBITION
CONSTRAINTS

- Type 1
  - $|expr| \leq 1$

- Type 2
  - $expr = \phi$ or $|expr| = 0$

- Type 3
  - $|expr_1| < |expr_2|$

SIMPLE OBLIGATION
CONSTRAINTS

- Type 1
  - $expr \neq 0$ or $|expr| > 0$

- Type 2
  - Set $X = \text{Set Y}$

- Type 3
  - obligation constraints $\Rightarrow$ obligation constraints

- Type 4
  - $|expr| = 1$
    - $|expr| = 1 = |expr| \leq 1 \land |expr| > 0$
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