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

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

BIG PICTURE

- Constraint Specification
- Constraint Analysis
- Constraint Enforcement

OBLIGATION

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

WHO IS THE USER

- Security Researcher
- Security Policy Designer
- Security Architect

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

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
- \( \mathcal{RH} \subseteq R \times R \) : role hierarchy
- \( \text{OBJ} \) : a set of objects
- \( \text{OP} \) : a set of operations
- \( P = \text{OP} \times \text{OBJ} \) : a set of permissions
- \( S \) : a set of sessions

BASIC ELEMENT
(beyond RBAC96)

- \( \text{CR} \) : all conflicting role sets
- \( \text{CU} \) : all conflicting user sets
- \( \text{CP} \) : all conflicting permission sets

BASIC ELEMENT
(from RBAC96)

- \( \text{UA} \) : a many-to-many user-to-role assignment relation
- \( \text{PA} \) : a many-to-many permissions-to-role assignment relation

BASIC ELEMENT
(beyond RBAC96)

- \( \text{CR}_1 \) : all conflicting role sets
- \( \text{CR}_2 \) : all conflicting role sets
- \( \text{CR}_3 \) : all conflicting role sets
- ....

SYSTEM FUNCTIONS
(from RBAC96)

- \( \text{user} \) : \( R \rightarrow 2^U \)
- \( \text{roles, roles}^* \) : \( U \cup P \cup S \rightarrow 2^R \)
- \( \text{sessions} \) : \( U \rightarrow 2^S \)
- \( \text{permissions, permissions}^* \) : \( R \rightarrow 2^P \)
- \( \text{operations} \) : \( R \times \text{OBJ} \rightarrow 2^{OP} \)
- \( \text{object} \) : \( P \rightarrow 2^{OBJ} \)

NON-DETERMINISTIC
FUNCTIONS (beyond RBAC96)

- introduced by Chen and Sandhu (1995)
- oneelement (OE)
  - oneelement(\( X \)) = \( x \) where \( x \in X \)
- allother (AO)
  - allother(\( X \)) = \( X - \{ \text{OE}(X) \} \)
    = \( X - \{ x \} \)
  - should occur along with OE function
EXAMPLES OF CONSTRAINT EXPRESSION

Conflicting roles cannot have common users

\[ \{ \text{roles}(OE(U)) \cap \text{OE}(CR) \} \subseteq 1 \]

Conflicting users cannot have common roles

\[ \{ \text{roles}(OE(U)) \cap \text{roles}(AO(OE(CR))) \} \subseteq 0 \]

Users cannot activate two conflicting roles

\[ \{ \text{roles}(OE(U)) \cap \text{OE}(CR) \} \subseteq 1 \]

Users cannot activate two conflicting roles in a single session

\[ \{ \text{roles}(OE(U)) \cap \text{OE}(CR) \} \subseteq 1 \]

FORMAL SEMANTICS

\[ \text{Reduction Algorithm} \]

- to convert a constraint expression to a restricted form of first order predicate logic (RFOPL)

\[ \text{Construction Algorithm} \]

- to construct a constraint expression from RFOPL

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

\[ \{ \text{roles}(U) \cap \text{roles}(CR) \} \subseteq 1 \]

1. \( \forall cr \in CR, \forall r \in \text{roles}(U) : (r \cap \{r\}) \cap \text{roles}(OE(U)) \subseteq 0 \)

2. \( \forall cr \in CR : \text{roles}(OE(U)) \Rightarrow (cr \cap \text{roles}(CR)) \cap \text{roles}(OE(U)) \subseteq 0 \)

3. \( \forall cr \in CR, \forall r \in \text{roles}(OE(U)) : (r \cap \text{roles}(CR)) \cap \text{roles}(OE(U)) \subseteq 0 \)

4. \( \forall cr \in CR : \text{roles}(OE(U)) \Rightarrow \text{AO}(OE(CR)) \cap \text{roles}(OE(U)) \subseteq 0 \)
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 $a$.
  \[ R(C(b)) = b' \]

PERMISSION-CENTRIC SOD CONSTRAINT EXPRESSION

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

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

SEPARATION OF DUTY CONSTRAINTS

- Identify new SOD properties
  - Role-centric
  - User-centric
  - Permission-centric

USER-CENTRIC SOD CONSTRAINT EXPRESSION

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

ROLE-CENTRIC SOD CONSTRAINT EXPRESSION

- **Static SOD**
  - Conflicting roles cannot have common users
    \[ U = \{u_1, u_2, ..., u_n\}, \ R = \{r_1, r_2, ..., r_n\}, \ CR = \{c_{r_1}, c_{r_2}\} : c_{r_1} = \{r_1, r_2, r_3\}, \ c_{r_2} = \{r_4, r_5, r_6\} \]
  - |roles(\text{OE}(U)) \cap \text{OE}(\text{CR})| \leq 1

DYNAMIC SOD

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

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

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

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

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

PROHIBITION CONSTRAINTS

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

LATTICE-BASED ACCESS CONTROL

Subject can write object o only if $\lambda(s) \leq \lambda(o)$
Subject can read object o only if $\lambda(o) \leq \lambda(s)$

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

OBLIGATION CONSTRAINTS

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

CONSTRAINTS CHARACTERIZATION

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)
SIMPLE PROHIBITION CONSTRAINTS

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

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**
  - $|expr| = 1$
    - $|expr| = 1 \Rightarrow |expr| \leq 1 \land |expr| > 0$

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**