The \textit{RCL2000} Language for Specifying Role-Based Authorization Constraints

Gail-Joon Ahn

\textbf{ABSTRACT}

\begin{itemize}
  \item This presentation includes
    \begin{itemize}
      \item The first formal (and intuitive) language for role-based authorization constraints
      \item A formal semantics for this language
      \item Demonstration of the expressive power of the language
      \item Characterization of role-based constraints into prohibition and obligation constraints
    \end{itemize}
\end{itemize}
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

© Gail J. Ahn 2000

SEPARATION OF DUTY (2)

PURCHASING MANAGER

ACCOUNTING PAYABLE MANAGER

© Gail J. Ahn 2000
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 ⊆ 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

© Gail J. Ahn 2000

BASIC ELEMENT
(from RBAC96)

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

© Gail J. Ahn 2000
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 ∈ 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
● \(|\text{roles}(\text{OE}(U)) \cap \text{OE}(\text{CR})| \leq 1\)

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

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

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

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

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

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

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

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

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

RFOPL STRUCTURE

- \text{sequence part} : \text{predicate}
- \( \forall \text{r} \in \text{R}, \forall \text{u} \in \text{U} : \text{r} \in \text{roles}(\text{u}) \)
- \( \forall \text{x}_2 \in \text{x}_1, \forall \text{x}_3 \in \text{x}_2, \forall \text{x}_4 \in \text{x}_3 : \text{predicate} \)
CONSTRUCTION ALGORITHM

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

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

2. \[ \forall c r \in CR : OE(cr) \in \text{roles}(OE(U)) \implies (cr - \{OE(cr)\}) \cap \text{roles}(OE(U)) = \emptyset \]

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

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

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 \( \beta' \) which is logically equivalent to \( \beta \) can be reconstructed from \( \alpha \).

\[ R(C(\beta)) = \beta' \]
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\}, 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\}
  - |\text{roles(OE(U))} \cap \text{OE(CR)}| \leq 1
PERMISSION-CENTRIC SOD CONSTRAINT EXPRESSION

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

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

USER-CENTRIC SOD CONSTRAINT EXPRESSION

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

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

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

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

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

CASE STUDIES

- **Lattice-based access control**

- **Chinese Wall policy**
  - Ravi Sandhu (1992)

- **Discretionary access control**
  - Sandhu and Munawer (1998)
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 $LW$

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

© Gail J. Ahn 2000
CONSTRAINTS
CHARACTERIZATION

SIMPLE PROHIBITION
CONSTRAINTS

◆ Type 1
  ● |expr| ≤ 1

◆ Type 2
  ● expr = φ or |expr| = 0

◆ Type 3
  ● |expr1| < |expr2|
SIMPLE OBLIGATION CONSTRAINTS

- Type 1
  - \( \text{expr} \neq 0 \) or \(|\text{expr}| > 0\)
- Type 2
  - Set \( X = Y \)
- Type 3
  - obligation constraints \( \Rightarrow \) obligation constraints
- Type 4
  - \(|\text{expr}| = 1\)
    - \(|\text{expr}| = 1 \equiv |\text{expr}| \leq 1 \land |\text{expr}| > 0\)

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