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

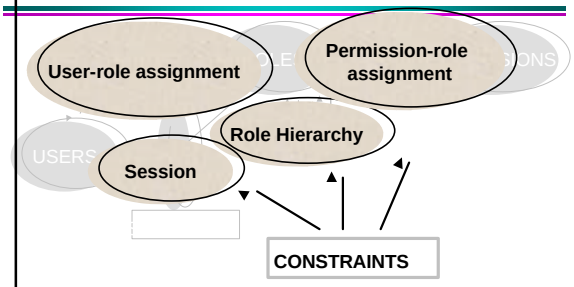
Gail-Joon Ahn

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

© Gail J. Ahn

RBAC96



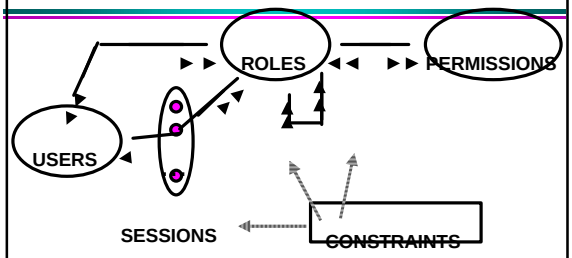
© Gail J. Ahn

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

RBAC96



© Gail J. Ahn

SEPARATION OF DUTY (2)

PURCHASING MANAGER

ACCOUNTING PAYABLE MANAGER

© Gail J. Ahn

PROHIBITION

- Separation of Duty constraints

© Gail J. Ahn

OBLIGATION

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

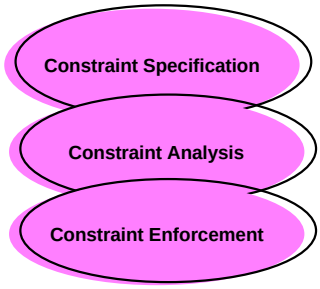
© Gail J. Ahn

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

© Gail J. Ahn

BIG PICTURE



© Gail J. Ahn

WHO IS THE USER

- Security Researcher
- Security Policy Designer
- Security Architect

© Gail J. Ahn

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

© Gail J. Ahn

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

© Gail J. Ahn

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

SYSTEM FUNCTIONS (from RBAC96)

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

© Gail J. Ahn

BASIC ELEMENT (beyond RBAC96)

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

© Gail J. Ahn

BASIC ELEMENT (beyond RBAC96)

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

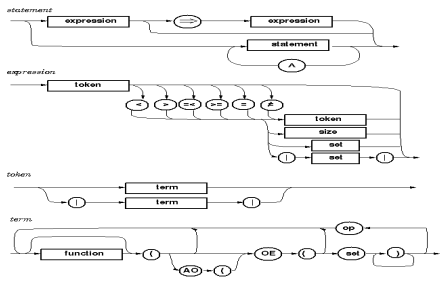
© Gail J. Ahn

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

© Gail J. Ahn

SYNTAX



© Gail J. Ahn

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**

© Gail J. Ahn

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**

© Gail J. Ahn

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

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

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

© Gail J. Ahn

RFOPL STRUCTURE

- **sequence part : predicate**
- **" r Î R, " u Î U : r Î roles(u)**
- **" x_2 Î x_1, " x_3 Î x_2, " x_4 Î x_3 : predicate**

© Gail J. Ahn

CONSTRUCTION ALGORITHM

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

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

2. " cr Î 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

© Gail J. Ahn

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´

© Gail J. Ahn

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**

© Gail J. Ahn

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)) Ç OE(CR)| £1**

© Gail J. Ahn

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**

© Gail J. Ahn

USER-CENTRIC SOD CONSTRAINT EXPRESSION

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

© Gail J. Ahn

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**

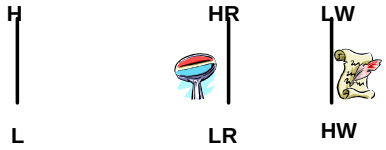
© Gail J. Ahn

CASE STUDIES

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

© Gail J. Ahn

LATTICE-BASED ACCESS CONTROL



- Subject *s* can write object *o* only if $l(s) \leq l(o)$
- Subject *s* can read object *o* only if $l(o) \leq l(s)$

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

© Gail J. Ahn

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)

© Gail J. Ahn

PROHIBITION CONSTRAINTS

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

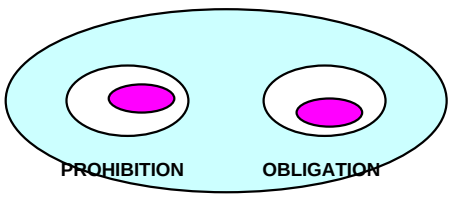
© Gail J. Ahn

OBLIGATION CONSTRAINTS

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

© Gail J. Ahn

CONSTRAINTS CHARACTERIZATION



CONSTRAINTS

© Gail J. Ahn

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

© Gail J. Ahn

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** ° ½*expr* ½£ **1** Ù ½*expr*½> **0**

© Gail J. Ahn

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**

© Gail J. Ahn

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**

© Gail J. Ahn