CS590U
Access Control: Theory and Practice

Lecture 12 (February 23)
Role Based Access Control
Role-Based Access Control Models.

RBAC96 Family of Models

- RBAC0: BASIC RBAC
  - RBAC1: Role Hierarchies
  - RBAC2: Constraints
  - RBAC3: Role Hierarchies + Constraints
RBAC0

User-Role Assignment

Permission-Role Assignment

Users

* *

Roles

Permissions

Sessions

1

* *

* *

* *

4
RBAC0: Formal Model

- U, R, P, S (users, roles, permissions, and sessions)
- PA ⊆ P × R (permission assignment)
- UA ⊆ U × R (user assignment)
- user: S → U
- roles: S → 2^R
  - requires roles(s) ⊆ { r | (user(s), r) ∈ UA }

Session s has permissions

\[ \bigcup_{r \in \text{roles}(s)} \{ p \mid (p, r) \in \text{PA} \} \]
Why RBAC

- Fewer relationships to manage
  - from $O(mn)$ to $O(m+n)$, where $m$ is the number of users and $n$ is the number of permissions
- Roles add a useful level of indirection
RBAC1: RBAC0+ Role Hierarchies

Primary-Care Physician

Specialist Physician

Physician

Health-Care Provider
RBAC1: Formal Model

- U, R, R, S, PA, UA, and user unchanged from RBAC0
- RH ⊆ R × R: a partial order on R, written as ≥
- roles: S → 2^R
  - requires roles(s) ⊆ 
    \{ r | \exists r' [(r' ≥ r) & (user(s), r') \in PA] \}

Session s has permissions

\[ \bigcup_{r \in \text{roles}(s)} \{ p | \exists r'' [(r \geq r'') & (p, r'') \in PA] \} \]
On Modeling Role Hierarchy As A Partial Order

- Modeling RH as a partial order may miss some important information.
- Consider the two examples to the right:
  - where the dashed edge is added and removed.
- Better approach seems to remember the base edges and then compute their transitive and reflexive closure.

EX1:

EX2:
Semantics of Role Hierarchies

- User inheritance
  - \( r_1 \supseteq r_2 \) means every user that is a member of \( r_1 \) is also a member of \( r_2 \)

- Permission inheritance
  - \( r_1 \supseteq r_2 \) means every permission that is authorized for \( r_2 \) is also authorized for \( r_1 \)

- Activation inheritance
  - \( r_1 \supseteq r_2 \) means that activating \( r_1 \) will also activate \( r_2 \)
RBAC2: RBAC0 + Constraints

- No formal model specified
- A list of examples are given
Static Mutual Exclusion
Constraints

- Two mutually exclusive roles: cannot both have the same user as members
- Two mutually exclusive roles: cannot both have the same permissions
  - why?
- Two mutually exclusive permissions: one role cannot have both permissions
  - why?
Cardinality Constraints

- On User-Role Assignment
  - at most k users can belong to the role
  - at least k users must belong to the role
  - exactly k users must belong to the role

- On activation
  - at most k users can activate a role
  - ...

Why Using Constraints?

- For laying out higher level organization policy
  - simply a convenience when admin is centralized
  - a tool to enforce high-level policies when admin is decentralized
RBAC3

- RBAC0 + Role Hierarchies + Constraints
Some Issues in RBAC
Whether to Allow Multiple Roles to be Activated?

- RBAC96 allows this
- [Baldwin’90] does not

Observations:
- one can define new role to achieve the effect of activating multiple roles
- dynamic constraints are implicit when only one role can be activated in a session
What is a Role?

- A set of users
- A set of permissions (named protection domains)
- A set of users and permissions
- Also affects how to interpret role hierarchies
- Maybe it is useful to have both roles and groups?
Roles vs. Groups

- What are the differences?
  - Answer 1: groups are sets of users, and roles are sets of users as well as permissions
    - doesn’t seem to be true.
  - Answer 2: one can activate and deactivate roles, but cannot deactivate groups
    - seems unimportant unless there is negative authorization
  - Answer 3: one can enumerate permissions that a role has
    - seems an implementation issue
Everything as an attribute?

- Some attributes are more intrinsic about properties of a user
- Some attributes are more intrinsic about job functionalities
The NIST Standard

- ANSI Standard
Overview of the NIST Standard for RBAC

Hierarchical RBAC

Static Separation of Duties

Dynamic Separation of Duties

Core RBAC
Core RBAC (1)

- USERS
- ROLES
- OBS
- OPS
- PRMS = $2^{(OPS \times OBS)}$
  - Op : (p: PRMS) $\rightarrow 2^{OPS}$
  - Ob : (p: PRMS) $\rightarrow 2^{OBS}$
Core RBAC (2)

- $\text{UA} \subseteq \text{USERS} \times \text{ROLES}$
  - $\text{assigned\_users} : (r : \text{Roles}) \rightarrow 2^{\text{USERS}}$

- $\text{PA} \subseteq \text{PRMS} \times \text{ROLES}$
  - $\text{assigned\_permissions} : (r : \text{Roles}) \rightarrow 2^{\text{PRMS}}$
Core RBAC (3)

- **SESSIONS**
- session_users : (s : SESSIONS) → USERS
  - user_sessions : (u : USERS) → 2^SESSIONS
- session_roles : (s : SESSIONS) → 2^ROLES
  - avail_session_perms : 
    (s : SESSIONS) → 2^{PRMS}
Hierarchical RBAC: Generalized Role Hierarchies

- $RH \subseteq \text{ROLES} \times \text{ROLES}$
  - user inheritance & permission inheritance
  - we say $r_1$ inherits $r_2$ if $r_1 \geq r_2$
- authorized_users : (r :Roles) $\rightarrow$ $2^{\text{USERS}}$
- authorized_permissions : (r :Roles) $\rightarrow$ $2^{\text{PRMS}}$
Hierarchical RBAC: Limited Role Hierarchies

- Role Hierarchies with the limitation that each role has at most one immediate senior
  - Role hierarchies form a forest
Constrained RBAC: Motivations

- Example of SoD
  - The following duties shall be performed by different individuals:
    1. Check request reviewer
    2. Check preparer
    3. Check issuer
    4. Check deliverer
    5. Ledger reviewer
Constrained RBAC: Static SoD

- \( SSD \subseteq (2^{\text{ROLES}} \times N) \) is a collection of pairs \((rs, n)\)
  - \( rs: \) a role set
  - \( n: \) \( n \geq 2 \) is a natural number
- For each \((rs, n)\), no user is authorized for \( n \) or more roles in \( rs \)
SoD with Role Hierarchies

- Two roles can be mutually exclusive only if neither one inherits the other.
- If two roles are mutually exclusive, no role can inherit from both.
- If two roles are mutually exclusive, there can be no “root” or “super user”.
Constrained RBAC: Dynamic SoD

- DSD $\subseteq (2^{\text{ROLES}} \times N)$ is a collection of pairs (rs, n)
  - rs: a role set
  - n: $n \geq 2$ is a natural number
- For each (rs, n), no user is allowed to activate n or more roles in rs in one session
Functional Specifications

- Administrative functions
- Supporting system functions
- Review functions
Next Lecture

- SoD policies and Constraints