Information Security
CS 526

Topic 22: Role and Attribute Based Access Control
Readings for This Lecture

- RBAC96 Family
Background: Role Based Access Control

- Non-role-based systems

- Role-Based Access Control Systems (RBAC)

Users:
- Alice
- Bob
- Carl
- Dave
- Eva

Permissions:
- DB2 Account
- WebSphere Account
- Windows Account
- Linux Account

Roles:
- DB Admin
- Web Admin
- Software Developer

Permissions:
- DB2 Account
- WebSphere Account
- Windows Account
- Linux Account
ROLE-BASED ACCESS CONTROL (RBAC)

• Motivating Problem: how to administer user-permission relation
  – Different from DAC and MAC, which deal with processes in operating systems

• Roles as a level of indirection
  – Butler Lampson or David Wheeler: "all problems in Computer Science can be solved by another level of indirection"

• RBAC is multi-faceted and open ended
  – Extensions: ARBAC (administrative), CBRAC (constraint), dRBAC (dynamic), ERBAC (enterprise), fRBAC (flexible), GRBAC (generalized), HRBAC (hierarchical), IRBAC (interoperability), JRBAC (Java), LRBAC (Location), MRBAC (Management), PRBAC (privacy), QRBAC (QoS), RRBAC(Rule), SRBAC(Spatial), TRBAC (temporal), V, W, x.
Why Roles?

- Fewer relationships to manage
  - possibly 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 abstraction
- Organizations operate based on roles
- A role may be more stable than
  - the collection of users and the collection of permissions that are associated with it
Groups vs. Roles

• Depending on the precise definition, can be the same or different.
• Some differences that may or may not be important, depending on the situation
  – Answer 1: sets of users vs. sets of users as well as permissions
  – Answer 2: roles can be activated and deactivated, groups cannot
    • Groups can be used to prevent access with negative authorization.
    • Roles can be deactivated for least privilege
  – Answer 3: can easily enumerate permissions that a role has, but not for groups
RBAC96 FAMILY OF MODELS (Sandhu et al.)

RBAC3
ROLE HIERARCHIES + CONSTRAINTS

RBAC1
ROLE HIERARCHIES

RBAC0
BASIC RBAC

RBAC2
CONSTRAINTS
RBAC0
PERMISSIONS

- Left abstract in the RBAC96 model
- Permissions are positive
- No negative permissions or denials
  - RBAC defines a closed policy, i.e., all accesses are denied unless they are explicitly authorized
- No duties or obligations
  - Example obligation: can access patient document, but must notify patient, or must delete after 30 days
RBAC0: Formal Model

- Vocabulary: U, R, P, S (users, roles, permissions, and sessions)
- Static relations:
  - PA ⊆ P × R (permission assignment)
  - UA ⊆ U × R (user assignment)
- Dynamic relations:
  - user: S → U each session has one user
  - roles: S → 2^R and some activated roles
    - requires roles(s) ⊆ \{ r | (user(s), r) ∈ UA \}

Session s has permissions

\[ ∪_{r ∈ \text{roles}(s)} \{ p | (p, r) ∈ PA \} \]
RBAC1

ROLE HIERARCHIES

USER-ROLE ASSIGNMENT

PERMISSION-ROLE ASSIGNMENT

SESSIONS

 USERS

 ROLES

 PERMISSIONS
HIERARCHICAL ROLES (ex 1)

Primary-Care Physician

Physician

Specialist Physician

Health-Care Provider
HIERARCHICAL ROLES (ex 2)

- Supervising Engineer
  - Hardware Engineer
  - Software Engineer
Semantics of Role Hierarchies

- User inheritance
  - $r_1 \geq r_2$ means every user that is a member of $r_1$ is also a member of $r_2$

- Permission inheritance
  - $r_1 \geq r_2$ means every permission that is authorized for $r_2$ is also authorized for $r_1$

- Activation inheritance
  - $r_1 \geq r_2$ means that activating $r_1$ will also activate $r_2$

Permission and Activation inheritance have different effect when there are constraints about activation.
RBAC1: Formal Model

- \( U, R, P, S, PA, UA \), and user unchanged from RBAC0
- \( RH \subseteq R \times R \) : a partial order on \( R \), written as \( \geq \)
  - When \( r_1 \geq r_2 \), we say \( r_1 \) is a senior than \( r_1 \), and \( r_2 \) is a junior than \( r_1 \)
- roles: \( S \rightarrow 2^R \)
  - requires roles(s) \( \subseteq \)
    \[ \{ r | \exists r' \ [(r' \geq r) \& (\text{user}(s), r') \in UA] \} \]

Session s includes permissions

\[ \bigcup_{r \in \text{roles}(s)} \{ p | \exists r'' \ [(r \geq r'') \& (p, r'') \in PA] \} \]
RBAC2: RBAC0 + Constraints

• No formal model specified
• Example constraints
  – Mutual exclusion
  – Pre-condition: Must satisfy some condition to be member of some role
    • E.g., a user must be an undergrad student before being assigned the UTA role
  – Cardinality
Mutual Exclusion Constraints

- Mutually Exclusive Roles
  - Static Exclusion: No user can hold both roles
    - often referred to as Static Separation of Duty constraints
    - Preventing a single user from having too much permissions
  - Dynamic Exclusion: No user can activate both roles in one session
    - Often referred to as Dynamic Separation of Duty constraints
    - Interact with role hierarchy interpretation
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
  – Only a tool for convenience and error checking when admin is centralized
    • Not absolutely necessary if admin is always vigilant, as admin can check all organization policies are met when making any changes to RBAC policies
  – A tool to enforce high-level policies when admin is decentralized
RBAC3

ROLE HIERARCHIES

USER-ROLE ASSIGNMENT

PERMISSIONS-ROLE ASSIGNMENT

SESSIONS

CONSTRAINTS
Products Using RBAC

• Data Base Management Systems (DBMS)

• Enterprise Security Management
  – IBM Tivoli Identity Manager (central administration and provisioning of accounts, resources, etc)

• Many operating systems claim to use roles
  – Though only in very limited way
The NIST Standard


- American National Standards Institute Standard, 2004
  - Has a number of flaws, including with typos, errors in math definitions, and others high-level design choices
Overview of the NIST Standard for RBAC

- Hierarchical RBAC
- Core RBAC
- Static Separation of Duties
- Dynamic Separation of Duties
Research Challenges in RBAC

• Role engineering
  – Design roles for an access control scenario.
  – Top down approach: start from analyzing business requirement.
  – Bottom up approach: Role Mining: mine existing access control data for roles

• Effective administration of RBAC systems
  – Especially help ensure updates still lead to useful states

• Effective usage of constraints
Attribute-Based Access Control

- An access control method where subject requests to perform operations on objects are granted or denied based on
  - assigned attributes of the subject,
    - E.g., job role, clearance, division/unit, location
  - assigned attributes of the object,
    - E.g.,
  - environment conditions,
    - E.g., time, state of emergency
  - and a set of policies that are specified in terms of those attributes and conditions.
    - E.g., a list of rules, as in firewall policies,
Coming Attractions …

• Network Security