NC STATE UNIVERSITY Computer Science

CSC 474/574
Information Systems Security

Topic 4.4

Role-Based Access Control (RBAC)

OUTLINE

- Role-based Access Control
  - Motivation
  - Features
  - Models
  - Issues

OWNER-BASED DAC

- owner has all-or-nothing power
  - superuser fallacy
- spaghetti of intent
- negative permissions make for messier spaghetti
- Trojan horses can subvert intent

MAC/Lattice-Based AC/BLP

- enforce one-directional information flow in a lattice of security labels
- can be used for
  - confidentiality
  - integrity
  - aggregation (Chinese Wall)
  - combinations of these

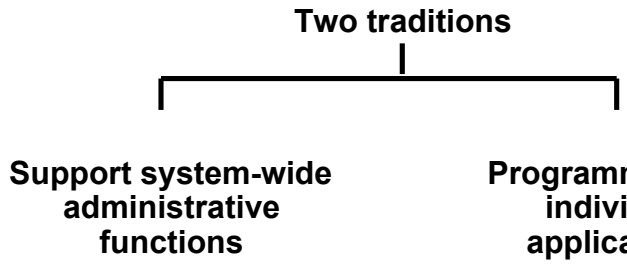
RBAC

- A user's permissions are determined by the user's roles
  - rather than identity (DAC) or clearance (MAC)
  - roles can encode arbitrary attributes
- Facilitates
  - administration of permissions
  - articulation of policy
- ranges from very simple to very sophisticated

RBAC

- Policy neutral
- Policy oriented
  - least privilege
  - separation of duties
  - encapsulation of primitive permissions
  - Roles are a semantic construct around which to build policy

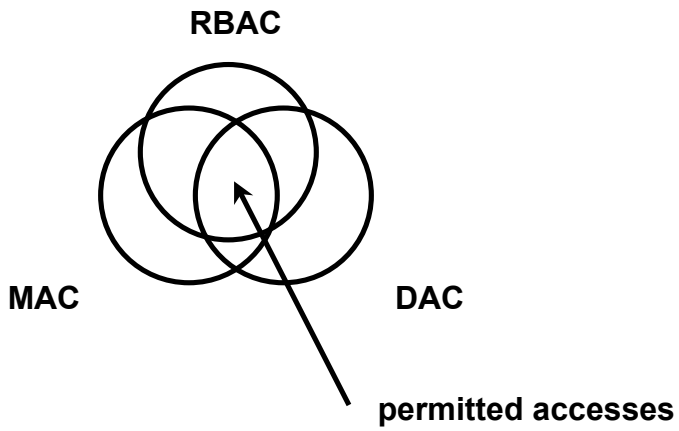
RBAC



RBAC: WHAT'S NEW

- Extend system support into application domain
- Use RBAC to manage RBAC

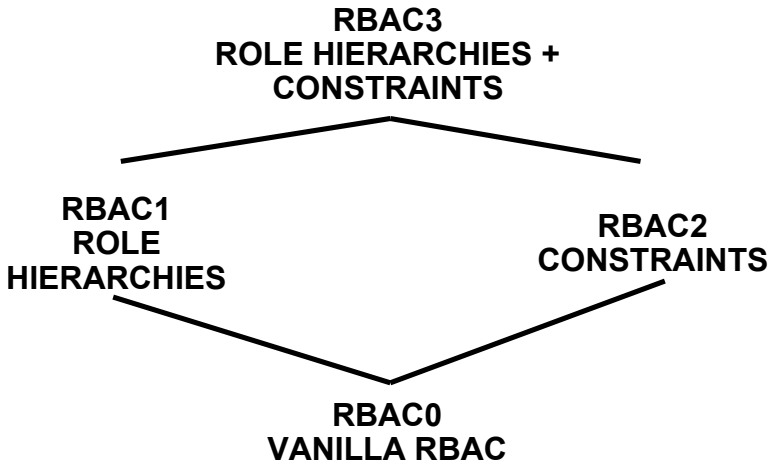
INTERACTION OF RBAC, MAC AND DAC



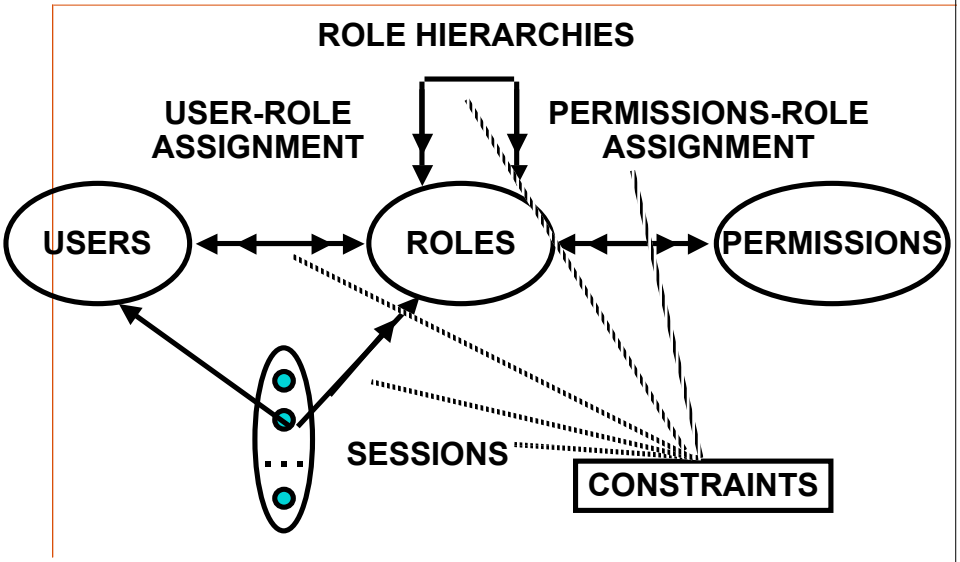
RBAC MODELS

- RBAC96 Family
  - RBAC0: Basic Model
  - RBAC1: Hierarchical Roles
  - RBAC2: Constrained Roles
  - RBAC3: RBAC1 + RBAC2

RBAC96 FAMILY



RBAC3



USERS

- Users are
  - human beings or
  - other active agents
- Each individual should be known as exactly one user

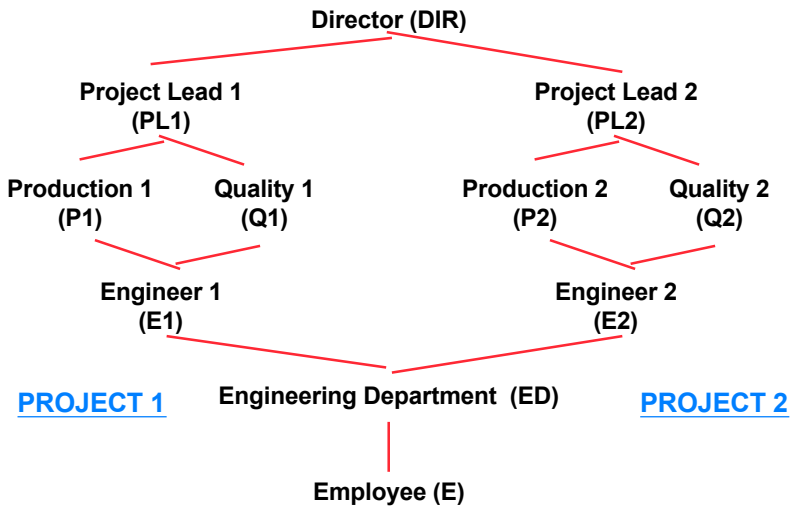
ROLES AS POLICY

- A role brings together
  - a collection of users and
  - a collection of permissions
- These collections will vary over time
  - A role has significance and meaning beyond the particular users and permissions brought together at any moment

ROLES VERSUS GROUPS

- Groups are often defined as
  - a collection of users
- A role is
  - a collection of users and
  - a collection of permissions
- Some authors define role as
  - a collection of permissions

HIERARCHICAL ROLES



PERMISSIONS

- Primitive permissions
  - read, write, append, execute
- Abstract permissions
  - credit, debit, inquiry

PERMISSIONS

- System permissions
  - auditorObject permissions
  - read, write, append, execute, credit, debit, inquiry

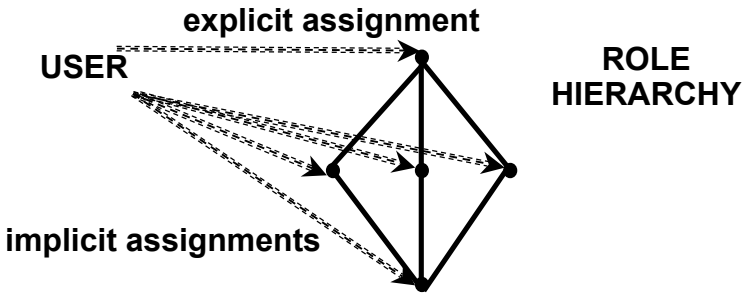
PERMISSIONS

- Permissions are positive
- No negative permissions or denials
  - negative permissions and denials can be handled by constraints
- No duties or obligations
  - outside scope of access control

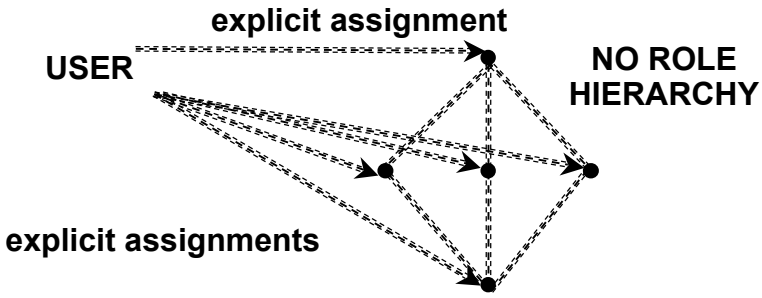
USER-ROLE ASSIGNMENT

- A user can be a member of many roles
- Each role can have many users as members

IMPLICIT USER ASSIGNMENT



EXPLICIT USER ASSIGNMENT



PERMISSION-ROLE ASSIGNMENT

- A permission can be assigned to many roles
- Each role can have many permissions

SESSIONS

- A user can invoke multiple sessions
- In each session a user can invoke any subset of roles that the user is a member of

CONSTRAINTS

- Applied to all components in RBAC
- Example : Mutually Exclusive Roles
  - Static Exclusion: The same individual can never hold both roles
  - Dynamic Exclusion: The same individual can never hold both roles in the same context

Exercise 1

- Consider an on-line grading system.
  - TAs can view (V) and add (A) everybody's grade;
  - Instructors can add (A), update (U), and view (V) everybody's grade.

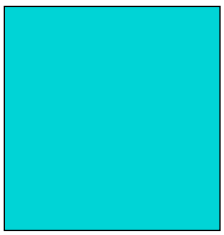
Exercise 1 (Cont'd)

- Assume a generic framework of RBAC0.
  - Customize it for the following class:
    - Instructor: Peng Ning (PN);
    - TA: Kun Sun (KS)
    - Students: John Smith (JS), Jane Davis (JD), Bret Moore (BM)
  - Users = {____, ____, ____, ____, ____}
  - Roles = {_____, _____}
  - Permissions = {_____, _____, _____}
  - PermissionAssignments = {(___, ___), (___, ___), (___, ___), (___, ___), (___, ___)}
  - UserAssignments = {(____, ____), (____, ____)}
  - Sessions = {____, ____} (Assume two sessions. No unique solution.)
  - Users (S→U): ____→ ____, ____→____
  - Roles (S→2^R): ____→{____}, ____→{____}

Exercise 1 (Cont'd)

- How about RBAC1?
  - Customize it for the following class:
    - Instructor: Peng Ning (PN);
    - TA: Kun Sun (KS)
    - Students: John Smith (JS), Jane Davis (JD), Bret Moore (BM)
  - What can be changed?

Role hierarchy:



Exercise 2

- Consider an on-line grading system.
  - Each student can only view his/her own grade;
  - TAs can view and add everybody's grade;
  - Instructors can add, update (*i.e.*, modify), and view everybody's grade.

Exercise 2 (Cont'd)

- Assume a generic framework of RBAC0.
  - Customize it for the following class:
    - Instructor: Peng Ning (PN)
    - TA: Kun Sun (KS)
    - Students: John Smith (JS), Jane Davis (JD), Bret Moore (BM)
  - Users = {____, ____, ____, ____, ____}
  - Roles = ?
  - Permissions = ?
  - PermissionAssignments = ?
  - UserAssignments = ?
  - …
- What are the difficulties?

Exercise 2 (Cont'd)

- How about RBAC2?
  - Customize it for the following class:
    - Instructor: Peng Ning (PN)
    - TA: Kun Sun (KS)
    - Students: John Smith (JS), Jane Davis (JD), Bret Moore (BM)
  - Assume there exists a global variable user-name that stores the name of the activated user.
  - Users = {____, ____, ____, ____, ____}
  - Roles = {_____, _____, _____}
  - Permissions = {V(student-name), _____, _____}
  - PermissionAssignments = {(___, ___), (___, ___), (___, ___), (___, ___) , (___, ___) , (___, ___)}
  - PA constraint: _____
  - UserAssignments = { }
  - …