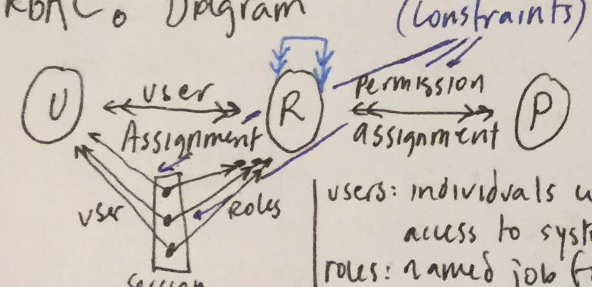
CS370 Notes
Week 6

Role Based Access Control (RBAC)
- roles are defined by a set of permissions
- Each user is assigned ≥1 role
- Rbac is useful: User-based changes often, whereas roles' needs are relatively static
- Roles: collection of permissions
- group: Collection of users

Diff between RBAC And DAC
- Rbac is organization perspective
- DAC is wrt users /user persp.
- DAC allows a user to grant permissions ~~they~~ on objects they own
- RBAC does not ↑

RBAC Models
- $RBAC_0$: Basic, minimal functionality
- $RBAC_1$: $RBAC_0$ + Role Hierarchies
- $RBAC_2$: $RBAC_0$ + Constraints
- $RBAC_3$: ≤ $RBAC_{0,1,2}$

$RBAC_0$ Diagram

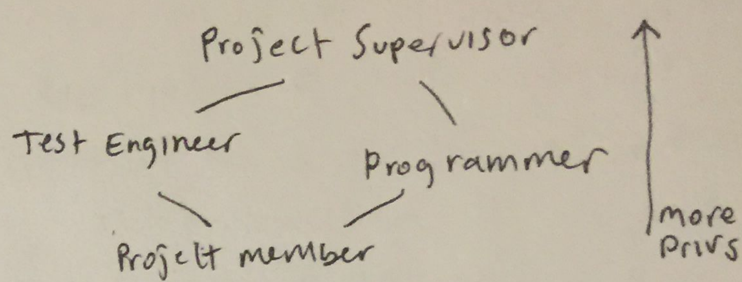


users: individuals with access to system
roles: named job fxns
perm: approval to perform an operat. on an obj.

session: maps user to role(s)

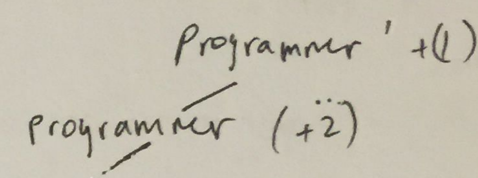
Role Hierarchies $RBAC_1$
- help reduce permission management more

Example Role Hierarchy



Line indicates inheritance

- Private role: Inherits from 1...



Constraints ($RBAC_2$) + Principles
- Can establish mutually exclusive roles
- Cardinality can be set to limit number of people assigned to a role
- Prerequisite constraints
  - ↳ Can help support least privilege
- Static Separation of Duty $\boxed{SSD}$
  - ↳ prevent conflict of interest
    - ↳↳ put Cardinality /other rules.

  SSD := (rs, n) | no user is assigned to n or more roles from their role set (rs).
- Dynamic separation of duty DSD
  - ↳ DSD := (rs, n) (n ≥ 2) | no user session may activate ≥ n roles from rs

Role Engineering
- designing an RBAC Role set
- Top Down / Bottom Up

- Top Down: use business rules/ procs. to understand the job functions, then accesses or things done by the job function, then permissions used
- Bottom Up: look at current Access Control. then use ML to discover roles. Good because it is Quick, bad because the ML miner is not perfect.

BLP Model of MAC

- MAC vs DAC
  - → DAC: users can change Access control state
  - → MAC: Access Descisions cannot be changed and enforced by a system-wide set of rules. "user's cannot trusted"
  - → MAC: designed to preserve confidentiallity
- BLP model
  - Subjects and objects are all associated with a security level, i.e. "top secret"
  - A "subject's level" is their security clearance
  - An "object's level" is its classification
  - $\underbrace{\text{NO READ UP}}_{\text{simple sec. prop.}}$; $\underbrace{\text{NO WRITE DOWN}}_{\text{* property}}$
  - DAC IN MAC: ds-property: Also check DAC policy after MAC

Advanced Security Classes

- apart from security level (i.e. "top secret"), there are categories that are included. and together these are a security label. helps least priv.
  - ↳ sets of security labels (some) cant be compared to others.

↳ (A1, C1) Dominates (A2, C2) iff: A2 <= A1 and $C2 \subseteq C1$

↳ SSP: subject s can (read only) object o if label(s) Dominates label(o)

↳ subject s can append into object o if label(o) dominates label(s)

Security Clearance flexibility

- define a max, and current level for subjects.
- AKA Tranquility
  - ↳ strong tranquility: immutable clearances and classifications
  - ↳ Weak Tranquility (used) can change wrt policy

BIBA Integrity Model

- MAC. Differs from BLP as BLP focuses on confidentiality, BIBA focuses on integrity
- Works on integrity levels
- Higher level, higher confidence the data is accurate / program will execute properly
- strict Integrity Property
  - $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$: NO WRITE UP
- Integrity confinement property
  - $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
  - ↳ NO READ DOWN
- Invocation Property
  - $s_1 \in S$ can invoke $s_2 \in S$ iff $i(s_2) \leq i(s_1)$

Integrity Labels
↳ i.e. Trusted : {proj₁, proj₂} = trusted only for projects (1,2)
↳ form partial order or Lattice

- Comparing integrity labels
(A1, C1) dom (A2, C2) iff $A2 <= A1$ and $C2 \subseteq C1$
↳ s can write to o if label (s) dom label (o). ← SIP
↳ ICP: Read requires label (o) dom Label (s)
↳ IP: invoke: Label(s_1) dom Label (s_2)

Chinese Wall Security Model
- addresses conflict of interest
- Broken Down into:
→ ~~Object~~ Subject
→ ~~Information~~ Information
↳ Objects
↳ Dataset (DS) = all objects that concern one corporation
↳ Conflict of Interest Class (CI): all datasets whose corporations are in competition
→ Access Rules

- Rules:
→ s can read o if: O is in the same DS as something already accessed by s
→ O belongs to CI from which s has not yet accessed any info.
→ s can write to O if s can read O
→ All objects s can read are in the same DS as O.