Security Perspective of Policy

William H. Winsborough
Center for Secure Information Systems
George Mason University

Issue

- "Are authorization policies of the form *permit/deny <action> when <condition>* sufficient?"
- No. At a minimum, need delegation, too
- Leads to the following tension:
  - Portions of policy are not under local control, and may evolve independently
  - Security view: policy should be vetted before "going live"

Issue

- "If goals are policies, can we distinguish between policies and requirements?"
- Yes.
  - Policy is *intentional*
    - It gives rules characterizing the author's intentions for system behavior
  - Requirements need not be
    - *E.g.*, they may talk about the net effect of policies, without concern for the reasons

Requirements Provide a Basis for Vetting Policy

- Vetting policy
  - Unpleasant surprises can be avoided by providing as many views of a policy as possible
  - Requirements can be thought of as sanity checks
- Requirements are not enforced directly (as policy is)

Authorization-Policy Analysis: An Alternative View

- Authors of policy statements need assistance in understanding global impact of delegations & revocations
- ["Beyond Proof of Compliance: Safety and Availability Analysis in Trust Management," Li, Winsborough, and Mitchell, *Oakland 2003*]

Example Analysis Problems

- "Can Alice ever get access to the database?"
  - Simple Safety -- Existential
- "Will Bob always have access to the database?"
  - Simple Availability -- Universal
- "Can anyone besides you and me ever get access?"
  - Bounded Safety -- Universal
- "Will there always be somebody that has access?"
  - Liveness -- Existential
- "Can anyone ever be both a buyer and an accountant?"
  - Mutual Exclusion -- Universal
- "Will all managers always have access?"
  - Containment: Availability -- Universal
- "Can anyone who is not an employee ever get access?"
  - Containment: Safety -- Universal

Policy Language and Restriction Rule

- $\mathcal{P}$ is an *RT* policy
- $\mathcal{R}$ gives two sets of roles, $\mathcal{G}$ and $\mathcal{S}$
  - Growth restriction: additional statements defining roles in $\mathcal{G}$ cannot be added to state
  - Shrink restriction: statements defining roles in $\mathcal{S}$ cannot be removed from state

Example P and R

- SA.access ← HR.manager
- SA.access ← HR.manager.access ∩ HR.employee
- HR.employee ← HR.manager
- HR.employee ← HR.programmer
- HR.manager ← Alice
- HR.programmer ← Bob
- HR.programmer ← Carl
- Alice.access ← Bob
- $\mathcal{G}$ = { SA.access, HR.employee }
- $\mathcal{S}$ = { SA.access, HR.employee, HR.manager }

Example Problem Instance (1 of 4)

- SA.access ← HR.manager
- SA.access ← HR.manager.access ∩ HR.employee
- HR.employee ← HR.manager
- HR.employee ← HR.programmer
- HR.manager ← Alice
- HR.programmer ← Bob
- HR.programmer ← Carl
- Alice.access ← Bob
- $\mathcal{G}$ = { SA.access, HR.employee }
- $\mathcal{S}$ = { SA.access, HR.employee, HR.manager }
- Simple safety: Is SA.access ⊒ { Eve } possible? (Yes)

Example Problem Instance (2 of 4)

- SA.access ← HR.manager
- SA.access ← HR.manager.access ∩ HR.employee
- HR.employee ← HR.manager
- HR.employee ← HR.programmer
- HR.manager ← Alice
- HR.programmer ← Bob
- HR.programmer ← Carl
- Alice.access ← Bob
- $\mathcal{G}$ = { SA.access, HR.employee }
- $\mathcal{S}$ = { SA.access, HR.employee, HR.manager }
- Simple availability: Is SA.access ⊒ { Alice } necessary? (Yes)

Example Problem Instance (3 of 4)

- SA.access ← HR.manager
- SA.access ← HR.manager.access ∩ HR.employee
- HR.employee ← HR.manager
- HR.employee ← HR.programmer
- HR.manager ← Alice
- HR.programmer ← Bob
- HR.programmer ← Carl
- Alice.access ← Bob
- $\mathcal{G}$ = { SA.access, HR.employee }
- $\mathcal{S}$ = { SA.access, HR.employee, HR.manager }
- Bounded safety: Is { Alice, Bob } ⊒ SA.access necessary? (No)

Example Problem Instance (4 of 4)

- SA.access ← HR.manager
- SA.access ← HR.manager.access ∩ HR.employee
- HR.employee ← HR.manager
- HR.employee ← HR.programmer
- HR.manager ← Alice
- HR.programmer ← Bob
- HR.programmer ← Carl
- Alice.access ← Bob
- $\mathcal{G}$ = { SA.access, HR.employee }
- $\mathcal{S}$ = { SA.access, HR.employee, HR.manager }
- Containment: Is HR.employee ⊒ SA.access necessary? (Yes)

Analysis Use Cases

- Security requirement = analysis problem instance + acceptable answer
  - Organization defines a set of requirements
- Sanity check
  - Some principals are assumed to be trustworthy
  - They analyze proposed policy changes with respect organization's requirements before committing
- Insider threat assessment
  - Can vary the principals that are assumed trustworthy
  - In this way, organization can determine how it is exposed

Issue

- "Do different policy models require different analysis techniques?"
- Yes.
- For example, policy management of systems that have state
  - Requirements like: inconsistent states do not arise
  - ["Using event analysis to formalize policy specification and analysis," Bandara, Lupu, Russo, *Policy 2003*]