ECS 235B Module 14
Security Policy Languages
High-Level Policy Languages

• Constraints expressed independent of enforcement mechanism
• Constraints restrict entities, actions
• Constraints expressed unambiguously
  • Requires a precise language, usually a mathematical, logical, or programming-like language
Example: Ponder

- Security and management policy specification language
- Handles many types of policies
  - Authorization policies
  - Delegation policies
  - Information filtering policies
  - Obligation policies
  - Refrain policies
Entities

• Organized into hierarchical domains

• Network administrators
  • Domain is /NetAdmins
  • Subdomain for net admin trainees is
  • /NetAdmins/Trainees

• Routers in LAN
  • Domain is /localnet
  • Subdomain that is a testbed for routers is
  • /localnet/testbed/routers
Authorization Policies

• Allowed actions: netadmins can enable, disable, reconfigure, view configuration of routers

```plaintext
inst auth+ switchAdmin {
  subject /NetAdmins;
  target /localnetwork/routers;
  action enable(), disable(), reconfig(), dumpconfig();
}
```
Authorization Policies

• Disallowed actions: trainees cannot test performance between 8AM and 5PM

inst auth- testOps {
  subject /NetEngineers/trainees;
  target /localnetwork/routers;
  action testperformance();
  when Time.between("0800", "1700");
}
Delegation Policies

• Delegated rights: net admins delegate to net engineers the right to enable, disable, reconfigure routers on the router testbed

```plaintext
inst deleg+ (switchAdmin) delegSwitchAdmin {
  grantee /NetEngineers;
  target /localnetwork/testNetwork/routers;
  action enable(), disable(), reconfig();
  valid Time.duration(8);
}
```
Information Filtering Policies

• Control information flow: net admins can dump everything from routers between 8PM and 5AM, and config info anytime

```plaintext
inst auth+ switchOpsFilter {
  subject  /NetAdmins;
  target   /localnetwork/routers;
  action   dumpconfig(what)
            { in partial = "config"; }
            if (Time.between("2000", "0500")){
              in partial = "all"; }
}
```
Refrain Policies

• Like authorization denial policies, but enforced by the subjects: net engineers cannot send test results to net developers while testing in progress

```plaintext
inst refrain testSwitchOps {
    subject s=/NetEngineers;
    target /NetDevelopers;
    action sendTestResults();
    when s.teststate="in progress"
}
```
Obligation Policies

• Must take actions when events occur: on 3\textsuperscript{rd} login failure, net security admins will disable account and log event

\textbf{inst oblig} loginFailure \{ \\
\textbf{on} \quad \text{loginfail}(\text{userid}, 3); \\
\textbf{subject} \quad s=/\text{NetAdmins}/\text{SecAdmins}; \\
\textbf{target} \quad t=/\text{NetAdmins}/users \ ^\wedge (\text{userid}); \\
\textbf{do} \quad t.\text{disable}() \rightarrow s.\text{log}(\text{userid}); \\
\}

Example

- Policy: separation of duty requires 2 different members of Accounting approve check

```plaintext
inst auth+ separationOfDuty {
  subject  s=/*Accountants;*
  target   t=checks;
  action   approve(), issue();
  when     s.id <> t.issuerid;
}
```
Low-Level Policy Languages

- Set of inputs or arguments to commands
  - Check or set constraints on system
- Low level of abstraction
  - Need details of system, commands
Example: X Window System

- UNIX X11 Windowing System
- Access to X11 display controlled by list
  - List says what hosts allowed, disallowed access
    - `xhost +groucho -chico`
- Connections from host groucho allowed
- Connections from host chico not allowed
Quiz

tripwire is a file system scanner that reports changes to file systems and file attributes. There is a configuration file that describes what might change. To initialize, tripwire records those values in a database. On future runs, it compares the current values to the ones in the database and reports differences. The database entry has the following fields:

- file name, version, bitmask for attributes, mode, inode number, number of links, UID, GID, size, times of creation, last modification, last access, cryptographic checksums

Is the language describing the file attributes to check a high level policy language or a low level policy language?